Algorithmes Efficaces en Calcul Formel
Notes du cours 2-22 du MPRI
Version du 20 novembre 2015

Alin Bostan

Frédéric Chyzak

Marc Giusti

Romain Lebreton

Grégoire Lecerf

Bruno Salvy

Éric Schost

Table des matières

Avant-propos

Le calcul formel traite des objets mathématiques exacts. Ce cours « Algorithmes efficaces en calcul formel » explore deux directions : la calculabilité et la complexité. La calculabilité étudie les classes d'objets mathématiques sur lesquelles des réponses peuvent être obtenues algorithmiquement. La complexité donne ensuite des outils pour comparer des algorithmes du point de vue de leur efficacité.

Les trois premières parties de ce cours passent en revue l'algorithmique efficace sur les objets fondamentaux que sont les entiers, les polynômes, les matrices, les séries et les solutions d'équations différentielles ou de récurrences linéaires. On y montre que de nombreuses questions portant sur ces objets admettent une réponse en complexité (quasi-)optimale, en insistant sur les principes généraux de conception d'algorithmes efficaces.

La quatrième partie est dédiée à la factorisation des polynômes. Il s'agit d'un des premiers problèmes d'une importance majeure en calcul formel, pour lequel il existe encore plusieurs problèmes ouverts conséquents. Nous y présentons les algorithmes classiques de factorisation séparable, sans carré et irréductible. Nous traitons tout d'abord le cas des polynômes à une variable dont les coefficients sont dans un corps finis puis des nombres rationnels. Le cas des polynômes à plusieurs variables vient ensuite et nous nous concentrons essentiellement sur le cas de deux variables pour les résultats de complexité.

La cinquième partie aborde les systèmes polynomiaux. Il s'agit de montrer comment répondre à des questions de nature géométrique posées sur les solutions de ces systèmes, tant par des approches à base de réécriture (bases standard ou de Gröbner) que plus directement liées à la structure de l'ensemble décrit (résolution géométrique).

La sixième et dernière partie du cours traite de l'intégration et de la sommation définie. De nombreux calculs d'intégrales et de sommes de fonctions ou de suites spéciales de la combinatoire ou de la physique mathématique peuvent être abordés algorithmiquement. Pour résoudre ces problèmes efficacement, presque tous les algorithmes abordés au cours des cinq premières parties sont utilisés.

Quelques thèmes classiques du calcul formel ne sont pas traités dans ce cours, comme la recherche de primitives (algorithme de Risch) ou de solutions liouvilliennes d'équations différentielles. Nous préférons en effet privilégier un point de vue où les équations sont vues comme des opérateurs.

Ces notes de cours sont une synthèse de matériaux rédigés lors de formations dédiées principalement à la recherche. Elles ne prétendent pas constituer un ouvrage de référence pour le calcul formel. Nous avons au contraire souvent choisi de mettre l'accent sur la présentation des idées clefs, quitte à présenter sous forme d'exercices les aspects les plus techniques, et à renvoyer le lecteur sur des ouvrages ou articles de recherche qui contiennent les solutions. Nous détaillons un grand nombre de résultats très récents qui conduisent souvent le lecteur à la frontière des connaissances.

Les prérequis mathématiques nécessaires à la lecture sont modestes et de nombreuses définitions classiques sont rappelées et illustrées au fur et à mesure des besoins. Le texte est par conséquent accessible tant à des étudiants en mathématiques qu'en informatique.

Nous espérons que cet ouvrage puisse être utile à tous les enseignants et chercheurs en mathématiques effectives, ou plus spécifiquement en calcul formel, désireux d'approfondir leurs connaissances ou à la recherche de matériaux pédagogiques.

Récemment l'essentiel de ces notes a été utilisé pendant plusieurs années pour un cours de deuxième année du *Master Parisien de Recherche en Informatique* de l'Université Paris Diderot, l'École Normale Supérieure de Cachan, l'École Normale Supérieure de Paris, et l'École polytechnique.

La partie concernant les systèmes polynomiaux provient aussi de notes de cours donnés au DEA *Méthodes algébriques* puis au *Master Algèbre et Géométrie* de l'Université Pierre-et-Marie-Curie, mais aussi au DEA *Informatique Mathématique et Applications* de l'École Normale Supérieure de Paris, l'École polytechnique, l'Université Pierre-et-Marie-Curie, l'Université Paris Diderot, et l'Université Paris-Sud.

Plusieurs parties ont aussi fait l'objet de mini-cours plus spécialisés donnés à l'occasion des *Journées Nationales de Calcul Formel* en 2007, 2010, 2011, et 2013.

CHAPITRE 1

Calcul formel et complexité

Résumé

Le calcul formel étudie dans quelle mesure un ordinateur peut "faire des mathématiques", et pour celles qu'il peut faire, à quelle vitesse. Ce chapitre introductif présente d'abord les questions de calculabilité. Le point de départ est le théorème plutôt négatif de Richardson-Matiyasevich. Ensuite, un survol rapide des questions qui seront abordées tout au long du cours montre que malgré ce théorème, bien des questions peuvent être traitées algorithmiquement. La seconde partie du chapitre aborde les questions de complexité, pose les conventions et les définitions principales, ainsi que les résultats de base sur le paradigme diviser-pour-régner.

1. Décider, calculer

1.1. Fondements logiques. D'une certaine manière, le calcul formel est fondé sur une contrainte d'origine logique.

THÉORÈME 1 (Richardson-Matiyasevich). *Dans la classe des expressions formées à partir d'une variable X et de la constante 1 par les opérations d'anneau $+, -, \times$ et la composition avec les fonctions* sin *et la valeur absolue $|\cdot|$, le test d'équivalence à 0 est indécidable.*

Autrement dit, il n'existe pas d'algorithme permettant pour toute expression de cette classe de déterminer en temps fini si elle vaut 0 ou non. Plus généralement tout test d'égalité peut bien entendu se ramener à tester l'égalité à zéro dès que la soustraction existe. Cette limitation de nature théorique explique la difficulté et parfois la frustration que rencontrent les utilisateurs débutants des systèmes de calcul formel face à des fonctions de « simplification », qui ne peuvent être qu'heuristiques.

Pour effectuer un calcul, il est pourtant souvent crucial de déterminer si des expressions représentent 0 ou non, en particulier pour évaluer une fonction qui possède des singularités (comme la division). L'approche du calculateur formel expérimenté consiste à se ramener autant que faire se peut à des opérations d'un domaine dans lequel le test à zéro est décidable. Le calcul formel repose ainsi de manière naturelle sur des constructions algébriques qui préservent la décidabilité du test à 0. En particulier, les opérations courantes sur les vecteurs, matrices, polynômes, fractions rationnelles, ne nécessitent pas d'autre test à 0 que celui des coefficients. La notion d'effectivité permet de préciser ce point de vue.

DEFINITION 1. Une structure algébrique (groupe, anneau, corps, espace vectoriel,...) est dite *effective* si l'on dispose :

- d'une structure de données pour en représenter les éléments ;
- d'algorithmes pour en effectuer les opérations et pour y tester l'égalité et autres prédicats.

Par exemple, dans un anneau effectif, outre l'égalité, les opérations requises sont l'addition, la soustraction et la multiplication. Pour une structure ordonnée, des algorithmes sont aussi requis pour effectuer les comparaisons. Cette notion d'effectivité est bien évidemment attachée à un modèle de calcul. Dans ce cours nous considérerons principalement les *machines à accès direct* (voir ci-dessous). Néanmoins il est connu que les fonctions calculables par ces machines sont les mêmes que pour les machines de Turing.

1.2. Structures et constructions de base. Les objets les plus fondamentaux sont assez faciles à représenter en machine de manière exacte. Nous considérons tour à tour les plus importants d'entre eux, en commençant par les plus élémentaires. Ils s'assemblent ensuite à l'aide de tableaux ou de listes pour en former de plus complexes.

Entiers machine. Les entiers fournis par les processeurs sont des entiers modulo une puissance de 2 (le nombre de bits d'un mot machine, typiquement 32 ou 64). Ils sont appelés des *entiers machine*. Les opérations rendues disponibles par le processeur sont l'addition, la soustraction, la multiplication et parfois la division. La norme ANSI du langage C fournit au programmeur la division et le modulo pour ces entiers, c'est-à-dire que le compilateur implante ces opérations si le processeur ne le fait pas.

Entiers. Pour manipuler des entiers dont la taille dépasse celle d'un mot machine, il est commode de les considérer comme écrits dans une base B assez grande :

$$N = a_0 + a_1 B + \cdots + a_k B^k.$$

L'écriture est unique si l'on impose $0 \leq a_i < B$. (Le signe est stocké séparément.) Ces nombres peuvent être stockés dans des tableaux d'entiers machine. Les objets obtenus sont des entiers de taille arbitraire appelés parfois *bignums*.

L'addition et le produit peuvent alors être réduits à des opérations sur des entiers inférieurs à B^2, au prix de quelques opérations de propagation de retenue. Le choix de B dépend un peu du processeur. Si le processeur dispose d'une instruction effectuant le produit de deux entiers de taille égale à celle d'un mot machine, renvoyant le résultat dans deux mots machines, alors B pourra être pris aussi grand que le plus grand entier tenant dans un mot machine. Sinon, c'est la racine carrée de ce nombre qui sera utilisée pour B. Quoiqu'il en soit, nous avons donc.

LEMME 1. *L'anneau $\mathbb{Z}$ des entiers relatifs est effectif.*

Entiers modulaires. Les calculs avec des polynômes, des fractions rationnelles ou des matrices à coefficients entiers souffrent souvent d'une maladie propre au calcul formel : la croissance des expressions intermédiaires. Les entiers produits comme coefficients des expressions intervenant lors du calcul sont de taille disproportionnée par rapport à ceux qui figurent dans l'entrée et dans la sortie.

EXEMPLE 1. Voici le déroulement typique du calcul du plus grand diviseur commun (pgcd) de deux polynômes à coefficients entiers par l'algorithme d'Euclide

(Chapitre 6) :

$$P_0 = 7X^5 - 22X^4 + 55X^3 + 94X^2 - 87X + 56,$$
$$P_1 = 62X^4 - 97X^3 + 73X^2 + 4X + 83,$$
$$P_2 = \mathrm{rem}(P_0, P_1) = \frac{113293}{3844}X^3 + \frac{409605}{3844}X^2 - \frac{183855}{1922}X + \frac{272119}{3844},$$
$$P_3 = \mathrm{rem}(P_1, P_2) = \frac{18423282923092}{12835303849}X^2 - \frac{15239170790368}{12835303849}X + \frac{10966361258256}{12835303849},$$
$$P_4 = \mathrm{rem}(P_2, P_3) = -\frac{216132274653792395448637}{44148979404824831944178}X - \frac{631179956389122192280133}{882979588096496638883356},$$
$$P_5 = \mathrm{rem}(P_3, P_4) = \frac{20556791167692068695002336923491296504125}{3639427682941980248860941972667354081}.$$

Chaque étape calcule le reste (noté rem pour *remainder*) de la division euclidienne des deux polynômes précédents. Les coefficients de ces polynômes intermédiaires font intervenir des entiers qui croissent de manière exponentielle, alors que le résultat recherché est 1.

Les entiers modulaires remédient à ce problème de deux manières. D'une part, pour un calcul de décision, de dimension, ou de degré, l'exécution de l'algorithme sur la réduction de l'entrée modulo un nombre premier donne un algorithme *probabiliste* répondant à la question. Cette technique peut aussi servir de base à un algorithme *déterministe* lorsque les nombres premiers pour lesquels la réponse est fausse peuvent être maîtrisés. C'est le cas du pgcd : en évitant les premiers qui divisent les coefficients de tête des deux polynômes, le degré du pgcd modulaire est le même que le degré du pgcd exact.

D'autre part, les entiers modulaires sont utilisés dans les algorithmes reposant sur le théorème des restes chinois. Ce théorème indique qu'un entier inférieur au produit de nombres premiers $p_1 \cdots p_k$ peut être reconstruit à partir de ses réductions modulo $p_1, \ldots, p_k$. Lorsqu'une borne sur la taille du résultat est disponible, il suffit d'effectuer le calcul modulo suffisamment de nombres premiers (choisis en pratique assez grands pour que leur nombre soit faible et assez petits pour que les opérations tiennent dans un mot machine), pour ensuite reconstruire le résultat, court-circuitant de la sorte toute croissance intermédiaire.

Vecteurs et matrices. Une fois donnée une représentation exacte pour des coefficients, il est facile de construire des vecteurs ou matrices comme des tableaux, ou plus souvent comme des tableaux de pointeurs sur les coefficients. Les opérations de produit par un scalaire, de produit de matrices ou de produit d'une matrice par un vecteur se réduisent aux opérations d'addition et de multiplication sur les coefficients. Il en va de même pour la recherche de noyau ou d'inverse de matrices.

PROPOSITION 1. *Si $\mathbb{K}$ est un corps effectif, l'espace vectoriel $\mathbb{K}^n$ l'est aussi, ainsi que l'anneau $\mathcal{M}_n(\mathbb{K})$.*

Polynômes. Les polynômes peuvent être représentés de plusieurs manières, et la meilleure représentation dépend des opérations que l'on souhaite effectuer. Pour un polynôme en une variable, les choix principaux sont :

- la représentation dense : comme pour les entiers, le polynôme est représenté comme un tableau de (pointeurs sur les) coefficients ;
- la représentation creuse : le polynôme est représenté comme une liste de paires (coefficient, exposant) généralement triée par les exposants.

Dans les deux cas, nous avons clairement :

PROPOSITION 2. *Si $\mathbb{A}$ est un anneau effectif, alors $\mathbb{A}[X]$ l'est aussi.*

L'usage itéré de cette proposition fournit les polynômes à plusieurs variables.

Fractions rationnelles. Les rationnels peuvent être stockés comme des paires où numérateur et dénominateur sont des entiers de taille arbitraire. Les opérations d'addition et de multiplication se réduisent aux opérations analogues sur les entiers et le test d'égalité à zéro se réduit au test d'égalité à 0 sur le numérateur. De même, les fractions rationnelles sont représentées par des paires de polynômes. Les opérations d'addition, produit, division, se réduisent aux additions et multiplications sur les coefficients. Plus généralement, nous obtenons :

PROPOSITION 3. *Si $\mathbb{A}$ est un anneau intègre effectif, alors son corps des fractions est effectif.*

Séries tronquées. Les séries tronquées

$$\sum_{k=0}^{N} a_k X^k + O(X^{N+1})$$

se représentent pratiquement comme des polynômes. La différence principale apparaît lors du produit : les coefficients des termes d'exposant au moins $N+1$ n'ont pas besoin d'être calculés, ni stockés. La structure considérée alors est l'anneau $A[X]/(X^{N+1})$ obtenu en faisant le quotient de l'anneau $A[X]$ par l'idéal qu'y engendre X^{N+1}. Les éléments de ce quotient sont représentés par le $N+1$-uplet de coefficients $(a_0, \ldots, a_N)$. Le test à 0 revient à tester que ces coefficients sont tous nuls.

PROPOSITION 4. *Si $\mathbb{A}$ est un anneau effectif et $N \in \mathbb{N}$, alors $\mathbb{A}[X]/(X^{N+1})$ est un anneau effectif.*

Cette structure de données joue un rôle très important non seulement pour des calculs d'approximations, mais aussi comme une représentation *exacte*. En voici trois exemples :

1. Une fraction rationnelle dont les numérateurs et dénominateurs ont degré borné par d peut être reconstruite à partir d'un développement en série à l'ordre $2d+1$. Cette représentation joue ainsi un rôle clé dans la manipulation des nombres algébriques (Chapitre 3), et dans le calcul efficace de la division euclidienne de polynômes, de suites récurrentes linéaires (comme le calcul rapide du $10\,000^{\text{e}}$ nombre de Fibonacci au Chapitre 4) et du polynôme minimal d'une matrice creuse (Chapitre 9).
2. Il est possible de reconstruire une équation différentielle linéaire à coefficients polynomiaux à partir du développement en série d'une solution et de bornes sur l'ordre et le degré des coefficients. De façon analogue, il est possible de reconstruire une récurrence linéaire à coefficients polynomiaux à partir des premières valeurs d'une de ses solutions. L'outil algorithmique pour effectuer ces calculs de *devinette* (*guessing* en anglais) est le calcul rapide d'approximants de Padé-Hermite (Chapitre 7).
3. Un polynôme en deux variables peut être reconstruit à partir du développement en série d'une solution. L'efficacité de la factorisation des polynômes à deux variables, ou encore de la résolution de systèmes polynomiaux par la méthode dite de *résolution géométrique*, abordée au Chapitre 26 repose de manière cruciale sur cette opération, qui doit être effectuée rapidement.

1.3. Équations comme structures de données. Une fois construits les objets de base que sont les polynômes, les séries ou les matrices, il est possible d'aborder des objets mathématiques construits *implicitement*. Ainsi, il est bien connu qu'il n'est pas possible de représenter toutes les solutions de polynômes de haut degré par radicaux, mais de nombreuses opérations sur ces solutions sont aisées en prenant le polynôme lui-même comme structure de données. Ce point de vue permet d'étendre le domaine d'application du calcul formel pourvu que des algorithmes soient disponibles pour effectuer les opérations souhaitées (typiquement addition, multiplication, multiplication par un scalaire, test d'égalité) par manipulation des équations elles-mêmes.

Nombres algébriques. C'est ainsi que l'on nomme les solutions de polynômes à une variable. Le résultat est spectaculaire.

PROPOSITION 5. *Si $\mathbb{K}$ est un corps effectif, alors sa clôture algébrique $\overline{\mathbb{K}}$ l'est aussi.*

Les opérations d'addition et de multiplication peuvent être effectuées à l'aide de résultants (Chapitre 6). Ceux-ci peuvent être calculés efficacement à l'aide de séries (Chapitre 3). La division s'obtient par l'algorithme d'Euclide sur les polynômes (Chapitre 6), et le test à zéro se déduit du pgcd. Par exemple, il est possible de prouver assez facilement une identité comme

$$\frac{\sin\frac{2\pi}{7}}{\sin^2\frac{3\pi}{7}} - \frac{\sin\frac{\pi}{7}}{\sin^2\frac{2\pi}{7}} + \frac{\sin\frac{3\pi}{7}}{\sin^2\frac{\pi}{7}} = 2\sqrt{7} \tag{1}$$

une fois que l'on reconnaît qu'il s'agit d'une égalité entre nombres algébriques.

Systèmes polynomiaux. Vu l'importance des systèmes polynomiaux, une grande partie du cours leur sera consacrée. Un résultat important de cette partie est le suivant.

PROPOSITION 6. *Si $\mathbb{K}$ est un corps effectif, $f_1, \ldots, f_p$ des polynômes dans $\mathbb{K}[X_1, \ldots, X_n]$ et $\mathcal{I}$ l'idéal qu'ils engendrent, alors le quotient $\mathbb{K}[X_1, \ldots, X_p]/\mathcal{I}$ est un anneau effectif.*

Ce quotient est un outil de base pour répondre à de nombreuses questions naturelles sur un système de polynômes, comme l'existence de solutions, la dimension de l'espace des solutions (qui indique s'il s'agit d'une surface, d'une courbe, ou de points isolés), le degré, ou le calcul d'une paramétrisation de l'ensemble des solutions.

Il est également possible d'éliminer une ou des variables entre des polynômes. Cette opération s'interprète géométriquement comme une projection (Chapitre 23). Dans le cas le plus simple, elle permet de calculer un polynôme s'annulant sur les abscisses des intersections de deux courbes (Chapitre 6). Une autre application est l'implicitisation, qui permet par exemple de calculer une équation pour une courbe donnée sous forme paramétrée.

Équations différentielles linéaires. Cette structure de données permet de représenter de nombreuses fonctions usuelles (exponentielle, fonctions trigonométriques et trigonométriques hyperboliques, leurs réciproques), ainsi que de nombreuses fonctions spéciales de la physique mathématique (fonctions de Bessel, de Struve, d'Anger, ..., fonctions hypergéométriques et hypergéométriques généralisées), ainsi bien sûr que de multiples fonctions auxquelles n'est pas attaché un nom

classique. Les opérations d'addition et de produit sont effectuées par des variantes noncommutatives du résultant qui se ramènent à de l'algèbre linéaire élémentaire (Chapitre 14). Le test à zéro se ramène à tester l'égalité d'un nombre fini de conditions initiales. Une partie de ces résultats se résume ainsi :

PROPOSITION 7. *Si $\mathbb{K}$ est un corps effectif, les séries formelles de $\mathbb{K}[[X]]$ qui sont solutions d'équations différentielles linéaires à coefficients dans $\mathbb{K}[X]$ forment un anneau effectif.*

En d'autres termes, des structures de données finies permettent de manipuler ces objets infinis et d'en tester l'égalité ou la nullité.

Ainsi, des identités élémentaires comme $\sin^2 X + \cos^2 X = 1$ sont non seulement facilement prouvables algorithmiquement, mais elles sont également calculables, c'est-à-dire que le membre droit se calcule à partir du membre gauche.

Les relations étroites entre équations différentielles linéaires et récurrences linéaires — les séries solutions des unes ont pour coefficients les solutions des autres — amènent aux mêmes réponses algorithmiques à des questions sur des suites. Par exemple, l'identité de Cassini sur les nombres de Fibonacci

$$F_{n+2}F_n - F_{n+1}^2 = (-1)^{n+1}, \qquad n \geq 0$$

est exactement du même niveau de difficulté que $\sin^2 X + \cos^2 X = 1$. Le pendant du résultat précédent est donc :

PROPOSITION 8. *Si $\mathbb{K}$ est un corps effectif, l'ensemble des suites de $\mathbb{K}^{\mathbb{N}}$ solutions de récurrences linéaires à coefficients dans $\mathbb{K}[n]$ forme un anneau effectif.*

Systèmes d'équations différentielles et de récurrences linéaires. Ces systèmes sont aux équations différentielles ou de récurrences ce que les systèmes polynomiaux sont aux polynômes en une variable. Les mêmes opérations sont disponibles. En particulier, l'élimination s'étend dans ce cadre en introduisant des algèbres d'opérateurs adaptés (Chapitre 29). Une application très importante, le *téléscopage créatif*, permet de calculer automatiquement des sommes et des intégrales définies. Ainsi,

$$\sum_{k=0}^{n}\left(\sum_{j=0}^{k}\binom{n}{j}\right)^3 = n2^{3n-1} + 2^{3n} - 3n2^{n-2}\binom{2n}{n},$$

$$\sum_{n=0}^{\infty} H_n(x)H_n(y)\frac{u^n}{n!} = \frac{\exp\left(\frac{4u(xy-u(x^2+y^2))}{1-4u^2}\right)}{\sqrt{1-4u^2}},$$

$$\frac{1}{2}J_0(x)^2 + J_1(x)^2 + J_2(x)^2 + \cdots = \frac{1}{2},$$

$$\int_{-1}^{+1} \frac{e^{-px}T_n(x)}{\sqrt{1-x^2}}\,dx = (-1)^n\pi I_n(p),$$

$$\int_0^{+\infty} xe^{-px^2}J_n(bx)I_n(cx)\,dx = \frac{1}{2p}\exp\left(\frac{c^2-b^2}{4p}\right)J_n\left(\frac{bc}{2p}\right),$$

$$\int_0^{+\infty} xJ_1(ax)I_1(ax)Y_0(x)K_0(x)\,dx = -\frac{\ln(1-a^4)}{2\pi a^2},$$

$$\sum_{k=0}^{n}\frac{q^{k^2}}{(q;q)_k(q;q)_{n-k}} = \sum_{k=-n}^{n}\frac{(-1)^k q^{(5k^2-k)/2}}{(q;q)_{n-k}(q;q)_{n+k}},$$

sont des formules qui mettent en jeu diverses fonctions spéciales ou polynômes orthogonaux classiques, et qui peuvent être prouvées automatiquement. Les algorithmes correspondants seront décrits au Chapitre 30. L'énoncé en termes d'anneaux effectif est un peu lourd et omis ici.

En conclusion, les exemples ci-dessus illustrent bien la manière dont le calcul formel parvient à effectuer de nombreux calculs utiles dans les applications malgré l'indécidabilité révélée par le théorème de Richardson et Matiyasevich.

2. Calculer rapidement

En pratique, la calculabilité n'indique que la faisabilité. Il faut disposer d'algorithmes efficaces et d'une bonne implantation pour pouvoir effectuer des calculs de grande taille. La première partie de ce cours est consacrée aux algorithmes efficaces sur les structures de base du calcul formel. L'efficacité sera mesurée par la théorie de la complexité et nous ferons ressortir des principes récurrents dans la conception d'algorithmes efficaces.

Exemple 2. Pour donner une idée de ce que veut dire rapidement, voici ce qui peut être calculé en *une seconde* avec le système Maple sur un ordinateur portable d'aujourd'hui [1], en notant $\mathbb{K}$ le corps $\mathbb{Z}/p\mathbb{Z}$ à p éléments, $p = 67\,108\,879$ étant un nombre premier de 26 bits (dont le carré tient sur un mot machine) :

1. Entiers :
   - produit de deux entiers avec 30 000 000 chiffres ;
   - factorielle de 1 300 000 (environ 7 000 000 chiffres) ;
   - factorisation d'un entier de 42 chiffres (produit de deux nombres premiers de taille la moitié).
2. Polynômes dans $\mathbb{K}[X]$:
   - produit de deux polynômes de degré 650 000 ;
   - pgcd et résultant de deux polynômes de degré 12 500 ;
   - factorisation d'un polynôme de degré 170 (produit comme ci-dessus).
3. Polynômes dans $\mathbb{K}[X, Y]$:
   - résultant de deux polynômes de degré total 20 (sortie de degré 400) ;
   - factorisation d'un polynôme de degré 160 en deux variables.
4. Matrices :
   - produit de deux matrices 850×850 à coefficients dans $\mathbb{K}$;
   - déterminant d'une matrice $1\,400 \times 1\,400$ à coefficients dans $\mathbb{K}$;
   - polynôme caractéristique d'une matrice 500×500 à coefficients dans $\mathbb{K}$;
   - déterminant d'une matrice 200×200 dont les coefficients sont des entiers 32 bits.

Ces exemples montrent qu'il est relativement aisé de calculer avec des objets de taille colossale. Dans la plupart de ces exemples, les algorithmes naïfs mettraient plusieurs années pour le même calcul. On voit aussi qu'en une seconde, les algorithmes sont déjà dans leur régime asymptotique et qu'une analyse asymptotique suffira à obtenir des informations précises. Tout ceci donne envie d'introduire une mesure de complexité des différents algorithmes permettant d'expliquer, voire de prédire, les différences entre les tailles atteintes pour ces questions.

1. Ce texte est écrit en 2015. Les ordres de grandeurs sont les mêmes sur différents ordinateurs et pour les différents systèmes de calcul formel principaux, avec des forces et des faiblesses différentes.

2.1. Mesures de complexité. Pour bien définir la complexité, il faut se donner : un modèle de machine ; les opérations disponibles sur cette machine ; leur coût unitaire. La complexité en espace mesure la mémoire utilisée par l'exécution de l'algorithme, et la complexité en temps, la somme des coûts unitaires des opérations effectuées par l'algorithme. Dans ce cours, nous n'abordons pas la complexité en espace.

Machine à accès direct. Le modèle que nous utiliserons est celui de la *machine à accès direct* (MAD), appelée aussi *Random Access Machine* (RAM) en anglais. Dans ce modèle, un programme lit et écrit des entiers sur deux bandes différentes et utilise un nombre arbitraire de registres entiers pour ses calculs intermédiaires. Les opérations élémentaires (l'assembleur de la machine) sont la lecture, l'écriture (sur bande ou en registre), l'addition, la soustraction, le produit, la division et trois instructions de saut : saut inconditionnel, saut si un registre est nul et saut si un registre est positif. Un point technique est que le programme ne fait pas partie des données, il n'est donc pas modifiable. Ce cours ne rentre jamais dans les détails de l'utilisation de ce modèle, qui ne sert qu'à fixer précisément les mesures utilisées.

Complexité arithmétique ou binaire. Nous considérerons deux mesures de complexité :

1. En calcul formel, de nombreux algorithmes peuvent être décrits comme opérant de façon abstraite sur une structure algébrique $\mathbb{A}$ effective. D'une façon concrète, la représentation des éléments de $\mathbb{A}$ importe peu et l'utilisation de chaque opération élémentaire de $\mathbb{A}$ (opération arithmétique ou prédicat) peut être parfaitement identifiée au cours de l'exécution de l'algorithme.

   La *complexité arithmétique* d'un tel algorithme est alors définie comme le nombre total d'opérations arithmétiques et de tests de prédicats effectués dans $\mathbb{A}$. Cette mesure ne prend volontairement pas en compte les opérations de copie, celles sur les compteurs de boucle, les indirections, etc. En pratique cette mesure reflète souvent le coût réel de l'algorithme si le coût des opérations dans $\mathbb{A}$ est prépondérant et que chaque opération requiert un temps essentiellement constant. C'est le cas par exemple si $\mathbb{A}$ est un corps fini. Ce n'est en revanche pas le cas si $\mathbb{A}$ est le corps des nombres rationnels $\mathbb{Q}$ et que la taille de ces nombres croît de façon significative durant les calculs.

2. Pour étudier le coût des algorithmes opérant sur des entiers, la complexité arithmétique avec $\mathbb{A} = \mathbb{Z}$ n'est pas pertinente. Il convient alors de décomposer les entiers dans une base B, qui est en pratique une puissance de 2 fixée (par exemple 2^{32}). Chaque entier est alors vu comme un vecteur d'éléments de $\{0, \ldots, B-1\}$. Nous appellerons *complexité binaire* la complexité arithmétique associée à $\mathbb{A} = \{0, \ldots, B-1\}$, muni des tests d'égalité et comparaisons, ainsi que des opérations arithmétiques modulo B. En pratique cette mesure de complexité reflète en général bien les temps observés pour les algorithmes opérant sur des nombres entiers ou rationnels. Néanmoins, précisons que cette mesure de complexité ne correspond pas à celle utilisée classiquement sur une machine de Turing, puisqu'elle néglige les coûts d'accès mémoire. Par exemple le coût binaire pour transposer une matrice de taille $n \times n$ à coefficients dans $\mathbb{A} = \{0, \ldots, B-1\}$ est nul avec notre définition.

EXEMPLE 3. L'addition de deux entiers de n bits prend donc $O(1)$ opérations arithmétiques et $O(n)$ opérations binaires (quelque soit la base B, puisque la différence entre les bases joue sur la constante cachée dans le $O()$).

EXEMPLE 4. Le calcul de $n!$ par la méthode naïve requiert n opérations arithmétiques et $O(n^2 \log^2 n)$ opérations binaires. Le choix de la base B n'intervient que dans la constante cachée dans le $O(\cdot)$. Nous verrons au Chapitre 15 qu'il est possible d'abaisser ce coût à seulement $O(n^{1/2} \log n)$ opérations arithmétiques dans $\mathbb{Z}$, et $O(n \log^3 n)$ opérations binaires. Les algorithmes rapides permettant d'atteindre ces complexités fournissent le meilleur algorithme connu de factorisation déterministe d'entiers et des algorithmes très efficaces pour le calcul de millions de décimales de π, $\log 2$ et de nombreuses autres constantes. (La notation $O(\cdot)$ est rappelée en Section 2.1).

Taille. Un algorithme et une structure de données sont généralement dotés d'une notion naturelle de taille et il s'agit d'étudier le coût de l'algorithme en fonction de cette taille. Pour simplifier, il est souvent commode de considérer le comportement asymptotique de ce coût lorsque la taille tend vers l'infini. Il est important de comprendre que la complexité d'un problème n'a de sens qu'une fois la structure de données fixée pour l'entrée comme pour la sortie.

Par exemple, pour les polynômes, le choix de la représentation dense mène à mesurer la complexité par rapport au degré, alors que le choix de la représentation creuse met en avant le nombre de monômes. Pour la factorisation, la complexité est polynomiale en le degré, mais exponentielle en le nombre de monômes, dans le cas le pire.

Cas le pire, cas moyen. La complexité dans le cas le pire est le maximum des complexités pour toutes les entrées d'une taille donnée. C'est celle que nous étudierons. Il est souvent utile de considérer aussi la complexité en moyenne, lorsque l'on peut mettre une mesure sur l'ensemble des entrées de taille bornée. Pour la plupart des algorithmes que nous étudierons dans la première partie de ce cours, il n'y a pas de différence importante entre les deux. Ce n'est plus le cas en revanche pour la complexité des algorithmes de factorisation, ou pour ceux qui opèrent sur les systèmes polynomiaux.

Bornes inférieures. La recherche de bornes inférieures de complexité est très difficile. Par exemple, à l'heure actuelle on ne sait pas prouver que la multiplication de matrices est nécessairement plus coûteuse qu'un nombre borné d'additions. Dès qu'il est possible de montrer que tous les bits de l'entrée doivent être pris en compte, la somme de la taille de l'entrée et de la taille de la sortie est une borne inférieure sur la complexité. En effet, dans le modèle MAD, chacune des écritures et des lectures prend une opération.

DEFINITION 2. Si N est la somme de la taille de l'entrée et de la taille de la sortie, un algorithme sera dit *quasi-optimal* lorsque sa complexité est bornée par $O(N \log^k N)$ pour un $k \geq 0$ arbitraire.

L'essentiel de la première partie du cours consistera à rechercher des algorithmes quasi-optimaux pour les opérations de base sur les structures de données fondamentales.

La notation $O(\cdot)$**.** Nous utilisons la notation $O(\cdot)$ pour exprimer une borne sur la complexité des algorithmes. La signification précise de la notation

$$f(n) = O(g(n)), \qquad n \to \infty$$

est qu'il existe $K > 0$ et $A > 0$ tels que pour tout $n > A$, f et g sont liés par l'inégalité

$$|f(n)| \leq K|g(n)|.$$

Lorsque plusieurs paramètres interviennent dans l'analyse de la complexité, il faut absolument préciser lequel tend vers l'infini pour que cette notation ait un sens. Si plusieurs d'entre eux tendent vers l'infini, soit ils sont liés par des inégalités qui seront précisées, soit la définition ci-dessus s'étend avec une constante K qui ne dépend d'aucun des paramètres.

Afin de simplifier les bornes de complexité nous utiliserons parfois la notation $\tilde{O}(\cdot)$ de sorte à cacher des facteurs logarithmiques. Plus précisément, nous écrirons

$$f(n) = \tilde{O}(g(n)), \qquad n \to \infty$$

lorsque qu'il existe un entier $k \geq 0$ tel que

$$f(n) = O(g(n) \log_2^k(\max(|g(n)|, 2))).$$

La notation $O(\cdot)$ intervient aussi dans ce cours pour représenter la troncature des séries. L'expression

$$f(X) := g(X) + O(X^N)$$

signifiera que le polynôme ou la série g est tronqué après son N^{e} terme et que le résultat, un polynôme, est stocké dans f.

2.2. Diviser pour régner. Le principe le plus important de la conception d'algorithmes efficaces est le paradigme « diviser pour régner ». Il consiste à résoudre un problème en le réduisant à un certain nombre m d'entrées de tailles divisées par p (le plus souvent $p = 2$) puis à recombiner les résultats (voir Figure 1). Le coût de la recombinaison et éventuellement du découpage préliminaire est borné par une fonction T de la taille des entrées. Lorsque les entrées ont une taille inférieure à p ou suffisamment petite, notée s, un autre algorithme de coût κ indépendant de n est invoqué. Le coût total obéit alors souvent à une récurrence de la forme

$$C(n) \leq \begin{cases} T(n) + mC(\lceil n/p \rceil), & \text{si } n \geq s(\geq p) \\ \kappa & \text{sinon.} \end{cases} \tag{2}$$

La notation $\lceil x \rceil$ désigne l'entier k tel que $k - 1 < x \leq k$ (le « plafond » de x).

Qualitativement, le coût total de cette approche dépend fortement de la fonction T. Lorsque T est relativement élevée, les premiers niveaux de l'arbre contribuent à l'essentiel du coût et, à une constante près, le coût est donc dominé par la première étape de récursion. Les algorithmes à base d'itération de Newton du Chapitre 3 sont de ce type. À l'inverse, pour une fonction T assez faible, le bas de l'arbre domine le coût qui sera proportionnel au nombre de feuilles [2] $O((n/s)^{\log_p m})$. L'algorithme de multiplication de polynômes de Karatsuba du Chapitre 2 et l'algorithme de Strassen pour la multiplication de matrices (Chapitre 8) rentrent dans cette catégorie. Enfin, il est possible que tous les $O(\log_p(n/s))$ niveaux de l'arbre contribuent de manière assez équilibrée, menant à un coût en $O(T(n) \log n)$. La transformée de Fourier rapide (Chapitre 2) et le classique algorithme de tri fusion sont dans cette dernière catégorie.

Un cadre commode pour nos applications est résumé dans la proposition suivante, dont une version simplifiée que nous utiliserons dans la plupart des cas est

2. La notation $\log_p x$ représente le logarithme en base p de x, c'est-à-dire $\log x / \log p$. L'identité facile à vérifier $a^{\log_p b} = b^{\log_p a}$ sera utile.

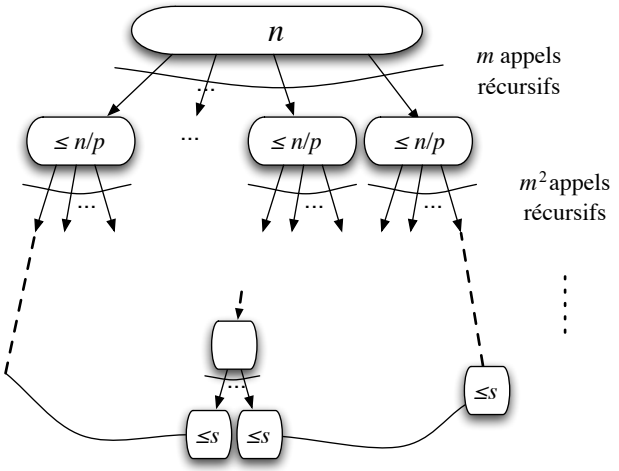


FIGURE 1. Les appels récursifs d'un algorithme « diviser pour régner ».

donnée ensuite par le théorème 2. Nous commençons par le cas simple où la taille est une puissance de p, où il est possible de donner des inégalités précises, ce qui prépare le passage au cas général et à l'asymptotique.

PROPOSITION 9. *Soit C une fonction obéissant à l'inégalité (2) avec $m > 0$, $\kappa > 0$ et T une fonction telle que*

$$T(pn) \geq qT(n), \qquad n \in \mathbb{N}, \tag{3}$$

avec $q > 1$. Alors, si n est une puissance positive de p,

$$C(n) \leq \begin{cases} \left(1 - \frac{m}{q}\right)^{-1} T(n) + \kappa\, n^{\log_p m} & \text{si } q > m, \\ T(n) \log_p n + \kappa\, n^{\log_p q} & \text{si } q = m, \\ n^{\log_p m} \left(\frac{T(n)}{n^{\log_p q}} \frac{q^{\log_p s}}{m-q} + \kappa \right) & \text{si } q < m. \end{cases}$$

DÉMONSTRATION. Une première preuve, laissée en exercice, consiste à prouver ce résultat par récurrence, mais nous préférons faire voir d'où viennent ces formules. Par récurrence, l'inégalité (3) entraîne d'abord $q^{\log_p n} T(1) \leq T(n)$. Ensuite, l'utilisation répétée de l'inégalité sur C donne

$$\begin{aligned} C(n) &\leq T(n) + mC\left(\frac{n}{p}\right), \\ &\leq T(n) + mT\left(\frac{n}{p}\right) + \cdots + m^{k-1} T\left(\frac{n}{p^{k-1}}\right) + m^k C\left(\frac{n}{p^k}\right), \qquad (4) \\ &\leq T(n)\left(1 + \frac{m}{q} + \cdots + \left(\frac{m}{q}\right)^{k-1}\right) + m^k \kappa, \qquad (5) \end{aligned}$$

où la dernière ligne résulte de (3) et du choix de $k = \lfloor \log_p \frac{n}{s} \rfloor + 1 \in [\log_p \frac{n}{s}, \log_p n]$. Ce choix entraine la suite d'inégalités suivante :

$$m^k \leq m \cdot m^{\log_p (n/s)} = n^{\log_p m} \cdot m^{1 - \log_p s} \leq n^{\log_p m}$$

qui permet de borner le deuxième terme de (5).

Si $m < q$, la somme entre parenthèses est majorée par la série géométrique.

Si $m = q$, la somme comporte k termes égaux.

Si $m > q$, récrire la somme sous la forme

$$\left(\frac{m}{q}\right)^{k-1}\left(1+\frac{q}{m}+\cdots+\left(\frac{q}{m}\right)^{k-1}\right)$$

montre que le premier terme de (5) est majoré par

$$T(n)\left(\frac{m}{q}\right)^{k}\frac{1}{m-q}.$$

Ensuite, les inégalités $q^{-k} \le q^{-\log_p n/s} = n^{-\log_p q} q^{\log_p s}$ permettent de conclure. □

En pratique la régularité de T est souvent plus forte que celle donnée par (3) et on ne cherche qu'à comparer des estimations de $O()$. Le résultat prend alors une forme plus simple, et sans restriction sur n.

THÉORÈME 2 (Diviser pour régner). *Soit C une fonction obéissant à l'inégalité (2) avec $m > 0$ et $\kappa > 0$, et soit T une fonction* croissante *telle qu'existent q et r avec $1 < q \le r$ vérifiant*

$$qT(n) \le T(pn) \le rT(n), \qquad \textit{pour tout } n \textit{ assez grand.} \tag{6}$$

Alors, lorsque $n \to \infty$

$$C(n) = \begin{cases} O(T(n)), & \textit{si } q > m, \\ O(T(n)\log n), & \textit{si } q = m, \\ O\left(n^{\log_p m}\frac{T(n)}{n^{\log_p q}}\right) & \textit{si } q < m. \end{cases}$$

L'hypothèse (6) est en particulier vérifiée par les fonctions courantes $n^\alpha \log^\beta n$ avec $\alpha > 0$.

DÉMONSTRATION. Soit N la puissance de p telle que $n \le N < pn$. En déroulant l'inégalité (2) on obtient une inégalité similaire à (4) où la division par p est remplacée par l'opération $x \mapsto \lceil x/p \rceil$. La fonction T étant supposée croissante, toutes ces valeurs de T sont majorées par les valeurs en N/p^i, $i = 0, \ldots, k-1$ et on obtient donc une majoration (5) où n est remplacé par N. Il ne reste plus qu'à majorer cette expression. La croissance de T et l'hypothèse (6) donnent pour n assez grand $T(N) \le T(pn) = O(T(n))$, ce qui permet de conclure. □

EXEMPLE 5. La complexité de l'algorithme de Karatsuba du Chapitre 2 vérifie la récurrence

$$K(n) \le 4n + 3K(\lceil n/2 \rceil), \qquad K(1) = 1.$$

Le lemme 9 et le théorème 2 s'appliquent avec $m = 3$, $p = s = 2$, $\kappa = 1$ et T linéaire, si bien que $q = r = 2$, d'où une complexité $K(n) = O(n^{\log_2 3})$, et plus précisément

$$K(n) \le n^{\log_2 3}\left(\frac{4n}{n}\frac{2}{1}+1\right) = 9n^{\log_2 3}$$

pour n une puissance de 2, et donc en majorant par la valeur en la puissance de 2 immédiatement supérieure, le cas général est borné par

$$K(n) \le 9 \cdot 2^{\lceil \log_2 n \rceil \log_2 3} = 9 \cdot 3^{\lceil \log_2 n \rceil}.$$

Cette borne est assez fine, comme le montre la Figure 2.

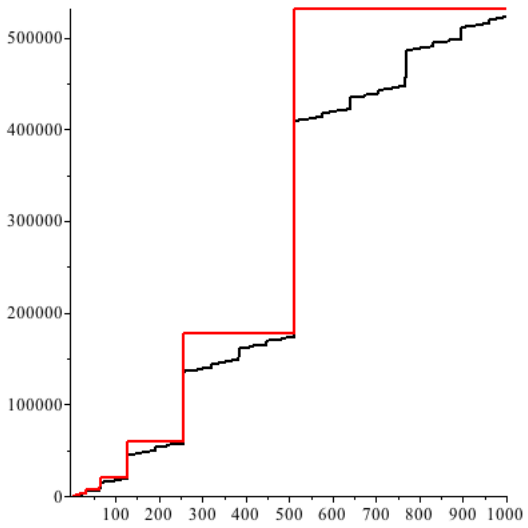


FIGURE 2. La suite solution de $C(n) = 4n + 3C(\lceil n/2 \rceil)$, $C(1) = 1$ (en noir) et la borne obtenue (en rouge).

Conventions

Pour toute la suite du cours, et sauf mention contraire, nous adoptons les notations et terminologies suivantes :

- tous les anneaux et corps sont supposés effectifs, commutatifs et unitaires ;
- les estimations de complexité sont en nombre d'opérations arithmétiques dans l'anneau de base ;
- le symbole □ marque la fin d'une démonstration ;
- les logarithmes sont en base 2, lorsque la base n'est pas spécifiée ;
- le lemme « diviser pour régner » fait référence à l'énoncé de la Proposition 9 ;
- le théorème « diviser pour régner » fait référence à l'énoncé du Théorème 2.

Notes

Les références générales sur les algorithmes du calcul formel sont deux livres : celui de von zur Gathen et Gerhard [8] et celui, plus élémentaire, de Geddes, Czapor et Labahn [9]. La complexité est également utilisée comme fil conducteur dans le livre plus difficile de Bürgisser, Clausen et Shokrollahi [4]. Une bonne introduction à la seconde partie du cours, sur les systèmes polynomiaux, est le livre de Cox, Little et O'Shea [5]. De premiers éléments pour aborder la partie 3, sur les équations différentielles et les récurrences linéaires, sont donnés dans le livre $A = B$ de Petkovšek, Wilf et Zeilberger [12].

Le théorème de Richardson-Matiyasevich que nous énonçons a ses racines dans les travaux sur le 10$^{\text{e}}$ problème de Hilbert portant sur l'existence d'un algorithme permettant de trouver les racines entières des polynômes multivariés à coefficients entiers. Une étape importante dans la résolution de ce problème est due à Davis, Putnam et Robinson [6] qui montrent en 1961 l'indécidabilité de ce problème si en plus des polynômes on considère la fonction $x \mapsto 2^x$. Quelques années plus tard, Richardson [13] ramène les problèmes de racines réelles en plusieurs variables à ces questions, et, à l'aide d'un codage astucieux par le sinus, y ramène aussi les

problèmes en une variable. Par rapport à notre énoncé, son théorème fait intervenir la fonction exp et la constante $\ln 2$ (pour utiliser le résultat de Davis, Putnam et Robinson), ainsi que la constante π (pour utiliser la nullité du sinus aux multiples *entiers* de π). Deux ans plus tard, en 1970, Matiyasevich prouve que la fonction $x \mapsto 2^x$ elle-même peut s'exprimer à l'aide de polynômes, concluant ainsi la preuve de l'indécidabilité du 10e problème de Hilbert. Le théorème de Richardson peut alors être simplifié, et c'est ce que fait Matiyasevich dans son livre [11], où il montre aussi comment se débarrasser de la constante π. Il faut noter que par contre, si l'on remplace le sinus par l'exponentielle, alors la simplification devient décidable [14].

Ces théorèmes s'appliquent à des fonctions. Pour des constantes, l'approche la plus récente réduit le test à zéro à une conjecture de théorie des nombres due à Schanuel qui exprime que les seules relations entre exponentielles et logarithmes sont celles qui découlent des formules d'addition et de multiplication. Si l'on accepte cette conjecture de Schanuel, alors un résultat relativement récent de Macintyre et Wilkie en 1996 [10] entraîne l'existence d'un algorithme de reconnaissance de 0 pour la classe des constantes obtenues à partir de 1 par addition, soustraction, multiplication et exponentielle. Un tel algorithme, à base d'évaluation numérique et de LLL (voir Chap. 31) a ensuite été donné en 1997, à nouveau par Richardson [15].

Les différents modèles de complexité (machine MAD, *straight-line program*, machine de Turing, ...) sont bien présentés par Aho, Hopcrof et Ullman dans [1].

La jolie identité (1) est tirée de [2].

Les résultats de complexité autour de l'algorithmique du « diviser pour régner » sont nombreux. Le premier remonte sans doute à Bentley, Haken et Saxe [3] qui font apparaître la série géométrique. Une version moderne tenant compte des planchers et des plafonds est présentée par Cormen *et alii* qui l'appellent *Master Theorem*. Des variantes sophistiquées sont connues [7, 16]. Dans cette direction, Yap [17] donne un théorème prêt à l'emploi dans le cas où plusieurs divisions avec des valeurs de p distinctes ont lieu.

Bibliographie

[1] Aho, Alfred V., John E. Hopcroft et Jeffrey D. Ullman (1974). *The design and analysis of computer algorithms*. Addison-Wesley Publishing Co., x+470 pages.

[2] Beck, Matthias, Bruce C. Berndt, O-Yeat Chan et Alexandru Zaharescu (2005). « Determinations of Analogues of Gauss Sums and Other Trigonometric Sums ». In : *International Journal of Number Theory*, vol. 1, n°3, p. 333–356.

[3] Bentley, Jon Louis, Dorothea Haken et James B. Saxe (1980). « A general method for solving divide-and-conquer recurrences ». In : *SIGACT News*, vol. 12, n°3, p. 36–44. doi : 10.1145/1008861.1008865.

[4] Bürgisser, Peter, Michael Clausen et M. Amin Shokrollahi (1997). *Algebraic complexity theory*. Vol. 315. Grundlehren der Mathematischen Wissenschaften. Springer-Verlag, xxiv+618 pages.

[5] Cox, David, John Little et Donal O'Shea (1996). *Ideals, varieties, and algorithms*. 2e éd. Springer-Verlag, xiv+536 pages.

[6] Davis, Martin, Hilary Putnam et Julia Robinson (1961). « The decision problem for exponential diophantine equations ». In : *Annals of Mathematics. Second Series*, vol. 74, p. 425–436.

[7] Drmota, Michael et Wojciech Szpankowski (2011). « A master theorem for discrete divide and conquer recurrences ». In : *SODA'11 : ACM-SIAM Symposium on Discrete Algorithms*. San Francisco, CA : SIAM, p. 342–361.

[8] Gathen, Joachim von zur et Jürgen Gerhard (1999). *Modern computer algebra*. Cambridge University Press, xiv+753 pages.

[9] Geddes, Keith O., Stephen R. Czapor et George Labahn (1992). *Algorithms for Computer Algebra*. Kluwer Academic Publishers.

[10] Macintyre, Angus et A. J. Wilkie (1996). « On the decidability of the real exponential field ». In : *Kreiseliana*. A. K. Peters, p. 441–467.

[11] Matiyasevich, Yuri V. (1993). *Hilbert's tenth problem*. Foundations of Computing Series. Translated from the 1993 Russian original by the author, With a foreword by Martin Davis. MIT Press, xxiv+264 pages.

[12] Petkovšek, Marko, Herbert S. Wilf et Doron Zeilberger (1996). $A = B$. A. K. Peters, xii+212 pages.

[13] Richardson, Daniel (1968). « Some Undecidable Problems Involving Elementary Functions of a Real Variable ». In : *Journal of Symbolic Logic*, vol. 33, n°4, p. 514–520.

[14] — (1969). « Solution of the identity problem for integral exponential functions ». In : *Zeitschrift fur mathematischen Logik und Grundlagen der Mathematik*, vol. 15, p. 333–340.

[15] — (1997). « How to recognize zero ». In : *Journal of Symbolic Computation*, vol. 24, n°6, p. 627–645.

[16] Roura, Salvador (2001). « Improved master theorems for divide-and-conquer recurrences ». In : *Journal of the Association for Computing Machinery*, vol. 48, n°2, p. 170–205. doi : 10.1145/375827.375837.

[17] Yap, Chee (2011). « A real elementary approach to the master recurrence and generalizations ». In : *Theory and applications of models of computation*. Vol. 6648. Lecture Notes in Computer Science. Tokyo, Japan : Springer-Verlag, p. 14–26. doi : 10.1007/978-3-642-20877-5_3.

Première partie

Polynômes et séries

CHAPITRE 2

Multiplication rapide

Résumé

Les algorithmes rapides de calcul de produit de polynômes et d'entiers sont au cœur de l'algorithmique efficace en calcul formel. La plupart des gains de complexité dans les chapitres ultérieurs reposent sur l'efficacité de la multiplication. Pour multiplier deux polynômes de degré n à coefficients dans un anneau $\mathbb{A}$, la méthode classique requiert $O(n^2)$ opérations dans $\mathbb{A}$. De même, l'algorithme scolaire de multiplication de deux entiers à n chiffres nécessite un nombre d'opérations binaires en $O(n^2)$. Nous présentons dans ce chapitre plusieurs algorithmes de multiplication rapide, dont celui de Karatsuba, de complexité $O(n^{1,59})$, ainsi que ceux utilisant la transformée de Fourier rapide, dont la complexité est quasiment linéaire en n.

1. Introduction, résultats principaux

Les problèmes abordés dans ce chapitre concernent la complexité arithmétique de la multiplication des polynômes à une variable et la complexité binaire de la multiplication des entiers. Au vu de l'exemple suivant, il est facile de se convaincre de la similitude des deux questions :

Polynômes : Soient à multiplier $3X^2 + 2X + 1$ et $6X^2 + 5X + 4$ dans $\mathbb{Z}[X]$.

$$(3X^2 + 2X + 1) \times (6X^2 + 5X + 4)$$

$$= (3 \cdot 6)X^4 + (3 \cdot 5 + 2 \cdot 6)X^3 + (3 \cdot 4 + 2 \cdot 5 + 1 \cdot 6)X^2 + (2 \cdot 4 + 1 \cdot 5)X + (1 \cdot 4)$$

$$= 18X^4 + 27X^3 + 28X^2 + 13X + 4.$$

Nombres entiers : Soient à multiplier 321 et 654 en base 10.

$$(3 \cdot 10^2 + 2 \cdot 10 + 1) \times (6 \cdot 10^2 + 5 \cdot 10 + 4)$$

$$= (3 \cdot 6)10^4 + (3 \cdot 5 + 2 \cdot 6)10^3 + (3 \cdot 4 + 2 \cdot 5 + 1 \cdot 6)10^2 + (2 \cdot 4 + 1 \cdot 5)10 + (1 \cdot 4)$$

$$= 18 \cdot 10^4 + 27 \cdot 10^3 + 28 \cdot 10^2 + 13 \cdot 10 + 4$$

$$= 2 \cdot 10^5 + 9 \cdot 10^3 + 9 \cdot 10^2 + 3 \cdot 10 + 4 = 209934.$$

Dans les deux cas, nous avons retranscrit l'algorithme naïf, et la suite des calculs est essentiellement la même, si ce n'est que, dans le cas des entiers, il faut en outre gérer les retenues (dernière égalité de l'exemple). On ne sera donc pas surpris que les résultats obtenus dans les deux cas soient très semblables.

Résultats. Dans toute la suite, $(\mathbb{A}, +, \times)$ désignera un anneau commutatif. Tout d'abord, nous considérons la complexité arithmétique ; il s'agit de minimiser le nombre d'opérations $(+, -, \times)$ dans $\mathbb{A}$ pour multiplier des polynômes en degré borné. Les premiers résultats à retenir de ce chapitre sont les suivants.

La multiplication des polynômes de degré au plus n dans $\mathbb{A}[X]$ requiert :

- $O(n^2)$ *opérations dans* $\mathbb{A}$ *par* l'algorithme naïf *;*
- $O(n^{1,59})$ *opérations dans* $\mathbb{A}$ *par* l'algorithme de Karatsuba *;*
- $O(n \log n \log\log n)$*, voire dans certains cas* $O(n \log n)$ *opérations dans* $\mathbb{A}$*, via* la transformée de Fourier rapide (FFT).

Ainsi, la multiplication des polynômes peut se faire en un coût arithmétique *essentiellement linéaire* en leur degré.

La multiplication des polynômes est omniprésente : les algorithmes de calcul de pgcd (plus grand commun diviseur), de pgcd étendu, de factorisation en une ou plusieurs variables, de composition des séries formelles, d'évaluation multipoint, d'interpolation, font tous intervenir des produits de polynômes.

L'analogie entre les entiers et les polynômes va très loin ; la plupart des réponses apportées dans le cadre de la complexité arithmétique trouvent un équivalent en complexité binaire. Cependant, aucun théorème d'équivalence n'est connu ; il se trouve que les mêmes idées algorithmiques s'adaptent plus ou moins facilement dans les deux cadres. Ainsi, on dispose des résultats suivants dans le modèle binaire.

On peut multiplier des entiers de n chiffres binaires par :

- l'algorithme naïf *en* $O(n^2)$ *opérations binaires ;*
- l'algorithme de Karatsuba *en* $O(n^{1,59})$ *opérations binaires ;*
- l'algorithme de Schönhage-Strassen *en* $O(n \log n \log\log n)$ *opérations binaires.*

Les preuves de ces résultats de complexité binaire sont plus délicates que celles de leurs analogues polynomiaux, à cause des problèmes de gestion des retenues. Ainsi, dans la suite, nous ne traitons d'abord en détail que les versions polynomiales de ces résultats, le cas entier étant ensuite brièvement passé en revue.

En pratique. Les constantes cachées dans les $O(\cdot)$ sont déterminantes pour l'efficacité pratique de tels algorithmes. Par exemple, lorsque $\mathbb{A}$ est un corps fini de taille « raisonnable » (typiquement, dont les éléments sont représentés sur quelques mots machine), pour le produit de polynômes dans les meilleures implantations actuelles (Magma, NTL) :

- l'algorithme de Karatsuba bat l'algorithme naïf pour des degrés d'environ 20 ;
- les méthodes à base de FFT en $O(n \log n)$ battent l'implantation de Karatsuba pour des degrés de l'ordre de 100, mais ne peuvent pas être utilisées pour des degrés arbitrairement grands (vient un moment où on manque de racines de l'unité, voir plus loin) ;
- l'algorithme de type FFT en $O(n \log n \log\log n)$ est utilisé pour des degrés de l'ordre de quelques dizaines ou centaines de milliers.

Certains problèmes, en cryptologie ou en théorie des nombres, nécessitent de manipuler des polynômes de degré de l'ordre de 100 000, tailles auxquelles les algorithmes rapides sont indispensables. Plus fréquemment, des degrés de l'ordre de la centaine ou du millier font partie du quotidien du calculateur formel (au moins dans des calculs intermédiaires).

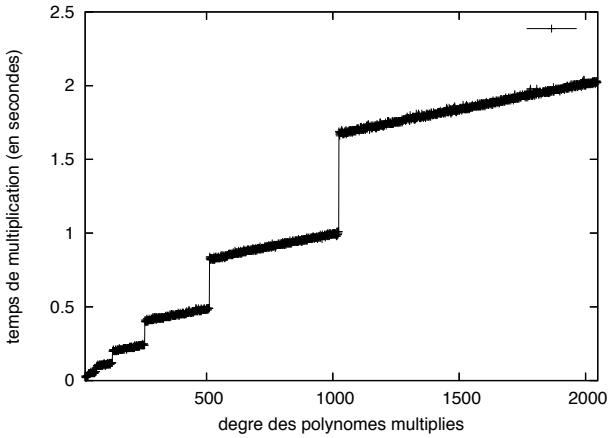


FIGURE 1. Courbe de complexité pratique de la multiplication dans $\mathbb{A}[X]$, où $\mathbb{A} = \mathbb{Z}/4179340454199820289\,\mathbb{Z}$.

Ces algorithmes ne sont pas concurrents, mais complémentaires. Les bonnes implantations passent automatiquement sur l'algorithme le plus efficace en fonction du degré. Nous donnons dans la Figure 1 la courbe de complexité pratique [1] de la multiplication polynomiale à coefficients dans le corps fini $\mathbb{A} = \mathbb{Z}/4179340454199820289\,\mathbb{Z}$.

L'allure de cette courbe confirme que les estimations théoriques de complexité sont respectées en pratique. Le comportement quasi-linéaire *par morceaux*, agrémenté de sauts entre les puissances successives de 2, est typique des implantations actuelles de la FFT.

La situation pour les entiers est similaire, même si l'implantation des algorithmes rapides pour les entiers est bien plus délicate en raison des retenues. Dans les meilleures implantations actuelles (Magma, GMP) :

— l'algorithme de Karatsuba bat l'algorithme naïf pour des nombres de l'ordre de 100 chiffres binaires ;

— les méthodes à base de FFT (Schönhage-Strassen) gagnent pour des nombres d'environ 10 000 chiffres binaires.

À nouveau, des entiers de quelques centaines de chiffres sont courants dès que l'on sort des exemples jouets. Au-delà, des problèmes venus de la cryptologie ou de la théorie des nombres demandent de manipuler des nombres de taille colossale (de l'ordre de 100 000 000 de chiffres ; il faut 10 Mo pour stocker un tel nombre). Ceci justifie amplement les efforts d'implantation d'algorithmes rapides.

NOTATION 1. On travaille avec des polynômes F et G à coefficients dans un anneau $\mathbb{A}$, ayant un degré au plus $n-1$, formellement

$$F = f_0 + \cdots + f_{n-1}X^{n-1} \quad \text{et} \quad G = g_0 + \cdots + g_{n-1}X^{n-1};$$

le problème est alors de calculer (les coefficients de)

$$H = FG = h_0 + \cdots + h_{2n-2}X^{2n-2}.$$

2. Algorithme naïf

Cet algorithme consiste à développer le produit, c'est-à-dire à écrire

$$H = FG = \sum_{i=0}^{2n-2} h_i X^i \quad \text{avec} \quad h_i = \sum_{j+k=i} f_j g_k.$$

1. Calculs effectués avec le logiciel Magma, version V2.12-1, sur un Opteron 150 (2,4 GHz).

Ainsi, calculer tous les h_i demande $O(n^2)$ opérations dans $\mathbb{A}$. C'est un algorithme de complexité arithmétique *quadratique*.

EXERCICE 1. Montrer que, pour multiplier deux polynômes de degrés m et n, l'algorithme naïf demande au plus $(m+1)\times(n+1)$ multiplications dans $\mathbb{A}$ et mn additions dans $\mathbb{A}$.

EXERCICE 2. Montrer que, pour multiplier deux entiers à n chiffres chacun, l'algorithme naïf demande $O(n^2)$ opérations binaires.

EXERCICE 3. Estimer la complexité binaire de la méthode naïve lorsque les polynômes ont degré n et des coefficients entiers bornés en valeur absolue par un entier H.

EXERCICE 4. Alexander J. Yee et Shigeru Kondo ont calculé 10,000,000,000,000 décimales de π sur un PC de bureau en 2011 [2]. Ce type de calcul repose sur la multiplication d'entiers. En supposant que leur machine était capable d'effectuer 10^{12} opérations à la seconde, montrer qu'ils n'ont pas utilisé l'algorithme naïf.

3. Algorithme de Karatsuba

Un premier raffinement de l'algorithme naïf repose sur la remarque suivante : il est possible de gagner *une* multiplication pour le produit des polynômes de degré 1, parmi les 4 du produit par l'algorithme quadratique. Soient en effet à multiplier les polynômes

$$F = f_0 + f_1 X \quad \text{et} \quad G = g_0 + g_1 X.$$

Leur produit $H = FG = h_0 + h_1 X + h_2 X^2$ peut être obtenu par une forme d'interpolation sur les points $0,1,\infty$: en 0 on a $H(0) = h_0 = f_0 g_0$; de même, le coefficient de plus haut degré (qui correspond intuitivement à l'évaluation en l'infini) vaut $h_2 = f_1 g_1$. Enfin, la valeur en 1 vaut $h_0 + h_1 + h_2 = F(1)G(1) = (f_0 + f_1)(g_0 + g_1)$. Ainsi, on obtient $h_1 = (f_0+f_1)(g_0+g_1) - h_0 - h_2$ pour seulement une multiplication supplémentaire, donc l'ensemble des coefficients du produit pour 3 multiplications et 4 additions.

Quelques additions sont perdues par rapport à l'algorithme naïf, mais le gain d'une multiplication va se transformer en gain dans l'*exposant* de l'algorithme, par application récursive.

En effet, dans le cas général des degrés quelconques, il suffit de scinder F et G en deux et de procéder de la même manière. Si F et G sont de degré au plus $n-1$, avec $k = \lceil n/2 \rceil$, on pose

$$F = F^{(0)} + F^{(1)} X^k, \quad G = G^{(0)} + G^{(1)} X^k, \tag{1}$$

pour des polynômes $F^{(0)}, F^{(1)}, G^{(0)}, G^{(1)}$ de degrés au plus $k-1$. Le produit $H = FG$ s'écrit

$$H = F^{(0)}G^{(0)} + (F^{(0)}G^{(1)} + F^{(1)}G^{(0)})X^k + F^{(1)}G^{(1)}X^{2k}.$$

L'algorithme est présenté en Figure 2.

Le théorème qui suit établit la complexité de l'algorithme.

THÉORÈME 1. *Si n est une puissance de 2, l'algorithme de Karatsuba calcule le produit de deux polynômes de degré au plus $n-1$ en au plus $9n^{\log_2 3}$ opérations dans $\mathbb{A}$.*

DÉMONSTRATION. Un appel en degré $< n$ effectue 3 appels récursifs en degré $< n/2$, plus quelques additions, comptées comme suit : l'étape 4 effectue 2 additions en tailles $< n/2$; l'étape 6 effectue 2 additions en tailles $< n$; quant à l'étape 7, A_1

2. Voir http://www.numberworld.org/misc_runs/pi-5t/announce_en.html.

Algorithme de Karatsuba

Entrée : F, G de degré au plus $n-1$.

Sortie : $H = FG$.

1. Si $n = 1$, renvoyer FG.
2. Décomposer F et G selon l'équation (1).
3. Calculer $A_1 = F^{(0)}G^{(0)}$ et $A_2 = F^{(1)}G^{(1)}$ récursivement.
4. Calculer $A_3 = F^{(0)} + F^{(1)}$ et $A_4 = G^{(0)} + G^{(1)}$.
5. Calculer $A_5 = A_3A_4$ récursivement.
6. Calculer $A_6 = A_5 - A_1$ et $A_7 = A_6 - A_2$.
7. Renvoyer $A_1 + A_7X^k + A_2X^{2k}$.

FIGURE 2. Multiplication de polynômes par l'algorithme de Karatsuba.

et A_2X^n sont à supports monomiaux distincts, donnant la somme $A_1 + A_2X^n$ sans coût arithmétique, puis la somme totale en 1 addition en taille $< n$. Le coût $K(n)$ satisfait donc à la récurrence

$$K(n) \leq 3K(n/2) + 4n,$$

où le terme $4n$ vient compter le nombre d'additions dans $\mathbb{A}$. Le lemme « diviser pour régner » permet alors de conclure avec $p = q = s = 2$, $m = 3$, $T(n) = 4n$, et $\kappa = 1$ (coût de l'algorithme naïf en degré 0). □

Le théorème « diviser pour régner » fournit ensuite.

COROLLAIRE 1. *On peut multiplier deux polynômes de degré n (quelconque) en $O(n^{\log_2 3}) = O(n^{1,59})$ opérations dans $\mathbb{A}$.*

Une manière d'estimer plus précisément la constante cachée dans cette estimation a été présentée au Chapitre 1.

En fonction de la nature de l'anneau ou du corps de base, on peut vouloir arrêter les appels récursifs avant d'avoir atteint le degré 0. Ce choix dépend des vitesses relatives de l'addition et de la multiplication.

EXERCICE 5. Soit n une puissance de 2. Établir un algorithme hybride de multiplication dans $\mathbb{A}[X]$, qui fait appel à l'algorithme de Karatsuba pour $n > 2^d$ et à l'algorithme naïf pour $n \leq 2^d$. Montrer que la complexité arithmétique $\mathsf{C}(n)$ de cet algorithme vérifie $\mathsf{C}(n) \leq \gamma(d)n^{\log_2 3} - 8n$ pour tout $n \geq 2^d$, où $\gamma(d)$ est une fonction qui dépend uniquement de d. Trouver la valeur de d qui minimise $\gamma(d)$ et, par comparaison avec le résultat du Théorème 1, estimer le gain obtenu par cette optimisation.

EXERCICE 6. Est-il possible de multiplier par un algorithme universel, c'est-à-dire indépendant de l'anneau de base $\mathbb{A}$, deux polynômes de degré au plus 1 en utilisant seulement 2 multiplications dans $\mathbb{A}$?

EXERCICE 7 (Algorithme de Toom-Cook). Soit $\mathbb{A}$ un anneau et soient A et B deux polynômes de degré au plus 3 dans $\mathbb{A}[X]$.

1. Estimer le nombre de multiplications de $\mathbb{A}$ requises par l'algorithme de Karatsuba pour calculer le produit AB.
2. On suppose que 2, 3 et 5 sont inversibles dans $\mathbb{A}$ et que la division d'un élément de $\mathbb{A}$ par 2, 3 et 5 est gratuite. Donner un algorithme qui multiplie A et B en utilisant au plus 7 multiplications dans $\mathbb{A}$.

Multiplication par FFT

Entrée : F, G deux polynômes, n un entier, ω une racine principale n^{e} de 1.

Sortie : $H = FG \bmod X^n - 1$.

1. *Précalcul.* Calculer les puissances $\omega^2, \ldots, \omega^{n-1}$.
2. *Évaluation.* Calculer les valeurs :
$$\mathsf{Ev}(F) = \big(F(\omega^0), \ldots, F(\omega^{n-1})\big); \quad \mathsf{Ev}(G) = \big(G(\omega^0), \ldots, G(\omega^{n-1})\big).$$
3. *Produit point à point.*
$$\big(\mathsf{Ev}(F), \mathsf{Ev}(G)\big) \mapsto \mathsf{Ev}(FG) = \big(FG(\omega^0), \ldots, FG(\omega^{n-1})\big).$$
4. *Interpolation.*
$$\mathsf{Ev}(FG) \mapsto FG.$$

FIGURE 3. Multiplication de polynômes par transformée de Fourier discrète.

3. On suppose que 2, 3 et 5 sont inversibles dans $\mathbb{A}$. Donner un algorithme de multiplication polynomiale dans $\mathbb{A}[X]$ de complexité arithmétique $O(n^{1,4})$.

Dans la suite de l'exercice, on suppose que l'anneau $\mathbb{A}$ est de caractéristique nulle.

4. Montrer que, pour tout entier $\alpha \geq 2$, il existe un algorithme de multiplication polynomiale dans $\mathbb{A}[X]$ de complexité arithmétique $O(n^{\log_\alpha(2\alpha-1)})$.
5. Montrer que pour tout $\varepsilon > 0$, il existe un algorithme de multiplication polynomiale dans $\mathbb{A}[X]$ de complexité arithmétique $O(n^{1+\varepsilon})$, où la constante dans le $O(\cdot)$ dépend de ε, mais pas de n.

4. Transformée de Fourier rapide

Les méthodes à base de transformée de Fourier rapide (appelée aussi FFT pour *Fast Fourier Transform*) sont ce que l'on sait faire de mieux pour multiplier les polynômes. Pour simplifier la présentation, on suppose ici que l'on cherche à multiplier des polynômes F et G dans $\mathbb{A}[X]$, de degrés strictement inférieurs à $n/2$ (ou plus généralement tels que $\deg FG < n$).

4.1. Idée de l'algorithme. En supposant que l'anneau $\mathbb{A}$ le permette, l'idée générale de l'algorithme est présentée en Figure 3. Il s'agit d'évaluer en des points bien choisis, de multiplier les évaluations, et de reconstruire les coefficients du produit à partir de ces valeurs (à condition que cette opération d'interpolation soit possible, voir ci-dessous). Avec l'hypothèse $\deg H < n$, les coefficients de $H \bmod X^n - 1$ qui sont renvoyés sont bien ceux de H. Le coût des étapes de précalcul et de produit point à point est linéaire en n, et il reste donc à voir comment effectuer rapidement les opérations d'évaluation et d'interpolation.

4.2. Racines primitives et principales de l'unité.

DEFINITION 1. L'élément ω de $\mathbb{A}$ est une *racine n^e de l'unité* si $\omega^n = 1$; c'est une racine n^{e} *primitive* de l'unité si de plus $\omega^t \neq 1$ pour $t < n$; c'est une racine n^{e} *principale* de l'unité si de plus $\omega^t - 1$ est non diviseur de zéro dans $\mathbb{A}$ pour $t \in \{1, \ldots, n-1\}$ (c'est-à-dire que $\alpha(\omega^t - 1) = 0$ implique $\alpha = 0$).

EXEMPLE 1. Si $\mathbb{A}$ est un corps ou même un anneau intègre, les racines principales et primitives coïncident et la condition revient à dire que ω engendre le groupe des racines n^{es} de l'unité. Par exemple, dans $\mathbb{C}$, -1 n'est pas une racine 4^{e}

primitive de l'unité, alors que i l'est. Plus généralement, lorsque $\mathbb{A} = \mathbb{C}$, les racines primitives n^{e} de l'unité sont de la forme $\exp(2qi\pi/n)$, pour q premier avec n.

EXEMPLE 2. Dans $\mathbb{A} = \mathbb{Z}/25\mathbb{Z}$, les premières puissances de 6 sont 6,11,16,21,1. Comme $16-1 = 15$ est diviseur de 0, 6 est une racine 5^{e} de l'unité qui est primitive mais n'est pas principale. En revanche, les premières puissances de 7 sont 7,24,18,1, ce qui montre que 7 est une racine principale 4^{e} de 1 dans $\mathbb{A}$.

L'intérêt des racines principales de 1 dans des anneaux qui ne sont même pas intègres apparaît en fin de ce chapitre pour l'algorithme de Schönhage et Strassen. Jusque là, il est suffisant de considérer le cas où $\mathbb{A} = \mathbb{C}$ pour comprendre les idées. Les propriétés que nous utiliserons sont résumées dans le lemme suivant.

LEMME 1. *Si ω est racine primitive ou principale n^e de l'unité, alors*

1. *ω^{-1} aussi ;*
2. *si $n = pq$ alors ω^p est une racine q^e de l'unité de même nature que ω ;*
3. *pour $\ell \in \{1, \dots, n-1\}$, et ω une racine principale de l'unité alors*

$$\sum_{j=0}^{n-1} \omega^{\ell j} = 0.$$

DÉMONSTRATION. Tout d'abord ω est bien inversible : l'identité $\omega^n = 1$ montre que ω^{n-1} est un inverse de ω. Ensuite, ω^{-1} est une racine de l'unité : le produit de l'identité précédente par ω^{-n} donne $1 = \omega^{-n}$. Enfin, elle est principale si ω l'est : lorsque $\omega^t - 1$ n'est pas diviseur de 0, son produit par l'inversible ω^{-t} non plus. Le même argument s'applique dans le cas où ω est primitive.

La deuxième propriété est immédiate. Pour la troisième, le produit de la somme par $1 - \omega^\ell$ télescope les sommants, ce qui donne

$$(1-\omega^\ell)\sum_{j=0}^{n-1} \omega^{\ell j} = 1 - \omega^{\ell n} = 1 - (\omega^n)^\ell = 0.$$

Comme $1 - \omega^\ell$ n'est pas diviseur de 0, la somme est bien nulle. □

REMARQUE. Dans la définition de racine primitive ou principale, la condition *t diviseur strict de n* suffit, elle entraîne la propriété pour $t \in \{1, \dots, n-1\}$.

En effet, si t n'est pas diviseur de n, son pgcd g avec n l'est. Il existe d'après le théorème de Bézout deux entiers $p, q \in \mathbb{N}$ tels que $g = tp + nq$. Alors, l'égalité $\alpha(\omega^t - 1) = 0$ entraîne alors $0 = \alpha(\omega^t - 1)(1 + \omega^p + \cdots + \omega^{t(p-1)}) = \alpha(\omega^{tp} - 1) = \alpha(\omega^{g-nq} - 1) = \alpha(\omega^g - 1)$, et donc $\alpha = 0$, puisque g est un diviseur strict de n.

4.3. Transformée de Fourier rapide. L'opération

$$\mathsf{DFT} : F \in \mathbb{A}[X] \mapsto (F(1), F(\omega), \dots, F(\omega^{n-1})),$$

où ω est une racine principale n^{e} de l'unité, s'appelle la *transformée de Fourier discrète*. Son calcul rapide est effectué par un algorithme de type « diviser pour régner ».

Pour appliquer cette idée, supposons que n est pair, $n = 2k$. Alors, $\omega^k = -1$ puisque

$$(\omega^k - 1)(\omega^k + 1) = \omega^n - 1 = 0$$

et le premier facteur n'est pas diviseur de 0. Le polynôme F est décomposé par division euclidienne de deux façons :

$$F = Q_0(X^k - 1) + R_0 \quad \text{et} \quad F = Q_1(X^k + 1) + R_1,$$

Transformée de Fourier Rapide

Entrée : $F = f_0 + \cdots + f_{n-1}X^{n-1}$; les puissances $1, \omega, \cdots, \omega^{n-1}$ d'une racine n^{e} principale de l'unité ω, n étant une puissance de 2.

Sortie : $F(1), \ldots, F(\omega^{n-1})$.

1. Si $n = 1$, renvoyer f_0.
2. Sinon, soit $k = n/2$. Calculer

$$R_0(X) = \sum_{j=0}^{k-1} (f_j + f_{j+k})X^j, \quad \bar{R}_1(X) = R_1(\omega X) = \sum_{j=0}^{k-1} (f_j - f_{j+k})\omega^j X^j.$$

3. Calculer récursivement $R_0(1), R_0(\omega^2), \ldots, R_0\big((\omega^2)^{k-1}\big)$ et $\bar{R}_1(1), \bar{R}_1(\omega^2), \ldots, \bar{R}_1\big((\omega^2)^{k-1}\big)$.
4. Renvoyer $R_0(1), \bar{R}_1(1), R_0(\omega^2), \bar{R}_1(\omega^2), \ldots, R_0\big((\omega^2)^{k-1}\big), \bar{R}_1\big((\omega^2)^{k-1}\big)$.

FIGURE 4. La transformée de Fourier rapide (FFT).

avec $\deg R_0 < k$ et $\deg R_1 < k$. Ces décompositions vont nous permettre le calcul de F sur les puissances paires et impaires de ω. En effet, si ℓ est pair, $\omega^{k\ell} = 1$ et donc $F(\omega^\ell) = R_0(\omega^\ell)$. De même, si ℓ est impair, $F(\omega^\ell) = R_1(\omega^\ell)$.

Outre l'application récursive, le point crucial qui est la source de l'efficacité de l'algorithme de transformée de Fourier rapide (Figure 4) et qui conduit au choix de racines primitives de l'unité, est que le calcul de R_0 et R_1 est très simple (étape 2). Le fait que les ω^{2^i} utilisés dans les appels récursifs sont bien des racines primitives découle du Lemme 1.

THÉORÈME 2. *L'algorithme de Transformée de Fourier Rapide de la Figure 4 requiert au plus* $\frac{3n}{2} \log n$ *opérations dans* $\mathbb{A}$ *; les multiplications font toutes intervenir une puissance de* ω.

DÉMONSTRATION. Puisque les puissances de ω sont connues, le coût de l'appel en degré n est d'au plus $2 \times n/2$ additions et soustractions (pour le calcul de R_0 et R_1) et de $n/2$ multiplications (pour le calcul de $\bar{R}_1$), plus 2 appels récursifs en degré $n/2$. Sa complexité $F(n)$ satisfait donc à la récurrence :

$$F(n) \leq \frac{3n}{2} + 2F\left(\frac{n}{2}\right)$$

et le lemme « diviser pour régner » (avec $p = q = m = s = 2$, $T(n) = 3n/2$ et $\kappa = 0$) permet de conclure. □

EXERCICE 8. Montrer que l'algorithme de la Figure 4 requiert $n \log n$ additions dans $\mathbb{A}$, $\frac{1}{2}n \log n$ multiplications d'éléments de $\mathbb{A}$ par des puissances de ω, mais aucune autre multiplication dans $\mathbb{A}$.

REMARQUE. La transformée de Fourier discrète est un morphisme d'algèbres sur $\mathbb{A}$ de $\mathbb{A}[X]/(X^n - 1)$ dans $\mathbb{A}^n$ avec comme multiplication dans $\mathbb{A}^n$ la multiplication coordonnée par coordonnée. Cette observation permet d'économiser des transformées inverses en effectuant plusieurs calculs directement sur les transformées.

4.4. Interpolation. En termes matriciels, l'opération $F \mapsto \mathsf{Ev}(F)$ est linéaire et sa matrice (pour des polynômes F de degré au plus $n-1$, dans la base monomiale

$\{1, X, \ldots, X^{n-1}\}$) est la matrice de Vandermonde

$$V_\omega = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{n-1} \\ \vdots & & & \vdots \\ 1 & \omega^{n-1} & \cdots & \omega^{(n-1)^2} \end{bmatrix}.$$

LEMME 2. *Si $\omega \in \mathbb{A}$ est racine n^e principale de l'unité, alors $V_{\omega^{-1}} V_\omega = nI_n$.*

DÉMONSTRATION. D'après le Lemme 1, ω est inversible. La matrice $V_{\omega^{-1}}$ est donc bien définie. Le coefficient de la i^e ligne et j^e colonne du produit s'écrit

$$\sum_{k=0}^{n-1} \omega^{(i-1)k} \omega^{-(j-1)k} = \sum_{k=0}^{n-1} \omega^{(i-j)k}.$$

D'après le Lemme 1, cette dernière somme est nulle pour $0 < i-j < n$. Par symétrie des matrices, elle est aussi nulle si $0 < j - i < n$. Les coefficients en dehors de la diagonale sont donc tous nuls. Sur la diagonale, la somme comporte n termes, tous égaux à 1, d'où le résultat. □

Autrement dit, l'interpolation sur les puissances de ω se ramène à une FFT sur les puissances de ω^{-1}, qui est bien principale d'après le Lemme 1.

4.5. Conclusion. Les résultats de cette section sont résumés dans le théorème suivant.

THÉORÈME 3. *Si 2 est inversible dans $\mathbb{A}$ et n une puissance de 2, étant donnée une racine principale n^e dans $\mathbb{A}$, le produit de deux polynômes dont la somme des degrés est inférieure à n peut être calculé en $\frac{9}{2} n \log n + O(n)$ opérations dans $\mathbb{A}$. Seuls n des produits sont entre deux éléments de $\mathbb{A}$ qui ne sont pas des puissances de ω.*

DÉMONSTRATION. La complexité de l'algorithme de multiplication par FFT (Figure 3) est de 3 FFT en degré n, soit $\frac{9}{2} n \log n$ opérations, plus $O(n)$ divisions par n (ce qui est possible puisque 2 est inversible) et $O(n)$ multiplications pour calculer les puissances de ω. Les multiplications de la FFT font intervenir des puissances de ω, les autres sont celles de l'étape de produit point à point, au nombre de n. □

EXERCICE 9. Montrer que sous les hypothèses du théorème précédent, on peut multiplier ces polynômes en utilisant $3n \log n + O(n)$ additions dans $\mathbb{A}$, $\frac{3}{2} n \log n + O(n)$ multiplications par des puissances de ω, n multiplications arbitraires dans $\mathbb{A}$ et n divisions par n.

EXERCICE 10. Soit n dans $\mathbb{N}$, soit n_0 la plus petite puissance de 2 supérieure ou égale à n, et supposons qu'il existe dans $\mathbb{A}$ une racine $n_0{}^e$ principale de l'unité. Cette racine étant connue, on peut multiplier les polynômes de degré au plus $\frac{1}{2} n - 1$ en $9n \log n + O(n)$ opérations dans $\mathbb{A}$.

EXERCICE 11. Soit n une puissance de 2, et supposons qu'on dispose d'une racine n^e principale de l'unité $\omega \in \mathbb{A}$. Soit P et Q deux polynômes dans $\mathbb{A}[X]$ de degré au plus $n-1$. Supposons que les coefficients de $X^0, X^1, \ldots, X^{n-1}$ du produit $R = PQ$ sont connus. Montrer que R peut être calculé en $\frac{9}{2} n \log n + O(n)$ opérations dans $\mathbb{A}$.

4.6. Mais où sont nos racines ? Même dans le cas favorable où $\mathbb{A}$ est un corps, il n'y existe pas nécessairement toutes les racines primitives de l'unité nécessaires pour appliquer les techniques des algorithmes précédents. Dans le cas particulier des corps finis, on sait donner une réponse précise à cette question d'existence.

PROPOSITION 1. *Le corps fini à q éléments $\mathbb{F}_q$ contient une racine n^e primitive de l'unité si et seulement si n divise $q-1$.*

EXERCICE 12.
1. Prouver la proposition précédente.
2. Montrer que si n divise $q-1$ et si α est un élément primitif de $\mathbb{F}_q$ (*i. e.* tel que α engendre le groupe multiplicatif $(\mathbb{F}_q \setminus \{0\}, \times)$) alors $\alpha^{(q-1)/n}$ est une racine n^{e} primitive de l'unité.

Pour pouvoir appliquer le théorème 2 avec n assez grand et $\mathbb{A} = \mathbb{Z}/p\mathbb{Z}$ (c'est-à-dire $q = p$ premier), ce résultat mène à la notion de *premiers de Fourier*, qui sont les nombres premiers p tels que $p-1$ soit divisible par une grande puissance de 2, donc de la forme $\ell 2^e + 1$, avec e « suffisamment grand » (qui est appelé *l'exposant* de p). Par exemple,

$$4179340454199820289 = 29 \times 2^{57} + 1$$

est un tel nombre premier. Ainsi, dans $\mathbb{Z}/4179340454199820289\,\mathbb{Z}$, on dispose de racines primitives $2^{57\text{e}}$ de l'unité (21 en est une) ; on peut donc y multiplier des polynômes de degrés colossaux par cet algorithme en $O(n \log n)$ opérations.

Ces premiers sont donc très utiles pour la FFT sur des corps finis. On peut montrer qu'il y a approximativement $(x/\log x)/2^{e-1}$ premiers de Fourier d'exposant e inférieurs à x. Il s'ensuit qu'il y a environ 130 corps finis de la forme $k = \mathbb{F}_p$ tel que p a la taille d'un (demi-)mot machine (32 bits) et tel que dans k on puisse multiplier par FFT des polynômes de degré de l'ordre d'un million. Les racines de l'unité de $k = \mathbb{F}_p$ peuvent être calculées à partir des éléments primitifs. Si on choisit au hasard un élément de $\mathbb{F}_p$, lorsque $p \to \infty$, la probabilité qu'il soit primitif tend vers $6/\pi^2$, soit plus de $0,6$.

5. L'algorithme de Schönhage et Strassen

Quand les racines de l'unité font défaut, il reste possible de faire fonctionner les idées à base de transformée de Fourier. Ceci est réalisé par l'algorithme de Schönhage et Strassen. Cet algorithme s'applique quel que soit l'anneau de base, pourvu que 2 y soit inversible ; l'idée est de rajouter les racines de l'unité qui manquent en étendant l'anneau de base de manière judicieuse.

5.1. Racines virtuelles de l'unité. Le point de départ de l'algorithme est résumé dans le résultat suivant.

LEMME 3. *Si 2 est inversible dans $\mathbb{A}$ et n est une puissance de 2, alors $\omega = Y \bmod (Y^n+1)$ est une racine $2n^e$ principale de l'unité dans $\mathbb{A}[Y]/(Y^n+1)$.*

DÉMONSTRATION. Comme $\omega^n = -1$, ω est bien une racine $2n^{\text{e}}$ de l'unité. Il reste à prouver qu'elle est principale. D'après la remarque en p. 27, il s'agit de montrer que $\omega^t - 1$ n'est pas diviseur de 0, quel que soit le diviseur strict t de $2n$. Puisque n est une puissance de 2, l'entier t divise n, et donc $\omega^t - 1$ divise l'élément $\omega^n - 1 = -2$ qui est supposé inversible, ce qui conclut.

□

EXEMPLE 3. Même lorsque $\mathbb{A}$ est un corps, ces anneaux ne sont en général pas intègres. Par exemple, dans $\mathbb{Z}/3\mathbb{Z}$, la factorisation $Y^4+1 = (Y^2+Y+2)(Y^2+2Y+2)$ exhibe deux polynômes diviseurs de 0 dans $\mathbb{Z}/3\mathbb{Z}[Y]/(Y^4+1)$.

Schönhage-Strassen sur les polynômes

Entrée : F et G de degrés $< n$, où $n = 2^k$, $k > 2$.

Sortie : $FG \bmod X^n + 1$.

1. Soient $d = 2^{\lfloor k/2 \rfloor}$ et $\delta = n/d$. Récrire F et G sous la forme
$$\overline{F}(X,Y) = F_0(X) + F_1(X)Y + \cdots + F_{\delta-1}(X)Y^{\delta-1},$$
$$\overline{G}(X,Y) = G_0(X) + G_1(X)Y + \cdots + G_{\delta-1}(X)Y^{\delta-1},$$
de sorte que les F_i et G_i aient degré $< d$ et
$F(X) = \overline{F}(X, X^d)$, $G(X) = \overline{G}(X, X^d)$.
2. Calculer $\overline{H} := \overline{FG} \bmod Y^\delta + 1$ dans $\mathbb{B}[Y]$ par FFT, où $\mathbb{B} = \mathbb{A}[X]/(X^{2d}+1)$ et les produits dans $\mathbb{B}$ sont effectués par le même algorithme. Pour cela,
   - si k est pair, calculer $\overline{H}(X, X^2Y) \bmod Y^\delta - 1$ avec X^4 comme racine de l'unité, puis reconstruire $\overline{H}$ avec $X^{-2} = -X^{2d-2}$;
   - si k est impair, calculer $\overline{H}(X, XY) \bmod Y^\delta - 1$ avec X^2 comme racine de l'unité, puis reconstruire $\overline{H}$ avec $X^{-1} = -X^{2d-1}$.
3. Renvoyer $H(X, X^d)$.

FIGURE 5. L'algorithme de Schönhage-Strassen sur les polynômes.

5.2. L'algorithme. L'algorithme est décrit en Figure 5. Il calcule le produit FG modulo $X^n + 1$, ce qui permet l'utilisation récursive mais ajoute une petite complication pour le produit par FFT.

EXEMPLE 4. Un exemple jouet permet de mieux comprendre comment se déroule cet algorithme et pourquoi il est correct. La multiplication de polynômes de $\mathbb{Z}/3\mathbb{Z}[X]$ par FFT ne peut pas aller au-delà du degré 2, puisqu'il n'y a pas de racines n^{e} de l'unité d'ordre plus grand. Pour calculer le produit de
$$F = 1 + X^7 \quad \text{et} \quad G = 1 + X + X^6 \quad \text{modulo } X^{16} + 1$$
dans $\mathbb{Z}/3\mathbb{Z}[X]$, l'algorithme introduit $d = 4, \delta = 4$ et
$$\overline{F} = 1 + X^3Y, \quad \overline{G} = 1 + X + X^2Y,$$
et cherche à calculer le produit de ces deux polynômes
$$\overline{H}(X,Y) = \overline{F}(X,Y)\overline{G}(X,Y) \bmod Y^4 + 1,$$
vus comme polynômes en Y à coefficients dans $\mathbb{Z}/3\mathbb{Z}[X]/(X^8+1)$. Ce sera suffisant pour connaître le résultat puisqu'en $Y = X^4$, $Y^4 + 1 = X^{16} + 1$, polynôme modulo lequel l'algorithme calcule le produit.

Pour se ramener à la situation d'une FFT, il suffit de considérer $\overline{H}(X, X^2Y)$: le remplacement de Y par X^2Y dans l'identité ci-dessus montre qu'on calcule alors modulo $X^8Y^4 + 1 = -Y^4 + 1$. On commence donc par calculer une DFT de
$$\overline{F}(X, X^2Y) = 1 + X^5Y, \quad \overline{G}(X, X^2Y) = 1 + X + X^4Y$$
avec X^4 comme racine 4^{e} de l'unité. On obtient les évaluations en $1, X^4, X^8 = -1, X^{12} = -X^4$, à savoir
$$(1 + X^5, 1 - X, 1 - X^5, 1 + X) \quad \text{et} \quad (1 + X + X^4, X, 1 + X - X^4, 2 + X)$$
qui se calculent par une FFT très peu chère, puisque les multiplications par des puissances de X ne demandent pas d'opération arithmétique.

Ensuite, il faut multiplier deux à deux ces valeurs dans $\mathbb{Z}/3\mathbb{Z}[X]/(X^8+1)$ pour obtenir les évaluations de $\overline{H}$. L'algorithme procède récursivement parce que c'est

encore possible (lorsque n devient trop petit, $2d = n$ ce qui mènerait à une boucle infinie ; on a alors recours à une autre méthode, par exemple un produit naïf en ces petits degrés.) Ces calculs récursifs mènent aux valeurs suivantes pour $\overline{H}(X, X^2Y)$ en $1, X^4, X^8 = -1, X^{12} = -X^4$:

$$c_0 = 1 + X^4 + X^5 + X^6, c_1 = X - X^2, c_2 = 1 - X^4 - X^5 - X^6, c_3 = 2 + X^2.$$

L'interpolation par FFT consiste à évaluer le polynôme $c_0 + c_1Y + c_2Y^2 + c_3Y^3$ en les puissances de $X^{-4} = -X^4$, à savoir $1, -X^4, -1, X^4$, ce qui donne

$$1 + X, X^4 + X^5 + X^6, 2X, 0$$

et donc finalement (après multiplication par $1/n = 1/4 = 1$)

$$\overline{H}(X, X^2Y) = 1 + X + X^4(1 + X + X^2)Y + 2XY^2.$$

En utilisant $X^{-2} = -X^6$ pour remplacer Y par YX^{-2}, il vient

$$\overline{H}(X, Y) = 1 + X + X^2(1 + X + X^2)Y + X^5Y^2.$$

Le résultat s'obtient enfin en évaluant en $Y = X^4$:

$$FG = 1 + X + X^6(1 + X + X^2) + X^{13} + X^{12} = 1 + X + X^6 + X^7 + X^8 + X^{13}.$$

En ce qui concerne la complexité, la première étape ne demande pas d'opération arithmétique, et l'évaluation de la dernière étape se fait en complexité linéaire, ainsi que les changements de Y en X^kY avant et après la FFT.

Ensuite, d'après le théorème 3, la multiplication $\overline{FG}$ nécessite :

- $O(d \log d)$ opérations $(+, -)$ dans $\mathbb{B}$, chacune en $O(d)$ opérations dans $\mathbb{A}$;
- $O(d \log d)$ multiplications par une puissance de ω dans $\mathbb{B}$, chacune en $O(d)$ opérations dans $\mathbb{A}$, par simples décalages des indices et éventuels changements de signes ;
- $O(d)$ divisions par d dans $\mathbb{B}$, chacune en $O(d)$ opérations dans $\mathbb{A}$;
- au plus δ produits dans $\mathbb{B}$.

Le coût $C(n)$ de cet algorithme obéit donc à l'inégalité

$$C(n) \leq Kd \log d \times d + \delta C(2d)$$

où K est une constante indépendante de n et de $\mathbb{A}$. En divisant par $n = d\delta$, cette inégalité devient

$$\frac{C(n)}{n} \leq K \frac{d}{\delta} \log d + 2\frac{C(2d)}{2d} \leq K' \log \frac{n}{2} + 2\frac{C(2d)}{2d}$$

pour n assez grand, et une constante K' indépendante de n et de $\mathbb{A}$. Ici $2d = 2^{\lfloor k/2 \rfloor + 1}$, et pour se ramener au théorème diviser pour régner, il sera plus commode de faire apparaître $\lfloor (k+1)/2 \rfloor = \lceil k/2 \rceil$. Pour cela, on considère donc d'abord

$$\frac{C(2n)}{2n} \leq K' \log n + 2\frac{C(2^{\lceil k/2 \rceil + 1})}{2^{\lceil k/2 \rceil + 1}}$$

et on pose $c(k) = C(2^{k+1})/2^{k+1}$, ce qui donne par réécriture

$$c(k) \leq K''k + 2c(\lceil k/2 \rceil),$$

pour une nouvelle constante K''. Le résultat final s'en déduit par application du théorème « diviser pour régner ».

THÉORÈME 4. *Soit $\mathbb{A}$ un anneau dans lequel 2 est inversible (d'inverse connu). On peut multiplier des polynômes de $\mathbb{A}[X]$ de degré au plus n en $O(n \log n \log\log n)$ opérations $(+, -, \times)$ dans $\mathbb{A}$.*

Il est possible d'étendre cette idée au cas où 3 est inversible (FFT triadique), et plus généralement à un anneau quelconque. On obtient alors l'algorithme de complexité $O(n \log n \log\log n)$ mentionné dans l'introduction.

6. Algorithmes pour les entiers

Les algorithmes ainsi que les résultats présentés ci-dessus s'étendent à la multiplication des entiers[3]. Nous allons brièvement présenter cette problématique à travers un exemple.

Soient à multiplier les entiers 2087271 et 1721967, qu'on suppose donnés en base 2,

$$A = 11111110100101100111 \quad \text{et} \quad B = 11010010000110011011111,$$

chacun ayant $D = 21$ chiffres binaires. On peut ramener leur multiplication à un produit de polynômes. Plus exactement, on associe à A et B les polynômes de degré strictement inférieur à D :

$$P = X^{20}+X^{19}+X^{18}+X^{17}+X^{16}+X^{15}+X^{14}+X^{12}+X^{11}+X^{8}+X^{6}+X^{5}+X^{2}+X+1$$

et

$$Q = X^{20} + X^{19} + X^{17} + X^{14} + X^{10} + X^{9} + X^{6} + X^{5} + X^{3} + X^{2} + X + 1.$$

La stratégie est la suivante : on calcule $R = PQ$ dans $\mathbb{Z}[X]$ puis on évalue R en $X = 2$. Pour multiplier P et Q dans $\mathbb{Z}[X]$, il suffit d'effectuer leur produit dans $\mathbb{A}[X] = (\mathbb{Z}/p\mathbb{Z})[X]$, où p est un nombre premier tel que $2D > p > D$. Par le Théorème 4, cette multiplication polynomiale en degré D peut se faire en $O(D \log D \log\log D)$ opérations $(+, -, \times)$ dans $\mathbb{A} = \mathbb{Z}/p\mathbb{Z}$.

Puisque chaque opération de $\mathbb{A}$ nécessite au plus $O(\log^2 D)$ opérations binaires, il s'ensuit que le coût binaire du calcul de R est de $O(D \log^3 D \log\log D)$. Ici $p = 23$ et R (écrit en base 2) vaut $R = X^{40} + 10X^{39} + 10X^{38} + 11X^{37} + 11X^{36} + 11X^{35} + 100X^{34} + 11X^{33} + 11X^{32} + 100X^{31} + 11X^{30} + 100X^{29} + 101X^{28} + 11X^{27} + 101X^{26} + 1000X^{25} + 101X^{24} + 101X^{23} + 1000X^{22} + 1001X^{21} + 1010X^{20} + 1000X^{19} + 111X^{18} + 1000X^{17} + 110X^{16} + 110X^{15} + 110X^{14} + 11X^{13} + 100X^{12} + 110X^{11} + 100X^{10} + 11X^{9} + 100X^{8} + 100X^{7} + 100X^{6} + 11X^{5} + 10X^{4} + 11X^{3} + 11X^{2} + 10X + 1$. Enfin, l'évaluation de R en 2 revient à gérer les retenues et cela peut se faire en un coût négligeable. Ici $AB = R(2) = 11010001001101011110110111011010110110101001$, ou encore, en écriture décimale $AB = 3594211782057$.

Une légère amélioration est possible si l'on choisit pour p un premier de Fourier supérieur à $2D$. Dans notre exemple, on peut prendre $p = 193 = 3 \cdot 2^6 + 1$. En effet, il existe dans $\mathbb{A} = \mathbb{Z}/193\,\mathbb{Z}$ une racine primitive $\omega = 125$, d'ordre $2^6 = 64$, donc supérieur à $2D = 40$. Ce choix mène à un algorithme de complexité binaire $O(D \log^3 D)$.

Une autre approche est de calculer PQ dans $\mathbb{Z}[X]$ par DFT en calculant avec des nombres complexes représentés par des approximations numériques flottantes. Une estimation des erreurs numériques montre qu'il suffit de calculer en précision fixe $O(\log^2 D)$. La complexité binaire est donc toujours $O(D \log^3 D)$ via cette approche.

On peut faire un peu mieux, en remplaçant la base 2 par une base B telle que $B \log B$ soit de l'ordre de $\log D$ et en appliquant l'un des raisonnements précédents ; on peut aboutir ainsi à un algorithme de complexité binaire $O(D \log^2 D)$. Qui plus est, en appelant récursivement l'algorithme pour multiplier les petits entiers, on peut descendre cette complexité à $O(D \log D \log\log D \log\log\log D \ldots)$.

Une meilleure solution est cependant détenue par la version entière, à base de FFT dans des anneaux de type $\mathbb{Z}/(2^{2^\kappa}+1)\mathbb{Z}$, de l'algorithme de Schönhage-Strassen, dont nous nous contentons de mentionner l'existence.

Théorème 5. *On peut multiplier deux entiers de D chiffres binaires en utilisant $O(D \log D \log\log D)$ opérations binaires.*

3. Historiquement, les algorithmes pour la multiplication des entiers ont été introduits *avant* leurs homologues polynomiaux, alors que ces derniers sont souvent bien plus simples à énoncer.

En élaborant largement au-delà de ces techniques de FFT, la complexité du produit entier peut être ramenée à $O\big(n \log n \cdot 2^{O(\log^* n)}\big)$, où $\log^* n$ est le nombre d'itérations du logarithme pour ramener n en dessous de 1. Par exemple (pour la base 2), $\log^* 256 = 4$, car $\log 256 = 8$, $\log 8 = 3$, $1 < \log 3 < 2$ et enfin $\log\log 3 < 1$.

7. Un concept important : les fonctions de multiplication

Un bon nombre des résultats de complexité donnés dans la suite du cours reposent sur la notion de *fonction de multiplication*. Une fonction $\mathsf{M} : \mathbb{N} \to \mathbb{N}$ sera dite fonction de multiplication polynomiale (pour un anneau $\mathbb{A}$) si :

— on peut multiplier les polynômes de $\mathbb{A}[X]$ de degré au plus n en au plus $\mathsf{M}(n)$ opérations dans $\mathbb{A}$;

— M vérifie l'inégalité $\mathsf{M}(n+n') \geq \mathsf{M}(n) + \mathsf{M}(n')$.

Ainsi, on sait d'après ce qui précède que des fonctions de la forme $n^{\log_2 3}$ ou $n \log n$ sont (à des constantes près) des fonctions de multiplication (respectivement pour tous les anneaux ou ceux qui permettent la transformée de Fourier). L'intérêt de cette notion est qu'elle permet d'énoncer des résultats de complexité indépendants du choix de l'algorithme utilisé pour multiplier les polynômes (même si parfois cela mène à des estimations légèrement pessimistes, parce que les algorithmes n'exploitent pas la remarque page 28).

La seconde condition est utilisée pour écarter l'hypothèse d'une fonction qui croît trop lentement (si M est constante, elle ne vérifie pas cette condition). Concrètement, elle permettra par exemple d'établir que dans des algorithmes du type « inversion de série formelle par l'opérateur de Newton » (Chap. 3), le coût total est essentiellement celui de la dernière étape.

De la même manière, on est amené à introduire la notion de *fonction de multiplication* pour les entiers. Une fonction $\mathsf{M}_{\mathbb{Z}} : \mathbb{N} \to \mathbb{N}$ est une fonction de multiplication pour les entiers si :

— on peut multiplier des entiers de n chiffres en $\mathsf{M}_{\mathbb{Z}}(n)$ opérations binaires.

— $\mathsf{M}_{\mathbb{Z}}$ vérifie l'inégalité $\mathsf{M}_{\mathbb{Z}}(n+n') \geq \mathsf{M}_{\mathbb{Z}}(n) + \mathsf{M}_{\mathbb{Z}}(n')$.

Ce concept est très proche de son analogue polynomial et les contextes d'utilisation sont souvent les mêmes : on utilise la seconde condition pour contrôler les coûts de multiplication lors de l'analyse d'algorithmes récursifs.

Exercices

Exercice 13. Soit $\mathbb{A}$ un anneau, soit $a \in \mathbb{A}$ et soit P un polynôme de $\mathbb{A}[X]$, de degré au plus n. On se propose de calculer le polynôme $Q(X) = P(X+a)$.

1. Donner un algorithme direct pour le calcul de Q et estimer sa complexité en termes de n ;
2. Montrer qu'il est possible de calculer $Q(X)$ en n'utilisant que $O\big(\mathsf{M}(n)\log n\big)$ opérations $(+, -, \times)$ dans $\mathbb{A}$. (Un algorithme en $O(\mathsf{M}(n))$ opérations sera donné au Chapitre 3 par somme composée, et une autre solution en $O(\mathsf{M}(n))$ est donnée en Exercice 2 en Annexe.)

Exercice 14. Soit $\mathbb{A}$ un anneau, soit $\kappa, n \in \mathbb{N}$ et soit P un polynôme de $\mathbb{A}[X]$, de degré au plus n.

1. Donner un algorithme direct pour le calcul de $P(X)^\kappa$ et estimer sa complexité en fonction de κ et de n ;
2. Montrer qu'il est possible de calculer $P(X)^\kappa$ en n'utilisant que $O\big(\mathsf{M}(n\kappa)\big)$ opérations $(+, -, \times)$ dans $\mathbb{A}$.

Exercice 15 (Produit court). Pour deux polynômes $A, B \in \mathbb{Q}[X]$ de degré $< n$, le polynôme $AB \bmod X^n$ est appelé *le produit court* de A et B. Le but de cet exercice est de montrer qu'il est possible de calculer plus efficacement le produit court que le produit entier AB. Le gain est d'un facteur constant.

1. Prouver que tel est le cas avec la multiplication polynomiale naïve.

On suppose que l'on dispose d'un algorithme de multiplication polynomiale qui, en degré n, utilise $\mathsf{M}_\alpha(n) = Kn^\alpha$ opérations dans $\mathbb{Q}$, avec $\alpha > 1$ et $K > 0$.

2. Donner un algorithme de type « diviser pour régner » pour le calcul du produit court de A, B.
3. Montrer que la complexité de cet algorithme est $\frac{1}{2^\alpha - 2}\,\mathsf{M}_\alpha(n) + O(n \log n)$. En déduire l'impact sur la multiplication naïve, et celle de Karatsuba.
4. Modifier la conception de l'algorithme précédent, en découpant en deux sous-problèmes de tailles βn et $(1-\beta)n$, avec $1/2 \le \beta \le 1$ laissé en paramètre. Prouver que le nouvel algorithme utilise
$$\frac{\beta^\alpha}{1 - 2(1-\beta)^\alpha} \cdot \mathsf{M}_\alpha(n) + O(n \log n)$$
opérations dans $\mathbb{Q}$.
5. Montrer que le minimum de la fonction $\beta \mapsto \frac{\beta^\alpha}{1-2(1-\beta)^\alpha}$ est atteint en $\beta_{\min} = 1 - \left(\frac{1}{2}\right)^{\frac{1}{\alpha-1}}$ et en déduire qu'on peut calculer le produit court en $C(\alpha)\,\mathsf{M}_\alpha(n) + O(n \log n)$ opérations dans $\mathbb{Q}$, où
$$C(\alpha) = \frac{\left(2^{\frac{1}{\alpha-1}} - 1\right)^\alpha}{2^{\frac{\alpha}{\alpha-1}} - 2}.$$
6. Conclure sur le gain obtenu (à l'aide d'un calcul numérique).

Exercice 16. Soient $P_1, \ldots, P_t$ des polynômes de $\mathbb{A}[X]$, de degrés $d_1, \ldots, d_t$. Montrer que le produit $P_1 \cdots P_t$ peut s'effectuer en $O(\mathsf{M}(n) \log t)$ opérations dans $\mathbb{A}$, où $n = \sum_i d_i$.

Notes

Le mathématicien russe A. N. Kolmogorov avait conjecturé au début des années 1960 qu'il serait impossible de multiplier deux entiers en un coût binaire sous-quadratique. En 1962 cette conjecture fut infirmée par Karatsuba et Ofman [28]. Une généralisation de leur algorithme a été proposée peu de temps après par Toom [39] et Cook [8]. L'algorithme de Toom-Cook a une complexité binaire de $O(n \cdot 32^{\sqrt{\log n}})$ pour multiplier deux entiers de taille binaire n ; ce résultat peut être importé dans le monde polynomial (voir Exercice 7 pour une version plus faible).

L'algorithme FFT a une longue histoire, qui remonte à Gauss [10, 12, 25]. Il s'agit d'un progrès algorithmique très célèbre : Dongarra et Sullivan [15] le placent parmi les dix algorithmes qui ont le plus marqué le développement des sciences de l'ingénieur du 20e siècle. L'article fondateur de Cooley et Tukey [11] est l'un des plus cités en informatique[4]. La variante de la DFT décrite dans ce chapitre, appelée *decimation-in-frequency* en anglais, est due à Gentleman et Sande [21] ; il s'agit d'une version duale (au sens du principe de transposition de Tellegen évoqué au Chapitre 12) de l'algorithme *decimation-in-time* [11]. De nombreux livres [3, 34, 40] et articles [1, 16, 17] permettent de se familiariser avec la myriade de techniques de type FFT ; en particulier, Frigo et Johnson [17] décrivent l'une des implantations

4. Plus de 2000 citations, d'après la base bibliographique *Science Citation Index (SCI ®)*.

les plus efficaces de la transformée de Fourier complexe (appelée FFTW, pour *Fastest Fourier Transform in the West.*)

Une avancée importante a été la découverte de Schönhage et Strassen [37] du résultat équivalent pour la multiplication des nombres entiers. Pendant longtemps, on a cru que cet algorithme ne pourrait présenter qu'un intérêt purement théorique. À ce jour, la plupart des logiciels généralistes de calcul formel disposent d'une multiplication rapide d'entiers qui inclut la FFT. Maple, Mathematica et Sage utilisent la bibliothèque GMP. Il faut noter que cette bibliothèque est le résultat d'un travail de nombreuses années, qui comporte une partie importante de code assembleur consacré à la multiplication sur chacun des processeurs produits dans une période récente.

L'analogue polynomial de l'algorithme de Schönhage-Strassen traité en Section 5 a été suggéré par Nussbaumer [33]. Le cas particulier de la caractéristique 2 est traité par Schönhage [36]. Plus récemment, Cantor et Kaltofen [5] ont donné un algorithme qui multiplie des polynômes de degré n sur une algèbre $\mathbb{A}$ (non nécessairement commutative, ni associative) en $O(n \log n \log\log n)$ opérations dans $\mathbb{A}$.

En ce qui concerne la terminologie, nous suivons Demazure [14] pour la définition des racines principales de l'unité. D'autres auteurs [19] utilisent la troisième propriété du Lemme 1. D'autres enfin [20] les appellent simplement primitives.

La borne de complexité binaire $O\big(n \log n \cdot 2^{O(\log^* n)}\big)$ mentionnée en fin de Section 6, pour multiplier des entiers de n chiffres, a été obtenue récemment par Fürer [13, 18, 19]. Dans [23], Harvey, van der Hoeven et Lecerf ont montré que l'algorithme de Fürer pouvait être un peu modifié pour atteindre un coût binaire en $O\big(n \log n \cdot 16^{\log_2^* n}\big)$. Ils ont ensuite conçu un autre algorithme ayant un coût $O\big(n \log n \cdot 8^{\log_2^* n}\big)$. À ce jour, on ne connaît pas de borne analogue pour la complexité arithmétique de la multiplication polynomiale en degré n en toute généralité. Néanmoins si les coefficients sont dans le corps fini $\mathbb{F}_q$ alors la complexité binaire du produit en degré n est en $O\big(n \log q \log(n \log q) \cdot 8^{\log_2^*(n \log q)}\big)$ [24].

Un problème ouvert est de trouver un algorithme de multiplication polynomiale en complexité $O(n)$, ou de prouver qu'un tel algorithme n'existe pas. Morgenstern [31] a donné une première réponse partielle, en montrant que dans un modèle simplifié où $\mathbb{A} = \mathbb{C}$ et où les seules opérations admises sont les additions et les multiplications par des nombres complexes de module inférieur à 1, au moins $\frac{1}{2} n \log n$ opérations sont *nécessaires* pour calculer une DFT en taille n. Plus récemment, Bürgisser et Lotz [4] ont étendu ce type de borne à la multiplication dans $\mathbb{C}[X]$. Concernant les entiers, Cook et Aanderaa [9] ont prouvé une borne inférieure de la forme $cn \log n/(\log\log n)^2$ pour la multiplication en taille binaire n sur une machine de Turing. Cette borne a été améliorée à $\Omega(n \log n)$ pour un modèle restreint (*multiplication en ligne*) [35].

Malgré la simplicité de leur idée de base, les algorithmes de Karatsuba et de Toom-Cook soulèvent encore des questions délicates ; les travaux récents [2, 7, 30, 41] traitent diverses généralisations (degrés déséquilibrés ou non-puissances, caractéristique positive de l'anneau des scalaires). Par exemple, connaître le nombre minimum de multiplications $\mathsf{sm}(n)$ (resp. $\mathsf{sm}_p(n)$) suffisantes pour multiplier des polynômes de degré donné n sur un anneau arbitraire (resp. sur un anneau de caractéristique p) est un problème important et difficile. Pour les premières valeurs de n, les meilleures bornes connues actuellement, dues à Montgomery [30], sont $\mathsf{sm}(1) \leq 3$, $\mathsf{sm}(2) \leq 6$, $\mathsf{sm}(3) \leq 9$, $\mathsf{sm}(4) \leq 13$, $\mathsf{sm}(5) \leq 17$ et $\mathsf{sm}(6) \leq 22$. L'optimalité de ces bornes est un problème difficile. Pour $\mathsf{sm}_p(n)$, on sait par exemple que $\mathsf{sm}_2(1) = 3, \mathsf{sm}_2(2) = 6, \mathsf{sm}_2(3) = 9$ [27] et que $\mathsf{sm}_7(4) = 10$, $\mathsf{sm}_5(4) \leq 11$ [6].

Une autre question importante est de savoir si les n coefficients du *produit court* $FG \bmod X^n$ de deux polynômes F et G de degré au plus n peuvent être calculés

plus efficacement que l'ensemble des $2n$ coefficients du produit FG. L'exercice 15, inspiré d'un résultat de Mulders [32], apporte une réponse positive dans le cas où l'algorithme de multiplication est celui de Karatsuba. Par contre, on ne connaît aucun élément de réponse dans le cas où la multiplication polynomiale repose sur la DFT. Une question liée, le calcul du *produit médian* [22], sera abordée au Chapitre 12.

Une faiblesse de l'algorithme DFT est son comportement quasi-linéaire *par morceaux* (voir la Figure 1), dû aux sauts de complexité entre deux puissances successives de 2. Récemment, van der Hoeven [26] a donné un nouvel algorithme, appelé TFT (de *Truncated Fourier Transform* en anglais), qui coïncide avec la DFT si n est une puissance de 2 et a une courbe de complexité « plus lisse » que la DFT.

Le produit de t polynômes de degrés arbitraires $d_1, \ldots, d_t$ peut s'effectuer [38] en $O(\mathsf{M}(n)(1 + H(d_1, \ldots, d_t)))$ opérations, où $H(d_1, \ldots, d_t) = -\sum_i \frac{d_i}{n} \cdot \log \frac{d_i}{n}$ est l'*entropie* du vecteur $(d_1/n, \ldots, d_t/n)$. La multiplication des polynômes à plusieurs variables peut se ramener à une variable grâce à une technique connue sous le nom de *substitution de Kronecker* [29] (voir aussi l'exercice 9 du Chapitre 5).

Bibliographie

[1] Bernstein, Daniel J. (2001). *Multidigit multiplication for mathematicians*. URL : http://cr.yp.to/papers.html (visible en 2001).

[2] Bodrato, Marco et Alberto Zanoni (2007). « Integer and polynomial multiplication : towards optimal Toom-Cook matrices ». In : *ISSAC'07 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Dongming Wang. ACM Press, p. 17–24. DOI : 10.1145/1277548.1277552.

[3] Brigham, E. Oran (1974). *The fast Fourier transform*. Prentice-Hall, xiii+252 pages.

[4] Bürgisser, Peter et Martin Lotz (2004). « Lower bounds on the bounded coefficient complexity of bilinear maps ». In : *Journal of the Association for Computing Machinery*, vol. 51, n°3, p. 464–482. DOI : 10.1145/990308.990311.

[5] Cantor, D. G. et E. Kaltofen (1991). « On Fast Multiplication of Polynomials over Arbitrary Algebras ». In : *Acta Informatica*, vol. 28, n°7, p. 693–701.

[6] Cenk, M., C. K. Koc et F. Ozbudak (2009). « Polynomial Multiplication over Finite Fields Using Field Extensions and Interpolation ». In : *ARITH'09 : IEEE Symposium on Computer Arithmetic*. IEEE Computer Society, p. 84–91. DOI : 10.1109/ARITH.2009.11.

[7] Chung, Jaewook et M. Anwar Hasan (2007). « Asymmetric Squaring Formulae ». In : *ARITH'07 : IEEE Symposium on Computer Arithmetic*. IEEE, p. 113–122. DOI : 10.1109/ARITH.2007.11.

[8] Cook, S. (1966). « On the minimum computation time of functions ». Thèse de doct. Harvard University.

[9] Cook, S. A. et S. O. Aanderaa (1969). « On the minimum computation time of functions ». In : *Transactions of the American Mathematical Society*, vol. 142, p. 291–314.

[10] Cooley, James W. (1990). « How the FFT gained acceptance ». In : *A history of scientific computing*. ACM Press Hist. Ser. P : ACM, p. 133–140.

[11] Cooley, James W. et John W. Tukey (1965). « An algorithm for the machine calculation of complex Fourier series ». In : *Mathematics of Computation*, vol. 19, p. 297–301.

[12] — (1993). « On the Origin and Publication of the FFT Paper ». In : *Current Contents*, vol. 33, n°51–52, p. 8–9. URL : www.garfield.library.upenn.edu/classics1993/%20A1993MJ84500001.pdf.

[13] De, A., P. P. Kurur, C. Saha et R. Saptharishi (2008). « Fast integer multiplication using modular arithmetic ». In : *STOC'08 : ACM Symposium on Theory of Computing*. Victoria, British Columbia, Canada : ACM, p. 499–506. DOI : 10.1145/1374376.1374447.

[14] Demazure, Michel (1997). *Cours d'algèbre*. Nouvelle Bibliothèque Mathématique n°1. Cassini, Paris, xviii+302 pages.

[15] Dongarra, J. et F. Sullivan (2000). « Top ten algorithms ». In : *Computing in Science & Engineering*, vol. 2, n°1, p. 22–23.

[16] Duhamel, P. et M. Vetterli (1990). « Fast Fourier transforms : a tutorial review and a state of the art ». In : *Signal Processing. An Interdisciplinary Journal*, vol. 19, n°4, p. 259–299.

[17] Frigo, M. et S. G Johnson (2005). « The Design and Implementation of FFTW3 ». In : *Proceedings of the IEEE*, vol. 93, n°2, p. 216–231.

[18] Fürer, Martin (2007). « Faster integer multiplication ». In : *STOC'07 : ACM Symposium on Theory of Computing*. New York : ACM, p. 57–66.

[19] — (2009). « Faster integer multiplication ». In : *SIAM Journal on Computing*, vol. 39, n°3, p. 979–1005. DOI : 10.1137/070711761.

[20] Gathen, J. von zur et J. Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press.

[21] Gentleman, W. M. et G. Sande (1966). « Fast Fourier Transforms : for fun and profit ». In : *AFIPS'66*. San Francisco, California : ACM, p. 563–578. DOI : 10.1145/1464291.1464352.

[22] Hanrot, Guillaume, Michel Quercia et Paul Zimmermann (2004). « The Middle Product Algorithm I ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 14, n°6, p. 415–438.

[23] Harvey, David, Joris van der Hoeven et Grégoire Lecerf (2014a). « Even faster integer multiplication ». http://arxiv.org/abs/1407.3360.

[24] — (2014b). « Faster polynomial multiplication over finite fields ». http://arxiv.org/abs/1407.3361.

[25] Heideman, Michael T., Don H. Johnson et C. Sidney Burrus (1985). « Gauss and the history of the fast Fourier transform ». In : *Archive for History of Exact Sciences*, vol. 34, n°3, p. 265–277.

[26] Hoeven, Joris van der (2004). « The truncated Fourier transform and applications ». In : *ISSAC'04 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 290–296.

[27] Kaminski, Michael (1985). « A lower bound for polynomial multiplication ». In : *Theoretical Computer Science*, vol. 40, n°2-3, p. 319–322. DOI : 10.1016/0304-3975(85)90174-4.

[28] Karatsuba, A. et Y. Ofman (1963). « Multiplication of multidigit numbers on automata ». In : *Soviet Physics Doklady*, vol. 7, p. 595–596.

[29] Moenck, Robert T. (1976). « Another polynomial homomorphism ». In : *Acta Informatica*, vol. 6, n°2, p. 153–169.

[30] Montgomery, Peter L. (2005). « Five, Six, and Seven-Term Karatsuba-Like Formulae ». In : *IEEE Transactions on Computers*, vol. 54, n°3, p. 362–369. DOI : 10.1109/TC.2005.49.

[31] Morgenstern, Jacques (1973). « Note on a lower bound of the linear complexity of the fast Fourier transform ». In : *Journal of the Association for Computing Machinery*, vol. 20, p. 305–306.

[32] Mulders, Thom (2000). « On Short Multiplications and Divisions ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, n°1, p. 69–88.

[33] Nussbaumer, Henri J. (1980). « Fast polynomial transform algorithms for digital convolution ». In : *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, n°2, p. 205–215.

[34] — (1981). *Fast Fourier transform and convolution algorithms*. Vol. 2. Springer Series in Information Sciences. Springer-Verlag, x+248 pages.

[35] Paterson, M. S., M. J. Fischer et A. R. Meyer (1974). « An improved overlap argument for on-line multiplication ». In : *Complexity of computation*. AMS, p. 97–111.

[36] Schönhage, A. (1976/77). « Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2 ». In : *Acta Informatica*, vol. 7, n°4, p. 395–398.

[37] Schönhage, A. et V. Strassen (1971). « Schnelle Multiplikation großer Zahlen ». In : *Computing*, vol. 7, p. 281–292.

[38] Strassen, V. (1983). « The computational complexity of continued fractions ». In : *SIAM Journal on Computing*, vol. 12, n°1, p. 1–27.

[39] Toom, A. L. (1963). « The complexity of a scheme of functional elements simulating the multiplication of integers ». In : *Doklady Akademii Nauk SSSR*, vol. 150, p. 496–498.

[40] Van Loan, Charles (1992). *Computational frameworks for the fast Fourier transform*. Vol. 10. Frontiers in Applied Mathematics. SIAM, xiv+273 pages.

[41] Weimerskirch, André et Christof Paar (2006). *Generalizations of the Karatsuba Algorithm for Efficient Implementations*. Cryptology ePrint Archive : 2006/224. (Visible en 2006).

CHAPITRE 3

Calculs rapides sur les séries

Résumé

La multiplication rapide de polynômes vue au Chapitre 2 permet des calculs efficaces comme l'inversion, l'exponentiation ou la prise de logarithme d'une série. Ces opérations sont effectuées grâce à une version formelle de la méthode de Newton. La composition de séries est en général plus coûteuse que le produit, mais peut aussi tirer parti d'une multiplication rapide.

Ce chapitre porte sur le calcul des premiers termes de développements en séries. Pour fixer les notations, N sera utilisé pour représenter le nombre de termes à calculer (en comptant les coefficients nuls), et « série » sera employé pour « série formelle tronquée à l'ordre N ». Ainsi, « calculer une série » voudra toujours dire en calculer les N premiers termes. Comme dans le Chapitre 2, $\mathsf{M}(N)$ dénote une borne supérieure sur le nombre d'opérations arithmétiques nécessaires pour multiplier deux polynômes de degré inférieur à N, borne qui est supposée vérifier l'inégalité $\mathsf{M}(n+n') \geq \mathsf{M}(n) + \mathsf{M}(n')$. L'efficacité des algorithmes sera mesurée par leur complexité arithmétique.

L'opérateur de Newton est très largement utilisé en calcul numérique, pour trouver des solutions approchées d'équations. C'est un processus itératif, qui consiste à chaque étape à remplacer la fonction dont on cherche un zéro par son linéarisé au voisinage de la dernière approximation trouvée. Ainsi, si Φ est une fonction C^1 de $\mathbb{R}$ dans $\mathbb{R}$, pour résoudre $\Phi(y) = 0$ dans $\mathbb{R}$, on choisit $y_0 \in \mathbb{R}$ et on itère

$$y_{k+1} = \mathcal{N}(y_k) = y_k - \frac{\Phi(y_k)}{\Phi'(y_k)}. \tag{1}$$

Cette itération est représentée en figure 1.

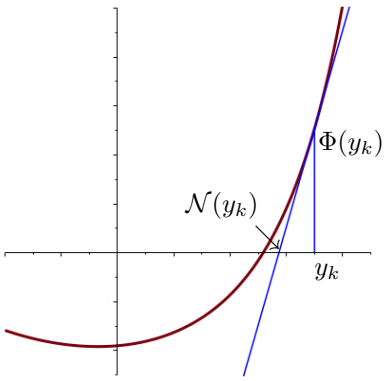


FIGURE 1. Opérateur de Newton de $\mathbb{R}$ dans $\mathbb{R}$.

La solution serait exacte si Φ était une fonction linéaire ; dans le cas général, si y_0 est bien choisi (par exemple assez proche d'une racine simple), l'itération est bien définie ($\Phi'(y_k)$ n'est jamais nul) et la suite y_k converge vers une limite y_∞. En outre, à l'itération k la distance à la limite est de l'ordre du carré de cette distance à l'itération $k-1$. On parle dans ce cas de convergence *quadratique*. Dans ces conditions, pour k suffisamment grand, le nombre de décimales correctes est approximativement *doublé* à chaque itération. De nombreuses généralisations de cette méthode existent, d'abord à des systèmes (la division par l'inverse est alors remplacée par une multiplication à gauche par l'inverse de la matrice jacobienne), puis à des espaces de Banach (où la matrice jacobienne est remplacée par la différentielle de l'opérateur).

Dans le cadre formel, la fonction Φ est remplacée par une application de l'algèbre des séries formelles (définie en Section 1) dans elle-même. Cet algèbre n'est pas un espace normé (donc pas un espace de Banach), mais dans bien des cas, l'itération de Newton converge (pour la métrique de la valuation, voir plus loin), et le nombre de *coefficients* corrects est doublé à chaque itération, ce qui mène à une bonne complexité. La composition des séries reste une opération plus coûteuse, à laquelle est consacrée la fin de ce chapitre.

1. Séries formelles

Dans cette section et la suivante les séries ont leurs coefficients dans un anneau qui n'est pas nécessairement commutatif, sauf lorsque c'est explicitement indiqué. À partir de la section 3, l'utilisation de la formule de Taylor contraint à se restreindre au cas commutatif.

Définition. Si $\mathbb{A}$ est un anneau, on note $\mathbb{A}[[X]]$ l'ensemble des séries formelles sur $\mathbb{A}$. Ses éléments sont des suites $(f_i)_{i\in\mathbb{N}}$ de $\mathbb{A}$, notées

$$F(X) = \sum_{i\geq 0} f_i X^i.$$

Le coefficient f_i est appelé le i^{e} coefficient de $F(X)$, le coefficient f_0 est appelé terme constant de $F(X)$, et parfois noté $F(0)$. L'indice du premier coefficient non-nul est appelé la *valuation* de F et noté $\operatorname{val} F$. Par convention, $\operatorname{val} 0 = \infty$.

Les opérations de $\mathbb{A}[[X]]$ sont l'addition des suites et une multiplication (appelée parfois produit de Cauchy) qui généralise la multiplication des polynômes :

$$\sum_{i\geq 0} f_i X^i \times \sum_{i\geq 0} g_i X^i = \sum_{i\geq 0} h_i X^i, \qquad \text{avec} \quad h_i = \sum_{j+k=i} f_j g_k,$$

la dernière somme étant finie.

Exemple 1. La série formelle

$$1 = 1\cdot X^0 + 0\cdot X + 0\cdot X^2 + \cdots,$$

où 1 est l'unité de $\mathbb{A}$, est élément neutre pour la multiplication de $\mathbb{A}[[X]]$. La formule donnant les coefficients du produit se réduit alors à un terme.

Exemple 2. Si $F = 1 + X$ et $G = 1 - X + X^2 - X^3 + \cdots$, alors le produit vaut $H = FG = 1$: le coefficient de X^n dans H vaut $1 - 1 = 0$ pour $n > 0$ et 1 sinon. La série G est donc l'inverse de F pour la multiplication, que l'on peut noter $(1+X)^{-1}$.

Métrique. Lorsque les coefficients sont des nombres complexes, la question de la convergence des séries entières représentées par ces séries formelles ne se posera pas pour les algorithmes considérés dans ce chapitre. En revanche, quel que soit l'anneau de coefficients $\mathbb{A}$, il est possible de définir une distance entre deux séries F et G par $d(F,G) = 2^{-\operatorname{val}(F-G)}$.

EXERCICE 1. Vérifier que la fonction d définie ci-dessus est bien une distance.

Muni de cette distance, l'ensemble des séries formelles forme un espace métrique *complet* (les suites de Cauchy convergent). En effet, dire qu'une suite (S_n) de séries est de Cauchy signifie qu'étant donné $k \in \mathbb{N}$, il existe un entier K tel que pour tous $m, n \geq K$, on ait $d(S_m, S_n) < 2^{-k}$, autrement dit après l'indice K, les k premiers termes des S_n sont fixés, ce sont ceux de la série limite.

Composition. Si F et G sont deux séries formelles, avec terme constant $G(0) = 0$, on définit la composition comme

$$F(G(X)) = f_0 + f_1 G(X) + f_2 G(X)^2 + \cdots$$

Les points de suspension signifient que l'on considère la limite des sommes H_n obtenues en arrêtant la somme au terme $f_n G(X)^n$. Cette limite existe puisque pour $m \geq n$, la distance obéit à $d(H_m, H_n) \leq 2^{-n}$, ce qui fait de $(H_n)_n$ une suite de Cauchy.

Inverse. Si F est de la forme $F = 1 + XG$ avec $G \in \mathbb{A}[[X]]$, alors F est inversible et son inverse est donné par

$$1 - XG + X^2 G^2 - \cdots,$$

composition de $(1+X)^{-1}$ par GX. La preuve découle de cette composition : $(1 + XG)(1 - XG + \dots) = H(XG)$ où $H = (1+X)(1+X)^{-1} = 1$.

Si $a = F(0)$ est inversible, la série F se récrit $F = a(1 + XG)$ avec $G = a^{-1}(F - a)/X$, donc F est inversible, d'inverse $(1 + XG)^{-1}a^{-1} = a^{-1} - Ga^{-1}X + G^2 a^{-1} X^2 + \cdots$.

Ces ingrédients mènent à la structure d'anneau de l'ensemble de séries formelles.

PROPOSITION 1. *L'ensemble $\mathbb{A}[[X]]$ des séries formelles à coefficients dans $\mathbb{A}$ est un anneau, qui est commutatif si $\mathbb{A}$ l'est. Ses éléments inversibles sont les séries de terme constant inversible.*

DÉMONSTRATION. L'addition est commutative comme celle de $\mathbb{A}$. L'associativité et la distributivité du produit sont obtenues comme pour les polynômes. L'unité pour le produit est la série 1. La première partie de la proposition est donc prouvée.

Si $F = \sum f_i X^i \in \mathbb{A}[[X]]$ est inversible, et $G = \sum g_i X^i$ est son inverse, alors l'extraction du coefficient de X^0 dans l'identité $FG = 1$ donne $f_0 g_0 = 1$ ce qui prouve qu'une série inversible a un terme constant inversible. La réciproque a été présentée ci-dessus. □

Séries à plusieurs variables. Comme $\mathbb{A}[[X]]$ est un anneau, il peut être utilisé comme anneau de base pour définir des séries formelles en une autre variable Y, ce qui définit de la même manière l'anneau des séries formelles à deux variables $\mathbb{A}[[X]][[Y]]$, noté aussi $\mathbb{A}[[X,Y]]$.

Dérivation. La dérivée d'une série est définie formellement coefficient par coefficient via l'identité

$$\left(\sum_{i\geq 0} f_i X^i\right)' = \sum_{i\geq 0} (i+1) f_{i+1} X^i.$$

Les relations habituelles $(F+G)' = F' + G'$ et $(FG)' = F'G + FG'$ sont prouvées comme pour les polynômes. Dans le cas de séries en plusieurs variables, on utilise la notation des dérivées partielles $\partial F/\partial X$ pour désigner la dérivée par rapport à la variable X.

Troncatures. Algorithmiquement, on ne manipule pas de série à précision « infinie », mais seulement des troncatures, c'est-à-dire un certain nombre des premiers termes. Les séries tronquées deviennent alors de simples polynômes, pour lesquels on dispose en particulier d'algorithmes rapides de multiplication. Étant donnés les N premiers termes d'une série F, l'objectif de ce chapitre est de donner des algorithmes permettant de calculer efficacement les N premiers termes d'autres séries définies à partir de F.

Pour étendre la notation utilisée avec les polynômes, étant donnée la série

$$S = \sum_{i\geq 0} a_i X^i,$$

on notera $S \bmod X^N$ le polynôme

$$S \bmod X^N := \sum_{0\leq i<N} a_i X^i.$$

On notera $S = f + O(X^N)$ si les séries ou polynômes S et f coïncident jusqu'au terme de degré $N-1$, et même plus généralement $S = O(T^k)$ si $S = O(X^{k\,\mathrm{val}(T)})$.

Formule de Taylor. Dans le cas où l'anneau de coefficients $\mathbb{A}$ est commutatif, le lien entre la composition et la dérivation est donné par la formule de Taylor. Si F, G, H sont trois séries formelles, avec $G(0) = H(0) = 0$, et les entiers $2, 3, \ldots, k-1$ sont inversibles dans $\mathbb{A}$, alors

$$F(G+H) = F(G) + F'(G)H + F''(G)\frac{H^2}{2!} + \cdots + O(H^k).$$

La formule est classique pour les polynômes, et les coefficients de X^N dans les deux membres de la formule sont les mêmes que ceux de l'identité entre polynômes obtenue en considérant les troncatures de F, G et H modulo X^{N+1}, ce qui permet de conclure.

Pour cette identité, la commutativité de $\mathbb{A}$ est cruciale comme le montre l'exemple de $F = X^2$: on a alors $(G+H)^2 = G^2+GH+HG+H^2$ et la partie linéaire $GH+HG$ est en général différente de $2GH = F'(G)H$.

Intégration. Il s'agit de l'opération inverse de la dérivation. On suppose que les entiers sont inversibles dans $\mathbb{A}$ ($\mathbb{A}$ est une $\mathbb{Q}$-algèbre) et on définit alors

$$\int \sum_{i\geq 0} f_i X^i = \sum_{i\geq 0} f_i \frac{X^{i+1}}{i+1}.$$

Exemple 3. La série $\log(1+X)$ vaut

$$\log(1+X) = \int (1+X)^{-1} = X - \frac{1}{2}X^2 + \frac{1}{3}X^3 + \cdots$$

Inverse multiplicatif de séries formelles

Entrée : un entier $N > 0$, $F \bmod X^N$ une série tronquée ;

Sortie : $F^{-1} \bmod X^N$.

Si $N = 1$, alors renvoyer f_0^{-1}, où $f_0 = F(0)$.

Sinon,

1. Calculer récursivement l'inverse G de $F \bmod X^{\lceil N/2 \rceil}$;
2. Renvoyer $G + (1 - GF)G \bmod X^N$.

FIGURE 2. Inverse de série par itération de Newton.

2. La méthode de Newton pour le calcul d'inverses

Partant de la fonction $\Phi(y) = 1/y - f$, dont l'unique racine est l'inverse $1/f$, l'itération (1) devient

$$\mathcal{N}(g_k) = g_k + g_k^2(1/g_k - f) = g_k + g_k(1 - g_k f).$$

Cette itération, ou des variantes, permet le calcul d'inverses dans plusieurs contextes.

2.1. Convergence quadratique pour l'inverse d'une série formelle.

LEMME 1. *Soit $F \in \mathbb{A}[[X]]$ une série formelle de terme constant inversible et G une série telle que $G - F^{-1} = O(X^n)$ $(n \geq 1)$, alors la série*

$$\mathcal{N}(G) = G + (1 - GF)G \tag{2}$$

vérifie $\mathcal{N}(G) - F^{-1} = O(X^{2n})$.

DÉMONSTRATION. Par hypothèse, on peut définir $H \in \mathbb{A}[[X]]$ par $1 - GF = X^n H$. Il suffit alors de récrire $F = G^{-1}(1 - X^n H)$ et d'inverser :

$$F^{-1} = (1 + X^n H + O(X^{2n}))G = G + X^n HG + O(X^{2n})G = \mathcal{N}(G) + O(X^{2n}).$$

□

2.2. Algorithme.

LEMME 2. *L'algorithme d'inversion de la figure 2 est correct.*

DÉMONSTRATION. La preuve est une récurrence sur les entiers. Pour $N = 1$ la propriété est claire. Si la propriété est vraie pour $1 \leq N < k$, alors elle l'est pour $1 \leq \lceil N/2 \rceil < k$ donc le résultat de l'étape 1 vaut $F \bmod X^{\lceil N/2 \rceil}$. Alors, d'après le lemme 1, $G + (1 - GF)G = F^{-1} \bmod X^{2\lceil N/2 \rceil}$ et la troncature modulo X^N est donc bien correcte. □

2.3. Complexité. Chaque itération coûte essentiellement deux multiplications, et l'ensemble est donc effectué en $O(\mathsf{M}(N))$ opérations d'après le lemme « diviser pour régner ». Une estimation plus précise est donnée par la proposition suivante.

PROPOSITION 2. *Soit F dans $\mathbb{A}[[X]]$ avec $F(0)$ inversible et $N \geq 1$. On peut calculer l'inverse de F modulo X^N en $3\mathsf{M}(N) + O(N)$ opérations dans $\mathbb{A}$ lorsque N est une puissance de 2.*

DÉMONSTRATION. L'étape 2 de l'algorithme demande deux multiplications : comme G est de degré inférieur à $\lceil N/2 \rceil$, celle de G par F coûte $2M(\lceil N/2 \rceil)$ et produit un résultat de la forme $X^{\lceil N/2 \rceil} R_N$ avec R_N de degré inférieur à $\lceil N/2 \rceil$. Sa

multiplication par G coûte alors $M(\lceil N/2 \rceil)$ opérations. S'ajoutent à ces multiplications $\lambda \cdot N$ opérations supplémentaires (additions, ...), λ étant une constante que l'on ne va pas déterminer. Notons $\mathsf{C}(N)$ le coût du calcul modulo N ; on a alors

$$\mathsf{C}(N) \leq \mathsf{C}(\lceil N/2 \rceil) + 3\mathsf{M}(\lceil N/2 \rceil) + \lambda N.$$

L'application du lemme « diviser pour régner » donne alors

$$\mathsf{C}(N) \leq 3\mathsf{M}(N) + 2\lambda N$$

pour N puissance de 2 en utilisant l'inégalité $\mathsf{M}(d) \leq \frac{1}{2}\mathsf{M}(2d)$. □

Comme d'habitude, lorsque N n'est pas une puissance de 2, la complexité est bornée par celle de la puissance de 2 immédiatement supérieure, ce qui rajoute au plus un facteur 2.

Optimisations. Le coût de la Proposition 2 peut être encore amélioré. Le produit GF est de la forme $1+X^{\lceil N/2 \rceil}R_N$. Il s'agit donc de calculer le produit d'un polynôme de degré $N/2$ par un polynôme de degré N, alors que l'on connaît à l'avance les $N/2$ premiers coefficients du produit. Dans cette situation, il est possible d'abaisser le coût du produit de $\mathsf{M}(N)$ à $\mathsf{M}(N/2)$, en utilisant un produit médian et des techniques non triviales de « transposition » de code abordées au Chapitre 12.

2.4. Division de séries. Le quotient H/F où H et F sont des séries et F est inversible, peut être obtenu comme le produit $H \times F^{-1}$. Ce calcul peut être amélioré d'un facteur constant. Il suffit pour cela de remplacer la dernière itération $G+(1-GF)G$ par $Y+(H-YF)G$ où $Y := HG \bmod X^{\lceil N/2 \rceil}$. Cette astuce remplace un produit de la forme $(N/2, N)$ (pour GF), un produit de la forme $(N/2, N/2)$ (pour $(1-GF)G$) et un produit final (N, N) par deux produits $(N/2, N/2)$ (pour Y et $(H-YF)G$) et un produit $(N/2, N)$ pour calculer YF. L'économie est donc la différence $\mathsf{M}(N) - \mathsf{M}(N/2)$.

2.5. Application à la division euclidienne. L'algorithme de la figure 2 est l'élément clé de la division euclidienne rapide, traitée au Chapitre 4.

2.6. Application au logarithme. Si $F \in \mathbb{A}[[X]]$ est telle que $F(0) = 0$, on définit la série $\log(1+F)$ par composition avec $\log(1+X)$. Pour calculer cette série lorsque $\mathbb{A}$ est commutatif, il suffit d'utiliser l'identité

$$\log(1+F) = \int \frac{F'}{1+F}.$$

Exercice 2. Prouver cette identité à partir des définitions de log, $\int$ et de la dérivation données plus haut.

Le calcul demande une division de séries, une dérivation et une intégration, mais ces deux dernières opérations sont de complexité linéaire. Le bilan est donc une complexité en $O(\mathsf{M}(N))$.

2.7. Inverse de matrices. L'anneau $\mathbb{A}$ n'étant pas supposé commutatif, il est possible de traiter directement des matrices de séries comme des séries de matrices : le lemme 1 et l'algorithme de la figure 2 s'appliquent. La complexité de cet algorithme s'exprime à l'aide de la fonction $\mathsf{MM}(k, N)$ complexité du produit de matrices $k \times k$ de polynômes de degré au plus N. Le Corollaire 1 du Chapitre 5 montre que $\mathsf{MM}(k, N) = O(k^\theta N + k^2\mathsf{M}(N))$.

Proposition 3. *Soit A une matrice $k \times k$ à coefficients dans $\mathbb{A}[[X]]$, avec $A(0)$ inversible. La matrice $A^{-1} \bmod X^N$ peut être calculée en $O(\mathsf{MM}(k, N))$ opérations dans $\mathbb{A}$.*

DÉMONSTRATION. Comme pour l'inverse (Figure 2), le coût est dominé à une constante près par celui du calcul de $G+(I-GF)G \bmod X^N$: la complexité vérifie l'inégalité $C(N) \leq C(\lceil N/2 \rceil) + 2\mathsf{MM}(k,N)$ et le résultat se déduit de l'hypothèse $\mathsf{MM}(k,N) \leq \frac{1}{2}\mathsf{MM}(k,2N)$. □

3. Itération de Newton formelle et applications

Dans toute la suite de ce chapitre, l'anneau $\mathbb{A}$ est supposé commutatif.

3.1. Un résultat général.

THÉORÈME 1 (Itération de Newton sur les séries). *Soit $\Phi \in \mathbb{A}[[X,Y]]$ une série à deux variables en X et Y, telle que $\Phi(0,0) = 0$ et $\frac{\partial \Phi}{\partial Y}(0,0)$ est inversible dans $\mathbb{A}$. Il existe alors une unique série $S \in \mathbb{A}[[X]]$, telle que $\Phi(X,S) = 0$ et $S(0) = 0$. Si F est une série telle que $S - F = O(X^n)$ $(n \geq 1)$, alors*

$$\mathcal{N}(F) = F - \frac{\Phi(X,F)}{\frac{\partial \Phi}{\partial Y}(X,F)}$$

vérifie $S - \mathcal{N}(F) = O(X^{2n})$.

Ce résultat est un résultat local en $(0,0)$. Il exprime que si le point (0,0) est solution, il s'étend en une courbe solution. Par translation, d'autres situations où la solution n'est pas en $(0,0)$ s'y ramènent. La première partie est une version du théorème des fonctions implicites pour les séries formelles.

DÉMONSTRATION. D'abord il faut observer que l'itération est bien définie : la composition de Φ et de $\frac{\partial \Phi}{\partial Y}$ avec F sont possibles parce que $F(0) = S(0) = 0$. Pour la même raison, $\frac{\partial \Phi}{\partial Y}(X,F)$ est inversible puisque son terme constant $\frac{\partial \Phi}{\partial Y}(0,0)$ est inversible. De ce fait, on déduit aussi $\operatorname{val}(\mathcal{N}(F) - F) = \operatorname{val}(\Phi(X,F))$.

Ensuite, la preuve se déroule en deux temps : l'itération est d'abord utilisée pour montrer l'existence d'une série solution S, puis on montre l'unicité en même temps que la vitesse de convergence vers S.

La valuation de $\Phi(X,F)$ est doublée par l'itération : la formule de Taylor donne

$$\begin{aligned} \Phi(X,\mathcal{N}(F)) &= \Phi(X,F) + \frac{\partial \Phi}{\partial Y}(X,F)(\mathcal{N}(F) - F) + O((\mathcal{N}(F) - F)^2) \\ &= O(\Phi(X,F)^2). \end{aligned} \tag{3}$$

La suite définie par $S_0 = 0$ et $S_{k+1} = \mathcal{N}(S)$ vérifie donc $\operatorname{val}(S_{k+2} - S_{k+1}) = \operatorname{val}(\Phi(X,S_{k+1})) \geq 2\operatorname{val}(S_{k+1} - S_k)$. Cette suite est donc de Cauchy et sa limite S existe et vérifie $\Phi(X,S) = \lim(\Phi(X,S_k)) = 0$.

La vitesse de convergence vers S vient encore de la formule de Taylor. En effet, l'égalité

$$\Phi(X,F) = \Phi(X,S) + \frac{\partial \Phi}{\partial Y}(X,F)(S - F) + O((S - F)^2) \tag{4}$$

donne $S - F = O(\Phi(X,F))$. On en déduit $S - \mathcal{N}(F) = O(\Phi(X,\mathcal{N}(F))) = O(X^{2n})$ au vu des hypothèses et de (3). Cette égalité donne aussi l'unicité de la solution : si F est une autre solution avec $F(0) = 0$, elle doit vérifier $S - F = O((S-F)^2)$, ce qui entraîne $\operatorname{val}(S - F) = \infty$ et donc $F = S$. □

3.2. Algorithme.

LEMME 3. *L'algorithme de résolution détaillé en figure 3 est correct.*

DÉMONSTRATION. Pour $N = 1$, les valeurs renvoyées sont les termes constants de S et T. Si la propriété est vraie pour $1 \leq k < N$, alors l'étape 1 renvoie $T \bmod X^{\lceil \lceil N/2 \rceil / 2 \rceil}$ et $S \bmod X^{\lceil N/2 \rceil}$. D'après le lemme 1, l'étape 2 calcule $T \bmod X^{\lceil N/2 \rceil}$. L'étape 3 calcule alors $S \bmod X^N$: les formules (3) et (4) montrent que

Résolution d'équation en série

Entrée : un entier $N > 0$, $\Phi \bmod (X^N, Y^N)$ une série tronquée avec $\Phi(0,0) = 0$ et $(\partial\Phi/\partial Y)(0,0)$ inversible.

Sortie : $S \bmod X^N$, $T \bmod X^{\lceil N/2 \rceil}$, telles que $S(0) = 0$, $\Phi(X, S) = 0$, $T = (\partial\Phi/\partial Y(X, S))^{-1}$.

Si $N = 1$, alors renvoyer 0, $((\partial\Phi/\partial Y)(0,0))^{-1}$.

Sinon,

1. Calculer récursivement $F := S \bmod X^{\lceil N/2 \rceil}$ et $G := T \bmod X^{\lceil \lceil N/2 \rceil /2 \rceil}$;
2. Calculer $G := G + (1 - G\frac{\partial\Phi}{\partial Y}(X, F))G \bmod X^{\lceil N/2 \rceil}$;
3. Calculer $F := F - G\Phi(X, F) \bmod X^N$;
4. Renvoyer F, G.

FIGURE 3. Résolution de $\Phi(X, Y) = 0$ par itération de Newton.

seuls les $\lceil N/2 \rceil$ premiers coefficients de $\partial\Phi/\partial Y(X, S)$ sont utiles pour doubler la précision. □

3.3. Applications. Un exemple d'application est fourni par l'inverse de la section précédente avec $\Phi(X, Y) = f - 1/(f(0)^{-1} + Y)$ (lorsque $\mathbb{A}$ est commutatif).

La complexité de l'algorithme de la figure 3 dépend en particulier de celle du calcul de Φ et de $\partial\Phi/\partial Y$. Les applications les plus directes de la méthode de Newton sont résumées dans les théorèmes suivants.

THÉORÈME 2. *Dans le cas où la série Φ du Théorème 1 est un polynôme en Y de degré $O(1)$, les N premiers termes de sa solution série telle que $S(0) = 0$ sont obtenus en $O(\mathsf{M}(N))$ opérations dans $\mathbb{A}$.*

DÉMONSTRATION. L'argument est le même que pour la complexité de l'inverse. Pour Φ polynôme de degré $O(1)$, les évaluations de Φ et de Φ' coûtent $O(\mathsf{M}(N))$, le reste de l'argument est inchangé. □

Dans le cas où Φ est un polynôme à la fois en X et en Y, un algorithme de meilleure complexité $O(N)$ (donc linéaire au lieu de quasi-linéaire !) sera présenté au Chapitre 14.

THÉORÈME 3. *Soit f une série de $\mathbb{A}[[X]]$. On peut calculer en $O(\mathsf{M}(N))$ opérations dans $\mathbb{A}$ les N premiers termes de :*

1. *f^{-1}, lorsque $f(0)$ est inversible ;*
2. *$\log f$ et f^α lorsque $f(0) = 1$ et $\alpha \in \mathbb{K}$;*
3. *$\exp(f)$ lorsque $f(0) = 0$.*

Pour les points (2) et (3), nous supposons également que $2, 3, \ldots, N-1$ sont des éléments inversibles de $\mathbb{A}$.

DÉMONSTRATION. La preuve pour l'inverse et le logarithme provient des sections 2.3 et 2.6. L'exponentielle est traitée dans la section qui vient, et la puissance se déduit des précédentes par

$$f^\alpha = \exp(\alpha \log f).$$

Une application remarquable de cette identité évidente apparaît dans la Section 3.8 sur l'inverse compositionnel. □

Sommes de Newton

Entrée : un polynôme $P \in \mathbb{K}[X]$ de degré d, un entier k.

Sortie : les sommes de Newton $p_0, \ldots, p_k$ de P.

1. Former les polynômes $N = T^{d-1}P'(1/T)$ et $D = T^d P(1/T)$;
2. Calculer $N/D \bmod T^{k+1} = p_0 + p_1 T + \cdots + p_k T^k$;
3. Renvoyer les coefficients.

FIGURE 4. Calcul des sommes de Newton.

Coefficients

Entrée : les sommes de Newton $p_0, \ldots, p_d$ du polynôme unitaire $P \in \mathbb{K}[X]$.

Sortie : les coefficients de P.

1. Former la série $S_1 := p_1 + p_2 T + \cdots + p_d T^{d-1}$;
2. Calculer $S_2 := \exp(-\int S_1) \bmod T^{d+1}$;
3. Renvoyer $X^{p_0} S_2(1/X)$.

FIGURE 5. Transformation inverse.

3.4. Exponentielle. Si $F \in \mathbb{A}[[X]]$ est telle que $F(0) = 0$, on définit la série $\exp(F)$ par

$$\exp(F) = 1 + F + F^2/2! + F^3/3! + \cdots$$

lorsque $2, 3, \ldots$ sont inversibles dans $\mathbb{A}$.

Pour calculer cette série, l'idée est d'appliquer une méthode de Newton avec

$$\Phi(Y) = F - \log Y, \quad Y(0) = 1.$$

EXERCICE 3. Montrer que les conditions du Théorème 1 sont vérifiées.

Il s'ensuit une convergence quadratique vers $\exp(F)$ pour l'itération

$$E_{k+1} = \mathcal{N}(E_k) = E_k + E_k(F - \log E_k).$$

Chaque itération demande donc le calcul d'une multiplication et d'un logarithme, d'où la complexité en $O(\mathsf{M}(N))$.

Une autre itération légèrement plus efficace est présentée en Exercice 8.

EXERCICE 4. Donner une itération de Newton qui calcule directement la racine carrée, sans passer par l'exponentielle et le logarithme. Plus difficile, estimer le gain par rapport à l'algorithme général de calcul de puissance.

EXERCICE 5. Donner des bornes sur les constantes dans les $O(\cdot)$ pour le logarithme, l'exponentielle et la puissance.

3.5. Sommes de Newton. Une conséquence intéressante de l'efficacité du calcul de l'exponentielle est l'utilisation efficace des *sommes de Newton* comme structure de données.

Si $P \in \mathbb{K}[X]$ est un polynôme de degré d, avec $\mathbb{K}$ un corps, alors P a d racines $\alpha_1, \ldots, \alpha_d$ dans la clôture algébrique $\overline{\mathbb{K}}$ de $\mathbb{K}$. Les *sommes de Newton* de P sont les sommes $p_i = \alpha_1^i + \cdots + \alpha_d^i$. Ce sont des éléments de $\mathbb{K}$, puisqu'ils sont fonctions symétriques des racines de P. (Tout polynôme symétrique en les racines d'une équation algébrique s'exprime comme un polynôme en les coefficients de l'équation.)

Leur utilisation efficace repose sur la proposition suivante.

Produit composé

Entrée : les polynômes unitaires P et Q de $\mathbb{K}[X]$, de degrés d_p et d_q.

Sortie : leur produit composé $P \otimes Q$ de degré $d := d_p d_q$.

1. Calculer les sommes de Newton $(p_0, \ldots, p_{d^2})$ et $(q_0, \ldots, q_{d^2})$ de P et Q.
2. Renvoyer le polynôme dont les sommes de Newton sont $(p_0 q_0, \ldots, p_{d^2} q_{d^2})$.

FIGURE 6.

Somme composée

Entrée : les polynômes unitaires P et Q de $\mathbb{K}[X]$, de degrés d_p et d_q.

Sortie : leur produit composé $P \otimes Q$ de degré $d := d_p d_q$.

1. Calculer les sommes de Newton $(p_0, \ldots, p_{d^2})$ et $(q_0, \ldots, q_{d^2})$ de P et Q.
2. Former les polynômes
$$S_p := p_0 + p_1 \frac{T}{1!} + \cdots + p_{d^2} \frac{T^{d^2}}{(d^2)!} \quad \text{et} \quad S_q := q_0 + q_1 \frac{T}{1!} + \cdots + q_{d^2} \frac{T^{d^2}}{(d^2)!}$$
3. Calculer $S := S_p \times S_q \bmod T^{d^2+1} = s_0 + s_1 T + s_2 T^2 + \cdots$
4. Renvoyer le polynôme dont les sommes de Newton sont
$$(s_0, 1!s_1, 2!s_2, \ldots, (d^2)!s_{d^2}).$$

FIGURE 7.

PROPOSITION 4. *Les sommes de Newton $p_1, \ldots, p_d$ d'un polynôme $P \in \mathbb{K}[X]$ de degré d peuvent être calculées par l'algorithme de la Figure 4 en $O(\mathsf{M}(d))$ opérations dans $\mathbb{K}$. Lorsque la caractéristique de $\mathbb{K}$ est 0 ou strictement supérieure à d, la conversion inverse est également possible en $O(\mathsf{M}(d))$ opérations par l'algorithme de la Figure 5.*

DÉMONSTRATION. Si $P = c\prod_\alpha (X - \alpha)$, la décomposition en éléments simples
$$S = \frac{XP'}{P} = \sum_{P(\alpha)=0} \frac{X}{X-\alpha} = \sum_{\substack{P(\alpha)=0 \\ i \geq 0}} \alpha^i X^{-i},$$
où α décrit les zéros de P comptés avec multiplicité, montre que les coefficients du développement en puissances de X^{-1} de XP'/P sont les p_i. Pour se ramener à des calculs de séries formelles, la première étape introduit une nouvelle variable par $X = 1/T$. Le calcul de la dérivée a complexité linéaire en d ; les conversions en puissances de T ne demandent pas d'opérations arithmétiques. Le calcul des $k = d$ premiers termes de la division demande $O(\mathsf{M}(d))$ opérations, comme montré en Section 2.4.

L'opération inverse, à savoir le calcul de P à partir des d premiers termes de la série $S = XP'/P$ est effectué en $O(\mathsf{M}(d))$ opérations par la formule
$$P = \exp \int S/X.$$
À nouveau, il est plus simple de travailler en T pour rester dans le domaine des séries formelles. Le signe '-' dans le calcul de S_2 provient du signe de l'intégration des puissances négatives de X. La contrainte sur la caractéristique permet de définir l'exponentielle tronquée par son développement en série. □

3.6. Application à la somme et au produit composés. On suppose dans cette section que $\mathbb{K}$ est un corps de caractéristique nulle ou strictement supérieure à d.

THÉORÈME 4. *Soient P et Q deux polynômes unitaires de $\mathbb{K}[X]$, de degrés d_p et d_q et soit $d = d_p d_q$. Alors, les polynômes*

$$P \oplus Q = \prod_{\substack{P(\alpha)=0\\ Q(\beta)=0}} \big(X - (\alpha+\beta)\big), \qquad P \otimes Q = \prod_{\substack{P(\alpha)=0\\ Q(\beta)=0}} (X - \alpha\beta)$$

peuvent être calculés en $O\big(\mathsf{M}(d)\big)$ opérations dans $\mathbb{K}$ par les algorithmes des Figures 6 et 7.

Comme ces polynômes sont de degré d, il s'agit là encore d'une complexité quasi-optimale. Dans cet énoncé, les polynômes sont définis à l'aide des racines de P et Q dans une clôture algébrique, mais leurs coefficients sont dans $\mathbb{K}$. Il est également possible de les définir sur un anneau quelconque comme des *résultants* en deux variables. Les résultants et leur calcul sont présentés au Chapitre 6. En général, avec deux variables, il n'existe pas (encore ?) d'algorithme quasi-optimal pour le résultant et le résultat ci-dessus est l'un des rares cas particuliers qui se traite efficacement.

Les polynômes constituent une structure de données bien commode pour coder des nombres algébriques et calculer avec : les polynômes $P \oplus Q$ et $P \otimes Q$ fournissent l'addition et la multiplication avec cette structure de données.

DÉMONSTRATION. Le produit de sommes de Newton est la somme de Newton du produit :

$$\sum_{\alpha} \alpha^i \sum_{\beta} \beta^i = \sum_{\alpha,\beta} (\alpha\beta)^i.$$

Il suffit donc de multiplier *terme à terme* les séries génératrices des sommes de Newton de P et Q pour obtenir la série génératrice des sommes de Newton de $P \otimes Q$. Si l'on a calculé d termes de ces séries, le résultant, dont le degré est d, est alors reconstruit par une exponentielle en $O(\mathsf{M}(d))$ opérations.

Diviser le i^{e} coefficient de XP'/P par $i!$, pour tout $i \geq 0$, produit la série

$$\sum_{\substack{P(\alpha)=0\\ i\geq 0}} \frac{\alpha^i X^{-i}}{i!} = \sum_{P(\alpha)=0} \exp(\alpha/X).$$

En effectuant la même opération sur Q et en multipliant les séries, on obtient donc

$$\sum_{P(\alpha)=0} \exp(\alpha/X) \sum_{Q(\beta)=0} \exp(\beta/X) = \sum_{\substack{P(\alpha)=0\\ Q(\beta)=0}} \exp((\alpha+\beta)/X).$$

Il suffit alors de remultiplier le i^{e} coefficient par $i!$ pour obtenir la série génératrice des sommes de Newton du polynôme $P \otimes Q$, qui est reconstruit par une exponentielle. L'ensemble des opérations a une complexité bornée par $O(\mathsf{M}(d))$. □

3.7. Systèmes. Le théorème 1 s'étend à des systèmes d'équations.

THÉORÈME 5. *Soient $\Phi = (\Phi_1, \ldots, \Phi_k)$ des séries en $k+1$ indéterminées X et $Y = (Y_1, \ldots, Y_k)$, telles que $\Phi(0) = 0$ et la matrice jacobienne $(\frac{\partial \Phi}{\partial Y})$ est inversible dans $\mathbb{A}$ en 0. Alors, le système*

$$\Phi_1(X, Y_1, \ldots, Y_k) = \cdots = \Phi_k(X, Y_1, \ldots, Y_k) = 0$$

admet une solution $S = (S_1, \ldots, S_k) \in \mathbb{A}[[X]]^k$ *telle que* $S(0) = 0$. *Si* $F = (F_1, \ldots, F_k)$ *est tel que* $S - F = O(X^n)$ *(*$n \geq 1$*), alors*

$$\mathcal{N}(F) = \begin{pmatrix} F_1 \\ \vdots \\ F_k \end{pmatrix} - \left(\frac{\partial \Phi}{\partial Y}(X, F_1, \ldots, F_k)\right)^{-1} \cdot \begin{pmatrix} \Phi_1(X, F_1, \ldots, F_k) \\ \vdots \\ \Phi_k(X, F_1, \ldots, F_k) \end{pmatrix}$$

vérifie $S - \mathcal{N}(F) = O(X^{2n})$.

La preuve est la même que celle du théorème 1, la formule de Taylor pour les fonctions de plusieurs variables s'exprimant alors à l'aide de la matrice jacobienne.

LEMME 4. *L'algorithme 3 fonctionne sans changement pour un système* Φ *satisfaisant aux hypothèses du théorème 5.*

DÉMONSTRATION. La preuve est la même, en observant que l'étape 2 est écrite de telle sorte que le lemme 1 s'applique : cette étape calcule l'inverse de la matrice jacobienne à la précision requise et la dernière étape effectue un produit matrice-vecteur pour en déduire la correction de la nouvelle itération. □

Le corollaire suivant joue un rôle important dans l'algorithme de résolution géométrique du Chapitre 26.

COROLLAIRE 1 (Systèmes polynomiaux). *Sous les mêmes hypothèses, si* $\Phi_i \in \mathbb{A}[[X]][Y_1, \ldots, Y_k]$ *pour* $1 \leq i \leq k$, *alors les solutions séries à précision* N *peuvent être calculées en* $O(\mathsf{MM}(k, N))$ *opérations dans* $\mathbb{A}$.

DÉMONSTRATION. L'étape 2 demande des produits de matrices $k \times k$ de polynômes de degré $\lceil N/2 \rceil$, et l'étape 3 demande un produit matrice-vecteur en précision N. La complexité de l'application récursive est donc dominée par celle de l'étape 2, le résultat est donc le même que pour l'inverse de matrices de séries. □

3.8. Inverse compositionnel. Étant donnée une série F avec $F(0) = 0$ et $F'(0)$ inversible, il s'agit ici de calculer une série $F^{(-1)}$ telle que $F(F^{(-1)}(X)) = F^{(-1)}(F(X)) = X$.

Les N premiers termes. Les conditions d'application de la méthode de Newton (Théorème 1) sont vérifiées pour la fonction

$$\Phi(X, Y) = F(Y) - X.$$

L'opérateur de Newton correspondant est donc

$$\mathcal{N}(G) = G - \frac{F(G) - X}{F'(G)}.$$

La dérivation de $F(G) = X$ donne $F'(G)G' = 1$ et cet opérateur se simplifie en

$$\mathcal{N}(G) = G - G' \times (F(G) - X).$$

Le coût est alors dominé soit par celui du produit, soit plus généralement par celui de la composition par F (voir Section 4).

EXERCICE 6. La k^{e} racine positive de l'équation $x = \tan x$ admet un développement asymptotique de la forme

$$r_k = (2k+1)\frac{\pi}{2} + \frac{a_1}{k} + \frac{a_2}{k^2} + \ldots.$$

Pour tout i, le coefficient a_i s'écrit $a_i = f_i(1/\pi)$, où f_i est un polynôme impair de degré $2i + 1$. Borner le nombre d'opérations dans $\mathbb{Q}$ nécessaires pour calculer $f_1, \ldots, f_N$, pour N grand.

Exemple. La série génératrice exponentielle $T \in \mathbb{Q}[[X]]$ des arbres généraux étiquetés (appelés aussi *arbres de Cayley*) vérifie l'équation

$$T = X \exp(T),$$

c'est-à-dire que le coefficient de X^n dans la série T est le nombre d'arbres de taille n divisé par $n!$. Ces coefficients sont obtenus en $O(\mathsf{M}(N))$ opérations dans $\mathbb{Q}$ en calculant d'abord ceux de $Y \exp(-Y)$ par les méthodes de la section précédente, puis en inversant ce développement par la méthode présentée ci-dessus.

Pour déterminer le comportement asymptotique des coefficients ainsi calculés lorsque N tend vers l'infini (c'est-à-dire l'asymptotique du nombre d'arbres de taille N), une méthode utilisée en analyse d'algorithmes passe par le calcul du développement de T au voisinage du point singulier $\exp(-1) = 1/e$, où T vaut 1. En posant $u = \sqrt{2}\sqrt{1 - eX}$, on voit qu'il s'agit alors de calculer le développement de y solution de

$$\sqrt{2 - 2y \exp(1 - y)} = u,$$

au voisinage de $y = 1$ et $u = 0$, équation à laquelle la méthode présentée dans cette section s'applique pour donner le résultat en $O(\mathsf{M}(N))$ opérations.

Le N^e terme. Bien qu'en général le coût de l'inverse compositionnel soit dominé par celui de la composition et donc en $O(\sqrt{N \log N}\mathsf{M}(N))$ (voir la section suivante), il est possible d'obtenir le N^{e} terme sans calculer les précédents pour $O(\mathsf{M}(N))$ opérations. L'idée repose sur la *formule d'inversion de Lagrange* :

$$[X^N]F^{(-1)}(X) = \frac{1}{N}[X^{N-1}]\frac{1}{(F(X)/X)^N}, \tag{5}$$

où la notation $[X^k]F$ représente le coefficient de X^k dans la série F. Ainsi, le calcul par cette formule ne requiert qu'une puissance et donc peut être effectué en $O(\mathsf{M}(N))$ opérations.

Remarque. Cette formule donne immédiatement une jolie expression pour les coefficients de T solution de $T = X \exp(T)$ de l'exemple précédent : le n^{e} coefficient vaut $n^{n-1}/n!$. Cette expression permet de calculer les N premiers coefficients en $O(N)$ opérations, ce qui est légèrement mieux que le $O(\mathsf{M}(N))$ du cas général. L'asymptotique mentionnée plus haut redonne dans cet exemple la formule de Stirling.

4. La composition des séries

Si $F(X)$ et $G(X)$ sont des séries, avec $G(0) = 0$, il s'agit de calculer efficacement la série $F(G(X))$. À la différence des algorithmes vus dans les sections précédentes, il s'avère que les algorithmes connus n'atteignent pas une complexité quasi-optimale (c'est-à-dire linéaire en N, à des facteurs logarithmiques près).

4.1. Méthode naïve. La composition peut être calculée par la méthode de Horner. Si $F(X) = \sum_{i \geq 0} f_i X^i + O(X^N)$, le calcul utilise alors

$$F(G(X)) = f_0 + G \times (f_1 + G \times (f_2 + \dots)) + O(G^N)$$

et $O(G^N) = O(X^N)$.

Exercice 7. Montrer que cette formule amène à une complexité $O(N\mathsf{M}(N))$.

4.2. Pas de bébés, pas de géants. Une technique dite « pas de bébés, pas de géants » réduit ce coût à $O(\sqrt{N}(\mathsf{M}(N)+\mathsf{MM}(\sqrt{N})))=O(\sqrt{N}(\mathsf{M}(N)+N^{\theta/2}))$, où θ est un exposant faisable pour la complexité du produit de matrices, présenté au Chapitre 8. La série F est d'abord écrite

$$F(X)=F_0(X)+X^kF_1(X)+\cdots+X^{k\lceil N/k\rceil}F_{\lceil N/k\rceil}(X)+O(X^N),$$

où les polynômes F_i, en nombre $1+\lceil N/k\rceil$, ont degré au plus $k-1$. Ensuite, on calcule les puissances $G^2,\dots,G^k$ de G en $k\mathsf{M}(N)$ opérations. Soient $f_{i,j}$ les coefficients des F_i et $g_{i,j}$ ceux des G^i. Le développement

$$F_i(G)=\sum_{j=0}^{N-1}\left(\sum_{\ell=0}^{k-1}f_{i,\ell}g_{\ell,j}\right)X^j+O(X^N)$$

montre que les coefficients des $F_i(G)$ sont les coefficients du produit de la matrice $(f_{i,j})$ de taille $(\lceil N/k\rceil+1)\times k$ par la matrice $(g_{i,j})$ de taille $k\times N$. En découpant chacune de ces matrices en blocs de taille $k\times k$ — la première devient un vecteur colonne d'environ N/k^2 blocs, la seconde un vecteur ligne d'environ N/k blocs —, ce produit est effectué en au plus $O(N^2k^{\theta-3})$ opérations. Ensuite, il ne reste plus qu'à effectuer $\lceil N/k\rceil$ produits à précision N suivis d'autant d'additions pour un coût total de $O(N\mathsf{M}(N)/k)$. Le coût total est minimisé par le choix $k=\lfloor\sqrt{N}\rfloor$ qui mène à la complexité annoncée.

4.3. L'algorithme de Brent & Kung. Bien qu'il n'y ait pas d'algorithme permettant d'abaisser la complexité de la composition au même ordre que $\mathsf{M}(N)$, il est possible de faire sensiblement mieux que la méthode de 4.2, en évitant la multiplication matricielle, grâce à un algorithme sophistiqué dû à Brent & Kung, détaillé en Figure 8.

THÉORÈME 6. *Si $2,3,\dots,\lceil\sqrt{N\log N}\rceil$ sont inversibles dans l'anneau $\mathbb{A}$, $F(X)$ et $G(X)$ sont deux séries de $\mathbb{A}[[X]]$ avec $G(0)=0$ et $G'(0)$ inversible, alors la série $F(G(X)) \bmod X^N$ peut être calculée en $O(\sqrt{N\log N}\mathsf{M}(N))$ opérations arithmétiques, ou en $O(\sqrt{N}\mathsf{M}(N))$ opérations arithmétiques si $\log\mathsf{M}(N)/\log N\to\gamma>1$. L'algorithme est donné en Figure 8.*

L'idée de départ de l'algorithme permettant d'aboutir à cette complexité est d'utiliser la formule de développement de Taylor, sous la forme

$$F(G_1+X^mG_2)=F(G_1)+F'(G_1)X^mG_2+F''(G_1)\frac{X^{2m}G_2^2}{2!}+\cdots, \tag{6}$$

où G_1 est un polynôme de degré inférieur à m ; le choix de m est ajusté plus loin pour minimiser la complexité.

Le premier gain de complexité par rapport à la méthode naïve provient de l'observation suivante.

LEMME 5. *Étant données les séries G et $F\circ G$, avec $G'(0)$ inversible, la série $F'\circ G$ peut être calculée en $O(\mathsf{M}(N))$ opérations arithmétiques.*

DÉMONSTRATION. La dérivation de séries a une complexité linéaire, et la division est en $O(\mathsf{M}(N))$. Le résultat provient alors de la formule de dérivation d'une composée (déjà utilisée en page 52) :

$$F'\circ G=\frac{(F\circ G)'}{G'}.$$

L'hypothèse sur $G'(0)$ permet de garantir l'inversion de la série du dénominateur. □

Composition de séries

Entrée : deux séries tronquées $F(X) \bmod X^N$ et $G(X) \bmod X^N$ avec $G(0) = 0$.

Sortie : leur composition $F(G(X)) \bmod X^N$.

1. Soit $m = \lceil \sqrt{N} \rceil$ si la multiplication utilisée emploie la FFT,
   $m = \lceil \sqrt{N/\log N} \rceil$ sinon ;
2. Découper G en $G_1 = G \bmod X^m$ et $G_2 = G - G_1$, poser $H := 1$;
3. Calculer $U = 1/G_1' \bmod X^N$;
4. Calculer $R = S_0 := F(G_1) \bmod X^N$ par la méthode récursive suivante :
   - Si $\deg F = 1$ renvoyer $F(0) + (F - F(0))G_1 \bmod X^N$
   - sinon
     - soit $k = \lceil \deg F/2 \rceil$;
     - découper F en $A := F \bmod X^k$ et $B := (F - A)/X^k$;
     - calculer récursivement $G_1^{\lceil k/2 \rceil}$, $A(G_1)$, et $B(G_1)$ modulo X^N
     - renvoyer $A(G_1) + G_1^k B(G_1) \bmod X^N$
5. Pour $i = 1, \ldots, \lceil N/m \rceil$ faire
   - $H := HG_2 \bmod X^N$;
   - $S_i := S_{i-1}' U \bmod X^N$;
   - $R := R + S_i H \bmod X^N$;
6. Renvoyer R

FIGURE 8. Algorithme de composition de Brent et Kung

L'utilisation répétée de cette idée dans la formule (6) mène à une complexité

$$C(F(G_1)) + \frac{N}{m} O(\mathsf{M}(N)) \text{ opérations}$$

pour l'ensemble de la composition, où $C(F(G_1))$ représente la complexité du calcul de la première composition avec un *polynôme*, donnée par le lemme suivant.

LEMME 6. *Soient F et G deux* polynômes *de degrés k et m, avec $G(0) = 0$. Le calcul des N premiers coefficients de la* série *$F \circ G$ peut être effectué en $O(km \log N \mathsf{M}(N)/N)$ opérations arithmétiques, ou en $O(km\mathsf{M}(N)/N)$ opérations si $\log \mathsf{M}(N)/\log N \to \gamma > 1$.*

DÉMONSTRATION DU THÉORÈME À L'AIDE DE CE LEMME. Ce lemme fournit l'estimation $C(F(G_1)) = O(m\mathsf{M}(N) \log N)$. La complexité totale est donc bornée par

$$O(m\mathsf{M}(N) \log N) + \frac{N}{m} O(\mathsf{M}(N)) = O\left(\mathsf{M}(N)(m \log N + \frac{N}{m})\right).$$

Cette dernière somme est minimisée par le choix de $m = \sqrt{N/\log N}$ qui donne le résultat du théorème.

Le cas sans le facteur $\log N$ est laissé en exercice. □

DÉMONSTRATION DU LEMME. Comme d'habitude, on commence par traiter le cas où le degré k est une puissance de 2. Ensuite, quitte à considérer que les coefficients de F pour les degrés allant de $k+1$ jusqu'à la puissance de 2 immédiatement supérieure sont nuls, la perte de complexité est d'au plus un facteur 2, absorbé dans la constante du $O(\cdot)$.

L'idée est d'effectuer ce calcul par « diviser pour régner » en récrivant le polynôme F sous la forme

$$F = F_1 + X^{k/2} F_2,$$

où F_1 et F_2 sont deux polynômes de degré au plus $k/2$. Dans la composition

$$F(G) = F_1(G) + G^{k/2} F_2(G),$$

les deux polynômes du second sommant ont degré au plus $km/2$. La complexité C_k^m découlant de l'utilisation de cette formule vérifie donc

$$C_k^m \leq 2C_{k/2}^m + 2\mathsf{M}(\min(N, km/2)) + O(N).$$

C'est dans cette prise de minimum que réside toute la subtilité : tant que k est grand, les calculs sont tronqués à l'ordre N, et dès que k devient suffisamment petit, on exploite l'arithmétique polynomiale. Il vient donc

$$\begin{aligned} C_k^m &\leq 2\mathsf{M}(N) + 4\mathsf{M}(N) + \cdots + 2^{\ell-1}\mathsf{M}(N) \\ &\quad + 2^\ell \left(\mathsf{M}\left(\frac{km}{2^\ell}\right) + 2\mathsf{M}\left(\frac{km}{2^{\ell+1}}\right) + \cdots \right), \\ &\leq 2^\ell \mathsf{M}(N) + 2^\ell \mathsf{M}\left(\frac{km}{2^\ell}\right) \log\left(\frac{km}{2^\ell}\right), \end{aligned}$$

où ℓ est déterminé par

$$\frac{km}{2^\ell} < N \leq \frac{km}{2^{\ell-1}}.$$

Cette valeur de ℓ, combinée à l'utilisation du théorème « diviser pour régner » conclut la preuve du lemme. □

Exercices

Exercice 8 (Exponentielle de série). Soit s une série de terme constant nul et $n = 2^p$. On étudie l'itération

$$\begin{aligned} z_k &= z_{k-1} + z_{k-1}(1 - y_{k-1} z_{k-1}) \mod X^{2^{k-1}}, \\ y_k &= y_{k-1} - y_{k-1} \int z_k (y'_{k-1} - s' y_{k-1}) \mod X^{2^k} \end{aligned}$$

avec $y_0 = 1 = z_0$.

1. Montrer que $y_p - \exp(s) = O(X^n)$;
2. Estimer la complexité du calcul des n premiers coefficients de la série $\exp(s)$ par cette itération ;
3. (Plus difficile) interpréter cette itération comme l'itération de Newton associée à l'opérateur $Y \mapsto Y' - s'Y$.

Exercice 9 (Composition avec l'exponentielle). Soit $f(x) \in \mathbb{Q}[[X]]$ une série à coefficients rationnels. Le but de cet exercice est de montrer que les N premiers coefficients de la série $S(X) = f(e^X - 1)$ peuvent être calculés en $O\big(\mathsf{M}(N) \log N\big)$ opérations arithmétiques dans $\mathbb{Q}$.

1. Montrer que $S(X) - A(e^X - 1) = O(X^N)$, où $A(X)$ est l'unique polynôme de degré inférieur à N tel que $f(X) = A(X) + O(X^N)$.
2. On pose $B(X) = A(X - 1)$; donner un algorithme pour calculer les coefficients de B à partir de ceux de A en $O\big(\mathsf{M}(N) \log N\big)$ opérations dans $\mathbb{Q}$.

3. La transformée de Laplace formelle $\mathcal{L}$ est définie sur $\mathbb{Q}[[X]]$ comme l'application $\mathbb{Q}$-linéaire telle que $\mathcal{L}(X^k) = k!\,X^k$, $k \in \mathbb{N}$. Si $B(X) = b_0 + b_1 X + \cdots + b_{N-1}X^{N-1}$, montrer que
$$\mathcal{L}\big(B(e^X)\big) = \sum_{i=0}^{N-1} \frac{b_i}{1 - iX}.$$
4. Donner un algorithme pour calculer les N premiers coefficients de $B(e^X)$ à partir de ceux de $B(X)$ en $O\big(\mathsf{M}(N)\log N\big)$ opérations dans $\mathbb{Q}$.
5. Donner finalement un algorithme calculant les N premiers coefficients de S à partir de ceux de f en $O\big(\mathsf{M}(N)\log N\big)$ opérations dans $\mathbb{Q}$.

Exercice 10 (Composition de séries avec arcsinus). Soit $\mathbb{K}$ un corps de caractéristique nulle, et soit $F \in \mathbb{K}[X]$ une série formelle de terme constant nul. Écrire un algorithme à base d'itérations de Newton permettant de calculer les N premiers termes de la série composée $(\arcsin \circ F)$, à partir des N premiers termes de la série F, en $O(\mathsf{M}(N))$ opérations arithmétiques dans $\mathbb{K}$.

Notes

La méthode de Newton remonte à 1669, et se trouve bien décrite dans son traité des fluxions [17]. Cette méthode a ensuite été raffinée et étendue de nombreuses façons [23, 24].

L'utilisation de l'itération de Newton en informatique est très ancienne. L'inversion de matrice de la Section 2.7 remonte à Schulz [21] pour des matrices réelles. L'importance de la méthode de Newton pour le calcul rapide avec des séries formelles a été soulignée par Lipson [16] : « *In a symbolic mathematics system setting, we feel that Newton's method offers a unified and easily implemented algorithm for solving a wide variety of power series manipulation problems that can be cast into a root-finding mold. Our experience attests to the practicality of Newton's method. We feel that it is a great algebraic algorithm.* »

Du point de vue de la complexité, Sieveking [22] a donné l'algorithme d'inversion rapide de séries (notre figure 2). Kung [14] a observé qu'il s'agissait d'une itération de Newton et a amélioré la constante dans la complexité. Ensuite, Brent [7] a montré comment l'exponentielle et le logarithme s'obtiennent également en $O(\mathsf{M}(N))$ opérations. Les algorithmes de la Section 3.5 sont dus à Schönhage [20] ; l'application aux sommes et produits composés en Section 3.6 est beaucoup plus récente [4]. La méthode de Newton permet également la résolution de systèmes linéaires à coefficients polynomiaux (Chapitre 11) et la résolution d'équations ou de systèmes *différentiels* (Chapitre 13). L'idée d'incorporer le dividende dans la dernière itération de Newton, afin de gagner un facteur constant pour la division de séries (Section 2.4), est due à Karp et Markstein [12].

Les algorithmes non triviaux de composition décrits en Section 4 sont dus à Brent et Kung [6]. L'idée de l'algorithme « pas de bébés, pas de géants » en Section 4.2 remonte à Paterson et Stockmeyer [18]. La composition $F \circ G \bmod X^N$ peut s'effectuer en complexité quasi-optimale $O(\mathsf{M}(N)\log N)$ si G est un polynôme [6], ou plus généralement lorsque G est une série algébrique [11]. Des compositions avec d'autres fonctions particulières peuvent aussi être calculées en $O(\mathsf{M}(N)\log N)$, ou parfois $O(\mathsf{M}(N))$ [5]. Savoir si cela est encore possible pour des séries quelconques est un grand problème ouvert. La formule d'inversion évoquée en page 53 a été découverte par Lagrange en 1768 [15].

Dans le cas où la caractéristique p de $\mathbb{A}$ est finie, Bernstein [2] a montré qu'il est possible de descendre la composition à une complexité quasi-optimale en la précision N des séries, pour le prix d'une dépendence linéaire en p. Cet algorithme est

intéressant lorsque la caractéristique p est petite. Une avancée récente importante est due à Kedlaya et Umans [13], qui ont donné le premier algorithme de complexité binaire quasi-linéaire à la fois en N et en $\log(p)$, pour le calcul de $F \circ G \bmod X^N$, lorsque F et G sont des séries à coefficients dans le corps fini $\mathbb{F}_p$.

Ritzmann [19] a montré que la composition des séries entières peut s'effectuer en complexité *binaire* quasi-optimale : si $F, G \in \mathbb{Z}[X]$ ont des coefficients bornés par ℓ en valeur absolue, alors il est possible de calculer $F \circ G \bmod X^N$ en $O(\mathsf{M}_{\mathbb{Z}}(N^2 \log N \log \ell))$ opérations binaires. Il montre également que le problème de la composition de séries à coefficients dans un corps $\mathbb{K}$ est d'une difficulté équivalente à celui du calcul du N^{e} coefficient de $F \circ G$. Le problème de l'inverse compositionnel leur est également équivalent [6].

En pratique, les constantes cachées dans les $O(\cdot)$ jouent un grand rôle, et nécessitent généralement l'implantation à la fois des algorithmes asymptotiquement meilleurs et des algorithmes plus classiques, souvent plus efficaces en petite taille. Dans le modèle de multiplication polynomiale par FFT, les meilleurs constantes devant $\mathsf{M}(N)$ sont dues à Harvey [9, 10] : $\frac{13}{9}$ pour l'inverse, $\frac{4}{3}$ pour la racine carrée, $\frac{13}{6}$ pour l'exponentielle. Des articles de synthèse dus à Bernstein [1, 3] décrivent diverses techniques utilisées pour éliminer certains calculs redondants dans la méthode de Newton.

L'exercice 8 est inspiré par [8], et l'exercice 9 est tiré de [5].

Bibliographie

[1] Bernstein, Daniel J. (2001). *Removing redundancy in high-precision Newton iteration*. URL : http://cr.yp.to/fastnewton.html (visible en 2001).

[2] — (1998). « Composing power series over a finite ring in essentially linear time ». In : *Journal of Symbolic Computation*, vol. 26, n°3, p. 339–341.

[3] — (2008). « Fast multiplication and its applications ». In : *Algorithmic number theory : lattices, number fields, curves and cryptography*. Vol. 44. Publications of the Research Institute for Mathematical Sciences. Cambridge University Press, p. 325–384.

[4] Bostan, Alin, Philippe Flajolet, Bruno Salvy et Éric Schost (2006). « Fast Computation of Special Resultants ». In : *Journal of Symbolic Computation*, vol. 41, n°1, p. 1–29. DOI : 10.1016/j.jsc.2005.07.001.

[5] Bostan, Alin, Bruno Salvy et Éric Schost (2008). « Power Series Composition and Change of Basis ». In : *ISSAC'08 : International Symposium on Symbolic and Algebraic Computation*. Éd. par David J. Jeffrey. ACM Press, p. 269–276. DOI : 10.1145/1390768.1390806.

[6] Brent, R. P. et H. T. Kung (1978). « Fast algorithms for manipulating formal power series ». In : *Journal of the ACM*, vol. 25, n°4, p. 581–595.

[7] Brent, Richard P. (1976). « Multiple-precision zero-finding methods and the complexity of elementary function evaluation ». In : *Analytic computational complexity*. Proceedings of a Symposium held at Carnegie-Mellon University, Pittsburgh, PA, 1975. Academic Press, p. 151–176.

[8] Hanrot, Guillaume et Paul Zimmermann (2004). *Newton iteration revisited*. URL : http://www.loria.fr/~zimmerma/papers (visible en 2004).

[9] Harvey, David (2010). *Faster exponentials of power series*. Rapp. tech. arXiv. eprint : abs/0911.3110.

[10] — (2011). « Faster algorithms for the square root and reciprocal of power series ». In : *Mathematics of Computation*, vol. 80, p. 387–394.

[11] Hoeven, Joris van der (2002). « Relax, but don't be too lazy ». In : *Journal of Symbolic Computation*, vol. 34, n°6, p. 479–542.

[12] Karp, A. H. et P. Markstein (1997). « High-precision division and square root ». In : *ACM Transactions on Mathematical Software*, vol. 23, n°4, p. 561–589.

[13] Kedlaya, Kiran S. et Christopher Umans (2008). « Fast Modular Composition in any Characteristic ». In : *FOCS'08 : IEEE Conference on Foundations of Computer Science*. IEEE Computer Society, p. 146–155. doi : 10.1109/FOCS.2008.13.

[14] Kung, H. T. (1974). « On computing reciprocals of power series ». In : *Numerische Mathematik*, vol. 22, p. 341–348.

[15] Lagrange, Joseph-Louis (1768). « Nouvelle méthode pour résoudre les équations littérales par le moyen des séries ». In : *Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Berlin*, vol. 24, p. 251–326.

[16] Lipson, J. D. (1976). « Newton's Method : A Great Algebraic Algorithm ». In : *SYMSAC'76 : ACM Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 260–270.

[17] Newton, Isaac (1740). *La méthode des fluxions, et les suites infinies*. Traduction française par G. Buffon du texte de 1671 de Newton. de Bure. url : http://www.gallica.fr.

[18] Paterson, M. S. et L. J. Stockmeyer (1973). « On the Number of Nonscalar Multiplications Necessary to Evaluate Polynomials ». In : *SIAM Journal on Computing*, vol. 2, n°1, p. 60–66.

[19] Ritzmann, P. (1986). « A fast numerical algorithm for the composition of power series with complex coefficients ». In : *Theoretical Computer Science*, vol. 44, p. 1–16.

[20] Schönhage, A. (1982). *The fundamental theorem of algebra in terms of computational complexity*. Preliminary Report. 73 pages. Tübingen, Germany : Mathematisches Institut der Universität.

[21] Schulz, G. (1933). « Iterative Berechnung der reziproken Matrix ». In : *Zeitschrift für angewandte Mathematik und Physik*, vol. 13, p. 57–59.

[22] Sieveking, M. (1972). « An algorithm for division of powerseries ». In : *Computing*, vol. 10, p. 153–156.

[23] Yamamoto, Tetsuro (2000). « Historical developments in convergence analysis for Newton's and Newton-like methods ». In : *Journal of Computational and Applied Mathematics*, vol. 124, n°1-2, p. 1–23.

[24] Ypma, Tjalling J. (1995). « Historical development of the Newton-Raphson method ». In : *SIAM Review*, vol. 37, n°4, p. 531–551. doi : 10.1137/1037125.

CHAPITRE 4

Division euclidienne, fractions rationnelles et récurrences linéaires à coefficients constants

Résumé

La multiplication et l'inversion rapide de séries permettent le calcul efficace de la division euclidienne de polynômes et du développement en série des fractions rationnelles. Ils mènent en particulier au calcul rapide d'un ou de plusieurs termes d'une suite récurrente linéaire à coefficients constants.

1. Introduction

Les suites récurrentes linéaires à coefficients constants sont extrêmement présentes dans la nature. On estime ainsi qu'environ 12% des suites de la littérature [1] sont de ce type. L'exemple le plus célèbre est celui de la suite de Fibonacci, définie par $F_0 = 0, F_1 = 1$ et

$$F_{n+2} = F_{n+1} + F_n, \quad n \geq 0. \tag{1}$$

L'utilisation directe de cette récurrence permet de calculer les N premiers éléments de la suite de manière optimale du point de vue de la complexité. En revanche, pour le calcul d'un seul terme d'indice élevé, le passage par le calcul de tous ses prédécesseurs est inutilement coûteux. Une première idée de ce chapitre consiste à utiliser une version matricielle des récurrences pour aboutir à une complexité quasi-optimale pour cette question.

La récurrence (1) n'est que d'ordre 2. Pour des suites définies par des récurrences d'ordre d plus grand, il faut aussi chercher à limiter le coût des calculs vis-à-vis de d. Simplement dérouler la récurrence n'est plus alors la meilleure façon de calculer les N premiers termes de la suite. Une autre idée importante de ce chapitre est d'exploiter le lien entre les récurrences linéaires à coefficients constants et les fractions rationnelles, qui mène à une complexité quasi-optimale vis-à-vis à la fois de N et d pour cette question, ainsi que pour celle du calcul des N premiers termes, et aussi pour le développement en série des fractions rationnelles.

Les applications algorithmiques de ces idées sont nombreuses. Certaines d'entre elles sont présentées dans des exercices de ce chapitre, comme le calcul rapide d'un polynôme sur $1, 2, \ldots, N$, une partie du calcul du polynôme minimal d'une matrice, et des tests probabilistes de primalité d'entiers.

2. Division de polynômes

Étant donnés deux polynômes F et G de $\mathbb{A}[X]$, avec G unitaire, à coefficients dans un anneau $\mathbb{A}$, il existe d'uniques polynômes Q et R, quotient et reste de la division euclidienne de F par G, tels que :

$$F = QG + R \quad \text{avec} \quad \deg R < \deg G.$$

1. Recensées très largement par l'encyclopédie en ligne des suites d'entiers http://www.research.att.com/~njas/sequences/Seis.html.

Calculer rapidement le quotient Q et le reste R est d'importance vitale dans toute la suite du cours. Outre l'application aux suites récurrentes qui sera détaillée plus loin, les algorithmes de division sont utilisés au Chapitre 5 pour l'évaluation multipoint et l'interpolation et au Chapitre 6 pour le calcul de pgcd. À leur tour, ces deux algorithmes sont centraux dans nombre d'applications.

2.1. Méthode naïve. L'algorithme naïf pour calculer Q et R consiste à poser la division. Pour cela, les termes dominants sont d'abord isolés :

$$F = aX^m + T_F, \quad G = X^n + T_G, \qquad \text{avec} \quad \deg T_F < m, \quad \deg T_G < n.$$

Si $m < n$, il n'y a rien à faire. Sinon, la boucle élémentaire de la division de F par G consiste à « tuer » le terme de plus haut degré de F en effectuant la soustraction

$$F - aX^\delta G = (aX^m + T_F) - \left(aX^{\delta+n} + aX^\delta T_G\right) = T_F - aX^\delta T_G,$$

où $\delta = m-n$. Le polynôme obtenu est congru à F modulo G, mais de degré strictement inférieur à m. Il n'y a plus qu'à itérer ce procédé jusqu'à obtenir un polynôme de degré strictement inférieur à n, et à garder trace des quotients successifs.

La complexité de cet algorithme est facile à estimer. La boucle élémentaire présentée ci-dessus s'effectue en $O(n)$ opérations de type $(+, -, \times)$; dans le pire des cas, l'algorithme effectue $(m - n + 1)$ passages dans cette boucle, de sorte que la complexité de la division de F par G est de $O(n(m - n))$ opérations. Le pire des cas est atteint pour $m = 2n$. C'est donc un algorithme *quadratique* (mais un bon algorithme quadratique : la constante dans le $O(\cdot)$ est en fait petite, c'est 2). Le même genre d'estimation s'obtient pour la division euclidienne classique des entiers.

2.2. Algorithme rapide. Un bien meilleur résultat peut être obtenu à base d'itération de Newton.

L'idée principale de l'algorithme part de la réécriture de $F = QG + R$ en

$$\frac{F}{G} = Q + \frac{R}{G}.$$

Si on pense que $\mathbb{A} = \mathbb{R}$, on voit donc que le développement asymptotique de F/G *à l'infini* a ses premiers termes donnés par Q, puisque R/G tend vers 0 à l'infini (à cause des contraintes de degré sur R et G). Ceci suggère que l'on va obtenir Q par calcul du développement de Taylor de F/G au voisinage de l'infini.

Concrètement, il suffit de ramener d'abord l'infini en zéro (par le changement de variable $X = 1/T$), pour réduire le calcul à la division de séries formelles. Plus précisément, le point de départ est l'identité

$$\frac{T^m F(1/T)}{T^n G(1/T)} = T^{m-n} Q(1/T) + \frac{T^m R(1/T)}{T^n G(1/T)}.$$

Dans cette identité les numérateurs et dénominateurs des fractions sont des polynômes, et le polynôme $T^n G(1/T)$ a 1 pour terme constant. En outre, la valuation du second sommant du membre droit est supérieure à $m - n$. En découle donc l'algorithme de division rapide présenté en Figure 1.

THÉORÈME 1. *Soient G un polynôme unitaire de $\mathbb{A}[X]$ de degré n et $F \in \mathbb{A}[X]$ de degré $m \geq n$. Alors on peut calculer le quotient Q et le reste R de la division euclidienne de F par G en $5\mathsf{M}(m-n) + \mathsf{M}(n) + O(m)$ opérations $(+, -, \times)$ de $\mathbb{A}$.*

Ainsi, la division euclidienne ne coûte pas plus cher que la multiplication (à une constante près). En particulier, si on utilise une multiplication à base de FFT, le coût de la division est *linéaire* en le degré, à des facteurs logarithmiques près.

Division euclidienne

Entrée : F, G de degrés m et n.

Sortie : Q et R tels que $F = QG + R$ avec $\deg R < \deg G$.

1. Calculer $T^m F(1/T)$ et $T^n G(1/T)$ (aucune opération arithmétique n'est nécessaire, il suffit d'inverser l'ordre des coefficients) ;
2. calculer le quotient $(T^m F(1/T))/(T^n G(1/T)) \bmod T^{m-n+1}$ par une inversion de série formelle suivie d'un produit ;
3. en déduire Q en inversant l'ordre des coefficients ;
4. en déduire R, qui est donné par $R = F - QG \bmod X^n$.

FIGURE 1. Division euclidienne rapide.

DÉMONSTRATION. La complexité s'obtient en suivant les étapes de l'algorithme. D'après la Proposition 2 du Chapitre 3 et la remarque qui la suit, l'inverse du dénominateur à l'étape 2 s'obtient en

$$4\mathsf{M}(m-n) + O(m-n)$$

opérations dans $\mathbb{A}$, puis une multiplication supplémentaire donne Q. Pour retrouver R, le dernier produit est effectué modulo X^n, c'est-à-dire pour un coût de $\mathsf{M}(n)$. Toutes les additions sont prises en compte dans le terme $O(m)$. □

REMARQUE. Si Q a un degré beaucoup plus petit que G, on peut accélérer le calcul de QG, en découpant G en « tranches » de taille $m-n$; de la sorte, le dernier produit peut se faire en $\frac{n}{m-n}\mathsf{M}(m-n)$ opérations. Enfin, si de nombreuses réductions sont à faire modulo le même polynôme G, il est évidemment recommandé de stocker l'inverse du réciproque $T^n G(1/T)$ de G.

2.3. Le cas des entiers. Comme pour la multiplication, il est possible de poser le problème dans $\mathbb{Z}$ comme dans un anneau de polynômes. C'est ce dernier cas qui est le plus simple à étudier, puisqu'il n'y a pas à y gérer de retenue. On obtient cependant un résultat analogue sur les entiers.

THÉORÈME 2. *Soit $\mathsf{M}_{\mathbb{Z}}$ une fonction de multiplication pour $\mathbb{Z}$, et f et g deux entiers positifs avec $g \leq f \leq 2^n$. Soient q et r les quotient et reste de la division euclidienne de f par g :*

$$f = qg + r \quad \textit{avec} \quad 0 \leq r < g.$$

On peut calculer q et r en $O(\mathsf{M}_{\mathbb{Z}}(n))$ opérations binaires.

2.4. Application aux calculs modulaires. Une application importante de la division euclidienne rapide est le calcul efficace dans des structures algébriques de type « quotient », par exemple dans n'importe quel corps fini.

COROLLAIRE 1. *Si $\mathbb{A}$ est un anneau et si P est un polynôme unitaire de degré n de $\mathbb{A}[X]$, alors dans $\mathbb{A}[X]/(P)$, l'addition et la multiplication par un élément de $\mathbb{A}$ ont une complexité linéaire en n, la multiplication a une complexité en $O(\mathsf{M}(n))$.*

Le calcul de l'inverse (des éléments inversibles) dans $\mathbb{A}[X]/(P)$ est un peu plus complexe. Un algorithme de complexité $O(\mathsf{M}(n)\log n)$ est présenté au Chapitre 6.

DÉMONSTRATION. Il s'agit de calculer modulo P. L'addition et la multiplication par un scalaire s'effectuent terme à terme, ainsi leur complexité est linéaire en le degré n de P. Ensuite, pour multiplier deux éléments $A + (P)$ et $B + (P)$ de $\mathbb{A}[X]/(P)$, on commence par multiplier A et B dans $\mathbb{A}[X]$, puis on réduit le résultat modulo P. La complexité découle donc de ce qui précède. □

La question du calcul dans $\mathbb{Z}/n\mathbb{Z}$ est légèrement plus délicate que son analogue dans $\mathbb{A}[X]$, à cause des retenues. Malgré tout, les résultats s'étendent.

3. Suites récurrentes linéaires à coefficients constants

DEFINITION 1. Une suite $(a_n)_{n\geq 0}$ d'éléments de l'anneau $\mathbb{A}$ est appelée suite récurrente linéaire à coefficients constants (srlcc) d'ordre d si elle satisfait une récurrence de la forme

$$a_{n+d} = p_{d-1}a_{n+d-1} + \cdots + p_0 a_n, \qquad n \geq 0,$$

où les p_i sont des éléments de $\mathbb{A}$. Le polynôme $P = X^d - p_{d-1}X^{d-1} - \cdots - p_0$ de $\mathbb{A}[X]$ est appelé *polynôme caractéristique* de la suite $(a_n)_{n\geq 0}$.

Il faut noter que tout multiple d'un polynôme caractéristique est aussi caractéristique. Le pgcd de ces polynômes caractéristiques est appelé polynôme *minimal* de la suite.

3.1. Exemples.

EXERCICE 1. Soit M une matrice de taille $d \times d$ à coefficients dans $\mathbb{A}$, de polynôme caractéristique $\chi_M(X) = \det(XI_d - M)$, soient u une matrice ligne et v une matrice colonne. Montrer que la suite $(uM^n v)_{n\geq 0}$ est une suite récurrente linéaire à coefficients constants admettant χ_M comme polynôme caractéristique.

EXERCICE 2. Soit $a_1, \ldots, a_r$ des entiers strictement positifs. Soit A_n et B_n le nombre de r-uplets non-ordonnés, resp. ordonnés, $(x_1, \ldots, x_r) \in \mathbb{N}^r$, solutions de l'équation $a_1x_1 + \cdots + a_rx_r = n$. Montrer que les suites (A_n) et (B_n) sont récurrentes linéaires à coefficients constants, admettant $(X^{a_1} - 1)\cdots(X^{a_r} - 1)$ et $X^{a_1} + \cdots + X^{a_r} - 1$ comme polynômes caractéristiques.

EXERCICE 3. Si $P \in \mathbb{A}[X]$ est de degré d, alors la suite $(P(n))_{n\geq 0}$ est une srlcc de polynôme caractéristique $(X-1)^{d+1}$.

3.2. Méthode naïve. L'approche naïve pour le calcul des N premiers termes d'une srlcc consiste tout simplement à dérouler la récurrence et requiert $O(dN)$ opérations dans $\mathbb{A}$.

3.3. Exponentiation binaire. Pour calculer uniquement le N^{e} terme de la suite, une alternative à cette méthode naïve est basée sur l'exponentiation binaire de la matrice compagnon associée à la suite. Par exemple, la récurrence (1) se récrit matriciellement

$$\begin{bmatrix} F_N \\ F_{N+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}}_{C} \begin{bmatrix} F_{N-1} \\ F_N \end{bmatrix} = C^N \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}, \qquad N \geq 1.$$

La puissance de la matrice constante C se calcule alors récursivement par

$$C^N = \begin{cases} (C^{N/2})^2, & \text{si } N \text{ est pair,} \\ C \cdot (C^{\frac{N-1}{2}})^2, & \text{sinon.} \end{cases}$$

On en déduit que le calcul de F_N peut être effectué sans calculer les précédents en $O(\log N)$ produits de matrices 2×2, soit $O(\log N)$ opérations dans $\mathbb{A}$.

Cette idée s'étend facilement à des récurrences d'ordre arbitraire.

EXERCICE 4. Montrer que le N^{e} terme de toute récurrence linéaire d'ordre d à coefficients constants peut se calculer en $O(d^3 \log N)$ opérations dans $\mathbb{A}$.

L'utilisation d'une multiplication matricielle rapide abaisse cette complexité à $O(\mathsf{MM}(d) \log N) = O(d^\theta \log N)$ opérations dans $\mathbb{A}$ (voir le Chapitre 8). La dépendance en d reste cependant plus que quadratique. La section suivante présente une meilleure méthode, de complexité quasi-linéaire en d.

Exponentiation modulaire

Entrée : $(p_0, \dots, p_{d-1})$ les coefficients d'une récurrence linéaire ; $(a_0, \dots, a_{d-1})$ des conditions initiales ; un entier N.

Sortie : Le terme a_N de la suite définie par
$a_{n+d} = p_{d-1}a_{n+d-1} + \cdots + p_0 a_n$.

1. Poser $P = X^d - p_{d-1}X^{d-1} - \cdots - p_0$ et $x = X \bmod P$;
2. Calculer récursivement $Q = x^N = q_0 + q_1 x + \cdots + q_{d-1}x^{d-1}$ par
$$x^k = \begin{cases} (x^{k/2})^2, & \text{si } k \text{ est pair,} \\ x \cdot (x^{\frac{k-1}{2}})^2, & \text{sinon,} \end{cases}$$
3. Renvoyer $q_0 a_0 + \cdots + q_{d-1}a_{d-1}$.

FIGURE 2. Calcul du N^{e} terme d'une slrcc

3.4. Exponentiation modulaire. La complexité arithmétique de l'algorithme de la section précédente (Exercice 4) est quasi-optimale par rapport à l'indice N, mais n'est pas très bonne par rapport à l'ordre d de la récurrence.

Une méthode plus efficace est obtenue en exploitant mieux la structure du problème. En effet, les matrices multipliées ne sont pas quelconques, mais sont des puissances d'une matrice compagnon. Cette structure peut être exploitée grâce à l'observation suivante.

LEMME 1. *Soit $(a_n)_{n\geq 0}$ une suite récurrente linéaire à coefficients constants de polynôme caractéristique $P(X)$. Soit $\mathbb{B} = \mathbb{A}[X]/(P)$ et x la classe de X dans $\mathbb{B}$. Alors la matrice compagnon C de P est à la fois*

1. *telle que $[a_{n+1}, \dots, a_{n+d}] = [a_n, \dots, a_{n+d-1}] \cdot C$ pour tout $n \geq 0$;*
2. *matrice de l'application $\mathbb{A}$-linéaire de multiplication par x dans $\mathbb{B}$ dans la base canonique $\{1, x, \dots, x^{d-1}\}$.*

DÉMONSTRATION. Il suffit d'écrire la matrice pour le voir. Si $P = X^d - p_{d-1}X^{d-1} - \cdots - p_0$, la matrice C vaut par définition
$$C = \begin{pmatrix} 0 & 0 & \dots & 0 & p_0 \\ 1 & 0 & \dots & 0 & p_1 \\ 0 & 1 & \dots & 0 & p_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & p_{d-1} \end{pmatrix}.$$
La première propriété traduit la définition de la récurrence :
$$a_{n+d} = p_{d-1}a_{n+d-1} + \cdots + p_0 a_n.$$
La seconde exprime que modulo P, $X^d = p_{d-1}X^{d-1} + \cdots + p_0$. □

COROLLAIRE 2. *Avec les mêmes notations, pour tous entiers positifs i et k,*
$$a_{k+i} = [a_i, \dots, a_{d+i-1}] \cdot V_k,$$
où V_k a pour coordonnées les coefficients du polynôme $X^k \bmod P$.

DÉMONSTRATION. Le vecteur V_k est la première colonne de C^k, et d'après le lemme précédent, cette matrice est celle de la multiplication par X^k dans $\mathbb{B}$. Les coefficients de la première colonne sont donc les coordonnées de $x^k \cdot 1$ dans la base $\{1, x, \dots, x^{d-1}\}$. □

Le résultat de complexité est alors le suivant.

THÉORÈME 3. *Soit (a_n) une suite récurrente linéaire à coefficients constants, donnée par une récurrence d'ordre d et des conditions initiales $a_0, \dots, a_{d-1}$. Soit $N \geq d$. Alors, le calcul du N^e terme a_N peut se faire en seulement $O(\mathsf{M}(d) \log N)$ opérations dans $\mathbb{A}$.*

DÉMONSTRATION. L'algorithme est présenté en Figure 2. Chaque étape du calcul récursif a lieu dans $\mathbb{A}[X]/(P)$ et coûte $O(\mathsf{M}(d))$ opérations dans $\mathbb{A}$ d'après le Corollaire 1, le nombre d'étape est en $\log N$; la dernière multiplication ne prend que $O(d)$ opérations. La complexité est donc dominée par $O(\mathsf{M}(d) \log N)$. □

4. Développement de fractions rationnelles

Il est également possible de calculer efficacement les N premiers éléments d'une suite récurrente linéaire à coefficients constants, ou de manière équivalente, le développement de Taylor des fractions rationnelles.

4.1. Séries rationnelles et suites récurrentes. Ces deux notions sont deux aspects de la même question.

LEMME 2. *Soit $A(X) = \sum_{n \geq 0} a_n X^n$ la série génératrice de la suite $(a_n)_{n \geq 0}$. Les assertions suivantes sont équivalentes :*

(i) La suite (a_n) est une srlcc, avec polynôme caractéristique P de degré d ;

(ii) $A(X) = N_0/\bar{P}$ où $\bar{P} = P(1/X)X^d$, pour un certain $N_0 \in \mathbb{K}[X]$ avec $\deg(N_0) < d$.

De plus, si P est le polynôme minimal de (a_n), alors $d = \max\{1 + \deg(N_0), \deg(\bar{P})\}$ et $\operatorname{pgcd}(N_0, \bar{P}) = 1$.

DÉMONSTRATION. Supposons que P s'écrive $P = g_d X^d + \cdots + g_0$. Pour l'équivalence, on utilise uniquement le fait que le coefficient de X^{d+i} dans $\bar{P} \cdot A(X)$ est égal à $g_d a_{i+d} + \cdots + g_0 a_i$ et que $\bar{P}(0) = g_d \neq 0$.

Soit maintenant P le polynôme minimal de (a_n). On a $\deg(\bar{P}) \leq d$ avec égalité si et seulement si $g_0 \neq 0$, c'est-à-dire $X \nmid P$. Donc $d \geq \max\{1 + \deg(N_0), \deg(\bar{P})\}$. Supposons par l'absurde que cette inégalité est stricte. Alors $X \mid P$ et on a que P/X est aussi polynôme caractéristique de (a_n), ce qui contredit la minimalité de P. Donc $d = \max\{1 + \deg(N_0), \deg(\bar{P})\}$.

Soit enfin $u := \operatorname{pgcd}(N_0, \bar{P})$. Alors $P_u := P/\bar{u}$ est un polynôme de degré $d - \deg(u)$ qui est caractéristique de (a_n), car $\bar{P}/u = \bar{P}_u$ et $(\bar{P}/u) \cdot A(X) = N_0/u$ est un polynôme de degré $< d - \deg(u)$. Par la minimalité, cela implique que $\deg(u) = 0$, donc N_0 et $\bar{P}$ sont bien premiers entre eux. □

Une conséquence immédiate de cette équivalence et du Théorème 3 est alors :

COROLLAIRE 3. *Soit $F(X)/G(X)$ dans $\mathbb{A}(X)$ de degré au plus d, avec $G(0)$ inversible. Le N^e coefficient du développement en série $F(X)/G(X) = \sum_{n \geq 0} a_n X^n$ peut être calculé en $O(\mathsf{M}(d) \log N)$ opérations.*

4.2. Développement. On peut calculer le développement de Taylor à l'ordre N d'une fraction rationnelle de degré $d \leq N$ en $O(\mathsf{M}(N))$ opérations, en utilisant la méthode de Newton, décrite au Chapitre 3. Mais $O(\mathsf{M}(N))$ est en général beaucoup plus gros que les $O(dN)$ opérations requises par la méthode naïve. Une complexité linéaire en N tout en ne croissant pas trop par rapport à d est en fait possible.

THÉORÈME 4. *Soit $F(X)/G(X)$ dans $\mathbb{A}(X)$ avec numérateur et dénominateur de degrés au plus d, et $G(0)$ inversible. Le développement de Taylor de $F(X)/G(X)$ à précision $N \geq d$ peut se calculer en $O(N\mathsf{M}(d)/d)$ opérations $(+, -, \times)$ dans $\mathbb{A}$.*

Série rationnelle

Entrée : Deux polynômes $F(X), G(X)$ avec $\deg G = d$, $G(0) \neq 0$ et $\deg F < d$, un entier $N > d$.

Sortie : Les coefficients $a_0, \ldots, a_N$ du développement $F(X)/G(X) = \sum_{n \geq 0} a_n X^n$.

1. Calculer les $2d-1$ premiers coefficients $a_0, \ldots, a_{2d-2}$.
2. Poser $y_1 = X^{d-1} \bmod \bar{G}(X)$ et $y^\star = X^d \bmod \bar{G}(X)$, où $\bar{G}(X) = X^d G(1/X)$.
3. Pour $i = 2, \ldots, \lceil N/d \rceil$,
   (a) Calculer $y_i = y^\star y_{i-1} \bmod \bar{G}(X)$;
   (b) Calculer $P = (a_{2d-2} + \cdots + a_0 X^{2d-2}) y_i$;
   (c) Extraire $a_{id-1}, a_{id}, \ldots, a_{(i+1)d-2}$ les coefficients des monômes $X^{2d-2}, \ldots, X^d, X^{d-1}$ dans P.
4. Renvoyer $a_0, \ldots, a_N$.

FIGURE 3. Développement en série d'une fraction rationnelle.

Si l'anneau $\mathbb{A}$ permet la FFT, cette estimation de complexité devient $O(N \log d)$ ce qui est quasi-optimal, simultanément vis-à-vis de N et de d.

Le point de départ est le Corollaire 2 qui montre comment calculer un coefficient de la suite en fonction de coefficients distants. L'idée est de calculer les coefficients par tranches de longueur d. Il ne suffit pas d'utiliser d fois le corollaire, mais il faut encore une fois exploiter une structure supplémentaire, qui provient ici de ce que les coefficients cherchés sont consécutifs. Cette idée est résumée par le résultat suivant.

LEMME 3. *Soit $F(X)/G(X)$ dans $\mathbb{A}(X)$ avec $G(0)$ inversible. Soit d le degré de G et soit $k \geq 0$. Alors les coefficients $a_k, a_{k+1}, \ldots, a_{k+d-1}$ du développement de $F(X)/G(X) = \sum_i a_i X^i$ sont les coefficients de $X^{2d-2}, \ldots, X^d, X^{d-1}$ du produit*

$$(a_{2d-2} + \cdots + a_0 X^{2d-2})(X^k \bmod G(1/X)X^d).$$

DÉMONSTRATION. Le polynôme $G(1/X)X^d$ est polynôme caractéristique de la suite des coefficients d'après le Lemme 2. Le coefficient de X^{k+i} pour $i = 0, \ldots, d-1$ dans le produit ci-dessus vaut exactement le produit scalaire $[a_i, \ldots, a_{d+i-1}] \cdot V_k$ d'après le Corollaire 2. □

PREUVE DU THÉORÈME 4. L'algorithme est résumé en Figure 3. La première étape se calcule par itération de Newton en $O(\mathsf{M}(d))$ opérations dans $\mathbb{A}$, ce qui représente un coût négligeable sur l'ensemble. L'obtention de $y^\star$ est une réécriture à partir de $\bar{G}$ et ne demande pas d'opération. Ensuite, chaque itération utilise un produit dans $\mathbb{A}[X]/(\bar{G})$ et un produit d'un polynôme de degré $2d-2$ par un polynôme de degré $d-1$. Au total, sont donc effectuées $O(NM(d)/d)$ opérations dans $\mathbb{A}$. □

La traduction de ce résultat au niveau des srlcc est immédiate.

COROLLAIRE 4. *Soit (a_n) une suite récurrente linéaire à coefficients constants, donnée par une récurrence d'ordre d, et des conditions initiales $a_0, \ldots, a_{d-1}$. Soit $N \geq d$. Alors, les termes $a_0, \ldots, a_N$ peuvent être calculés en $O(N\mathsf{M}(d)/d)$ opérations dans $\mathbb{A}$.*

5. Applications

5.1. Évaluation d'un polynôme sur une progression arithmétique.

COROLLAIRE 5. *Un polynôme P de degré d peut être évalué aux $N+1 \gg d$ points $b, a+b, 2a+b, \ldots, Na+b$ au prix total de $O(N\,\mathsf{M}(d)/d)$ opérations arithmétiques.*

DÉMONSTRATION. L'opérateur $\Delta_a(P) = P(X+a) - P(X)$ fait décroître le degré des polynômes. Il s'ensuit que $\Delta_a^{d+1} P = 0$ et donc $P(an+b)$ est une suite récurrente linéaire d'ordre $d+1$ et de polynôme caractéristique $(X-1)^{d+1}$. □

5.2. Propriétés de clôture. La classe des srlcc admet de nombreuses propriétés de clôture : si $\mathbf{a} = (a_n)_n$ et $\mathbf{b} = (b_n)_n$ sont deux srlcc de polynômes caractéristiques P et Q, alors

1. la somme $\mathbf{a} + \mathbf{b} = (a_n + b_n)_n$ et le produit de Cauchy $\mathbf{a} \star_{\mathbf{C}} \mathbf{b}$, de terme général $\sum_{i=0}^{n} a_i b_{n-i}$, sont deux srlcc de polynôme caractéristique PQ ;
2. le produit d'Hadamard $\mathbf{a} \odot \mathbf{b} = (a_n b_n)_n$ est une srlcc de polynôme caractéristique égal au produit composé $P \otimes Q$ défini au Chapitre 3 ;
3. la suite $\left(\sum_{i=0}^{n} \binom{n}{i} a_i b_{n-i}\right)_n$ est une srlcc de polynôme caractéristique égal à la somme composée $P \oplus Q$ définie au Chapitre 3.

Ils se calculent donc tous en bonne complexité (quasi-linéaire en la taille de la sortie).

EXERCICE 5. Prouver les assertions précédentes.

5.3. Tests de primalité. Une application du calcul rapide d'un terme d'une récurrence est une famille de tests probabilistes de primalité, de complexité polynomiale. L'idée est de construire une suite récurrente (a_n) d'entiers telle que la primalité de n soit équivalente (ou presque équivalente) à $a_n \equiv 0 \bmod n$.

Un cas particulier important en est *le test de Fermat* (pour lequel $a_n = a^{n-1} - 1$) implanté dans la plupart des systèmes de calcul formel. Bien qu'il soit probabiliste (si n ne passe pas le test, n est composé, mais si n passe le test, alors il est premier seulement avec une grande probabilité), sa grande simplicité le rend souvent préférable à d'autres algorithmes sophistiqués.

EXERCICE 6. Soit (a_n) une srlcc d'ordre d. Montrer qu'il existe des constantes entières $c_0, c_1, \ldots, c_d$ telles que p divise $c_0 + c_1 a_{p-1} + \cdots + c_d a_{p-d}$ dès lors que p est un nombre premier. De plus, pour tout premier p, les constantes $c_i \bmod p$ peuvent être trouvées en $O(\mathsf{M}(d))$ opérations arithmétiques dans $\mathbb{A} = \mathbb{Z}/p\mathbb{Z}$. [Indication : pour A une matrice carrée d'entiers et p un nombre premier, la trace de A^p et celle de A sont congrues modulo p.]

Par exemple, si (F_n) est la suite de Fibonacci, alors p divise $F_{p-2} + 3F_{p-1} - 1$ dès lors que p est premier et la réciproque est vraie avec une bonne probabilité. L'exercice précédent fournit un test de primalité similaire au test de Fermat. D'après le Théorème 3, son coût est de $O(\mathsf{M}(d) \log p)$ opérations arithmétiques dans $\mathbb{Z}/p\mathbb{Z}$, soit $O(\mathsf{M}(d)\mathsf{M}_{\mathbb{Z}}(\log p) \log p)$ opérations binaires.

EXERCICE 7. Soient a et N deux entiers premiers entre eux. Montrer que N est premier *si et seulement si* $X^N + a = (X+a)^N \bmod N$ dans $\mathbb{Z}[X]$.

EXERCICE 8. Montrer que si N est premier, $0 \leq a < N$ et $P(X) \in \mathbb{Z}[X]$, alors $X^N + a = (X+a)^N \bmod P(X)$ dans $\mathbb{Z}/N\mathbb{Z}[X]$; si de plus, $P(X)$ est de degré $r = O(\log^c N)$, pour un $c > 0$, alors cette égalité peut être testée « en temps polynomial », c'est-à-dire en un nombre d'opérations binaires polynomial en $\log N$.

Notes

La suite de Fibonacci, introduite en 1202 par Leonardo Pisano (mieux connu sous le pseudonyme de Fibonacci) dans un problème récréatif décrivant la croissance d'une population de lapins jouit de riches propriétés algébriques, arithmétiques et combinatoires. Par exemple : (a) F_n est le nombre de façons différentes de paver un rectangle $2 \times (n-1)$ au moyen de dominos 2×1 ; (b) $(F_n)_n$ est une *suite de divisibilité, i. e.* F_n divise F_m dès lors que n divise m ; (c) si n est impair, alors

$$F_n = 2^{n-1} \prod_{k=1}^{\frac{n-1}{2}} \left(\frac{1}{4} + \cos^2 \frac{k\pi}{n} \right).$$

Exercice 9. Prouver les assertions (a)–(c).

Malgré sa simplicité, la suite de Fibonacci fait l'objet de nombreux problèmes ouverts. Par exemple, on ignore s'il existe une infinité de nombres de Fibonacci premiers. Les nombres de Fibonacci sont omniprésents en mathématiques et en informatique [2] : ils interviennent aussi bien dans l'analyse de l'algorithme d'Euclide pour le calcul du plus grand commun diviseur de deux entiers, que dans la solution négative du dixième problème de Hilbert par Matiyasevich [3].

Historiquement, le premier algorithme rapide pour la division des polynômes est dû à Moenck et Borodin [7]. Son point clé est que *le quotient de la division euclidienne de deux polynômes ne dépend que de leurs coefficients de poids fort.* Cette remarque est également à la base du calcul rapide de pgcd, étudié au Chapitre 6.

L'algorithme de la Section 2.2 est dû à Strassen [11]. Une alternative, de même complexité asymptotique, à l'algorithme esquissé en Section 2.4 pour les calculs modulaires a été proposée par Montgomery [8].

Calculer les N premiers termes d'une srlcc de polynôme caractéristique fixé est une opération linéaire en les conditions initiales ; c'est le *dual* de l'opération de division d'un polynôme de degré N par un polynôme fixé. Les conséquences algorithmiques de ce fait seront décrites au Chapitre 12.

Le théorème de Skolem-Mahler affirme que pour toute srlcc (a_n), l'ensemble de ses zéros (les indices i pour lesquels $a_i = 0$) est la réunion d'un ensemble fini et d'un nombre fini de suites arithmétiques. Son étude est une question subtile. Par exemple, déterminer si l'ensemble des zéros est vide est un problème NP-difficile [2]. Le livre d'Everest, van der Poorten, Shparlinski et Ward [4] est une excellente référence pour en savoir plus sur les questions reliées aux srlcc.

L'exercice 1 est le point clé de la méthode de Wiedemann pour la résolution de systèmes linéaires creux, traitée au Chapitre 9.

Le calcul rapide d'un terme de la suite de Fibonacci par exponentiation binaire de la matrice compagnon associée relève du folklore mathématique. Sa généralisation (exercice 4) est décrite par Miller et Brown [6], mais était probablement connue bien avant.

Le Théorème 4 et son Corollaire 4 sont dus à Fiduccia [5] et Shoup [10]. Leur récente généralisation au cas des matrices polynomiales est à la base du meilleur algorithme pour la résolution de systèmes linéaires à coefficients polynomiaux, exposé au Chapitre 11.

Il n'existe pas de tests *déterministes* de primalité basés sur le test modulaire d'un terme d'une récurrence à coefficients constants. Par contre, on peut caractériser un nombre premier N à l'aide de suites qui vérifient des récurrences à coefficients polynomiaux (comme la factorielle, via le test de Wilson $(N-1)! = -1 \bmod N$).

2. Le journal *The Fibonacci Quarterly* est entièrement dédié à l'étude de leurs propriétés.

3. Ce problème proposait de trouver un algorithme pour décider si un système d'équations diophantiennes (polynômes à coefficients entiers) admet une solution en nombres entiers.

Malheureusement, cela ne fournit pas d'algorithme efficace. Le premier algorithme déterministe qui prouve la primalité en temps polynomial est très récent [1] ; il part de la caractérisation de type Fermat donnée dans l'exercice 8, et exhibe une constante c et un polynôme P tels que la primalité de N est impliquée par la vérification de l'identité de l'Exercice 8 pour seulement $r^{1/2}\log(N)$ valeurs de a.

Bibliographie

[1] Agrawal, Manindra, Neeraj Kayal et Nitin Saxena (2004). « PRIMES is in P ». In : *Annals of Mathematics. Second Series*, vol. 160, n°2, p. 781–793.

[2] Blondel, Vincent D. et Natacha Portier (2002). « The presence of a zero in an integer linear recurrent sequence is NP-hard to decide ». In : *Linear Algebra and its Applications*, vol. 351/352. Fourth special issue on linear systems and control, p. 91–98.

[3] Cerlienco, L., M. Mignotte et F. Piras (1987). « Suites récurrentes linéaires. Propriétés algébriques et arithmétiques ». In : *L'Enseignement Mathématique. II*, vol. 33, p. 67–108.

[4] Everest, Graham, Alf van der Poorten, Igor Shparlinski et Thomas Ward (2003). *Recurrence sequences*. Vol. 104. Mathematical Surveys and Monographs. American Mathematical Society, xiv+318 pages.

[5] Fiduccia, C. M. (1985). « An efficient formula for linear recurrences ». In : *SIAM Journal on Computing*, vol. 14, n°1, p. 106–112.

[6] Miller, J. C. P. et D. J. Spencer Brown (1966). « An algorithm for evaluation of remote terms in a linear recurrence sequence ». In : *Computer Journal*, vol. 9, p. 188–190.

[7] Moenck, R. T. et A. Borodin (1972). « Fast modular transforms via division ». In : *Thirteenth Annual IEEE Symposium on Switching and Automata Theory*, p. 90–96.

[8] Montgomery, Peter L. (1985). « Modular multiplication without trial division ». In : *Mathematics of Computation*, vol. 44, n°170, p. 519–521.

[9] Poorten, A. J. van der (1989). « Some facts that should be better known, especially about rational functions ». In : *Number theory and applications*. Proceedings of a Conference held at Banff, AB, 1988. Dordrecht : Kluwer, p. 497–528.

[10] Shoup, V. (1991). « A Fast Deterministic Algorithm for Factoring Polynomials over Finite Fields of Small Characteristic ». In : *ISSAC'91 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 14–21.

[11] Strassen, V. (1972/73). « Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten ». In : *Numerische Mathematik*, vol. 20, p. 238–251.

CHAPITRE 5

Calculs modulaires, évaluation et interpolation

Résumé

Le concept d'algorithme modulaire est central en calcul formel. Il permet de pallier le phénomène d'explosion des expressions intermédiaires. Des cas particuliers importants sont l'évaluation multipoint et l'interpolation, pour lesquels il existe des algorithmes rapides qui peuvent être vus comme des généralisations de la FFT.

1. Introduction

Un ingrédient essentiel en calcul formel est l'utilisation de différents types de représentation pour les objets manipulés. Par exemple, un polynôme est classiquement codé par la liste de ses coefficients, mais on peut très bien le représenter par les valeurs qu'il prend en un nombre suffisant de points. D'ailleurs, nous avons vu au Chapitre 2 que c'est bien cette seconde représentation qui est la plus adaptée pour le calcul efficace, via la FFT, du produit de polynômes. Par ailleurs, d'autres opérations (comme la division polynomiale, traitée au Chapitre 4) se font mieux dans la première représentation. D'autres exemples de codages alternatifs déjà rencontrés dans ce cours sont l'écriture des entiers en différentes bases de numération, ou encore la représentation des suites récurrentes linéaires à coefficients constants, soit par leurs éléments, soit par leurs séries génératrices, soit par une récurrence et des conditions initiales.

L'exemple de la FFT montre à quel point il est crucial d'examiner dans quelle représentation un problème donné est plus facile à traiter, et aussi, de trouver des algorithmes rapides pour la conversion d'une représentation à l'autre.

Une incarnation de ce concept général est la notion d'*algorithme modulaire*, dont le paradigme *évaluation-interpolation* est un cas particulier très important. L'approche modulaire consiste à choisir des *modulos* m_i, à faire des calculs modulo chaque m_i et à reconstruire tout à la fin le résultat à l'aide d'une version effective du *théorème des restes chinois*. Pour garantir l'unicité du résultat et la correction de ce schéma, il suffit que les modulos soient suffisamment *nombreux et indépendants*. Typiquement, s'il s'agit d'entiers, ils peuvent être choisis premiers entre eux et tels que leur produit dépasse le résultat final. En particulier, pour mettre en place une approche modulaire, on a besoin de bornes *a priori* sur la taille de l'objet calculé.

Cette approche porte ses fruits surtout lorsque les coefficients dans les calculs intermédiaires sont beaucoup plus gros que ceux du résultat final. Il s'agit du phénomène d'*explosion des expressions intermédiaires*, qui se présente par exemple dans le calcul du déterminant d'une matrice entière ou polynomiale, ou encore au cours du calcul du pgcd de polynômes à coefficients entiers.

Exemple 1 (calcul du déterminant d'une matrice polynomiale). Soit à calculer le déterminant d'une matrice $n \times n$, dont les éléments sont des polynômes de degré au plus d (le cas particulier $d = 1$ correspond au calcul d'un polynôme caractéristique).

Le point de départ est la remarque que le pivot de Gauss produit sur la i^{e} ligne des fractions rationnelles de degré $2^i d$. Ainsi, sa complexité ne semble pas polynomiale par rapport à n. Une approche évaluation-interpolation résout cette anomalie. Le déterminant étant un polynôme de degré au plus nd, il suffit de choisir un ensemble de $nd+1$ points, d'y évaluer les éléments de la matrice, de calculer les $nd+1$ déterminants de matrices scalaires et de finir par une interpolation fournissant le determinant cherché. Le même raisonnement s'applique aux matrices entières.

Choix des modulos. Dans certaines situations, le programmeur a le choix des modulos. Dans l'exemple précédent, on peut choisir comme modulos des polynômes de la forme $X-\omega^i$, dans le cas favorable où l'anneau de base contient une racine principale de l'unité ω d'ordre suffisamment élevé ($\approx 2dn$). En théorie, ce choix est donc possible dès lors que l'anneau de base permet une multiplication polynomiale à base de FFT. En pratique, il existe des plages de degrés pour lesquelles, même si elle est faisable, la FFT n'est pas encore rentable ; ce choix est alors déconseillé. Dans d'autres situations, le choix des modulos est intrinsèquement imposé par le problème à traiter ; c'est le cas pour le calcul de la factorielle exposé au Chapitre 15.

2. Présentation, résultats

Les problèmes d'évaluation et d'interpolation sont inverses l'un de l'autre. Dans leur version standard, ils s'énoncent ainsi. Soient $a_0,\dots,a_{n-1}$ des points dans $\mathbb{A}$, où $\mathbb{A}$ est un anneau commutatif.

Évaluation multipoint. Étant donné un polynôme P dans $\mathbb{A}[X]$, de degré strictement inférieur à n, calculer les valeurs :

$$P(a_0),\dots,P(a_{n-1}).$$

Interpolation. Étant donnés $b_0,\dots,b_{n-1}\in\mathbb{A}$, trouver un polynôme $P\in\mathbb{A}[X]$ de degré strictement inférieur à n tel que

$$P(a_0)=b_0,\dots,P(a_{n-1})=b_{n-1}.$$

Le problème d'interpolation admet toujours une solution unique sous l'hypothèse technique suivante :

(H) $\qquad a_i-a_j$ est inversible dans $\mathbb{A}$ lorsque $i\neq j$.

En effet, si $\mathbf{V}=\left(a_{i-1}^{j-1}\right)_{i,j=1}^{n}$ est la matrice de Vandermonde associée aux points a_i, dont le déterminant vaut $\det(\mathbf{V})=\prod_{i<j}\left(a_j-a_i\right)$, alors le problème d'interpolation se traduit par la résolution en l'inconnue $\mathbf{p}$ du système linéaire $\mathbf{Vp}=\mathbf{b}$, où $\mathbf{b}$ est le vecteur des b_i. Or, ce système admet une solution unique, puisque l'hypothèse (H) entraîne l'inversibilité du déterminant $\det(\mathbf{V})$ et donc aussi celle de la matrice $\mathbf{V}$.

Exercice 1. Montrer que $\det(\mathbf{V})=\prod_{i<j}(a_j-a_i)$.

Cette discussion suggère un algorithme naïf pour l'interpolation, produisant l'unique solution $\mathbf{p}=\mathbf{V}^{-1}\mathbf{b}$ du système $\mathbf{Vp}=\mathbf{b}$ à l'aide du pivot de Gauss en $O(n^3)$ opérations dans $\mathbb{A}$, ou encore en $O(\mathsf{MM}(n))=O(n^\theta)$ opérations ($2\le\theta<3$), en utilisant l'algorithmique matricielle rapide (Chapitre 8). De manière analogue, l'évaluation multipoint de P en les a_i se traduit par le produit matrice-vecteur $\mathbf{Vp}$, où $\mathbf{p}$ est le vecteur des coefficients de P ; elle peut donc être effectuée de manière naïve en $O(n^2)$ opérations arithmétiques. En s'y prenant *vraiment naïvement*, l'interpolation est donc plus coûteuse que l'évaluation multipoint. Nous verrons un peu plus loin qu'une méthode exploitant *la formule d'interpolation de Lagrange* permet de résoudre le problème d'interpolation en complexité *quadratique* en n.

L'objectif de ce chapitre est de montrer qu'il est possible d'atteindre une complexité quasi-optimale, tant pour l'évaluation multipoint que pour l'interpolation, en utilisant des algorithmes de type « diviser pour régner » qui ramènent ces questions à des multiplications de polynômes. Les principaux résultats sont les suivants :

THÉORÈME 1 (« Évaluation-interpolation »). *On peut effectuer l'évaluation et l'interpolation sur n points vérifiant l'hypothèse* (H) *en utilisant $O(\mathsf{M}(n)\log n)$ opérations $(+,-,\times)$ de $\mathbb{A}$. Cette complexité peut être abaissée à $O(\mathsf{M}(n))$ si les points forment une progression géométrique de raison inversible dans $\mathbb{A}$.*

En termes pratiques, si l'anneau de base $\mathbb{A}$ est un corps fini, les algorithmes rapides deviennent avantageux pour des degrés de l'ordre de quelques dizaines : c'est sensiblement mieux que pour les algorithmes rapides de type « diviser pour régner » utilisés aux Chapitres 6 et 7 pour le calcul rapide de pgcd ou d'approximants de Padé-Hermite ; cela reflète une relative simplicité des algorithmes.

Extensions. L'évaluation multipoint et l'interpolation sont des instances particulières des problèmes *modulos multiples* et *restes chinois* s'énonçant comme suit :

Modulos multiples. Étant donnés des polynômes $P, m_1, \ldots, m_r$ dans $\mathbb{A}[X]$, avec $\deg(P) < n = \sum_i \deg(m_i)$, calculer les restes :

$$P \bmod m_1, \ldots, P \bmod m_r.$$

Restes chinois. Étant donnés des polynômes $b_1, \ldots, b_r, m_1, \ldots, m_r$ dans $\mathbb{A}[X]$, avec m_i unitaires, trouver $P \in \mathbb{A}[X]$ de degré inférieur à $n = \sum_i \deg(m_i)$ tel que

$$P \bmod m_1 = b_1, \ldots, P \bmod m_r = b_r.$$

Les techniques de ce chapitre s'y généralisent [1] et mènent aux résultats suivants :

THÉORÈME 2. *On peut résoudre les deux problèmes ci-dessus en $O(\mathsf{M}(n)\log n)$ opérations $(+,-,\times)$ dans $\mathbb{A}$.*

Naturellement, on peut se poser les questions précédentes dans le cas où l'anneau de polynômes $\mathbb{A}[X]$ est remplacé par l'anneau $\mathbb{Z}$, les polynômes m_i sont remplacés par des modulos entiers a_i et où l'on cherche un $P \in \mathbb{Z}$ à n chiffres. Il est possible d'obtenir le même type de résultats de complexité, cette fois en comptant les opérations binaires, mais nous nous limiterons dans la suite au cas polynomial.

La suite de ce chapitre est organisée comme suit : dans la Section 3 est décrite la méthode d'interpolation de Lagrange, de complexité quadratique. La Section 4 est consacrée aux algorithmes rapides sous-jacents à la preuve du Théorème 1.

3. Interpolation de Lagrange

Un algorithme de complexité quadratique pour l'interpolation polynomiale repose sur une écriture explicite du polynôme interpolant. Soient b_i les valeurs à interpoler et a_i les points d'interpolation vérifiant l'hypothèse (H). On peut alors écrire P sous la forme :

$$P(X) \;=\; \sum_{i=0}^{n-1} b_i \prod_{0\le j\le n-1,\ j\neq i} \frac{X-a_j}{a_i-a_j}.$$

1. Pour les restes chinois, l'unicité du résultat est garantie par l'hypothèse que le résultant $\mathrm{Res}(m_i, m_j)$ est un élément inversible dans $\mathbb{A}$, pour tous $i \neq j$. Cette condition technique généralise la condition d'inversibilité des $a_i - a_j$; se rapporter au Chapitre 6 pour la définition du résultant.

Cette égalité est usuellement appelée la *formule d'interpolation de Lagrange*. Pour la prouver, il suffit d'observer que pour tout i, le produit s'annule en a_j ($j \neq i$), et vaut 1 en a_i. Afin de simplifier l'écriture de la formule de Lagrange, posons

$$A = \prod_j (X - a_j) \quad \text{et} \quad A_i = \prod_{j \neq i} (X - a_j) = \frac{A}{X - a_i},$$

pour $i = 0, \ldots, n-1$. Pour $i \neq j$, on a donc les relations $A(a_i) = 0$ et $A_i(a_j) = 0$. On obtient alors l'écriture

$$P = \sum_{i=0}^{n-1} b_i \frac{A_i(X)}{A_i(a_i)}.$$

Une approche directe pour l'interpolation revient à calculer tout d'abord les polynômes A_i, puis leurs valeurs en les points a_i, et enfin à effectuer les combinaisons linéaires nécessaires. Le seul point non-trivial est le calcul des A_i : si on n'y fait pas attention, on risque de sortir des bornes en $O(n^2)$ (par exemple, si on les calcule tous indépendamment, et chacun de manière quadratique). Pour faire mieux, on partage le gros des calculs, en calculant d'abord le polynôme A.

Interpolation de Lagrange

Entrée : $a_0, \ldots, a_{n-1} \in \mathbb{A}$ vérifiant (H) et $b_0, \ldots, b_{n-1}$ dans $\mathbb{A}$.

Sortie : L'unique polynôme $P \in \mathbb{A}[X]$ de degré inférieur à n tel que $P(a_i) = b_i$ pour tout i.

1. $A \leftarrow 1$, $P \leftarrow 0$
2. Pour $i = 0, \ldots, n-1$ faire
   $A \leftarrow A \cdot (X - a_i)$
3. Pour $i = 0, \ldots, n-1$ faire
   $A_i \leftarrow A/(X - a_i)$
   $q_i \leftarrow A_i(a_i)$
   $P \leftarrow P + b_i A_i / q_i$

PROPOSITION 1. *L'algorithme ci-dessus utilise $O(n^2)$ opérations dans $\mathbb{A}$.*

DÉMONSTRATION. La multiplication d'un polynôme de degré d par un polynôme de degré 1 prend un temps linéaire en d, disons Cd opérations, C étant une constante. Calculer A demande donc $C(1 + 2 + \cdots + (n-1)) = O(n^2)$ opérations. Ensuite, chaque passage dans la seconde boucle prend un temps linéaire en n. Il y a n passages, d'où à nouveau un coût quadratique. □

4. Algorithmes rapides

4.1. Les idées. L'idée fondamentale de l'algorithme d'évaluation multipoint rapide est d'utiliser une technique de type « diviser pour régner ». Supposons pour simplifier que n est pair et séparons l'ensemble des points $a_0, \ldots, a_{n-1}$ en deux paquets, $a_0, \ldots, a_{n/2-1}$ et $a_{n/2}, \ldots, a_{n-1}$. Il est naturel d'associer à P deux polynômes P_0 et P_1 de degré strictement inférieur à $n/2$, tels que :

$$P_0(a_0) = P(a_0), \ldots, P_0(a_{n/2-1}) = P(a_{n/2-1}) \quad \text{et}$$
$$P_1(a_{n/2}) = P(a_{n/2}), \ldots, P_1(a_{n-1}) = P(a_{n-1}).$$

Pour effectuer cette construction, le choix suivant s'impose :

$$P_0 = P \bmod (X - a_0) \cdots (X - a_{n/2-1})$$
$$P_1 = P \bmod (X - a_{n/2}) \cdots (X - a_{n-1}).$$

On a donc ramené le calcul en degré n à deux calculs en degré $n/2$, plus deux divisions euclidiennes ; on va ensuite itérer ce processus. En bout de course, on retrouve les valeurs $P(a_i)$, car celles-ci peuvent également s'écrire $P \bmod (X - a_i)$.

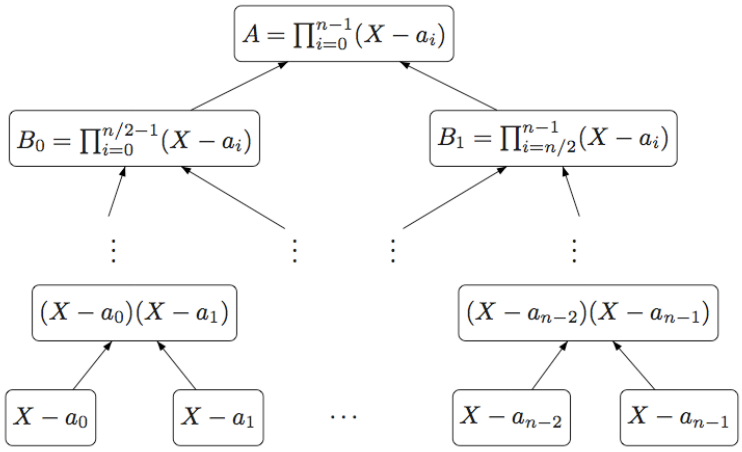
Remarquons que les polynômes par lesquels on effectue les divisions euclidiennes ne sont pas « gratuits » : on doit de les (pré)calculer. Pour cela, l'idée est de les empiler dans une structure d'arbre binaire, dont les nœuds internes sont étiquetés par des polynômes.

4.2. L'arbre des sous-produits. Pour simplifier, dans tout ce qui suit, on suppose que le nombre de points est une puissance de 2, $n = 2^\kappa$. Quand ce n'est pas le cas, on adapte toutes les constructions (on fabrique toujours un arbre binaire, mais il lui manque alors des nœuds internes à certains niveaux). On peut donner la définition récursive suivante pour cet arbre (noté $\mathcal{B}$) :

— Si $\kappa = 0$, $\mathcal{B}$ se réduit à un nœud unique qui contient le polynôme $X - a_0$.

— Sinon, soient $\mathcal{B}_0, \mathcal{B}_1$ les arbres associés à $a_0, \ldots, a_{2^{\kappa-1}-1}$ et $a_{2^{\kappa-1}}, \ldots, a_{2^\kappa-1}$. Soient B_0 et B_1 les polynômes présents aux racines de $\mathcal{B}_0$ et $\mathcal{B}_1$. Alors $\mathcal{B}$ est l'arbre dont la racine contient le produit $B_0 B_1$ et dont les fils sont $\mathcal{B}_0$ et $\mathcal{B}_1$.

Graphiquement, on a la représentation suivante :



L'intérêt de cette structure d'arbre est double : d'une part, il contient tous les polynômes par lesquels on va diviser lors de l'algorithme d'évaluation, d'autre part, le coût de son calcul s'amortit, et devient essentiellement linéaire en le degré.

Proposition 2. *L'algorithme* ArbreSousProduits *calcule tous les polynômes contenus dans l'arbre $\mathcal{B}$ pour $O(\mathsf{M}(n)\log n)$ opérations arithmétiques dans $\mathbb{A}$.*

Démonstration. Soit $\mathsf{T}(n)$ le coût du calcul de l'arbre pour n points. La définition récursive implique l'inégalité suivante pour $\mathsf{T}(n)$:

$$\mathsf{T}(n) \leq 2\mathsf{T}\left(\frac{n}{2}\right) + \mathsf{M}\left(\frac{n}{2}\right).$$

Le résultat s'ensuit en invoquant le théorème « diviser pour régner ». □

ArbreSousProduits

Entrée : $a_0, \dots, a_{n-1}$ dans $\mathbb{A}$, avec n de la forme $n = 2^\kappa$ ($\kappa \in \mathbb{N}$).

Sortie : l'arbre $\mathcal{B}$ des sous-produits associé aux points a_i.

1. Si $n = 1$, renvoyer $X - a_0$
2. Calculer (par un appel récursif) $\mathcal{B}_0 = \text{ArbreSousProduits}(a_0, \dots, a_{n/2-1})$
3. Calculer (par un appel récursif) $\mathcal{B}_1 = \text{ArbreSousProduits}(a_{n/2}, \dots, a_{n-1})$
4. Calculer $A = B_0 B_1$, où B_0 et B_1 sont les racines de $\mathcal{B}_0$ et de $\mathcal{B}_1$.
5. Renvoyer l'arbre de racine A et dont les fils sont $\mathcal{B}_0$ et $\mathcal{B}_1$.

FIGURE 1. Algorithme rapide pour le calcul de l'arbre des sous-produits.

EvaluationRapide

Entrée : $a_0, \dots, a_{n-1}$ dans $\mathbb{A}$, avec n de la forme $n = 2^\kappa$ ($\kappa \in \mathbb{N}$),
l'arbre $\mathcal{B}$ des sous-produits associé aux points a_i,
P dans $\mathbb{A}[X]$, avec $\deg P < n$.

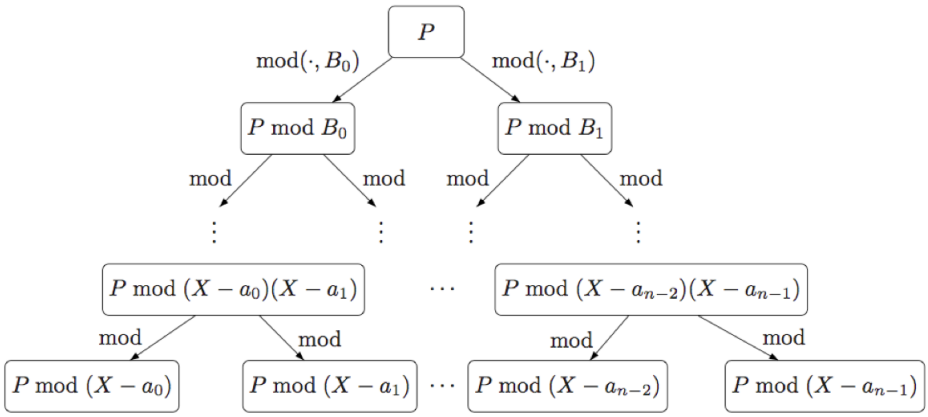
Sortie : $P(a_0), \dots, P(a_{n-1})$.

1. Si $n = 1$, renvoyer P
2. $P_0 \leftarrow P \bmod (X - a_0) \cdots (X - a_{n/2-1})$
3. $P_1 \leftarrow P \bmod (X - a_{n/2}) \cdots (X - a_{n-1})$
4. Calculer (par un appel récursif) $L_0 = P_0(a_0), \dots, P_0(a_{n/2-1})$
5. Calculer (par un appel récursif) $L_1 = P_1(a_{n/2}), \dots, P_1(a_{n-1})$
6. Renvoyer L_0, L_1

FIGURE 2. Algorithme rapide d'évaluation multipoint par division répétée.

4.3. Évaluation multipoint rapide. L'algorithme d'évaluation multipoint devient simple à écrire de manière récursive, si on s'autorise un peu de souplesse dans l'écriture, en supposant précalculés tous les nœuds de l'arbre.

Graphiquement, cela correspond à faire *descendre* les restes de divisions euclidiennes dans l'arbre des sous-produits :



Or, la division avec reste par un polynôme unitaire de degré n peut s'effectuer en $O(\mathsf{M}(n))$ opérations de $\mathbb{A}$ (Chapitre 4). Nous obtenons le résultat de complexité suivant :

PROPOSITION 3. *La complexité de l'algorithme* EvaluationRapide (Figure 2) *est de* $O(\mathsf{M}(n)\log n)$ *opérations dans* $\mathbb{A}$.

DÉMONSTRATION. Soit $\mathsf{T}(n)$ le coût du calcul pour n points. L'algorithme ci-dessus implique la récurrence suivante pour $\mathsf{T}(n)$:

$$\mathsf{T}(n) \leq 2\,\mathsf{T}\left(\frac{n}{2}\right) + O(\mathsf{M}(n)),$$

et le résultat en découle grâce au théorème « diviser pour régner ». □

EXERCICE 2. Si $\omega \in \mathbb{A}$ est une racine principale n^{e} de l'unité, montrer que l'algorithme FFT vu au Chapitre 2 est un cas particulier de l'algorithme en Figure 2.

EXERCICE 3. Estimer la complexité de la variante de l'algorithme EvaluationRapide dans laquelle l'arbre des sous-produits n'est pas précalculé, les polynômes modulo lesquels s'effectuent les divisions étant calculés à la volée.

4.4. Sommes de fractions. Un problème intermédiaire à traiter avant de résoudre efficacement le problème d'interpolation est celui du calcul rapide d'une somme de fractions. Rappelons les définitions introduites dans la Section 3. À partir des points a_i, nous y avions défini les polynômes

$$A = \prod_j (X - a_j) \qquad \text{et} \qquad A_i = \prod_{j \neq i} (X - a_j) = \frac{A}{X - a_i}.$$

Il était apparu que pour effectuer l'interpolation, il suffisait de savoir effectuer la tâche suivante : étant données des valeurs c_i, calculer le numérateur et le dénominateur de la fraction rationnelle

$$\sum_{i=0}^{n-1} \frac{c_i}{X - a_i}. \tag{1}$$

C'est à cette question que nous allons répondre maintenant ; à nouveau, on utilise une idée de type « diviser pour régner » : par deux appels récursifs, on calcule

$$S_1 = \sum_{i=0}^{n/2-1} \frac{c_i}{X - a_i} \quad \text{et} \quad S_2 = \sum_{i=n/2}^{n} \frac{c_i}{X - a_i}$$

et on renvoie $S = S_1 + S_2$. Soit $\mathsf{T}(n)$ le coût du calcul pour n points. Le coût de l'algorithme SommesFractions esquissé ci-dessus satisfait à la récurrence :

$$\mathsf{T}(n) \leq 2\mathsf{T}\left(\frac{n}{2}\right) + O(\mathsf{M}(n)),$$

d'où l'on déduit le résultat suivant, en faisant encore une fois appel au théorème « diviser pour régner ».

PROPOSITION 4. *L'algorithme* SommesFractions *calcule le dénominateur et le numérateur de la fraction rationnelle* (1) *en* $O(\mathsf{M}(n)\log n)$ *opérations dans* $\mathbb{A}$.

4.5. Interpolation. À ce stade, l'interpolation ne pose plus de réelle difficulté. Au vu des formules de la Section 3, le polynôme interpolant les valeurs b_i aux points a_i est donné par :

$$P(X) \;=\; \sum_{i=0}^{n-1} b_i \frac{A_i(X)}{A_i(a_i)} \;=\; A(X) \times \sum_{i=0}^{n-1} \frac{b_i / A_i(a_i)}{X - a_i}.$$

InterpolationRapide

Entrée : $a_0, \dots, a_{n-1} \in \mathbb{A}$ vérifiant (H) et $b_0, \dots, b_{n-1}$ dans $\mathbb{A}$.

Sortie : L'unique polynôme $P \in \mathbb{A}[X]$ de degré inférieur à n tel que $P(a_i) = b_i$ pour tout i.

1. Calculer l'arbre des sous-produits associé aux points a_i (de racine A).
2. $(d_0, \dots, d_{n-1}) \leftarrow \mathsf{EvaluationRapide}(A', (a_0, \dots, a_{n-1}))$
3. $(c_0, \dots, c_{n-1}) \leftarrow (b_0/d_0, \dots, b_{n-1}/d_{n-1})$
4. $R \leftarrow \mathsf{SommesFractions}(c_0/(X - a_0), \dots, c_{n-1}/(X - a_{n-1}))$
5. Renvoyer le numérateur de R.

FIGURE 3. Algorithme d'interpolation rapide.

D'après les paragraphes précédents, on sait comment calculer rapidement la fraction rationnelle $\sum c_i/(X-a_i)$; il ne reste plus qu'à calculer les constantes $c_i = b_i/A_i(a_i)$. Le calcul itératif des constantes c_i reviendrait à évaluer n polynômes différents, chacun en un point, et aurait donc une complexité trop élevée, quadratique en n. La proposition suivante permet de remplacer ces n évaluations par une évaluation multipoint d'un unique polynôme en n points, ce qui mène à un gain de complexité.

PROPOSITION 5. *Pour tout $i = 0, \dots, n-1$, on a $A_i(a_i) = A'(a_i)$.*

DÉMONSTRATION. On dérive A, ce qui fournit l'égalité :

$$A' = \sum_{i=0}^{n-1} \prod_{j \neq i} (X - a_j) = \sum_{i=0}^{n-1} A_i.$$

On a vu précédemment que $A_i(a_j) = 0$ pour $i \neq j$, ce qui donne le résultat. □

L'algorithme d'interpolation et l'estimation de son coût s'en déduisent facilement.

PROPOSITION 6. *La complexité de l'algorithme* InterpolationRapide *en Figure 3 est $O(\mathsf{M}(n) \log n)$ opérations de $\mathbb{A}$.*

DÉMONSTRATION. C'est une conséquence des Propositions 2, 3 et 4. □

4.6. Évaluation et interpolation sur une suite géométrique. Dans cette section nous montrons que tout polynôme de degré n peut être évalué et interpolé sur des points en progression géométrique $\{1, q, \dots, q^{n-1}\}$ en $O(\mathsf{M}(n))$ opérations.

PROPOSITION 7. *Soit q un élément inversible de $\mathbb{A}$, et $a_i = q^i$ pour $0 \leq i < n$. Alors :*

1. *On peut évaluer tout polynôme $P \in \mathbb{A}[X]$ de degré strictement inférieur à n sur les points $a_0, \dots, a_{n-1}$ en $O(\mathsf{M}(n))$ opérations dans $\mathbb{A}$.*
2. *Si les points (a_i) vérifient l'hypothèse (H) on peut effectuer l'interpolation en les points $a_0, \dots, a_{n-1}$ en $O(\mathsf{M}(n))$ opérations dans $\mathbb{A}$.*

DÉMONSTRATION. Soit $P = p_0 + p_1 X + \cdots + p_{n-1} X^{n-1}$. Pour $i = 0, \dots, 2n-2$, on introduit les nombres triangulaires $t_i = i(i-1)/2$ et la suite $\beta_i = q^{t_i}$; pour $i = 0, \dots, n-1$ on construit $\gamma_i = p_i/\beta_i$. Tous les éléments q^{t_i}, γ_i et β_i peuvent être calculés en $O(n)$ opérations, en utilisant l'égalité $q^{t_{i+1}} = q^i q^{t_i}$. L'observation clé est que $ij = t_{i+j} - t_i - t_j$ et donc $q^{ij} = \beta_i^{-1} \beta_j^{-1} \beta_{i+j}$. L'algorithme d'évaluation repose alors sur la formule

$$P(q^i) = \sum_{j=0}^{n-1} p_j q^{ij} = \beta_i^{-1} \cdot \sum_{j=0}^{n-1} \gamma_j \beta_{i+j},$$

qui montre que les valeurs $P(q^i)$ sont, à des facteurs constants près, données par les coefficients de $X^{n-1}, \ldots, X^{2n-2}$ dans le produit de $\sum_{i=0}^{n-1} \gamma_i X^{n-i-1}$ et $\sum_{i=0}^{2n-2} \beta_i X^i$.

Pour l'interpolation, on adapte l'algorithme InterpolationRapide en Figure 3, de façon à économiser un facteur log en tirant parti de la structure des points $a_i = q^i$.

On commence par remarquer que *la racine* $A(X)$ de l'arbre des sous-produits peut être calculée en $O(\mathsf{M}(n))$ opérations dans $\mathbb{A}$, par un algorithme *qui ne nécessite pas la construction de tout l'arbre des sous-produits*, donc différent de l'algorithme donné en Figure 1. En effet, puisque les points a_i forment une suite géométrique, il est possible de calculer récursivement $A = \prod_{i=0}^{n-1}(X - a_i)$ par *un seul appel récursif* calculant $B_0 = \prod_{i=0}^{n/2-1}(X - a_i)$, suivi du calcul de $B_1 = \prod_{i=n/2}^{n-1}(X - a_i)$ à partir de B_0 par une homothétie de coût $O(n)$, et enfin du calcul du produit $A = B_0 B_1$.

Par ailleurs, observons que dans l'algorithme InterpolationRapide, le calcul de tout l'arbre des sous-produits a été utilisé *uniquement* pour l'évaluation multipoint de A'. Or, par la première partie de la Proposition 7, l'évaluation de A' sur les points a_i peut s'effectuer en $O(\mathsf{M}(n))$ opérations.

Il nous reste à prouver que la somme des fractions (1), qui devient ici

$$\frac{N}{A} = \sum_{i=0}^{n-1} \frac{c_i}{X - q^i} \tag{2}$$

peut également être déterminée pour le même prix.

On adopte la stratégie suivante : on calcule le développement en série à l'ordre n de la fraction (2). Celui-ci suffit pour en déduire son numérateur $N(X)$. Pour ce faire, on utilise la formule $1/(a - X) = \sum_{j \geq 0} a^{-j-1} X^j$ valable dans $\mathbb{A}[[X]]$ pour tout $a \in \mathbb{A}$ inversible. On obtient :

$$\sum_{i=0}^{n-1} \frac{c_i}{X - q^i} \mod X^n = -\sum_{i=0}^{n-1} \left(\sum_{j=0}^{n-1} c_i q^{-i(j+1)} X^j \right) = -\sum_{j=0}^{n-1} C(q^{-j-1}) X^j,$$

où $C(X)$ est le polynôme $\sum_{i=0}^{n-1} c_i X^i$. Or, les points $q^{-1}, q^{-2}, \ldots, q^{-n}$ étant en progression géométrique, on sait évaluer C sur ces points en $O(\mathsf{M}(n))$. □

Exercice 4. Montrer que tout l'arbre des sous-produits associé aux points $a_i = q^i$, $i = 0, \ldots, n-1$, peut être calculé en $\mathsf{M}(n) + O(n \log n)$ opérations dans $\mathbb{A}$.

Une approche évaluation-interpolation reposant sur la Proposition 7 permet d'obtenir un bon algorithme pour la multiplication des matrices polynomiales.

Corollaire 1. *Soit $\mathbb{K}$ un corps possédant au moins $2d$ éléments. Il est possible de multiplier deux matrices polynomiales de $\mathcal{M}_n(\mathbb{K}[X])$ de degré au plus d en $\mathsf{MM}(n, d) = O(d\,\mathsf{MM}(n) + n^2\,\mathsf{M}(d))$ opérations dans $\mathbb{K}$.*

Exercices

Exercice 5 (Calcul rapide de sommes de puissances). Soient $x_1, x_2, \ldots, x_n$ des éléments de $\mathbb{K}$ et soit $P_k = \sum_{i=1}^{n} x_i^k$ pour $k \geq 1$. Montrer que, à partir de la donnée des $x_1, \ldots, x_n$, il est possible de calculer :

1. tous les P_k dont les indices $k \leq n$ sont une puissance de 2 en $O(n \log(n))$ opérations dans $\mathbb{K}$.
2. tous les $P_1, \ldots, P_n$ en $O(\mathsf{M}(n) \log(n))$ opérations dans $\mathbb{K}$.

Exercice 6. Estimer la complexité de l'algorithme direct pour évaluer un polynôme de degré au plus n et ses premières n dérivées en un point. Imaginer un algorithme de complexité quasi-linéaire résolvant ce problème.

EXERCICE 7. Soient $a, b, c, d \in \mathbb{A}$. Montrer qu'on peut évaluer tout polynôme de degré au plus n en $\{ba^{2i} + ca^i + d, \quad 0 \leq i < n\}$, en $O(\mathsf{M}(n))$ opérations dans $\mathbb{A}$.

EXERCICE 8. Soit $\mathbb{K}$ un corps, soient $m_1, m_2 \in \mathbb{K}[X]$ des polynômes premiers entre eux, de degrés au plus n, et soient $v_1, v_2 \in \mathbb{K}[X]$ de degrés inférieurs à n. Donner un algorithme de complexité $O(\mathsf{M}(n) \log n)$ qui calcule $f \in \mathbb{K}[X]$ de degré inférieur à $2n$ tel que $f = v_1 \bmod m_1$ et $f = v_2 \bmod m_2$. [Indication : utiliser un pgcd étendu.]

En déduire un algorithme de type « diviser pour régner » pour l'interpolation polynomiale en degré n, de complexité $O(\mathsf{M}(n) \log^2 n)$.

EXERCICE 9 (Multiplication de polynômes à deux variables par substitution de Kronecker). Soient f et g dans $\mathbb{K}[X, Y]$ de degrés au plus d_X en X et au plus d_Y en Y.

1. Montrer qu'il est possible de calculer tous les coefficients de $h = fg$ en $O(\mathsf{M}(d_X d_Y))$ opérations dans $\mathbb{K}$. Utiliser la substitution $X \leftarrow Y^{2d_Y+1}$ pour se ramener à une multiplication de polynômes à une variable.
2. Améliorer ce résultat en donnant un schéma d'évaluation-interpolation qui permette le calcul des coefficients de h en $O(d_X \mathsf{M}(d_Y) + d_Y \mathsf{M}(d_X))$ opérations de $\mathbb{K}$ si $\mathbb{K}$ est assez grand.

Notes

Pour évaluer un polynôme $P(X) = p_{n-1}X^{n-1} + \cdots + p_0$ en un seul point a, le *schéma de Horner* $P(a) = (\cdots((p_{n-1}a + p_{n-2})a + p_{n-3})a + \cdots + p_0)$ minimise le nombre d'opérations (additions ou multiplications). Cet algorithme effectue $n-1$ additions et $n-1$ multiplications dans $\mathbb{A}$, et on ne peut pas faire mieux en général (si les coefficients du polynôme sont algébriquement indépendants — moralement, s'ils sont des indéterminées). Ce résultat d'optimalité a été conjecturé par Ostrowski [15] et prouvé par Pan [16]. Par contre, pour un polynôme donné, il peut être possible de faire mieux : l'évaluation de $P(X) = X^{n-1}$ se fait en temps logarithmique en n.

Une forme particulière du théorème des restes chinois a été enoncée il y a plus de 2000 ans par le mathématicien chinois Sun Tsu. Le traitement moderne est dû à Gauss [9]. Ce théorème admet une formulation très générale, en termes d'idéaux d'anneaux non nécessairement commutatifs : Si R est un anneau et $I_1, \ldots, I_r$ des idéaux (à gauche) de R mutuellement premiers (*i. e.* $I_i + I_j = R$ pour $i < j$), alors l'anneau quotient $R/\cap_i I$ est isomorphe à l'anneau produit $\prod R/I_i$ via l'isomorphisme $x \bmod I \quad \mapsto (x \bmod I_1, \ldots, x \bmod I_r)$. La formule d'interpolation de Lagrange est due à Waring [23]. Les avantages pratiques de l'arithmétique modulaire et les premières applications en informatique du théorème des restes chinois ont été mis en évidence dans les années 1950 par Svoboda et Valach [21], et indépendamment par Garner [8] et Takahasi et Ishibashi [22].

Le premier algorithme sous-quadratique, de complexité arithmétique $O\big(n^{1,91}\big)$, pour l'évaluation multipoint est dû à Borodin et Munro [4] et repose sur une approche *pas de bébés / pas de géants* exploitant la multiplication sous-cubique des matrices de Strassen [19]. Horowitz [11] a étendu ce résultat à l'interpolation. La Proposition 2 calcule efficacement les fonctions symétriques élémentaires de n éléments $a_0, \ldots, a_{n-1}$ de $\mathbb{A}$; elle est due également à Horowitz [11].

S'inspirant de la FFT, Fiduccia [7] eut l'idée d'un algorithme pour l'évaluation multipoint à base de division polynomiale récursive. Ne disposant pas de division de complexité sous-quadratique, son algorithme récursif reste quadratique. Borodin et Moenck corrigent ce point dans deux articles successifs [3, 13], reposant sur les algorithmes quasi-optimaux pour la division de [13, 20]. Montgomery [14] améliore

la constante dans la complexité $O\big(\mathsf{M}(n)\log n\big)$ de l'évaluation multipoint, sous l'hypothèse que la FFT est utilisée pour la multiplication polynomiale. L'algorithme d'évaluation sur une suite géométrique exposé dans ce chapitre vient de [2, 17]. À l'origine, cet algorithme a permis de montrer qu'une DFT_ω, pour ω racine primitive n^{e} de l'unité, peut être effectuée en $O(n\log n)$ opérations même si n n'est pas une puissance de 2. L'algorithme d'interpolation est une adaptation de [12]. Les algorithmes fournissant les meilleures constantes actuellement connues dans les $O(\cdot)$ du Théorème 1 sont obtenus à l'aide du principe de transposition de Tellegen [5, 6].

On connaît peu de choses sur la complexité intrinsèque de l'évaluation multipoint et de l'interpolation en degré n. Strassen [20] a prouvé une borne inférieure de type $n\log n$. Shoup et Smolensky [18] ont montré que ces bornes restent essentiellement vraies même dans un modèle où des précalculs en quantité illimitée sur les points d'évaluation sont permis. Cependant, l'optimalité des meilleurs algorithmes connus, de complexité $O(n\log^2 n)$, reste un problème ouvert. Même dans le cas particulier où les points d'évaluation sont en progression arithmétique, on ne dispose d'aucun algorithme d'évaluation multipoint de complexité $O(\mathsf{M}(n))$, ni d'une preuve qu'un tel algorithme n'existe pas. De même, pour des points quelconques $a_0,\dots,a_{n-1}$ on ne connaît pas d'algorithme de complexité $O(\mathsf{M}(n))$ pour calculer le polynôme $\prod_i(X-a_i)$. Notons toutefois que de tels algorithmes existent dans les cas particuliers où les a_i forment une progression arithmétique ou géométrique [6].

L'exercice 8 provient de [10] ; historiquement, ce fut le premier algorithme rapide pour les restes chinois. Les exercices 6 et 7 sont tirés de [1].

Bibliographie

[1] Aho, A. V., K. Steiglitz et J. D. Ullman (1975). « Evaluating polynomials at fixed sets of points ». In : *SIAM Journal on Computing*, vol. 4, n°4, p. 533–539.

[2] Bluestein, L. I. (1970). « A linear filtering approach to the computation of the discrete Fourier transform ». In : *IEEE Trans. Electroacoustics*, vol. AU-18, p. 451–455.

[3] Borodin, A. et R. T. Moenck (1974). « Fast modular transforms ». In : *Comput. Sys. Sci.* vol. 8, n°3, p. 366–386.

[4] Borodin, A. et I. Munro (1971). « Evaluating polynomials at many points ». In : *Information Processing Letters*, vol. 1, n°2, p. 66–68. doi : 10.1016/0020-0190(71)90009-3.

[5] Bostan, Alin, Grégoire Lecerf et Éric Schost (2003). « Tellegen's principle into practice ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par J. R. Sendra. ACM Press, p. 37–44.

[6] Bostan, Alin et Éric Schost (2005). « Polynomial evaluation and interpolation on special sets of points ». In : *Journal of Complexity*, vol. 21, n°4, p. 420–446.

[7] Fiduccia, C. M. (1972). « Polynomial evaluation via the division algorithm : The fast Fourier transform revisited. » In : *STOC'72 : ACM Symposium on Theory of Computing*, p. 88–93.

[8] Garner, Harvey L. (1959). « The residue number system ». In : *IRE-AIEE-ACM'59 Western joint computer conference*. San Francisco, California : ACM, p. 146–153. doi : 10.1145/1457838.1457864.

[9] Gauss, Carl Friedrich (1966). *Disquisitiones arithmeticae*. Translated into English by Arthur A. Clarke, S. J. Yale University Press, xx+472 pages.

[10] Heindel, L. E. et E. Horowitz (1971). « On decreasing the computing time for modular arithmetic ». In : *12th symposium on switching and automata theory*. IEEE, p. 126–128.

[11] Horowitz, E. (1972). « A fast Method for Interpolation Using Preconditioning ». In : *Information Processing Letters*, vol. 1, n°4, p. 157–163.

[12] Mersereau, Russell M. (1974). « An algorithm for performing an inverse chirp z-transform ». In : *IEEE Transactions on Audio and Electroacoustics*, vol. ASSP-22, n°5, p. 387–388.

[13] Moenck, R. T. et A. Borodin (1972). « Fast modular transforms via division ». In : *Thirteenth Annual IEEE Symposium on Switching and Automata Theory*, p. 90–96.

[14] Montgomery, P. L. (1992). « An FFT extension of the elliptic curve method of factorization ». Thèse de doct. University of California, Los Angeles CA.

[15] Ostrowski, A. (1954). « On two problems in abstract algebra connected with Horner's rule ». In : *Studies in mathematics and mechanics presented to Richard von Mises*. NY : Academic Press Inc., p. 40–48.

[16] Pan, V. Ya. (1966). « Methods of computing values of polynomials ». In : *Russian Mathematical Surveys*, vol. 21, n°1, p. 105–136.

[17] Rabiner, L. R., R. W. Schafer et C. M. Rader (1969). « The chirp z-transform algorithm and its application ». In : *Bell System Technical Journal*, vol. 48, p. 1249–1292.

[18] Shoup, Victor et Roman Smolensky (1996/97). « Lower bounds for polynomial evaluation and interpolation problems ». In : *Computational Complexity*, vol. 6, n°4, p. 301–311.

[19] Strassen, V. (1969). « Gaussian Elimination is Not Optimal ». In : *Numerische Mathematik*, vol. 13, p. 354–356.

[20] — (1972/73). « Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten ». In : *Numerische Mathematik*, vol. 20, p. 238–251.

[21] Svoboda, A. et M. Valach (1955). « Operátorové obvody (Operational circuits) ». In : *Stroje na Zpracování Informací (Information Processing Machines)*, vol. 3, p. 247–295.

[22] Takahasi, H. et Y. Ishibashi (1961). « A new method for exact calculation by a digital computer ». In : *Information Processing in Japan*, p. 28–42.

[23] Waring, E. (1779). « Problems concerning interpolations ». In : *Philosophical Transactions of the Royal Society of London*, vol. 59, p. 59–67.

CHAPITRE 6

Pgcd et résultant

Résumé

L'algorithme d'Euclide classique permet de calculer le pgcd et le pgcd étendu. Il est relié aux résultants, qui permettent un calcul d'élimination. Le pgcd et le résultant peuvent être calculés en complexité quasi-optimale.

Les calculs de pgcd sont cruciaux pour la simplification des fractions, qu'il s'agisse de fractions d'entiers ou de fractions de polynômes. Les algorithmes efficaces de factorisation de polynômes reposent par ailleurs de manière essentielle sur le pgcd de polynômes. Comme pour la multiplication, les algorithmes efficaces de pgcd sont plus complexes dans le cas des entiers que dans le cas des polynômes, à cause des retenues, et nous n'en détaillons que la version polynomiale. Dans les définitions et résultats de nature moins algorithmique, nous utilisons le cadre algébrique des anneaux euclidiens qui permet de traiter simultanément toutes les applications que nous avons en vue.

L'algorithme d'Euclide pour le calcul du pgcd est présenté en Section 1. Dans le cas des polynômes de $\mathbb{K}[X]$, sa complexité est quadratique en nombre d'opérations dans le corps $\mathbb{K}$. Le résultant est également lié à l'algorithme d'Euclide, ses propriétés et son calcul sont présentés en Section 2. En outre, une technique dite « des sous-résultants » permet de réduire l'explosion de la taille des coefficients intermédiaires dont souffre l'algorithme d'Euclide pour le pgcd dans $\mathbb{Q}[X]$. Le chapitre se termine en Section 3 sur un algorithme de complexité quasi-optimale pour le pgcd dans $\mathbb{K}[X]$, exploitant la multiplication polynomiale rapide.

Dans ce chapitre, $\mathbb{A}$ désignera toujours un anneau intègre.

1. Algorithme d'Euclide

1.1. Le pgcd. Des pgcd peuvent être définis dans tout anneau intègre $\mathbb{A}$: on appelle *plus grand diviseur commun* (pgcd) de A et B tout $G \in \mathbb{A}$ qui divise A et B et tel que tout diviseur de A et de B divise aussi G. Contrairement à l'usage, nous notons dans ce chapitre $\mathrm{pgcd}(A, B)$ tout pgcd de A et de B sans faire de choix de normalisation parmi les pgcd de A et de B.

L'algorithme d'Euclide présenté dans cette section permet le calcul du pgcd dans $\mathbb{Z}$ ou dans l'anneau $\mathbb{K}[X]$, où $\mathbb{K}$ est un corps. Il repose sur l'existence d'une *division euclidienne* dans ces anneaux. Un anneau intègre $\mathbb{A}$ est appelé *anneau euclidien* s'il existe une fonction de taille $d : \mathbb{A} \to \mathbb{N} \cup \{-\infty\}$ telle que pour tout $a \in \mathbb{A}$, $b \in \mathbb{A} \setminus \{0\}$, il existe q et r dans $\mathbb{A}$ avec

$$a = qb + r, \qquad d(r) < d(b).$$

En particulier, $d(0) < d(b)$ dès que b est non nul. Dans le cas des entiers, la valeur absolue fournit une telle fonction d, et le degré remplit ce rôle pour les polynômes. La notation $r := a \bmod b$ sert dans les deux cas à définir r à partir de a et de b. L'entier ou le polynôme r est appelé *reste* de la division euclidienne de a par b.

Euclide(A, B)

Entrée : A et B dans $\mathbb{A}$.

Sortie : Un pgcd de A et B.

1. $R_0 := A$; $R_1 := B$; $i := 1$.
2. Tant que R_i est non nul, faire :
   $R_{i+1} := R_{i-1} \bmod R_i$
   $i := i + 1$
3. Renvoyer R_{i-1}.

FIGURE 1. L'algorithme d'Euclide.

EXERCICE 1. Un élément d'un anneau est dit irréductible si les produits qui lui sont égaux font tous intervenir un élément inversible. Tout anneau euclidien est *factoriel* (c'est-à-dire que tout élément non nul y a une factorisation unique en irréductibles, à l'ordre près de la factorisation et à des multiplications près des facteurs par des inversibles de l'anneau). Le pgcd se lit aussi sur les factorisations. Si A et B se factorisent sous la forme

$$A = am_1^{k_1} \cdots m_s^{k_s}, \qquad B = bm_1^{\ell_1} \cdots m_s^{\ell_s},$$

où a et b sont inversibles dans $\mathbb{A}$, les m_i sont irréductibles et les k_i et ℓ_i sont des entiers positifs ou nuls, montrer qu'alors un pgcd de A et B est

$$G = m_1^{\min(k_1,\ell_1)} \cdots m_s^{\min(k_s,\ell_s)}.$$

Autrement dit, on rassemble dans G les facteurs irréductibles communs de A et B, avec la plus grande multiplicité possible. La factorisation dans $\mathbb{Z}$ ou dans $\mathbb{K}[X]$ est plus coûteuse que le pgcd et cette propriété n'est donc pas utilisée pour le calcul du pgcd.

1.2. Calcul du pgcd. Étant donnés A, B dans un anneau euclidien $\mathbb{A}$, l'algorithme d'Euclide en Figure 1 calcule une suite de restes successifs dont la taille décroît, jusqu'à atteindre le pgcd.

La terminaison provient de la décroissance stricte de la taille à chaque étape. La correction de cet algorithme se déduit de la relation

$$\operatorname{pgcd}(F, G) = \operatorname{pgcd}(H, G) \qquad \text{pour} \qquad H := F \bmod G,$$

dont la preuve est laissée en exercice. Par récurrence, il s'ensuit que $\operatorname{pgcd}(F, G) = \operatorname{pgcd}(R_i, R_{i+1})$ pour tout i. Si en outre R_{i+1} est nul, alors $\operatorname{pgcd}(R_i, R_{i+1}) = R_i$, ce qui prouve la correction.

EXEMPLE 1. Soient $A = X^4 - 13X^3 + 2X^2 - X - 1$ et $B = X^2 - X - 1$ dans $\mathbb{Q}[X]$. La suite des restes est :

$$\begin{aligned} R_0 &= X^4 - 13X^3 + 2X^2 - X - 1, \\ R_1 &= X^2 - X - 1, \\ R_2 &= -22X - 10, \\ R_3 &= -41/121, \\ R_4 &= 0, \end{aligned}$$

de sorte que $-41/121$ est un pgcd de A et B et donc 1 aussi.

THÉORÈME 1. *L'algorithme d'Euclide calcule un pgcd de A et B dans $\mathbb{K}[X]$ en $O(\deg A \deg B)$ opérations dans $\mathbb{K}$.*

DÉMONSTRATION. La correction a été prouvée dans le cas général. Pour l'étude de complexité, nous supposons d'abord que $\deg A \geq \deg B$. D'après le Chapitre 3 (p. 62), le calcul naïf de $P \bmod Q$ peut être effectué en $2\deg Q(\deg P - \deg Q + 1)$ opérations de $\mathbb{K}$. Il s'ensuit que le coût de l'algorithme d'Euclide est borné par la somme des $2\deg(R_i)(\deg R_{i-1} - \deg R_i + 1)$, pour $i \geq 1$. Tous les $\deg(R_i)$ sont majorés par $\deg A$, de sorte que le coût est borné par $2\deg A \sum_{i\geq 1}(\deg R_{i-1} - \deg R_i + 1) = 2\deg A(\deg R_0 + \deg B) = O(\deg A \deg B)$.

Si le degré de B est supérieur à celui de A, la première étape ne coûte pas d'opération arithmétique, et la borne ci-dessus s'applique au reste du calcul. □

EXERCICE 2. Énoncer et prouver l'analogue entier du résultat du Théorème 1.

La borne de complexité quadratique reflète bien le comportement de l'algorithme : dans une exécution typique de l'algorithme, les quotients ont degré 1 à chaque itération, les degrés des restes successifs diminuent de 1 à chaque itération et leur calcul est linéaire ; le nombre de coefficients intermédiaires calculés est quadratique et ils sont calculés aussi efficacement que possible par un algorithme qui les calcule tous.

1.3. Pgcd étendu et inversion modulaire.

Relation de Bézout. Une relation de la forme $G = UA + VB$ qui donne le pgcd G de deux éléments A et B de $\mathbb{A}$ avec deux cofacteurs U et V dans $\mathbb{A}$ est appelée une *relation de Bézout*. L'algorithme d'Euclide étendu est une modification légère de l'algorithme d'Euclide qui calcule non seulement le pgcd, mais aussi une relation de Bézout particulière,

$$UA + VB = G, \quad \text{avec } d(UG) < d(B) \text{ et } d(VG) < d(A). \tag{1}$$

Une fois le pgcd G choisi, les cofacteurs U et V sont rendus uniques par la contrainte sur les tailles. Dans de nombreuses applications, c'est davantage de ces *cofacteurs* U et V que du pgcd lui-même qu'on a besoin. On parle alors de calcul de pgcd *étendu*. Ce calcul intervient par exemple de manière importante dans la manipulation des nombres algébriques (Exemple 3 ci-dessous), dans le calcul dans des algèbres quotient $\mathbb{K}[X]/(P)$ (Exemples 2 et 5 ci-dessous, cf. aussi la Section 2.4 en page 63), et dans le développement rapide des séries algébriques (Chapitre 14). Il intervient également dans d'autres algorithmes qui ne seront pas abordés dans ce cours, par exemple pour le calcul des « restes chinois rapides », pour la factorisation de polynômes dans $\mathbb{Z}[X]$ ou $\mathbb{Q}[X]$ par des techniques de type « Hensel » (Chapitre 32), ou pour l'intégration symbolique (à la Hermite).

Mais, avant tout, les calculs de pgcd étendu permettent d'effectuer des *inversions modulaires*. Lorsque A et B sont premiers entre eux, (c'est-à-dire si G est un élément inversible de A), alors l'élément V est un inverse de B modulo A.

EXEMPLE 2. La relation de Bézout pour $a + bX$ et $1 + X^2$ dans $\mathbb{R}[X]$ s'écrit :

$$(a - bX)(a + bX) + b^2(1 + X^2) = a^2 + b^2.$$

L'inverse de $B = a + bX$ modulo $A = 1 + X^2$ vaut donc

$$V = \frac{a - bX}{a^2 + b^2}.$$

Puisque le corps des complexes s'obtient par le quotient $\mathbb{R}[X]/(X^2 + 1)$, cette relation n'est autre que la formule d'inversion familière

$$\frac{1}{a + ib} = \frac{a - ib}{a^2 + b^2}$$

où $i = \sqrt{-1}$.

Si $A \in \mathbb{A}$ est irréductible, alors $\mathbb{A}/(A)$ est un corps, et le calcul de la relation de Bézout permet d'y effectuer la division : si $B \neq 0 \mod A$, alors $N/B = NV \mod A$.

Exemple 3. La « simplification » de la fraction rationnelle

$$r = \frac{\phi^4 - \phi + 1}{\phi^7 - 1} = \frac{25\phi - 34}{169}$$

où ϕ est le « nombre d'or », défini par $A(\phi) = 0$ pour $A(X) = X^2 - X - 1$, est obtenue par les trois calculs suivants :

— calcul du reste $U = 13X + 7$ par $U := X^7 - 1 \mod A$,

— détermination de la relation de Bézout

$$\frac{13X - 20}{169} U - A = 1,$$

— calcul du reste $V = 25X - 34$ par $V := (13X - 20)(X^4 - X + 1) \mod A$.

Plus généralement, dans le cas où $A \in \mathbb{K}[X]$ est un polynôme irréductible, ces calculs montrent que le corps $\mathbb{K}(\alpha)$ des fractions rationnelles en α racine de A est un espace vectoriel dont une base est $1, \alpha, \alpha^2, \ldots, \alpha^{\deg A - 1}$. Les algorithmes d'Euclide et d'Euclide étendu fournissent un moyen de calcul dans cette représentation. Il est ainsi possible de manipuler de manière exacte les racines d'un polynôme de degré arbitraire sans « résoudre ».

Exemple 4. Le même calcul qu'à l'exemple 3 fonctionne sur des entiers :

$$r = \frac{25}{33} \equiv 5 \mod 7$$

se déduit de

$$33 \bmod 7 = 5, \qquad 3 \times 5 - 2 \times 7 = 1, \qquad 3 \times 25 \bmod 7 = 5.$$

Exemple 5. Lorsque l'élément $A \in \mathbb{A}$ n'est pas irréductible, $\mathbb{A}/(A)$ n'est pas un corps. Cependant, les éléments inversibles peuvent y être inversés par le même calcul que ci-dessus. Lorsqu'un élément B non nul n'est pas inversible, la relation de Bézout se produit avec un G différent de 1. Il est alors possible de tirer parti de cette information (un facteur de A) en scindant le calcul d'une part sur G et d'autre part sur B/G.

Exercice 3. Soit p un nombre premier et soit a un entier non multiple de p. Donner un algorithme pour le calcul de l'inverse de a modulo p utilisant le petit théorème de Fermat ($a^{p-1} = 1 \mod p$) et l'exponentiation binaire. Analyser la complexité binaire de cet algorithme et comparer avec l'approche par Euclide étendu.

Calcul des cofacteurs. L'idée-clé est de suivre pendant l'algorithme d'Euclide la décomposition de chacun des R_i sur A et B. Autrement dit, pour tout i, l'algorithme calcule des éléments U_i et V_i tels que

$$U_i A + V_i B = R_i,$$

dont les tailles sont bien contrôlées. Pour $i = 0$, il suffit de poser $U_0 = 1, V_0 = 0$, ce qui correspond à l'égalité

$$1 \cdot A + 0 \cdot B = A = R_0.$$

Pour $i = 1$, l'égalité

$$0 \cdot A + 1 \cdot B = B = R_1$$

AEE(A, B)

Entrée : A et B dans $\mathbb{A}$.

Sortie : Un pgcd G de A et B, et les cofacteurs U et V de l'identité de Bézout $UA + VB = G$.

1. $R_0 := A$; $U_0 := 1$; $V_0 := 0$; $R_1 := B$; $U_1 := 0$; $V_1 := 1$; $i := 1$.
2. Tant que R_i est non nul, faire :
   $(Q_i, R_{i+1}) :=$ QuotRem(R_{i-1}, R_i) ; # division euclidienne
   $U_{i+1} := U_{i-1} - Q_i U_i$; $V_{i+1} := V_{i-1} - Q_i V_i$;
   $i := i + 1$
3. Renvoyer $R_{i-1}, U_{i-1}, V_{i-1}$.

FIGURE 2. L'algorithme d'Euclide étendu.

est obtenue avec $U_1 = 0, V_1 = 1$. Ensuite, la division euclidienne

$$R_{i-1} = Q_i R_i + R_{i+1}$$

donne la relation

$$R_{i+1} = R_{i-1} - Q_i R_i = (U_{i-1} - Q_i U_i)A + (V_{i-1} - Q_i V_i)B,$$

qui pousse à définir U_{i+1} par $U_{i-1} - Q_i U_i$ et V_{i+1} par $V_{i-1} - Q_i V_i$. À partir de cette relation, une preuve par récurrence montre que les conditions de tailles de la relation de Bézout sont satisfaites.

L'algorithme est résumé en Figure 2. À nouveau, dans le cas des polynômes ou des entiers, la complexité est quadratique.

2. Résultant

2.1. Matrice de Sylvester. L'algorithme d'Euclide pour *deux* polynômes A, B à *une* variable est étroitement relié à la matrice de Sylvester. Si

$$A = a_m X^m + \cdots + a_0, \quad (a_m \neq 0), \qquad \text{et} \qquad B = b_n X^n + \cdots + b_0, \quad (b_n \neq 0),$$

appartiennent à $\mathbb{A}[X]$, cette matrice, carrée de taille $m + n$, s'écrit

$$\mathsf{Syl}(A, B) = \begin{pmatrix} a_m & a_{m-1} & \dots & a_0 & & & \\ & a_m & a_{m-1} & \dots & a_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & a_m & a_{m-1} & \dots & a_0 \\ b_n & b_{n-1} & \dots & b_0 & & & \\ & b_n & b_{n-1} & \dots & b_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & b_n & b_{n-1} & \dots & b_0 \end{pmatrix}$$

avec les n premières lignes contenant les coefficients de A et les m suivantes contenant les coefficients de B.

La transposée de cette matrice représente l'application linéaire

$$(U, V) \mapsto AU + BV,$$

en se limitant aux paires (U, V) tels que $\deg U < \deg B$ et $\deg V < \deg A$, et en utilisant la base des monômes $1, X, \dots, X^{m+n-1}$ à l'arrivée. Les combinaisons linéaires des lignes de la matrice de Sylvester donnent donc les coefficients des polynômes qui peuvent être obtenus comme image.

Lorsque $\mathbb{A}$ est un corps, au vu de la relation de Bézout (1), le pgcd de A et B peut être obtenu ainsi et il est caractérisé comme l'élément de plus petit degré qui peut l'être. Ceci permet de le lire sur une forme *échelonnée en ligne* de la matrice $\mathsf{Syl}(A,B)$ (cf. Définition 3, page 132).

PROPOSITION 1. *Soient A, B deux polynômes de $\mathbb{K}[X]$ où $\mathbb{K}$ est un corps. La forme échelonnée en ligne de la matrice de Sylvester $\mathsf{Syl}(A,B)$ contient sur sa dernière ligne non nulle les coefficients d'un pgcd de A et B. En outre, la dimension du noyau de cette matrice est le degré du pgcd.*

EXEMPLE 6. Pour les polynômes $A = X^4 - X^3 - 7X^2 + 2X + 3$ et $B = X^3 - 4X^2 + 2X + 3$, la matrice de Sylvester et sa forme échelonnée en ligne sont :

$$\begin{pmatrix} 1 & -1 & -7 & 2 & 3 & 0 & 0 \\ 0 & 1 & -1 & -7 & 2 & 3 & 0 \\ 0 & 0 & 1 & -1 & -7 & 2 & 3 \\ 1 & -4 & 2 & 3 & 0 & 0 & 0 \\ 0 & 1 & -4 & 2 & 3 & 0 & 0 \\ 0 & 0 & 1 & -4 & 2 & 3 & 0 \\ 0 & 0 & 0 & 1 & -4 & 2 & 3 \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} 1 & -1 & -7 & 2 & 3 & 0 & 0 \\ 0 & 1 & -1 & -7 & 2 & 3 & 0 \\ 0 & 0 & 1 & -1 & -7 & 2 & 3 \\ 0 & 0 & 0 & -14 & 45 & -3 & -18 \\ 0 & 0 & 0 & 0 & -\frac{5}{7} & \frac{12}{7} & \frac{9}{7} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{10} & -\frac{3}{10} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

La dernière ligne non nulle donne le pgcd $X - 3$.

DEFINITION 1. Le *résultant* de A et B est le déterminant de la matrice de Sylvester $\mathsf{Syl}(A,B)$. Il est noté $\mathrm{Res}(A,B)$, ou $\mathrm{Res}_X(A,B)$ si l'on veut insister sur l'élimination de la variable X.

L'utilisation des résultants pour l'élimination demande de travailler dans ce chapitre avec des matrices à coefficients dans un anneau. Les définitions et propriétés du déterminant persistent dans ce cas (c'est une forme multilinéaire alternée, le déterminant de la matrice identité vaut 1, on peut le développer par rapport à une ligne ou une colonne, il ne change pas par addition de combinaisons linéaires de lignes ou de colonnes, et s'exprime comme somme sur les permutations).

Le résultant peut être vu comme une condition de cohérence pour le système formé par les deux polynômes A et B.

COROLLAIRE 1. *Soient A et B deux polynômes de $\mathbb{K}[X]$. Alors A et B sont premiers entre eux si et seulement si* $\mathrm{Res}(A,B) \neq 0$.

DÉMONSTRATION. Le déterminant d'une matrice carrée se lit sur la forme échelonnée en ligne en faisant le produit des éléments diagonaux. Il est non nul si et seulement si tous les éléments diagonaux le sont et dans ce cas un pgcd non nul est sur la dernière ligne. À l'inverse, s'il existe un pgcd G non nul, alors les polynômes $X^k G$ montrent l'existence de lignes avec un coefficient de tête non nul dans chaque colonne de la forme échelonnée. □

EXEMPLE 7. Le discriminant de A à coefficients dans un corps $\mathbb{K}$ est le polynôme $\mathrm{disc}(A)$ défini par

$$\mathrm{Res}(A, A') = (-1)^{m(m-1)/2} a_m \,\mathrm{disc}(A).$$

Il s'annule lorsque ces deux polynômes ont une racine commune dans une clôture algébrique de $\mathbb{K}$, c'est-à-dire, en caractéristique nulle, lorsque A a une racine multiple.

EXERCICE 4. Calculer le discriminant de aX^2+bX+c en prenant le déterminant de la matrice de Sylvester.

2.2. Applications du résultant.

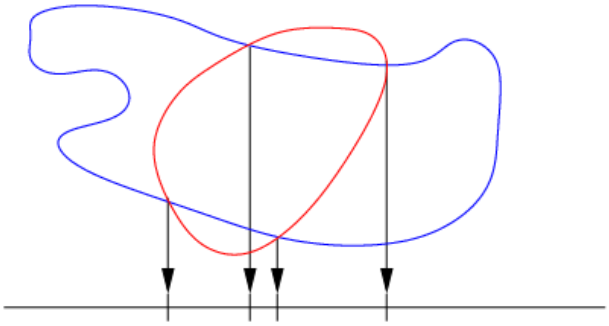


FIGURE 3. Le résultant calcule des projections.

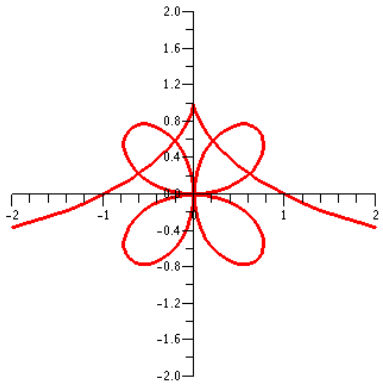


FIGURE 4. Les deux courbes de l'exemple 8.

Calculs de projections. Algébriquement, la « résolution » d'un système polynomial se ramène souvent à une question d'*élimination*. Lorsqu'elle est possible, l'élimination successive des variables amène le système d'entrée sous une forme triangulaire. De proche en proche, la résolution d'un tel système se réduit alors à la manipulation de polynômes à une variable, pour lesquels compter, isoler, ... , les solutions est bien plus facile.

Géométriquement, l'élimination correspond à une projection. Cette section montre comment le résultant de deux polynômes permet de traiter ce genre de problèmes, pour les systèmes de deux équations en deux inconnues. Le schéma général est représenté en Figure 3. Cette opération est à la base d'une technique plus générale pour un nombre arbitraire d'équations et d'inconnues, la *résolution géométrique*, qui sera présentée au Chapitre 26.

EXEMPLE 8. Les deux courbes de la Figure 4 ont pour équations

$$A = (X^2+Y^2)^3 - 4X^2Y^2 = 0, \qquad B = X^2(1+Y) - (1-Y)^3 = 0.$$

Ces deux polynômes sont irréductibles et leur pgcd, qui vaut 1, ne renseigne pas sur leurs racines communes. Les résultants par rapport à X et Y donnent en revanche des polynômes qui s'annulent sur les coordonnées de ces points d'intersection :

$$\mathrm{Res}_X(A,B) = (4Y^7 + 60Y^6 - 152Y^5 + 164Y^4 - 95Y^3 + 35Y^2 - 9Y + 1)^2,$$
$$\mathrm{Res}_Y(A,B) = 16X^{14} + 6032X^{12} - 1624X^{10} + 4192X^8 - 815X^6 - 301X^4 - 9X^2 + 1.$$

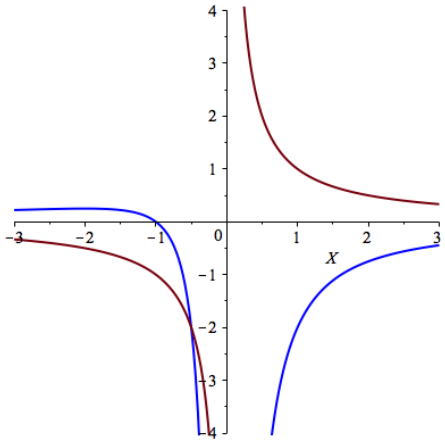


FIGURE 5. Les courbes de l'exemple 9.

Il n'y a que 4 points d'intersection visibles sur la figure 4, les 10 autres ont au moins une coordonnée qui n'est pas réelle. Le caractère bicarré du résultant en Y provient de la symétrie de la figure par rapport à l'axe des Y. C'est pour cette même raison que le premier résultant est un carré.

Le résultat général est le suivant.

PROPOSITION 2. *Soient $A = a_m Y^m + \cdots$ et $B = b_n Y^n + \cdots$ où les coefficients a_i et b_j sont dans $\mathbb{K}[X]$ où $\mathbb{K}$ est un corps algébriquement clos. Alors les racines du polynôme $\mathrm{Res}_Y(A, B) \in \mathbb{K}[X]$ sont d'une part les abscisses des solutions du système $A = B = 0$, d'autre part les racines des termes de tête a_m et b_n.*

DÉMONSTRATION. La preuve repose sur des résultats de la section suivante, qui sont indépendants de cette proposition.

Tout d'abord, les racines des termes de tête a_m et b_n sont racines du résultant d'après la formule de Poisson (Théorème 2 ci-dessous).

Ensuite, d'après la proposition 3 ci-dessous, il existe U et V dans $\mathbb{K}[X, Y]$ tels que

$$\mathrm{Res}_Y(A, B) = UA + VB.$$

En évaluant cette identité en (x, y) tels que $A(x, y) = B(x, y) = 0$, on voit qu'un tel x est racine de $\mathrm{Res}_Y(A, B)$. À l'inverse, si x annule le résultant, alors d'après le corollaire 1, les polynômes $A(x, Y)$ et $B(x, Y)$ ont un pgcd de degré au moins 1. Comme le corps est algébriquement clos, il existe alors $y \in \mathbb{K}$ racine de ce pgcd tel que $A(x, y) = B(x, y) = 0$. □

Graphiquement, les racines « dégénérées » du second cas se traduisent par la présence d'asymptotes verticales.

EXEMPLE 9. Avec $A = X^2Y + X + 1$, $B = XY - 1$, on obtient $\mathrm{Res}_Y(A, B) = -X(2X + 1)$ (asymptote en $X = 0$, « vraie » solution en $X = -\frac{1}{2}$). Les courbes correspondantes sont représentées en Figure 5.

Une fois calculé le résultant donnant les coordonnées des abscisses des intersections des deux courbes, il est souhaitable d'obtenir les ordonnées correspondantes. Dans le cas simple où l'avant-dernier reste de l'algorithme d'Euclide est de degré 1, il fournit bien une telle paramétrisation.

Exemple 10. Sur les deux polynômes A et B de l'exemple 8, vus comme polynômes dans $\mathbb{Q}(X)[Y]$, la suite des restes est

$$(X^4 - 13X^2 + 15)Y^2 - 4(X^2+2)(X^2+3)Y + (X^6 - 2X^4 - 8X^2 + 10),$$
$$(4X^8 - 344X^6 - 1243X^4 - 301X^2 - 21)Y + (84X^8 - 372X^6 + 169X^4 + 143X^2 + 15),$$
$$1.$$

Un calcul de pgcd du coefficient de Y dans l'avant-dernier reste avec $\mathrm{Res}_Y(A,B)$ montre que ces deux polynômes sont premiers entre eux. Pour chaque racine X de $\mathrm{Res}_Y(A,B)$ il y a donc un unique point d'intersection aux deux courbes, d'ordonnée donnée par

$$Y = -\frac{84X^8 - 372X^6 + 169X^4 + 143X^2 + 15}{4X^8 - 344X^6 - 1243X^4 - 301X^2 - 21}.$$

En utilisant un calcul de pgcd étendu, ceci peut être récrit, comme expliqué plus haut, en un polynôme de degré au plus 13 en X.

Le même calcul sur A et B vus comme polynômes dans $\mathbb{Q}(Y)[X]$ donne B comme dernier reste avant le pgcd. Ceci correspond aux deux points ayant la même projection sur l'axe des Y.

Implicitation. Une autre application géométrique du résultant est l'*implicitation* de courbes du plan. Étant donnée une courbe paramétrée

$$X = A(T), \quad Y = B(T), \qquad A, B \in \mathbb{K}(T),$$

il s'agit de calculer un polynôme non trivial en X et Y qui s'annule sur la courbe. Il suffit pour cela de prendre le résultant en T des numérateurs de $X - A(T)$ et de $Y - B(T)$.

Exemple 11. La courbe « en fleur » de la Figure 4 peut aussi être donnée sous la forme

$$X = \frac{4T(1-T^2)^2}{(1+T^2)^3}, \quad Y = \frac{8T^2(1-T^2)}{(1+T^2)^3}.$$

Il suffit d'effectuer le calcul du résultant

$$\mathrm{Res}_T((1+T^2)^3X - 4T(1-T^2)^2, (1+T^2)^3Y - 8T^2(1-T^2))$$

pour retrouver (à un facteur constant près) le polynôme A de l'exemple 8.

2.3. Propriétés et calcul quadratique. L'essentiel des propriétés du résultant est contenu dans le théorème suivant.

Théorème 2. *[Formule de Poisson] Si les polynômes A et B de $\mathbb{A}[X]$ s'écrivent*

$$A = a(X-\alpha_1)\cdots(X-\alpha_m), \quad B = b(X-\beta_1)\cdots(X-\beta_n),$$

alors, le résultant de A et B vaut

$$\begin{aligned}\mathrm{Res}(A,B) &= a^n b^m \prod_{i,j}(\alpha_i - \beta_j) = (-1)^{mn} b^m \prod_{1\le j\le n} A(\beta_j) \\ &= a^n \prod_{1\le i\le m} B(\alpha_i) = (-1)^{mn}\,\mathrm{Res}(B,A).\end{aligned}$$

Démonstration. Il suffit de prouver la première égalité. Les trois suivantes en sont des conséquences immédiates.

Le facteur $a^n b^m$ est une conséquence de la multilinéarité du déterminant. On peut donc se restreindre au cas où $a = b = 1$. Il est commode de considérer le cas générique où les α_i et β_j sont des indéterminées et où l'anneau $\mathbb{A}$ est $\mathbb{Z}[\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_n]$. Si le résultat, qui est une identité entre polynômes, est

vrai dans cet anneau, il l'est aussi par spécialisation pour des valeurs arbitraires des α_i et β_j. Le corollaire 1 avec $\mathbb{A} = \mathbb{Q}(\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_n)$ montre que le produit des $\alpha_i - \beta_j$ divise le résultant. Par ailleurs, le degré en α_i de chacune des n premières lignes de la matrice de Sylvester est 1, et ce degré est nul pour les m autres lignes. Ceci donne une borne n pour le degré en chaque α_i du résultant et de la même manière une borne m pour son degré en chaque β_j. Il s'ensuit que le résultant est égal au produit des $\alpha_i - \beta_j$ à un facteur constant près. Ce facteur est indépendant des α_i et β_j. En choisissant $\alpha_1 = \cdots = \alpha_m = 0$, c'est-à-dire $A = X^m$, la matrice de Sylvester est triangulaire, et son déterminant vaut $B(0)^m$, ce qui donne le facteur 1 et conclut la preuve. □

EXEMPLE 12. Si $A = a\prod_{i=1}^{m}(X - x_i)$, la formule de Poisson donne

$$\operatorname{disc}(A) = a^{m-2}\prod_{i=1}^{m} A'(x_i) = a^{m-2}\prod_{i=1}^{m} a\prod_{j\neq i}(x_j - x_i) = a^{2m-2}\prod_{i<j}(x_i - x_j)^2.$$

COROLLAIRE 2 (Multiplicativité du résultant). *Pour tous polynômes A, B, C de $\mathbb{A}[X]$,* $\operatorname{Res}(AB, C) = \operatorname{Res}(A, C)\operatorname{Res}(B, C)$.

DÉMONSTRATION. L'anneau $\mathbb{A}$ étant intègre, il possède un corps de fractions qui a lui-même une clôture algébrique dans laquelle les polynômes A, B et C peuvent s'écrire sous la forme utilisée dans le théorème précédent. L'identité sur les résultants (qui appartiennent à $\mathbb{A}$ par définition) est alors vérifiée en considérant les produits deux à deux de racines. □

Une dernière propriété utile du résultant est la suivante.

PROPOSITION 3. *Il existe U et V dans $\mathbb{A}[X]$ tels que le résultant de A et de B s'écrive* $\operatorname{Res}(A, B) = UA + VB$.

DÉMONSTRATION. En ajoutant à la dernière colonne de la matrice de Sylvester le produit des colonnes précédentes par des puissances adaptées de X, on fait apparaître dans la dernière colonne les polynômes A et B, sans avoir changé le déterminant. Le développement du déterminant par rapport à la dernière colonne permet alors de conclure. □

L'algorithme d'Euclide étendu montre que le résultant $\operatorname{Res}(A, B)$ (qui est un multiple par un élément du corps des fractions $\mathbb{K}$ de l'un des restes euclidiens « standards »), peut s'écrire comme une combinaison polynomiale de A et B, à coefficients dans $\mathbb{K}[X]$. La proposition 3 montre que cette combinaison se fait « sans division », c'est-à-dire avec des coefficients dans $\mathbb{A}[X]$.

En utilisant la définition 1 et les résultats du Chapitre 8, le résultant de deux polynômes A, B de $\mathbb{K}[X]$ peut se calculer naïvement en $O(\mathsf{MM}(n)) = O(n^\theta)$ opérations dans $\mathbb{K}$, où $n = \max(\deg(A), \deg(B))$ et où $2 \leq \theta \leq 3$ est un exposant faisable pour la multiplication matricielle. Un algorithme plus efficace, de complexité seulement quadratique, s'obtient comme conséquence du Théorème 2 et est laissé en exercice.

EXERCICE 5 (Algorithme de type Euclide pour le résultant). Soient A et B deux polynômes de $\mathbb{K}[X]$ de degrés $m = \deg(A)$ et $n = \deg(B)$.

1. Soit $A = QB + R$ la division euclidienne dans $\mathbb{K}[X]$ de A par B. Si $r = \deg(R)$, et si b_n est le coefficient dominant de B, montrer que
$$\operatorname{Res}(A, B) = (-1)^{mn} \cdot b_n^{m-r} \cdot \operatorname{Res}(B, R).$$
2. Écrire un algorithme de type Euclide (*i.e.* à base de divisions répétées), qui calcule le résultant $\operatorname{Res}(A, B)$ en $O(mn)$ opérations dans $\mathbb{K}$.

2.4. Calcul avec des nombres algébriques. L'idée que les polynômes sont de bonnes structures de données pour représenter leur racines amène à chercher des algorithmes pour effectuer les opérations de base sur ces racines, comme la somme ou le produit. Le résultant répond à cette attente.

PROPOSITION 4. *Soient $A = \prod_i (X - \alpha_i)$, $B = \prod_j (X - \beta_j)$ et C des polynômes unitaires de $\mathbb{K}[X]$, avec C et A premiers entre eux. Alors*

$$\begin{aligned}
\mathrm{Res}_X(A(X), B(T - X)) &= \prod_{i,j} (T - (\alpha_i + \beta_j)),\\
\mathrm{Res}_X(A(X), B(T + X)) &= \prod_{i,j} (T - (\beta_j - \alpha_i)),\\
\mathrm{Res}_X(A(X), X^{\deg B} B(T/X)) &= \prod_{i,j} (T - \alpha_i \beta_j),\\
\mathrm{Res}_X(A(X), C(X)T - B(X)) &= (-1)^{\deg A} C(0) \prod_i \left(T - \frac{B(\alpha_i)}{C(\alpha_i)}\right).
\end{aligned}$$

DÉMONSTRATION. C'est une application directe du Théorème 2. □

EXEMPLE 13. On sait que $\sqrt{2}$ est racine de $X^2 - 2$, tout comme $\sqrt{3}$ est racine de $X^2 - 3$. Un polynôme ayant pour racine $\sqrt{2} + \sqrt{3}$ est donné par

$$\mathrm{Res}_X(X^2 - 2, (T - X)^2 - 3) = T^4 - 10T^2 + 1.$$

Ces opérations ne distinguent pas les racines de A et B, les quatre racines du résultant sont donc $\pm\sqrt{2} \pm \sqrt{3}$.

EXEMPLE 14. Ramanujan a prouvé l'identité suivante

$$\sqrt[3]{\cos\frac{2\pi}{7}} + \sqrt[3]{\cos\frac{4\pi}{7}} + \sqrt[3]{\cos\frac{8\pi}{7}} = \sqrt[3]{\frac{5 - 3\sqrt[3]{7}}{2}}.$$

Pour prouver automatiquement cette identité, une méthode consiste à construire un polynôme annulant son membre gauche, puis à isoler la racine correspondant numériquement à sa valeur, et à vérifier qu'il s'agit bien du membre droit.

En posant $x = \exp(2i\pi/7)$, les trois racines cubiques sont annulées par les polynômes

$$P_1 = X^3 - \frac{x + x^{-1}}{2}, \quad P_2 = X^3 - \frac{x^2 + x^{-2}}{2}, \quad P_3 = X^3 - \frac{x^4 + x^{-4}}{2}.$$

Un premier polynôme $P_4(Y) = \mathrm{Res}_X(P_1(Y - X), P_2(X))$ de degré 9 est d'abord obtenu, puis $P_5 = \mathrm{Res}_X(P_4(Y - X), P_3(X))$ de degré 27, et enfin le résultant de $x^7 - 1$ et du numérateur de P_5 vu maintenant comme polynôme en x fournit

$$\begin{aligned}
&32768Y^6(Y - 3)(Y^2 + 3)^2(Y^2 + 3Y + 9)(Y^2 + 3Y + 3)^3(Y^2 - 3Y + 3)^3\\
&\qquad \times (4Y^9 - 30Y^6 + 75Y^3 + 32)^6\\
&\quad \times (64Y^{18} + 768Y^{15} + 7728Y^{12} + 17152Y^9 + 16008Y^6 + 3864Y^3 + 343)^6.
\end{aligned}$$

Une évaluation numérique montre que c'est le facteur

$$4Y^9 - 30Y^6 + 75Y^3 + 32$$

qui annule le membre gauche (il faut faire un peu attention : Ramanujan utilise la convention que la racine cubique d'un réel négatif est un réel négatif). On peut alors résoudre ce polynôme par les formules de Cardan pour découvrir le membre droit : la seule racine réelle est

$$-\sqrt[3]{\frac{3\sqrt[3]{7} - 5}{2}}.$$

On a donc non seulement prouvé l'identité, mais découvert automatiquement le membre droit.

Calcul du résultant en deux variables. On dispose à l'heure actuelle d'algorithmes rapides (de complexité quasi-optimale) pour le calcul du résultant avec une seule variable. Le meilleur algorithme connu actuellement pour le calcul du résultant en deux variables est une méthode d'évaluation-interpolation qui se ramène à une seule variable (voir l'exercice 9). Cet algorithme n'est pas quasi-optimal. En revanche, pour les trois premières opérations de la Proposition 4, des algorithmes quasi-optimaux à base de multiplication rapide de séries existent, ils ont été présentés au Chapitre 3 (Section 3.6, page 51).

2.5. Sous-résultants.

La croissance des coefficients dans l'algorithme d'Euclide. Le nombre d'opérations dans le corps des coefficients n'est pas une mesure suffisante de la complexité des calculs lorsque les opérations elles-mêmes ont une complexité variable. C'est le cas pour le calcul de pgcd dans $\mathbb{Q}[X]$ et dans $\mathbb{K}(Y)[X]$. Dans ces corps de coefficients, on constate empiriquement les phénomènes suivants :

- l'algorithme d'Euclide amène à faire des divisions, et introduit des dénominateurs au cours du calcul ;
- la taille des coefficients croît rapidement ;
- ces coefficients peuvent souvent se « simplifier ».

Exemple 15. L'exécution de l'algorithme d'Euclide sur

$$A = 115X^5 + 7X^4 + 117X^3 + 30X^2 + 87X + 44,$$
$$B = 91X^4 + 155X^3 + 3X^2 + 143X + 115.$$

produit les restes successifs suivants :

$$\frac{3601622}{8281}X^3 - \frac{1196501}{8281}X^2 + \frac{151912}{637}X + \frac{2340984}{8281}$$
$$\frac{189886027626841}{12971681030884}X^2 - \frac{57448278681703}{3242920257721}X - \frac{17501090665331}{3242920257721}$$
$$\frac{3748556212578804983806085060}{435414847094570987735100 1}X + \frac{14183336091512396932801 4892}{334934497765054605950077}.$$

En simplifiant ces restes par des multiplications par des constantes bien choisies, on obtient les restes :

$$3601622X^3 - 1196501X^2 + 1974856X + 2340984,$$
$$22930325761X^2 - 27749440252X - 8453612204,$$
$$288979986761465X + 142143002707719.$$

Il est souhaitable d'éviter le calcul sur les rationnels (ou les fractions rationnelles), pour lesquels l'addition requiert des calculs de multiplications et éventuellement de pgcd. Une première idée consiste alors à éviter totalement les divisions en utilisant des *pseudo-restes*.

Definition 2. Si A et B sont des polynômes de $\mathbb{A}[X]$ et b est le coefficient de tête de B, le *pseudo-reste* $\overline{R}$ de A et B est défini par

$$b^{\deg A - \deg B + 1} A = \overline{Q}B + \overline{R},$$

où $\overline{Q}, \overline{R} \in \mathbb{A}[X]$ avec $\deg \overline{R} < \deg \overline{Q}$.

Sous-résultants(A, B)

Entrée : A et B dans $\mathbb{A}[X]$ avec $\deg B < \deg A$.

Sortie : Le dernier sous-résultant non nul.

1. $f = g = s = 1$
2. Tant que $\deg B > 0$
   (a) $d = \deg A - \deg B$
   (b) $R =$pseudoReste(A, B)
   (c) Si les degrés de A et B sont impairs, faire $s = -s$.
   (d) Faire $A := B$ et mettre dans B le quotient de la division de R par fg^d
   (e) Prendre $f :=$coefficientDominant(A)
   (f) Mettre à jour g en lui affectant le quotient de la division de f^d par g^{d-1}
3. $d = \deg A$;
4. Renvoyer le quotient de la division de sB^d par g^{d-1}.

FIGURE 6. L'algorithme des sous-résultants.

Remplacer les calculs de restes de l'algorithme d'Euclide par des calculs de pseudo-restes évite d'introduire des dénominateurs. Cette idée seule n'est pas suffisante : les coefficients de ces pseudo-restes croissent trop.

EXEMPLE 16. Sur le même exemple, la modification de l'algorithme d'Euclide produit la suite de pseudo-restes

$$3601622X^3 - 1196501X^2 + 1974856X + 2340984,$$
$$189886027626841X^2 - 229793114726812X - 70004362661324,$$
$$257057096083899261191107914552182660X$$
$$+126440823512296156588839626542149756.$$

Une possibilité pour éviter cette croissance est de diviser ces polynômes par le pgcd de leurs coefficients à chaque étape. C'est ce que nous avons fait dans l'exemple 15 ci-dessus. On parle alors de suite de pseudo-restes *primitifs*. Cependant, ces calculs de pgcd de coefficients sont trop coûteux.

L'algorithme des sous-résultants donné en Figure 6 est une modification de l'algorithme d'Euclide qui évite les divisions, tout en *prévoyant* des facteurs communs qui apparaissent dans les coefficients de la suite des pseudo-restes de sorte à limiter leur croissance. Ces pseudo-restes sont appelés des sous-résultants. Dans cet algorithme, on se contente de renvoyer le dernier sous-résultant non nul ; on conçoit aisément comment modifier l'algorithme pour obtenir *n'importe* quel sous-résultant (spécifié par des conditions de degré, par exemple).

EXEMPLE 17. Toujours sur les mêmes polynômes, la suite des sous-résultants redonne les polynômes que nous avions déjà calculés par simplification :

$$3601622X^3 - 1196501X^2 + 1974856X + 2340984,$$
$$22930325761X^2 - 27749440252X - 8453612204,$$
$$288979986761465X + 142143002707719.$$

La suite des sous-résultants de cet exemple est donc primitive : les facteurs prédits par l'algorithme du sous-résultant ont suffi à éliminer tout facteur commun entre les coefficients des pseudo-restes.

Comme le résultant, les sous-résultants sont liés à la matrice de Sylvester et à l'algorithme d'Euclide. Une formule de Cramer sur une sous-matrice de la matrice de Sylvester donne le résultat suivant.

PROPOSITION 5. *Toutes les divisions effectuées au cours de l'algorithme des sous-résultants sont exactes.*

La preuve de ce résultat est technique et omise ici.

En corollaire, on remarque en outre qu'il est assez facile de déduire de ce résultat des estimations sur la « taille » des coefficients qui apparaissent au cours de l'algorithme. Un cas particulier intéressant est celui où $\mathbb{A}$ est l'anneau de polynômes $\mathbb{K}[Y]$. Alors, si A et B ont des degrés totaux m et n, tous les sous-résultants ont des degrés bornés par mn. Ces bornes permettent un calcul du résultant par évaluation-interpolation dès que l'on dispose d'un algorithme efficace à une variable.

Calcul de la paramétrisation. Les sous-résultants permettent de finir la « résolution » (commencée en page 89) des systèmes de deux polynômes à deux variables. Pour simplifier, nous faisons l'hypothèse que pour tout x racine du résultant $R = \mathrm{Res}_Y(A, B)$, il n'y a qu'un seul y tel que $A(x, y) = B(x, y) = 0$. Dans ce cas, on peut montrer que le sous-résultant S_1 (de degré 1 en Y) est non nul ; écrivons-le sous la forme $S_1 = P_0(X)Y + Q_0(X)$. Comme pour le résultant, il existe des polynômes U et V dans $\mathbb{K}[X, Y]$ tels qu'on ait l'égalité

$$AU + BV = P_0(X)Y + Q_0(X).$$

On en déduit donc que toutes les solutions (x, y) du système $A = B = 0$ satisfont l'équation $P_0(x)y = Q_0$. Autrement dit, en écartant les solutions dégénérées où $P_0(x) = 0$, on obtient l'ordonnée des points solutions en évaluant la fraction rationnelle $-Q_0/P_0$ sur les racines du résultant R. Autrement dit, cette procédure permet de décrire les solutions du système sous la forme

$$\begin{cases} y = Q(x) \\ R(x) = 0 \end{cases}$$

Géométriquement, ce polynôme Q permet de retrouver y à partir de x, il permet donc d'effectuer l'opération symbolisée sur la Figure 7, la paramétrisation des solutions du système par les racines de R.

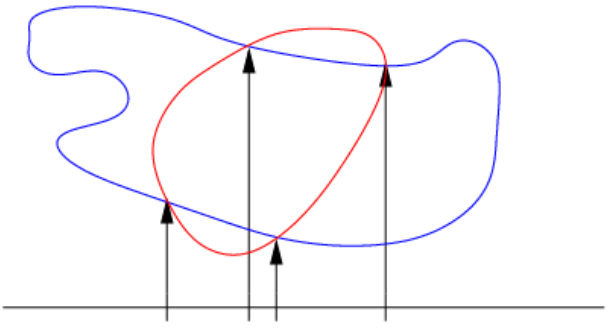


FIGURE 7. Paramétrisation des solutions d'un système algébrique par les racines d'un résultant.

EXEMPLE 18. Sur les mêmes polynômes qu'à l'exemple 8, le sous-résultant redonne la paramétrisation calculée à partir de l'algorithme d'Euclide, mais en étant plus économe dans les opérations sur les coefficients.

3. Algorithme d'Euclide rapide

Le point central de cette section est la présentation d'un algorithme rapide pour le pgcd, qui est un algorithme récursif à base de multiplications et de divisions euclidiennes rapides. Historiquement, c'est d'abord un algorithme rapide pour le pgcd d'entiers (le « demi-pgcd ») qui est apparu, il a ensuite été étendu aux polynômes. Nous décrivons maintenant l'algorithme dit du « demi-pgcd ». Pour simplifier, nous nous plaçons dans le cas du pgcd de polynômes, mais cette approche s'étend au cas des entiers.

Soient donc A et B dans $\mathbb{K}[X]$, avec $n = \deg A$ et $\deg B < n$ (ce qui n'est pas une forte restriction). Posons comme précédemment $R_0 = A$, $R_1 = B$. Soient ensuite R_i les restes successifs de l'algorithme d'Euclide et soit en particulier $R_N = \operatorname{pgcd}(A, B)$ le dernier non nul d'entre eux. D'après les égalités en page 86, il existe une matrice 2×2 $M_{A,B}$ de cofacteurs telle que :

$$M_{A,B} \begin{bmatrix} R_0 \\ R_1 \end{bmatrix} = \begin{bmatrix} U_N & V_N \\ U_{N+1} & V_{N+1} \end{bmatrix} \begin{bmatrix} R_0 \\ R_1 \end{bmatrix} = \begin{bmatrix} R_N \\ 0 \end{bmatrix}.$$

L'algorithme de pgcd rapide calcule d'abord cette matrice ; à partir de là, on en déduit aisément le pgcd, pour $O(\mathsf{M}(n))$ opérations supplémentaires.

Notons qu'une étape de l'algorithme d'Euclide classique peut elle aussi s'écrire sous une forme matricielle. En effet, si Q_i est le quotient de la division de R_{i-1} par R_i, on peut écrire :

$$\begin{bmatrix} 0 & 1 \\ 1 & -Q_i \end{bmatrix} \begin{bmatrix} R_{i-1} \\ R_i \end{bmatrix} = \begin{bmatrix} R_i \\ R_{i+1} \end{bmatrix}.$$

Ainsi, la matrice $M_{A,B}$ est un produit de matrices élémentaires de ce type. Elle est inversible.

3.1. Demi-pgcd : définition. Comme étape intermédiaire pour obtenir une division euclidienne rapide, on utilise l'algorithme dit du « demi-pgcd » (*half-gcd* en anglais, d'où la notation HGCD), qui permet de faire des « pas de géant » dans la liste des restes successifs.

Le demi-pgcd est défini comme suit. Les degrés des restes successifs R_i décroissent strictement, donc il existe un unique indice j tel que :

- $\deg R_j \geq \frac{n}{2}$;
- $\deg R_{j+1} < \frac{n}{2}$.

On a vu en page 86 qu'il existe $U_j, V_j, U_{j+1}, V_{j+1}$ tels que :

$$U_j R_0 + V_j R_1 = R_j \quad \text{et} \quad U_{j+1} R_0 + V_{j+1} R_1 = R_{j+1},$$

ces polynômes étant en outre uniques si on rajoute les conditions de degré voulues sur la relation de Bézout. Ceci peut se réécrire sous la forme matricielle suivante :

$$\begin{bmatrix} U_j & V_j \\ U_{j+1} & V_{j+1} \end{bmatrix} \begin{bmatrix} R_0 \\ R_1 \end{bmatrix} = \begin{bmatrix} R_j \\ R_{j+1} \end{bmatrix}.$$

Pour fixer les idées, en première approximation, on peut estimer que les polynômes $U_j, V_j, U_{j+1}, V_{j+1}$ ont des degrés de l'ordre de $\frac{n}{2}$ (attention, ce n'est qu'une estimation, pas une égalité).

Notons M_{hgcd} la matrice ci-dessus donnant les restes R_j et R_{j+1}. L'algorithme du demi-pgcd (noté HGCD ci-dessous) a pour vocation de calculer cette matrice. Avant d'en étudier le fonctionnement, voyons comment il permet d'obtenir le calcul du pgcd étendu. L'idée est qu'un appel à HGCD en degré n permet d'obtenir les restes de degré approximativement $\frac{n}{2}$; alors, un appel en degré $\frac{n}{2}$ permet d'obtenir les restes de degré approximativement $\frac{n}{4}$, et ainsi de suite jusqu'à trouver le pgcd. L'algorithme est détaillé en Figure 8.

Euclide via HGCD

Entrée : $R_0 = A$ et $R_1 = B$ dans $\mathbb{K}[X]$ avec $\deg B < \deg A = n$.

Sortie : La matrice $M_{A,B}$.

1. Soit $M_{\text{hgcd}} = \mathsf{HGCD}(A, B)$.
2. Calculer R_j et R_{j+1}.
3. Si $R_{j+1} = 0$, renvoyer M_{hgcd}.
4. Soit M la matrice de la division euclidienne de R_j par R_{j+1}, et R_{j+2} le reste correspondant.
5. Si $R_{j+2} = 0$, renvoyer MM_{hgcd}.
6. Calculer $M_{R_{j+1},R_{j+2}}$ par un appel récursif, et renvoyer $M_{R_{j+1},R_{j+2}} M M_{\text{hgcd}}$.

FIGURE 8. Algorithme d'Euclide rapide via le demi-pgcd rapide.

À l'étape 4, R_j et R_{j+1} ont par définition des degrés qui encadrent $\frac{n}{2}$, mais on n'a pas de borne supérieure sur le degré de R_j, qui peut être proche de n. Aussi, pour obtenir deux polynômes de degré au plus $\frac{n}{2}$ pour l'appel récursif, il est nécessaire d'effectuer une étape de division euclidienne.

Soit $\mathsf{H}(n)$ la complexité de l'algorithme de demi-pgcd sur des entrées de degré au plus n. Pour estimer la complexité de l'algorithme de pgcd rapide, on fait l'hypothèse que $\mathsf{H}(n+n') \geq \mathsf{H}(n)+\mathsf{H}(n')$ et que $\mathsf{M}(n)$ est négligeable devant $\mathsf{H}(n)$. Ces hypothèses sont vérifiées pour l'algorithme proposé en Section 3.2.

PROPOSITION 6. *L'algorithme ci-dessus calcule $M_{A,B}$ en $O(\mathsf{H}(n))$ opérations.*

DÉMONSTRATION. La validité de l'algorithme est immédiate, puisque toutes les matrices calculées ne sont au fond que des matrices qui font la transition de deux restes successifs à deux autres restes successifs. Multiplier ces matrices permet de composer ces opérations de transition.

Estimons la complexité. Le coût de l'appel à HGCD est $\mathsf{H}(n)$. Ensuite, toutes les opérations des étapes 2 à 5 ont une complexité en $O(\mathsf{M}(n))$, en utilisant pour l'étape 4 une division euclidienne rapide (cf. Théorème 1 en page 62). On effectue ensuite un appel récursif, puis de nouveau des opérations dont le coût est en $O(\mathsf{M}(n))$. Ainsi, la complexité $\mathsf{B}(n)$ de l'algorithme de pgcd rapide satisfait à la récurrence :

$$\mathsf{B}(n) \leq \mathsf{B}(n/2) + \mathsf{H}(n) + C\mathsf{M}(n),$$

C étant une constante. Le théorème « diviser pour régner » permet de conclure. □

Autrement dit, le coût du pgcd est, à une constante près, le même que celui du demi-pgcd. Il devient donc légitime de se consacrer uniquement à ce dernier.

3.2. Demi-pgcd : algorithme. Pour mettre en place une stratégie « diviser pour régner » pour le demi-pgcd, on utilise un moyen de « couper les polynômes » en deux. L'idée est de ne garder que les coefficients de poids forts des polynômes d'entrée et d'utiliser le fait que *le quotient de la division euclidienne de deux polynômes ne dépend que de leurs coefficients de poids fort*, d'une façon tout à fait quantifiée par la suite.

Remarquons qu'on accède aux coefficients de poids fort d'un polynôme en prenant son quotient par un polynôme de la forme X^k : par exemple, si A est

$$X^{10} + 2X^9 + 5X^8 + 3X^7 + 11X^6 + \cdots,$$

HGCD

Entrée : A et B dans $\mathbb{K}[X]$, avec $\deg B < \deg A = n$.

Sortie : La matrice M_{hgcd} de A et B.

1. Soit $m = \lceil \frac{n}{2} \rceil$. Si $\deg B < m$, renvoyer la matrice identité.
   C'est terminé.
2. Soient f et g les quotients respectifs de A et B par X^m.
   f et g ont des degrés approximativement $n/2$.
3. Soient $M = \mathsf{HGCD}(f, g)$, et A', B' définis par
$$M \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} A' \\ B' \end{bmatrix}.$$
4. Si $\deg B' < m$, renvoyer M.
   C'est terminé.
5. Soient $C' = A' \bmod B'$, et M' la matrice de la division euclidienne correspondante.
   Moralement, B' a un degré de l'ordre au pire de $\frac{3n}{4}$. Par contre, A' peut avoir un degré plus grand. L'étape de division euclidienne permet de se ramener à des polynômes de degrés de l'ordre de $\frac{3n}{4}$.
6. Soient $\ell = 2m - \deg B'$, b et c les quotients respectifs de B' et C' par X^ℓ.
   ℓ est de l'ordre de $\frac{n}{4}$, de sorte de b et c on des degrés de l'ordre $\frac{n}{2}$.
7. Soit $M'' = \mathsf{HGCD}(b, c)$. Renvoyer la matrice $M''M'M$.

FIGURE 9. Algorithme du demi-pgcd rapide.

le quotient de A par X^6 est

$$X^4 + 2X^3 + 5X^2 + 3X + 11.$$

Voyons comment cette idée permet d'obtenir notre algorithme. Étant donnés deux polynômes A et B de degrés de l'ordre de n, on leur associe f et g de degré approximativement $\frac{n}{2}$, en nommant f et g les quotients respectifs de A et B par $X^{\frac{n}{2}}$: on a jeté les coefficients de petits degrés.

On calcule la matrice M du demi-pgcd de f et g (moralement, les éléments de cette matrice ont degré $\frac{n}{4}$). La remarque ci-dessus permet de montrer qu'on obtient ainsi une matrice de transition vers les restes de A et B eux-mêmes.

On applique donc cette matrice aux polynômes initiaux ; on obtient des restes B' et C' dont les degrés sont de l'ordre de $\frac{3n}{4}$. On effectue alors une seconde fois le même type d'opération, avec un appel récursif en degré $\frac{n}{2}$, pour gagner à nouveau $\frac{n}{4}$, et finalement arriver en degré $\frac{n}{2}$.

Les choix exacts des degrés de troncature sont subtils, et on admettra que les valeurs données en Figure 9 permettent d'assurer la correction de l'algorithme.

PROPOSITION 7. *L'algorithme ci-dessus calcule la matrice du demi-pgcd de A et B en $O(\mathsf{M}(n)\log(n))$ opérations dans $\mathbb{K}$.*

DÉMONSTRATION. Comme indiqué plus haut, on admet que les choix des degrés de troncature permettent d'assurer la validité de l'algorithme. Il est plus facile d'en effectuer l'analyse de complexité. Le coût $\mathsf{H}(n)$ peut être majoré par deux fois le coût en degré au plus $\frac{n}{2}$ (les deux appels ont lieu aux lignes 3 et 7), plus un certain nombre de multiplications de polynômes et une division euclidienne. En remarquant

que tous les produits se font en degré au pire n, on en déduit la récurrence

$$\mathsf{H}(n) \leq 2\mathsf{H}(n/2) + C\mathsf{M}(n),$$

où C est une constante. La conclusion découle comme d'habitude du lemme « diviser pour régner ». □

3.3. Complément : calcul d'un reste choisi. On peut en tirer un raffinement fort utile de l'algorithme de demi-pgcd : le calcul d'un reste particulier (sélectionné par une condition de degré), ainsi que des cofacteurs associés.

THÉORÈME 3. *Soient $R_0 = A$ et $R_1 = B$, avec $\deg A = n$ et $\deg B < \deg A$, et soient R_i les restes successifs de l'algorithme d'Euclide appliqué à A et B.*

Soit $\ell < n$, et R_j le premier reste de degré inférieur à ℓ. On peut calculer R_j et les cofacteurs associés U_j et V_j en $O(\mathsf{M}(n)\log(n))$ opérations de $\mathbb{K}$.

DÉMONSTRATION (SUCCINTE). Si ℓ est plus petit que $\frac{n}{2}$, on fait un appel à HGCD pour se ramener en degré $\frac{n}{2}$, et on effectue un appel récursif. Si ℓ est plus grand que $\frac{n}{2}$, on fait un appel à HGCD sur les polynômes divisés par $X^{n-2\ell}$. □

3.4. Le cas des entiers. Les mêmes idées algorithmiques s'étendent au calcul sur les entiers, mais il est beaucoup plus difficile d'écrire un algorithme correct dans ce cas. On se contentera d'énoncer les résultats de complexité suivants :

THÉORÈME 4. *Soient $R_0 = A \in \mathbb{N}$, $R_1 = B \in \mathbb{N}$, avec $B \leq A$ et R_i les restes de l'algorithme d'Euclide appliqué à A et B. On peut calculer :*

- *le pgcd de A et B,*
- *le premier reste inférieur à un entier $\ell < A$,*

ainsi que les cofacteurs associés en $O(\mathsf{M}_{\mathbb{Z}}(\log A)\log(\log(A)))$ opérations binaires.

Exercices

EXERCICE 6 (Factorisation sans carré). Soit $\mathbb{K}$ un corps de caractéristique nulle. Si $F \in \mathbb{K}[X]$ se factorise en irréductibles comme $\prod_i f_i^{\alpha_i}$, le polynôme $\overline{F} = \prod_i f_i$ est appelé la *partie sans carré de F*. Si $n = \deg(F)$, montrer qu'on peut calculer les coefficients de la partie sans carré de F à partir des coefficients de F en $O(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$.

EXERCICE 7. Soient $f, g \in \mathbb{K}[X]$ des polynômes unitaires.

1. Soit $N \in \mathbb{N} \setminus \{0\}$. Montrer que l'unique polynôme unitaire de $\mathbb{K}[X]$ dont les racines sont les puissances N-ièmes des racines de f peut être obtenu par un calcul de résultant.
2. Si f est le polynôme minimal d'un nombre algébrique α, montrer qu'on peut déterminer un polynôme annulateur de $g(\alpha)$ à l'aide d'un résultant.
3. Calculer le polynôme minimal sur $\mathbb{Q}$ de $\alpha = \sqrt[3]{4} + \sqrt[3]{2} + 1$.

EXERCICE 8. Soit $f \in \mathbb{K}[X]$ un polynôme unitaire de degré $d \geq 1$. Pour $N \geq 1$, on note $G_N(f)$ l'unique polynôme unitaire de degré d dont les racines sont les puissances N-ièmes des racines de f.

(a) Exprimer $G_N(f)$ à l'aide d'un résultant de polynômes à deux variables.

(b) Justifier l'appartenance à $\mathbb{K}[X]$ du polynôme $G_N(f)$.

(c) Utiliser (a) pour donner un algorithme pour le calcul de $G_N(f)$; estimer sa complexité.

(d) Montrer que $G_2(f)$ peut se calculer en $O(\mathsf{M}(d))$ opérations dans $\mathbb{K}$.

(e) Si N est une puissance entière de 2, montrer qu'on peut calculer $G_N(f)$ en $O(\mathsf{M}(d)\log(N))$ opérations dans $\mathbb{K}$.

Exercice 9 (Résultants en deux variables). Soient $F, G \in \mathbb{K}[X,Y]$ ayant des degrés en X et en Y bornés par un entier $D \geq 1$. Montrer que le polynôme $R(Y) := \mathrm{Res}_X(F,G)$ est de degré au plus $2D^2$. Donner un algorithme de type évaluation-interpolation pour le calcul des coefficients de R et estimer son coût en opérations dans $\mathbb{K}$.

Notes

Pour les déterminants sur des anneaux, nous renvoyons au livre de Lang [19].

L'introduction du résultant et des sous-résultants remonte à Bézout et Sylvester [2, 34, 35]. Ces notions ont été revisitées en calcul formel dans les années 1965 par Collins [7, 8, 9], Brown et Traub [4, 5]. L'article de synthèse [10] retrace leur histoire. De nombreuses propriétés des résultants et de l'algorithme d'Euclide peuvent sont établies de manière élémentaire à l'aide des fonctions symétriques par Lascoux [20]. On peut consulter les livres [11, 12, 18], ou [19] pour une approche un peu plus complète.

L'idée de base l'algorithme de demi-pgcd (le quotient de la division euclidienne de deux entiers ne dépend que de leurs chiffres de poids fort) est due à Lehmer [21]. Knuth [17] en a fait le premier algorithme sous-quadratique, qui calcule le pgcd de deux entiers de taille binaire n en complexité binaire $O(\mathsf{M}_{\mathbb{Z}}(n)\log^4(n))$. Schönhage [28] a montré comment abaisser cette complexité à $O(\mathsf{M}_{\mathbb{Z}}(n)\log(n))$. Son algorithme a été adapté au cas polynomial par Moenck [25], mais seulement pour le cas où la suite des restes est normale. Brent, Gustavson et Yun [3, 14] ont donné un algorithme de type LKS (Lehmer-Knuth-Schönhage) pour le cas polynomial général, de complexité arithmétique $O(\mathsf{M}(n)\log(n))$ en degré n. Cet algorithme utilise $O(n\log n)$ opérations non scalaires. Strassen [32, 33] a montré que $\Omega(n\log n)$ opérations non scalaires sont aussi nécessaires, du moins pour une entrée générique, donc l'algorithme LKS est optimal de ce point de vue.

Schwartz [30] a montré que le résultant de deux polynômes peut se calculer également en $O(\mathsf{M}(n)\log(n))$ opérations arithmétiques. Schönhage a donné un algorithme (probabiliste) pour le calcul rapide du pgcd de deux polynômes de $\mathbb{Z}[X]$. Pour deux polynômes de degré borné par n et coefficients d'au plus ℓ chiffres, l'algorithme a une complexité binaire $\tilde{O}(n(n+\ell))$, il est donc quasi-optimal (en la taille binaire de l'entrée) lorsque $\ell > n$ [29].

L'exercice 6 est inspiré par un résultat de Yun [37], qui montre qu'il est possible de déterminer toute la factorisation sans carré en la même complexité. En fait, les problèmes du calcul du pgcd et de la factorisation sans carré sont équivalents du point de vue de la complexité [38].

Il est très difficile de trouver une référence complète, lisible et sans erreur sur les algorithmes de pgcd rapide. Le Chapitre 11 de l'ouvrage de von zur Gathen et Gerhard [11] fournit une bonne partie des résultats techniques implicitement utilisés dans la Section 3, mais l'algorithme qu'il propose est erroné. Les notes du livre de Yap [36] nous ont beaucoup inspirés, mais elles contiennent des erreurs pour le pgcd entier. On pourra également consulter des articles plus récents [26, 27, 31].

Ces algorithmes de pgcd rapide sont aussi assez délicats à implanter de manière efficace. Par exemple, pour le pgcd de polynômes, une bonne implantation doit prendre en compte les algorithmes de multiplication utilisés. Si on utilise la FFT, il est possible d'économiser des transformées de Fourier directes et inverses ; il faut alors régler le nombre de points d'évaluation en fonction des degrés du résultat final, et pas des résultats intermédiaires, etc. Si l'on travaille sur un corps

fini « raisonnable » (les coefficients faisant de quelques bits jusqu'à quelques dizaines voire centaines de bits), on peut estimer que le seuil au-delà duquel utiliser l'algorithme rapide se situe autour des degrés 200 ou 300. Il faut noter que très peu de systèmes disposent de telles implantations (c'est le cas pour Magma et la bibliothèque NTL). En ce qui concerne le pgcd des entiers, la situation est sensiblement plus délicate, toujours à cause du problème des retenues (une bonne partie des algorithmes présentés dans les ouvrages de référence sont incorrects, à cause de ce problème). Pour donner une idée, dans les systèmes Magma, Mathematica, ou des bibliothèques telles que GMP (code toujours en développement et utilisé entre autres par les entiers de Maple, Mathematica et Sage), le seuil se situe autour de nombres de 30 000 bits (10 000 chiffres décimaux).

Le pgcd et le pgcd étendu peuvent aussi être définis dans un contexte non-commutatif. Ils sont alors utiles au calcul avec des opérateurs différentiels ou de récurrence. Le résultant différentiel de deux équations différentielles a été défini par Berkovich et Tsirulik dans [1]. Chardin [6] a défini les sous-résultants de deux équations différentielles. Li [23, 24] a étendu la définition aux polynômes de Ore. On trouve dans [13, 22] des algorithmes efficaces pour le calcul du pgcd de deux opérateurs différentiels ; ces algorithmes ont été étendus aux sous-résultants dans [23, 24]. Dans [6], Chardin a donné une expression du résultant de deux équations différentielles en termes de leurs solutions. Cela a été généralisé au cadre des polynômes de Ore par Hong [15, 16].

Bibliographie

[1] Berkovich, L. M. et V. G. Tsirulik (1986). « Differential resultants and some of their applications ». In : *Differentsial′nye Uravneniya*, vol. 22, n°5. English translation in : Differential Equations, Plenum Publ. Corp., Vol. 22, no. 5, pp. 530–536. NY Plenum 1986, p. 750–757, 914.

[2] Bézout, É. (1764). « Recherches sur le degré des équations résultantes de l'évanouissement des inconnues ». In : *Histoire de l'académie royale des sciences*, p. 288–338.

[3] Brent, Richard P., Fred G. Gustavson et David Y. Y. Yun (1980). « Fast solution of Toeplitz systems of equations and computation of Padé approximants ». In : *Journal of Algorithms*, vol. 1, n°3, p. 259–295.

[4] Brown, W. S. (1971). « On Euclid's algorithm and the computation of polynomial greatest common divisors ». In : *SYMSAC'71 : ACM Symposium on Symbolic and Algebraic Manipulation*. Los Angeles, California, United States : ACM, p. 195–211. doi : 10.1145/800204.806288.

[5] Brown, W. S. et J. F. Traub (1971). « On Euclid's algorithm and the theory of subresultants ». In : *Journal of the Association for Computing Machinery*, vol. 18, p. 505–514.

[6] Chardin, Marc (1991). « Differential resultants and subresultants ». In : *Fundamentals of computation theory*. Vol. 529. Lecture Notes in Computer Science. Gosen : Springer-Verlag, p. 180–189.

[7] Collins, G. E. (1966). « Polynomial remainder sequences and determinants ». In : *The American Mathematical Monthly*, vol. 73, p. 708–712.

[8] Collins, George E. (1967). « Subresultants and reduced polynomial remainder sequences ». In : *Journal of the Association for Computing Machinery*, vol. 14, p. 128–142.

[9] — (1971). « The calculation of multivariate polynomial resultants ». In : *Journal of the Association for Computing Machinery*, vol. 18, p. 515–532.

[10] Gathen, Joachim von zur et Thomas Lücking (2003). « Subresultants revisited ». In : *Theoretical Computer Science*, vol. 297, n°1-3, p. 199–239.

[11] Gathen, Joachim von zur et Jürgen Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press, xiv+785 pages.

[12] Geddes, Keith O., Stephen R. Czapor et George Labahn (1992). *Algorithms for Computer Algebra*. Kluwer Academic Publishers.

[13] Grigoriev, D. Yu. (1990). « Complexity of factoring and calculating the GCD of linear ordinary differential operators ». In : *Journal of Symbolic Computation*, vol. 10, n°1, p. 7–37.

[14] Gustavson, Fred G. et David Y. Y. Yun (1979). « Fast algorithms for rational Hermite approximation and solution of Toeplitz systems ». In : *IEEE Transactions on Circuits and Systems*, vol. 26, n°9, p. 750–755.

[15] Hong, Hoon (2001a). « Ore principal subresultant coefficients in solutions ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, n°3, p. 227–237.

[16] — (2001b). « Ore subresultant coefficients in solutions ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 12, n°5, p. 421–428.

[17] Knuth, Donald E. (1971). « The analysis of algorithms ». In : *Actes du Congrès International des Mathématiciens*. Vol. 3. Nice : Gauthier-Villars, p. 269–274.

[18] — (1997). *The Art of Computer Programming*. 3e éd. Vol. 2 : Seminumerical Algorithms. Computer Science and Information Processing. Addison-Wesley Publishing Co., xiv+762 pages.

[19] Lang, Serge (2002). *Algebra*. 3e éd. Vol. 211. Graduate Texts in Mathematics. Springer-Verlag, xvi+914 pages.

[20] Lascoux, Alain (2003). *Symmetric functions and combinatorial operators on polynomials*. Vol. 99. CBMS Regional Conference Series in Mathematics. Published for the Conference Board of the Mathematical Sciences, Washington, DC, xii+268 pages.

[21] Lehmer, D. H. (1938). « Euclid's Algorithm for Large Numbers ». In : *The American Mathematical Monthly*, vol. 45, n°4, p. 227–233.

[22] Li, Z. et I. Nemes (1997). « A modular algorithm for computing greatest common right divisors of Ore polynomials ». In : *ISSAC'97 : International Symposium on Symbolic and Algebraic Computation*. Maui, Hawai : ACM Press, p. 282–289.

[23] Li, Ziming (1998). « A subresultant theory for Ore polynomials with applications ». In : *ISSAC'98 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 132–139.

[24] — (2000). « Greatest common right divisors, least common left multiples, and subresultants of Ore polynomials ». In : *Mathematics mechanization and applications*. San Diego, CA : Academic Press, p. 297–324.

[25] Moenck, R. T. (1973). « Fast computation of GCDs ». In : *STOC'73 : ACM Symposium on Theory of Computing*. Austin : ACM, p. 142–151.

[26] Möller, Niels (2008). « On Schönhage's algorithm and subquadratic integer GCD computation ». In : *Mathematics of Computation*, vol. 77, n°261, p. 589–607.

[27] Pan, V. Y. et X. Wang (2002). « Acceleration of Euclidean algorithm and extensions ». In : *ISSAC'02 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Teo Mora. Lille : ACM Press, p. 207–213.

[28] Schönhage, A. (1971). « Schnelle Berechnung von Kettenbruchentwicklugen ». In : *Acta Informatica*, vol. 1, p. 139–144.

[29] — (1988). « Probabilistic computation of integer polynomial GCDs ». In : *Journal of Algorithms*, vol. 9, n°3, p. 365–371.

[30] Schwartz, J. T. (1980). « Fast probabilistic algorithms for verification of polynomial identities ». In : *Journal of the Association for Computing Machinery*, vol. 27, n°4, p. 701–717.

[31] Stehlé, D. et P. Zimmermann (2004). « A binary recursive Gcd algorithm ». In : *ANTS-VI*. Vol. 3076. Lecture Notes in Computer Science. Springer-Verlag, p. 411–425.

[32] Strassen, V. (1981). « The computational complexity of continued fractions ». In : *SYMSAC'81*. Snowbird, Utah, United States : ACM, p. 51–67. doi : 10.1145/800206.806371.

[33] — (1983). « The computational complexity of continued fractions ». In : *SIAM Journal on Computing*, vol. 12, n°1, p. 1–27.

[34] Sylvester, James Joseph (1839). « On rational derivation from equations of coexistence, that is to say, a new and extended theory of elimination, *1839–40* ». In : *The Collected mathematical papers of James Joseph Sylvester*. Baker, H. F., p. 40–53.

[35] — (1840). « A method of determining by mere inspection the derivatives from two equations of any degree ». In : *Philosophical Magazine Series 3*, vol. 16, n°101, p. 132–135.

[36] Yap, Chee (2000). *Fundamental Problems in Algorithmic Algebra*. Oxford University Press.

[37] Yun, David Y.Y. (1976). « On square-free decomposition algorithms ». In : *SYMSAC'76*. Yorktown Heights, New York, United States : ACM, p. 26–35. doi : 10.1145/800205.806320.

[38] — (1977). « On the equivalence of polynomial GCD and squarefree factorization problems ». In : *MACSYMA Users' Conference*. NASA, Langley Res. Center, Washington D.C., United States, p. 65–70.

CHAPITRE 7

Approximants de Padé et de Padé-Hermite

Résumé

L'algorithme d'Euclide étendu permet le calcul d'approximants de Padé. Plus généralement, il permet la reconstruction des fractions rationnelles. Les approximants de Padé-Hermite sont une généralisation des approximants de Padé. Leur calcul peut s'effectuer grâce à un algorithme qui peut être vu comme une généralisation de l'algorithme d'Euclide étendu. L'approximation de Padé-Hermite rend possible la reconstruction d'équations linéaires à coefficients polynomiaux reliant des séries formelles.

Un premier problème abordé dans ce chapitre est le calcul d'approximants de Padé. Plus généralement, on s'intéresse à la *reconstruction rationnelle*, dont un autre cas particulier important est l'interpolation des fractions rationnelles. En Section 1, nous ramenons la reconstruction rationnelle à l'algorithme d'Euclide étendu, et l'appliquons à la reconnaissance d'une suite récurrente linéaire à partir de ses premiers termes (algorithme de Berlekamp-Massey). Les applications en sont nombreuses : pour la résolution de systèmes différentiels à coefficients constants (Chapitre 13), pour le calcul de polynômes minimaux de matrices creuses (Chapitre 9), pour la résolution de systèmes linéaires à coefficients polynomiaux (Chapitre 11).

Un second problème, traité en Section 2, est le calcul d'approximants de Padé-Hermite. Il peut s'effectuer grâce à un algorithme qui généralise celui pour le calcul des approximants de Padé. L'approximation de Padé-Hermite permet d'effectuer la reconstruction de polynômes annulant une série algébrique, et d'opérateurs différentiels annulant une série D-finie. Il permet également de deviner une récurrence à coefficients polynomiaux d'ordre arbitraire à partir des premiers termes de la suite.

1. Reconstruction rationnelle

Definition 1. Soit $\mathbb{K}$ un corps, $A \in \mathbb{K}[X]$ un polynôme de degré $n > 0$ et $B \in \mathbb{K}[X]$ de degré $< n$. Pour un $k \in \{1, \ldots, n\}$ fixé, la *reconstruction rationnelle de B modulo A* est la recherche d'un couple de polynômes $(R, V) \in \mathbb{K}[X]^2$ vérifiant :

$$\text{(RR)} \qquad \operatorname{pgcd}(V, A) = 1, \quad \deg(R) < k, \quad \deg(V) \leq n - k \quad \text{et} \quad \frac{R}{V} \equiv B \bmod A.$$

On parle d'*approximation de Padé* lorsque $A = X^n$, et d'*interpolation de Cauchy* lorsque $A = \prod_i (X - u_i)$ avec $u_1, \ldots, u_n \in \mathbb{K}$ deux à deux distincts.

Remarque. Si $k = n$, alors clairement $(R, V) = (B, 1)$ est une solution du problème (RR).

Si $k < n$, il est possible que (RR) n'admette aucune solution. C'est le cas par exemple en prenant $n = 3, k = 2$ et $A = X^3, B = X^2 + 1 \in \mathbb{K}[X]$. En effet, si $V(X) = aX + b$ avec $b \neq 0$, alors $R \equiv (aX + b)(X^2 + 1) = bX^2 + aX + b \bmod X^3$, ce qui est incompatible avec $\deg(R) \leq 1$.

Par ailleurs, si (RR) admet une solution (R, V), alors la fraction rationnelle R/V est forcément unique. En effet, si $(R_1, V_1) \in \mathbb{K}[X]^2$ est une autre solution de (RR), alors $R_1/V_1 \equiv R/V \bmod A$, donc A divise $R_1V - V_1R$. Or, le polynôme $R_1V - V_1R$

ayant un degré strictement inférieur à celui de A, il doit être identiquement nul. Donc les fractions R/V et R_1/V_1 coïncident.

Un problème plus simple. Si $R, V \in \mathbb{K}[X]$ sont tels que (R, V) est solution du problème (RR), alors (R, V) vérifie aussi le problème plus simple

$$\text{(RRS)} \qquad \deg(R) < k, \quad \deg(V) \leq n-k \quad \text{et} \quad R \equiv VB \bmod A,$$

où, à la différence de (RR), on a mis de côté la contrainte sur le pgcd de A et de V.

Remarquons tout de suite que le problème (RRS) admet *toujours* une solution non triviale $(R, V) \neq (0, 0)$: en effet, il se traduit en termes d'algèbre linéaire en un système linéaire homogène ayant $k + (n-k+1) = n+1$ inconnues (les coefficients de R et de V) et n équations. Par ailleurs, la solution R/V du problème (RRS) est encore unique (même idée de preuve que pour (RR) : si A divise à la fois $R - VB$ et $R_1 - V_1 B$, alors il divise également la combinaison linéaire $RV_1 - R_1 V = (R - VB)V_1 - (R_1 - V_1 B)V$).

Lien avec le pgcd étendu. Nous allons prouver en §1.1 que le problème (RRS) peut être résolu en utilisant l'algorithme d'Euclide étendu AEE en page 87. Nous allons en déduire ensuite une procédure de décision et calcul pour (RR).

L'intuition du lien entre le problème (RRS) et le calcul du pgcd étendu s'appuie sur la remarque simple suivante : la congruence $R \equiv VB \bmod A$ équivaut à l'existence d'un polynôme U tel que $R = UA + VB$; or, cette dernière égalité est une *relation de type Bézout*.

Pour préciser un peu cette remarque, traitons brièvement le problème (RRS) dans le cas particulier $k = 1$. Si A et B sont premiers entre eux, alors on prend $R = 1$ et la relation de Bézout $R = UA + VB$ avec $\deg(V) < \deg(A) = n$ fournit la réponse. Sinon, on prend $R = 0$, et comme le ppcm de A et B a un degré $< m+n$, le polynôme $V = \operatorname{ppcm}(A, B)/B$ est de degré au plus $n-1$ et vérifie $VB = 0 \bmod A$.

1.1. Calcul de la reconstruction rationnelle.

Rappelons que l'algorithme d'Euclide étendu (AEE, page 87) calcule une suite de restes successifs dont le degré décroît, ainsi qu'une suite de cofacteurs (U_i, V_i). Les éléments de la i-ième itération vérifient l'identité de Bézout $U_i A + V_i B = R_i$. L'écriture matricielle de l'itération

$$\begin{bmatrix} U_i & V_i \\ U_{i+1} & V_{i+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -Q_i \end{bmatrix} \times \begin{bmatrix} U_{i-1} & V_{i-1} \\ U_i & V_i \end{bmatrix}$$

permet de déduire aisément que $U_i V_{i+1} - V_i U_{i+1} = (-1)^i$ quel que soit i, et en particulier que U_i et V_i sont premiers entre eux. Cela entraîne également que $\operatorname{pgcd}(R_i, V_i) = \operatorname{pgcd}(A, V_i)$.

La clé du résultat principal de cette section (Théorème 1 ci-dessous) est le lemme suivant, qui montre que les cofacteurs V_i dans l'algorithme AEE ont des degrés modérés et bien maîtrisés.

LEMME 1. *Soient* $n_0 := \deg(A) > n_1 := \deg(B) > n_2 > \ldots > n_\ell$ *les degrés des restes* R_i *dans* AEE(A, B). *Alors :*

$$\deg(V_i) = n - n_{i-1} \qquad \textit{pour} \qquad 1 \leq i \leq \ell + 1. \tag{1}$$

DÉMONSTRATION. On procède par récurrence complète sur i. L'initialisation est claire, puisque $\deg(V_1) = 0$. Prouvons maintenant que si l'égalité (1) est vérifiée pour $1 \leq i \leq k$, alors elle l'est aussi pour $i = k+1$.

On part de l'observation que $\deg(V_{k-1}) = n - n_{k-2} < n - n_{k-1} = \deg(V_k)$, qui implique que le degré de $V_{k+1} = V_{k-1} - Q_k V_k$ est égal à $\deg(Q_k V_k) = \deg(Q_k) + \deg(V_k) = (n_{k-1} - n_k) + (n - n_{k-1}) = n - n_k$. □

REMARQUE. Dans une situation « générique », les quotients successifs Q_i ont tous degré 1, et donc $n_{i+1} = n_i - 1$ (la suite des restes est appelée *normale*), si bien que le lemme précédent implique l'égalité $\deg(V_i) = i - 1$.

Nous sommes en mesure de prouver que le problème de reconstruction rationnelle (RR) peut être résolu à l'aide de l'algorithme AEE.

THÉORÈME 1. *Soit $A \in \mathbb{K}[X]$ de degré $n > 0$ et soit $B \in \mathbb{K}[X]$ de degré $< n$. Soit $k \in \{1, 2, \ldots, n\}$ et soit (R_j, U_j, V_j) la j-ième ligne dans $\mathsf{AEE}(A, B)$, où j est choisi minimal tel que $\deg(R_j) < k$.*

1. *Il existe une solution $(R, V) \neq (0, 0)$ de (RRS), à savoir $(R, V) = (R_j, V_j)$. Si, de plus, $\operatorname{pgcd}(R_j, V_j) = 1$, alors (R, V) est aussi solution de (RR).*
2. *Si (RR) admet une solution et si $R/V \in \mathbb{K}(X)$ en est une forme irréductible, alors il existe une constante $\alpha \in \mathbb{K} \setminus \{0\}$ telle que $R = \alpha R_j$ et $V = \alpha V_j$.*

Le problème (RR) admet donc une solution si et seulement si $\operatorname{pgcd}(A, V_j) = 1$.

DÉMONSTRATION. Par construction, R_j vaut bien $U_j A + V_j B \equiv V_j B$ modulo A. Par ailleurs, le Lemme 1 montre que le degré de V_j vaut $n - \deg(R_{j-1})$ et est donc borné par $n - k$ (par la minimalité de j). Donc $(R, V) = (R_j, V_j)$ vérifie bien (RRS).

De plus, le $\operatorname{pgcd}(A, V_j)$ est égal au $\operatorname{pgcd}(R_j, V_j)$. Par conséquent, si ce dernier vaut 1, alors V_j est inversible modulo A et donc $(R, V) = (R_j, V_j)$ vérifie aussi (RR).

L'unicité (à une constante près) est plus délicate à prouver. Supposons que (R, V) est une solution de (RR). Alors R s'écrit $UA + VB$ pour un certain $U \in \mathbb{K}[X]$. Nous allons prouver que $(R, V) = (\alpha R_j, \alpha V_j)$ pour un certain $\alpha \in \mathbb{K}[X] \setminus \{0\}$.

Commençons par montrer l'égalité $U_j V = U V_j$. Supposons le contraire et considérons le système linéaire

$$\begin{pmatrix} U_j & V_j \\ U & V \end{pmatrix} \times \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} R_j \\ R \end{pmatrix}.$$

Le système étant supposé de Cramer, A s'écrit comme quotient de deux déterminants

$$A = \begin{vmatrix} R_j & V_j \\ R & V \end{vmatrix} \Big/ \begin{vmatrix} U_j & V_j \\ U & V \end{vmatrix}.$$

L'hypothèse $\deg(R_j) < k \leq \deg(R_{j-1})$ et le Lemme 1 entraînent que le degré du membre droit de l'égalité précédente est majoré par

$$\begin{aligned} \deg(R_j V - R V_j) &\leq \max\{k - 1 + \deg(V), \deg(R) + n - \deg(R_{j-1})\} \\ &\leq \max\{n - 1, (k - 1) + n - \deg(R_{j-1})\} = n - 1, \end{aligned}$$

ce qui contredit $\deg(A) = n$. Nous avons donc bien l'égalité $U_j V = U V_j$. Celle-ci implique que V_j divise $U_j V$, et puisque U_j et V_j sont premiers entre eux, il s'ensuit que V_j divise V. En écrivant $V = \alpha V_j$, avec $\alpha \in \mathbb{K}[X]$, on obtient $U V_j = V U_j = \alpha U_j V_j$, et donc $U = \alpha U_j$. Enfin, $R = UA + VB = \alpha(U_j A + V_j B) = \alpha R_j$.

Les polynômes R et V étant premiers entre eux, il s'ensuit que α est une constante de $\mathbb{K} \setminus \{0\}$, et que donc $\operatorname{pgcd}(R_j, V_j) = 1$. □

L'algorithme qui s'ensuit est donné en Figure 1 ; à noter la ressemblance avec l'algorithme AEE.

1.2. Approximants de Padé. On rappelle que si $n > 0$, $k \in \{1, 2, \ldots, n\}$ et si $B \in \mathbb{K}[X]$ est de degré $< n$, alors un *approximant de Padé pour B de type $(k - 1, n - k)$* est une fraction rationnelle $R/V \in \mathbb{K}[X]$ telle que

$$X \nmid V, \quad \deg(R) < k, \quad \deg(V) \leq n - k \quad \text{et} \quad \frac{R}{V} \equiv B \bmod X^n. \tag{2}$$

Le Théorème 1 (pour $A = X^n$) entraîne la conséquence suivante.

ARR(A, B, k)

Entrée : $A, B \in \mathbb{K}[X]$ avec $n = \deg(A) > \deg(B)$, et $k \in \{1, \dots, n\}$.

Sortie : Une solution (R, V) de (RR), ou 0 si une telle solution n'existe pas.

1. $R_0 := A$; $U_0 := 1$; $V_0 := 0$; $R_1 := B$; $U_1 := 0$; $V_1 := 1$; $i := 1$.
2. Tant que $\deg(R_i) \geq k$, faire :
   $(Q_i, R_{i+1}) := \mathsf{QuotRem}(R_{i-1}, R_i)$; # la division euclidienne
   $U_{i+1} := U_{i-1} - Q_i U_i$; $V_{i+1} := V_{i-1} - Q_i V_i$;
   $i := i + 1$
3. Si $\mathrm{pgcd}(A, V_i) = 1$, renvoyer (R_i, V_i) ; sinon renvoyer 0.

FIGURE 1. Algorithme de reconstruction rationnelle.

Berlekamp-Massey

Entrée : Une borne $N \in \mathbb{N}$ sur le degré du polynôme minimal de la suite $(a_n)_{n\geq 0}$ et les $2N$ premiers termes $a_0, \dots, a_{2N-1} \in \mathbb{K}$.

Sortie : Le polynôme générateur minimal de $(a_n)_{n\geq 0}$.

1. $A := a_0 + a_1 X + \cdots + a_{2N-1} X^{2N-1}$.
2. Calculer $R, V \in \mathbb{K}[X]$ la solution de (3) telle que $V(0) = 1$.
3. $d := \max\{1 + \deg(R), \deg(V)\}$. Renvoyer $\bar{V} := V(1/X)X^d$.

FIGURE 2. L'algorithme de Berlekamp-Massey.

COROLLAIRE 1. *Soit $B \in \mathbb{K}[X]$ de degré $< n$. Soit (R_j, U_j, V_j) la j-ième ligne dans* AEE(X^n, B)*, où j est choisi minimal tel que $\deg(R_j) < k$. Alors :*

1. *L'équation (2) admet une solution si et seulement si $\mathrm{pgcd}(R_j, V_j) = 1$.*
2. *Si $\mathrm{pgcd}(R_j, V_j) = 1$, alors R_j/V_j est l'unique approximant de Padé pour B de type $(k-1, n-k)$.*

L'algorithme qui s'en déduit est un cas particulier de l'algorithme ARR, en prenant $A = X^n$ et en remplaçant le test $\mathrm{pgcd}(A, V_i) = 1$ de l'étape (3) par $V_i(0) \neq 0$.

1.3. Algorithme de Berlekamp-Massey. L'algorithme décrit dans cette section permet de deviner des récurrences à coefficients constants d'ordre arbitraire à partir de termes consécutifs de la suite.

On se donne dans un corps $\mathbb{K}$ les $2N$ premiers éléments d'une suite récurrente linéaire $(a_n)_{n\geq 0}$ pour laquelle on sait qu'il existe un polynôme caractéristique de degré $d \leq N$, c'est-à-dire un polynôme

$$f(X) = f_d X^d + \cdots + f_0, \quad \text{tel que} \quad f_d a_{n+d} + \cdots + f_0 a_n = 0, \quad \text{pour tous} \quad n \geq 0.$$

Le problème est de calculer le polynôme minimal (*i.e.* le polynôme caractéristique de degré minimal) de $(a_n)_{n\geq 0}$. Le Lemme 2 (page 66) montre que ce calcul équivaut à la résolution du problème de Padé

$$\frac{R}{V} \equiv A \bmod X^{2N}, \quad X \not| \ V, \ \deg(R) < N, \ \deg(V) \leq N \ \text{ et } \ \mathrm{pgcd}(R, V) = 1. \tag{3}$$

En effet, il implique que $(R, V) = (N_0, \bar{P})$ est solution de (3). L'algorithme qui s'en déduit est donné en Figure 2.

1.4. Interpolation rationnelle de Cauchy. L'interpolation des fractions rationnelles, appelée aussi *interpolation de Cauchy*, est une généralisation naturelle de l'interpolation polynomiale (de Lagrange).

Soient $k \in \{1, \dots, n\}$, $v_0, \dots, v_{n-1} \in \mathbb{K}$ et soient $u_0, \dots, u_{n-1}$ des points distincts de $\mathbb{K}$. On cherche une fraction rationnelle $R/V \in \mathbb{K}(X)$ avec $R, V \in \mathbb{K}[X]$ satisfaisant aux contraintes

$$V(u_i) \neq 0, \quad \frac{R(u_i)}{V(u_i)} = v_i \quad \text{pour } 0 \leq i < n, \quad \deg(R) < k,\ \deg(V) \leq n-k. \tag{4}$$

Le problème (4) est un cas particulier de reconstruction rationnelle. En effet, si l'on note $B \in \mathbb{K}[X]$ l'unique polynôme de degré $< n$ tel que $B(u_i) = v_i$, alors (4) équivaut à la reconstruction rationnelle de B modulo $A := (X-u_0)\cdots(X-u_{n-1})$.

Le Théorème 1 implique la procédure de décision et de calcul suivante : on calcule le polynôme interpolant B et le polynôme $A = (X-u_0)\cdots(X-u_{n-1})$. On calcule les éléments (R_j, V_j) de la j-ième ligne de $\mathsf{AEE}(A, B)$ avec j minimal tel que $\deg(R_j) < k$. Si $\operatorname{pgcd}(A, V_j) = 1$, alors R_j/V_j est l'unique forme canonique de la fraction rationnelle cherchée. Sinon, le problème (4) n'a pas de solution.

Une application importante de l'interpolation rationnelle est la reconnaissance de récurrences linéaires d'ordre 1 à coefficients polynomiaux.

Corollaire 2 (Devinette de récurrences hypergéométriques). *Soit (a_n) une suite d'éléments non nuls de $\mathbb{K}$ vérifiant la récurrence $p_1(n)a_{n+1} + p_0(n)a_n = 0$, dont les coefficients (inconnus) $p_0(X)$ et $p_1(X)$ sont des polynômes de $\mathbb{K}[X]$, premiers entre eux. Si l'on connaît une borne d sur le degré de p_0 et de p_1 telle que $p_0(j) \neq 0$ pour $0 \leq j \leq 2d+1$, alors on peut déterminer p_0 et p_1 par un calcul d'interpolation rationnelle, à partir des $2d+1$ premiers termes de la suite (a_n).*

1.5. Algorithmes rapides. En combinant les résultats du Théorème 1 ci-dessus, du Théorème 3 en page 100, et du Chapitre 5 pour le calcul rapide des polynômes A et B définis en Section 1.4, on déduit le résultat de complexité suivant.

Théorème 2. *Soient $A \in \mathbb{K}[X]$ de degré $n > 0$, $B \in \mathbb{K}[X]$ de degré $< n$ et $k \in \{1, 2, \dots, n\}$. Soient $v_0, \dots, v_{n-1} \in \mathbb{K}$, et soient $u_0, \dots, u_{n-1}$ des points distincts de $\mathbb{K}$. Il est possible de calculer en $O(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$:*

1. *une solution du problème reconstruction rationnelle (RR) ;*
2. *un approximant de Padé pour B de type $(k-1, n-k)$;*
3. *une fraction rationnelle $F = R/V \in \mathbb{K}(X)$ telle que $\deg(V) \leq n-k$, $\deg(R) < k$, et $F(u_i) = v_i$ pour $0 \leq i < n$.*

Le cas entier. L'analogue pour les entiers de la reconstruction rationnelle est également calculable en complexité quasi-optimale. Si n et B sont deux entiers, $0 \leq B < n$, et étant donné $k \leq n$, il s'agit de calculer des entiers R et V tels que

$$\operatorname{pgcd}(n, V) = 1 \quad \text{et} \quad B = \frac{R}{V} \mod n, \quad |R| < k, \quad 0 \leq V \leq \frac{n}{k}. \tag{5}$$

Théorème 3. *À partir de k, n et B, on peut calculer R et V satisfaisant à (5) en $O(\mathsf{M}_{\mathbb{Z}}(\log n)\log\log n)$ opérations binaires.*

Le cas des récurrences. Une conséquence utile du Théorème 2 à la reconnaissance rapide des suites est contenue dans le corollaire suivant. Sa généralisation aux cas des suites P-récurrentes arbitraires sera traitée en Section 2.

Corollaire 3. *Soit (a_n) une suite d'éléments de $\mathbb{K}$ vérifiant*

— *soit une récurrence linéaire à coefficients constants d'ordre au plus N ;*

— *soit une récurrence linéaire d'ordre 1 à coefficients polynomiaux de degré au plus N.*

À partir des premiers $2N+1$ termes de la suite, on peut retrouver les coefficients de la récurrence en $O(\mathsf{M}(N)\log N)$ opérations dans $\mathbb{K}$.

2. Approximants de Padé-Hermite

DEFINITION 2. Soit $\mathbf{F} = {}^t(f_1, \ldots, f_n)$ un vecteur de $n \geq 1$ séries formelles de $\mathbb{K}[[X]]$, et soit $\mathbf{d} = (d_1, \ldots, d_n) \in \mathbb{N}^n$. Un vecteur non nul $\mathbf{P} = (P_1, \ldots, P_n)$ de polynômes de $\mathbb{K}[X]$ est appelé *approximant de Padé-Hermite de type* $\mathbf{d}$ *de* $\mathbf{F}$ si :

1. La valuation $\mathrm{val}(\mathbf{P} \cdot \mathbf{F})$ de la série $\mathbf{P} \cdot \mathbf{F} = \sum_{i=1}^{n} P_i f_i$ est au moins égale à $\sigma := \sum(d_i + 1) - 1$;
2. $\deg(P_i) \leq d_i$ pour tout $1 \leq i \leq n$.

L'entier σ est alors appelé *l'ordre de l'approximant.*

EXEMPLE 1. Dans la terminologie de la section précédente, si $R/V \in \mathbb{K}(X)$ est un approximant de Padé de type $(k, n-k)$ de $B \in \mathbb{K}[[X]]$, alors (R, V) est un approximant de Padé-Hermite pour $(-1, B)$, de type $(k, n-k)$.

EXERCICE 1. Soient A, B deux polynômes de $\mathbb{K}[X]$, avec $n = \deg(A) > \deg(B)$. Montrer que (R, V) est solution du problème de reconstruction rationnelle (RRS) défini en page 106 si et seulement s'il existe un polynôme $U \in \mathbb{K}[X]$ tel que (R, V, U) soit un approximant de Padé-Hermite de $(-1, B, A)$ de type $(k-1, n-k, n-1)$.

Le premier résultat de cette section montre l'existence des approximants de Padé-Hermite et fournit également un premier algorithme pour leur calcul.

THÉORÈME 4. *Tout vecteur de séries formelles $\mathbf{F} = (f_1, \ldots, f_n) \in \mathbb{K}[[X]]^n$ admet un approximant de Padé-Hermite de type $\mathbf{d} = (d_1, \ldots, d_n) \in \mathbb{N}^n$ donné.*

DÉMONSTRATION. On procède par coefficients indéterminés : en écrivant $P_i = \sum_{j=0}^{d_i} p_{i,j} X^j$, on obtient un système linéaire homogène à $\sigma = \sum_i (d_i + 1) - 1$ équations en les $\sigma + 1$ inconnues $p_{i,j}$. Puisqu'il a moins d'équations que d'inconnues, ce système admet forcément une solution non triviale. □

L'algorithme qui découle de la preuve du Théorème 4 repose sur la recherche d'un élément non trivial dans le noyau d'une matrice à coefficients dans $\mathbb{K}$, de taille $\sigma \times (\sigma+1)$. En utilisant les résultats du Chapitre 8, on aboutit donc à un algorithme de complexité $O(\mathsf{MM}(\sigma)) = O(\sigma^\theta)$ pour $2 \leq \theta \leq 3$.

Le but de la suite de ce chapitre est de présenter un algorithme plus efficace, dû à Derksen, de complexité seulement quadratique en σ. En anticipant un peu, cet algorithme revient à effectuer une sorte de pivot de Gauss sur une matrice à coefficients dans $\mathbb{K}[X]$, mais de taille bien plus petite que σ (seulement linéaire en $\max(\mathbf{d})$). Dans le cas particulier $n = 2$, l'algorithme de Derksen a essentiellement la même complexité que le calcul d'approximants de Padé via l'algorithme d'Euclide étendu vu en Section 1.

2.1. Algorithme de Derksen : idées et résultats préliminaires.

Pour simplifier la présentation, on se restreint dans la suite au cas où le type de l'approximant cherché est de la forme $\mathbf{d} = (d, \ldots, d) \in \mathbb{N}^n$, pour un certain $d \in \mathbb{N}$.

L'idée de l'algorithme de Derksen est de construire non pas un seul approximant de Padé-Hermite, mais toute une famille de tels approximants, et cela de manière incrémentale. Plus exactement, pour $s = 0, 1, \ldots$, il construit une base d'une forme spéciale, appelée « base minimale », du $\mathbb{K}[X]$-module

$$V_s := \{\mathbf{P} \in \mathbb{K}[X]^n \quad | \quad \mathrm{val}(\mathbf{P} \cdot \mathbf{F}) \geq s\}.$$

Comme nous le verrons plus loin, une telle base de V_σ pour $\sigma = nd+n-1$ contiendra alors nécessairement un approximant de Padé-Hermite de type $\mathbf{d} = (d, \ldots, d)$ de $\mathbf{F}$.

Observons que, grâce aux inclusions $X^s\mathbb{K}[X]^n \subseteq V_s \subseteq \mathbb{K}[X]^n$, le module V_s est libre de rang n. Précisons ce que l'on entend par « base minimale ». Pour ce faire, introduisons d'abord une notion de *degré* et de *type* d'un vecteur de polynômes.

DEFINITION 3 (degré et type d'un vecteur de polynômes). Pour un vecteur de polynômes $\mathbf{P} = (P_1, \ldots, P_n) \in \mathbb{K}[X]^n$, son *degré* $\deg(\mathbf{P})$ et son *type* $\mathrm{type}(\mathbf{P})$ sont donnés par :

$$\deg(\mathbf{P}) = \max\{\deg(P_1), \ldots, \deg(P_n)\} \quad \text{et} \quad \mathrm{type}(\mathbf{P}) = \max\{i \mid \deg(\mathbf{P}) = \deg(P_i)\}.$$

DEFINITION 4 (base minimale). Soit $V \subseteq \mathbb{K}[X]^n$ un sous-module libre de rang n. Une suite $\mathbf{Q}_1, \ldots, \mathbf{Q}_n$ est appelée « base minimale » de V si, pour tout i, le vecteur $\mathbf{Q}_i$ est non nul, de type i, et de degré minimal parmi les éléments de $V \setminus \{0\}$ de type i.

Le résultat suivant précise le lien entre une base minimale et l'approximation de Padé-Hermite.

LEMME 2. *Soit $d \geq 1$. Supposons que $\mathbf{Q}_1, \ldots, \mathbf{Q}_n$ est une base minimale de V_{nd+n-1}. Soit ℓ tel que $\deg(\mathbf{Q}_\ell)$ est minimal. Alors $\mathbf{Q}_\ell$ est un approximant de Padé-Hermite de type $(d, \ldots, d)$ pour $\mathbf{F}$.*

DÉMONSTRATION. Le Théorème 4 montre l'existence d'un vecteur non nul $\mathbf{P}$ de V_{nd+n-1} tel que $\deg(\mathbf{P}) \leq d$. Soit i le type de $\mathbf{P}$. La suite d'inégalités

$$\deg(\mathbf{Q}_\ell) \leq \deg(\mathbf{Q}_i) \leq \deg(\mathbf{P}) \leq d,$$

prouve que $\mathbf{Q}_\ell$ est un approximant de Padé-Hermite de type $(d, \ldots, d)$ de $\mathbf{F}$. □

Le résultat suivant montre qu'une base minimale est nécessairement une base du $\mathbb{K}[X]$-module V au sens usuel.

THÉORÈME 5. *Soit $V \subseteq \mathbb{K}[X]^n$ un sous-module libre de rang n. Toute base minimale de V est une base du $\mathbb{K}[X]$-module V.*

DÉMONSTRATION. Montrons d'abord qu'il s'agit d'un système de générateurs. Soit $W := \mathbb{K}[X]\mathbf{Q}_1 + \cdots + \mathbb{K}[X]\mathbf{Q}_n \subseteq V$. On suppose par l'absurde que $V \neq W$. Soit $\mathbf{P} \in V \setminus W$ un élément minimal dans $V \setminus W$ pour l'ordre $\mathbf{P} < \mathbf{Q}$ défini par

$$\begin{gathered} \deg(\mathbf{P}) < \deg(\mathbf{Q}), \quad \text{ou} \\ \deg(\mathbf{P}) = \deg(\mathbf{Q}) \quad \text{et} \quad \mathrm{type}(\mathbf{P}) < \mathrm{type}(\mathbf{Q}). \end{gathered} \tag{6}$$

Autrement dit, $\mathbf{P}$ est de type minimal parmi les éléments de degré minimal de $V \setminus W$.

Soit i le type de $\mathbf{P}$. Les relations $\mathrm{type}(\mathbf{P}) = \mathrm{type}(\mathbf{Q}_i)$ et $\deg(\mathbf{P}) \geq \deg(\mathbf{Q}_i)$ entraînent l'existence d'un monôme $q \in \mathbb{K}[X]$ de degré $\deg(q) = \deg(\mathbf{P}) - \deg(\mathbf{Q}_i)$ tel que $\mathrm{type}(\mathbf{P} - q\mathbf{Q}_i) < \mathrm{type}(\mathbf{P})$. Puisque $\deg(\mathbf{P} - q\mathbf{Q}_i) \leq \deg(\mathbf{P})$, on obtient que

$$\mathbf{P} - q\mathbf{Q}_i < \mathbf{P}.$$

Par la minimalité de $\mathbf{P}$, il s'ensuit que $\mathbf{P} - q\mathbf{Q}_i$ appartient à W, donc $\mathbf{P} \in W$, ce qui contredit le choix de $\mathbf{P}$.

Pour conclure la preuve, montrons que les $\mathbf{Q}_i$ forment une famille libre. Si $\sum_i a_i\mathbf{Q}_i = 0$ est une combinaison polynomiale nulle des $\mathbf{Q}_i$, on a que pour tout i, le vecteur $a_i\mathbf{Q}_i$ est de type i. L'assertion découle du lemme suivant. □

LEMME 3. *Si $\mathbf{P}, \mathbf{Q}$ ont types distincts, alors* $\mathrm{type}(\mathbf{P}+\mathbf{Q}) \in \{\mathrm{type}(\mathbf{P}), \mathrm{type}(\mathbf{Q})\}$.

DÉMONSTRATION. Supposons $j = \mathrm{type}(\mathbf{P}) > i = \mathrm{type}(\mathbf{Q})$. Si $\deg(\mathbf{P}) \geq \deg(\mathbf{Q})$, alors $\mathrm{type}(\mathbf{P} + \mathbf{Q}) = j$ et si $\deg(\mathbf{P}) < \deg(\mathbf{Q})$, alors $\mathrm{type}(\mathbf{P} + \mathbf{Q}) = i$. □

2.2. Algorithme de Derksen : fonctionnement. L'idée de l'algorithme est de construire de proche en proche une base minimale de V_s, partant de la base minimale des $\mathbf{Q}_k = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$ (avec 1 en position k) de V_0. D'après le Lemme 2, l'élément de degré minimal dans une base minimale de V_{nd+n-1} fournit un approximant de Padé-Hermite de type $(d, \ldots, d)$ de $\mathbf{F}$.

Le résultat suivant montre comment construire une base minimale de V_{s+1} à partir d'une base minimale de V_s, et il constituera le cœur de l'itération.

THÉORÈME 6. *Soit $\mathbf{Q}_1, \ldots, \mathbf{Q}_n$ une base minimale de*

$$V_s = \{\mathbf{P} \in \mathbb{K}[X]^n \mid \mathrm{val}(\mathbf{P} \cdot \mathbf{F}) \geq s\}.$$

1. *Si $\mathrm{val}(\mathbf{Q}_i \cdot \mathbf{F}) \geq s+1$ quel que soit i, alors V_{s+1} et V_s coïncident, et $\{\mathbf{Q}_1, \ldots, \mathbf{Q}_n\}$ est une base minimale de V_{s+1}.*
2. *Supposons que $1 \leq i \leq n$ est tel que les deux conditions suivantes soient réunies :*
   - *$\mathrm{val}(\mathbf{Q}_i \cdot \mathbf{F}) = s$;*
   - *si $\mathrm{val}(\mathbf{Q}_\ell \cdot \mathbf{F}) = s$ pour un $\ell \neq i$, alors $\mathbf{Q}_i < \mathbf{Q}_\ell$, où $<$ est l'ordre (6).*

   Alors :
   (a) *pour $\ell \neq i$, il existe un scalaire $\lambda_\ell \in \mathbb{K}$ tel que $\tilde{\mathbf{Q}}_\ell := \mathbf{Q}_\ell - \lambda_\ell \mathbf{Q}_i$ vérifie $\mathrm{val}(\tilde{\mathbf{Q}}_\ell \cdot \mathbf{F}) > s$;*
   (b) *en posant $\tilde{\mathbf{Q}}_i = X\mathbf{Q}_i$, la suite $\tilde{\mathbf{Q}}_1, \ldots, \tilde{\mathbf{Q}}_n$ forme une base minimale de V_{s+1}.*

DÉMONSTRATION. (1) L'inclusion $V_{s+1} \subseteq V_s$ est évidente, et inversement, le Théorème 5 montre que tout $\mathbf{P} \in V_s$ s'écrit comme combinaison linéaire $\sum_i a_i \mathbf{Q}_i$. Ainsi $\mathbf{P} \cdot \mathbf{F} = \sum_i a_i (\mathbf{Q}_i \cdot \mathbf{F})$ est de valuation au moins $s+1$, donc $\mathbf{P} \in V_{s+1}$.

(2a) Si $\mathrm{val}(\mathbf{Q}_\ell \cdot \mathbf{F}) > s$, on pose $\lambda_\ell = 0$; si $\mathrm{val}(\mathbf{Q}_\ell \cdot \mathbf{F}) = s$, alors $\mathbf{Q}_\ell \cdot \mathbf{F} = c_\ell X^s + \cdots$ et $\mathbf{Q}_i \cdot \mathbf{F} = c_i X^s + \cdots$, avec $c_i \neq 0$, et alors $\lambda_\ell := c_\ell / c_i$ convient.

Pour (2b), commençons par montrer que $\tilde{\mathbf{Q}}_1, \ldots, \tilde{\mathbf{Q}}_{i-1}, \mathbf{Q}_i, \tilde{\mathbf{Q}}_{i+1}, \ldots, \tilde{\mathbf{Q}}_n$ reste une base minimale de V_s. Il suffit pour cela de montrer que pour $\ell \neq i$, le vecteur $\tilde{\mathbf{Q}}_\ell$ a même type et même degré que $\mathbf{Q}_\ell$. Si $\mathrm{val}(\mathbf{Q}_\ell \cdot \mathbf{F}) > s$, cela est évident, car $\lambda_\ell = 0$ et donc $\tilde{\mathbf{Q}}_\ell = \mathbf{Q}_\ell$. Sinon, le choix de i assure que $\mathbf{Q}_\ell > \mathbf{Q}_i$, et donc $\mathbf{Q}_\ell$ et $\mathbf{Q}_\ell - \lambda_\ell \mathbf{Q}_i$ ont même degré et même type.

Montrons maintenant que $(\tilde{\mathbf{Q}}_1, \ldots, \tilde{\mathbf{Q}}_n)$ est une base minimale de V_{s+1}. Comme la multiplication par un polynôme ne change pas le type, celui de $\tilde{\mathbf{Q}}_i = X\mathbf{Q}_i$ est bien i. Il suffit donc de montrer que si $\mathbf{P} \in V_{s+1}$ est de type $\ell \in \{1, 2, \ldots, n\}$, alors $\deg(\mathbf{P}) \geq \deg(\mathbf{Q}_\ell)$. Si $\ell \neq i$, ceci est une évidence : comme $\mathbf{P}$ appartient à $V_{s+1} \subseteq V_s$, et comme la suite $\tilde{\mathbf{Q}}_1, \ldots, \tilde{\mathbf{Q}}_{i-1}, \mathbf{Q}_i, \tilde{\mathbf{Q}}_{i+1}, \ldots, \tilde{\mathbf{Q}}_n$ forme une base minimale de V_s, le degré de $\mathbf{P}$ est nécessairement au moins égal à $\deg(\tilde{\mathbf{Q}}_\ell) = \deg(\mathbf{Q}_\ell)$.

Dans la suite de la preuve, on peut donc supposer que $\mathbf{P} \in V_{s+1}$ est de type i, le but étant de montrer que $\deg(\mathbf{P}) \geq \deg(X\mathbf{Q}_i)$. Le Théorème 5 montre que $\mathbf{P}$ s'écrit $\mathbf{P} = \sum_{j \neq i} a_j \tilde{\mathbf{Q}}_j + a_i \mathbf{Q}_i$. Puisque $\mathrm{type}(\mathbf{P}) = i$, le Lemme 3 entraîne $a_i \neq 0$. De plus, le degré de $\mathbf{P}$ est égal à celui de $a_i \mathbf{Q}_i$, d'après le Lemme 4 ci-dessous.

Comme $\mathrm{val}(\mathbf{P} \cdot \mathbf{F}) > s$ et comme pour $k \neq i$, $\mathrm{val}(\mathbf{Q}_k \cdot \mathbf{F}) > s$, on a nécessairement que $\mathrm{val}(a_i) > 0$. En particulier $\deg(a_i) > 0$ et donc $\deg(\mathbf{P}) \geq 1 + \deg(\mathbf{Q}_i)$. □

LEMME 4. *Si $\mathrm{type}(\mathbf{P}) = \mathrm{type}(\mathbf{Q}) = i$ et $\mathrm{type}(\mathbf{P}+\mathbf{Q}) < i$, alors $\deg(\mathbf{P}) = \deg(\mathbf{Q})$.*

DÉMONSTRATION. Soient $\mathbf{P} = (P_1, \ldots, P_n)$ et $\mathbf{Q} = (Q_1, \ldots, Q_n)$. L'hypothèse entraîne les égalités $\deg(P_i) = \deg(\mathbf{P})$ et $\deg(Q_i) = \deg(\mathbf{Q})$. Cela implique que P_i et Q_i ont le même degré ; sinon, le type de $\mathbf{P} + \mathbf{Q}$ serait égal à i. □

Derksen

Entrée : $\mathbf{F} = (f_1, \ldots, f_n) \in \mathbb{K}[[X]]^n$ et $d \geq 1$.

Sortie : Un approximant de Padé-Hermite de $\mathbf{F}$, de type $(d, \ldots, d)$.

pour k de 1 à n définir
  $\mathbf{Q}_k := (0, 0, \ldots, 0, 1, 0, \ldots, 0)$, avec 1 en position k.
pour j de 0 à $nd + n - 2$ faire
  $i := 0$
  pour k de 1 à n faire
    $c_k := \text{coeff}(\mathbf{Q}_k \cdot \mathbf{F}, j)$
    si $c_k \neq 0$ et ($\mathbf{Q}_k < \mathbf{Q}_i$ ou $i = 0$), alors $i := k$
  si $i \neq 0$ alors
    $\mathbf{Q}_i := c_i^{-1}\mathbf{Q}_i$
    pour k de 1 à n faire
      si $k \neq i$ alors
        $\mathbf{Q}_k := \mathbf{Q}_k - c_k\mathbf{Q}_i$
    $\mathbf{Q}_i := X\mathbf{Q}_i$
$p := 1$
pour k de 2 à n faire
  si $\deg(\mathbf{Q}_k) < \deg(\mathbf{Q}_p)$, alors $p := k$
Renvoyer $\mathbf{Q}_p$.

FIGURE 3. L'algorithme de Derksen pour les approximants de Padé-Hermite.

L'algorithme qui se déduit de la conjonction du Lemme 2 et du Théorème 6 est donné en Figure 3. Il est de complexité seulement quadratique en σ. En résumé, nous avons le résultat suivant.

THÉORÈME 7. *Soit $\mathbf{F} = (f_1, \ldots, f_n) \in \mathbb{K}[[X]]^n$. Il est possible de calculer un approximant de Padé-Hermite de type $(d, \ldots, d)$ de $\mathbf{F}$ en $O(n\sigma^2) = O(n^3d^2)$ opérations dans $\mathbb{K}$.*

EXERCICE 2. Écrire les détails de la preuve du Théorème 7.

REMARQUE. Dans le cas « générique » (dit *normal*), la sortie $\mathbf{Q} = \mathbf{Q}_1, \ldots, \mathbf{Q}_n$ de l'algorithme de Derksen est de degré

$$\begin{bmatrix} d+1 & d & \cdots & d & d \\ d+1 & d+1 & \cdots & d & d \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ d+1 & d+1 & \cdots & d+1 & d \\ d & d & \cdots & d & d \end{bmatrix}.$$

C'est la dernière ligne $\mathbf{Q}_n$ qui fournit l'approximant désiré.

2.3. Approximants de Padé-Hermite de type arbitraire. Pour calculer un approximant de Padé-Hermite de type $\mathbf{d} = (d_1, \ldots, d_n)$, il suffit de remplacer, dans l'algorithme donné en Figure 3, deg par $\deg_{\mathbf{d}}$ et type par $\text{type}_{\mathbf{d}}$, où :

$$\deg_{\mathbf{d}}(\mathbf{P}) = \max_{1 \leq i \leq n} \{\deg(P_i) - d_i\}, \quad \text{type}_{\mathbf{d}}(\mathbf{P}) = \max\{i \mid \deg(P_i) - d_i = \deg_{\mathbf{d}}(\mathbf{P})\}.$$

2.4. Applications. Si une série est connue comme rationnelle de degré au plus d, l'approximation de Padé de type (d, d) suffit pour reconstruire la fraction rationnelle. Une question naturelle est comment se généralise cette observation dans le cadre de l'approximation de Padé-Hermite. Les réponses sont partielles, mais entièrement satisfaisantes dans la pratique.

Recherche de relations à coefficients polynomiaux entre séries formelles. Supposons qu'il existe une combinaison linéaire $\sum_{i=1}^{n} P_i f_i = 0$, à coefficients des polynômes $P_i(X) \in \mathbb{K}[X]$ de degrés au plus d. Par ailleurs, supposons calculé un approximant de Padé-Hermite $\mathbf{Q} := (Q_1, \ldots, Q_n)$ de $\mathbf{F} = (f_1, \ldots, f_n)$ de type $\mathbf{d} = (d, \ldots, d)$, via l'algorithme de Derksen. La question est donc : quel lien y a-t-il entre $\mathbf{P} = (P_1, \ldots, P_n)$ et $\mathbf{Q}$?

D'abord, dans le cas *générique*, la réponse est très simple : $\mathbf{P}$ et $\mathbf{Q}$ sont identiques, à un coefficient scalaire près. En effet, $\mathbf{Q} = \mathbf{Q}_n$ et $\mathbf{Q}_1, \ldots, \mathbf{Q}_n$ est une base de V_{nd+n-1} dont les degrés sont décrits en page 113. En particulier, $\mathbf{P}$ doit être une combinaison linéaire des $\mathbf{Q}_i$. Des considérations sur les degrés, utilisant la forme bien particulière des degrés des $\mathbf{Q}_i$, mènent à la conclusion désirée.

En effet, si $(c_1(X), \ldots, c_n(X)) \cdot {}^t(\mathbf{Q}_1, \ldots, \mathbf{Q}_n) = \mathbf{P}$, alors $c_1 Q_{11} + \cdots + c_n Q_{n1} = P_1$, et comme $\deg(Q_{11}) = d + 1$ et $\deg(Q_{j1}) = d$ pour $j > 1$ et $\deg(P_1) \leq d$, on obtient que $c_1 = 0$. De même, $c_2 = \cdots = c_{n-1} = 0$ et c_n doit être une constante $c \in \mathbb{K}$ telle que $\mathbf{P} = c\mathbf{Q}$.

Dans le cas général, un argument de noethérianité permet de prouver que pour $D \gg 0$, V_{nD+n-1} contient la relation $\mathbf{P}$, qui sera trouvée par l'algorithme de Derksen. Seulement, on ne dispose pas de borne *a priori* en fonction de d, sur le D minimal avec cette propriété. En effet, si on note

$$W_j := \{\mathbf{Q} \in \mathbb{K}[X]^n \mid \operatorname{val}(\mathbf{Q} \cdot \mathbf{F}) \geq j \quad \text{et} \quad \deg(\mathbf{Q}) \leq \deg(\mathbf{P})\},$$

et $W_\infty = \cap_{j \geq 0} W_j$, alors W_∞ contient toutes les relations de F en degré d, et en particulier $\mathbf{P}$. Puisque $W_0 \supseteq W_1 \supseteq W_2 \supseteq \cdots$ est une suite décroissante d'espaces vectoriels de dimension finie, elle est stationnaire, donc il existe un N tel que $W_\infty = W_N = W_{N+1} = \cdots$. Le cas *normal* correspond à la situation où $\dim(W_{k+1}) = \dim(W_k) - 1$ pour chaque k (noter la ressemblance avec la normalité de la suite des restes dans l'algorithme d'Euclide).

Reconstruction d'équations algébriques et différentielles. Deux cas particuliers importants, pour lesquels on peut être encore plus précis, sont les approximants algébriques et différentiels.

Soit $A \in \mathbb{K}[[X]]$ une série formelle algébrique. Le problème d'approximation algébrique consiste à retrouver, à partir des premiers termes de f un polynôme $P(X, Y) \in \mathbb{K}[X, Y]$ tel que $P(X, A(X)) = 0$. Si un tel P, de degré d en X et n en Y, existe, alors les coefficients des puissances de Y formeront un approximant de Padé-Hermite de type $(d, \ldots, d)$ du vecteur de séries $(1, A, \ldots, A^n)$. La difficulté vient de ce que un calcul d'approximation de Padé-Hermite ne trouve, *a priori*, qu'un polynôme $Q \in \mathbb{K}[X, Y]$ tel que $Q(X, A(X)) = 0 \bmod X^\sigma$, où $\sigma = (n+1)d - 1$. On dit alors qu'on a *deviné* un polynôme annulateur Q de A. Pour les approximants différentiels, la problématique est la même, la seule différence étant qu'on calcule un approximant de Padé-Hermite du vecteur des dérivées successives $(A, A', \ldots, A^{(n)})$.

Certification d'identités algébriques. Pour un approximant algébrique, il se pose la question de la *certification a posteriori*. Pour un polynôme annulateur deviné Q, cela veut dire qu'on souhaiterait déduire que non seulement $Q(X, A(X))$ est nul modulo X^σ, mais aussi que $Q(X, A(X)) = 0$.

Le résultat suivant apporte une réponse partielle à la question de la certification. Son avantage est qu'il ne dépend pas de l'algorithme utilisé pour produire l'approximant de Padé-Hermite.

THÉORÈME 8. *Supposons que $A \in \mathbb{K}[[X]]$ est racine d'un polynôme irréductible de $\mathbb{K}[X,Y]$ de degré au plus d en X et au plus n en Y. Soit $\mathbf{Q} = (Q_0, Q_1, \ldots, Q_n)$ un approximant de Padé-Hermite de type $(d, \ldots, d)$ de $\mathbf{F} = (1, A, \ldots, A^n)$.*

Si $\mathrm{val}(\mathbf{Q} \cdot \mathbf{F}) \geq 2dn$, alors $\mathbf{Q} \cdot \mathbf{F} = 0$, c'est-à-dire que A est racine du polynôme $Q = \sum_{i=0}^{n} Q_i Y^i$.

DÉMONSTRATION. Soit $P \in \mathbb{K}[X,Y]$ un polynôme irréductible de degré au plus d en X et au plus n en Y tel que $P(X,A) = 0$. Le polynôme $\mathrm{Res}_Y(P,Q) \in \mathbb{K}[X]$ est de degré borné par $\deg_X(P)\deg_Y(Q) + \deg_Y(P)\deg_X(Q) \leq 2dn$. Comme il est de la forme $uP + vQ$ pour u et v deux polynômes, avec $\deg_Y(v) < \deg_Y(P)$, et qu'il ne change pas lorsqu'on remplace la variable Y par la série A, c'est un $O(X^{2dn})$. Par conséquent $uP + vQ = 0$, donc P divise vQ et, étant irréductible, il divise v ou Q. Or $\deg_Y(v) < \deg_Y(P)$, donc finalement $Q \mid P$ et $Q(X,A)$ est la série nulle. □

Reconstruction de récurrences. Soit $(a_n)_n \in \mathbb{K}^{\mathbb{N}}$ une suite vérifiant une récurrence linéaire à coefficients polynomiaux. Comment, à partir des premiers termes de la suite, retrouver les coefficients de cette récurrence ?

Une idée consiste à utiliser la dualité montrée au Théorème 1 en page 200, et à chercher une équation différentielle, à coefficients polynomiaux, portant sur la série génératrice de la suite. Cette équation peut être devinée comme expliqué précédemment, grâce à une approximation de Padé-Hermite. Il suffit ensuite de passer de l'équation différentielle à l'expression de la récurrence, ce qui n'est pas trivial mais purement formel et a été décrit au Chapitre 14.

2.5. Calcul quasi-optimal. Il se trouve que pour le calcul d'approximants de Padé-Hermite, on dispose d'un algorithme quasi-optimal, dont l'efficacité est utilisée en calcul formel comme la base de nombreux autres algorithmes. Cet algorithme peut être vu comme une alternative à l'approche par *matrices structurées*, de même complexité, décrite au Chapitre 10.

THÉORÈME 9. *Soient $\mathbf{F} = (f_1, \ldots, f_n) \in \mathbb{K}[[X]]^n$, $\mathbf{d} = (d_1, \ldots, d_n) \in \mathbb{N}^n$ et $\sigma = d_1 + \cdots + d_n + n - 1$. Il est possible de calculer un approximant de Padé-Hermite de type $\mathbf{d}$ de $\mathbf{F}$ en $O(\mathsf{MM}(n,\sigma)\log\sigma) = O(n^\theta \mathsf{M}(\sigma)\log(\sigma))$ opérations dans $\mathbb{K}$.*

ESQUISSE DE DÉMONSTRATION. Comme pour l'algorithme quadratique de Derksen, une première idée est de calculer non pas un vecteur, mais toute une matrice d'approximants. Une seconde idée est d'utiliser un « diviser pour régner » tout en exploitant la multiplication rapide de matrices polynomiales. Voici l'esquisse de l'algorithme dans le cas où $\mathbf{d} = (d, \ldots, d)$:

1. Si $N = 1$, renvoyer une base de l'orthogonal à $\mathbf{F}(0)$.
2. Sinon :
   (a) calculer récursivement une matrice polynomiale d'approximants $\mathbf{P}_1$ telle que $\mathbf{P}_1 \cdot \mathbf{F} = O(X^{\sigma/2})$ et $\deg(\mathbf{P}_1) \sim d/2$;
   (b) calculer le reste $\mathbf{R}$ tel que $\mathbf{P}_1 \cdot \mathbf{F} = X^{\sigma/2} \cdot \big(\mathbf{R} + O(X^{\sigma/2})\big)$;
   (c) calculer récursivement une matrice polynomiale $\mathbf{P}_2$ d'approximants du reste : $\mathbf{P}_2 \cdot \mathbf{R} = O(X^{\sigma/2})$, avec $\deg(\mathbf{P}_2) \sim d/2$;
   (d) renvoyer $\mathbf{P} := \mathbf{P}_2 \cdot \mathbf{P}_1$.

Le choix précis des degrés des approximants $\mathbf{P}_1$ et $\mathbf{P}_2$ est une question délicate et ne sera pas détaillé. En admettant ces choix, la correction de l'algorithme et son analyse de complexité se déduisent aisément. □

Exercices

EXERCICE 3 (Interpolation creuse). Soit $\mathbb{K}$ un corps effectif de caractéristique zéro. Soit F une fonction d'arité n sur $\mathbb{K}$ qui calcule les valeurs d'un polynôme P de $\mathbb{K}[X_1, \ldots, X_n]$ composé d'au plus t termes non nuls. Le calcul de tous les termes non nuls de P en évaluant seulement F sur plusieurs points bien choisis est appelé *interpolation creuse*.

Soient $p_1, \ldots, p_n$ des nombres premiers distincts (en pratique on choisit les plus petits).

1. Soient $X_1^{e_1} \cdots X_n^{e_n}$ et $X_1^{f_1} \cdots X_n^{f_n}$ deux monômes distincts. Expliquer brièvement pourquoi leurs valeurs en $(p_1, \ldots, p_n)$ sont distinctes.
2. Supposons que P contienne exactement $\tau \leq t$ termes non nuls, et qu'il s'écrive sous la forme $\sum_{i=1}^{\tau} P_i X_1^{e_{i,1}} \cdots X_n^{e_{i,n}}$. Montrer l'identité suivante :
$$\sum_{i \geq 0} F(p_1^i, \ldots, p_n^i) t^i = \sum_{i=1}^{\tau} \frac{P_i}{1 - p_1^{e_{i,1}} \cdots p_n^{e_{i,n}} t}.$$
3. Déduire de la formule précédente un algorithme pour obtenir les coefficients du polynôme $D(t) = \prod_{i=1}^{\tau} (1 - p_1^{e_{i,1}} \cdots p_n^{e_{i,n}} t)$ de $\mathbb{Q}[t]$. Estimer le coût de l'algorithme en fonction du nombre d'appels à la fonction F et du nombre d'opérations arithmétiques dans $\mathbb{K}$.
4. En admettant que l'on dispose d'un algorithme pour calculer toutes les racines de D dans $\mathbb{Q}$, expliquer comment obtenir toutes les valeurs $e_{i,j}$, $1 \leq i \leq \tau$, $1 \leq j \leq n$.
5. En utilisant les algorithmes rapides du chapitre 5, et les racines de D, proposer une façon efficace de calculer tous les coefficients P_i de P.

Notes

La présentation de la Section 1 est fortement inspirée par l'article [26], celle de la Section 2 par l'article [13]. Les approximants de Padé sont des objets très classiques en analyse numérique ; on pourra par exemple consulter le livre [1] qui leur est entièrement dédié.

L'algorithme de Berlekamp-Massey a été initialement conçu pour décoder des codes BCH, puis étendu par Zierler [39] et Massey [25], qui ont mis en évidence le lien avec les suites récurrentes linéaires à coefficients constants. Les liens forts entre l'algorithme d'Euclide, l'algorithme de Berlekamp-Massey, l'approximation de Padé et le calcul de fractions continues ont été étudiés dans [10, 15, 27, 38]. L'utilisation de ces méthodes pour l'interpolation creuse est à la base de l'algorithme dû à Ben-Or et Tiwari [5] adapté dans l'exercice 3.

Les premiers algorithmes rapides pour la reconstruction rationnelle sont dus à [18]. Ces algorithmes ont été améliorés dans [6, 9, 17]. La problématique de la reconstruction rationnelle pour les entiers est classique en théorie des nombres [19] ; en calcul formel, elle a été très étudiée en lien avec les algorithmes modulaires, par exemple pour la résolution de systèmes linéaires par remontées p-adiques [14]. Des algorithmes rapides ont été récemment proposés dans [30, 37].

Les approximants de Padé et de Padé-Hermite ont été introduits et étudiés par Hermite et Padé dans [20, 22, 28, 29]. Une exposition plus moderne se trouve dans [24]. Le problème d'approximation de Padé-Hermite a été utilisé par Hermite en 1873 dans sa preuve [21] de la transcendance de e, qui utilise le choix très particulier $\mathbf{F} = (1, e^X, e^{2X}, \ldots)$. Deux autres cas particuliers importants sont : les « approximants algébriques » [33, 34] avec $\mathbf{F} = (1, f, f^2, \ldots)$ et les « approximants différentiels » [23] avec $\mathbf{F} = (f, f', f'', \ldots)$, où f est une série de $\mathbb{K}[[X]]$ donnée.

En calcul formel, les approximants de Padé-Hermite ont été introduits par Della Dora et Dicrescenzo [11, 12]. Un premier algorithme de complexité quadratique a été donné par Sergeyev [32]. Cet article est resté méconnu, et d'autres algorithmes quadratiques ont été proposés dans [7, 8, 31, 36]. L'algorithme de complexité quasi-optimale esquissé en Section 2.5 a été proposé dans [2].

Il existe diverses généralisations utiles du problème d'approximation de Padé-Hermite, par exemple les « approximants de Padé-Hermite simultanés et matriciels », ou encore des approximants modulo un polynôme arbitraire de degré $\sigma = \sum_i (d_i + 1) - 1$ au lieu de X^σ. Des algorithmes rapides existent pour toutes ces généralisations [2, 3, 4, 16, 35].

Bibliographie

[1] Baker Jr., George A. et Peter Graves-Morris (1996). *Padé approximants*. 2e éd. Vol. 59. Encyclopedia of Mathematics and its Applications. Cambridge University Press, xiv+746 pages.

[2] Beckermann, B. et G. Labahn (1994). « A uniform approach for the fast computation of matrix-type Padé approximants ». In : *SIAM Journal on Matrix Analysis and Applications*, vol. 15, n°3, p. 804–823.

[3] Beckermann, Bernhard (1992). « A reliable method for computing M-Padé approximants on arbitrary staircases ». In : *Journal of Computational and Applied Mathematics*, vol. 40, n°1, p. 19–42.

[4] Beckermann, Bernhard et George Labahn (2000). « Fraction-free computation of matrix rational interpolants and matrix GCDs ». In : *SIAM Journal on Matrix Analysis and Applications*, vol. 22, n°1, p. 114–144.

[5] Ben-Or, M. et P. Tiwari (1988). « A deterministic algorithm for sparse multivariate polynomial interpolation ». In : *STOC'88 : ACM Symposium on Theory of Computing*. ACM Press, p. 301–309.

[6] Brent, Richard P., Fred G. Gustavson et David Y. Y. Yun (1980). « Fast solution of Toeplitz systems of equations and computation of Padé approximants ». In : *Journal of Algorithms*, vol. 1, n°3, p. 259–295.

[7] Cabay, S. et G. Labahn (1989). « A fast, reliable algorithm for calculating Padé-Hermite forms ». In : *ISSAC'89 : International Symposium on Symbolic and Algebraic Computation*. Portland, Oregon, United States : ACM Press, p. 95–100. doi : 10.1145/74540.74553.

[8] Cabay, S., G. Labahn et B. Beckermann (1992). « On the theory and computation of nonperfect Padé-Hermite approximants ». In : *Journal of Computational and Applied Mathematics*, vol. 39, n°3, p. 295–313.

[9] Cabay, Stanley et Dong Koo Choi (1986). « Algebraic computations of scaled Padé fractions ». In : *SIAM Journal on Computing*, vol. 15, n°1, p. 243–270.

[10] Cheng, Unjeng (1984). « On the continued fraction and Berlekamp's algorithm ». In : *IEEE Transactions on Information Theory*, vol. 30, n°3, p. 541–544.

[11] Della Dora, J. et C. Dicrescenzo (1984a). « Approximants de Padé-Hermite. I. Théorie ». In : *Numerische Mathematik*, vol. 43, n°1, p. 23–39.

[12] — (1984b). « Approximants de Padé-Hermite. II. Programmation ». In : *Numerische Mathematik*, vol. 43, n°1, p. 41–57.

[13] Derksen, Harm (1994). *An algorithm to compute generalized Padé-Hermite forms*. Report n°9403. Dept. of Math., Catholic University Nijmegen.

[14] Dixon, John D. (1982). « Exact solution of linear equations using p-adic expansions ». In : *Numerische Mathematik*, vol. 40, n°1, p. 137–141.

[15] Dornstetter, Jean-Louis (1987). « On the equivalence between Berlekamp's and Euclid's algorithms ». In : *IEEE Transactions on Information Theory*, vol. 33, n°3, p. 428–431.

[16] Giorgi, Pascal, Claude-Pierre Jeannerod et Gilles Villard (2003). « On the complexity of polynomial matrix computations ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par J. R. Sendra. New York : ACM Press, p. 135–142.

[17] Gragg, William B., Fred G. Gustavson, Daniel D. Warner et David Y. Y. Yun (1982). « On fast computation of superdiagonal Padé fractions ». In : *Mathematical Programming Study*, n°18. Algorithms and theory in filtering and control (Lexington, Ky., 1980), p. 39–42.

[18] Gustavson, Fred G. et David Y. Y. Yun (1979). « Fast algorithms for rational Hermite approximation and solution of Toeplitz systems ». In : *IEEE Transactions on Circuits and Systems*, vol. 26, n°9, p. 750–755.

[19] Hardy, G. H. et E. M. Wright (2008). *An introduction to the theory of numbers*. 6e éd. Oxford University Press, xxii+621 pages.

[20] Hermite, C. (1873a). « Extrait d'une lettre de Monsieur Ch. Hermite à Monsieur Paul Gordan ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 78, p. 303–311.

[21] — (1873b). « Sur la fonction exponentielle ». In : *Comptes-rendus de l'Académie des sciences de Paris*, vol. 77, p. 18–24.

[22] — (1893). « Sur la généralisation des fractions continues algébriques ». In : *Annali di Matematica pura ed applicata*, vol. 21, n°2, p. 289–308.

[23] Khan, M. A. H. (2002). « High-order differential approximants ». In : *Journal of Computational and Applied Mathematics*, vol. 149, n°2, p. 457–468.

[24] Mahler, K. (1968). « Perfect systems ». In : *Compositio Mathematica*, vol. 19, 95–166 (1968).

[25] Massey, James L. (1969). « Shift-register synthesis and BCH decoding ». In : *IEEE Transactions on Information Theory*, vol. IT-15, p. 122–127.

[26] McEliece, Robert J. et James B. Shearer (1978). « A property of Euclid's algorithm and an application to Padé approximation ». In : *SIAM Journal on Applied Mathematics*, vol. 34, n°4, p. 611–615.

[27] Mills, W. H. (1975). « Continued fractions and linear recurrences ». In : *Mathematics of Computation*, vol. 29, p. 173–180.

[28] Padé, H. (1892). « Sur la Représentation Approchée d'une Fonction par des Fractions Rationelles ». In : *Annales scientifiques de l'É.N.S.* vol. 9, p. 3–93.

[29] — (1894). « Sur la généralisation des fractions continues algébriques ». In : *Journal de mathématiques pures et appliquées*, vol. 10, n°4, p. 291–330.

[30] Pan, Victor Y. et Xinmao Wang (2004). « On rational number reconstruction and approximation ». In : *SIAM Journal on Computing*, vol. 33, n°2, p. 502–503.

[31] Paszkowski, Stefan (1987). « Recurrence relations in Padé-Hermite approximation ». In : *Journal of Computational and Applied Mathematics*, vol. 19, n°1, p. 99–107.

[32] Sergeyev, A. V. (1986). « A recursive algorithm for Padé-Hermite approximations ». In : *USSR Comput. Maths Math. Phys*, vol. 26, n°2, p. 17–22.

[33] Shafer, R. E. (1974). « On quadratic approximation ». In : *SIAM Journal on Numerical Analysis*, vol. 11, p. 447–460.

[34] Tourigny, Y. et Ph. G. Drazin (2000). « The asymptotic behaviour of algebraic approximants ». In : *The Royal Society of London. Proceedings. Series A. Mathematical, Physical and Engineering Sciences*, vol. 456, n°1997, p. 1117–1137.

[35] Van Barel, M. et A. Bultheel (1992). « A general module-theoretic framework for vector M-Padé and matrix rational interpolation ». In : *Numerical Algorithms*, vol. 3, n°1-4, p. 451–461.

[36] Van Barel, Marc et Adhemar Bultheel (1991). « The computation of nonperfect Padé-Hermite approximants ». In : *Numerical Algorithms*, vol. 1, n°3, p. 285–304.

[37] Wang, Xinmao et Victor Y. Pan (2003). « Acceleration of Euclidean algorithm and rational number reconstruction ». In : *SIAM Journal on Computing*, vol. 32, n°2, p. 548–556.

[38] Welch, L. R. et R. A. Scholtz (1979). « Continued fractions and Berlekamp's algorithm ». In : *IEEE Transactions on Information Theory*, vol. 25, n°1, p. 19–27.

[39] Zierler, N. (1968). « Linear recurring sequences and error-correcting codes ». In : *Error Correcting Codes (Proc. Sympos. Math. Res. Center, Madison, Wis., 1968)*. Wiley, p. 47–59.

Deuxième partie

Matrices

CHAPITRE 8

Algèbre linéaire dense : de Gauss à Strassen

Résumé

L'algèbre linéaire est au cœur de nombreux problèmes algorithmiques. La multiplication matricielle usuelle et la méthode du pivot de Gauss ont une complexité arithmétique cubique en la taille. Dans ce chapitre, nous montrons que pour la plupart des questions d'algèbre linéaire dense – multiplication, inversion, déterminant, résolution de système – il existe des algorithmes plus efficaces, de complexité strictement sous-cubique.

1. Introduction

En mathématiques, il est de coutume de considérer qu'un problème est rendu trivial lorsqu'il est ramené à une question d'algèbre linéaire. Du point de vue calculatoire se pose cependant la question de l'efficacité des opérations matricielles.

1.1. L'algorithmique des matrices : une tentative de classification. Les problèmes algorithmiques en algèbre linéaire cachent des difficultés très subtiles. En général, les premières questions que l'on est amené à se poser en rapport avec la manipulation des matrices concernent le calcul efficace du produit matrice-matrice, du produit matrice-vecteur, de l'inverse, ainsi que la résolution de systèmes.

Les réponses à ces questions varient fortement selon le type de matrices considérées. Une classification possible est la suivante.

Les matrices denses. Ce sont les matrices quelconques, sans aucune structure particulière. Nous verrons que l'algorithmique des matrices denses peut se ramener essentiellement au produit matriciel, dont la complexité est une question extrêmement délicate.

Les matrices creuses. Beaucoup de problèmes linéaires se formulent en termes de matrices possédant un nombre important d'éléments nuls, dites « creuses ». Dans ce cas, l'algorithmique dense est inappropriée ; il est possible de tirer parti de la forme creuse en utilisant des outils mieux adaptés, à base de récurrences linéaires.

Les matrices structurées. Enfin, il existe des familles de matrices particulières, souvent associées à des applications linéaires entre espaces de polynômes, telles que les matrices de type Sylvester pour le résultant, de type Vandermonde pour l'évaluation et l'interpolation, de type Toeplitz et Hankel pour l'approximation de Padé et de Padé-Hermite, etc ... Pour ces types de matrices clairement identifiés, on développe soit une algorithmique *ad hoc*, soit une théorie unifiée reposant sur le concept de « rang de déplacement ».

Schématiquement, l'algorithmique des matrices denses est la plus lente, de complexité située entre $O(n^2)$ et $O(n^3)$, en taille n. La complexité du calcul avec les matrices creuses est de l'ordre de $O(n^2)$, alors que celle des matrices structurées est en $O(n)$, à des facteurs logarithmiques près.

L'algorithmique des matrices denses constitue l'objet d'étude de ce chapitre. Les matrices creuses seront abordées brièvement au Chapitre 9 et les matrices structurées feront l'objet d'une étude plus approfondie au Chapitre 10.

Nous allons travailler avec un corps effectif noté $\mathbb{K}$ et avec l'algèbre des matrices carrées $\mathcal{M}_n(\mathbb{K})$ à coefficients dans $\mathbb{K}$. Signalons toutefois que la plupart des résultats de ce chapitre s'étendent au cas où le corps $\mathbb{K}$ est remplacé par un anneau effectif $\mathbb{A}$, et les matrices carrées par des matrices rectangulaires à coefficients dans $\mathbb{A}$.

1.2. Résultat principal. Les questions qui seront étudiées, ou simplement évoquées, dans ce chapitre sont celles de la complexité du calcul du produit matriciel dans $\mathcal{M}_n(\mathbb{K})$, de l'inverse et du déterminant, du polynôme caractéristique ou minimal, de la résolution de systèmes linéaires, ou encore de la mise sous diverses « formes canoniques » (échelon, LUP, compagnon par blocs, ...)

Pour ces questions, on dispose d'une première famille d'algorithmes « naïfs », reposant sur l'application directe des définitions ou des propriétés mathématiques. Si l'algorithme naïf de multiplication a une complexité raisonnable, cubique en la taille, dans d'autres cas l'emploi des algorithmes naïfs est vivement déconseillé. Par exemple, si $A = (a_{i,j})_{i,j=1}^{n}$ est une matrice de $\mathcal{M}_n(\mathbb{K})$, on n'utilisera pas la définition

$$\det(A) = \sum_{\sigma \in \mathcal{S}_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$$

pour le calcul de son déterminant ; en effet, cela mènerait à un algorithme de complexité $O(n \cdot n!)$, donc hyper-exponentielle en la taille de A. De même, la définition du rang d'une matrice comme la taille maximale d'un mineur non-nul ne se prête pas directement, via une recherche exhaustive, à une algorithmisation efficace. Dans un registre légèrement différent, les formules de Cramer pour résoudre un système linéaire ne sont pas d'une grande utilité pratique (en revanche, elles servent à borner les tailles des solutions, et s'avèrent souvent utiles dans les estimations de complexité).

Une seconde classe d'algorithmes, basés sur la *méthode d'élimination de Gauss*, permet de résoudre de manière raisonnablement efficace la plupart des problèmes évoqués plus haut. Appliquée à une matrice A de $\mathcal{M}_n(\mathbb{K})$, cette méthode fournit des algorithmes de complexité $O(n^3)$ pour le calcul du rang $\operatorname{rg}(A)$, du déterminant $\det(A)$, de l'inverse A^{-1} (si A est inversible), pour le calcul d'une forme échelonnée et d'une décomposition LUP de A, ainsi que pour la résolution d'un système linéaire de matrice A.

Le résultat principal de ce chapitre est que l'on peut faire mieux, et qu'il existe une constante $2 \leq \omega < 3$ (dépendante *a priori* du corps de base $\mathbb{K}$), qui contrôle la complexité du produit matriciel et de toutes les opérations d'algèbre linéaire.

Pour formaliser ce point, on associe à chacun des problèmes son *exposant*. Dans le cas du produit, il est défini comme suit.

DEFINITION 1 (Exposant de la multiplication matricielle). On dit que le réel θ est un *exposant faisable* pour la multiplication de matrices sur le corps $\mathbb{K}$ s'il existe un algorithme de complexité arithmétique $O(n^\theta)$ pour multiplier deux matrices arbitraires de $\mathcal{M}_n(\mathbb{K})$. On définit $\omega_{\text{mul}} = \inf\{\theta \mid \theta \text{ est un exposant faisable}\}$.

A priori, on devrait mettre en indice le corps $\mathbb{K}$ pour les exposants θ et ω_{mul}. Les résultats présentés dans la suite sont cependant indépendants de $\mathbb{K}$ [1]. L'algorithme naïf pour le produit est de complexité $O(n^3)$, donc $\theta = 3$ est un exposant faisable

1. On peut montrer que la constante ω_{mul} ne dépend que de la caractéristique du corps de base $\mathbb{K}$ et il est conjecturé qu'elle est même entièrement indépendante du corps $\mathbb{K}$.

et $\omega_{\text{mul}} \leq 3$. D'un autre côté, ω_{mul} est forcément au moins égal à 2 : le nombre d'éléments d'une matrice est n^2 et il faut bien autant d'opérations pour l'écrire.

Ensuite, on peut définir de la même manière les exposants de tous les autres problèmes, de la forme $\omega_{\text{inv}}, \omega_{\text{polcar}}, \omega_{\text{det}}, \omega_{\text{LUP}}$, etc ... On a alors le résultat suivant.

THÉORÈME 1. *Les exposants ω_{mul}, ω_{inv}, ω_{polcar}, ω_{det}, ω_{LUP}, sont tous égaux, et inférieurs à 2,38 ; on note ω leur valeur commune. L'exposant correspondant à la résolution de systèmes linéaires est inférieur ou égal à ω.*

Ce théorème contient deux (familles de) résultats :

— des preuves d'*équivalence* entre problèmes,

— des bornes supérieures sur leur complexité, c'est-à-dire des algorithmes.

Dans le présent chapitre, il est impossible de démontrer ce théorème dans son intégralité. Nous nous contenterons de prouver (et encore, sous des hypothèses simplificatrices) que le produit et l'inverse ont le même exposant, et que celui-ci est inférieur à $\log_2(7) < 2{,}81$.

Nous montrerons également que $\omega_{\text{det}} \leq \omega_{\text{polcar}} \leq \omega_{\text{mul}} = \omega_{\text{inv}}$ et laisserons en exercice (Exercice 12) la preuve de l'inégalité $\omega_{\text{inv}} \leq \omega_{\text{det}}$ qui permet de conclure que cette suite d'inégalités est en fait une suite d'égalités.

1.3. Applications. Les problèmes d'algèbre linéaire mentionnés ci-dessus sont très fréquemment rencontrés en pratique ; les résultats du Théorème 1 seront invoqués à plusieurs reprises dans ce cours. Par exemple, l'algorithme de Beckermann-Labahn (Chapitre 7) pour le calcul d'approximants de Padé-Hermite repose ultimement sur des produits de matrices de polynômes. C'est aussi le cas de l'algorithme de Storjohann pour la résolution de systèmes linéaires à coefficients polynomiaux (Chapitre 11), et de l'algorithme de recherche de solutions séries d'équations algébriques (Chapitre 3) ou différentielles linéaires (Chapitre 13).

De nombreux algorithmes de résolution de systèmes polynomiaux reposent aussi sur de l'algèbre linéaire : les méthodes de calcul d'une base de Gröbner (Chapitre 21) peuvent se ramener à des systèmes linéaires (éventuellement creux) en très grande taille ; l'algorithme de « résolution géométrique » (Chapitre 26) utilise une itération de Newton faisant intervenir des calculs d'inverses de matrices de séries formelles. Dans un contexte différent, les algorithmes pour les séries D-finies (Chapitre 14) et les algorithmes de sommation et d'intégration (Chapitre 30) font intervenir comme sous-problème la résolution d'un système linéaire.

L'algèbre linéaire est au cœur de nombreux autres problèmes algorithmiques qui ne seront pas abordés dans ce cours ; c'est le cas de la factorisation des entiers et des polynômes (Chapitre 32), ou encore du logarithme discret en cryptanalyse. Mentionnons enfin qu'une large partie des ordinateurs du monde passent leurs cycles à faire des calculs d'algèbre linéaire, que ce soit pour faire des calculs d'optimisation combinatoire (programmation linéaire par l'algorithme du simplexe), de la simulation numérique (méthode des éléments finis), ou de la recherche de pages web (le système de classement PageRank utilisé par le moteur de recherche Google repose sur la recherche d'un vecteur propre d'une gigantesque matrice creuse).

2. Multiplication de matrices

Tout comme le produit d'entiers et de polynômes, la multiplication matricielle est une opération fondamentale en calcul formel, dont l'étude de complexité s'avère être un problème hautement non trivial.

L'algorithme naïf pour effectuer le produit d'une matrice de taille $m \times n$ par un vecteur de taille n utilise $O(mn)$ opérations arithmétiques : mn multiplications

et $m(n-1)$ additions. Un résultat de Winograd affirme qu'en général on ne peut pas faire mieux : par exemple, pour les matrices réelles dont les coefficients sont algébriquement indépendants sur $\mathbb{Q}$, l'algorithme naïf *est optimal*. Puisque le produit matriciel peut s'interpréter comme une succession de produits matrice-vecteur, une question naturelle est de savoir si la multiplication *naïve* de deux matrices est aussi quasi-optimale.

La réponse à cette question, et les résultats les plus importants de cette section, classés par ordre chronologique de découverte, sont regroupés dans le théorème suivant.

THÉORÈME 2 (La multiplication matricielle naïve n'est pas optimale). *Soit $\mathbb{K}$ un corps commutatif. On peut multiplier deux matrices de $\mathcal{M}_n(\mathbb{K})$ en*

(a) $O(n^3)$ opérations dans $\mathbb{K}$, par l'algorithme naïf ;

(b) $n^2\lceil\frac{n}{2}\rceil + 2n\lfloor\frac{n}{2}\rfloor \simeq \frac{1}{2}n^3 + n^2$ multiplications dans $\mathbb{K}$;

(c) $n^2\lceil\frac{n}{2}\rceil + (2n-1)\lfloor\frac{n}{2}\rfloor \simeq \frac{1}{2}n^3 + n^2 - \frac{n}{2}$ multiplications dans $\mathbb{K}$ si la caractéristique de $\mathbb{K}$ est différente de 2 et si la division par 2 est gratuite ;

(d) $O(n^{\log_2(7)}) \simeq O(n^{2,81})$ opérations dans $\mathbb{K}$.

Ces résultats restent valables sur un anneau commutatif $\mathbb{A}$; pour (a) et (d) l'hypothèse de commutativité peut même être supprimée. La partie (b) est due à Winograd et fut historiquement la première amélioration de l'algorithme naïf. Elle réduit de moitié le nombre de multiplications, en augmentant en revanche le nombre d'additions. Cela constitue déjà un progrès pour une large classe d'anneaux $\mathbb{A}$ dans lesquels la multiplication est plus coûteuse que l'addition [2]. Par exemple, la technique du « scindage binaire » qui sera utilisée au Chapitre 15 pour le calcul d'un terme d'une récurrence, ou encore pour le calcul du développement décimal de e, requiert de multiplier des matrices contenant de gros nombres entiers.

L'amélioration (c) a été obtenue par Waksman, et implique qu'on peut effectuer le produit de deux matrices 2×2 en 7 multiplications. L'inconvénient commun des algorithmes de Winograd et de Waksman est l'utilisation de la commutativité de $\mathbb{K}$, qui devient un obstacle à une application récursive, ne permettant pas d'améliorer (la borne supérieure 3 sur) l'exposant ω_{mul}. Beaucoup pensaient que tout résultat de type (b) et (c) serait optimal, au sens que $\frac{1}{2}n^3$ multiplications dans $\mathbb{K}$ seraient nécessaires pour le produit dans $\mathcal{M}_n(\mathbb{K})$. Ceci fut contredit par la découverte par Strassen de (d) ; ce résultat découle d'un fait d'une simplicité déconcertante, mais d'une portée considérable : deux matrices 2×2 peuvent être multipliées en 7 multiplications même si $\mathbb{K}$ n'est pas commutatif.

La non-optimalité de la multiplication matricielle naïve justifie l'introduction de la définition suivante.

DEFINITION 2 (Fonction de multiplication matricielle). On dit que l'application $\mathsf{MM} : \mathbb{N} \to \mathbb{N}$ est une *fonction de multiplication matricielle* (pour un corps $\mathbb{K}$) si :

- on peut multiplier deux matrices arbitraires de $\mathcal{M}_n(\mathbb{K})$ en au plus $\mathsf{MM}(n)$ opérations dans $\mathbb{K}$;
- MM est croissante et vérifie l'inégalité $\mathsf{MM}(n/2) \leq \mathsf{MM}(n)/4$ pour $n \in \mathbb{N}$.

Ainsi, le Théorème 2 montre que les fonctions $n \mapsto n^3$ ou $n \mapsto n^{\log_2(7)}$ sont (à des constantes près) des fonctions de multiplication matricielles. Comme dans le cas de la fonction de multiplication polynomiale M introduite au Chapitre 2, l'intérêt de cette notion est qu'elle permet d'énoncer des résultats de complexité indépendants du choix de l'algorithme utilisé pour multiplier les matrices.

2. C'est par exemple le cas de $\mathbb{Z}$ et de $\mathbb{Z}[X]$, mais pas de $\mathbb{Q}$ ou de $\mathbb{Q}(X)$.

La seconde condition de la Définition 2 établit la cohérence avec la Définition 1 : si $\mathsf{MM}(n) = c \cdot n^\theta$, pour une certaine constante $c > 0$, alors θ doit être supérieur ou égal à 2. Cette condition permettra de montrer que dans des algorithmes de type « diviser pour régner » pour les matrices, le coût total est essentiellement celui du dernier appel récursif.

Le reste de cette section est dédié à la preuve du Théorème 2.

2.1. Multiplication naïve. L'algorithme « cubique » de multiplication prend en entrée deux matrices $A = (a_{i,j})_{i,j=1}^n$ et $X = (x_{i,j})_{i,j=1}^n$ de $\mathcal{M}_n(\mathbb{K})$, et calcule leur produit $R = AX$ en utilisant la formule

$$r_{i,k} = \sum_{j=1}^{n} a_{i,j} x_{j,k}.$$

Le coût nécessaire pour obtenir un élément de R est donc de n multiplications et $n-1$ additions, de sorte que la complexité totale est en $O(n^3)$.

2.2. Algorithme de Winograd. Il est possible de diviser essentiellement par deux le nombre de multiplications scalaires suffisantes pour le calcul du produit AX. Ceci montre au passage que la multiplication matricielle naïve n'est pas optimale. Supposons pour simplifier que n est pair, $n = 2k$. L'algorithme de Winograd repose sur l'identité suivante, héritière de celle de Karatsuba : si ℓ est un vecteur ligne $[a_1 \cdots a_n]$ et c est un vecteur colonne ${}^t[x_1 \cdots x_n]$, alors le produit scalaire $\langle \ell, c\rangle = \sum_i a_i x_i$ s'écrit

$$\langle \ell, c\rangle = (a_1 + x_2)(a_2 + x_1) + \cdots + (a_{2k-1} + x_{2k})(a_{2k} + x_{2k-1}) - \sigma(\ell) - \sigma(c), \tag{1}$$

où $\sigma(\ell) = a_1 a_2 + \cdots + a_{2k-1} a_{2k}$ et $\sigma(c) = x_1 x_2 + \cdots + x_{2k-1} x_{2k}$. Notons que $\sigma(\ell)$ se calcule en $k = n/2$ multiplications et $k - 1 = n/2 - 1$ additions.

Si maintenant $\ell_1, \ldots, \ell_n$ sont les lignes de A et $c_1, \ldots, c_n$ sont les colonnes de X, l'élément (i, j) du produit AX vaut $\langle \ell_i, c_j\rangle$. L'idée est alors de précalculer les quantités $\sigma(\ell_i)$ et $\sigma(c_i)$ pour $1 \le i \le n$; ce précalcul demande $n(\frac{n}{2} + \frac{n}{2}) = n^2$ multiplications et $n(\frac{n}{2} - 1 + \frac{n}{2} - 1) = n^2 - 2n$ additions. Une fois déterminés les $\sigma(\ell_i)$ et $\sigma(c_i)$, on peut calculer l'ensemble des éléments $r_{i,j}$ de $R = AX$ grâce à la formule (1) en $n^2 \cdot \frac{n}{2} = \frac{1}{2}n^3$ multiplications et $n^2 \cdot (n + (\frac{n}{2} - 1) + 2) = \frac{3}{2}n^3 + n^2$ additions. Au total, le coût de la multiplication de l'algorithme de Winograd est de $\frac{1}{2}n^3 + n^2$ multiplications et $\frac{3}{2}n^3 + 2n^2 - 2n$ additions. Ainsi, on a essentiellement transformé $n^3/2$ multiplications en additions, ce qui est appréciable.

2.3. Algorithme de Waksman. Pour effectuer le produit de matrices 2×2

$$R = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x & y \\ z & t \end{bmatrix},$$

l'algorithme de Winograd ci-dessus utilise 8 multiplications, en écrivant

$$R = \begin{bmatrix} (a+z)(b+x) - ab - zx & (a+t)(b+y) - ab - ty \\ (c+z)(d+x) - cd - zx & (c+t)(d+y) - cd - ty \end{bmatrix}.$$

Il n'apporte donc aucun gain par rapport à l'algorithme naïf dans cette taille.

L'algorithme suivant, dû à Waksman, peut être vu comme une amélioration de l'algorithme de Winograd. Il effectue le produit de matrices 2×2 en 7 opérations, dès lors que dans le corps de base $\mathbb{K}$ l'élément 2 est inversible et que la division par 2 y est gratuite. L'idée est d'écrire la matrice R comme

$$R = \frac{1}{2} \begin{bmatrix} \alpha_1 - \alpha_2 & \beta_1 - \beta_2 \\ \gamma_1 - \gamma_2 & \delta_1 - \delta_2 \end{bmatrix},$$

où

$$\begin{array}{ll} \alpha_1 = (a+z)(b+x), & \alpha_2 = (a-z)(b-x), \\ \beta_1 = (a+t)(b+y), & \beta_2 = (a-t)(b-y), \\ \gamma_1 = (c+z)(d+x), & \gamma_2 = (c-z)(d-x), \\ \delta_1 = (c+t)(d+y), & \delta_2 = (c-t)(d-y), \end{array}$$

et de faire observer que, grâce à l'identité

$$(\gamma_1+\gamma_2)+(\beta_1+\beta_2) = (\alpha_1+\alpha_2)+(\delta_1+\delta_2) = 2(ab+cd+xz+ty),$$

l'élément δ_2 est une combinaison linéaire des sept autres produits $\alpha_1, \alpha_2, \ldots, \delta_1$, et donc n'a pas besoin d'être obtenu lui aussi par un produit.

L'algorithme qui s'en déduit, tout comme celui de Winograd, ne peut être utilisé de manière récursive : la preuve des formules sous-jacentes repose sur le fait que les éléments du corps de base $\mathbb{K}$ commutent ($bz = zb, bt = tb, \ldots$). Lors des appels récursifs, les objets à multiplier seraient eux-mêmes des matrices, pour lesquelles la loi de commutation n'est plus vérifiée. On dit donc que les algorithmes de Winograd et de Waksman sont des algorithmes *commutatifs*.

Exercice 1. Finir la preuve des parties (b) et (c) du Théorème 2, en généralisant les arguments précédents à des matrices de taille n, où $n \in \mathbb{N}$ n'est pas nécessairement pair.

2.4. Algorithme de Strassen. L'algorithme de Strassen fut le premier à descendre en-dessous de $O(n^3)$. Tout comme l'algorithme de Karatsuba pour les polynômes, il repose sur le gain d'une multiplication pour les matrices 2×2, qui devient un gain dans l'exposant lors d'une application récursive. La non-commutativité du produit matriciel se traduit en revanche par une contrainte supplémentaire : afin de se prêter à un emploi récursif, le nouvel algorithme doit, à la différence de l'algorithme de Waksman, satisfaire à une contrainte de non-commutativité.

Soient donc A et X des matrices de taille 2 à multiplier par un algorithme non-commutatif. Intuitivement, cela se traduit par le fait que lors des multiplications utilisées par un tel algorithme, les éléments a, b, c, d de A doivent rester à gauche, et les éléments x, y, z, t de X à droite (cela n'est pas le cas pour les algorithmes de Winograd et de Waksman, qui *mélangent* les éléments de A et X).

L'algorithme naïf est bien non-commutatif, mais demande d'effectuer 8 multiplications. L'algorithme de Strassen que nous allons présenter revient à

- calculer des combinaisons linéaires des éléments de A et des éléments de X,
- effectuer **7** produits de ces combinaisons linéaires,
- recombiner les résultats pour obtenir les quatre éléments du produit AX.

Explicitement, les formules à appliquer sont données dans les étapes 3 et 4 de l'algorithme présenté en Figure 1.

Effectuons maintenant le pas récursif. Quitte à rajouter des colonnes et des lignes nulles, on peut supposer que les matrices A et X sont de taille $n = 2^k$, avec $k \in \mathbb{N}$. On procède à une découpe en blocs de A et de X :

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \qquad X = \begin{bmatrix} x & y \\ z & t \end{bmatrix}.$$

Dans cette écriture a, b, c, d et x, y, z, t sont donc des matrices carrées de taille $n/2$.

Le point-clé est le suivant : grâce à la non-commutativité, les formules données en Figure 1 pour le cas 2×2 s'appliquent toujours si a, b, c, d et x, y, z, t sont des matrices. Ainsi, elles permettent de ramener le produit en taille n à 7 produits en taille $n/2$, plus un certain nombre C (ici $C = 18$) d'additions en taille $n/2$,

Multiplication matricielle rapide

Entrée : Deux matrices $A, X \in \mathcal{M}_n(\mathbb{K})$, avec $n = 2^k$.

Sortie : Le produit AX.

1. Si $n = 1$, renvoyer AX.
2. Décomposer $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ et $X = \begin{bmatrix} x & y \\ z & t \end{bmatrix}$, avec a, b, c, d et x, y, z, t dans $\mathcal{M}_{n/2}(\mathbb{K})$.
3. Calculer récursivement les produits
$$\left| \begin{array}{lcl} q_1 & = & a(x+z) \\ q_2 & = & d(y+t) \\ q_3 & = & (d-a)(z-y) \\ q_4 & = & (b-d)(z+t) \\ q_5 & = & (b-a)z \\ q_6 & = & (c-a)(x+y) \\ q_7 & = & (c-d)y. \end{array} \right.$$
4. Calculer les sommes
$$\left| \begin{array}{lcl} r_{1,1} & = & q_1 + q_5 \\ r_{1,2} & = & q_2 + q_3 + q_4 - q_5 \\ r_{2,1} & = & q_1 + q_3 + q_6 - q_7 \\ r_{2,2} & = & q_2 + q_7. \end{array} \right.$$
5. Renvoyer $\begin{bmatrix} r_{1,1} & r_{1,2} \\ r_{2,1} & r_{2,2} \end{bmatrix}$.

FIGURE 1. Algorithme de Strassen pour multiplier deux matrices.

utilisées pour fabriquer les combinaisons linéaires des blocs de A et de X, et celles des produits q_i.

La complexité $\mathsf{S}(n)$ de l'algorithme de Strassen satisfait donc à la récurrence

$$\mathsf{S}(n) \leq 7\,\mathsf{S}\left(\frac{n}{2}\right) + C\left(\frac{n}{2}\right)^2.$$

En invoquant le lemme « diviser pour régner » avec les choix $m = 7, p = s = 2$, $\kappa = 1$ et $q = 4$, on déduit l'inégalité

$$\mathsf{S}(n) \leq \left(1 + \frac{C}{3}\right) n^{\log_2(7)},$$

qui prouve la partie (d) du Théorème 2.

L'idée de base de l'algorithme de Strassen est tout à fait générale : améliorer la multiplication en petite taille permet d'améliorer l'exposant.

EXERCICE 2. Supposons que, pour un certain entier r fixé, on sache multiplier deux matrices de taille r à coefficients dans un corps $\mathbb{K}$ en r^θ multiplications non-commutatives dans $\mathbb{K}$ et un nombre arbitraire d'additions dans $\mathbb{K}$. Montrer que θ est un exposant faisable pour la multiplication des matrices à coefficients dans $\mathbb{K}$.

En pratique, il peut être intéressant de ne pas descendre récursivement jusqu'à la multiplication des matrices 1×1, mais d'arrêter la récursion à une taille $s > 1$ et d'utiliser l'algorithme naïf (ou tout autre algorithme, même commutatif !) en cette taille. Cela permet d'optimiser la constante du $O(\cdot)$.

EXERCICE 3. Soit n une puissance de 2. Établir un algorithme hybride de multiplication matricielle en taille $n \times n$, qui fait appel à l'algorithme de Strassen

pour $n \geq 2^d$ et à l'algorithme naïf pour $n < 2^d$. Montrer que la complexité $\mathsf{MM}(n)$ de cet algorithme vérifie $\mathsf{MM}(n) \leq \mu(d) n^{\log_2(7)}$ pour tout $n \geq 2^d$, où $\mu(d)$ est une fonction qui dépend uniquement de d. Trouver la valeur de d qui minimise $\mu(d)$.

Lorsque la taille n des matrices à multiplier n'est pas une puissance de 2, on peut rajouter artificiellement des lignes et des colonnes aux deux matrices de façon à se ramener au cas où n est une puissance de 2.

Exercice 4. Écrire une version récursive de l'algorithme de Strassen valable en taille arbitraire n (non nécessairement une puissance de 2), et qui procède en rajoutant une ligne et une colonne lorsque n est impair. Montrer que la complexité de cet algorithme vérifie $\mathsf{MM}(n) \leq C(n) n^{\log_2(7)}$ et comparer $C(n)$ à ce que l'on obtient en remplaçant n par la puissance de 2 supérieure à n.

L'algorithme de Strassen s'applique aussi au cas où le corps de base $\mathbb{K}$ est remplacé par un anneau $\mathbb{A}$. Lorsque cet anneau $\mathbb{A}$ est une extension de $\mathbb{K}$, la question est d'estimer sa complexité en termes de nombre d'opérations dans $\mathbb{K}$.

Exercice 5. Soit $\mathbb{K}$ un corps et soient d et n deux entiers naturels strictement positifs. Estimer la complexité de l'algorithme de multiplication de Strassen appliqué à deux matrices carrées de taille n, dont les éléments sont des polynômes de degré au plus d de $\mathbb{K}[X]$.

Plus généralement, tout algorithme de multiplication dans $\mathcal{M}_n(\mathbb{K})$ utilisant $\mathsf{MM}(n) = O(n^\theta)$ opérations dans $\mathbb{K}$ fournit par application à $\mathbb{A} = \mathbb{K}[X]/(X^{2d+1})$ un algorithme de multiplication de matrices polynomiales de degré d de complexité $\mathsf{MM}(n,d) = O(n^\theta\, \mathsf{M}(d))$ opérations de $\mathbb{K}$. Un algorithme de meilleure complexité permettant d'obtenir $\mathsf{MM}(n,d) = O(n^\theta d + n^2\, \mathsf{M}(d))$ sera présenté au Chapitre 5.

2.5. Interprétation des formules de Strassen. Il n'est pas immédiat de donner une intuition aux formules de Strassen (à la différence de celle de Karatsuba, qui repose de manière cachée sur l'évaluation de polynômes en $0, 1, +\infty$). Nous allons présenter un raisonnement qui permet de motiver les formules de Strassen et de les généraliser. L'explication appartient à Fiduccia ; elle est postérieure à la découverte de l'algorithme [3].

Le point de départ est l'observation que la recherche d'un algorithme non-commutatif pour effectuer le produit de matrices $R = AX$ avec

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \text{et} \qquad X = \begin{bmatrix} x & y \\ z & t \end{bmatrix}$$

revient à donner un algorithme pour la multiplication matrice-vecteur Mv, où

$$M = \begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix} \qquad \text{et} \qquad v = \begin{bmatrix} x \\ z \\ y \\ t \end{bmatrix}.$$

3. Strassen lui-même ne se souvient pas exactement de l'origine de ses formules. Selon Landsberg, « *Strassen was attempting to prove, by process of elimination, that such an algorithm did not exist when he arrived at it* ». En tout cas, Strassen affirme « *First I had realized that an estimate tensor rank* < 8 *for two by two matrix multiplication would give an asymptotically faster algorithm. Then I worked over* $\mathbb{Z}/2\mathbb{Z}$ *— as far as I remember — to simplify matters* ». Autrement dit, il y a dû y avoir recherche « à la main ».

La suite de l'argument repose sur la remarque suivante : toute matrice (de taille arbitraire) de la forme

$$\begin{bmatrix} \alpha & \alpha & \\ \varepsilon\alpha & \varepsilon\alpha & \\ & & \end{bmatrix}, \quad \begin{bmatrix} \alpha & -\alpha & \\ \varepsilon\alpha & -\varepsilon\alpha & \\ & & \end{bmatrix} \text{ ou } \begin{bmatrix} \alpha & \varepsilon\alpha & \\ -\alpha & -\varepsilon\alpha & \\ & & \end{bmatrix},$$

où $\varepsilon \in \{0,1\}$ et $\alpha \in \mathbb{K}$, dont tous les éléments, sauf les quatre marqués explicitement, sont nuls, peut être multipliée par un vecteur en une seule multiplication dans $\mathbb{K}$. On appelle une telle matrice *élémentaire*.

L'idée est alors de décomposer la matrice M en somme de 7 matrices élémentaires. On commence par soustraire à M les deux matrices élémentaires

$$E_1 = \begin{bmatrix} a & a & \\ a & a & \\ & & \end{bmatrix} \quad \text{et} \quad E_2 = \begin{bmatrix} & & \\ & d & d \\ & d & d \end{bmatrix},$$

correspondant aux éléments diagonaux de A. La matrice ainsi obtenue est

$$M_1 = \begin{bmatrix} & b-a & & \\ c-a & d-a & & \\ & & a-d & b-d \\ & & c-d & \end{bmatrix}.$$

Par construction, elle admet la propriété que l'élément $(1,1)$ du deuxième bloc est l'opposé de l'élément $(2,2)$ du premier. En soustrayant à M_1 la matrice élémentaire

$$E_3 = \begin{bmatrix} & & & \\ & d-a & a-d & \\ & d-a & a-d & \\ & & & \end{bmatrix}$$

on obtient la matrice

$$M_2 = \begin{bmatrix} & b-a & & \\ c-a & & d-a & \\ & a-d & & b-d \\ & & c-d & \end{bmatrix},$$

qui se décompose comme $M_3 + M_4$, avec

$$M_3 = \begin{bmatrix} & b-a & & \\ & & & \\ & a-d & b-d & \\ & & & \end{bmatrix} \quad \text{et} \quad M_4 = \begin{bmatrix} c-a & & d-a & \\ & & & \\ & & c-d & \\ & & & \end{bmatrix}.$$

Or, puisque $a-d = (b-d)-(b-a)$ et $d-a = (c-a)-(c-d)$, chacune des matrices M_3 et M_4 s'écrit comme somme de 2 matrices élémentaires

$$M_3 = E_4 + E_5, \quad \text{où} \quad E_4 = \begin{bmatrix} & b-a & & \\ & & & \\ & a-b & & \\ & & & \end{bmatrix}, \quad E_5 = \begin{bmatrix} & & & \\ & b-d & b-d & \\ & & & \end{bmatrix},$$

et

$$M_4 = E_6 + E_7, \qquad \text{où} \qquad E_6 = \begin{bmatrix} c-a & c-a \\ & \end{bmatrix}, \quad E_7 = \begin{bmatrix} & d-c \\ & c-d \end{bmatrix}.$$

En revenant à l'égalité ${}^t\begin{bmatrix} r_{1,1} & r_{2,1} & r_{1,2} & r_{2,2} \end{bmatrix} = \left(\sum_{i=1}^{7} E_i\right) \times {}^t v$ traduisant $R = AX$, on obtient finalement les formules données dans les étapes 3 et 4 de l'algorithme décrit en Figure 1.

Exercice 6. Généraliser l'algorithme de Strassen aux matrices de taille arbitraire. Plus précisément, étendre la preuve des formules de Strassen pour le produit matriciel en taille 2×2 afin de montrer que le produit de deux matrices carrées de taille n peut s'effectuer en $n^3 - n(n-1)/2$ multiplications scalaires non-commutatives.

Est-ce que cela permet d'améliorer la borne de Strassen $\log_2(7)$ sur l'exposant de l'algèbre linéaire ?

2.6. Peut-on faire mieux que 2,81 **?** Strassen ayant prouvé que l'exposant ω_{mul} du produit de matrices était sous-cubique, la question naturelle est devenue de savoir quelle est sa véritable valeur.

Pour ce faire, depuis les années 1970, de nombreux outils ont été développés, les uns plus sophistiqués que les autres, et l'on fit preuve de beaucoup d'ingéniosité. Des notions nouvelles sont apparues, comme celles de *complexité bilinéaire*, de *rang tensoriel* et d'*algorithme approché*, utilisées de manière intensive notamment par Bini, Pan, Schönhage, Strassen, Winograd et d'autres. Ces notions ne seront pas abordées dans ce cours.

Le meilleur algorithme actuel, datant de 2014 et dû à François Le Gall, est de complexité $O(n^{2,3728639})$. Il est cependant conjecturé que $\omega_{\text{mul}} = 2$, mais pour l'instant la preuve (ou la réfutation) de cette conjecture semble hors de portée.

2.7. En pratique. Les algorithmes rapides pour l'algèbre linéaire ont longtemps eu mauvaise presse ; concrètement, l'algorithme de Strassen, et, de manière plus marginale, un algorithme d'exposant 2,77 dû à Pan et repris par Kaporin sont les seuls algorithmes « rapides » (avec un exposant inférieur à 3) qui ont été implantés avec succès.

Dans une bonne implantation, disons sur un corps fini défini modulo un entier de taille « raisonnable », les algorithmes de Winograd et de Waksman sont immédiatement rentables. L'algorithme de Strassen devient ensuite meilleur pour des tailles de l'ordre de quelques dizaines (64 est une borne raisonnable).

L'immense majorité des autres algorithmes connus reposent sur des techniques très complexes, qui se traduisent par la présence d'énormes constantes (et de facteurs logarithmiques) dans leurs estimations de complexité. En conséquence, dans leurs versions actuelles, il ne peuvent pas être rentables pour des tailles inférieures à des millions ou des milliards ...

3. Autres problèmes d'algèbre linéaire

Il est important de savoir multiplier les matrices, mais il est encore plus utile de les inverser, calculer leur polynôme caractéristique, etc ...

3.1. Élimination de Gauss. Rappelons que l'algorithme classique d'élimination de Gauss permet de résoudre la plupart des problèmes classiques en algèbre linéaire de manière bien plus efficace que l'algorithmique naïve.

Definition 3 (Forme échelonnée). Une matrice est *échelonnée en ligne* lorsque

1. les lignes nulles sont toutes en dessous des lignes non nulles,
2. le premier coefficient non nul (appelé *pivot*) de chaque ligne non nulle est strictement à droite du premier coefficient non nul de la ligne précédente.

Une forme *échelonnée en ligne* d'une matrice A est une matrice échelonnée en ligne B équivalente à gauche à A (c'est-à-dire telle qu'il existe une matrice carrée inversible P avec $A = PB$).

L'algorithme de Gauss (sans pivot sur les colonnes) calcule, en n'utilisant que des opérations élémentaires sur les lignes (qui correspondent à des multiplications à gauche par des matrices inversibles), une forme échelonnée en ligne de A, sur laquelle le rang et le déterminant de A se lisent directement.

En effet, les lignes non nulles d'une forme échelonnée B forment une base de l'espace vectoriel engendré par les lignes de la matrice de départ A. Le nombre de lignes non nulles de B est égal au rang de A. Si A est carrée, son déterminant est égal au produit des éléments diagonaux de B.

Une variante de l'algorithme de Gauss, dite de *Gauss-Jordan*, produit même une *forme échelonnée réduite* de A : il s'agit d'une forme échelonnée de A, dont les pivots valent 1, les autres coefficients dans les colonnes des pivots étant nuls. Appliquée à la matrice augmentée $\tilde{A} = [A \,|\, I_n]$ obtenue par concaténation de la matrice A et de la matrice identité I_n, cette variante est utilisée dans le calcul de l'inverse A^{-1}. En effet, l'algorithme de Gauss-Jordan transforme la matrice $\tilde{A}$ en une matrice équivalente dont le bloc gauche est l'identité, c'est-à-dire qu'il remplace $\tilde{A}$ par la matrice $[I_n \,|\, A^{-1}]$. Le même algorithme permet de résoudre le système $Ax = b$, où $b \in \mathbb{K}^n$, en bordant la matrice A par le vecteur b.

Une autre variante de l'algorithme d'élimination de Gauss permet de calculer une *décomposition LUP* qui, à son tour, permet de résoudre des systèmes linéaires, d'inverser des matrices, de calculer des déterminants, etc.

Definition 4 (Matrices L, U, P et décomposition LUP). Une matrice est dite *triangulaire supérieure*, resp. *triangulaire inférieure*, si les éléments situés en dessous (resp. au dessus) de la diagonale principale sont nuls. Une *matrice de permutation* est une matrice carrée P dont les coefficients valent 0 ou 1, et dont chaque ligne et colonne ne contient qu'un seul élément non nul. Une *décomposition LU*, resp. *LUP*, d'une matrice A est une factorisation de la forme $A = LU$, resp. $A = LUP$, avec L triangulaire inférieure, U triangulaire supérieure et P une matrice de permutation.

En résumé, nous allons admettre l'énoncé suivant.

Théorème 3 (Élimination Gaussienne). *Pour toute matrice $A \in \mathcal{M}_n(\mathbb{K})$, il est possible de calculer en $O(n^3)$ opérations dans $\mathbb{K}$:*

- *le rang* $\mathrm{rg}(A)$*, le déterminant* $\det(A)$*, et l'inverse A^{-1} si A est inversible ;*
- *une base (affine) de solutions de $Ax = b$, pour tout b dans $\mathbb{K}^n$;*
- *une décomposition LUP, et une forme échelonnée réduite de A.*

3.2. Résultat principal. Les résultats importants de cette section sont résumés dans l'énoncé suivant.

Théorème 4 (L'élimination Gaussienne n'est pas optimale). *Soit $\mathbb{K}$ un corps et soit θ un exposant faisable pour la multiplication des matrices à coefficients dans $\mathbb{K}$. Pour toute matrice A de $\mathcal{M}_n(\mathbb{K})$, il est possible de calculer :*

(a) le déterminant $\det(A)$ *;*

(b) l'inverse A^{-1} (si A est inversible) ;

(c) la solution $x \in \mathbb{K}^n$ du système $Ax = b$, pour tout $b \in \mathbb{K}^n$, si A est inversible ;

(d) le polynôme caractéristique de A ;

(e) une décomposition LUP de A ;

(f) le rang $\mathrm{rg}(A)$ *et une forme échelonnée réduite de A ;*

(g) une base du noyau de A ;

en $\tilde{O}(n^\theta)$ *opérations dans* $\mathbb{K}$.

La preuve des parties (e), (f) et (g) dépasse le cadre de ce cours. En termes d'exposants, les assertions (a) et (d) impliquent les inégalités $\omega_{\mathrm{inv}} \leq \omega_{\mathrm{mul}}$ et $\omega_{\mathrm{polcar}} \leq \omega_{\mathrm{mul}}$. L'inégalité $\omega_{\mathrm{det}} \leq \omega_{\mathrm{polcar}}$ est évidente, puisque le terme constant du polynôme caractéristique de A est égal à $(-1)^n \det(A)$.

Dans la suite, nous allons prouver (a)–(d), sous des hypothèses simplificatrices. Nous montrerons également que $\omega_{\mathrm{mul}} \leq \omega_{\mathrm{inv}}$ et laisserons en exercice (Exercice 12) la preuve de $\omega_{\mathrm{inv}} \leq \omega_{\mathrm{det}}$. Utilisées conjointement, ces inégalités permettent de prouver le Théorème 1.

3.3. La multiplication n'est pas plus difficile que l'inversion. Soient A et B des matrices $n \times n$. On souhaite calculer $C = AB$. Pour cela, on pose

$$D = \begin{bmatrix} I_n & A & 0 \\ 0 & I_n & B \\ 0 & 0 & I_n \end{bmatrix}.$$

On a alors l'identité remarquable

$$D^{-1} = \begin{bmatrix} I_n & -A & AB \\ 0 & I_n & -B \\ 0 & 0 & I_n \end{bmatrix},$$

qui permet donc de ramener le produit en taille n à l'inversion en taille $3n$. Cela prouve notre affirmation et l'inégalité

$$\omega_{\mathrm{mul}} \leq \omega_{\mathrm{inv}}.$$

Exercice 7. Soit $\mathbb{K}$ un corps et soit $\mathsf{T}(n)$ la complexité du produit de deux matrices triangulaires inférieures de taille $n \times n$ et à coefficients dans $\mathbb{K}$. Montrer qu'il est possible de multiplier deux matrices arbitraires de $\mathcal{M}_n(\mathbb{K})$ en $O(\mathsf{T}(n))$ opérations dans $\mathbb{K}$.

3.4. L'inversion, le calcul de déterminant et la résolution de système ne sont pas plus difficiles que la multiplication. Nous allons montrer qu'il est possible d'inverser des matrices $n \times n$ par un algorithme de type « diviser pour régner » en $\tilde{O}(n^\theta)$ opérations arithmétiques, quel que soit l'exposant faisable θ.

Inverse. L'algorithme d'inversion matricielle présenté ici, dû à Strassen, est un algorithme récursif, qui peut s'interpréter comme un « pivot de Gauss par blocs ». Cet algorithme nécessite d'inverser certaines sous-matrices de A ; afin de garantir sa correction, nous allons faire l'hypothèse simplificatrice que *toutes ces matrices sont inversibles.* Le traitement du cas général (A matrice arbitraire) est plus délicat, et passe par le calcul efficace, toujours de type « diviser pour régner », d'une décomposition LUP de A.

Le point de départ est l'identité suivante (non-commutative !), dans laquelle on suppose $n = 2$ et $a, Z \in \mathbb{K} \setminus \{0\}$:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ ca^{-1} & 1 \end{bmatrix} \times \begin{bmatrix} a & 0 \\ 0 & Z \end{bmatrix} \times \begin{bmatrix} 1 & a^{-1}b \\ 0 & 1 \end{bmatrix}, \tag{2}$$

et où $Z = d - ca^{-1}b$ est le *complément de Schur* de a dans A.

Inversion matricielle rapide

Entrée : Une matrice « générique » $A \in \mathcal{M}_n(\mathbb{K})$, avec $n = 2^k$.

Sortie : Son inverse A^{-1}.

1. Si $n = 1$, renvoyer A^{-1}.
2. Décomposer $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, avec a, b, c, d dans $\mathcal{M}_{n/2}(\mathbb{K})$.
3. Calculer $e := a^{-1}$ récursivement.
4. Calculer $Z := d - ceb$.
5. Calculer $t := Z^{-1}$ récursivement.
6. Calculer $y := -ebt$, $z := -tce$, $x := e + ebtce$.
7. Renvoyer $\begin{bmatrix} x & y \\ z & t \end{bmatrix}$.

FIGURE 2. Algorithme de Strassen pour inverser une matrice.

Cette identité se déduit aisément grâce à la méthode du pivot de Gauss appliquée à A et permet d'obtenir la factorisation suivante

$$\begin{aligned} \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} &= \begin{bmatrix} 1 & -a^{-1}b \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} a^{-1} & 0 \\ 0 & Z^{-1} \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ -ca^{-1} & 1 \end{bmatrix} \\ &= \begin{bmatrix} a^{-1} + a^{-1}bZ^{-1}ca^{-1} & -a^{-1}bZ^{-1} \\ -Z^{-1}ca^{-1} & Z^{-1} \end{bmatrix}. \end{aligned} \tag{3}$$

Le point important est que l'identité (3) est « non-commutative » : elle reste valable dans tout anneau non-commutatif, pourvu que les éléments a et $Z = d - ca^{-1}b$ soient inversibles. Elle permet donc une application récursive.

L'algorithme qui en résulte est présenté en Figure 2. Dans chaque appel récursif l'algorithme effectue deux inversions, ainsi qu'un certain nombre de produits. Plus précisemment, avec les notations de la Figure 2, les 6 produits ce, bt, $ceb = (ce)b$, $ebt = e(bt)$, $(ebt)(ce)$ et $t(ce)$ sont suffisants. Cette analyse mène au résultat suivant.

THÉORÈME 5. *Si toutes les sous-matrices rencontrées au cours de l'algorithme en Figure 2 sont inversibles, le coût de l'inversion de A est de $3\,\mathsf{MM}(n) + O(n^2)$.*

DÉMONSTRATION. L'algorithme ci-dessus montre que la complexité $\mathsf{I}(n)$ de l'inversion satisfait à la récurrence

$$\mathsf{I}(n) \leq 2\,\mathsf{I}\left(\frac{n}{2}\right) + 6\,\mathsf{MM}\left(\frac{n}{2}\right) + Cn^2,$$

C étant une constante. Il suffit alors d'appliquer le lemme « diviser pour régner » avec les choix $m = p = s = 2$, $\kappa = 1$ et $q = 4$. □

Une première conséquence immédiate du Théorème 5 est que la résolution du système linéaire $Ax = b$, pour $b \in \mathbb{K}^n$, et A inversible vérifiant les hypothèses du Théorème 5, peut s'effectuer également en $O(\mathsf{MM}(n)) = O(n^\theta)$ opérations dans $\mathbb{K}$.

Déterminant. La factorisation (2) montre qu'une légère adaptation de l'algorithme d'inversion permet de calculer le déterminant de A en même temps que son inverse, en la même complexité. Pour cela, il suffit de remplacer les étapes 1, 3, 5 et 7 de l'algorithme en Figure 2 par les étapes 1′, 3′, 5′ et 7′ décrites ci-dessous.

1′. Si $n = 1$, renvoyer A^{-1} et A.

3′. Calculer $e := a^{-1}$ et $d_a := \det(a)$ récursivement.
5′. Calculer $t := Z^{-1}$ et $d_Z := \det(Z)$ récursivement.
7′. Renvoyer $\begin{bmatrix} x & y \\ z & t \end{bmatrix}$ et $d_a d_Z$.

Affaiblissement des hypothèses. À ce stade, nous avons démontré les parties (a)–(c) du Théorème 4 sous une hypothèse supplémentaire. Dans le cas où $\mathbb{K}$ est un sous-corps du corps des réels $\mathbb{R}$, on peut s'affranchir de cette hypothèse, en observant que, quelle que soit la matrice A inversible dans $\mathcal{M}_n(\mathbb{R})$, la matrice $B = {}^tA \cdot A$ est définie positive et vérifie les hypothèses du Théorème 5, et donc l'inverse de A peut se calculer grâce à l'égalité $A^{-1} = B^{-1} \cdot {}^tA$ en $O(\mathsf{MM}(n)) = O(n^\theta)$ opérations dans $\mathbb{K}$.

Il n'est pas difficile de prouver que si la matrice A est de type L (triangulaire inférieure) ou U (triangulaire supérieure) alors l'algorithme de Strassen calcule son inverse sans avoir besoin d'*aucune* hypothèse supplémentaire. Cette remarque est à la base d'un algorithme général d'inversion dû à Bunch et Hopcroft. Cet algorithme calcule d'abord une décomposition LUP d'une matrice inversible arbitraire [4] *sur un corps quelconque* $\mathbb{K}$ en $O(\mathsf{MM}(n))$ opérations, et arrive à en déduire le calcul d'inverse (et aussi la résolution de système) pour le même prix.

L'algorithme d'inversion de Strassen est en outre à la base de toute une algorithmique pour les matrices structurées, qui sera présentée au Chapitre 10.

3.5. Calcul du polynôme caractéristique. Soit A une matrice de $\mathcal{M}_n(\mathbb{K})$. Dans cette section, nous présentons un algorithme efficace, dû à Keller-Gehrig, pour le calcul du polynôme caractéristique $\chi_A(X) = \det(XI_n - A)$ de A. Nous ne décrirons en détail qu'un cas particulier très simple (et très fréquent). Plus exactement, nous ferons dans la suite l'hypothèse que $\chi_A(X) = X^n + p_{n-1}X^{n-1} + \cdots + p_0$ est irréductible dans $\mathbb{K}[X]$; en particulier, il coïncide avec le polynôme minimal de A.

L'algorithme repose de façon cruciale sur la propriété suivante.

LEMME 1. *Soit A une matrice de $\mathcal{M}_n(\mathbb{K})$, dont le polynôme caractéristique est irréductible dans $\mathbb{K}[X]$. Soit v un vecteur de $\mathbb{K}^n \setminus \{0\}$ et soit $P \in \mathcal{M}_n(\mathbb{K})$ la matrice dont la j^e colonne vaut $A^{j-1}v$, pour $1 \leq j \leq n$. Alors P est inversible et la matrice $P^{-1}AP$ est de type compagnon.*

Rappelons qu'une matrice $C = (c_{i,j})_{i,j=1}^n$ de $\mathcal{M}_n(\mathbb{K})$ est appelée « de type compagnon » si ses $n-1$ premières colonnes ne contiennent que des éléments nuls, à l'exception des éléments $c_{2,1}, c_{3,2}, \ldots, c_{n,n-1}$ qui valent tous 1. L'intérêt du Lemme 1 vient du fait classique que le polynôme caractéristique de la matrice C vaut $X^n - \sum_{i=1}^n c_{i,n}X^{i-1}$ et n'a donc besoin d'aucune opération arithmétique pour être calculé.

DÉMONSTRATION. Pour prouver ce lemme, on commence par observer que, grâce à l'hypothèse sur A, la famille $\mathcal{B} = \{v_1, v_2, \ldots, v_n\}$, où $v_i = A^{i-1}v$, forme une base du $\mathbb{K}$-espace vectoriel $\mathbb{K}^n$, et donc P est inversible. En effet, en supposant le contraire, on déduit l'existence d'un polynôme non nul $Q \in \mathbb{K}[X]$ de degré strictement inférieur à n, tel que $Q(A)v = 0$. Sans perte de généralité, ce polynôme peut être supposé de degré minimal avec cette propriété. Comme par ailleurs $\chi_A(A)v = 0$, il s'ensuit que Q divise χ_A ; or, ce dernier étant supposé irréductible, cela entraîne que Q est constant, et donc $v = 0$, une contradiction.

4. La détection de l'inversibilité est également faisable en $O(\mathsf{MM}(n))$ opérations, grâce à un algorithme qui calcule le rang de A en cette complexité ; cet algorithme dépasse notre cadre.

Algorithme de Keller-Gehrig

Entrée : Une matrice A de $\mathcal{M}_n(\mathbb{K})$, avec $n = 2^k$.

Sortie : Son polynôme caractéristique $\det(XI_n - A)$.

1. Choisir v dans $\mathbb{K}^n \setminus \{0\}$;
2. $M := A$; $P := v$;
3. Pour i de 1 à k faire
   $P := \mathsf{Cat}(P, MP)$;
   $M := M^2$;
4. $C := P^{-1}AP$;
5. Renvoyer $\left(X^n + p_{n-1}X^{n-1} + \cdots + p_0\right)$, où ${}^t[-p_0, \ldots, -p_{n-1}]$ est la dernière colonne de C.

FIGURE 3. Calcul du polynôme caractéristique par l'algorithme de Keller-Gehrig.

Puisque $Av_{i-1} = v_i$, la matrice C de l'application linéaire $w \mapsto Aw$ dans la base $\mathcal{B}$ est de type compagnon. Le théorème du changement de base fournit alors l'égalité $C = P^{-1}AP$ et permet donc de conclure. □

Les matrices A et $C = P^{-1}AP$ étant semblables, elles ont le même polynôme caractéristique. L'idée de l'algorithme de Keller-Gehrig est alors naturelle : construire la matrice P du Lemme 1, pour ensuite déterminer la matrice C à l'aide d'une multiplication et d'une inversion, et lire les coefficients du polynôme caractéristique de A sur les éléments de la dernière colonne de C.

Du point de vue de la complexité, la seule étape coûteuse est la construction de la matrice P. En effet, la méthode directe qui consiste à calculer la suite (dite *de Krylov*) $v, Av, \ldots, A^{n-1}v$ par multiplications successives d'un vecteur par la matrice A a un coût cubique en n. La remarque cruciale est que l'on peut regrouper les produits matrice–vecteur en plusieurs produits matrice–matrice, de la façon suivante : on calcule en $O(n^2)$ opérations les vecteurs v et Av, on détermine ensuite par *exponentiation binaire* les $O(\log(n))$ matrices $A, A^2, A^4, A^8, \ldots$ et on termine le calcul des colonnes de P par le calcul des produits $A^2 \times [\, v \mid Av \,] = [\, A^2v \mid A^3v \,]$, puis $A^4 \times [\, v \mid \cdots \mid A^3v \,] = [\, A^4v \mid \cdots \mid A^7v \,]$, etc. Chaque produit de ce type est effectué à l'aide d'un produit matriciel en taille $n \times n$, en rajoutant artificiellement des colonnes nulles aux facteurs droits.

L'algorithme complet est présenté en Figure 3 (où la notation $\mathsf{Cat}(U, V)$ désigne l'opération de concaténation horizontale des matrices U et V). Nous venons de prouver :

THÉORÈME 6. *Soit $\mathbb{K}$ un corps et supposons qu'on dispose d'un algorithme de multiplication dans $\mathcal{M}_n(\mathbb{K})$ de complexité $\mathsf{MM}(n) = O(n^\theta)$, avec $\theta \geq 2$. Si le polynôme caractéristique d'une matrice A de $\mathcal{M}_n(\mathbb{K})$ est irréductible dans $\mathbb{K}[X]$, l'algorithme en* Figure 3 *le calcule en $O(\mathsf{MM}(n)\log(n))$ opérations dans $\mathbb{K}$.*

Le calcul de la suite de Krylov sera utilisé au Chapitre 9 dans l'algorithme de Wiedemann pour la résolution des systèmes creux, et de façon moins transparente, au Chapitre 11 dans l'algorithme de Storjohann pour la résolution de systèmes linéaires à coefficients polynomiaux.

Exercices

EXERCICE 8 (Multiplication et division rapides de nombres complexes).

1. Montrer qu'il est possible d'effectuer le produit matrice-vecteur
$$\begin{bmatrix} a & -b \\ b & a \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix}$$
en utilisant seulement 3 multiplications scalaires.
2. En déduire que deux nombres complexes peuvent être multipliés en utilisant 3 multiplications réelles.
3. Montrer qu'on peut calculer le quotient de deux nombres complexes en utilisant au plus 7 multiplications et divisions réelles.
4. Donner un algorithme qui utilise seulement 6 opérations de ce type.

EXERCICE 9 (Multiplication rapide de quaternions).

1. Soit $A = (a_{ij})$ une matrice $n \times n$ à coefficients dans $\mathbb{K}$, telle que $a_{ij} = \pm a_{ji}$ pour $i < j$. Soit v un vecteur quelconque de $\mathbb{K}^n$. Si ℓ est le nombre d'éléments non-nuls de l'ensemble $\{a_{ij},\, i < j\}$, montrer que $n+\ell$ multiplications suffisent pour calculer le produit Av.
2. Montrer qu'on peut calculer le produit d'une matrice $n \times n$ et d'une matrice $n \times 2$ en utilisant $\frac{3}{2}n^2 + \frac{5}{2}n$ multiplications scalaires.

   Indication : Remplacer le produit $M \times [\ v_1 \mid v_2\ ]$ par $\begin{bmatrix} 0 & M \\ M & 0 \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$, puis décomposer la matrice $A = \begin{bmatrix} 0 & M \\ M & 0 \end{bmatrix}$ en somme de deux matrices vérifiant les hypothèses de la question précédente.
3. Un *quaternion* est une combinaison formelle $a \cdot 1 + b \cdot i + c \cdot j + d \cdot k$, avec $a, b, c, d \in \mathbb{R}$ et i, j, k éléments vérifiant la table de multiplication interne

$\cdot$	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

   (par exemple, $i \cdot j = k$ et $k \cdot j = -i$). La multiplication $\cdot$ se prolonge par $\mathbb{R}$-linéarité à l'ensemble $\mathbb{H}$ des quaternions et en fait un corps non-commutatif.

   Montrer que deux quaternions arbitraires peuvent être multipliés en utilisant seulement 10 multiplications réelles.

EXERCICE 10. Soit P un polynôme de degré au plus n à coefficients dans un corps $\mathbb{K}$. Soit $\theta > 2$ un réel tel que $O(n^\theta)$ opérations dans $\mathbb{K}$ suffisent pour multiplier deux matrices arbitraires de $\mathcal{M}_n(\mathbb{K})$.

1. Donner un algorithme pour évaluer P sur $\lceil \sqrt{n} \rceil$ éléments de $\mathbb{K}$ en $O(n^{\theta/2})$ opérations.
2. Si $A \in \mathcal{M}_n(\mathbb{K})$, montrer qu'on peut calculer la matrice $P(A)$ en $O(n^{\theta+1/2})$ opérations de $\mathbb{K}$.
3. Si $Q \in \mathbb{K}[X]$ est un autre polynôme de degré au plus n, montrer qu'il est possible de calculer les n premiers coefficients du polynôme $P(Q(X))$ en $O(n^{\frac{\theta+1}{2}})$ opérations dans $\mathbb{K}$.

Indication : Écrire P sous la forme $P_0(X) + P_1(X)X^d + P_2(X)(X^d)^2 + \cdots$, avec d bien choisi et $P_i(X)$ de degrés au plus $d-1$.

EXERCICE 11 (Inversion de Strassen des matrices polynomiales). Soit $M(X) \in \mathcal{M}_n(\mathbb{K}[X])$ une matrice polynomiale carrée de taille n et de degré borné par d.

Exhiber une borne supérieure pour la complexité (exprimée en termes d'opérations arithmétiques dans $\mathbb{K}$ et en fonction des deux paramètres n et d) de l'algorithme de Strassen pour le calcul de $M(X)^{-1}$, sous l'hypothèse que toutes les matrices rencontrées au cours de l'algorithme sont inversibles.

EXERCICE 12 (Dérivation automatique et calcul d'inverse). Le but de cet exercice est de démontrer que le calcul de l'inverse d'une matrice n'est pas plus difficile que le calcul du déterminant, en utilisant des techniques de dérivation automatique.

Pour cela, formalisons brièvement ce qu'on entend par *programme*. Dans notre contexte, un programme prend en entrée des indéterminées $Y_1, \ldots, Y_m$ sur un corps $\mathbb{K}$, et effectue une suite d'*instructions* $g_1, \ldots, g_L$. Pour tout $i = 1, \ldots, L$, l'instruction g_i consiste simplement en la définition d'un polynôme G_i, de la forme

$$g_i : \ G_i = \mathsf{Arg}_i \operatorname{op}_i \mathsf{Arg}'_i,$$

selon les règles suivantes :

— Arg_i et Arg'_i sont dans $\mathbb{K} \cup \{G_1, \ldots, G_{i-1}\} \cup \{Y_1, \ldots, Y_m\}$.

— op_i est dans $\{+, -, \times\}$.

La *taille* d'un tel programme est L. On dit qu'un tel programme *calcule* un polynôme F si F appartient à l'ensemble $\{G_1, \ldots, G_L\}$; on dit de même que le programme calcule $\{F_1, \ldots, F_s\}$ si tous les F_i sont dans $\{G_1, \ldots, G_L\}$.

Par exemple, le programme ayant pour entrées Y_1, Y_2, Y_3, Y_4, pour instructions

$$\begin{array}{rcl} G_1 & = & Y_1 \times Y_4 \\ G_2 & = & Y_2 \times Y_3 \\ G_3 & = & G_1 - G_2 \end{array}$$

calcule le déterminant $Y_1Y_4 - Y_2Y_3$ de la matrice

$$\begin{pmatrix} Y_1 & Y_2 \\ Y_3 & Y_4 \end{pmatrix};$$

il a taille 3.

Si G est dans $\mathbb{K}[Y_1, \ldots, Y_m]$, on appelle *gradient* de G l'ensemble de ses m dérivées partielles $\partial G/\partial Y_i$, $i = 1, \ldots, m$.

1. Soient G et H dans $\mathbb{K}[Y_1, \ldots, Y_m]$. Exprimer les dérivées partielles de $G+H$, $G - H$ et $G \times H$, en fonction de G, H et des dérivées partielles de G et H.

   En déduire qu'étant donné un programme de taille L, qui calcule un polynôme G de $\mathbb{K}[Y_1, \ldots, Y_m]$, on peut en déduire un programme de taille $O(Lm)$ qui calcule G et son gradient.

2. Soient G dans $\mathbb{K}[Y_1, \ldots, Y_m]$ et H dans $\mathbb{K}[Y_1, \ldots, Y_{m+1}]$, tels que

   $$G = H(Y_1, \ldots, Y_m, Y_i \times Y_j),$$

   avec $1 \leq i, j \leq m$. Exprimer les dérivées partielles de G en fonction de H et de ses dérivées partielles. Traiter ensuite les cas

   $$G = H(Y_1, \ldots, Y_m, Y_i \pm Y_j).$$

3. Soient G dans $\mathbb{K}[Y_1, \ldots, Y_m]$ et H dans $\mathbb{K}[Y_1, \ldots, Y_{m+1}]$, tels que

   $$G = H(Y_1, \ldots, Y_m, Y_i \text{ op } Y_j),$$

   avec $1 \leq i, j \leq m$, et op dans $\{+, -, \times\}$. On suppose connu un programme de taille L qui calcule H et son gradient. Montrer qu'on peut en déduire un programme de taille $L + O(1)$ qui calcule G et son gradient.

   En déduire par récurrence que si un polynôme G de $\mathbb{K}[Y_1, \ldots, Y_m]$ peut être calculé par un programme de taille L, G et son gradient peuvent être calculés par un programme de taille $O(L)$, où la constante dans le $O(\cdot)$ ne dépend pas du nombre de variables m.

4. Soient $X_{1,1}, \ldots, X_{1,n}, \ldots, X_{n,1}, \ldots, X_{n,n}$ des indéterminées, et soit D le déterminant de la matrice
$$M = \begin{pmatrix} X_{1,1} & \cdots & X_{1,n} \\ \vdots & & \vdots \\ X_{n,1} & \cdots & X_{n,n} \end{pmatrix}.$$
Montrer que le gradient de D permet de retrouver les entrées de la *comatrice* N de M, où N est l'unique matrice satisfaisant
$$M \cdot {}^tN = {}^tN \cdot M = DI_n,$$
I_n étant la matrice identité de taille n, et ${}^tN$ la transposée de N.
5. Montrer qu'étant donné un programme de taille L qui calcule D, on peut en déduire un programme de taille $O(L)$, effectuant éventuellement des divisions, qui calcule les coefficients de l'inverse de M.
6. En déduire que $\omega_{\text{inv}} \leq \omega_{\text{det}}$.

Notes

L'optimalité de l'algorithme naïf pour le produit matrice-vecteur mentionnée au début de la Section 2 a été prouvée par Winograd [60]. Les parties (b) et (c) du Théorème 2 sont dues à Winograd [61] et Waksman [57]. Paterson [44] a prouvé que, si l'on s'interdit les soustractions, la multiplication matricielle cubique est optimale. Le Théorème 4, dont le titre est emprunté à l'article de Strassen [52], montre que du point de vue de la complexité on peut faire mieux que la méthode du pivot de Gauss. C'est un résultat surprenant car Klyuyev et Kokovkin-Shcherbak [28] avaient prouvé que l'élimination Gaussienne est optimale si l'on se restreint à des opérations sur des lignes et des colonnes. C'est surtout un résultat qui a ouvert la voie à tout un domaine de recherche toujours très actif, *la théorie de la complexité bilinéaire*. Les livres de Pan [42], de Bürgisser, Clausen et Shokrollahi [11, §15–16] et de Abdeljaoued et Lombardi [1, §7–10], ainsi que les articles de synthèse de Pan [41] et de Strassen [54] constituent de bonnes références sur ce sujet.

L'interprétation des formules de Strassen donnée en Section 2.5 et l'exercice 6 sont dus à Fiduccia [16]. L'algorithme d'inversion des matrices génériques en Section 3.4 est dû à Strassen [52]. La remarque de Schönhage permettant de traiter l'inversion rapide d'une matrice réelle arbitraire est tirée de [48]. L'algorithme général d'inversion évoqué à la fin de la Section 3.4, ainsi que la preuve des égalités $\omega_{\text{inv}} = \omega_{\text{LUP}} = \omega_{\text{mul}}$ ont été donnés par Bunch et Hopcroft [9]. La preuve de l'inégalité $\omega_{\text{inv}} \leq \omega_{\text{det}}$ esquissée dans l'exercice 12 est due à Baur et Strassen [2]. La réduction présentée en Section 3.3 provient de l'article de Munro [36]. La partie (f) du Théorème 4 est due à Keller-Gehrig [27], et la partie (g) est due à Bürgisser, Karpinski et Lickteig [10]. Le fait que l'exposant ω_{mul} ne dépend que de la caractéristique du corps de base a été prouvé par Schönhage [49].

L'algorithme décrit dans la Section 3.5 a été conçu par Keller-Gehrig [27] ; il s'agit d'une version accélérée d'un algorithme classique de Krylov-Danilevsky [15, 29]. Il est possible d'adapter cet algorithme au cas où A est une matrice arbitraire. Plus précisément, l'algorithme de Keller-Gehrig calcule dans ce cas une base de $\mathbb{K}^n$ dans laquelle la matrice de l'endomorphisme de multiplication par A est triangulaire par blocs, avec des blocs diagonaux de type compagnon. Autrement dit, il procède à une mise en *forme de Frobenius faible*. Un raffinement de cet algorithme, dû à Giesbrecht [19], permet de calculer la *forme de Frobenius forte* (matrice diagonale par blocs de type compagnon) également en complexité $\tilde{O}(\mathsf{MM}(n))$; cet algorithme permet aussi le calcul, en la même complexité, du polynôme minimal et de tous les autres facteurs invariants de A. Les algorithmes de Giesbrecht [19]

sont probabilistes ; des versions déterministes ont été données par Storjohann [51]. Keller-Gehrig [27] propose également un algorithme différent de celui décrit dans la Section 3.5, calculant le polynôme caractéristique d'une matrice *générique* en seulement $O(\mathsf{MM}(n))$ opérations ; voir l'exercice 11 en page 398. Une version (probabiliste) de même complexité et sans hypothèse de généricité a été proposée par Pernet et Storjohann [45].

Giesbrecht [19] et Storjohann [51] montrent que $\omega_{\text{Frobenius}} = \omega_{\text{mul}}$ et que l'évaluation d'un polynôme de degré d en une matrice de $\mathcal{M}_n(\mathbb{K})$ peut se faire en $\tilde{O}(\mathsf{MM}(n) + d)$ opérations dans $\mathbb{K}$, ce qui améliore le résultat (2) de l'exercice 10. Cet exercice est inspiré des articles de Borodin et Munro [7] pour (1), de Paterson et Stockmeyer [43] pour (2) et de Brent et Kung [8] pour (3) ; le résultat de (1) sera amélioré au Chapitre 5 et celui de (3) au Chapitre 3.

À l'heure actuelle, on ne sait pas si les exposants des problèmes suivants

— résoudre un système ;

— calculer le rang ;

— décider si une matrice est inversible ;

sont eux aussi égaux à ω_{mul}. Grâce au Théorème 4, on sait que ces problèmes ne sont pas plus difficiles que la multiplication, mais peut-être sont-ils plus faciles ...

Un autre problème ouvert est le calcul du polynôme caractéristique par un algorithme déterministe de complexité $O(\mathsf{MM}(n))$.

Complexité des algorithmes commutatifs. Il est naturel de se demander s'il est possible d'obtenir des algorithmes *à la Winograd* plus efficaces, en regroupant les termes de (1) en paquets plus gros plutôt que deux par deux. La réponse est négative. Harter [20] a montré qu'il n'existe pas de généralisation de (1) de la forme

$$\langle \ell, c\rangle = \lambda(a_1, x_1, a_2, x_2, \ldots) + \mu(a_1, x_1, a_2, x_2, \ldots) + f(a_1, a_2, \ldots) + g(x_1, x_2, \ldots),$$

où λ et μ sont des formes linéaires et f et g des fonctions quelconques. Ja'Ja [24] a montré plus généralement qu'aucun algorithme commutatif ne peut apporter un gain d'un facteur supérieur à 2 sur l'algorithme naïf, et en particulier ne peut pas améliorer l'exposant 3. De ce point de vue, les algorithmes de Winograd et Waksman sont optimaux dans la classe des algorithmes commutatifs.

Algorithme de Strassen. Winograd [17] a donné une version de l'algorithme de Strassen utilisant seulement $C = 15$ additions et soustractions (au lieu de $C = 18$). Cela permet de descendre la constante dans le $O(n^{\log_2(7)})$ de 7 à 6 pour l'algorithme en Figure 1. Cette constante a été abaissée à 3,92 par Fischer [18].

Dans le cas où la taille n de la matrice n'est pas une puissance de 2, il vaut mieux éviter d'appliquer l'algorithme de Strassen à la matrice augmentée en taille la puissance de 2 supérieure à n, comme le montre l'Exercice 4. D'autres stratégies sont décrites dans de Huss-Lederman, Jacobson, Tsao, Turnbull et Johnson [23].

Bornes inférieures vs. bornes supérieures. Le problème des bornes inférieures est très difficile. En taille $n = 2$, il a été montré [21, 62] que le résultat de Strassen est optimal du point de vue des multiplications : on ne peut pas effectuer le produit de deux matrices carrées de taille 2 en seulement 6 multiplications scalaires. Récemment Landsberg [32] a obtenu une généralisation de ce résultat, en utilisant des outils géométriques. Probert [46] a montré que la version de l'algorithme de Strassen due à Winograd est optimale également vis-à-vis du nombre total d'opérations arithmétiques : il n'existe aucun algorithme pour multiplier deux matrices 2×2 en 7 multiplications et seulement 14 additions et soustractions.

La taille $n = 2$ est la seule pour laquelle on connaît le nombre minimal de multiplications scalaires requises pour effectuer le produit de matrices carées $n \times n$. En taille $n = 3$, la meilleure borne inférieure connue à ce jour est 19 [6], alors

que le meilleur algorithme commutatif utilise 22 multiplications [35] et le meilleur algorithme non-commutatif utilise 23 multiplications [30]. Pour une taille arbitraire, toutes les bornes inférieures connues sont seulement quadratiques en n : la meilleure, $\frac{5}{2}n^2-3n$, est due à Bläser [5]. Si le corps des scalaires est $\mathbb{R}$ ou $\mathbb{C}$, et dans un modèle restreint de complexité où les seules multiplications scalaires permises se font par des scalaires de module borné, Raz [47] a prouvé une borne inférieure en $c\,n^2\log n$.

Matrices de petite taille. Il existe un grand nombre d'autres schémas *à la Strassen* pour multiplier des matrices de petite taille. Un recensement des meilleurs algorithmes non-commutatifs connus pour le produit des matrices de petite taille a été effectué par Smith [50].

Par exemple, Sýkora [55] a montré qu'on peut multiplier deux matrices $n\times n$ en $n^3-(n-1)^2$ produits non-commutatifs ; cet algorithme généralise à la fois ceux de Strassen et de Laderman. Des techniques plus sophistiquées ont permis à Pan [38, 39] de montrer que le produit de matrices $n\times n$ peut s'effectuer en $n^3/3+9n^2/2-n/3$ multiplications non-commutatives. Chacun de ces schémas, appliqué récursivement, donne un algorithme de produit matriciel sous-cubique. Par exemple, Pan [39] montre que l'on peut multiplier des matrices 48×48 en 47216 multiplications non-commutatives, ce qui mène à $\omega_{\text{mul}}<2{,}781$.

À la recherche de $\omega_{\text{mul}}=2$. L'idée initiale de Strassen de combiner une bonne multiplication en taille fixée et la puissance de la récursivité pour améliorer l'exposant ω_{mul} n'a pas permis d'aller en dessous de 2,7. Les améliorations ultérieures reposent toutes sur une accumulation d'idées de plus en plus sophistiquées. Une première idée fut d'utiliser le produit des matrices rectangulaires. Si l'on dispose d'un algorithme pour multiplier deux matrices $(m\times n)\times(n\times p)$ en ℓ multiplications, alors on sait effectuer en ℓ multiplications chacun des produits en tailles $(n\times p)\times(p\times m)$ et $(p\times m)\times(m\times n)$ [22, 37]. Cet résultat est lié au théorème de transposition de Tellegen qui sera vu au Chapitre 12. Une décomposition par blocs permet alors d'obtenir un algorithme pour le produit de matrices carrées de taille mnp en ℓ^3 opérations, et d'obtenir ainsi la majoration $\omega_{\text{mul}}\le 3\log(\ell)/\log(mnp)$. Ce résultat n'a permis d'améliorer la majoration de ω_{mul} que combiné avec d'autres concepts, comme les *algorithmes approchés* et le *rang tensoriel limite* (*border rank* en anglais) introduits par Bini et ses collaborateurs [3, 4]. L'article de Bini, Capovani, Romani et Lotti [4] donne un algorithme approché qui utilise 10 multiplications pour le produit matriciel en taille $(2\times 2)\times(2\times 3)$; cela implique $\omega_{\text{mul}}<\log_{12}(1000)<2{,}77989$, mais les constantes dans les $O(\cdot)$ commencent à devenir astronomiques. D'autres idées pour obtenir des bornes de plus en plus fines sur ω_{mul} sont : l'utilisation des matrices creuses [49] qui aboutit à l'inégalité asymptotique pour les sommes directes de *matrices à trous* et à la majoration $\omega_{\text{mul}}<2{,}548$; la théorie de la déformation des modules sur des groupes algébriques linéaires conduisant à la *méthode laser* de Strassen [53] et à la borne $\omega_{\text{mul}}<2{,}4785$, et qui est à la base de la borne $\omega_{\text{mul}}<2{,}3729269$ due à Vassilevska Williams [58, 59], et améliorant le précédent record dû à Coppersmith et Winograd [14]. La meilleure borne actuellement connue, $\omega_{\text{mul}}<2{,}3728639$, est due à François Le Gall [34].

Théorie vs. pratique. Théoriquement, le principal problème ouvert est $\omega_{\text{mul}}=2$. D'un point de vue pratique, il n'est pas forcément intéressant de comparer les algorithmes en fonction de leur exposant, mais plutôt de répondre à des questions du type : quelle est la formule la plus simple qui mène à un exposant faisable $\theta<\log_2(7)$? Quel est l'algorithme le plus rapide pour une taille donnée ?

L'idée des algorithmes mentionnés en Section 2.7 est due à Pan [37, 40]. L'article de Laderman, Pan et Sha [31] est le premier à avoir mis en avant leur aspect pratique ; les derniers progrès basés sur cette idée sont dûs à Kaporin [25, 26].

La recherche théorique pour améliorer les bornes sur ω_{mul} a demandé beaucoup d'ingéniosité (et de chance !), mais a mené à des algorithmes de plus en plus éloignés du calcul réel. Smith [50] adopte un point de vue pragmatique et propose d'automatiser la recherche d'algorithmes rapides en petite taille. Il ramène la recherche d'un algorithme qui calcule le produit de deux matrices de types $(a \times b)$ et $(b \times c)$ en m multiplications non-commutatives à un problème d'optimisation dans un espace à $(ab + bc + ca)m$ dimensions. Cela lui permet de retrouver tous les algorithmes connus pour les petites tailles et d'en découvrir des nouveaux. Ses programmes retrouvent les formules de Strassen en quelques secondes et celles de Laderman en quelques minutes.

Un point de vue différent est celui développé par Landsberg [33], qui montre que les questions ouvertes concernant la complexité de la multiplication matricielle se formulent convenablement en termes géométriques et théorie de la représentation.

Enfin, une autre piste actuellement explorée pour prouver (ou infirmer) que $\omega_{\text{mul}} = 2$ est celle ouverte par Cohn et Umans [12, 13]. Ils développent une approche unifiée permettant de ramener la multiplication matricielle à l'étude de la transformée de Fourier discrète dans des algèbres de groupe associées à certains groupes finis, possédant de bonnes propriétés vis-à-vis de leurs représentations irréductibles. Ils retrouvent par ce biais $\omega_{\text{mul}} < 2{,}41$ et ramènent $\omega_{\text{mul}} = 2$ à des conjectures combinatoires et en théorie des groupes.

Déterminant et permanent. Le permanent d'une matrice $A \in \mathcal{M}_n(\mathbb{K})$ est défini comme

$$\operatorname{perm}(A) = \sum_{\sigma \in \mathcal{S}_n} \prod_{i=1}^{n} a_{i,\sigma(i)}.$$

Malgré la ressemblance entre cette définition et celle de $\det(A)$, on ne connaît pas d'algorithme de complexité polynomiale en n calculant $\operatorname{perm}(A)$ sur un corps $\mathbb{K}$ de caractéristique différente de 2. Valiant [56] a prouvé que le calcul du permanent pour les matrices $n \times n$ à coefficients dans $\{0, 1\}$ est un problème NP-difficile.

Bibliographie

[1] Abdeljaoued, J. et H. Lombardi (2004). *Méthodes matricielles : introduction à la complexité algébrique*. Vol. 42. Mathématiques & Applications. Springer-Verlag, xvi+376 pages.

[2] Baur, W. et V. Strassen (1983). « The complexity of partial derivatives ». In : *Theoretical Computer Science*, vol. 22, p. 317–330.

[3] Bini, D. (1980). « Relations between exact and approximate bilinear algorithms. Applications ». In : *Calcolo*, vol. 17, n°1, p. 87–97. doi : 10.1007/BF02575865.

[4] Bini, D., M. Capovani, F. Romani et G. Lotti (1979). « $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication ». In : *Information Processing Letters*, vol. 8, n°5, p. 234–235.

[5] Bläser, Markus (2001). « A $\frac{5}{2}n^2$-lower bound for the multiplicative complexity of $n \times n$-matrix multiplication ». In : *STACS 2001 (Dresden)*. Vol. 2010. Lecture Notes in Computer Science. Springer-Verlag, p. 99–109. doi : 10.1007/3-540-44693-1_9.

[6] — (2003). « On the complexity of the multiplication of matrices of small formats ». In : *Journal of Complexity*, vol. 19, n°1, p. 43–60.

[7] Borodin, A. et I. Munro (1971). « Evaluating polynomials at many points ». In : *Information Processing Letters*, vol. 1, n°2, p. 66–68. doi : 10.1016/0020-0190(71)90009-3.

[8] Brent, R. P. et H. T. Kung (1978). « Fast algorithms for manipulating formal power series ». In : *Journal of the ACM*, vol. 25, n°4, p. 581–595.

[9] Bunch, James R. et John E. Hopcroft (1974). « Triangular factorization and inversion by fast matrix multiplication ». In : *Mathematics of Computation*, vol. 28, p. 231–236.

[10] Bürgisser, P., M. Karpinski et T. Lickteig (1991). « Some computational problems in linear algebra as hard as matrix multiplication ». In : *Computational Complexity*, vol. 1, n°2, p. 131–155. doi : 10.1007/BF01272518.

[11] Bürgisser, Peter, Michael Clausen et M. Amin Shokrollahi (1997). *Algebraic complexity theory*. Vol. 315. Grundlehren der Mathematischen Wissenschaften. Springer-Verlag, xxiv+618 pages.

[12] Cohn, H., R. Kleinberg, B. Szegedy et C. Umans (2005). « Group-theoretic Algorithms for Matrix Multiplication ». In : *FOCS'05 : IEEE Conference on Foundations of Computer Science*. IEEE Computer Society, p. 379–388. doi : 10.1109/SFCS.2005.39.

[13] Cohn, Henry et Christopher Umans (2003). « A Group-Theoretic Approach to Fast Matrix Multiplication ». In : *FOCS'03 : IEEE Conference on Foundations of Computer Science*. Washington, DC, USA : IEEE Computer Society, p. 438.

[14] Coppersmith, Don et Shmuel Winograd (1990). « Matrix multiplication via arithmetic progressions ». In : *Journal of Symbolic Computation*, vol. 9, n°3, p. 251–280.

[15] Danilevskii, A. M (1937). « The numerical solution of the secular equation ». In : *Matematicheskiĭ Sbornik*, vol. 44, n°2. (in Russian), p. 169–171.

[16] Fiduccia, Charles M. (1972). « On obtaining upper bounds on the complexity of matrix multiplication ». In : *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*. New York : Plenum, p. 31–40.

[17] Fischer, P. C. et R. L. Probert (1974). « Efficient procedures for using matrix algorithms ». In : *Automata, languages and programming*. Vol. 14. Lecture Notes in Computer Science. Springer-Verlag, p. 413–427.

[18] Fischer, Patrick C. (1974). « Further schemes for combining matrix algorithms ». In : *Automata, languages and programming*. Vol. 14. Lecture Notes in Computer Science. Springer-Verlag, p. 428–436.

[19] Giesbrecht, Mark (1995). « Nearly optimal algorithms for canonical matrix forms ». In : *SIAM Journal on Computing*, vol. 24, n°5, p. 948–969.

[20] Harter, Richard (1972). « The optimality of Winograd's formula ». In : *Communications of the ACM*, vol. 15, n°5, p. 352. doi : 10.1145/355602.361315.

[21] Hopcroft, J. E. et L. R. Kerr (1971). « On minimizing the number of multiplications necessary for matrix multiplication ». In : *SIAM Journal on Applied Mathematics*, vol. 20, p. 30–36.

[22] Hopcroft, J. et J. Musinski (1973). « Duality applied to the complexity of matrix multiplication and other bilinear forms ». In : *SIAM Journal on Computing*, vol. 2, p. 159–173.

[23] Huss-Lederman, S., E. M. Jacobson, A. Tsao, T. Turnbull et J. R. Johnson (1996). « Implementation of Strassen's algorithm for matrix multiplication ». In : *Supercomputing'96*. 32 pp. Pittsburgh, PA : IEEE Computer Society. doi : 10.1145/369028.369096.

[24] Ja'Ja', Joseph (1979). « On the complexity of bilinear forms with commutativity ». In : *STOC'79 : ACM Symposium on Theory of Computing*. Atlanta, Georgia, United States : ACM, p. 197–208. doi : 10.1145/800135.804413.

[25] Kaporin, I (2004). « The aggregation and cancellation techniques as a practical tool for faster matrix multiplication ». In : *Theoretical Computer Science*, vol. 315, n°2-3, p. 469–510.

[26] Kaporin, Igor (1999). « A practical algorithm for faster matrix multiplication ». In : *Numerical Linear Algebra with Applications*, vol. 6, n°8, p. 687–700.

[27] Keller-Gehrig, Walter (1985). « Fast algorithms for the characteristic polynomial ». In : *Theoretical Computer Science*, vol. 36, n°2-3, p. 309–317.

[28] Klyuyev, V. V. et N. I. Kokovkin-Shcherbak (1965). « On the minimization of the number of arithmetic operations for the solution of linear algebraic systems of equations ». In : *USSR Computational Mathematics and Mathematical Physics*, vol. 5, p. 25–43.

[29] Krylov, A. N. (1931). « On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined ». In : *Izvestiya Akademii Nauk SSSR*, vol. 7, n°4. (in Russian), p. 491–539.

[30] Laderman, Julian D. (1976). « A noncommutative algorithm for multiplying 3×3 matrices using 23 muliplications ». In : *Bulletin of the American Mathematical Society*, vol. 82, n°1, p. 126–128.

[31] Laderman, Julian, Victor Pan et Xuan He Sha (1992). « On practical algorithms for accelerated matrix multiplication ». In : *Linear Algebra and its Applications*, vol. 162/164, p. 557–588.

[32] Landsberg, J. M. (2006). « The border rank of the multiplication of 2×2 matrices is seven ». In : *Journal of the American Mathematical Society*, vol. 19, n°2, p. 447–459.

[33] — (2008). « Geometry and the complexity of matrix multiplication ». In : *Bulletin of the American Mathematical Society*, vol. 45, n°2, p. 247–284.

[34] Le Gall, François (2014). « Powers of Tensors and Fast Matrix Multiplication ». In : *ISSAC'14 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Katsusuke Nabeshima. New York, NY, USA : ACM, p. 296–303.

[35] Makarov, O. M. (1986). « An algorithm for multiplying 3×3 matrices ». In : *USSR Computational Mathematics and Mathematical Physics*, vol. 26, n°1, p. 179–180. doi : 10.1016/0041-5553(86)90203-X.

[36] Munro, I. (1973). « Problems related to matrix multiplication ». In : *Proceedings Courant Institute Symposium on Computational Complexity, October 1971*. Éd. par R. Rustin. New York : Algorithmics Press, p. 137–151.

[37] Pan, V. (1972). « Computation schemes for a product of matrices and for the inverse matrix ». In : *Akademiya Nauk SSSR i Moskovskoe Matematicheskoe Obshchestvo. Uspekhi Matematicheskikh Nauk*, vol. 27, n°5(167), p. 249–250.

[38] Pan, V. Ya. (1978). « Strassen's algorithm is not optimal. Trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations ». In : *SFCS '78*. Washington, DC, USA : IEEE Computer Society, p. 166–176. doi : 10.1109/SFCS.1978.34.

[39] — (1980). « New fast algorithms for matrix operations ». In : *SIAM Journal on Computing*, vol. 9, n°2, p. 321–342.

[40] — (1981). « New combinations of methods for the acceleration of matrix multiplication ». In : *Computers & Mathematics with Applications. An International Journal*, vol. 7, n°1, p. 73–125.

[41] Pan, Victor (1984a). « How can we speed up matrix multiplication ? » In : *SIAM Review*, vol. 26, n°3, p. 393–415.

[42] — (1984b). *How to multiply matrices faster*. Vol. 179. Lecture Notes in Computer Science. Springer-Verlag, xi+212 pages.

[43] Paterson, M. S. et L. J. Stockmeyer (1973). « On the Number of Nonscalar Multiplications Necessary to Evaluate Polynomials ». In : *SIAM Journal on Computing*, vol. 2, n°1, p. 60–66.

[44] Paterson, Michael S. (1975). « Complexity of monotone networks for Boolean matrix product ». In : *Theoretical Computer Science*, vol. 1, n°1, p. 13–20.
[45] Pernet, Clément et Arne Storjohann (2007). « Faster algorithms for the characteristic polynomial ». In : *ISSAC'07 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Dongming Wang. ACM Press, p. 307–314.
[46] Probert, Robert L. (1976). « On the additive complexity of matrix multiplication ». In : *SIAM Journal on Computing*, vol. 5, n°2, p. 187–203.
[47] Raz, Ran (2003). « On the complexity of matrix product ». In : *SIAM Journal on Computing*, vol. 32, n°5, p. 1356–1369. DOI : 10.1137/S0097539702402147.
[48] Schönhage, A. (1972/73). « Unitäre Transformationen grosser Matrizen ». In : *Numerische Mathematik*, vol. 20, p. 409–417.
[49] — (1981). « Partial and total matrix multiplication ». In : *SIAM Journal on Computing*, vol. 10, n°3, p. 434–455.
[50] Smith, Warren D. (2002). « Fast matrix multiplication formulae – report of the prospectors ». Preprint. URL : http://www.math.temple.edu/~wds/prospector.pdf.
[51] Storjohann, Arne (2001). « Deterministic computation of the Frobenius form (extended abstract) ». In : *FOCS'01 : IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, p. 368–377.
[52] Strassen, V. (1969). « Gaussian Elimination is Not Optimal ». In : *Numerische Mathematik*, vol. 13, p. 354–356.
[53] — (1987). « Relative bilinear complexity and matrix multiplication ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 375/376, p. 406–443.
[54] Strassen, Volker (1990). « Algebraic complexity theory ». In : *Handbook of theoretical computer science, Vol. A*. Amsterdam : Elsevier, p. 633–672.
[55] Sýkora, Ondrej (1977). « A fast non-commutative algorithm for matrix multiplication ». In : *Mathematical foundations of computer science (Proc. Sixth Sympos., Tatranská Lomnica, 1977)*. Vol. 53. Lecture Notes in Computer Science. Springer-Verlag, p. 504–512.
[56] Valiant, L. G. (1979). « The complexity of computing the permanent ». In : *Theoretical Computer Science*, vol. 8, n°2, p. 189–201.
[57] Waksman, Abraham (1970). « On Winograd's algorithm for inner products ». In : *IEEE Transactions on Computers*, vol. C-19, n°4, p. 360–361.
[58] Williams, Virginia Vassilevska (2012). « Multiplying matrices faster than Coppersmith-Winograd ». In : *STOC'12 : ACM Symposium on Theory of Computing*. New York, NY : ACM, p. 887–898. DOI : 10.1145/2213977.2214056.
[59] — (2014). « Multiplying matrices in $O(n^{2.373})$ time ». http://theory.stanford.edu/~virgi/matrixmult-f.pdf.
[60] Winograd, S. (1967). « On the number of multiplications required to compute certain functions ». In : *Proceedings of the National Academy of Sciences of the United States of America*, vol. 58, p. 1840–1842.
[61] — (1968). « A new algorithm for inner-product ». In : *IEEE Transactions on Computers*, vol. 17, p. 693–694.
[62] — (1971). « On multiplication of 2×2 matrices ». In : *Linear Algebra and Applications*, vol. 4, p. 381–388.

CHAPITRE 9

Algèbre linéaire creuse : algorithme de Wiedemann

Résumé

Les matrices creuses sont les matrices contenant beaucoup d'éléments nuls. L'algorithme de Wiedemann est une méthode itérative pour résoudre des systèmes linéaires représentés par des matrices creuses. Il ramène la résolution au calcul du polynôme minimal, lui-même reposant sur la reconnaissance d'une suite récurrente à coefficients constants.

1. Introduction

Dans ce chapitre, on aborde des questions radicalement différentes de celles du Chapitre 8 : on ne cherche plus à effectuer les produits, inversions, ... de matrices quelconques, mais à manipuler des matrices *creuses*.

On ne commencera pas par définir précisément ce qu'est une matrice creuse. L'approche consiste à donner des algorithmes dont la complexité s'exprime en fonction du nombre d'éléments non nuls des matrices en entrée. De la complexité des algorithmes « creux » va découler une borne, typiquement de l'ordre $O(n)$ pour des matrices de taille $n \times n$, sur le nombre de coefficients non nuls pour que le changement de modèle soit pertinent.

Dans tout ce qui suit, on considère donc une matrice A de taille $n \times n$ à coefficients dans un corps $\mathbb{K}$ et vérifiant les hypothèses suivantes :

— A est inversible,

— A contient s éléments non nuls.

Le produit de A par un vecteur peut donc s'effectuer en $O(s)$ opérations dans $\mathbb{K}$.

On va décrire un algorithme dû à Wiedemann qui permet de calculer l'unique solution du système $Ax = y$ avec une complexité en $O(ns)$; l'algorithme original est plus général que celui présenté ici, en ce qu'il permet de traiter les matrices rectangulaires ou carrées et non inversibles, mais il se ramène au cas carré inversible.

Remarquons que si s est de l'ordre de n^2, caractérisant donc des matrices plutôt denses, on retombe dans une complexité de l'ordre de $O(n^3)$. Le cas intéressant est celui où s est de l'ordre de n, auquel cas l'algorithme est quadratique en n : pour simplifier, on retiendra que l'exposant de l'algèbre linéaire creuse est 2 dans les bons cas.

2. Polynôme minimal et résolution de systèmes

L'algorithme de Wiedemann passe par le calcul du *polynôme minimal* de A. Rappelons sa définition.

Il est possible d'associer à tout polynôme en une variable $P = \sum_i p_i X^i$ de $\mathbb{K}[X]$ la matrice $P(A) = \sum_i p_i A^i$. Le théorème de Cayley-Hamilton affirme que le polynôme caractéristique χ_A de A annule la matrice A, c'est-à-dire que $\chi_A(A) = 0$.

Definition 1. Le polynôme minimal de A est le polynôme unitaire de plus petit degré annulant A.

Algorithme de Wiedemann

Entrée : Une matrice creuse $A \in \mathcal{M}_n(\mathbb{K})$.

Sortie : Son polynôme minimal $\mu_A(X)$.

1. Choisir des vecteurs u et v aléatoires dans $\mathbb{K}^n$.
2. Pour $0 \leq i \leq 2n$, calculer les vecteurs $v_i = A^i v$, puis les scalaires $N_i = {}^t u\, v_i$.
3. Renvoyer le polynôme minimal de la suite (N_i).

FIGURE 1. Algorithme de Wiedemann pour calculer le polynôme minimal d'une matrice creuse.

Soit $\mu_A(X)$ le polynôme minimal de A, supposé connu. Puisqu'on a supposé A inversible, le terme constant de $\mu_A(X)$ n'est pas nul : il divise le terme constant du polynôme caractéristique, qui n'est lui-même pas nul.

L'égalité $\mu_A(A) = 0$ se réécrit sous la forme

$$A^{-1} = \frac{-1}{p_0}\left(A^{m-1} + p_{m-1}A^{m-2} + \cdots + p_1 I_n\right).$$

Pour résoudre le système $Ax = b$, $(b \in \mathbb{K}^n)$, on va calculer $x = A^{-1}b$, c'est-à-dire

$$\frac{-1}{p_0}\left(A^{m-1}b + p_{m-1}A^{m-2}b + \cdots + p_1 b\right).$$

Ainsi, il suffit de calculer les itérés $b, Ab, \ldots, A^{m-1}b$, puis d'additionner les vecteurs $p_1 b, p_2 Ab, \ldots, A^{m-1}b$, pour retrouver x. L'analyse de complexité est immédiate :

— Chacun des $A^i b$ s'obtient à partir de $A^{i-1}b$ en $O(s)$ opérations.

— Chacun des produits par p_i, et chaque addition, se fait en $O(n)$ opérations. Remarquer que $n \leq s$ (hypothèse d'inversibilité de A).

Au total, on a donc $O(ns)$ opérations à faire pour résoudre le système, si on connaît le polynôme minimal de A.

3. Calcul du polynôme minimal

Écrivons le polynôme minimal de A sous la forme

$$\mu_A(X) = X^m + p_{m-1}X^{m-1} + \cdots + p_0.$$

Alors la suite des puissances de A vérifie la récurrence linéaire

$$A^{k+m} + p_{m-1}A^{k+m-1} + \cdots + p_0 A^k = 0,$$

et ne vérifie aucune récurrence d'ordre plus petit.

Pour tous vecteurs u et v, la suite (de nombres) $N_i := {}^t u A^i v$ vérifie donc

$$N_{k+m} + p_{m-1}N_{k+m-1} + \cdots + p_0 N_k = 0.$$

Si u et v sont mal choisis (nuls, par exemple), la suite N_i vérifie une récurrence d'ordre plus petit que m. La proposition suivante montre qu'en choisissant u et v au hasard, on tombe vraisemblablement sur une suite N_i qui ne satisfait pas de récurrence d'ordre plus petit.

PROPOSITION 1. *Il existe un polynôme $D \in \mathbb{K}[U_1, \ldots, U_n, V_1, \ldots, V_n]$, non nul et de degré au plus $2n$, tel que si $D(u, v)$ est non nul, la suite N_i associée à u et v ne satisfait pas de récurrence d'ordre plus petit que m.*

L'idée est de choisir u et v au hasard. Si on n'a pas de chance, $D(u, v)$ est nul ; sinon, la suite des N_i associée à u et v permet de retrouver $\mu_A(X)$ grâce à l'algorithme de Berlekamp-Massey décrit en page 108 (attention, on ne connaît pas explicitement le polynôme D, mais son existence assure que la plupart des choix de u et v sont « chanceux »).

L'algorithme est donné en Figure 1. Il est probabiliste randomisé. L'analyse de probabilité est facile à faire en utilisant les résultats de l'exercice 1 ci-dessous.

La complexité de l'étape 2 est de $O(n)$ produits matrice-vecteur ou vecteur-vecteur, chacun prenant au pire $O(s)$ opérations ; au total on obtient donc $O(ns)$ opérations pour cette phase. L'algorithme d'approximants de Padé est négligeable, puisqu'il ne demande que $O(n^2) = O(sn)$ opérations.

4. Calcul du déterminant

L'algorithme en Figure 1 permet de détecter en cours de route l'inversibilité de la matrice A. En effet, si l'approximant de Padé calculé par l'algorithme de Berlekamp-Massey (page 108) a un dénominateur divisible par X, cela veut dire que $\det(A) = 0$.

Il est possible de calculer en bonne complexité le déterminant de la matrice creuse A, en exploitant le résultat suivant d'algèbre linéaire.

Lemme 1. *Si tous les mineurs principaux de $A \in \mathcal{M}_n(\mathbb{K})$ sont non nuls, alors les polynômes minimal et caractéristique de la matrice $B = A \cdot \mathrm{Diag}(X_1, \ldots, X_n)$ coïncident.*

L'algorithme qui s'en déduit est le suivant :

- Choisir une matrice de permutations aléatoire P, pour assurer que tous les mineurs principaux de la matrice AP sont inversibles ;
- choisir des éléments $x_1, \ldots, x_n$ aléatoirement dans $\mathbb{K}$;
- calculer le polynôme minimal μ_B de $B = A \cdot P \cdot \mathrm{Diag}(x_1, \ldots, x_n)$;
- renvoyer $\mu_B(0)/(x_1 \cdots x_n) \cdot \mathrm{sgn}(P)$.

À nouveau, la complexité de cet algorithme est quadratique en la taille n de A.

5. Calcul du rang

Le rang maximal de la matrice creuse A est détecté par les algorithmes précédents. Quitte à multiplier A par une matrice de permutations aléatoires, on peut donc supposer que son rang r est strictement inférieur à n, et que ses mineurs principaux A_i, pour $1 \leq i \leq r$, sont tous non nuls.

Sous ces hypothèses, il est possible de montrer que, pour un choix aléatoire d'éléments $x_1, \ldots, x_n$ dans $\mathbb{K}$, le rang r est égal à $\deg(\mu_{AD}) - 1$, où D est la matrice diagonale $D = \mathrm{Diag}(x_1, \ldots, x_n)$. L'algorithme qui s'ensuit calcule le polynôme minimal de la matrice creuse AD, et en déduit le rang de A, le tout pour une complexité quadratique en n.

Exercices

Exercice 1. Soit A une matrice dans $\mathcal{M}_n(\mathbb{K})$, soit b un vecteur de $\mathbb{K}^n \setminus \{0\}$ et soit f le polynôme minimal de la suite $(A^i \cdot b)_{i \geq 0}$.
Le but de l'exercice est d'estimer la probabilité $\mathcal{P}$ que f coïncide avec le polynôme minimal de la suite $({}^t u \cdot A^i \cdot b)_{i \geq 0}$ lorsque u est un vecteur de $\mathbb{K}^n$ dont les coordonnées sont choisies aléatoirement au hasard dans un sous-ensemble fini U de $\mathbb{K}$.

1. Montrer qu'il existe une application $\psi : \mathbb{K}^n \to \mathbb{A} = \mathbb{K}[X]/(f)$, $\mathbb{K}$-linéaire et surjective, telle que pour tout $u \in \mathbb{K}^n$ on ait

f est le polynôme minimal de $({}^t u \cdot A^i \cdot b)_{i\geq 0} \iff \psi(u)$ est inversible dans $\mathbb{A}$.

   [Indication : l'application $\phi : \mathbb{K}^n \to \mathbb{K}^{\mathbb{N}}$ définie par $\phi(u) = ({}^t u \cdot A^i \cdot b)_{i\geq 0}$ induit une application linéaire surjective $\mathbb{K}^n \to M_f$, où M_f est l'espace vectoriel des suites de $\mathbb{K}^{\mathbb{N}}$ admettant f comme polynôme annulateur. Par ailleurs, $\mathbb{A}$ et M_f sont isomorphes en tant qu'espaces vectoriels.]

2. Soit d le degré de f. Montrer qu'il existe un polynôme non identiquement nul $R \in \mathbb{K}[X_1, \ldots, X_n]$ de degré total au plus d tel que pour tout vecteur $u = (u_1, \ldots, u_n) \in \mathbb{K}^n$ on ait

   $\psi(u)$ est inversible dans $\mathbb{A}$ si et seulement si $R(u_1, \ldots, u_n) \neq 0$.

   [Indication : utiliser un résultant.]

3. Montrer que R admet au plus $d \cdot |U|^{n-1}$ racines dans U^n. En déduire que la probabilité qu'un élément de U^n dont les coordonnées sont choisies aléatoirement au hasard dans U soit racine de R est bornée par $d/|U|$.

4. Conclure que la probabilité $\mathcal{P}$ vérifie

$$\mathcal{P} \geq 1 - \frac{d}{|U|}.$$

Notes

L'algorithme de Wiedemann a été proposé dans [11]. Cet article contient également l'algorithme esquissé en Section 4. L'algorithme en Section 5 provient de [5]. Des versions « par blocs » de l'algorithme de Wiedemann ont été proposées dans [1, 8, 9, 10]. D'autres généralisations sont traités dans [3, 4, 6].

Les questions (2) et (3) de l'exercice 1 forment le cœur du « lemme de Zippel-Schwartz » [2, 7, 12].

Bibliographie

[1] Coppersmith, Don (1994). « Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm ». In : *Mathematics of Computation*, vol. 62, n°205, p. 333–350.

[2] DeMillo, Richard A. et Richard J. Lipton (1978). « A probabilistic remark on algebraic program testing ». In : *Information Processing Letters*, vol. 7, n°4, p. 193–195.

[3] Giesbrecht, M., A. Lobo et B. D. Saunders (1998). « Certifying inconsistency of sparse linear systems ». In : *ISSAC'98 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 113–119.

[4] Giesbrecht, Mark (2001). « Fast computation of the Smith form of a sparse integer matrix ». In : *Computational Complexity*, vol. 10, n°1, p. 41–69.

[5] Kaltofen, Erich et B. David Saunders (1991). « On Wiedemann's method of solving sparse linear systems ». In : *Applied algebra, algebraic algorithms and error-correcting codes (New Orleans, LA, 1991)*. Vol. 539. Lecture Notes in Computer Science. Springer-Verlag, p. 29–38.

[6] Mulders, Thom (2004). « Certified sparse linear system solving ». In : *Journal of Symbolic Computation*, vol. 38, n°5, p. 1343–1373.

[7] Schwartz, J. T. (1980). « Fast probabilistic algorithms for verification of polynomial identities ». In : *Journal of the Association for Computing Machinery*, vol. 27, n°4, p. 701–717.

[8] Thomé, E. (2002). « Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm ». In : *Journal of Symbolic Computation*, vol. 33, n°5, p. 757–775.

[9] Turner, William J. (2006). « A block Wiedemann rank algorithm ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 332–339.

[10] Villard, G. (1997). « Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems ». In : *ISSAC'97 : International Symposium on Symbolic and Algebraic Computation*. Kihei, Maui, Hawaii, United States : ACM Press, p. 32–39. doi : 10.1145/258726.258742.

[11] Wiedemann, D. (1986). « Solving sparse linear equations over finite fields ». In : *IEEE Transactions on Information Theory*, vol. IT-32, p. 54–62.

[12] Zippel, Richard (1979). « Probabilistic algorithms for sparse polynomials ». In : *EUROSAM'79*. Vol. 72. Lecture Notes in Computer Science. Springer-Verlag, p. 216–226.

CHAPITRE 10

Algèbre linéaire structurée

Résumé

Les matrices possédant une structure spéciale sont omniprésentes en calcul formel. Ce sont des matrices dont les éléments jouissent d'une certaine répétition, ou satisfont à certaines relations. Plus formellement, une matrice structurée est typiquement définie par $O(n)$ éléments, au lieu de n^2 pour une matrice dense, et peut être multipliée par un vecteur en $\tilde{O}(n)$ opérations arithmétiques, au lieu de $O(n^2)$ opérations pour une matrice dense. Ce chapitre présente une algorithmique unifiée pour manipuler des matrices structurées denses, comme les matrices de Toeplitz, Hankel, Vandermonde et Sylvester. Les calculs avec les matrices de ces classes sont liés aux calculs avec les polynômes, ce qui permet l'emploi des techniques de multiplication polynomiale rapide pour accélérer leur manipulation. Par exemple, on peut résoudre un système linéaire défini par une matrice structurée inversible en $n \times n$ en $\tilde{O}(n)$ opérations.

1. Introduction

Nous avons vu au Chapitre 8 qu'il est possible de manipuler les matrices denses de taille $n \times n$ à coefficients dans un corps $\mathbb{K}$ en $\mathsf{MM}(n) = O(n^\theta)$ opérations dans $\mathbb{K}$, où θ est un réel compris entre 2 et 3. Ici, par manipuler, on entend multiplier, inverser, calculer le déterminant, ou encore résoudre un système linéaire.

Dans certaines situations, les matrices qu'on est amené à manipuler présentent une structure que ces algorithmes généraux ne savent pas capturer et exploiter. Voici quelques exemples de matrices qui possèdent une structure.

Definition 1. Une matrice $A \in \mathcal{M}_n(\mathbb{K})$ est dite *de Toeplitz* si elle est invariante le long des diagonales, c'est-à-dire si ses éléments $a_{i,j}$ vérifient $a_{i,j} = a_{i+k,j+k}$ pour tout k.

Une telle matrice est complètement définie par sa première ligne et sa première colonne.

Definition 2. Une matrice $A \in \mathcal{M}_n(\mathbb{K})$ est dite *de Hankel* si elle est invariante le long des anti-diagonales, c'est-à-dire si ses éléments $a_{i,j}$ vérifient $a_{i,j} = a_{i-k,j+k}$ pour tout k.

Une telle matrice est complètement définie par sa première ligne et sa dernière colonne.

Exemple 1. La matrice de la multiplication polynomiale en degré fixé (dans les bases canoniques) par un polynôme fixé est de Toeplitz. Par exemple, la multiplication d'un polynôme de degré au plus 2 par le polynôme fixé $a_0 + a_1X + a_2X^2$

de $\mathbb{K}[X]$ se traduit matriciellement par l'égalité

$$\begin{bmatrix} a_0 & 0 & 0 \\ a_1 & a_0 & 0 \\ a_2 & a_1 & a_0 \\ 0 & a_2 & a_1 \\ 0 & 0 & a_2 \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} a_0b_0 \\ a_0b_1 + a_1b_0 \\ a_0b_2 + a_1b_1 + a_2b_0 \\ a_1b_2 + a_2b_1 \\ a_2b_2 \end{bmatrix}.$$

Dans cet exemple, la matrice Toeplitz est d'une forme particulière, appelée *bande*. En fait, toute matrice de Toeplitz peut être vue comme sous-matrice d'une matrice de Toeplitz bande, et cela entraîne le résultat suivant.

LEMME 1. *Le produit d'une matrice de Toeplitz (ou de Hankel) de $\mathcal{M}_n(\mathbb{K})$ par un vecteur de $\mathbb{K}^n$ peut s'effectuer en $O(\mathsf{M}(n))$ opérations.*

DÉMONSTRATION. Pour tout $0 \leq i \leq n-1$, l'élément c_i du produit matrice-vecteur

$$\begin{bmatrix} a_{n-1} & \cdots & a_0 \\ & \ddots & \\ a_{2n-2} & \cdots & a_{n-1} \end{bmatrix} \times \begin{bmatrix} b_0 \\ \vdots \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix}$$

est le coefficient de X^{n-1+i} dans le produit de polynômes

$$(b_0 + \cdots + b_{n-1}X^{n-1})(a_0 + \cdots + a_{2n-2}X^{2n-2}). \tag{1}$$

La preuve est tout à fait similaire pour une matrice de Hankel. □

REMARQUE. La preuve précédente montre que $2\,\mathsf{M}(n) + O(n)$ opérations suffisent pour multiplier une matrice Toeplitz ou Hankel de taille n par un vecteur. Cette borne peut être améliorée à $\mathsf{M}(n) + O(n)$, en observant que l'extraction de la *partie médiane* du produit (1) est suffisante, et en employant des algorithmes pour le produit médian, évoqués au Chapitre 12.

EXEMPLE 2. Une matrice de Sylvester est la concaténation de deux matrices Toeplitz bande.

DEFINITION 3. Une matrice $A = (a_{i,j})_{i,j=0}^{n-1}$ de $\mathcal{M}_n(\mathbb{K})$ est dite *de Vandermonde* si ses éléments s'écrivent $a_{i,j} = a_i^j$ pour $a_0, \ldots, a_{n-1} \in \mathbb{K}$ avec $a_i \neq a_j$ pour $i \neq j$.

DEFINITION 4. Une matrice $A = (a_{i,j})_{i,j=0}^{n-1}$ de $\mathcal{M}_n(\mathbb{K})$ est dite *de Cauchy* si ses éléments s'écrivent $a_{i,j} = 1/(a_i - b_j)$ pour $a_i, b_j \in \mathbb{K}$ avec $a_i \neq b_j$ pour tous i, j.

Il y a un certain nombre de points en commun entre ces exemples : la matrice est représentable par $O(n)$ éléments ; le produit matrice-vecteur peut s'effectuer plus rapidement que dans le cas générique, en $O(\mathsf{M}(n)\log n)$ opérations au lieu de $O(n^2)$; le produit par une matrice quelconque peut s'effectuer en complexité $O(n\,\mathsf{M}(n)\log n)$ —quasi-optimale en la taille de la sortie pour une multiplication polynomiale à base de FFT. En effet, dans chaque cas, le produit matrice-vecteur Av admet une interprétation analytique :

- multiplication polynomiale dans les cas Toeplitz, Hankel et Sylvester ;
- évaluation polynomiale multipoint dans le cas d'une matrice de Vandermonde (Chapitre 5) ;
- évaluation multipoint de fractions rationnelles de la forme $\sum_{j=1}^{n-1} c_j/(X - b_j)$.

EXERCICE 1. Montrer que le produit d'une matrice de Cauchy par un vecteur peut s'effectuer en $O(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$.

Au vu de ces exemples, on pourrait donc être tenté de définir comme *structurée* une matrice telle que son produit par un vecteur peut s'effectuer en $\tilde{O}(n)$ opérations. La question qui se pose naturellement est : peut-on exploiter cette définition de la structure de A pour résoudre le système $Ax = b$? Un premier constat positif est que, dans chacun des exemples précédents, la résolution admet aussi une interprétation analytique :

(i) devinette de récurrences à coefficients constants, si A est de Toeplitz ou de Hankel ;

(ii) interpolation polynomiale, si A est de Vandermonde ;

(iii) interpolation de fractions rationnelles, si A est de Cauchy.

En effet, si la suite (a_n) d'éléments de $\mathbb{K}$ vérifie une récurrence (inconnue) à coefficients constants de la forme

$$a_{n+d} = p_{d-1}a_{n+d-1} + \cdots + p_0 a_n, \qquad n \geq 0,$$

alors trouver les coefficients p_i de cette récurrence revient à résoudre le système de Hankel

$$\begin{bmatrix} a_0 & a_1 & \cdots & a_{d-1} \\ a_1 & a_2 & \iddots & a_d \\ \vdots & \iddots & \iddots & \vdots \\ a_{d-1} & a_d & \cdots & a_{2d-2} \end{bmatrix} \times \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{d-1} \end{bmatrix} = \begin{bmatrix} a_d \\ a_{d+1} \\ \vdots \\ a_{2d-1} \end{bmatrix}.$$

Un autre point encourageant est que, pour ces trois opérations, on dispose d'algorithmes de complexité quasi-optimale $O(\mathsf{M}(n)\log n)$, présentés au Chapitre 5 pour (ii), et au Chapitre 7 pour (i) et (iii).

Il y a cependant quelques points négatifs : un premier est que si A est une matrice inversible telle que l'application linéaire $v \mapsto Av$ se calcule en L opérations, cela n'implique pas l'existence d'un algorithme de complexité $O(L)$ pour l'application $v \mapsto A^{-1}v$. En d'autres termes, on manque d'un *principe d'inversion* analogue au *principe de transposition* présenté au Chapitre 12. Par exemple, pour les matrices creuses, le meilleur algorithme de résolution, dû à Wiedemann, est de complexité quadratique en n (Chapitre 9). Un second point négatif est que les classes vues plus haut ne sont pas stables par inversion : l'inverse d'une matrice de Toeplitz (resp. de Vandermonde) n'est pas de Toeplitz (resp. de Vandermonde).

On a donc besoin de rajouter des hypothèses dans la définition de la *bonne notion* de *matrice structurée*. Cette définition exploite la généralisation du caractère invariant par diagonale d'une matrice de Toeplitz, et vient de l'observation simple suivante : si A est de Toeplitz, alors la matrice

$$\phi(A) = A - (A \text{ décalée de 1 vers le bas et de 1 vers la droite})$$

admet une ligne et une colonne non nulles ; elle est donc de rang borné par 2. On dit que ϕ est un *opérateur de déplacement* et que *le rang de ϕ-déplacement* de A est au plus 2. On peut donc représenter $\phi(A)$ sous forme compacte, comme un produit $G \cdot {}^tH$, avec G et H des matrices rectangulaires de taille $n \times 2$.

Exemple 3. Si A est la matrice 3×3

$$A = \begin{bmatrix} c & d & e \\ b & c & d \\ a & b & c \end{bmatrix},$$

avec $d \neq 0$, alors $\phi(A)$ s'écrit

$$\phi(A) = \begin{bmatrix} c & d & e \\ b & 0 & 0 \\ a & 0 & 0 \end{bmatrix} = \begin{bmatrix} c & d \\ b & 0 \\ a & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & e/d \end{bmatrix}.$$

DEFINITION 5. Les matrices (G, H) de $\mathcal{M}_{n,2}(\mathbb{K})$ telles que $\phi(A) = G \cdot {}^tH$ sont appelées *générateurs de déplacement* pour la matrice de Toeplitz A.

Ces définitions s'étendent, et permettent d'introduire le concept de matrice *quasi-Toeplitz*.

DEFINITION 6. On appelle *opérateur de déplacement* ϕ_+ l'application $A \mapsto A - Z \cdot A \cdot {}^tZ$, où la matrice Z définie par

$$Z = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & \ddots & \dots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix}$$

est telle que ZA est la matrice A décalée de 1 ligne vers le bas, et $A \cdot {}^tZ$ est la matrice A décalée de 1 colonne vers la droite. Le *rang de déplacement* de A est l'entier $\alpha_+(A) = \text{rang}(\phi_+(A))$. On appelle *générateurs de déplacement* pour l'opérateur ϕ_+ un couple (G, H) de matrices de taille $n \times \alpha$ vérifiant l'égalité $\phi_+(A) = G \cdot {}^tH$. Si $\alpha_+(A) \ll n$, on dit que A est *quasi-Toeplitz*.

Intuitivement, le rang de déplacement mesure à quel point la matrice A est loin d'être de Toeplitz.

EXEMPLE 4. Les matrices de Toeplitz et Sylvester sont quasi-Toeplitz, de rang de déplacement 2.

Ces définitions s'étendent et permettent également de définir des matrices *quasi-Vandermonde* et *quasi-Cauchy*, mais pour des opérateurs de déplacement différents :

- $\mathbf{V_a} \mapsto \mathbf{V_a} - \text{Diag}_\mathbf{a} \cdot \mathbf{V_a} \cdot {}^tZ$ pour le cas Vandermonde ;
- $\mathbf{C_{a,b}} \mapsto \mathbf{C_{a,b}} - \text{Diag}_\mathbf{a}{}^{-1} \cdot \mathbf{C_{a,b}} \cdot \text{Diag}_\mathbf{b}$ pour le cas Cauchy.

Ici, pour deux vecteurs $\mathbf{a} = (a_0, \dots, a_{n-1})$ et $\mathbf{b} = (b_0, \dots, b_{n-1})$, on désigne par $\mathbf{V_a}$ la matrice de Vandermonde $(a_i^j)_{i,i=0}^{n-1}$, par $\text{Diag}_\mathbf{a}$ la matrice diagonale, dont la diagonale porte le vecteur $\mathbf{a}$, et par $\mathbf{C_{a,b}}$ la matrice de Cauchy $(1/(a_i - b_j))_{i,j=0}^{n-1}$.

Dans tous ces cas, $\phi_{M,N}$ est définie par $\phi_{M,N}(A) = A - MAN$, avec M et N bien choisies (en fonction de la structure visée), et le rang de déplacement de A est défini comme étant le rang de la matrice $\phi_{M,N}(A)$.

EXERCICE 2. Estimer les rangs de déplacement des matrices de Vandermonde et de Cauchy, pour les opérateurs de déplacement définis ci-dessus.

Résultat principal. L'idée-clé derrière les algorithmes rapides pour les matrices structurées de rang de déplacement α est l'utilisation des générateurs de déplacement comme structure de données compacte, de taille $O(\alpha n)$, donc proportionnelle à la taille n de la matrice, dans le cas où $\alpha \ll n$. L'énoncé suivant contient le résultat principal de ce chapitre.

THÉORÈME 1. *Soit $\phi(\cdot)$ l'un des opérateurs Toeplitz, Vandermonde, Cauchy définis ci-dessus. Soit A une matrice de $\mathcal{M}_n(\mathbb{K})$ donnée par des générateurs de déplacement de taille $n \times \alpha$, et soit b un vecteur de $\mathbb{K}^n$. Alors, il est possible de :*

1. *calculer le déterminant de A ;*
2. *calculer le rang de A ;*
3. *calculer une solution du système $Ax = b$, ou prouver qu'il n'y en a aucune*

en $\tilde{O}(\alpha^2 n)$ opérations dans $\mathbb{K}$. Plus exactement, cette complexité s'exprime en termes de la fonction de multiplication polynomiale, et vaut :

1. $O(\alpha^2\,\mathsf{M}(n)\log n)$ *dans le cas quasi-Toeplitz ;*
2. $O(\alpha^2\,\mathsf{M}(n)\log^2(n))$ *dans les cas quasi-Vandermonde et quasi-Cauchy.*

Comme conséquence du Théorème 1 nous obtenons une *algorithmique unifiée* pour résoudre en complexité quasi-optimale différents problèmes sur les polynômes et les séries.

Corollaire 1. *On peut calculer en* $O(\mathsf{M}(n)\log(n))$ *opérations arithmétiques :*

1. *le pgcd étendu et le résultant de deux polynômes de degré borné par* n *;*
2. *un approximant de Padé de type* (n, n) *d'une série tronquée donnée à précision* $2n$ *;*

Esquisse de démonstration. Le résultant de deux polynômes $A, B \in \mathbb{K}[X]$ s'obtient comme le déterminant de leur matrice de Sylvester $\mathsf{Syl}(A, B)$, qui a un rang de déplacement au plus 2. Le degré du pgcd $G = \mathrm{pgcd}(A, B)$ s'obtient par un calcul de rang de la matrice de Sylvester : $\deg(G) = \deg(A) + \deg(B) - \mathrm{rang}(\mathsf{Syl}(A, B))$. Une fois connu le degré du pgcd, la relation de Bézout $UA + VB = G$, avec les contraintes $\deg(U) < \deg(B) - \deg(G)$ et $\deg(V) < \deg(A) - \deg(G)$, se traduit en un système linéaire en les coefficients de U et de V, dont la matrice est quasi-Toeplitz, de rang de déplacement au plus 3. Des considérations similaires s'appliquent pour le calcul d'approximants de Padé. □

Corollaire 2. *Étant données* n *series* $f_1, \ldots, f_n$ *de* $\mathbb{K}[[X]]$ *connues à précision* $\sigma = \sum_i (d_i+1)-1$*, il est possible d'en calculer un approximant de Padé-Hermite* $(p_1, \ldots, p_n)$ *de type* $(d_1, \ldots, d_n)$ *en* $O(n^2\,\mathsf{M}(\sigma)\log(\sigma))$ *opérations dans* $\mathbb{K}$*.*

Esquisse de démonstration. Il s'agit d'un problème linéaire en les coefficients de l'approximant cherché. La matrice du système linéaire est quasi-Toeplitz, de rang de déplacement borné par n. □

Dans la suite, nous allons esquisser les grandes lignes de la preuve du Théorème 1 dans le cas particulier où la matrice A est inversible et « suffisamment générique » (tous les mineurs non nuls), et uniquement dans le cas quasi-Toeplitz.

2. Le cas quasi-Toeplitz

Dans le reste du chapitre, nous allons nous concentrer uniquement sur le cas quasi-Toeplitz, car il contient les principales idées algorithmiques, et il couvre beaucoup d'applications ; les cas quasi-Vandermonde et quasi-Cauchy s'y ramènent et sont plus techniques.

Nous allons montrer que la notion de matrice quasi-Toeplitz est une bonne notion de structure, car elle répond aux propriétés suivantes :

(P1) on peut effectuer quasi-optimalement le produit d'une matrice quasi-Toeplitz par un vecteur ;

(P2) la somme et le produit de deux matrices quasi-Toeplitz restent quasi-Toeplitz ;

(P3) l'inverse aussi.

Or, ces propriétés forment le minimum nécessaire pour concevoir un algorithme de type *inversion de Strassen* (page 135) dans la représentation compacte par générateurs de déplacement.

2.1. Produit matrice-vecteur, cas quasi-Toeplitz. Pour montrer la propriété (P1) ci-dessus, le point clé est le résultat suivant.

PROPOSITION 1. (*formule* ΣLU) *L'opérateur* $\phi_+ : A \mapsto A - Z \cdot A \cdot {}^tZ$ *est inversible. Plus exactement, on a la formule suivante, appelée* la formule ΣLU :

$$A - Z \cdot A \cdot {}^tZ = \sum_{i=1}^{\alpha} x_i \cdot {}^ty_i \quad \textit{si et seulement si} \quad A = \sum_{i=1}^{\alpha} L(x_i) \cdot U(y_i),$$

où les x_i *(resp.* y_i*) sont les colonnes du générateur* G *(resp.* H*), et où, pour un vecteur colonne* $v = {}^t[v_0 \cdots v_{n-1}]$*, on note* $L(v)$ *la matrice Toeplitz inférieure*

$$\begin{bmatrix} v_0 & 0 & \dots & 0 \\ v_1 & v_0 & \dots & 0 \\ \vdots & \ddots & \dots & \vdots \\ v_{n-1} & \cdots & v_1 & v_0 \end{bmatrix}$$

et $U(v)$ *la matrice Toeplitz supérieure* ${}^tL(v)$.

DÉMONSTRATION. Par linéarité, il suffit de traiter le cas $\alpha = 1$. Si $C = L(a)U(b)$, alors un calcul immédiat montre que $c_{i+1,j+1} = a_ib_j + a_{i-1}b_{j-1} + \cdots$, et donc $c_{i+1,j+1} - c_{i,j} = a_ib_j$ et $\phi_+(C) = (a_ib_j)_{i,j=0}^{n-1} = a \cdot {}^tb$.

L'implication inverse découle de l'injectivité de ϕ_+ et est laissée en exercice. □

La Proposition 1 permet de donner une définition équivalente pour le rang de déplacement.

DEFINITION 7. $\alpha_+(A)$ est le plus petit entier α tel qu'il existe une décomposition de la forme

$$A = \sum_{i=1}^{\alpha} L_iU_i$$

pour des matrices L_i de Toeplitz inférieures, et U_i de Toeplitz supérieures.

EXEMPLE 5. Si A est de Toeplitz, alors elle admet la décomposition $A = A_{\text{inf}} \cdot I_n + I_n \cdot A_{\text{sup}}$, donc $\alpha_+(A) \leq 2$.

La définition 7 permet de prouver la propriété (P1) en page 157.

COROLLAIRE 3. *Si* A *est donnée en représentation compacte par une paire de générateurs* (G, H) *de taille* $n \times \alpha$*, alors le produit matrice-vecteur* Av *peut s'effectuer en* $O(\alpha \mathsf{M}(n))$ *opérations arithmétiques.*

DÉMONSTRATION. Le produit Av s'écrit comme la somme des α termes de la forme $L(x_i)(U(y_i)v)$. Or, chacun de ces termes peut être calculé en $O(\mathsf{M}(n))$ opérations, grâce au lemme 1. □

2.2. Addition et produit en représentation compacte par générateurs. Pour justifier la propriété (P2) en page 157, on va montrer plus, à savoir que la représentation par générateurs permet un calcul efficace (en temps quasi-linéaire) des opérations d'additions et de multiplication de deux matrices quasi-Toeplitz.

PROPOSITION 2 (opérations matricielles en représentation compacte). *Soient* (T, U) *générateurs de déplacement de taille* $n \times \alpha$ *pour* A*, et* (G, H) *générateurs de déplacement de taille* $n \times \beta$ *pour* B*. Alors*

1. $([T \mid G], [U \mid H])$ *sont des générateurs de déplacement pour* $A + B$ *de longueur* $\alpha + \beta$ *;*
2. $([T \mid W \mid a], [V \mid H \mid -b])$ *sont des générateurs de déplacement pour* AB*, de longueur* $\alpha + \beta + 1$*,*

où $V := {}^tB \cdot U$, $W := Z \cdot A \cdot {}^tZ \cdot G$, *et où le vecteur* a *(resp.* b*) est la dernière colonne de* $Z \cdot A$ *(resp. de* $Z \cdot {}^tB$*).*

La preuve, basée sur une vérification immédiate, est laissée en exercice.

Corollaire 4. *En représentation compacte par générateurs de déplacement, de longueurs au plus* α *pour* A *et* B*, on peut calculer*

1. *la somme* $A + B$ *en* $O(\alpha n)$ *opérations dans* $\mathbb{K}$*;*
2. *le produit* AB *en* $\mathsf{Mul}(n, \alpha) = O(\alpha^2\, \mathsf{M}(n))$ *opérations dans* $\mathbb{K}$*.*

Démonstration. Le seul point non trivial est le calcul des matrices V, W, a et b. Tout repose sur la formule ΣLU. Si B s'écrit $\sum_{i=1}^{\beta} L(x_i)U(y_i)$, alors sa transposée s'écrit ${}^tB = \sum_{i=1}^{\beta} L(y_i)U(x_i)$, et le calcul de $V = {}^tB \cdot U$ se ramène à $\alpha\beta$ multiplications polynomiales, chacune pouvant s'effectuer en $O(\mathsf{M}(n))$. Le même raisonnement s'applique au calcul de W. Enfin, le calcul de a revient à multiplier A par la colonne ${}^t[0, \ldots, 0, 1]$ et une observation similaire pour b permet de conclure. □

2.3. Inversion en représentation compacte par générateurs. Pour prouver la propriété (P3) en page 157, il est commode d'introduire un *opérateur dual de déplacement.*

Definition 8. L'opérateur de ϕ_- déplacement d'une matrice A de $\mathcal{M}_n(\mathbb{K})$ est défini par l'égalité

$$\phi_-(A) = A - {}^tZ \cdot A \cdot Z = A - (A \quad \text{décalée de 1 vers le haut et vers la gauche}).$$

Le $(-)$ rang de déplacement α_- est défini par la formule

$$\alpha_-(A) = \operatorname{rang}(A - {}^tZ \cdot A \cdot Z).$$

On peut montrer (de la même façon que pour ϕ_+) qu'on dispose d'une définition équivalente pour ϕ_-, et que ϕ_+ et ϕ_- sont liés.

Lemme 2.

1. $\alpha_-(A)$ *est le plus petit entier* α *tel que* A *puisse s'écrire sous la forme* $\sum_{=1}^{\alpha} U_i L_i$*, avec* L_i *Toeplitz inférieures, et* U_i *Toeplitz supérieures.*
2. *On a la* formule ΣUL :

$$A - {}^tZ \cdot A \cdot Z = \sum_{i=1}^{\alpha} x_i \cdot {}^t y_i \quad \textit{si et seulement si} \quad A = \sum_{i=1}^{\alpha} U(\operatorname{rev}(x_i)) \cdot L(\operatorname{rev}(y_i)).$$

   Ici, pour $v = {}^t[v_0, \ldots, v_{n-1}]$*, on note* $\operatorname{rev}(v) = {}^t[v_{n-1}, \ldots, v_0]$*.*

3. *Une matrice de Toeplitz pour* ϕ_+ *est une matrice de Toeplitz pour* ϕ_-*.*

La preuve de ce résultat découle de la proposition suivante.

Proposition 3 (conversion $\Sigma LU \leftrightarrow \Sigma UL$). *Pour toute matrice* A*, l'inégalité* $|\alpha_+(A) - \alpha_-(A)| \leq 2$ *est satisfaite. De plus, on peut effectuer les conversions d'une représentation* $\sum LU$ *à une représentation* $\sum UL$*, et inversement, en* $O(\alpha\, \mathsf{M}(n))$ *opérations dans* $\mathbb{K}$*.*

Démonstration. Il suffit de prouver l'identité :

$$L(x) \cdot U(y) = I_n \cdot L(y') + U(x') \cdot I_n - U(x'') \cdot L(y''),$$

avec : $x'' = Z \cdot \operatorname{rev}(x)$, $y'' = Z \cdot \operatorname{rev}(y)$, $y' = \operatorname{rev}({}^tU(y) \cdot {}^tL(x) \cdot f)$, $x' = \operatorname{rev}(L(x) \cdot U(y) \cdot f)$, où $f = {}^t[0, \ldots, 0, 1]$. □

Théorème 2. *Soit* $A \in \mathcal{M}_n(\mathbb{K})$ *une matrice quasi-Toeplitz inversible. Alors son inverse est aussi quasi-Toeplitz et* $\alpha_+(A^{-1}) = \alpha_-(A)$*.*

Inversion rapide de matrices quasi-Toeplitz

Entrée : Une matrice « générique » $A \in \mathcal{M}_n(\mathbb{K})$, de taille $n = 2^k$, représentée par des générateurs (pour l'opérateur de déplacement ϕ_+).

Sortie : Son inverse A^{-1}, représentée par des générateurs (pour l'opérateur de déplacement ϕ_-).

1. Si $n = 1$, renvoyer A^{-1}.
2. Calculer des ϕ_+-générateurs pour $a, b, c, d \in \mathcal{M}_{n/2}(\mathbb{K})$, où $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$.
3. Calculer récursivement des ϕ_--générateurs pour $e := a^{-1}$.
4. Calculer des ϕ_+-générateurs pour $Z := d - ceb$.
5. Calculer récursivement des ϕ_--générateurs pour $t := Z^{-1}$.
6. Renvoyer des ϕ_--générateurs de $A^{-1} = \begin{bmatrix} x & y \\ z & t \end{bmatrix}$ via les formules de Strassen $y := -ebt$, $z := -tce$ et $x := e + ebtce$.

FIGURE 1. Algorithme de type Strassen pour inverser une matrice quasi-Toeplitz.

DÉMONSTRATION. On a la suite d'égalités

$$\begin{aligned}\alpha_-(A) &= \operatorname{rang}(A - {}^tZ \cdot A \cdot Z) = \operatorname{rang}(I_n - A^{-1} \cdot {}^tZ \cdot A \cdot Z) \\ &= \operatorname{rang}(I_n - Z \cdot A^{-1} \cdot {}^tZ \cdot A) = \operatorname{rang}(A^{-1} - Z \cdot A^{-1} \cdot {}^tZ) \\ &= \alpha_+(A^{-1}),\end{aligned}$$

dans laquelle nous avons utilisé un fait d'algèbre linéaire élémentaire : si A, B sont deux matrices quelconques, alors $\operatorname{rang}(I_n - A \cdot B) = \operatorname{rang}(I_n - B \cdot A)$. □

2.4. Résolution rapide de systèmes quasi-Toeplitz. Le Théorème 2 ne nous dit pas *comment* calculer des générateurs pour A^{-1}. Cela est fait dans le dernier résultat de ce chapitre.

THÉORÈME 3. *Soit $A \in \mathcal{M}_n(\mathbb{K})$ une matrice quasi-Toeplitz inversible, de rang de déplacement α, donnée par des générateurs G et H de taille $n \times \alpha$. On peut calculer des générateurs de taille $n \times \alpha$ pour A^{-1} en $O(\alpha^2\,\mathsf{M}(n)\log(n))$ opérations dans $\mathbb{K}$.*

À partir de cette représentation de l'inverse, le système $Ax = b$ ($b \in \mathbb{K}^n$) peut se résoudre pour un supplément de $O(\alpha\,\mathsf{M}(n))$ opérations dans $\mathbb{K}$.

DÉMONSTRATION. L'idée est d'adapter au cas structuré l'algorithme d'inversion de Strassen présenté au Chapitre 8 (page 135). La différence réside dans le choix de la structure de données utilisée pour représenter et calculer avec les matrices quasi-Toeplitz. Pour ce faire, on utilise comme structure de données compacte les générateurs de déplacement pour les opérateurs ϕ_+ et ϕ_- (ou, de façon équivalente, les représentations ΣLU et ΣUL).

L'algorithme, de type « diviser pour régner », est décrit en Figure 1. Sa correction est héritée de celle de l'algorithme d'inversion de Strassen ; l'hypothèse sur la généricité de A est utilisée pour garantir que tous les mineurs qu'on doit inverser au cours de l'algorithme sont bien inversibles.

Sa complexité arithmétique $\mathsf{C}(n)$ obéit à la récurrence

$$\mathsf{C}(n) \leq 2\,\mathsf{C}(n/2) + O(\mathsf{Mul}(n, \alpha)) + O(\alpha^2 n + \alpha\,\mathsf{M}(n)),$$

où le coût en $O(\alpha^2 n + \alpha\,\mathsf{M}(n))$ provient des conversions entre les représentations ϕ_+ et ϕ_- et des calculs de minimisation de longueurs de générateurs, et la notation $\mathsf{Mul}(n,\alpha)$ désigne la complexité du produit matriciel pour des matrices $n\times n$ données par générateurs de taille $n\times\alpha$. La preuve se conclut en utilisant l'estimation

$$\mathsf{Mul}(n,\alpha) = O(\alpha^2\,\mathsf{M}(n)),$$

qui entraîne $\mathsf{C}(n) = O(\alpha^2\,\mathsf{M}(n)\log(n))$. □

Exercices

Exercice 3. Soit $A \in \mathcal{M}_n(\mathbb{K})$ une matrice quasi-Toeplitz de rang de déplacement $\alpha \ll n$, représentée de façon compacte par des générateurs (G,H) de taille $n\times\alpha$, *i.e.*, tels que $\phi_+(A) = G\cdot {}^tH$.

1. Soient $v_1,\ldots,v_\alpha$ des vecteurs quelconques de $\mathbb{K}^n$. Montrer que le calcul de tous les produits $A\cdot v_\ell$, $1\le\ell\le\alpha$, se ramène, grâce à la formule ΣLU, au problème suivant :

> **(P)** Étant donnés des polynômes $G_j, H_j, V_j \in \mathbb{K}[X]$ $(j\le\alpha)$ de degrés au plus $n-1$, calculer
>
> $$A_\ell = \sum_{j=1}^{\alpha} G_j\,(H_jV_\ell \bmod X^n), \qquad 1\le\ell\le\alpha.$$

2. Donner un premier algorithme qui résout le problème **(P)** et estimer sa complexité.
3. Montrer que le problème **(P)** admet la reformulation matricielle suivante :

> **(MP)** Étant données des matrices polynomiales $\mathbf{G},\mathbf{V}$ dans $\mathcal{M}_{\alpha\times1}(\mathbb{K}[X])$ et $\mathbf{H}$ dans $\mathcal{M}_{1\times\alpha}(\mathbb{K}[X])$, toutes de degrés au plus $n-1$, calculer $(\mathbf{VH} \bmod X^n)\,\mathbf{G}$.

4. Soient $\mathbf{A},\mathbf{B}$ et $\mathbf{C}$ des matrices polynomiales de tailles $(n\times p)$, $(p\times n)$ et $(n\times p)$ et de degré au plus d. Montrer que le produit $\mathbf{ABC}$ peut être calculé en $O(\frac{n}{p}\,\mathsf{MM}(p,d))$ opérations dans $\mathbb{K}$.
5. Proposer un algorithme de type « diviser pour régner » (par rapport à n) résolvant le problème **(MP)** en complexité $O(\frac{1}{\alpha}\,\mathsf{MM}(\alpha,n))$.
6. Conclure qu'on peut multiplier, dans la représentation par générateurs de déplacement, deux matrices quasi-Toeplitz de $\mathcal{M}_n(\mathbb{K})$ de rang de déplacement au plus α en $O(\frac{1}{\alpha}\,\mathsf{MM}(\alpha,n))$ opérations dans $\mathbb{K}$.

Notes

Levinson [13] a donné le premier algorithme de complexité quadratique pour la résolution d'un système linéaire dont la matrice est de Toeplitz, symétrique et définie positive. Trench [18] a par la suite montré que toute l'inverse d'une telle matrice peut se calculer en la même complexité. Ces deux algorithmes sont décrits dans [7, §4.7].

L'algorithme de Trench [18] repose sur une formule, reprise et améliorée par Gohberg et Semencul [5], qui fournit une représentation explicite ΣLU à deux termes de l'inverse d'une matrice de Toeplitz A, les sommands étant construits à partir de la première ligne et de la première colonne de A^{-1}. Brent, Gustavson et Yun [3, 8] montrent comment ramener le calcul des premières ligne et colonne de A^{-1} à un calcul d'approximation de Padé. L'article [3] met en évidence les liens profonds entre la résolution des systèmes de Toeplitz, l'algorithme d'Euclide étendu et l'approximation de Padé, et propose le premier algorithme pour la résolution

d'un système défini par une matrice de Toeplitz inversible quelconque de taille n en $O(\mathsf{M}(n)\log n)$ opérations arithmétiques.

Parallèlement, la notion de rang de déplacement a été introduite par Kailath, Kung et Morf dans [10] et utilisée par les mêmes auteurs dans [9] pour donner un algorithme de complexité quadratique pour la résolution d'un système quasi-Toeplitz inversible. Plus exactement, l'algorithme de [9] calcule la solution d'un système défini par une matrice quasi-Toeplitz inversible de rang de déplacement α en $O(\alpha n^2)$ opérations. Une version rapide, de complexité $O(\alpha^d\,\mathsf{M}(n)\log n)$ a été obtenue indépendamment par Bitmead et Anderson [1] (avec $d=4$) et Morf [14] (avec $d=2$). Les algorithmes de [1, 14] fonctionnent seulement sous une hypothèse de forte régularité de l'entrée. Ils ont été adaptés dans [11, 12] au cas où la matrice du système est quelconque. Ces algorithmes ont été étendus au cas quasi-Cauchy dans [4, 17] et au cas quasi-Vandermonde dans [6, 15]. La meilleure complexité asymptotique vis-à-vis simultanément de la taille de la matrice et de son rang de déplacement est due à l'article [2], qui montre comment introduire du produit rapide de matrices denses dans l'algorithmique pour les matrices structurées, et réduit le coût à $\tilde{O}(\alpha^{\theta-1}n)$ pour la résolution des systèmes quasi-Toeplitz, quasi-Vandermonde et quasi-Cauchy.

Une bonne référence sur les matrices structurées est le livre [16].

Bibliographie

[1] Bitmead, Robert R. et Brian D. O. Anderson (1980). « Asymptotically fast solution of Toeplitz and related systems of linear equations ». In : *Linear Algebra and its Applications*, vol. 34, p. 103–116.

[2] Bostan, Alin, Claude-Pierre Jeannerod et Éric Schost (2008). « Solving structured linear systems with large displacement rank ». In : *Theoretical Computer Science*, vol. 407, n°1-3, p. 155–181.

[3] Brent, Richard P., Fred G. Gustavson et David Y. Y. Yun (1980). « Fast solution of Toeplitz systems of equations and computation of Padé approximants ». In : *Journal of Algorithms*, vol. 1, n°3, p. 259–295.

[4] Cardinal, Jean-Paul (1999). « On a property of Cauchy-like matrices ». In : *Comptes Rendus de l'Académie des Sciences. Série I. Mathématique*, vol. 328, n°11, p. 1089–1093.

[5] Gohberg, I. C. et A. A. Semencul (1972). « On the inversion of finite Toeplitz matrices and their continuous analogues (In Russian) ». In : *Matematicheskie Issledovaniya*, vol. 7, n°2, p. 201–223.

[6] Gohberg, I. et V. Olshevsky (1994). « Complexity of multiplication with vectors for structured matrices ». In : *Linear Algebra and its Applications*, vol. 202, p. 163–192.

[7] Golub, Gene H. et Charles F. Van Loan (1996). *Matrix computations*. 3e éd. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, xxx+698 pages.

[8] Gustavson, Fred G. et David Y. Y. Yun (1979). « Fast algorithms for rational Hermite approximation and solution of Toeplitz systems ». In : *IEEE Transactions on Circuits and Systems*, vol. 26, n°9, p. 750–755.

[9] Kailath, T., S. Y. Kung et M. Morf (1979a). « Displacement ranks of a matrix ». In : *American Mathematical Society. Bulletin. New Series*, vol. 1, n°5, p. 769–773.

[10] Kailath, Thomas, Sun Yuan Kung et Martin Morf (1979b). « Displacement ranks of matrices and linear equations ». In : *Journal of Mathematical Analysis and Applications*, vol. 68, n°2, p. 395–407.

[11] Kaltofen, Erich (1994). « Asymptotically fast solution of Toeplitz-like singular linear systems ». In : *ISSAC'94 : International Symposium on Symbolic and Algebraic Computation*. Oxford, United Kingdom : ACM Press, p. 297–304. doi : 10.1145/190347.190431.

[12] — (1995). « Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems ». In : *Mathematics of Computation*, vol. 64, n°210, p. 777–806.

[13] Levinson, Norman (1947). « The Wiener RMS (root mean square) error criterion in filter design and prediction ». In : *Journal of Mathematics and Physics*, vol. 25, p. 261–278.

[14] Morf, M. (1980). « Doubling algorithms for Toeplitz and related equations ». In : *IEEE Conference on Acoustics, Speech, and Signal Processing*, p. 954–959.

[15] Pan, Victor (1990). « On computations with dense structured matrices ». In : *Mathematics of Computation*, vol. 55, n°191, p. 179–190.

[16] Pan, Victor Y. (2001). *Structured matrices and polynomials*. Unified superfast algorithms. Birkhäuser Boston Inc., xxvi+278 pages.

[17] Pan, Victor Y. et Ailong Zheng (2000). « Superfast algorithms for Cauchy-like matrix computations and extensions ». In : *Linear Algebra and its Applications*, vol. 310, n°1-3, p. 83–108.

[18] Trench, William F. (1964). « An algorithm for the inversion of finite Toeplitz matrices ». In : *J. Soc. Indust. Appl. Math.* vol. 12, p. 515–522.

CHAPITRE 11

Solutions rationnelles de systèmes linéaires à coefficients polynomiaux

Résumé

L'algorithmique des systèmes linéaires à coefficients des polynômes en une variable est très similaire à celle des fractions rationnelles. En outre, il est important de tirer parti du produit rapide de matrices.

On considère le système linéaire

$$A(X)Y(X) = B(X), \tag{1}$$

où A et B sont donnés et Y est inconnu. A est une matrice $n \times n$ de polynômes, régulière (de déterminant non nul) et B est un vecteur de polynômes. De manière équivalente, on peut voir A (ou B) comme un polynôme à coefficients des matrices (ou des vecteurs). Pour fixer les notations, on suppose $\deg A \leq d$ et $\deg B < d$.

On notera $\mathcal{M}_{m,k}(R)$ l'ensemble des matrices de taille $m \times k$ à coefficients dans R. On notera $\mathbb{K}[X]_d$ l'ensemble des polynômes de degré au plus d à coefficients dans le corps $\mathbb{K}$. La fonction M est telle que la multiplication dans $\mathbb{K}[X]_d$ coûte au plus $\mathsf{M}(d)$ opérations dans $\mathbb{K}$. L'exposant θ est tel que la multiplication de deux matrices $n \times n$ à coefficients dans $\mathbb{K}$ coûte $\mathsf{MM}(n) = O(n^\theta)$ opérations dans $\mathbb{K}$.

Nous aurons aussi besoin de produits de matrices de $\mathcal{M}_n(\mathbb{K}[X]_d)$. Dans le cas le plus général, ce produit est connu pour pouvoir être exécuté en $\mathsf{MM}(n,d) = O(n^\theta\, \mathsf{M}(d))$ opérations dans $\mathbb{K}$, borne que nous utiliserons dans les estimations de complexité. Lorsque le corps $\mathbb{K}$ contient suffisamment de points, par évaluation-interpolation, ce coût descend à $O(n^\theta d + n^2\, \mathsf{M}(d) \log d)$; dans les mêmes conditions, en choisissant des points en progression géométrique, cette complexité peut être encore abaissée à $O(n^\theta d + n^2\, \mathsf{M}(d))$, voir p. 79.

1. Des séries aux solutions rationnelles

Une première observation est que la structure de la solution cherchée est donnée par les formules de Cramer.

Lemme 1. *Le système possède une solution dont les coordonnées sont des fractions rationnelles. Ses numérateurs et dénominateurs ont degrés bornés par $nd-1$ et nd.*

Démonstration. Le système (1) se récrit

$$A_1 y_1 + \cdots + A_n y_n = B,$$

où y_i est la $i$ème coordonnée de Y et A_i la $i$ème colonne de A. La formule de Cramer

$$\det(A_1, \ldots, A_{i-1}, B, A_{i+1}, \ldots, A_n) = y_i \det(A),$$

est obtenue en remplaçant B par sa valeur dans le membre gauche et en développant le déterminant.

Il en découle que y_i est le quotient de déterminants de matrices appartenant à $\mathcal{M}_n(\mathbb{K}[X]_d)$. Ces déterminants ont donc degré au plus $nd-1$ pour le numérateur et nd pour le dénominateur. □

L'algorithme de résolution suivant est justifié par la complexité quasi-optimale des approximants de Padé (Chapitre 7).

Résolution de $A(X)Y(X) = B(X)$ [Moenk-Carter 1979]

Entrée : $A \in \mathcal{M}_n(\mathbb{K}[X]_d)$, $B \in \mathcal{M}_{n,1}(\mathbb{K}[X]_{d-1})$.

Sortie : le vecteur de fractions rationnelles Y tel que $AY = B$.

1. Calculer le développement en série de $A^{-1}B$ à précision $2nd$.
2. Reconstruire les coefficients de Y par approximant de Padé.

Dans tous les algorithmes qui vont suivre, c'est la recherche de série solution qui va dominer la complexité.

2. Développement comme une fraction rationnelle

La Proposition 3 du Chapitre 3 (page 46) montre que la méthode de Newton fonctionne pour toute matrice inversible de séries. Avec cet algorithme, le calcul de la fraction rationnelle solution de (1) demande $O(n^\theta \mathsf{M}(nd))$ opérations dans $\mathbb{K}$.

Il est possible d'améliorer l'efficacité lorsque la matrice est une matrice de polynômes. La base de cette amélioration est contenue dans le lemme suivant.

LEMME 2. *Soit $A(X) \in \mathcal{M}_n(\mathbb{K}[X]_d)$ et $B(X) \in \mathcal{M}_{n,m}(\mathbb{K}[X]_{d-1})$, avec A inversible. Pour tout k, il existe une matrice $B_k \in \mathcal{M}_{n,m}(\mathbb{K}[X]_{d-1})$ telle que*

$$A^{-1}B = a_0 + a_1X + \cdots + a_{k-1}X^{k-1} + X^kA^{-1}B_k, \tag{2}$$

où $a_i \in \mathcal{M}_{n,m}(\mathbb{K})$.

DÉMONSTRATION. Si les a_i sont les coefficients du développement en série de $A^{-1}B$, alors

$$B - A(a_0 + \cdots + a_{k-1}X^{k-1})$$

est une matrice de $\mathcal{M}_{n,m}(\mathbb{K}[X]_{d+k-1})$ qui, par construction, est divisible par X^k, d'où le lemme. □

Ce résultat se traduit algorithmiquement comme suit.

Développement de $A^{-1}B$

Entrée : A, B, k et $S = A^{-1} \bmod X^k$.

Sortie : $a_0, \ldots, a_{k-1}$ et B_k définis par l'équation (2).

1. Calculer $SB =: a_0 + \cdots + a_{k-1}X^{k-1} \bmod X^k$.
2. Calculer $B_k = (B - A(a_0 + \cdots + a_{k-1}X^{k-1}))X^{-k}$.
3. Renvoyer $a_0, \ldots, a_{k-1}, B_k$.

La proposition suivante estime le coût de cet algorithme. Elle peut être vue comme l'analogue matriciel du Théorème 4 au Chapitre 4 (page 66).

PROPOSITION 1. *Soit $A \in \mathcal{M}_n(\mathbb{K}[X]_d)$ avec $A(0)$ inversible, alors on peut calculer les $N \geq d$ premiers coefficients de A^{-1} en $O(n^\theta N\, \mathsf{M}(d)/d)$ opérations dans $\mathbb{K}$. Pour $B \in \mathcal{M}_{n,1}(\mathbb{K}[X]_{d-1})$, on peut calculer les $N \geq d$ premiers coefficients de $A^{-1}B$ en $O(n^2 N\, \mathsf{M}(d)/d)$ opérations dans $\mathbb{K}$.*

DÉMONSTRATION. L'algorithme calcule d'abord l'inverse $S = A^{-1} \bmod X^d$ par l'algorithme de Newton, en $O(\mathsf{MM}(n,d)) = O(n^\theta\, \mathsf{M}(d))$ opérations dans $\mathbb{K}$.

Ensuite, on applique N/d fois l'algorithme ci-dessus avec $k = d$ pour calculer à chaque itération d coefficients et un nouveau $B_{(i+1)d}$. Si on part de $B_0 = I$, le résultat fournit les N premiers coefficients de A^{-1} ; si $B_0 = B$, alors on obtient les coefficients de $A^{-1}B$.

Les deux étapes de l'algorithme ci-dessus (avec $k = d$) coûtent $\mathsf{MM}(n, d) = O(n^\theta \mathsf{M}(d))$ opérations dans $\mathbb{K}$ si B a n colonnes, $O(n^2 \mathsf{M}(d))$ s'il n'en a qu'une. □

Avec cet algorithme, le calcul de la fraction rationnelle solution de (1) demande $O(n^3 \mathsf{M}(d))$ opérations dans $\mathbb{K}$.

3. L'algorithme de Storjohann

L'algorithme de Storjohann permet de calculer les N premiers coefficients du développement en série de $A^{-1}B$ en $O(n^{\theta-1} N \log N \mathsf{M}(d))$ opérations dans $\mathbb{K}$. Si $\theta < 3$, cette quantité croît avec n moins vite que la taille de l'inverse A^{-1}, qui n'est donc pas calculée en entier. Ainsi, la résolution du système linéaire (1) a un coût en $O(n^\theta \mathsf{M}(d) \log(nd))$ opérations dans $\mathbb{K}$.

L'algorithme repose sur le développement de $A^{-1}B$ de la section précédente, joint d'une part à une technique de type « diviser pour régner », d'autre part au regroupement des calculs intermédiaires pour remplacer des groupes de n produits matrice-vecteur par des produits matrice-matrice.

Pour commencer, on peut modifier l'algorithme de développement de $A^{-1}B$ de la section précédente pour calculer B_k sans calculer tous les coefficients $a_0, \ldots, a_{k-1}$. L'entrée nécessaire est moins grosse, et la complexité plus faible lorsque k est grand devant d. Pour une série $V = v_0 + v_1 X + \cdots$, on note

$$[V]_a^b := v_a X^a + \cdots + v_{a+b} X^{a+b},$$

avec la convention que les coefficients d'indice négatif de V sont nuls.

Développement de $A^{-1}B$ tronqué

Entrée : A, B, k et $S = [A^{-1}]_{k-2d+1}^{2d-2}$.

Sortie : B_k défini par l'équation (2).

1. Calculer $U := [SB]_{k-d}^{d-1}$.
2. Calculer $B_k := [B - AU]_k^{d-1} X^{-k}$.
3. Renvoyer B_k.

Démonstration. Pour prouver la correction de cet algorithme, il faut s'assurer que les troncatures de séries sont suffisantes pour calculer le même polynôme B_k que précédemment.

Pour la première étape de l'algorithme, il suffit d'observer que B ayant degré au plus $d-1$, le coefficient de X^i dans $A^{-1}B$ ne dépend que de $[A^{-1}]_{i-d-1}^{d+1}$, pour tout i. Donc les coefficients calculés par cette première étape sont les mêmes que les a_i que précédemment, pour $i = k-d, \ldots, k-1$.

Ensuite, il faut calculer $[X^k B_k]_k^{d-1}$. L'extraction des coefficients de l'équation (2) donne

$$\begin{aligned}[X^k B_k]_k^{d-1} &= [B - A(a_0 + \cdots + a_{k-1} X^{k-1})]_k^{d-1} \\ &= [B - A(a_{k-d} X^{k-d} + \cdots + a_{k-1} X^{k-1})]_k^{d-1},\end{aligned}$$

ce qui conclut la preuve. □

Une observation importante concernant cet algorithme est que c'est le même polynôme S qui peut servir pour calculer B_{i+k} pour tout i. L'idée est alors de grouper plusieurs vecteurs B_i et de calculer les B_{i+k} correspondants par des produits matrice-matrice. Noter que la ressemblance entre cette idée et celle de l'algorithme de Keller-Gehrig en page 137.

Ainsi, si l'on connaît $B_0, B_{2m}, \dots, B_{2sm}$ et $[A^{-1}]_{m-2d}^{2d}$, alors le calcul de la suite $B_m, B_{3m}, \dots, B_{(2s+1)m}$ ne requiert que $O(n^{\theta-1} s\, \mathsf{M}(d))$ opérations dans $\mathbb{K}$.

En itérant cette idée, en partant de B_0 et en supposant connus $[A^{-1}]_{2^k-2d+1}^{2d-2}$ pour $k = 0, \dots, \lceil \log(N/d) \rceil =: k_{\max}$, on obtient d'abord $B_0, B_{2^{k_{\max}}}$, puis à l'étape suivante $B_0, B_{2^{k_{\max}-1}}, B_{2^{k_{\max}}}, B_{3\cdot 2^{k_{\max}-1}}$, et ainsi de suite jusqu'à calculer toute la suite $B_0, B_d, B_{2d}, \dots, B_{d\lceil N/d \rceil}$ en $O(k_{\max}\, n^{\theta-1} N\, \mathsf{M}(d)/d)$ opérations dans $\mathbb{K}$. En multipliant alors ces vecteurs par $[A^{-1}]_0^d$ on obtient finalement les N premiers coefficients de $A^{-1}B$ pour le même coût.

Il reste à voir comment calculer les $[A^{-1}]_{2^k-2d-1}^{2d-2}$. Là encore, le point de départ est l'identité (2), avec $B = I$, $k = m$ et $k = p$:

$$\begin{aligned} A^{-1} &= a_0 + \cdots + a_{m-1}X^{m-1} + X^m A^{-1} B_m \\ &= a_0 + \cdots + a_{m-1}X^{m-1} + X^m(a_0 + \cdots + a_{p-1}X^{p-1} + X^p A^{-1} B_p) B_m. \end{aligned}$$

La seconde ligne est obtenue par substitution de la valeur de A^{-1} donnée par la première ligne avec $m = p$. Ces équations entraînent pour tout $\ell \geq 0$ tel que $m + p - \ell \geq d$

$$\begin{cases} B_{m+p} & = [-A[A^{-1}]_{p-2d+1}^{2d-2} B_m]_p^{d-1} X^{-p}, \\ [A^{-1}]_{m+p-\ell}^{\ell-1} & = [[A^{-1}]_{p-\ell-d+1}^{\ell+d-2} B_m]_{m+p-\ell}^{\ell-1}. \end{cases}$$

L'algorithme suivant s'en déduit en utilisant cette identité avec $m = 2^k - d$, $p = 2^k$ et $\ell = d - 1$.

Développement de A^{-1} — indices puissances de 2

Entrée : $S = [A^{-1}]_0^{d-1}$, $T = [A^{-1}]_{2^k-2d+1}^{2d-2}$ et B_{2^k-d} défini par (2) avec $B = I$.

Sortie : $B_{2^{k+1}-d}$ et $[A^{-1}]_{2^{k+1}-2d+1}^{2d-2}$.

1. Calculer $[A^{-1}]_{2^{k+1}-2d+1}^{d-2} = [T B_{2^k-d}]_{2^{k+1}-2d+1}^{d-2}$.
2. Calculer $B_{2^{k+1}-d} := [-AT B_{2^k-d}]_{2^k}^{d-1} / X^{2^k}$.
3. Calculer $[A^{-1}]_{2^{k+1}-d}^{d-1} = [X^{2^{k+1}-d} B_{2^{k+1}-d} S]_{2^{k+1}-d}^{d-1}$.
4. Renvoyer $B_{2^{k+1}-d}$ et $[A^{-1}]_{2^{k+1}-2d+1}^{2d-2}$.

Pour résumer, ces algorithmes mènent au résultat suivant.

THÉORÈME 1 (Storjohann 2002). *Soient A une matrice $n \times n$ polynomiale de degré d avec $A(0)$ inversible et B un vecteur polynomial de degré au plus $d-1$, alors on peut calculer le numérateur et le dénominateur de $A^{-1}B$ en $O(n^\theta\, \mathsf{M}(d) \log(nd))$ opérations dans $\mathbb{K}$.*

Notes

L'idée de résoudre des systèmes linéaires à coefficients polynomiaux (resp. entiers) par développement de Taylor (resp. p-adique) et reconstruction rationnelle provient de [1, 4]. L'algorithme en page 167 et la Proposition 1 sont tirés de [4].

Dans tout ce texte, nous avons supposé que $A(0)$ est inversible. En fait, les résultats s'étendent au cas où A est inversible sans que $A(0)$ le soit. Il suffit de tirer aléatoirement un point et d'effectuer les développements en série au voisinage de ce point. L'algorithme devient probabiliste. Si la caractéristique est positive, il se peut que A soit inversible sans que sa valeur en aucun point du corps ne le soit. On peut alors construire une extension du corps assez grande pour trouver un point où effectuer ces calculs. Ces considérations sont prises en compte dans [5, 6].

L'article [6] montre que le déterminant d'une matrice polynomiale de taille n et degré au plus d peut se calculer en $O(n^\theta \mathsf{M}(d) \log^2(n))$ opérations arithmétiques. Storjohann et Villard [8] montrent comment calculer le rang et une base du noyau d'une telle matrice en $\tilde{O}(n^\theta d)$ opérations arithmétiques.

Pour n une puissance de 2, Jeannerod et Villard [3] ont donné un algorithme qui calcule l'inverse d'une matrice polynomiale inversible de taille n et degré au plus d en $\tilde{O}(n^3 d)$ opérations arithmétiques. Des réductions entre diverses opérations sur les matrices polynomiales ont été étudiées dans [2].

L'algorithme de ce chapitre et tous les autres résultats de [5, 6] ont été étendus au cas entier dans [7]. Il n'est toujours pas connu si l'on peut calculer le polynôme caractéristique d'une matrice polynomiale de taille n et degré au plus d en $\tilde{O}(n^\theta d)$ opérations arithmétiques.

Bibliographie

[1] Dixon, John D. (1982). « Exact solution of linear equations using p-adic expansions ». In : *Numerische Mathematik*, vol. 40, n°1, p. 137–141.

[2] Giorgi, Pascal, Claude-Pierre Jeannerod et Gilles Villard (2003). « On the complexity of polynomial matrix computations ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par J. R. Sendra. New York : ACM Press, p. 135–142.

[3] Jeannerod, Claude-Pierre et Gilles Villard (2005). « Essentially optimal computation of the inverse of generic polynomial matrices ». In : *Journal of Complexity*, vol. 21, n°1, p. 72–86.

[4] Moenck, Robert T. et John H. Carter (1979). « Approximate algorithms to derive exact solutions to systems of linear equations ». In : *EUROSAM'79 : International Symposium on Symbolic and Algebraic Computation*. Vol. 72. Lecture Notes in Computer Science. London, UK : Springer-Verlag, p. 65–73.

[5] Storjohann, Arne (2002). « High-Order Lifting ». In : *ISSAC'02 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Teo Mora. ACM Press, p. 246–254.

[6] — (2003). « High-order lifting and integrality certification ». In : *Journal of Symbolic Computation*, vol. 36, n°3-4, p. 613–648.

[7] — (2005). « The shifted number system for fast linear algebra on integer matrices ». In : *Journal of Complexity*, vol. 21, n°4, p. 609–650.

[8] Storjohann, Arne et Gilles Villard (2005). « Computing the rank and a small nullspace basis of a polynomial matrix ». In : *ISSAC'05 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 309–316.

CHAPITRE 12

Principe de transposition de Tellegen

Résumé

Le principe de transposition est un ensemble de règles de transformation pour les algorithmes calculant des applications linéaires. Pour tout algorithme qui calcule des produits matrice-vecteur par une matrice fixée $\mathbf{M}$, ces règles de transformation permettent d'obtenir un algorithme dual qui calcule des produits matrice-vecteur par la matrice transposée de $\mathbf{M}$. En outre, la complexité arithmétique de l'algorithme dual est essentiellement égale à celle de l'algorithme initial.

1. Introduction

Le *théorème de transposition de Tellegen* affirme que, étant donnée une matrice $\mathbf{M}$ de taille $m \times n$ à coefficients dans un corps $\mathbb{K}$, sans lignes ni colonnes nulles, tout algorithme linéaire $\mathcal{A}$ qui calcule l'application linéaire $\mathbf{v} \mapsto \mathbf{M} \cdot \mathbf{v}$ de $\mathbb{K}^n \to \mathbb{K}^m$ en L opérations dans $\mathbb{K}$ peut être transformé en un *algorithme dual* ${}^t\mathcal{A}$ qui calcule l'application linéaire transposée $\mathbf{w} \mapsto {}^t\mathbf{M} \cdot \mathbf{w}$ de $\mathbb{K}^m \to \mathbb{K}^n$ en $L - n + m$ opérations dans $\mathbb{K}$. Ici, par *algorithme linéaire* on entend un algorithme qui n'utilise que des opérations linéaires en les éléments de l'entrée.

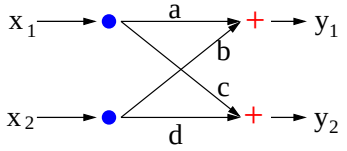
Par extension, on appelle *principe de Tellegen* l'ensemble des règles de transformation réalisant le passage de l'algorithme direct à l'algorithme dual.

Motivation. Si l'algorithme utilisé pour la multiplication matrice-vecteur est l'algorithme naïf, cet énoncé devient trivial. En effet, dans ce cas :

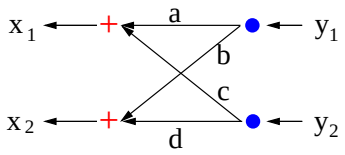
- $\mathbf{M} \cdot \mathbf{v}$ requiert $\sum_i (\alpha_i - 1) = E - m$ opérations $\pm$ et E opérations $\times$
- ${}^t\mathbf{M} \cdot \mathbf{w}$ requiert $\sum_j (\beta_j - 1) = E - n$ opérations $\pm$ et E opérations $\times$

où α_i (resp. β_j) est le nombre d'éléments non-nuls de la i^{e} ligne (resp. de la j^{e} colonne) de $\mathbf{M}$, et E est le nombre d'éléments non-nuls de $\mathbf{M}$. Par conséquent, pour une matrice complètement générique, le théorème de Tellegen ne présente aucun intérêt. Par contre, si la matrice $\mathbf{M}$ admet une structure, il est concevable que l'on dispose d'un algorithme plus rapide que l'algorithme naïf (quadratique) pour la multiplier par un vecteur de $\mathbb{K}^n$. En utilisant le principe de transposition, on obtient aussitôt un algorithme rapide pour multiplier ${}^t\mathbf{M}$ par un vecteur de $\mathbb{K}^m$.

Exemple 1. Considérons l'exemple suivant, qui illustre ce principe en utilisant la représentation par graphes des algorithmes linéaires. L'algorithme direct prend les valeurs x_1 et x_2 en entrée et renvoie $y_1 = ax_1 + bx_2$ et $y_2 = cx_1 + dx_2$ en sortie. Les arêtes représentent des multiplications par des valeurs scalaires a, b, c, d.



Transposer cet algorithme revient à inverser le flot du calcul, c'est-à-dire, inverser le sens des flèches, permuter les '+' avec les '•' et les entrées avec les sorties. L'algorithme ainsi obtenu prend y_1, y_2 en entrée et renvoie $x_1 = ay_1 + cy_2$ et $x_2 = by_1 + dy_2$. Il calcule donc bien l'application transposée de l'application initiale. De plus, le nombre d'opérations arithmétiques utilisées par les deux algorithmes est le même : 4 multiplications et 2 additions.



Utilité. Comme nous l'avons vu au Chapitre 10, beaucoup de problèmes en calcul formel s'expriment en termes d'algèbre linéaire structurée (multiplication par un vecteur ou résolution de système). Par exemple, les opérations élémentaires sur les polynômes (multiplication, division, évaluation-interpolation, interpolation rationnelle, extrapolation, etc.) sont *linéaires* si l'on fixe l'un des opérandes : ils se codent en termes de produits matrice-vecteur $\mathbf{M} \cdot \mathbf{v}$ ou $\mathbf{M}^{-1} \cdot \mathbf{v}$, où $\mathbf{M}$ est une matrice structurée (de type Toeplitz, Hankel, compagnon, Vandermonde, Cauchy,...)

Grâce au principe de Tellegen, comprendre, analyser et améliorer un algorithme se ramène à comprendre, analyser et améliorer son transposé. Deux sont ses utilités principales : *trouver* des solutions algorithmiques de meilleure complexité, et *clarifier* le statut de certains algorithmes existant dans la littérature, qui parfois sont simplement des transposés d'algorithmes bien connus. Cela permet ainsi le traitement algorithmique unifié des problèmes duaux. On peut résumer en disant que le principe de transposition permet de diviser par deux le nombre d'algorithmes linéaires qu'il reste à découvrir.

Un exemple moins trivial. Considérons le problème de l'évaluation d'un polynôme $P \in \mathbb{K}[X]$ de degré n en une valeur $a \in \mathbb{K}$. C'est une opération linéaire en les coefficients de P, de matrice $\mathbf{M} = [1, a, \ldots, a^n]$ dans les bases canoniques. Le problème transposé est donc le suivant : pour une valeur donnée $x_0 \in \mathbb{K}$, calculer les produits $a^i x_0$, pour $0 \leq i \leq n$. Pour ce problème, un algorithme naturel consiste à multiplier x_0 par a, ensuite multiplier le résultat par a, et ainsi de suite.

Le transposé de cet algorithme s'obtient en parcourant l'algorithme direct en sens inverse, tout en permutant les entrées avec les sorties, et en remplaçant chaque instruction par sa *transposée*, obtenue en appliquant un nombre restreint de règles syntaxiques. Dans ce processus, les boucles `for` montantes deviennent des boucles `for` descendantes.

De cette manière, on obtient *automatiquement* un algorithme pour le problème de départ, à savoir, l'évaluation de P sur a. Dans notre cas, il s'avère que l'algorithme transposé coïncide avec la fameuse *méthode de Horner*, voir la Figure 1. Observons que l'algorithme transposé utilise n opérations de plus que l'algorithme direct. Cette perte s'explique par le théorème de Tellegen : il s'agit tout simplement de la différence entre le nombre de colonnes et le nombre de lignes de $\mathbf{M}$. Cette observation peut être utilisée pour expliquer l'optimalité de la règle de Horner.

2. La version en termes de graphes du principe de Tellegen

Dans cette section nous donnons une version du théorème de Tellegen dans un modèle particulier, celui des graphes de calcul (DAG).

$$\mathbf{M} = [1, a, \dots, a^n]$$

$x_0 \mapsto [x_0, ax_0, \dots, a^n x_0]$ | $[p_0, \dots, p_n] \mapsto \sum_{i=0}^{n} p_i a^i$

Entrée x_0.	**Entrée** $p = [p_0, \dots p_n]$.
$p_0 \leftarrow x_0$;	**for** j **from** n **downto** 1 **do**
for j **from** 1 **to** n **do**	$p_j \leftarrow a p_j$;
$p_j \leftarrow p_{j-1}$;	$p_{j-1} \leftarrow p_j + p_{j-1}$;
$p_j \leftarrow a p_j$;	$x_0 \leftarrow p_0$;
Sortie $p = [p_0, \dots, p_n]$.	**Sortie** x_0.

FIGURE 1. Le schéma de Horner (à droite) obtenu en transposant l'algorithme qui résout le problème dual.

DEFINITION 1. Un *graphe acyclique orienté,* ou DAG (de l'anglais *directed acyclic graph*) est un graphe orienté $\mathsf{G} = (V, E)$ qui ne possède pas de cycle. Un DAG $\mathbb{K}$-linéaire est un DAG muni d'une fonction de poids $\lambda : E \to \mathbb{K}$.

Soit $I = \{x_1, \dots, x_n\}$ l'ensemble des nœuds d'entrée de G. À tout sommet $v \in V$ on associe une forme linéaire dans $\mathbb{K}[X_1, \dots, X_n]$ de la façon suivante :

— si $v = x_i \in I$, alors $h_v := X_i$,

— si $v \in V \setminus I$, alors $h_v := \displaystyle\sum_{e=(w,v)\in E} \lambda(e)\, h_w$.

DEFINITION 2. On dit que G calcule la matrice $\mathbf{M} = [a_{i,j}]$ de taille $m \times n$ si les formes linéaires associées aux nœuds de sortie sont $F_i = \sum_{j=1}^{n} a_{i,j} X_j$, $i = 1, \dots, m$. Le coût de G est le nombre d'opérations linéaires $c(\mathsf{G})$ induites par G.

Avec ces notations, le principe de Tellegen s'énonce comme suit.

THÉORÈME 1. *Soit* G *un* $\mathbb{K}$*-DAG qui calcule une matrice* $\mathbf{M}$ *de* $\mathcal{M}_{m,n}(\mathbb{K})$. *Alors le graphe transposé* ${}^t\mathsf{G}$*, obtenu en inversant le sens des flèches sans changer leur poids, calcule la matrice* ${}^t\mathbf{M}$. *De plus,*

$$c({}^t\mathsf{G}) = c(\mathsf{G}) - n + m.$$

ESQUISSE DE PREUVE. La forme linéaire calculée par un sommet v de G est

$$h_v = \sum_{j=1}^{n} \left(\sum_{p \in \text{ Chemin}(x_j, v)} \lambda(p) \right) X_j, \quad \text{où} \quad \lambda(p) = \prod_{e \text{ arête de } p} \lambda(e).$$

On a donc l'égalité $a_{i,j} = \displaystyle\sum_{p \in \text{ Chemin}(x_j, F_i)} \lambda(p)$, qui montre que ${}^t\mathsf{G}$ calcule bien ${}^t\mathbf{M}$.

L'égalité entre les coûts se déduit du fait que la quantité $c(\mathsf{G}) - |I(\mathsf{G})|$ ne dépend pas de l'orientation de G, comme montré par le lemme suivant. □

LEMME 1 (formule pour le coût d'un graphe de calcul). *Avec les notations précédentes, on a l'égalité*

$$c(\mathsf{G}) = \big|\{e \in E \mid \lambda(e) \neq \pm 1\}\big| + |E| - |V| + |I|.$$

ESQUISSE DE PREUVE. Tout sommet v qui reçoit $d(v) > 0$ arêtes contribue au coût de G avec $\big|\{e = (w, v) \in E \mid \lambda(e) \neq \pm 1\}\big| + d(v) - 1$ opérations dans $\mathbb{K}$. Graphiquement, cela se voit sur la représentation de la Figure 2.

La conclusion se déduit alors aisément de la suite d'égalités

$$\sum_{v \in V \setminus I} \big(d(v) - 1\big) = \sum_{v \in V \setminus I} d(v) - \sum_{v \in V \setminus I} 1 = |E| - (|V| - |I|).$$

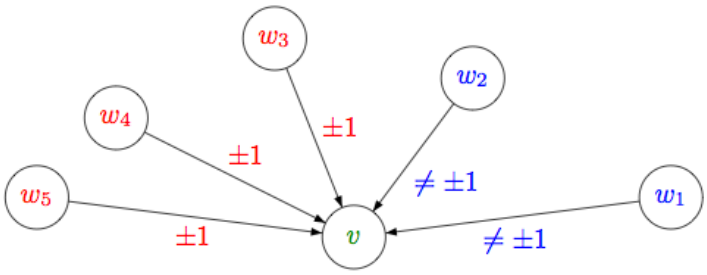


FIGURE 2. Le coût du DAG

□

3. Principe de Tellegen pour les programmes linéaires

Le but de cette section est de décrire le principe de Tellegen dans le modèle des programmes linéaires sur des machines à allocation de registres (machines à accès direct, MAD, ou random access memory, RAM, en anglais).

3.1. Machines à registres. Commençons par définir les machines MAD à instructions linéaires.

DEFINITION 3. On fixe un entier M ; il représente la mémoire disponible, de sorte que l'on peut accéder à M registres $R_1, \ldots, R_M$. On y stocke des nombres, c'est-à-dire des éléments du corps $\mathbb{K}$.

Un programme linéaire est la donnée des objets suivants :

- un sous-ensemble $R_{i_1}, \ldots, R_{i_n}$ des registres que l'on appelle *entrée* ;
- un sous-ensemble $R_{o_1}, \ldots, R_{o_m}$ des registres que l'on appelle *sortie* ;
- une suite d'instructions portant sur les registres, choisies parmi :
  - $R_i = \pm R_j \pm R_k$, avec $1 \leq i, j, k \leq M$,
  - $R_i = \lambda R_j$, avec $1 \leq i, j \leq M$ et λ dans $\mathbb{K}$.

Le fonctionnement d'un tel programme est le suivant : à l'initialisation, les registres d'entrée reçoivent des valeurs $x_1, \ldots, x_n \in \mathbb{K}$ et les autres sont mis à zéro. On effectue ensuite toutes les instructions, puis le programme renvoie les valeurs des registres de sortie. Noter que l'on calcule bel et bien une fonction linéaire des x_i.

3.2. Transposition de programme. Nous décrivons maintenant le principe de transposition des programmes linéaires sur une machine MAD.

Cas d'un jeu d'instructions réduit. Pour commencer, on traite le cas d'une machine avec un jeu d'instructions limité par rapport au cas général. On ne s'autorise que les instructions du type :

- $R_i = R_i \pm R_j$, avec $1 \leq i, j \leq M$;
- $R_i = \lambda R_i$, avec $1 \leq i \leq M, \lambda \in \mathbb{K}$.

Pour « transposer » un programme comportant des instructions de ce type, on interprète chacune de ces instructions comme une application linéaire $\mathbb{K}^M \to \mathbb{K}^M$.

Pour motiver ce procédé, considérons un exemple. Avec $M = 3$, l'instruction $R_1 = R_1 + R_3$ s'interprète comme l'application linéaire dont la matrice est

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

R_2 et R_3 n'ont pas changé et R_1 devient R_1+R_3. La transposée de cette application linéaire a pour matrice

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

on en déduit que l'on peut la traduire par l'instruction $R_3 = R_3 + R_1$.

De manière générale, la transposée de l'instruction $R_i = R_i+R_j$ est l'instruction $R_j = R_j + R_i$. Par écriture matricielle, on voit que l'instruction $R_i = \lambda R_i$ est inchangée par transposition, et que $R_i = R_i - R_j$ se transpose en $R_j = R_j - R_i$. On peut alors définir le programme transposé d'un programme.

Definition 4. Le programme transposé ${}^t\mathcal{P}$ du programme linéaire $\mathcal{P}$ est donné par :

- les entrées de ${}^t\mathcal{P}$ sont les sorties de $\mathcal{P}$;
- les sorties de ${}^t\mathcal{P}$ sont les entrées de $\mathcal{P}$;
- les instructions de ${}^t\mathcal{P}$ sont les transposées des instructions de $\mathcal{P}$, prises en sens inverse.

La dernière condition, le retournement de la liste des instructions, traduit l'égalité matricielle ${}^t(AB) = {}^tB \cdot {}^tA$. La définition montre que si $\mathcal{P}$ calcule une application linéaire $\mathbb{K}^n \to \mathbb{K}^m$, alors ${}^t\mathcal{P}$ calcule bien l'application transposée $\mathbb{K}^m \to \mathbb{K}^n$.

Exemple 2. Considérons le programme

```
R4=R4+R2
R3=3*R3
R4=R4+R3
R2=2*R2
R3=R3+R2
R3=R3+R1
```

dont les entrées sont R_1, R_2, R_3 et les sorties R_3, R_4. Ce programme calcule l'application linéaire dont la matrice est

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 3 \end{bmatrix}.$$

Le programme transposé s'écrit

```
R1=R1+R3
R2=R2+R3
R2=2*R2
R3=R3+R4
R3=3*R3
R2=R2+R4
```

et on vérifie qu'il calcule l'application linéaire dont la matrice est

$$\begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 3 & 3 \end{bmatrix}.$$

Cas général. Le cas du jeu d'instructions général se ramène au cas du jeu d'instructions réduit de manière immédiate. Ainsi, l'instruction $R_1 = R_2 + R_3$ est équivalente à la suite d'instructions

```
R1=0
R1=R1+R2
R1=R1+R3
```

Il est donc possible de le transposer sous la forme

```
R3=R3+R1
R2=R2+R1
R1=0
```

De même, on réécrit aisément l'instruction $R_i = \lambda R_j$ en utilisant le jeu d'instructions réduit, ce qui permet de la transposer, ...

3.3. Optimisations. Notre principe de transposition n'est pas complètement satisfaisant : au vu des règles de réécriture ci-dessus, il semble que l'on puisse perdre jusqu'à un facteur 3 (en termes de nombre de lignes). On peut optimiser le code produit, afin de regagner, autant que possible, les lignes perdues. On utilise pour ce faire les règles suivantes :

Suppression des zéros. La transformation pour passer du cas général au jeu d'instructions réduit introduit des lignes du type $R_i = 0$. On peut supprimer une telle ligne, à condition de réécrire les instructions suivantes en conséquence.

Suppression des recopies. Après avoir supprimé les zéros, il se peut qu'il reste des lignes du type $R_i = \pm R_j$. On peut également supprimer ce type de ligne, à condition de réécrire de manière judicieuse les instructions qui suivent.

Muni de ces règles supplémentaires, on peut montrer qu'il est possible d'obtenir un code transposé qui fait exactement le même nombre de lignes que l'original mais cela dépasse le cadre de ce cours.

4. Applications

Dans cette section, nous illustrons l'utilité pratique du principe de Tellegen à travers quelques exemples. Les techniques de transposition permettront d'engendrer des programmes linéaires pour effectuer les opérations suivantes sur les polynômes à une variable : multiplication, division euclidienne, évaluation et interpolation multipoint.

La figure 3 contient une liste (non-exhaustive) d'opérations linéaires de base sur les polynômes à une variable et leurs problèmes transposés. Tout algorithme résolvant un problème de la colonne de gauche peut être transposé en un algorithme de même complexité pour le problème correspondant de la colonne de droite.

4.1. Produit médian des polynômes. Commençons par transposer la multiplication des polynômes. Cette opération n'est pas linéaire, mais elle le devient lorsqu'on fixe l'un des deux opérandes. Fixons une fois pour toutes un polynôme f dans $\mathbb{K}[X]$ de degré m. Pour $n \geq 0$ quelconque, considérons l'application de multiplication $m_{f,n} : \mathbb{K}[X]_{\leq n} \to \mathbb{K}[X]_{\leq m+n}$ qui à un polynôme g associe le produit fg. Il est aisé de voir que la transposée de cette opération ${}^t m_{f,n} : \mathbb{K}[X]_{\leq m+n} \to \mathbb{K}[X]_{\leq n}$ consiste à extraire les coefficients de $X^m, X^{m+1}, \ldots, X^{m+n}$ du produit d'un polynôme de degré au plus $m+n$ par le polynôme réciproque $\tilde{f}$ de f.

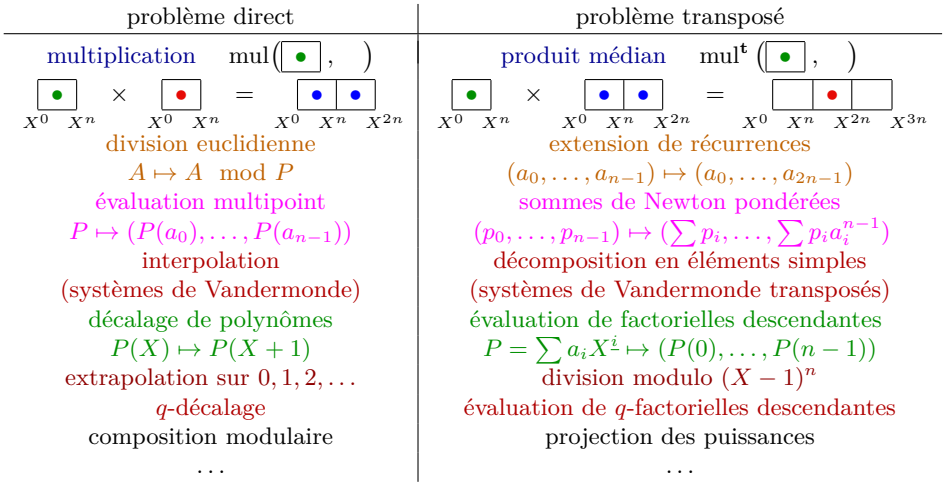


FIGURE 3. « Dictionnaire de Tellegen » pour les polynômes à une variable.

EXEMPLE 3. Par exemple, soit $f = 2 + X + 3X^2$; la matrice de $m_{f,2}$ dans les bases canoniques de $\mathbb{K}[X]_{\leq 2}$ et $\mathbb{K}[X]_{\leq 4}$ est donnée par la matrice de type Toeplitz

$$\begin{bmatrix} 2 & 0 & 0 \\ 1 & 2 & 0 \\ 3 & 1 & 2 \\ 0 & 3 & 1 \\ 0 & 0 & 3 \end{bmatrix}.$$

Sa transposée est

$$\begin{bmatrix} 2 & 1 & 3 & 0 & 0 \\ 0 & 2 & 1 & 3 & 0 \\ 0 & 0 & 2 & 1 & 3 \end{bmatrix}.$$

Cette dernière matrice est la partie médiane de la matrice de Toeplitz suivante

$$\begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 \\ \mathbf{2} & \mathbf{1} & \mathbf{3} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{2} & \mathbf{1} & \mathbf{3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{2} & \mathbf{1} & \mathbf{3} \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix},$$

qui représente le produit du polynôme $\tilde{f} = 3 + X + 2X^2$ par un polynôme h de degré 4. Cette partie médiane donne les coefficients de degré $2, 3, 4$ du produit $\tilde{f}h$. Cela explique pourquoi le produit transposé est appelé *produit médian*.

Le produit médian est une opération importante, qui intervient comme brique de base dans nombre d'algorithmes sur les polynômes et les séries. Son calcul efficace est un problème important à résoudre avant de passer à la transposition d'autres algorithmes plus sophistiqués.

Par exemple, l'utilisation du produit médian permet d'améliorer (d'un facteur constant) l'efficacité de l'opérateur de Newton pour l'inverse de séries formelles, et mène aux optimisations mentionnées en page 46. Cette remarque s'applique également aux autres opérations rapides sur les séries décrites au Chapitre 3. L'accélération se répercute aussi sur la division euclidienne rapide, qui repose sur une inversion de série (Théorème 1 du Chapitre 4).

En vertu du principe de Tellegen (Théorème 1) appliqué aux matrices de Toeplitz, le coût du produit médian est celui du produit fg, à m opérations près. En particulier, si $m = n$, cela permet d'effectuer le calcul de la partie médiane pour le coût d'une multiplication en degré n, au lieu des deux qu'on attendrait naïvement. Il y a plusieurs algorithmes pour multiplier des polynômes, à chacun correspond donc un algorithme de même complexité pour calculer le produit transposé ; cet algorithme transposé peut être obtenu en appliquant le principe de Tellegen, soit dans sa version graphes, soit par transformation de programmes linéaires. Plus exactement, on a le résultat général suivant.

THÉORÈME 2. *Tout algorithme de complexité arithmétique* $\mathsf{M}(n)$ *pour la multiplication polynomiale en degré* n *peut être transformé en un algorithme de complexité arithmétique* $\mathsf{M}(n) + O(n)$ *pour le produit médian en degrés* $(n, 2n)$.

EXERCICE 1. Vérifier le théorème dans le cas de la multiplication naïve des polynômes. Expliciter l'algorithme transposé dans ce cas.

EXEMPLE 4. En Figure 4, à gauche, est représenté un DAG qui calcule le produit de deux polynômes de degré 1 *à la Karatsuba* (en utilisant 3 multiplications au lieu de 4). Par transposition, on déduit un algorithme *à la Karatsuba* pour le produit transposé de deux polynômes de degré 1 et 2. L'algorithme direct utilise 6 opérations arithmétiques (3 additions et 3 multiplications) ; l'algorithme transposé en utilise 7 (4 additions et 3 multiplications). La différence de 1 opération est bien sûr prédite par le Théorème 1.

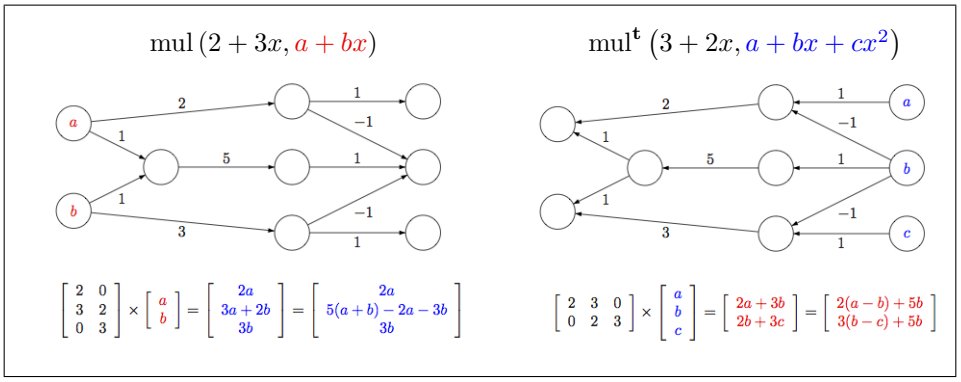


FIGURE 4. Produit direct et transposé *à la Karatsuba*, version DAG.

L'algorithme de Karatsuba et son code transposé sont donnés en Figure 5. Tous les deux ont la même complexité $O(n^{\log(3)})$.

EXEMPLE 5. La matrice de la DFT étant symétrique, à tout algorithme (linéaire) qui la calcule est associé un algorithme transposé de même coût qui la calcule également. Par exemple, à la variante de la DFT décrite au Chapitre 2 (due à Gentleman et Sande) correspond un algorithme de Cooley et Tukey. En figure 6 sont représentés les DAG de ces algorithmes pour une DFT sur 4 points.

4.2. Division avec reste et extension des récurrences à coefficients constants. On peut montrer que le dual du problème de la division avec reste d'un polynôme de degré N par un polynôme fixé de degré n est l'extension des récurrences linéaires à coefficients constants. Étant donnés les n premiers termes d'une suite qui vérifie une récurrence linéaire à coefficients constants d'ordre n, il s'agit de calculer la tranche des N termes suivants.

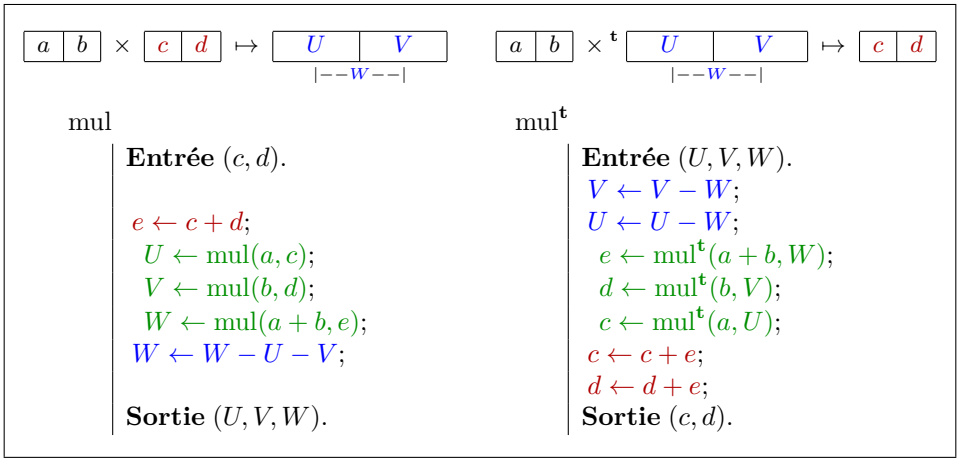


FIGURE 5. Algorithme de Karatsuba et son transposé.

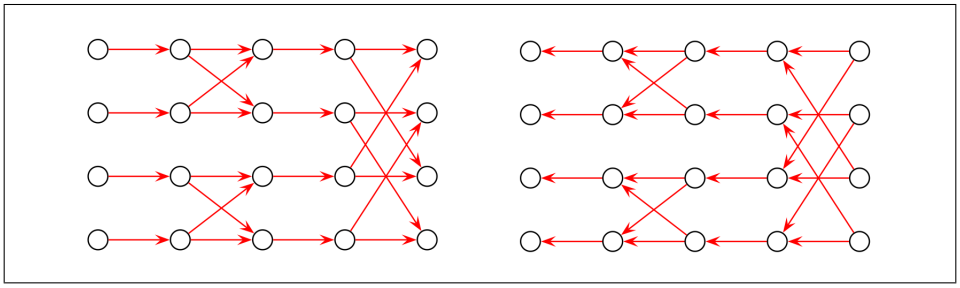


FIGURE 6. Dualité entre deux classes d'algorithmes pour la DFT : à gauche, l'algorithme *decimation-in-time* de Cooley–Tukey, à droite l'algorithme *decimation-in-frequency* de Gentleman–Sande.

Pour une récurrence d'ordre $n = 1$, la preuve est immédiate : étendre une récurrence $u_{n+1} = au_n$, de condition initiale $u_0 = x_0$ revient à calculer la suite des valeurs $x_0, ax_0, \ldots, a^N x_0$ à partir de x_0. Or, on a vu en page 172 que le dual de ce problème est l'évaluation polynomiale en a, autrement dit, la division avec reste modulo $X - a$.

EXERCICE 2. Finir la preuve dans le cas général (n quelconque).

EXERCICE 3. Transposer l'algorithme rapide de division euclidienne donné au Chapitre 4 dans la preuve du Théorème 1 (page 62). En déduire un algorithme de complexité $O(\mathsf{M}(n))$ permettant de calculer les $2n$ premiers termes d'une suite donnée par une récurrence linéaire d'ordre n à coefficients constants et n conditions initiales.

4.3. Évaluation multipoint et interpolation. Comme vu au Chapitre 5, l'évaluation multipoint se traduit en termes matriciels par un produit entre la matrice de Vandermonde associée aux points a_i et le vecteur des coefficients du polynôme P, voir la Figure 7. Ainsi, le problème transposé est la multiplication d'une matrice de Vandermonde transposée par un vecteur, c'est-à-dire, le calcul de sommes de Newton pondérées (on les appelle ainsi car si tous les p_i valent 1, on retrouve les sommes des puissances des a_i).

$$\begin{bmatrix} 1\, a_0 \cdots a_0^n \\ 1\, a_1 \cdots a_1^n \\ \vdots\ \vdots \qquad \vdots \\ 1\, a_n \cdots a_n^n \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} P(a_0) \\ P(a_1) \\ \vdots \\ P(a_n) \end{bmatrix} \quad \text{et} \quad \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a_0 & a_1 & \cdots & a_n \\ \vdots & \vdots & & \vdots \\ a_0^n & a_1^n & \cdots & a_n^n \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} \sum p_i \\ \sum a_i p_i \\ \vdots \\ \sum a_i^n p_i \end{bmatrix}.$$

FIGURE 7. L'évaluation multipoint et sa transposée, les sommes de Newton pondérées.

Il est possible d'exploiter algorithmiquement cette dualité, à l'aide du principe de transposition. L'idée est de proposer d'abord un algorithme rapide pour le calcul de ces sommes de Newton, et ensuite de le transposer.

Pour ce faire, le point de départ est l'observation que la série génératrice des sommes de Newton pondérées

$$\sum_{s\geq 0} \Big(\sum_{i=0}^{n-1} p_i a_i^s \Big) X^{-s-1}$$

est *rationelle* et vaut $Q = B/A$, où

$$A = (X - a_0) \cdots (X - a_{n-1}) \quad \text{et} \quad B = \sum_{i=0}^{n-1} p_i \frac{A}{X - a_i}.$$

D'où l'algorithme suivant : on calcule A et B et on retourne les n premiers coefficients du développement en série de Taylor à l'infini de $Q = B/A$. Le calcul de A se fait en utilisant l'algorithme ArbreSousProduits donné en Figure 1 (page 76), de complexité $\frac{1}{2}\mathsf{M}(n)\log(n) + O(\mathsf{M}(n))$. Celui de B peut se faire par l'algorithme de type « diviser pour régner » représenté en Figure 8 à gauche, de complexité $\mathsf{M}(n)\log(n) + O(\mathsf{M}(n))$. (Une solution équivalente serait de calculer simultanément A et B par l'algorithme SommesFractions en page 77). Enfin, le développement en série de Q s'effectue en $O(\mathsf{M}(n))$ opérations arithmétiques grâce à des itérations de Newton, comme décrit au Chapitre 3.

En résumé, on obtient un algorithme pour le calcul des sommes de Newton pondérées, de complexité

$$\frac{3}{2}\mathsf{M}(n)\log(n) + \mathcal{O}\big(\mathsf{M}(n)\big).$$

Par transposition, on construit un algorithme *de même complexité* pour l'évaluation multipoint. Cet algorithme est représenté graphiquement en Figure 8, à droite. Tout comme l'algorithme EvaluationRapide représenté en Figure 2 (page 76), le nouvel algorithme repose sur une stratégie « diviser pour régner », mais remplace, à chaque niveau de récursion, les divisions avec reste par des produits médians, moins coûteux, d'où le gain d'un facteur constant dans la complexité.

EXERCICE 4. Écrire les détails des deux algorithmes en Figure 8.

EXERCICE 5. Soit $\mathbb{K}$ un corps et soit n un entier. On considère le problème suivant :

> Étant données les valeurs en $0, 1, \ldots, n-1$ d'un polynôme (inconnu) de $\mathbb{K}[X]$ de degré au plus n, calculer les valeurs prises par ce polynôme en $n, n+1, \ldots, 2n-1$.

Déterminer le problème dual, donner un algorithme de complexité $O(\mathsf{M}(n))$ pour ce dernier, et en déduire un algorithme explicite de complexité $O(\mathsf{M}(n))$ pour le problème de départ.

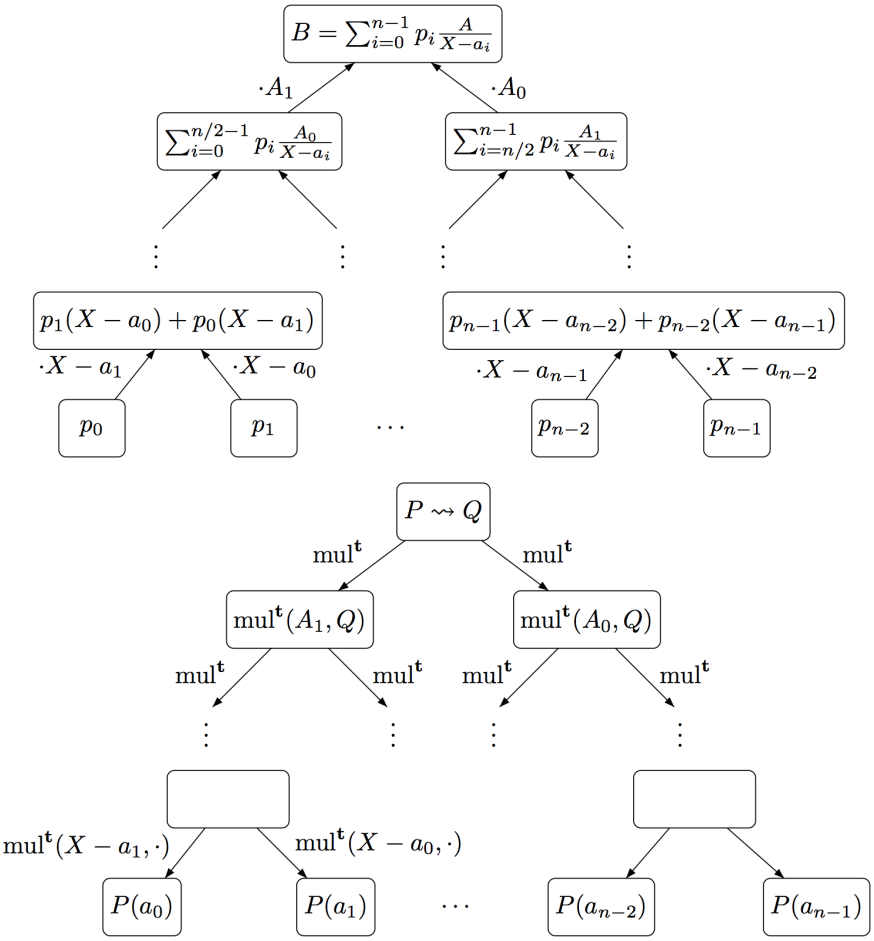


FIGURE 8. Algorithme rapide d'évaluation multipoint par produits médians répétés (à droite), obtenu par transposition d'un algorithme pour les sommes de Newton pondérées (à gauche).

4.4. Bonus : l'évaluation et l'interpolation ont des coûts équivalents. Un dernier exemple d'application du théorème de Tellegen, de nature plus théorique, est la preuve du fait que les problèmes d'évaluation multipoint et d'interpolation polynomiale sont équivalents du point de vue de la complexité.

THÉORÈME 3. *Tout algorithme qui effectue l'évaluation multipoint (resp. l'interpolation) sur une famille fixée de points peut être transformé en un algorithme d'interpolation (resp. d'évaluation multipoint) sur la même famille de points, de même complexité à un nombre constant de multiplications polynomiales près.*

Autrement dit, si $\mathsf{E}(n)$ *et* $\mathsf{I}(n)$ *représentent les complexités de l'évaluation multipoint et de l'interpolation en* n *points, alors :*

$$\mathsf{I}(n) \in O\big(\mathsf{E}(n) + \mathsf{M}(n)\big) \quad \textit{et} \quad \mathsf{E}(n) \in O\big(\mathsf{I}(n) + \mathsf{M}(n)\big).$$

ESQUISSE DE PREUVE. Supposons qu'on dispose d'un algorithme $\mathcal{I}$ de complexité $\mathsf{I}(n)$ pour l'interpolation sur les points $\mathbf{a} = (a_0, \ldots, a_{n-1})$. Le théorème de Tellegen montre qu'on peut construire un algorithme ${}^t\mathcal{I}$ de complexité $\mathsf{I}(n)$ pour *l'interpolation transposée* sur $\mathbf{a}$. On va montrer comment en déduire un algorithme

$\mathcal{E}$ pour l'évaluation multipoint sur $\mathbf{a}$. L'idée est d'utiliser la factorisation matricielle

$$\mathbf{V_a} = ({}^{\mathbf{t}}\mathbf{V_a})^{-1} \cdot \mathbf{H_a}, \quad \text{où} \quad \mathbf{H_a} = \begin{bmatrix} n & \sum_i a_i & \dots & \sum_i a_i^{n-1} \\ \sum_i a_i & \sum_i a_i^2 & \dots & \sum_i a_i^n \\ \vdots & \vdots & & \vdots \\ \sum_i a_i^{n-1} & \sum_i a_i^{n-1} & \dots & \sum_i a_i^{2(n-1)} \end{bmatrix},$$

qui suggère l'algorithme $\mathcal{E}$ suivant :

- Appliquer $\mathcal{I}$ pour calculer $A(X) = \prod_i (X - a_i)$, en complexité $\mathsf{I}(n)$;
- calculer $\mathbf{H_a}$ à partir des $2n-1$ premières sommes de Newton de $A(X)$, par l'algorithme de complexité $\mathcal{O}(\mathsf{M}(n))$ donné en Proposition 4, page 49 ;
- calculer $\mathbf{v} = \mathbf{H_a} \cdot \mathbf{p}$ en $\mathcal{O}(\mathsf{M}(n))$ opérations, comme au Chapitre 10 ;
- retourner $\mathbf{V_a} \cdot \mathbf{p} = {}^t(\mathbf{V_a^{-1}}) \cdot \mathbf{v}$ en $\mathsf{I}(n)$ opérations en utilisant l'algorithme ${}^t\mathcal{I}$.

La preuve du sens inverse est laissée en exercice. □

Exercice 6. Finir la preuve du Théorème 3.

Notes

La présentation de la Section 2 est inspirée de [25]. L'algorithme rapide de l'exercice 3 pour l'extension de récurrences a été donné par Shoup dans [30, 32]. La dualité *division polynomiale* ↔ *extension de récurrences* présentée en Section 4.2 est implicite dans [12, §IV.3]. Elle permet de clarifier le statut de l'algorithme de Shoup : c'est la transposée de l'algorithme de Strassen [34] pour la division euclidienne des polynômes. Ce fait a été mis en évidence dans [9].

L'algorithme présenté en Section 4.3, améliorant (d'un facteur constant) les algorithmes classiques d'évaluation, est dû à [9]. Le principe de transposition est également employé dans [9] pour accélérer (toujours par des facteurs constants) les algorithmes classiques pour l'interpolation et pour son problème dual (la résolution de systèmes de Vandermonde transposés). Les mêmes méthodes s'étendent au cas plus général du théorème des restes chinois [7].

Le Théorème 3 est dû à [8]. La factorisation matricielle utilisée dans sa preuve est tirée de [11].

Une autre application récente du principe de transposition est la suivante [10] : si $(P_\ell)_{\ell \geq 0}$ est une famille de polynômes de $\mathbb{K}[X]$, avec $\deg(P_\ell) = \ell$, dont la série génératrice exponentielle $\sum_{\ell \geq 0} P_\ell(X)/\ell! \cdot Y^\ell$ est de la forme $v(Y) \cdot \exp(X \cdot h(Y))$ [1], alors les conversions entre la base canonique $(X^\ell)_{\ell \geq 0}$ de $\mathbb{K}[X]$ et la base $(P_\ell)_{\ell \geq 0}$ peuvent s'effectuer en $O(\mathsf{M}(n))$ opérations dès lors que $v \bmod Y^n$ peut se calculer en $O(\mathsf{M}(n))$ opérations, et pour une large classe de séries h.

Historique. Le principe de transposition a une histoire longue et tortueuse. Cette technique de transformation des algorithmes linéaires est issue du domaine des circuits électroniques [1, 6, 28] et de la théorie du contrôle [21]. Le principe même remonte aux années 1950 ; on en trouve les traces dans un article de Tellegen [35] sur les réseaux électriques. Il a été généralisé aux filtres digitaux par Fettweis [14], où l'on trouve une version embryonnaire de la version par graphes. Le théorème de Tellegen a été redémontré plusieurs fois, dans divers contextes et degrés de généralité, par Fiduccia [15, 16], Hopcroft et Musinski [20], Knuth et Papadimitriou [26], et par Kaminski, Kirkpatrick et Bshouty [25]. En calcul formel, il a été popularisé dans les travaux de Ben-Or, Tiwari [3], Kaltofen, Canny, Lakshman [11, 22], Shoup [30, 31, 32], Lecerf, Schost [27], Hanrot, Quercia, Zimmermann [19], Zippel [37], von

1. Une telle famille de polynômes est appelée *suite de Sheffer* [29].

zur Gathen et Gerhard [17], ... Il est resté longtemps méconnu, comme le montre cet extrait de l'article [36] de Wiedemann : « I was not able to find an example of a linear operator that is easy to apply but whose transpose is difficult to apply ». Le nom *principe de transposition* semble avoir été introduit par Kaltofen et Shoup dans [24, 33]. Un point de vue légèrement différent sur les aspects historiques liés au principe de transposition peut être trouvé dans [4].

Dans la littérature du calcul formel, c'est surtout son caractère de théorème d'existence qui est exploité : connaissant un algorithme pour une certaine opération linéaire, on en déduit l'existence d'un algorithme de même complexité pour l'opération duale. Un fait mis en évidence plus récemment [9] est que la transposition des programmes peut se faire de manière systématique et (quasi-)automatique. Les algorithmes sont transposés directement, d'une manière similaire à celle utilisée en différentiation automatique [18], mais en tirant profit des spécificités linéaires. Par ailleurs, lorsqu'un problème linéaire doit être résolu efficacement, une démarche naturelle consiste à essayer de trouver un algorithme rapide résolvant le problème dual. Le cas échéant, une solution rapide pour le problème de départ est obtenue en re-transposant cet algorithme. C'est le point de vue adopté en Section 4.3.

Produit médian. Le concept de *produit médian* a été introduit dans l'article [19], qui souligne l'importance de cette *nouvelle opération* sur les polynômes.

En traitement du signal [5, 13], il était déjà connu que, si la multiplication est effectuée à base de DFT, le produit médian en degrés $(n, 2n)$ a la même complexité que le produit en degré n. En effet, il se code matriciellement en un produit matrice–vecteur par une matrice de Hankel de taille n ; or, celle-ci peut être plongée dans une matrice circulante de taille $2n$. Cela implique que dans le modèle de multiplication polynomiale par FFT, un produit médian $(n, 2n)$ peut être effectué en utilisant 3 DFT (deux directes et une inverse) de taille uniquement $2n$ (au lieu de $3n$).

Cette interprétation n'est plus utile dans le modèle de multiplication par l'algorithme de Karatsuba. L'article [19] est le premier à avoir proposé un algorithme de type Karatsuba pour le produit médian.

Lien avec la différentiation automatique. Il a été remarqué dans [11, 23] que le principe de transposition est lié au *mode inverse* en *différentiation automatique* pour le calcul du gradient d'une fonction. En effet, les éléments de ${}^t\mathbf{M} \cdot \mathbf{w}$ sont les dérivées partielles de ${}^t\mathbf{v} \cdot {}^t\mathbf{M} \cdot \mathbf{w}$ par rapport aux coordonnées de $\mathbf{v}$. Ce produit n'est autre que ${}^t\mathbf{w} \cdot \mathbf{M} \cdot \mathbf{v}$, et le théorème de Baur-Strassen [2] montre que l'on peut calculer ces dérivées partielles au prix d'un surcoût linéaire.

Lien avec la théorie de la complexité bilinéaire. Hopcroft et Musinski ont donné dans [20] une *version bilinéaire* de l'algorithme de Tellegen. Étant donné un algorithme bilinéaire qui utilise N multiplications pour le problème, noté (m, n, p), de la multiplication de deux matrices de tailles $m \times n$ et $n \times p$, il est possible d'engendrer cinq algorithmes duaux, chacun utilisant exactement N multiplications, pour les problèmes (n, m, p), (p, m, n), (m, p, n), (n, p, m) et (p, n, m). Ce résultat admet la conséquence utile suivante (mentionnée en page 142) : si l'on peut calculer en N multiplications dans $\mathbb{K}$ le produit de deux matrices quelconques sur $\mathbb{K}$ de tailles $m \times n$ et $n \times p$, alors $\omega \leq 3 \log_{mnp}(N)$.

Dualité pour l'évaluation des monômes à plusieurs variables. Knuth et Papadimitriou [26] ont montré que si $\mathbf{M} = (a_{i,j})$ est une matrice carrée inversible dont les éléments sont des entiers strictement positifs, alors, partant de $\{X_1, \ldots, X_n\}$, le nombre minimum de multiplications pour évaluer les monômes

$$\left\{X_1^{a_{11}} X_2^{a_{12}} \cdots X_n^{a_{1n}},\ X_1^{a_{21}} X_2^{a_{22}} \cdots X_n^{a_{2n}}, \ldots,\ X_1^{a_{n1}} X_2^{a_{n2}} \cdots X_n^{a_{nn}}\right\}$$

est le même que le nombre minimum de multiplications pour évaluer les monômes

$$\left\{X_1^{a_{11}} X_2^{a_{21}} \cdots X_n^{a_{n1}},\ X_1^{a_{12}} X_2^{a_{22}} \cdots X_n^{a_{n2}}, \ldots,\ X_1^{a_{1n}} X_2^{a_{2n}} \cdots X_n^{a_{nn}}\right\}.$$

Cet énoncé peut être vu comme cas particulier du théorème de Tellegen.

Lien entre les algorithmes de Keller-Gehrig et de Storjohann. L'itération de Keller-Gehrig (Chapitre 8, page 137) permettant le calcul de la suite $\mathtt{s} = [x, ax, a^2x, \ldots, a^Nx]$ ($N = 2^k - 1$) est la transposée de l'algorithme « diviser pour régner » pour d'évaluation d'un polynôme $P(x) = a_0 + \cdots + a_N x^N$ en a via la décomposition de type *decimation-in-time* $P(x) = P_0(x^2) + x \cdot P_1(x^2)$. L'algorithme « diviser pour régner » pour l'évaluation de P en a via la décomposition de type *decimation-in-frequency* $P(x) = P_0(x) + x^{\frac{N+1}{2}} P_1(x)$ admet comme dual l'algorithme de Storjohann (Chapitre 11, page 168) pour calculer $\mathtt{s}$.

Bibliographie

[1] Antoniou, A. (1979). *Digital Filters : Analysis and Design*. McGraw-Hill Book Co.

[2] Baur, W. et V. Strassen (1983). « The complexity of partial derivatives ». In : *Theoretical Computer Science*, vol. 22, p. 317–330.

[3] Ben-Or, M. et P. Tiwari (1988). « A deterministic algorithm for sparse multivariate polynomial interpolation ». In : *STOC'88 : ACM Symposium on Theory of Computing*. ACM Press, p. 301–309.

[4] Bernstein, Daniel J. *The transposition principle*. Preprint. URL : http://cr.yp.to/transposition.html.

[5] Bluestein, Leo I. (1968). « A linear filtering approach to the computation of the discrete Fourier transform ». In : *IEEE Northeast Electronics Research and Engineering Meeting*, vol. 10, p. 218–219.

[6] Bordewijk, J. L. (1956). « Inter-reciprocity applied to electrical networks ». In : *Appl. Sci. Res. B.* vol. 6, p. 1–74.

[7] Bostan, A., G. Lecerf, B. Salvy, É. Schost et B. Wiebelt (2004). « Complexity issues in bivariate polynomial factorization ». In : *ISSAC'04 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 42–49.

[8] Bostan, A. et É. Schost (2004). « On the complexities of multipoint evaluation and interpolation ». In : *Theoretical Computer Science*, vol. 329, n°1–3, p. 223–235.

[9] Bostan, Alin, Grégoire Lecerf et Éric Schost (2003). « Tellegen's principle into practice ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par J. R. Sendra. ACM Press, p. 37–44.

[10] Bostan, Alin, Bruno Salvy et Éric Schost (2008). « Power Series Composition and Change of Basis ». In : *ISSAC'08 : International Symposium on Symbolic and Algebraic Computation*. Éd. par David J. Jeffrey. ACM Press, p. 269–276. DOI : 10.1145/1390768.1390806.

[11] Canny, J., E. Kaltofen et L. Yagati (1989). « Solving systems of non-linear polynomial equations faster ». In : *ISSAC'89 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 121–128.

[12] Cerlienco, L., M. Mignotte et F. Piras (1987). « Suites récurrentes linéaires. Propriétés algébriques et arithmétiques ». In : *L'Enseignement Mathématique*. II, vol. 33, p. 67–108.

[13] Chu, Eleanor et Alan George (2000). *Inside the FFT black box*. Serial and parallel fast Fourier transform algorithms. CRC Press, xxii+312 pages.

[14] Fettweis, A. (1971). « A general theorem for signal-flow networks, with applications ». In : *Archiv für Elektronik und Übertragungstechnik*, vol. 25, n°12, p. 557–561.

[15] Fiduccia, C. M. (1972). « On obtaining upper bounds on the complexity of matrix multiplication ». In : *Complexity of computer computations*. IBM Thomas J. Watson Research Center, Yorktown Heights, New York : Plenum, p. 31–40.

[16] — (1973). « On the algebraic complexity of matrix multiplication ». Thèse de doct. Brown Univ., Providence, RI, Center Comput. Inform. Sci., Div. Engin.

[17] Gathen, Joachim von zur et Jürgen Gerhard (1999). *Modern computer algebra*. Cambridge University Press, xiv+753 pages.

[18] Gilbert, J.-C., G. Le Vey et J. Masse (1991). *La différentiation automatique de fonctions représentées par des programmes*. Rapp. tech. RR INRIA 1557.

[19] Hanrot, Guillaume, Michel Quercia et Paul Zimmermann (2004). « The Middle Product Algorithm I ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 14, n°6, p. 415–438.

[20] Hopcroft, J. et J. Musinski (1973). « Duality applied to the complexity of matrix multiplication and other bilinear forms ». In : *SIAM Journal on Computing*, vol. 2, p. 159–173.

[21] Kalman, R. E. (1959). « On the General Theory of Control Systems ». In : *IRE Transactions on Automatic Control*, vol. 4, n°3, p. 481–491.

[22] Kaltofen, E., R. M. Corless et D. J. Jeffrey (2000). « Challenges of symbolic computation : my favorite open problems ». In : *Journal of Symbolic Computation*, vol. 29, n°6, p. 891–919.

[23] Kaltofen, Erich (1993a). « Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems ». In : *Applied algebra, algebraic algorithms and error-correcting codes*. Vol. 673. Lecture Notes in Computer Science. Springer-Verlag, p. 195–212.

[24] — (1993b). « Computational differentiation and algebraic complexity theory ». In : *Workshop Report on First Theory Institute on Computational Differentiation*, p. 28–30.

[25] Kaminski, M., D. G. Kirkpatrick et N. H. Bshouty (1988). « Addition requirements for matrix and transposed matrix products ». In : *Journal of Algorithms*, vol. 9, n°3, p. 354–364.

[26] Knuth, Donald E. et Christos H. Papadimitriou (1981). « Duality in addition chains ». In : *Bulletin of the European Association for Theoretical Computer Science*, vol. 13, p. 2–4.

[27] Lecerf, G. et É. Schost (2003). « Fast Multivariate Power Series Multiplication in Characteristic Zero ». In : *SADIO Electronic Journal on Informatics and Operations Research*, vol. 5, n°1, p. 1–10.

[28] Penfield Jr., P., R. Spence et S. Duinker (1970). *Tellegen's theorem and electrical networks*. The M.I.T. Press, Cambridge, Mass.-London, xv+143 pages.

[29] Roman, Steven (1984). *The umbral calculus*. Vol. 111. Pure and Applied Mathematics. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], x+193 pages.

[30] Shoup, V. (1991). « A Fast Deterministic Algorithm for Factoring Polynomials over Finite Fields of Small Characteristic ». In : *ISSAC'91 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 14–21.

[31] — (1995). « A new polynomial factorization algorithm and its implementation ». In : *Journal of Symbolic Computation*, vol. 20, n°4, p. 363–397.

[32] — (1999). « Efficient computation of minimal polynomials in algebraic extensions of finite fields ». In : *ISSAC'99 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 53–58.

[33] Shoup, Victor (1994). « Fast construction of irreducible polynomials over finite fields ». In : *Journal of Symbolic Computation*, vol. 17, n°5, p. 371–391.

[34] Strassen, V. (1972/73). « Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten ». In : *Numerische Mathematik*, vol. 20, p. 238–251.

[35] Tellegen, B. (1952). « A general network theorem, with applications ». In : *Philips Research Reports*, vol. 7, p. 259–269.

[36] Wiedemann, D. (1986). « Solving sparse linear equations over finite fields ». In : *IEEE Transactions on Information Theory*, vol. IT-32, p. 54–62.

[37] Zippel, R. (1990). « Interpolating polynomials from their values ». In : *Journal of Symbolic Computation*, vol. 9, n°3, p. 375–403.

CHAPITRE 13

Équations différentielles à coefficients séries

Résumé

L'algorithme de Newton dans un cadre différentiel permet de calculer des séries tronquées solutions d'équations différentielles en complexité quasi-optimale. Pour certaines classes particulières importantes d'équations, la complexité peut être encore abaissée.

Comme pour le Chapitre 3 sur les calculs rapides de séries, N sera utilisé pour représenter le nombre de termes à calculer, et « série » sera employé pour « série tronquée à l'ordre N ». L'efficacité des algorithmes sera mesurée par leurs complexités arithmétiques. Nous nous attacherons à expliciter la dépendance de la complexité vis-à-vis des autres paramètres (ordre r de l'équation, degré d des coefficients lorsqu'ils sont polynomiaux). Pour le cas particulier très important des équations et des systèmes différentiels *linéaires*, les résultats de complexité de ce chapitre sont présentés au tableau 1, dans lequel il faut comparer les complexités aux tailles des entrées et des sorties. Par comparaison, la méthode la plus directe (les coefficients indéterminés), est quadratique en N pour le cas où les coefficients sont des séries.

Tous les algorithmes sont énoncés pour des coefficients dans un anneau $\mathbb{A}$ (avec en vue le cas de coefficients polynomiaux par exemple). Dans tout ce chapitre nous ferons *l'hypothèse supplémentaire que* $2, 3, \ldots, N$ *sont inversibles dans* $\mathbb{A}$. Pour les cas très importants d'équations différentielles linéaires et à coefficients polynomiaux ou constants (et non pas séries), des algorithmes encore plus rapides sont présentés en fin de chapitre.

1. Équations différentielles d'ordre 1

Le cas de l'ordre 1 est plus simple que le cas général. Sa présentation sert d'introduction à la section suivante, où les techniques employées ici seront généralisées.

Problème (entrée, sortie)	coefficients séries	coefficients polynômes	coefficients constants	taille résultat
(équation, base)	$O(r^2\mathsf{M}(N))$	$O(dr^2N)$	$O(rN)$	rN
(équation, 1 solution)	$O(r\,\mathsf{M}(N)\log N)$	$O(drN)$	$O\left(\mathsf{M}(r)\frac{N}{r}\right)$	N
(système, base)	$O(r^2\mathsf{M}(N))$	$O(dr^\theta N)$	$O(r\mathsf{M}(r)N)$	r^2N
(système, 1 solution)	$O(r^2\,\mathsf{M}(N)\log N)$	$O(dr^2N)$	$O(\mathsf{M}(r)N)$	rN

TABLE 1. Complexité de la résolution d'équations et de systèmes différentiels linéaires pour $N \gg r$.

1.1. L'équation linéaire homogène du premier ordre. Cette équation,

$$y' = A(X)y,$$

où $A(X)$ est une série, admet la solution

$$y(X) = y(0)\exp\left(\int A(X)\right). \tag{1}$$

Le calcul utilisant cette formule est dominé par celui de l'exponentielle, donc en $O(\mathsf{M}(N))$ par les algorithmes du Chapitre 3.

1.2. L'équation linéaire inhomogène du premier ordre. Il s'agit de l'équation

$$y' = A(X)y + B(X),$$

où A et B sont des séries. La méthode de la variation de la constante consiste à poser

$$y(X) = f(X)\exp\left(\int A(X)\right),$$

et à injecter cette expression dans l'équation pour obtenir que f vérifie

$$f'(X) = B(X)\exp\left(-\int A(X)\right).$$

L'ensemble du calcul de y est donc encore borné par $O(\mathsf{M}(N))$ opérations.

1.3. Le cas non-linéaire. Ces préliminaires avaient pour but de prouver le résultat plus général suivant.

Proposition 1. *Soit $F(X,Y) \in \mathbb{A}[[X,Y]]$ une série bivariée telle que, pour toute série $s(X) \in \mathbb{A}[[X]]$, les N premiers termes de $F(X,s(X))$ et de $\frac{\partial F}{\partial y}(X,s(X))$ se calculent en $O(\mathsf{L}(\mathsf{N}))$ opérations. Alors, l'équation différentielle*

$$y' = F(X,y), \qquad y(0) = a,$$

admet une série formelle solution, et ses N premiers termes peuvent être calculés en $O(\mathsf{L}(N) + \mathsf{M}(N))$ opérations.

L'énoncé de cette proposition en termes d'une fonction L permet de l'adapter à différents besoins : dans le cas le pire, il est possible d'invoquer l'algorithme de Brent et Kung du Chapitre 3 pour avoir $\mathsf{L}(N) = O(N\sqrt{N\log N}\,\mathsf{M}(N))$ qui domine alors la complexité de la résolution. Cependant, pour des équations plus simples, la composition avec F et sa dérivée pourra s'effectuer en $O(\mathsf{M}(N))$. Ce sera le cas par exemple si F s'obtient à l'aide d'un nombre constant de sommes, d'exponentielles, de logarithmes, et de puissances.

Démonstration. L'idée de la preuve repose sur la méthode de Newton, appliquée cette fois-ci à un opérateur :

$$\Phi : y \mapsto y' - F(X,y).$$

Le principe consiste à linéariser l'opérateur Φ au voisinage d'une approximation et à en annuler la partie linéaire, pour obtenir une nouvelle approximation. À partir de

$$\Phi(y+h) = \Phi(y) + h' - \frac{\partial F}{\partial y}(X,y)h + O(h^2),$$

il s'agit donc de calculer un incrément h solution de l'équation *linéaire* inhomogène du premier ordre

$$h' = \frac{\partial F}{\partial y}(X,y)h - \Phi(y),$$

qui peut être résolue par la méthode de la section précédente. On déduit l'itération suivante

$$y_{k+1} := y_k + h_k,$$
$$h_k := \exp\left(\int \frac{\partial F}{\partial y}(X, y_k)\right) \int \left(F(X, y_k) - y_k'\right) \exp\left(-\int \frac{\partial F}{\partial y}(X, y_k)\right) \bmod X^{2^{k+1}}.$$

Il reste à prouver la convergence quadratique. Comme pour le théorème sur l'itération de Newton sur les séries du Chapitre 3 (p. 47), la preuve se fait par récurrence sur k en prouvant simultanément $\Phi(y_k) = O(X^{2^k})$ et $h_k = O(X^{2^k})$. Les calculs sont les mêmes et sont donc laissés en exercice.

Pour obtenir le résultat de complexité, il suffit d'observer que l'itération requiert à chaque étape la résolution d'une équation différentielle linéaire inhomogène du premier ordre et d'invoquer le théorème « diviser pour régner ». □

2. Équations différentielles linéaires d'ordre supérieur et systèmes d'ordre 1

L'équation différentielle d'ordre r

$$a_r(X)y^{(r)}(X) + \cdots + a_1(X)y'(X) + a_0(X)y(X) = 0 \tag{2}$$

où les coefficients a_i sont des séries en X, avec $a_r(0) \neq 0$, peut être récrite comme une équation d'ordre 1

$$Y' = A(X)Y \tag{3}$$

en prenant pour Y le vecteur de séries $(y(X), \ldots, y^{(r-1)}(X))$ et A une matrice (compagnon) de séries en X.

À l'inverse, un système (3) induit une relation de dépendance linéaire à coefficients des séries entre $Y, Y', Y'', \ldots, Y^{(r)}$ si r est la dimension de A (on a $r+1$ vecteurs en dimension r). Le vecteur Y et chacun de ses coefficients vérifient donc une équation de type (2).

Résoudre un problème est donc équivalent à résoudre l'autre. Dans certains cas, exploiter le caractère compagnon de la matrice induite par (2) mène à une meilleure complexité pour le cas des équations.

Une différence importante avec le cas des équations d'ordre 1 est que nous avons, pour les équations de type (2) comme pour les systèmes (3) deux problèmes à considérer :

- calculer une base des séries solutions ;
- étant données des conditions initiales, calculer *la* série solution correspondante.

Les complexités de ces problèmes sont liées : si l'on sait résoudre le second pour un coût C, alors on sait résoudre le premier pour un coût borné par $r\mathsf{C}$. Si l'on sait résoudre le premier, alors une combinaison linéaire des solutions permet de résoudre le second en au plus rN opérations supplémentaires.

Il est cependant utile de considérer les quatre problèmes (système ou équation, base ou solution unique) car la structure de chacun des problèmes permet de concevoir des algorithmes plus efficaces que n'en donnent les conversions directes d'un problème à l'autre.

2.1. Une méthode par « diviser pour régner ». L'équation à résoudre à précision X^N est ici

$$Y' - AY = B, \quad Y(0) = v,$$

où A est une matrice de séries formelles. L'idée est de résoudre d'abord à précision moitié et de déterminer puis résoudre l'équation satisfaite par le reste. La condition

initiale est d'abord traitée séparément en écrivant $Y = v + XU$ et en l'injectant dans l'équation, ce qui mène à

$$XU' + (I - XA(X))U = C, \quad C = B + A(X)v.$$

Soient maintenant $d < N$ et $U = U_0 + X^d U_1$ où U_0 est un polynôme de degré au plus $d-1$ en X, l'équation satisfaite par U donne :

$$XU_1' + ((d+1)I - XA(X))U_1 = -X^{-d}(XU_0' + (I - XA(X))U_0 - C).$$

Si U_0 est une solution à précision d, le membre droit de l'équation est bien un polynôme. L'application du paradigme « diviser pour régner » amène donc naturellement à considérer l'équation plus générale

$$XY' + (pI - XA)Y = R,$$

où R est une série, A une matrice de séries et $p \in \{1, \dots, N\}$.

L'algorithme récursif est donc le suivant :

DiviserPourRégner(A, p, s)

Entrée : $A_0, \dots, A_{N-p}$ dans $\mathcal{M}_{r\times r}(\mathbb{A})$, $A = \sum A_i X^i$, $s_0, \dots, s_{N-p}$ dans $\mathcal{M}_{r\times \ell}(\mathbb{A})$, $s = \sum s_i X^i$, $p \in \{1, \dots, N\}$.

Sortie : y dans $\mathcal{M}_{r\times \ell}(\mathbb{A})[X]$ tel que $\deg_X y \leq N - p$ et $Xy' + (pI - XA)y = s + O(X^{N-p+1})$.

1. $m \leftarrow N - p$
2. Si $m = 1$ alors renvoyer $p^{-1}s(0)$
   sinon
   $d \leftarrow \lfloor m/2 \rfloor$
   $y_0 \leftarrow$ DiviserPourRégner$(A, p + d, s \bmod X^d)$
   $R \leftarrow [s - Xy_0' - (pI - XA)y_0]_d^m$
   $y_1 \leftarrow$ DiviserPourRégner$(A, p + d, R)$
   renvoyer $y_0 + X^d y_1$

La notation $[s]_d^m$ désigne le quotient de la division euclidienne de $s \bmod X^m$ par X^d.

Cet algorithme est invoqué par l'algorithme de résolution suggéré ci-dessus et dont voici une version détaillée.

SolDPR(A, N, B, v)

Entrée : $A_0, \dots, A_N$ dans $\mathcal{M}_{r\times r}(\mathbb{A})$, $A = \sum A_i X^i$, $B_0, \dots, B_N$ et v dans $\mathcal{M}_{r\times \ell}(\mathbb{A})$, $B = \sum B_i X^i$.

Sortie : $y = \sum_{i=0}^{N} y_i X^i$ dans $\mathcal{M}_{r\times \ell}(\mathbb{A})[X]$ tel que $y' - Ay = B$ et $y(0) = v$.

Renvoyer $v + X$DiviserPourRégner$(A, 1, B + Av)$

Les résultats de complexité se déduisent de cette méthode pour différentes valeurs de A et de ℓ. En particulier, le cas des équations a une meilleure complexité que le cas général d'un système d'ordre 1 car la matrice correspondante est une matrice compagnon. Le produit d'une telle matrice par un vecteur ne coûte que r opérations au lieu de r^2 dans le cas général.

THÉORÈME 1 (Diviser pour régner pour les équations différentielles). *Étant donnés les N premiers coefficients de $A \in \mathcal{M}_{r\times r}(\mathbb{A}[[X]])$, de $B \in \mathcal{M}_{r\times \ell}(\mathbb{A}[[X]])$ ainsi que des conditions initiales $v \in \mathcal{M}_{r\times \ell}(\mathbb{A})$, l'algorithme* SolDPR *calcule l'unique solution de*

$$Y' - AY = B + O(X^N), \qquad Y(0) = v$$

en un nombre d'opérations dans $\mathbb{A}$ borné par

1. $O(r^2 \ell \mathsf{M}(N) \log N)$ *en général ;*

2. $O(r\ell\mathsf{M}(N)\log N)$ *si* A *est une matrice compagnon.*

Il est possible d'améliorer les estimations de complexité pour cet algorithme lorsque $\ell = r$ (calcul de toute une base des solutions), mais dans ce cas, il vaut mieux employer les algorithmes de la section suivante, qui sont plus efficaces.

Démonstration. Il suffit de prouver le résultat pour N une puissance de 2, et le cas général s'en déduit par les hypothèses habituelles sur la fonction de multiplication M.

Les opérations de troncature ne demandent pas de calcul dans $\mathbb{A}$. Outre les deux appels récursifs, le calcul demande une dérivation (linéaire), un produit par p fois l'identité (linéaire) et un produit par A. La complexité $C(N)$ vérifie donc

$$C(N) = 2C(N/2) + \lambda\ell N + \mathsf{Mult}(A, V, N),$$

où $\mathsf{Mult}(A, V, N)$ désigne le coût du produit de la matrice A par la matrice $r \times \ell$ de séries à précision N. La conclusion s'obtient par le théorème « diviser pour régner » en observant les complexités de base pour $\mathsf{Mult}(A, V, N)$. □

2.2. L'itération de Newton matricielle. Comme dans la preuve de la Proposition 1, le point de départ de l'algorithme consiste à appliquer l'itération de Newton à l'opérateur dont on cherche les solutions, en linéarisant au voisinage d'une solution approchée. L'opérateur

$$Y \mapsto Y' - A(X)Y$$

étant linéaire, il est son propre linéarisé. Formellement, l'itération de Newton s'écrit donc

$$Y_{k+1} = Y_k - U_k, \quad U_k' - AU_k = Y_k' - AY_k.$$

Lorsque Y_k est une solution approchée, le membre droit de l'équation en U_k a une valuation strictement positive, et la solution U_k cherchée doit avoir aussi une telle valuation. Une telle solution est obtenue par la méthode de la variation de la constante *si l'on dispose d'une base des solutions*, c'est-à-dire dans notre cas si Y_k est une matrice $r \times r$ avec $\det Y_k(0) \neq 0$. Dans ce cas, la méthode de la variation de la constante suggère de poser $U_k = Y_k T$, ce qui donne

$$U_k' - AU_k = Y_k T' + \underbrace{AY_k T - AY_k T}_{0} = Y_k' - AY_k$$

d'où finalement

$$U_k = Y_k \int Y_k^{-1}(Y_k' - AY_k).$$

Cette méthode requiert le calcul de l'inverse d'une base de solution, inverse qui peut être calculé simultanément par l'itération de Newton :

$$Z_{k+1} = Z_k + Z_k(I - Y_k Z_k).$$

Une version précise de l'algorithme, avec les ordres de troncature, est donnée en Figure 1.

Lemme 1 (Correction de l'algorithme). *Soit* m *un entier pair, soient* y *et* z *dans* $\mathcal{M}_{r\times r}(\mathbb{A}[X])$ *tels que*

$$I - yz = O(X^{m/2}), \quad y' - Ay = O(X^m).$$

Soient ensuite Y *et* Z *définis par*

$$Z := z(2I - yz) \bmod X^m, \quad Y := y\left(I - \int Z(y' - Ay)\right) \bmod X^{2m}.$$

Alors Y *et* Z *vérifient*

$$I - YZ = O(X^m), \quad Y' - AY = O(X^{2m-1}).$$

SolSysDiffHom(A, N, Y_0)

Entrée : $Y_0, A_0, \dots, A_{N-2}$ dans $\mathcal{M}_{r\times r}(\mathbb{A})$, $A = \sum A_i X^i$.

Sortie : $Y = \sum_{i=0}^{N-1} Y_i X^i$ dans $\mathcal{M}_{r\times r}(\mathbb{A})[X]$ tel que $Y' = AY + O(X^{N-1})$, et $Z = Y^{-1} + O(X^{N/2})$.

1. $Y \leftarrow (I + XA_0)Y_0$
2. $Z \leftarrow Y_0^{-1}$
3. $m \leftarrow 2$
4. tant que $m \leq N/2$ faire
   $Z \leftarrow Z + Z(I_r - YZ) \bmod X^m$
   $Y \leftarrow Y - Y\left(\int Z(Y' - A \bmod X^{2m-1}Y)\right) \bmod X^{2m}$
   $m \leftarrow 2m$
5. renvoyer Y, Z

FIGURE 1. Résolution de $Y' = A(X)Y$, $Y(0) = Y_0$ par itération de Newton.

DÉMONSTRATION. D'après l'hypothèse sur $y' - Ay = O(X^m)$, $Y = y + O(X^m)$ et donc la convergence de l'inversion est donnée par

$$\begin{aligned} I - YZ &= I - y\big(z + z(I - yz)\big) + O(X^m) \\ &= (I - yz) - yz(I - yz) + O(X^m) \\ &= (I - yz)^2 + O(X^m) = O(X^m). \end{aligned}$$

La seconde propriété s'obtient en injectant la définition de Y dans $Y' - AY$ (on pose $Q = \int Z(y' - Ay)$) :

$$\begin{aligned} Y' - AY &= y' + y'Q - Ay - AyQ - yZ(y' - Ay) + O(X^{2m}), \\ &= (I - yZ)(y' - Ay) - (y' - Ay)Q + O(X^{2m}), \\ &= O(X^m)O(X^m) + O(X^m)O(X^{m-1}) + O(X^{2m}) = O(X^{2m-1}). \end{aligned}$$

□

Comme d'habitude, l'application du théorème « diviser pour régner » mène au résultat de complexité.

THÉORÈME 2 (Newton pour les systèmes différentiels linéaires). *L'algorithme* SolSysDiffHom *calcule les N premiers termes d'une base de solutions de $Y' = AY$ en $O(\mathsf{MM}(r, N))$ opérations dans $\mathbb{A}$.*

La notation $\mathsf{MM}(r, N)$ représente la complexité du produit de matrices $r \times r$ de polynômes de degré au plus N ; des bornes non triviales sont données au Chapitre 5 :

$$\mathsf{MM}(r, N) = O(r^\theta N + r^2 \mathsf{M}(N)).$$

EXERCICE 1. L'exponentielle d'une série est obtenue en $O(\mathsf{M}(N))$ opérations par cet algorithme lorsque $r = 1$. Vérifier que la constante est meilleure que celle de l'algorithme du Chapitre 3.

EXERCICE 2. Le cas d'un système *inhomogène* $Y' = AY + B$, où B est une matrice de séries, s'obtient à partir de l'algorithme précédent par variation de la constante. Étudier les précisions requises dans les troncatures, et donner le résultat de complexité.

3. Cas particuliers

3.1. Équations différentielles linéaires à coefficients polynomiaux. Pour ces équations, la méthode des coefficients indéterminés donne une complexité linéaire en N. L'explication tient à une propriété vue au Chapitre 14 : les solutions séries de ces équations ont des coefficients qui vérifient des récurrences linéaires.

Les résultats du Chapitre 15 sur les récurrences linéaires à coefficients polynomiaux s'appliquent alors, et en particulier les N premiers coefficients d'une solution s'obtiennent en $O(drN)$ opérations si d est une borne sur le degré des q_i. Le Chapitre 16 revient sur ces questions.

Le cas matriciel se traite de la même façon, la récurrence ci-dessus étant alors à coefficients matriciels. Les complexités sont de même nature, avec en outre la possibilité d'utiliser un produit rapide de matrices dans le cas du calcul d'une base de solutions. Les résultats correspondants sont donnés dans la table 1.

3.2. Équations différentielles linéaires à coefficients constants. Dans le cas où les coefficients de l'équation ou du système sont constants, la formule (1) s'applique encore et devient :

$$y(X) = \exp(XA)y(0).$$

Pour étudier les propriétés de cette solution, il est usuel de faire intervenir la forme de Jordan de la matrice, mais cette forme ne se prête pas facilement au calcul.

Un algorithme efficace est obtenu en exploitant la *transformée de Laplace formelle*. Si $S = s_0 + s_1X + s_2X^2 + \dots$ est une série, cette transformée est définie par

$$\mathcal{L}S = s_0 + 1!s_1X + 2!s_2X^2 + \dots.$$

Les troncatures de $\mathcal{L}S + O(X^N)$ et de la transformée inverse $\mathcal{L}^{-1}S + O(X^N)$ se calculent en N opérations. Cette transformation simplifie les systèmes différentiels linéaires à coefficients constants en les rendant linéaires non-différentiels. Ceci découle de

$$\begin{aligned}\mathcal{L}(y) &= \mathcal{L}((I + XA + \frac{1}{2!}X^2A^2 + \dots)y(0)) \\ &= (I + XA + X^2A^2 + \dots)y(0) \\ &= (I - XA)^{-1}y(0).\end{aligned}$$

Les vecteurs solutions ont donc des transformées de Laplace dont les coordonnées sont rationnelles. En outre, d'après les formules de Cramer, le numérateur et le dénominateur de ces fractions ont des degrés bornés par l'ordre r du système. L'algorithme de la Figure 3.2 s'en déduit.

L'étape 1 est effectuée par la méthode naïve en $O(r^3)$ opérations. Une méthode plus efficace, en $O(\mathsf{MM}(r)\log r)$ opérations, est à la base de l'algorithme de Keller-Gehrig pour le calcul du polynôme caractéristique d'une matrice présenté au Chapitre 8. L'étape 2 (a) se ramène à un calcul d'algèbre linéaire en posant des coefficients indéterminés pour les numérateurs et dénominateurs. Là aussi, le coût peut être diminué en faisant appel au calcul d'approximants de Padé (Chapitre 7), mais cette partie du calcul ne dépend pas de N. L'étape 2 (b) utilise l'algorithme de développement de fractions rationnelles du Chapitre 4 (Théorème 4, page 66), et c'est là que se concentre le gain en complexité.

4. Extensions

4.1. Composer se ramène à résoudre. Le seul algorithme du Chapitre 3 dont la complexité n'est pas quasi-optimale est l'algorithme de composition.

CoefficientsConstants(A, v, N)

Entrée : A dans $\mathcal{M}_{r\times r}(\mathbb{A})$, $v \in \mathcal{M}_{r\times 1}(\mathbb{A})$.

Sortie : $Y = \sum_{i=0}^{N-1} Y_i X^i$ dans $\mathcal{M}_{r\times r}(\mathbb{A})[X]$ tel que $Y' = AY + O(X^{N-1})$.

1. Calculer les vecteurs
   $v, Av, A^2v, A^3v, \ldots, A^{2r}v$;
2. Pour tout $j = 1, \ldots, r$:
   (a) reconstruire la fraction rationnelle ayant pour développement en série $\sum (A^i v)_j X^i$;
   (b) développer cette fraction en série à précision X^N en $O(N\,\mathsf{M}(r)/r)$ opérations ;
   (c) reconstruire le développement de y à partir de celui de z, en $O(N)$ opérations.

Dans les applications où l'on connaît une équation $\Phi(X, f, f', \ldots, f^{(m)}) = 0$ vérifiée par la série f il est parfois possible de calculer la série $h = f \circ g$ efficacement *sans passer par l'algorithme de composition* en remarquant qu'elle vérifie une équation du même ordre. Cette équation est obtenue en remplaçant X par g dans Φ et en composant les dérivées. Si Φ ne fait intervenir que des polynômes ou des fractions rationnelles, le résultat de cette opération a pour coefficients des séries, ce qui mène assez généralement à une complexité en $O(\mathsf{M}(N))$ opérations en utilisant les algorithmes de ce chapitre.

Exemple 1. Pour calculer le développement de $h = \tan(f)$, il est possible de calculer en $O(\mathsf{M}(N))$ ceux de $\sin(f)$ et $\cos(f)$ (via l'exponentielle) et de diviser, mais il est aussi possible d'observer que h vérifie l'équation $h' = 1 + h^2 f'$, et d'utiliser les méthodes ci-dessus. Il faut ensuite comparer les constantes dans les $O(\cdot)$ (ou deux implantations !) pour déterminer laquelle des deux méthodes est préférable.

Exercice 3. Les polynômes de Legendre sont définis par

$$P_n(X) = \frac{1}{2^n \cdot n!} \left((X^2 - 1)^n\right)^{(n)}.$$

Montrer que $P_n(X)$ vérifie l'équation différentielle

$$\frac{d}{dX}\left[(1 - X^2)\frac{d}{dX}P_n(X)\right] + n(n+1)P_n(X) = 0.$$

En déduire que pour toute série de terme constant nul $F \in \mathbb{Q}[[X]]$, et pour tout $N \geq 0$, la série composée $P_N \circ F \bmod X^N$ peut se calculer en $O(\mathsf{M}(N))$ opérations.

4.2. Systèmes non-linéaires. La linéarisation effectuée dans le cas des équations d'ordre 1 se généralise aux systèmes. L'énoncé devient alors le suivant.

Théorème 3. *Soient $\phi_1, \ldots, \phi_r$ r séries de $\mathbb{A}[[X, Y_1, \ldots, Y_r]]$, telles que pour tout r-uplet de séries $(s_1(X), \ldots, s_r(X)) \in \mathbb{A}[[X]]^r$, les N premiers termes des $\phi_i(X, s_1, \ldots, s_r)$ et de $\mathrm{Jac}(\phi)(X, s_1(X), \ldots, s_r(X))$ puissent être calculés en $O(\mathsf{L}(\mathsf{N}))$ opérations dans $\mathbb{A}$. On suppose en outre que $\mathsf{L}(\mathsf{n})/\mathsf{n}$ est une suite croissante. Alors, le système différentiel*

$$\begin{cases} y_1'(t) = \varphi_1(t, y_1(t), \ldots, y_r(t)), \\ \qquad \vdots \\ y_r'(t) = \varphi_r(t, y_1(t), \ldots, y_r(t)), \end{cases}$$

avec conditions initiales $(y_1, \ldots, y_r)(0) = v$ *admet une solution série formelle, et ses* N *premiers termes peuvent être calculés en*

$$O(\mathsf{L}(\mathsf{N}) + \min(\mathsf{MM}(\mathsf{r}, \mathsf{N}), \mathsf{r}^2\mathsf{M}(\mathsf{N}) \log \mathsf{N})).$$

Le symbole $\mathrm{Jac}(\phi)$ désigne la matrice jacobienne : son coefficient sur la ligne i et la colonne j vaut $\partial\phi_i/\partial y_j$.

L'intérêt d'énoncer le théorème à l'aide de la fonction L est le même que dans le cas des équations.

Démonstration. Il s'agit d'une généralisation de la Proposition 1. La linéarisation de $\Phi : Y \mapsto Y' - \varphi(Y)$ au voisinage d'une solution approchée y s'obtient en écrivant :

$$\Phi(y + h) = \Phi(y) + h' + \mathrm{Jac}(\phi)(y)h + O(h^2).$$

L'équation à résoudre pour une itération de Newton est donc le système linéaire inhomogène

$$h' = \mathrm{Jac}(\phi)(y)h - (y' - \phi(y)).$$

Si h est un vecteur solution de ce système à précision $2m$ et y une solution de Φ à précision m, ces équations montrent que $y + h$ est solution à précision $2m$. Le cas non-linéaire est ainsi ramené au cas linéaire. □

Notes

Les résultats de la Section 1 sont tirés de [2]. Une bonne partie des résultats de ce chapitre est tirée de [1]. La technique de la Section 2.2 dans le cas particulier de l'exponentielle d'une série est due à [3]. L'algorithme « diviser pour régner » de la Section 2.1 se trouve au moins implicitement dans [4]. Des algorithmes plus récents se trouvent dans [5].

Bibliographie

[1] Bostan, A., F. Chyzak, F. Ollivier, B. Salvy, É. Schost et A. Sedoglavic (2007). « Fast computation of power series solutions of systems of differential equations ». In : *SODA'07 : ACM-SIAM Symposium on Discrete Algorithms*. SIAM, p. 1012–1021.

[2] Brent, R. P. et H. T. Kung (1978). « Fast algorithms for manipulating formal power series ». In : *Journal of the ACM*, vol. 25, n°4, p. 581–595.

[3] Hanrot, Guillaume, Michel Quercia et Paul Zimmermann (2004). « The Middle Product Algorithm I ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 14, n°6, p. 415–438.

[4] Hoeven, Joris van der (2002). « Relax, but don't be too lazy ». In : *Journal of Symbolic Computation*, vol. 34, n°6, p. 479–542.

[5] — (2010). « Newton's method and FFT trading ». In : *Journal of Symbolic Computation*, vol. 45, n°8, p. 857–878.

Troisième partie

Équations différentielles et récurrences linéaires

CHAPITRE 14

Séries différentiellement finies

Résumé

Les séries différentiellement finies se manipulent plus rapidement que les séries arbitraires. L'équation différentielle linéaire les définissant fournit une structure de données adaptée sur laquelle plusieurs opérations utiles sont algorithmiques. En particulier, la multiplication de deux séries différentiellement finies s'effectue en temps linéaire.

Les séries différentiellement finies (c'est-à-dire solutions d'équations différentielles linéaires à coefficients polynomiaux) ont des coefficients qui satisfont une récurrence linéaire, ce qui permet d'en calculer les N premiers en $O(N)$ opérations, donc plus vite que la plupart des autres séries. Il est de ce fait crucial de reconnaître les séries qui sont différentiellement finies et de disposer des équations différentielles les définissant. De plus, les coefficients des séries différentiellement finies forment des suites qui sont appelées polynomialement récurrentes, dont l'algorithmique est évidemment étroitement liée à celle des séries différentiellement finies.

L'importance de ces séries et suites provient en grande partie de leur omniprésence dans les applications. Ainsi, le *Handbook of Mathematical Functions*, référence importante en physique, chimie et mathématiques appliquées, comporte environ 60% de fonctions solutions d'équations différentielles linéaires ; de même, les suites polynomialement récurrentes forment environ un quart des plus de 100 000 suites référencées dans la version en ligne de l'*Encyclopedia of Integer Sequences* de N. Sloane.

Enfin, l'algorithmique de ces séries permet *la preuve automatique d'identités*, présentée dans ce chapitre en présence d'une seule variable. La généralisation à plusieurs variables est traitée dans les chapitres de la cinquième partie.

1. Équations différentielles et récurrences

1.1. Définitions. Si $\mathbb{K}$ désigne un corps, l'anneau des séries formelles à coefficients dans $\mathbb{K}$ est noté $\mathbb{K}[[X]]$. Ses principales propriétés ont été présentées au Chapitre 3. Son corps des fractions, noté $\mathbb{K}((X))$, est égal à $\mathbb{K}[[X]][1/X]$. Ses éléments sont appelés des séries de Laurent formelles. C'est une algèbre non seulement sur $\mathbb{K}$, mais aussi sur le corps des fractions rationnelles $\mathbb{K}(X)$.

Definition 1. Une série formelle $A(X)$ à coefficients dans un corps $\mathbb{K}$ est dite *différentiellement finie* (ou D-finie) lorsque ses dérivées successives A, A', …, engendrent un espace vectoriel de dimension finie sur le corps $\mathbb{K}(X)$ des fractions rationnelles.

De manière équivalente, cette série est solution d'une équation différentielle linéaire à coefficients dans $\mathbb{K}(X)$: si c'est le cas, alors l'équation différentielle permet de récrire toute dérivée d'ordre supérieur à celui de l'équation en termes des dérivées d'ordre moindre (en nombre borné par l'ordre). À l'inverse, si l'espace est de dimension finie, alors pour m suffisamment grand, $A, A', \ldots, A^{(m)}$ sont liées, et une relation de liaison entre ces dérivées est une équation différentielle linéaire.

EXEMPLE 1. De nombreuses séries classiques sont de ce type : $\exp(X)$, $\log(1+X)$, $(1+X)^a$, $\sin X$, $\sinh X$, $\cos X$, $\cosh X$, $\arcsin X$, $\arccos X$, $\arctan X$,..., les séries hypergéométriques généralisées, et d'autres fonctions classiques de la physique mathématique.

DEFINITION 2. Une suite $(a_n)_{n\geq 0}$ d'éléments d'un corps $\mathbb{K}$ est appelée suite *polynomialement récurrente* (ou P-récurrente) si elle satisfait à une récurrence de la forme

$$p_d(n)a_{n+d} + p_{d-1}(n)a_{n+d-1} + \cdots + p_0(n)a_n = 0, \qquad n \geq 0, \tag{1}$$

où les p_i sont des polynômes de $\mathbb{K}[X]$.

Dans la suite, $\mathbb{K}$ aura toujours caractéristique nulle. On peut donc penser sans rien perdre aux idées à $\mathbb{K} = \mathbb{Q}$.

On pourrait aussi donner une définition en termes d'espaces vectoriels, mais pour éviter les problèmes dus aux pôles entiers des coefficients rationnels, il faut alors considérer les *germes* de suites à l'infini, c'est-à-dire les classes d'équivalences de suites pour la relation « être égal à partir d'un certain rang ».

EXEMPLE 2. Outre les solutions de récurrences linéaires à coefficients constants comme les nombres de Fibonacci, des suites classiques de ce type sont : $n!$, $1/n!$, $\binom{2n}{n}$, les nombres harmoniques H_n, les nombres de Catalan, de Motzkin,...

EXEMPLE 3. Le cas particulier des récurrences d'ordre 1 ($d = 1$) donne lieu aux suites *hypergéométriques*, qui jouent un rôle important dans les algorithmes de sommation symbolique, abordés au Chapitre 27.

1.2. Équivalence. Le résultat qu'il s'agit de considérer d'un point de vue algorithmique est le suivant. Il s'agit d'un analogue du Lemme 2 au Chapitre 4.

THÉORÈME 1. *Une série formelle est D-finie si et seulement si la suite de ses coefficients est P-récurrente.*

DÉMONSTRATION. Soit $A(X) = a_0 + a_1X + \cdots$ une série D-finie et

$$q_0(X)A^{(m)}(X) + \cdots + q_m(X)A(X) = 0 \tag{2}$$

une équation différentielle qui l'annule. En notant $[X^n]f(X)$ le coefficient de X^n dans la série $f(X)$ avec la convention que ce coefficient est nul pour $n < 0$, on a pour $n \geq 0$

$$[X^n]f'(X) = (n+1)[X^{n+1}]f(X), \qquad [X^n]X^k f(X) = [X^{n-k}]f(X). \tag{3}$$

Par conséquent, l'extraction du coefficient de X^n de (2) fournit une récurrence linéaire sur les a_n valide pour tout $n \geq 0$ avec la convention $a_k = 0$ pour $k < 0$. Pour obtenir une récurrence de la forme (1) il faut décaler les indices de $n_0 := \max_{0\leq i\leq m}(\deg q_i + i - m)$ s'il est strictement positif. Les équations obtenues alors pour les indices moindres fournissent des contraintes linéaires sur les premiers coefficients a_n pour qu'ils correspondent aux coefficients d'une série solution de (2).

À l'inverse, soit (a_n) une suite vérifiant la récurrence (1). Les identités analogues à (3) sont maintenant

$$\sum_{n\geq 0} n^k a_n X^n = \left(X\frac{d}{dX}\right)^k A(X), \quad \sum_{n\geq 0} a_{n+k}X^n = (A(X) - a_0 - \cdots - a_{k-1}X^{k-1})/X^k,$$

où A est la série génératrice des coefficients a_n et la notation $(X\frac{d}{dX})^k$ signifie que l'opérateur $X\frac{d}{dX}$ est appliqué k fois. En multipliant (1) par X^n et en sommant pour n allant de 0 à ∞, puis en multipliant par une puissance de X on obtient donc une équation différentielle linéaire de la forme

$$q_0(X)A^{(d)}(X) + \cdots + q_d(X)A(X) = p(X),$$

où le membre droit provient des conditions initiales. Il est alors possible, quitte à augmenter l'ordre de l'équation de 1, de faire disparaître ce membre droit, par une dérivation et une combinaison linéaire. □

EXEMPLE 4. L'équation $A' - X^k A = 0$ ($k \in \mathbb{N}$) donne la récurrence linéaire $(n+1)a_{n+1} - a_{n-k} = 0$ valide pour tout $n \geq 0$, avec la convention que les a_n d'indice négatif sont nuls. On en déduit que a_0 est libre, puis les contraintes $a_1 = \cdots = a_k = 0$, et les coefficients suivants sont fournis par la récurrence décalée $(n+k+1)a_{n+k+1} - a_n = 0$, valide pour $n \geq 0$. On reconnaît ainsi les coefficients de $a_0 \exp(X^{k+1}/(k+1))$.

EXERCICE 1. Retrouver la récurrence satisfaite par la suite $1/n!$ à partir de l'équation $A' = A$, et réciproquement.

EXERCICE 2. Calculer une équation différentielle linéaire satisfaite par la série génératrice des nombres de Fibonacci, définis par $F_{n+2} = F_{n+1} + F_n$, $F_0 = 0$, $F_1 = 1$.

EXERCICE 3. Estimer la complexité d'un algorithme direct utilisant cette idée en nombre d'opérations dans $\mathbb{K}$.

Un algorithme plus efficace pour passer d'une équation différentielle d'ordre élevé à la récurrence linéaire satisfaite par les coefficients des solutions est donné en §1.5.

1.3. Exemples d'applications.

EXEMPLE 5. Pour calculer le coefficient de X^{1000} dans

$$(1+X)^{1000}(1+X+X^2)^{500},$$

une méthode efficace part de l'observation que ce polynôme vérifie l'équation différentielle linéaire du premier ordre

$$\frac{y'}{y} = 1000\frac{1}{1+X} + 500\frac{2X+1}{1+X+X^2}.$$

Il s'ensuit que les coefficients de ce polynôme vérifient une récurrence d'ordre 3 à coefficients de degré 1. Cette récurrence permet d'obtenir le N^{e} coefficient en $O(N)$ opérations arithmétiques, voire $\tilde{O}(\sqrt{N})$ en utilisant la technique de « pas de bébés, pas de géants » présentée au Chapitre 15. Pour la complexité binaire, ce même chapitre présente la méthode de scindage binaire qui est quasi-optimale.

EXEMPLE 6. Pour calculer une racine du polynôme $P_N(X)$ défini par la série génératrice

$$\sum_{n\geq 0} P_n(X)\frac{Z^n}{n!} = \left(\frac{1+Z}{1+Z^2}\right)^X,$$

lorsque N est grand, il n'est pas nécessaire de calculer ce polynôme. Il suffit d'observer que cette série vérifie une équation différentielle linéaire d'ordre 1 (avec X en paramètre), ainsi que la série génératrice des dérivées des P_n, et d'utiliser les récurrences que l'on en déduit sur ces polynômes pour en calculer des valeurs. Ces valeurs permettent alors d'appliquer la méthode de Newton, par exemple pour résoudre le polynôme. Cette idée peut aussi être combinée avec la méthode de scindage binaire du Chapitre 15.

1.4. Test d'égalité.

LEMME 1. *Si (u_n) et (v_n) sont deux suites solutions de (1), et s'il existe $k \in \mathbb{N}$ tel que $u_0 = v_0, \ldots, u_{k-1} = v_{k-1}$ et $0 \notin p_d(k-d+1+\mathbb{N})$, alors ces suites sont égales.*

DÉMONSTRATION. Par récurrence, puisque $0 \notin p_d(k-d+1\mathbb{N})$ permet d'inverser le coefficient de tête de la récurrence et donc d'exprimer chaque terme à partir de l'indice d en fonction des précédents. □

EXEMPLE 7. Pour une récurrence à coefficients constants, il suffit de connaître une borne sur l'ordre de la récurrence pour tester l'égalité de deux solutions. Cette observation est utile en liaison avec les opérations de clôture de la section suivante.

COROLLAIRE 1. *Si $f(X)$ et $g(X)$ sont deux séries formelles solutions de (2), $f(0) = g(0), \ldots, f^{(m-1)}(0) = g^{(m-1)}(0)$ et $q_0(0) \neq 0$, alors ces séries sont égales.*

DÉMONSTRATION. D'après (3), un terme $cX^i A^{(j)}(X)$ intervient dans la récurrence sur les coefficients sous la forme $c(n-i+1)\cdots(n-i+j)a_{n-i+j}$. L'indice maximal est donc atteint pour $j-i$ maximal et donc pour $j=m$ si $i=0$. La récurrence obtenue, valide pour $n \geq 0$ est donc de la forme $q_0(0)(n+1)\cdots(n+m)a_{n+m} + \cdots = 0$. Les contraintes linéaires sur les conditions initiales jusqu'à l'indice n_0 introduit dans la preuve du Théorème 1 définissent les coefficients d'indice m à $m+n_0$ à partir des précédents et le lemme précédent s'applique pour les valeurs suivantes. □

1.5. Algorithme rapide pour la conversion. Vue l'efficacité obtenue grâce au passage de l'équation différentielle à la récurrence, il est souhaitable de maîtriser le coût de cette conversion. Il est possible d'obtenir la récurrence plus efficacement que par la méthode naïve, en mettant en œuvre des techniques qui exploitent la multiplication rapide.

Cas d'un opérateur donné en $\theta = X\frac{d}{dX}$. Si l'équation différentielle linéaire de départ est donnée non pas comme un polynôme en X et d/dX, mais comme un polynôme en X et $\theta = X\frac{d}{dX}$ (θ est parfois appelé *l'opérateur d'Euler*), alors la conversion en récurrence est assez facile : l'application de l'opérateur différentiel

$$\sum_{\substack{0\leq j\leq m\\ 0\leq i\leq d}} p_{i,j} X^j \theta^i,$$

à une série $A(X)$ suivie de l'extraction du coefficient de X^n donne au vu des relations (3) la récurrence

$$\sum p_{i,j}(n-j)^i a_{n-j} = 0.$$

De cette conversion découle le résultat de complexité suivant.

PROPOSITION 1. *La récurrence satisfaite par les coefficients des séries solutions d'une équation différentielle linéaire de degré d en $\theta = X\frac{d}{dX}$ et m en X se calcule en $O(m\,\mathsf{M}(d))$ opérations sur les coefficients.*

Par rapport au nombre dm de coefficients du résultat, cette complexité est quasi-optimale.

La preuve utilise la formule ci-dessus et l'observation que calculer les coefficients du polynôme $P(X-j)$ connaissant ceux du polynôme $P(X)$ de degré d ne requiert que $O(\mathsf{M}(d))$ opérations, par exemple en utilisant la somme composée (Chapitre 3).

Cas général. Quitte à le multiplier au préalable par une puissance de X égale au plus à son degré en $\partial = d/dX$, il est toujours possible de récrire un polynôme en X et ∂ en un polynôme en X et θ. Cette réécriture peut elle-même être effectuée assez rapidement.

Une première observation est que de la *commutation*

$$(\theta - i)X^i = X^i\theta$$

se déduit par multiplication à droite par ∂^i la relation

$$X^{i+1}\partial^{i+1} = (\theta - i)X^i\partial^i = (\theta - i)(\theta - i + 1)\cdots\theta.$$

Étant donnés des polynômes $p_i(X)$ de degré au plus $m+d$, il s'agit donc maintenant de calculer des polynômes $q_i(X)$ tels que

$$\sum_{i=0}^{d} p_i(X)X^i\partial^i = \sum_{i=0}^{d} p_i(X)(\theta - i + 1)\cdots\theta = \sum_{i=0}^{d} q_i(X)\theta^i.$$

Récrire le polynôme sous la forme

$$\sum_{j=0}^{m+d} X^j \sum_{i=0}^{d} p_{i,j}X^i\partial^i$$

s'effectue en nombre linéaire d'opérations et montre qu'il suffit de savoir traiter efficacement le cas où les p_i (et donc aussi les q_i) sont constants. La transition des uns vers les autres se calcule alors par évaluation-interpolation sur $\theta = 0, 1, 2, \ldots$. Soit P le polynôme à calculer. Les premières identités obtenues par évaluation sont

$$p_0 = q_0, \quad p_0 + p_1 = \sum q_i, \quad p_0 + 2p_1 + 2p_2 = \sum 2^i q_i,$$

et plus généralement

$$e^X \sum p_i X^i = \sum \frac{P(i)}{i!}X^i,$$

ce qui montre que les valeurs de P en $0, \ldots, d$ peuvent être obtenues en $O(\mathsf{M}(d))$ opérations à partir des coefficients p_i, et donc les coefficients de P en $O(\mathsf{M}(d)\log d)$ opérations, par interpolation rapide, en utilisant les techniques du Chapitre 5.

Au final, nous avons obtenu le résultat suivant.

THÉORÈME 2. *Le calcul de la récurrence linéaire satisfaite par les coefficients des séries solution d'une équation différentielle linéaire d'ordre d à coefficients des polynômes de degré au plus m requiert un nombre d'opérations arithmétiques borné par*

$$O((m + d)\mathsf{M}(d)\log d).$$

2. Propriétés de clôture

Les séries différentiellement finies jouissent de propriétés de clôture. En ce sens, elles sont un analogue différentiel des nombres algébriques. Ces propriétés de clôture généralisent celles sur les suites récurrentes linéaires à coefficients *constants* (srlcc) évoquées en Section 5.2 du Chapitre 4.

2.1. Somme et produit.

THÉORÈME 3. *L'ensemble des séries différentiellement finies à coefficients dans un corps $\mathbb{K}$ est une algèbre sur $\mathbb{K}$. L'ensemble des suites polynomialement récurrentes d'éléments de $\mathbb{K}$ est aussi une algèbre sur $\mathbb{K}$.*

Une conséquence algorithmique fondamentale de ce théorème est le fait que les séries différentiellement finies peuvent être multipliées plus rapidement que les séries arbitraires, en complexité *linéaire*.

COROLLAIRE 2. *Soient $A, B \in \mathbb{K}[[X]]$ deux séries différentiellement finies, solutions d'équations différentielles d'ordre $O(1)$ à coefficients polynomiaux de degré $O(1)$. On peut calculer le produit $AB \bmod X^N$ en $O(N)$ opérations dans $\mathbb{K}$.*

DÉMONSTRATION DU THÉORÈME. Les preuves pour les suites et les séries sont similaires. Les preuves pour les sommes sont plus faciles que pour les produits, mais dans le même esprit. Nous ne donnons donc que la preuve pour le produit $C = AB$ de deux séries différentiellement finies A et B. Par la formule de Leibniz, toutes les dérivées de C s'écrivent comme combinaisons linéaires de produits entre une dérivée $A^{(i)}$ de A et une dérivée $B^{(j)}$ de B. Les dérivées de A et de B étant engendrées par un nombre fini d'entre elles, il en va de même pour les produits $A^{(i)}B^{(j)}$, ce qui prouve la D-finitude de C. □

EXERCICE 4. Faire la preuve pour le cas du produit de suites polynomialement récurrentes.

Cette preuve donne également un algorithme pour trouver l'équation différentielle (resp. la récurrence) cherchée : il suffit de calculer les dérivées (resp. les décalées) successives en les récrivant sur un ensemble fini de générateurs. Une fois leur nombre suffisant (c'est-à-dire au pire égal à la dimension plus 1), il existe une relation linéaire entre elles. À partir de la matrice dont les lignes contiennent les coordonnées des dérivées successives (resp. des décalés successifs) sur cet ensemble fini de générateurs, la détermination de cette relation se réduit alors à celle du noyau de la transposée. Le calcul se ramène donc ultimement à la résolution d'un système linéaire à coefficients dans $\mathbb{K}[X]$, ce qui peut se faire efficacement grâce à l'algorithme de Storjohann (Chapitre 11).

EXEMPLE 8. Voici comment prouver (et même découvrir) l'identité

$$\arcsin(X)^2 = \sum_{k\geq 0} \frac{k!}{\left(\frac{1}{2}\right)\cdots\left(k+\frac{1}{2}\right)} \frac{X^{2k+2}}{2k+2}.$$

Le calcul consiste à partir d'une équation satisfaite par $\arcsin(X)$, en déduire une équation satisfaite par son carré, traduire cette équation en récurrence sur les coefficients, et conclure en constatant que cette récurrence est satisfaite par les coefficients de la série.

Le point de départ est la propriété $(\arcsin(X))' = 1/\sqrt{1-X^2}$, qui permet de représenter arcsin par l'équation différentielle $(1-X^2)y'' - Xy' = 0$ avec les conditions initiales $y(0) = 0$, $y'(0) = 1$.

Ensuite, en posant $h = y^2$, les dérivations et réductions successives donnent

$$\begin{aligned}
h' &= 2yy', \\
h'' &= 2y'^2 + 2yy'' = 2y'^2 + \frac{2X}{1-X^2}yy', \\
h''' &= 4y'y'' + \frac{2X}{1-X^2}(y'^2 + yy'') + \left(\frac{2}{1-X^2} + \frac{4X^2}{(1-X^2)^2}\right)yy', \\
&= \left(\frac{4X+2}{1-X^2} + \frac{6X^2}{(1-X^2)^2}\right)yy' + \frac{2X}{1-X^2}y'^2.
\end{aligned}$$

Les quatre vecteurs h, h', h'', h''' sont combinaisons linéaires des trois vecteurs y^2, yy', y'^2. Ils sont donc liés et une relation de liaison s'obtient en calculant le noyau de la matrice 3×4 qui découle de ce système. Le résultat est

$$(1-X^2)h''' - 3Xh'' - h' = 0.$$

La récurrence qui s'en déduit est

$$(n+1)(n+2)(n+3)a_{n+3} - (n+1)^3 a_{n+1} = 0.$$

Comme le facteur $(n+1)$ ne s'annule pas sur $\mathbb{N}$, il est possible de simplifier pour obtenir la récurrence équivalente

$$(n+2)(n+3)a_{n+3} - (n+1)^2 a_{n+1} = 0.$$

Si le membre droit de l'identité n'était pas connu, il serait possible de le déterminer grâce à cette récurrence à deux termes et aux conditions initiales $h(0) = 0$, $h'(0) = 0$, $h''(0) = 2$. Lorsque le membre droit est donné, la vérification que les coefficients de la série ci-dessus vérifient cette identité est facile.

Bornes. En outre, la preuve du théorème 2 permet de borner l'ordre des équations : l'ordre de l'équation satisfaite par une somme est borné par la somme des ordres des équations satisfaites par les sommants, et l'ordre de l'équation satisfaite par un produit est borné par le produit des ordres. On en déduit alors un algorithme simple de preuve d'identités.

Exemple 9. L'identité de Cassini sur les nombres de Fibonacci s'écrit

$$F_{n+2}F_n - F_{n+1}^2 = (-1)^n.$$

Pour prouver cette égalité, le point de départ est simplement la récurrence définissant les nombres de Fibonacci :

$$F_{n+2} = F_{n+1} + F_n,$$

qui exprime que tous les décalés de F_n sont des combinaisons linéaires de F_n et F_{n+1}. Les produits qui interviennent dans l'identité de Cassini s'expriment donc *a priori* comme combinaison linéaire de F_n^2, F_nF_{n+1} et F_{n+1}^2 et donc le membre de gauche vérifie une récurrence d'ordre borné par 4. Le membre droit vérifiant une récurrence d'ordre 1, leur différence vérifie une récurrence d'ordre au plus 5. *Il n'est pas nécessaire de calculer cette récurrence.* Il suffit de vérifier que ses 5 conditions initiales sont nulles. Autrement dit, vérifier l'identité pour $n = 0, \dots, 4$ la prouve !

Si le membre droit n'est pas donné, on calcule la récurrence satisfaite par le membre gauche en récrivant ses décalés sur les générateurs $F_{n+1}F_n, F_n^2, F_{n+1}^2$:

$$\begin{aligned} u_n &= F_{n+2}F_n - F_{n+1}^2 = F_{n+1}F_n + F_n^2 - F_{n+1}^2, \\ u_{n+1} &= F_{n+2}F_{n+1} + F_{n+1}^2 - F_{n+2}^2 = F_{n+1}^2 - F_n^2 - F_nF_{n+1} \\ &= -u_n. \end{aligned}$$

et on conclut en observant que $u_0 = 1$.

Exercice 5. De la même manière, montrer que $\sin^2 X + \cos^2 X = 1$, avec et sans calcul.

2.2. Produit d'Hadamard.

Corollaire 3. *Si $A = \sum_{n\geq 0} a_n X^n$ et $B = \sum_{n\geq 0} b_n X^n$ sont deux séries différentiellement finies, alors leur produit d'Hadamard*

$$A \odot B = \sum_{n\geq 0} a_n b_n X^n$$

l'est aussi.

La preuve est également un algorithme : des deux équations différentielles se déduisent deux récurrences satisfaites par les suites (a_n) et (b_n) ; d'après la section précédente, le produit (a_nb_n) vérifie alors une récurrence linéaire, dont se déduit enfin l'équation différentielle satisfaite par sa série génératrice.

EXEMPLE 10. Les polynômes orthogonaux de Hermite ont pour série génératrice

$$\sum_{n\geq 0} H_n(X)\frac{Z^n}{n!} = \exp(Z(2X-Z)).$$

À partir de là, la détermination du membre droit de l'identité suivante due à Mehler est entièrement algorithmique :

$$\sum_{n\geq 0} H_n(X)H_n(Y)\frac{Z^n}{n!} = \frac{\exp\left(\frac{4Z(XY-Z(X^2+Y^2))}{1-4Z^2}\right)}{\sqrt{1-4Z^2}}.$$

EXEMPLE 11. Si l'intégrale

$$I(t) = \int_{-\infty}^{\infty} e^{-xt^2} f(x)\,dx,$$

peut être calculée à partir du développement de Taylor de f en intervertissant sommation et intégration, alors $I(t)$ est D-finie si f l'est. Il en va de même pour la transformée de Laplace ou la transformée de Fourier.

EXERCICE 6. Montrer que dans les mêmes conditions, f est D-finie si I l'est, et donner un algorithme pour calculer une équation différentielle linéaire satisfaite par f à partir d'une équation satisfaite par I.

3. Séries algébriques

DEFINITION 3. Une série formelle $F(X) \in \mathbb{K}[[X]]$ est une *série algébrique* lorsqu'il existe un polynôme non nul $P(X,Y) \in \mathbb{K}[X,Y]$ tel que $P(X,F(X)) = 0$.

THÉORÈME 4. *Si la série $F(X)$ annule un polynôme $P(X,Y) \in \mathbb{K}[X,Y]$ de degré d en Y, où $\mathbb{K}$ est un corps de caractéristique 0, alors elle est solution d'une équation différentielle linéaire d'ordre au plus d.*

DÉMONSTRATION. La preuve est algorithmique. Quitte à diviser d'abord P par son pgcd avec sa dérivée P_Y par rapport à Y, il est possible de le supposer premier avec P_Y (car la caractéristique est nulle !). En dérivant $P(X,Y) = 0$ et en isolant Y', il vient

$$Y' = -\frac{P_X}{P_Y}.$$

Par *inversion modulaire* de P_Y (voir le Chapitre 6), cette identité se récrit via un calcul de pgcd étendu en

$$Y' = R_1(Y) \bmod P,$$

où R_1 est un polynôme en Y de degré au plus d et à coefficients dans $\mathbb{K}(X)$. Ceci signifie que Y' s'écrit comme combinaison linéaire de $1, Y, Y^2, \ldots, Y^{d-1}$ à coefficients dans $\mathbb{K}(X)$. Dériver à nouveau cette équation, puis récrire Y' et prendre le reste de la division par P mène à nouveau à une telle combinaison linéaire pour Y'' et plus généralement pour les dérivées successives de Y. Les $d+1$ vecteurs $Y, Y', \ldots, Y^{(d)}$ sont donc linéairement dépendants et la relation de liaison est l'équation cherchée. □

EXEMPLE 12. Les dénombrements d'arbres mènent naturellement à des équations algébriques sur les séries génératrices. Ainsi, la série génératrice $C(X)$ des nombres de Catalan (nombre d'arbres binaires à n sommets internes) vérifie

$$C = 1 + XC^2;$$

la série génératrice $M(X)$ des nombres de Motzkin (nombre d'arbres unaires-binaires à n sommets internes) vérifie

$$M = 1 + XM + XM^2.$$

Dans les deux cas, il est aisé d'obtenir d'abord une équation différentielle puis une récurrence qui permet de calculer efficacement ces nombres par scindage binaire (Chapitre 15). Dans le cas des nombres de Catalan, la récurrence est d'ordre 1, la suite est donc hypergéométrique et s'exprime aisément.

EXERCICE 7. Trouver une formule explicite des nombres de Catalan. Récrire les fonctions Γ qui pourraient apparaître dans le résultat en termes de factorielles, puis de coefficient binomial.

EXERCICE 8. À l'aide d'un système de calcul formel, calculer une récurrence linéaire satisfaite par les coefficients de la série $Y(X)$ solution de

$$Y = 1 + XY + XY^7.$$

Les mêmes arguments que ci-dessus mènent à une autre propriété de clôture des séries différentiellement finies.

COROLLAIRE 4. *Si F est une série D-finie et A une série algébrique sans terme constant, alors $F \circ A$ est D-finie.*

La preuve consiste à observer que les dérivées successives de $F \circ A$ s'expriment comme combinaisons linéaires des $F^{(i)}(A)A^j$ pour un nombre fini de dérivées de F (par D-finitude) et de puissances de A (par la même preuve que pour le théorème 4). Cette preuve fournit encore un algorithme.

EXEMPLE 13. À l'aide d'un système de calcul formel, calculer une récurrence linéaire satisfaite par les coefficients du développement en série de Taylor de

$$\exp\left(\frac{1-\sqrt{1-4X}}{2}\right).$$

COROLLAIRE 5. *Si A est une série algébrique, alors* $\exp\int A$ *est D-finie.*

EXERCICE 9. Faire la preuve.

EXEMPLE 14 (La moyenne arithmético-géométrique). Si a et b sont deux réels tels que $0 \le b \le a$, les deux suites définies par

$$a_{n+1} = \frac{a_n + b_n}{2}, \quad b_{n+1} = \sqrt{a_n b_n}, \quad a_0 = a, \quad b_0 = b$$

ont une limite commune dont l'existence se déduit de $b_n \le b_{n+1} \le a_{n+1} \le a_n$ et

$$a_{n+1}^2 - b_{n+1}^2 = \left(\frac{a_n - b_n}{2}\right)^2.$$

Cette limite est notée $M(a,b)$. L'identité suivante

$$M(a,b) = \frac{a}{F(\frac{1}{2},\frac{1}{2};1;1-\frac{b^2}{a^2})}.$$

due à Gauss, où la série hypergéométrique est définie comme

$$F(a,b;c;z) := \sum_{n\ge 0} \frac{(a)_n (b)_n}{(c)_n} \frac{z^n}{n!}, \quad \text{où} \quad (x)_n = x(x+1)\cdots(x+n-1),$$

peut se prouver par les algorithmes de ce chapitre.

Tout d'abord, la fonction M est clairement homogène : pour $\lambda > 0$, $M(\lambda a, \lambda b) = \lambda M(a,b)$ ce qui permet de concentrer l'étude sur la fonction de une variable $M(1,x)$, dont on peut prouver qu'elle est analytique au voisinage de $x = 1$. Ensuite, en utilisant l'homogénéité et le fait que $M(a_1,b_1) = M(a_0,b_0)$ avec $a_0 = 1 + x$ et $b_0 = 1 - x$, on obtient l'identité

$$M(1,\sqrt{1-x^2}) = (1+x)M\left(1,\frac{1-x}{1+x}\right).$$

En posant $M(1,x) = 1/A(1-x^2)$, cette équation entraine l'équation fonctionnelle

$$(1+x)A(x^2) - A\left(1 - \frac{(1-x)^2}{(1+x)^2}\right) = 0, \tag{4}$$

qui admet une seule solution série avec $A(0) = 1$. Il suffit donc de montrer que cette solution est $f(x) = F(\frac{1}{2}, \frac{1}{2}; 1; x)$.

Pour cela, on va construire une équation différentielle satisfaite par le membre gauche de l'équation ci-dessus évalué en $A(x) = f(x)$, puis observer que quelques conditions initiales suffisent à prouver la nullité.

La définition de la série hypergéométrique entraîne clairement une récurrence d'ordre 1 sur les coefficients

$$\frac{u_{n+1}}{u_n} = \frac{(a+n)(b+n)}{(c+n)(1+n)}.$$

En convertissant cette récurrence en équation différentielle, on obtient que $f(x)$ satisfait

$$4x(x-1)f''(x) + 4(x-2)f'(x) + f(x) = 0.$$

À partir de cette équation, on construit une équation pour $g(x) = f(x^2)$:

$$x(x^2-1)g''(x) + (3x^2-1)g'(x) + xg(x) = 0$$

et une autre pour $h(x) = g(1-(1-x)^2/(1+x)^2)$:

$$x(x-1)(x+1)^2h''(x) + (x+1)(x^2+2x-1)h'(x) - (x-1)h(x) = 0 \tag{5}$$

(il s'agit de cas simples du corollaire ci-dessus). En utilisant la clôture par somme, on peut alors construire une équation pour $(1+x)g(x) + h(x)$, et on regrouve exactement la même que pour h (ceci peut se voir directement : en remplaçant $g(x)$ dans l'équation qui le définit par $u(x)/(1+x)$, on obtient l'équation satisfaite par h).

À ce stade, l'équation (5) est donc satisfaite par le membre gauche de l'équation (4) lorsque A vaut f et il s'agit de prouver que cette solution de (5) est nulle. Le corollaire 1 ne s'applique pas ici puisque le coefficient de tête s'annule en 0. En revanche, on peut passer à la récurrence satisfaite par les coefficients des solutions séries de cette équation (5) :

$$(n+3)^2u_{n+3} + (n^2+2n-1)u_{n+2} - (n^2+4n+2)u_{n+1} - n^2u_n = 0,$$

ce qui montre qu'il suffit de vérifier les conditions initiales $h(0) = h'(0) = h''(0) = 0$. Pour cela, aucun calcul n'est nécessaire puisque la série A est définie comme solution de (4), ce qui conclut.

Dans cet exemple, nous sommes partis de l'équation satisfaite par la série hypergéométrique pour prouver qu'elle vérifiait une équation fonctionnelle. Il est également possible d'utiliser une méthode de coefficients indéterminés pour calculer suffisamment de coefficients de la solution $A(x)$ de (4), puis conjecturer cette équation différentielle par un calcul d'approximants de Padé-Hermite (Chapitre 7), et enfin de montrer comme nous l'avons fait que la série définie par cette équation conjecturée est bien solution de l'équation fonctionnelle.

4. Au-delà

En général, la composition de deux séries différentiellement finies n'est pas D-finie. Voici trois résultats plus forts, dont la preuve repose sur la théorie de Galois différentielle et dépasse le cadre de ce cours.

THÉORÈME 5. *Soient A et F deux séries de $\mathbb{K}[[X]]$.*

1. *Les séries F et $1/F$ sont simultanément différentiellement finies si et seulement si F'/F est algébrique.*

2. *Les séries F et $\exp(\int F)$ sont simultanément différentiellement finies si et seulement si F est algébrique.*
3. *Soit A algébrique de genre supérieur ou égal à 1, alors F et $A \circ F$ sont différentiellement finies si et seulement si F est algébrique.*

EXERCICE 10. Prouver le sens « si » de ces trois propriétés.

Exercices

EXERCICE 11. Parmi les suites données par les termes généraux suivants, indiquer celles qui sont polynomialement récurrentes et celles qui ne le sont pas :

$$\frac{1}{\binom{2n}{n}}, \qquad \cos n, \qquad 2^{2^n}, \qquad H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}, \qquad \lfloor \log n \rfloor.$$

Parmi les développements en série des fonctions suivantes, indiquer ceux qui sont D-finis et ceux qui ne le sont pas :

$$\exp\Big(\sqrt[3]{1-X}\Big), \qquad \prod_{i\geq 1} \frac{1}{1-X^i}, \qquad \sum_{i\geq 0} X^{2^i}, \qquad \sum_{n\geq 0} F_n^3 \frac{X^n}{n!}, \qquad \tan(X).$$

(F_n désigne le n^{e} nombre de Fibonacci, dont la suite vérifie $F_{n+2} = F_{n+1} + F_n$ et $F_1 = F_0 = 1$.)

EXERCICE 12.

1. Soit $(a_n)_{n\geq 0}$ une suite P-récurrente d'éléments d'un corps de caractéristique 0. Montrer que les suites extraites $(a_{2n})_{n\geq 0}$ et $(a_{2n+1})_{n\geq 0}$ sont également polynomialement récurrentes.
2. Soit $(g_n)_{n\geq 0}$ une suite dont la série génératrice exponentielle est
$$\sum_{n\geq 0} g_n \frac{X^n}{n!} = \frac{\exp(-X/2 - X^2/4)}{\sqrt{1-X}}.$$
Montrer que la suite $(g_n)_{n\geq 0}$ satisfait à une récurrence linéaire homogène, d'ordre au plus 3 et à coefficients polynomiaux de degrés au plus 2. La récurrence n'est pas demandée explicitement.
3. Si $N > 0$, quel est le coût binaire, en fonction de N, du calcul de g_N ? [Utiliser des résultats du Chapitre 15.]

EXERCICE 13 (Opérations de clôture pour les équations différentielles linéaires à coefficients constants). Soient $f(X)$ et $g(X)$ solutions des équations différentielles linéaires homogènes

$$a_m f^{(m)}(X) + \cdots + a_0 f(X) = 0, \qquad b_n g^{(n)}(X) + \cdots + b_0 g(X) = 0,$$

avec $a_0, \ldots, a_m, b_0, \ldots, b_n$ des coefficients rationnels.

1. Montrer que $f(X)g(X)$ et $f(X)+g(X)$ sont solutions d'équations différentielles du même type.

Si le polynôme $a_m X^m + \cdots + a_0$ se factorise sur $\mathbb{C}$ en

$$a_m (X-\alpha_1)^{d_1} \cdots (X-\alpha_k)^{d_k},$$

on rappelle qu'une base de l'espace des solutions de

$$a_m f^{(m)}(X) + \cdots + a_0 f(X) = 0$$

est donnée par $\big\{e^{\alpha_1 X}, Xe^{\alpha_1 X}, \ldots, X^{d_1-1}e^{\alpha_1 X}, \ldots,\ e^{\alpha_k X}, Xe^{\alpha_k X}, \ldots, X^{d_k-1}e^{\alpha_k X}\big\}$.

2. Montrer qu'une équation satisfaite par $f(X) + g(X)$ peut être calculée à l'aide de l'algorithme d'Euclide.

3. Montrer qu'une équation satisfaite par $f(X)g(X)$ peut être obtenue par un calcul de résultant.

EXERCICE 14 (Puissance symétrique d'équation différentielle). Soient m et d deux entiers naturels et soient $a(X), b(X)$ deux polynômes dans $\mathbb{Q}[X]$ de degrés au plus d.

1. Montrer qu'il existe une équation différentielle $\mathcal{E}_m$ linéaire homogène d'ordre $m+1$, à coefficients dans $\mathbb{Q}[X]$, qui admet $\phi(X)^m$ comme solution, quelle que soit la solution $\phi(X)$ de l'équation différentielle
$$(\mathcal{E}_1) \qquad y''(X) + a(X)y'(X) + b(X)y(X) = 0.$$
On admettra que pour toute base (ϕ_1, ϕ_2) de solutions de $\mathcal{E}_1$, les fonctions $\phi_1^i \phi_2^{m-i}$, $i = 0, \ldots, m$ sont linéairement indépendantes.
2. Montrer que si $a = 0$, alors $\mathcal{E}_2 : y'''(X) + 4b(X)y'(X) + 2b'(X)y(X) = 0$.
3. Expliciter un algorithme pour calculer $\mathcal{E}_m$, en ramenant le calcul final à un calcul sur des matrices de polynômes.
4. Estimer la complexité de cet algorithme en nombres d'opérations dans $\mathbb{Q}$ en fonction de m et d.
5. Pour m fixé, on associe à une fonction y de X des fonctions L_k par les valeurs initiales
$$L_0(X) = y(X), \quad L_1(X) = y'(X)$$
et la relation de récurrence
$$L_{k+1}(X) = L_k'(X) + ka(X)L_k(X) + k(m-k+1)b(X)L_{k-1}(X).$$
(a) Montrer que lorsque $y = \phi^m$, pour $0 \le k \le m$,
$$L_k(X) = m(m-1)\ldots(m-k+1)\phi^{m-k}(X)\phi'(X)^k,$$
et $L_{m+1}(X) = 0$.
(b) Montrer que $L_{m+1}(X) = 0$ n'est autre que l'équation $\mathcal{E}_m$.
(c) En déduire des bornes sur les degrés des coefficients de $\mathcal{E}_m$ et un algorithme de complexité $O(m^3\mathsf{M}(d))$ pour le calcul de $\mathcal{E}_m$.
6. * Montrer que si $a, b \in \mathbb{Q}$, alors $\mathcal{E}_m$ est à coefficients constants et qu'on peut calculer $\mathcal{E}_m$ en $O(\mathsf{M}(m)\log m)$ opérations dans $\mathbb{Q}$.

Notes

Les propriétés de clôture des séries différentiellement finies ont été décrites avec leurs applications par Stanley [12, 13] et Lipshitz [8].

L'utilisation des séries différentiellement finies pour les séries algébriques en combinatoire est exploitée à de nombreuses reprises par Comtet dans son livre [5], où il utilise l'algorithme décrit dans la preuve du Théorème 4. L'histoire de cet algorithme est compliquée. Il était connu d'Abel qui l'avait rédigé dans un manuscrit de 1827 qui n'a pas été publié. Ce manuscrit est décrit (p. 287) dans les œuvres complètes d'Abel [1]. Ensuite, ce résultat a été retrouvé par Sir James Cockle en 1860 et popularisé par le révérend Harley en 1862 [6]. Quelques années plus tard, il est encore retrouvé par Tannery dans sa thèse, dont le manuscrit est remarquablement clair et disponible sur le web [14]. Pour le calcul du développement de séries algébriques, il faut tenir compte de l'ordre assez élevé des équations obtenues [4], ce qui rend cet algorithme utile pour des très grandes précisions, l'itération de Newton du Chapitre 3 étant préférable pour des précisions modérées.

Les limitations de la dernière section ne sont pas très connues. Elles sont dues à Harris et Sibuya pour la première [7] et à Singer pour les deux autres [10].

En ce qui concerne les algorithmes et les implantations, la plupart des algorithmes présentés dans ce chapitre sont implantés dans le package `gfun` de Maple [9]. L'algorithme rapide de la Section 1.5 provient essentiellement de [3], qui donne une variante légèrement plus efficace (d'un facteur constant).

Bibliographie

[1] Abel, Niels Henrik (1992). *Œuvres complètes. Tome II*. Réimpression de la seconde édition (1881). Éditions Jacques Gabay, vi+716 pages. URL : http://www.gallica.fr.

[2] Abramowitz, Milton et Irene A. Stegun, éds. (1992). *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover Publications Inc., xiv+1046 pages.

[3] Bostan, Alin (2003). « Algorithmique efficace pour des opérations de base en Calcul formel ». Thèse de Doctorat. École polytechnique.

[4] Bostan, Alin, Frédéric Chyzak, Grégoire Lecerf, Bruno Salvy et Éric Schost (2007). « Differential equations for algebraic functions ». In : *ISSAC'07 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Dongming Wang. ACM Press, p. 25–32. DOI : 10.1145/1277548.1277553.

[5] Comtet, L. (1970). *Analyse Combinatoire*. 2 volumes. PUF.

[6] Harley, Rev. Robert (1862). « On the theory of the transcendental solution of algebraic equations ». In : *Quarterly Journal of Pure and Applied Mathematics*, vol. 5, p. 337–360.

[7] Harris, William A. et Yasutaka Sibuya (1985). « The reciprocals of solutions of linear ordinary differential equations ». In : *Advances in Mathematics*, vol. 58, n°2, p. 119–132.

[8] Lipshitz, L. (1989). « D-Finite Power Series ». In : *Journal of Algebra*, vol. 122, n°2, p. 353–373.

[9] Salvy, Bruno et Paul Zimmermann (1994). « Gfun : a Maple package for the manipulation of generating and holonomic functions in one variable ». In : *ACM Transactions on Mathematical Software*, vol. 20, n°2, p. 163–177. DOI : 10.1145/178365.178368.

[10] Singer, Michael F. (1986). « Algebraic relations among solutions of linear differential equations ». In : *Transactions of the American Mathematical Society*, vol. 295, n°2, p. 753–763.

[11] Sloane, N. J. A. (2006). *The On-Line Encyclopedia of Integer Sequences*. URL : http://www.research.att.com/~njas/sequences/.

[12] Stanley, R. P. (1980). « Differentiably Finite Power Series ». In : *European Journal of Combinatorics*, vol. 1, n°2, p. 175–188.

[13] Stanley, Richard P. (1999). *Enumerative combinatorics*. Vol. 2. Cambridge University Press, xii+581 pages.

[14] Tannery, Jules (1874). « Propriétés des intégrales des équations différentielles linéaires à coefficients variables ». Thèse de doctorat ès sciences mathématiques. Faculté des Sciences de Paris. URL : http://gallica.bnf.fr.

CHAPITRE 15

Récurrences linéaires à coefficients polynomiaux : N^{e} terme, N premiers termes

Résumé

Pour calculer tous les N premiers termes d'une suite polynomialement récurrente, l'algorithme direct par déroulement de la récurrence est quasi-optimal vis-à-vis de N : sa complexité arithmétique (resp. binaire) est linéaire en N (resp. quasi-quadratique en N). Le N-ième terme d'une telle suite peut être calculé plus rapidement que l'ensemble des N premiers termes, en essentiellement $\sqrt{N}$ opérations arithmétiques et N opérations binaires.

Le Chapitre 4 a montré comment calculer le N^{e} terme d'une suite donnée par une récurrence à coefficients constants en complexité arithmétique $O(\log N)$. Dans ce chapitre, nous nous attaquons au cadre plus général des récurrences à coefficients polynomiaux.

Le calcul du N^{e} terme ou des N premiers termes[1] d'une suite P-récurrente, donnée par la récurrence à coefficients polynomiaux

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \qquad (n \in \mathbb{N}), \tag{1}$$

intervient fréquemment en combinatoire et pour calculer des troncatures de séries, ainsi que comme étape clé dans des algorithmes de recherche de solutions polynomiales d'équations fonctionnelles linéaires, dans un algorithme de factorisation déterministe d'entiers, ou encore en cryptologie, dans des questions de comptage de points sur des courbes. Ce calcul intervient aussi dans l'évaluation numérique efficace des séries D-finies.

Si les coefficients $p_i(n)$ de la récurrence (1) ont degré au plus d en n, l'algorithme naïf par déroulement de la récurrence a une complexité arithmétique en $O(rdN)$ et une complexité binaire en $O\big(rN^2\mathsf{M}_{\mathbb{Z}}(d\log N)\big)$ (voir §1.3). Ces complexités sont quasi-optimales vis-à-vis de N pour le calcul de tous les N premiers termes. Pour l'application importante au calcul des N premiers termes d'une série solution d'une équation différentielle linéaire d'ordre d à coefficients de degré au plus m, il faut d'abord traduire l'équation différentielle en récurrence comme vu au Chapitre 14, et l'ensemble coûte donc

$$O\left((m+d)\,\mathsf{M}(d)\left(\log d + \frac{N}{d}\right)\right)$$

opérations arithmétiques.

Pour le calcul du N^{e} terme seul, les algorithmes présentés dans ce chapitre n'ont pas une aussi bonne complexité que pour des coefficients constants, mais cette fois encore, ils sont plus efficaces que le simple déroulement de la récurrence. Un nouveau phénomène distingue ces algorithmes de ceux qui ont été présentés jusqu'ici : les idées qui permettent le calcul du N^{e} terme en bonne complexité arithmétique ne sont pas les mêmes que pour abaisser la complexité binaire. Ainsi, l'algorithme par

1. Dans la suite on fait l'hypothèse que le coefficient de tête p_r de la récurrence (1) ne s'annule sur aucun des entiers $0, \ldots, N-r$, où N est l'indice du terme maximal que l'on souhaite calculer.

« pas de bébés, pas de géants » de la Section 2 donne un gain en racine de N sur la complexité arithmétique, par rapport à la méthode naïve, mais ne gagne rien sur la complexité binaire. Quant à l'algorithme par *scindage binaire* de la Section 3, il n'abaisse en rien la complexité arithmétique, et améliore pourtant la complexité binaire jusqu'à la rendre quasi-linéaire en la taille de sa sortie.

1. Calcul naïf de $N!$ et de suites P-récurrentes

1.1. Cas de la factorielle. La définition de la factorielle comme produit,

$$N! = 1 \times 2 \times \cdots \times N,$$

montre que $N!$ peut être calculé en $N-1$ opérations arithmétiques, ce qui est optimal pour le calcul des N premières factorielles.

L'estimation de la complexité binaire de cette méthode repose sur la *formule de Stirling* :

$$\log N! = N \log N - N \log e + \frac{1}{2} \log N + O(1), \qquad N \to \infty.$$

En particulier, nous retiendrons l'équivalence $\log N! \sim N \log N$ qui donne la taille de $N!$. La k^{e} étape du produit multiplie $k!$ par $k+1$, soit donc un entier de taille $\log k! \sim k \log k$ avec un entier de taille $\log(k+1) \sim \log k$. Ce produit déséquilibré coûte donc moins de $ck\mathsf{M}_{\mathbb{Z}}(\log k)$ opérations binaires pour une certaine constante c. Le calcul complet de $N!$ par la méthode naïve effectue donc moins de

$$\sum_{k=1}^{N} ck\mathsf{M}_{\mathbb{Z}}(\log k) = O\big(N^2 \mathsf{M}_{\mathbb{Z}}(\log N)\big)$$

opérations binaires. La formule de Stirling montre que cette complexité binaire est quasi-optimale pour le calcul conjoint des nombres $1!$, $2!$, …, $N!$.

1.2. Cas général – complexité arithmétique. Pour une suite P-récurrente donnée par une récurrence de la forme (1), le calcul par l'algorithme direct de u_{n+r} à partir des r valeurs précédentes demande $O(rd)$ opérations arithmétiques pour l'évaluation des polynômes (par exemple par le schéma de Horner) puis $O(r)$ opérations pour effectuer la combinaison linéaire des u_{n+i}. Le calcul naïf de $u_0, \ldots, u_N$ à partir des valeurs initiales u_0, …, u_{r-1} demande donc $O(rdN)$ opérations arithmétiques.

Une légère amélioration de cette borne vient de l'observation que les coefficients $p_i(n)$ peuvent être évalués sur les entiers $n = 0, 1, \ldots, N$ en utilisant des techniques d'évaluation multipoint rapide, avant de procéder au déroulement de (1). Cela permet d'abaisser la complexité arithmétique du calcul de $O(rdN)$ à $O(r\mathsf{M}(N+d)\log(N+d))$, voire à $O(r(N+d)M(d)/d)$ en utilisant l'observation du Corollaire 5 page 68, ce qui revient dans les deux cas à $\tilde{O}(r(N+d))$ si la multiplication polynomiale est à base de FFT.

1.3. Cas général – complexité binaire. Lorsque les coefficients des polynômes p_i sont entiers, la complexité binaire de l'algorithme naïf découle de nouveau essentiellement de la croissance de u_N. Pour estimer celle-ci, il est commode d'adopter une vision matricielle et de faire apparaître le rationnel u_N comme un quotient d'entiers. Introduisons la suite vectorielle des $U_n = {}^t(u_n, \ldots, u_{n+r-1})$, qui vérifie la

récurrence du premier ordre

$$U_{n+1} = \frac{1}{p_r(n)} A(n) U_n \quad \text{avec} \quad A(n) = \begin{pmatrix} 0 & p_r(n) & & & \\ & 0 & p_r(n) & & \\ & & \ddots & \ddots & \\ & & & 0 & p_r(n) \\ -p_0(n) & & \dots & & -p_{r-1}(n) \end{pmatrix}.$$

Considérons aussi la suite hypergéométrique (w_n) donnée par la condition initiale $w_0 = 1$ et la récurrence

$$w_{n+1} = p_r(n) w_n, \qquad (n \in \mathbb{N}),$$

qui permet de récrire U_n sous la forme

$$U_n = \frac{1}{w_n} \big(A(n-1) \cdots A(0)\big) U_0.$$

Le rationnel u_N s'écrit alors $u_N = v_N / w_N$, où v_N est le premier coefficient du vecteur $\big(A(N-1) \cdots A(0)\big) U_0$.

Pour estimer la croissance de v_N, introduisons pour un vecteur U de taille r et une matrice M de taille $r \times r$ la norme d'indice 1 et sa norme subordonnée

$$\|U\| = \sum_{j=1}^{r} |v_j| \qquad \text{et} \qquad \|M\| = \max_{1 \le j \le r} \sum_{i=1}^{r} |m_{i,j}|.$$

Il est alors facile de vérifier (et ce le sens de la norme subordonnée) que pour tout u, $\|Mu\| \le \|M\| \|u\|$. Soit ℓ un majorant commun pour la taille binaire des coefficients des p_i, $0 \le i \le r$, et des conditions initiales u_j, $0 \le j < r$. Il s'ensuit que la norme de $A(n)$ est bornée par

$$\|A(n)\| \le r\, 2^{\ell} (d+1)(n+1)^d,$$

et celle de U_0 par $r 2^{\ell}$. Les majorations

$$|v_N| \le \|A(N-1)\| \cdots \|A(0)\| \, \|U_0\| \le r^{N+1}\, 2^{\ell(N+1)} (d+1)^N (N!)^d$$

fournissent la borne $O\big(dN \log N + N\ell + N \log r\big)$ sur la taille $\log |v_N|$. Par le même raisonnement, la taille du dénominateur w_N est du même ordre (mêmes formules pour $r = 1$), si bien que par le même type d'argument que pour la factorielle, la complexité binaire du calcul naïf de u_N est $O\big(rN^2 \mathsf{M}_{\mathbb{Z}}(d \log N + \ell + \log r)\big)$.

2. Pas de bébés, pas de géants

On a vu au Chapitre 4 que le N^{e} terme d'une récurrence linéaire à coefficients constants peut être calculé en complexité arithmétique $O(\log N)$. Dans le cas des coefficients polynomiaux, les choses se compliquent : à ce jour, on ne connaît pas d'algorithme polynomial en $\log N$ (il est fort possible qu'un tel algorithme n'existe pas). L'exemple typique en est le calcul du N^{e} terme de la factorielle $u_N = N!$, qui vérifie la récurrence à coefficients polynomiaux $u_{n+1} = (n+1) u_n$ pour $n \ge 0$. Une solution efficace exploite le théorème « évaluation interpolation » (théorème 1 du Chapitre 5). Elle utilise la technique des « pas de bébés, pas de géants » et requiert un nombre d'opérations arithmétiques en $\sqrt{N}$, à des facteurs logarithmiques près. Pour simplifier la présentation, supposons que N est un carré parfait. L'idée de l'algorithme est de poser

$$P(X) = (X+1)(X+2) \cdots (X + N^{1/2}),$$

Pas de bébés, pas de géants

Entrée : Une récurrence de la forme (1) avec $\deg p_i \le d$, $i = 0, \dots, r$, des conditions initiales $u_0, \dots, u_{r-1}$, un entier N.

Sortie : Les termes $u_{N-r+1}, \dots, u_N$ de la solution.

1. Calculer $P(X) = A(X+m-1)\cdots A(X+1)A(X)$, pour $m = [(N/d)^{1/2}]$, où A est la matrice définie en page 215.
2. Évaluer les coefficients de $P(0), P(m), \dots, P(\lfloor \frac{N-r-m}{m} \rfloor m)$.
3. Évaluer ceux de $A(N-r), A(N-r-1), \dots, A(\lfloor \frac{N-r}{m} \rfloor m)$
4. Calculer le produit de matrices scalaires $M = A(N-r)\cdots A(\lfloor \frac{N-r}{m} \rfloor m)P(\lfloor \frac{N-r}{m} \rfloor m)\cdots P(0)$.
5. Si $p_r(n) \neq 1$, calculer $w = p_r(N-r)\cdots p_r(0)$ par ce même algorithme ; sinon $w = 1$.
6. Renvoyer $\frac{1}{w} M \cdot {}^t(u_0, \dots, u_r)$.

FIGURE 1. Pas de bébés, pas de géants pour les récurrences linéaires.

afin d'obtenir la valeur de u_N à l'aide de l'équation

$$u_N = \prod_{j=0}^{N^{1/2}-1} P\big(jN^{1/2}\big). \tag{2}$$

Cette égalité suggère la procédure suivante :

1. *Pas de bébés* : Calculer les coefficients de P. Ceci peut être fait en utilisant $O\big(\mathsf{M}(\sqrt{N})\log N\big)$ opérations arithmétiques (en construisant un arbre binaire de feuilles $X+i$ comme en Section 4.2 du Chapitre 5).
2. *Pas de géants* : Évaluer P sur les points 0, $\sqrt{N}$, $2\sqrt{N}$, …, $(\sqrt{N}-1)\sqrt{N}$ et retrouver la valeur de u_N à l'aide de l'équation (2). En utilisant les techniques d'évaluation multipoint rapide (Chapitre 5, Section 4.3), ceci peut se faire également en $O\big(\mathsf{M}(\sqrt{N})\log N\big)$ opérations arithmétiques.

Le coût total de cet algorithme est $O\big(\mathsf{M}(\sqrt{N})\log N\big)$ opérations arithmétiques. Si la FFT est utilisée pour la multiplication des polynômes, le gain par rapport à la méthode directe est de l'ordre de $\sqrt{N}$, à des facteurs logarithmiques près, typique pour la technique des « pas de bébés, pas de géants ». La technique gagne encore par rapport à l'algorithme naïf si l'on emploie une multiplication par Karatsuba, mais ne présente aucun intérêt avec la multiplication naïve.

Cette technique se généralise aux suites hypergéométriques quelconques ainsi qu'au calcul d'un terme d'une suite récurrente linéaire à coefficients polynomiaux. L'algorithme correspondant est présenté en Figure 1.

THÉORÈME 1. *L'algorithme « Pas de bébés, pas de géants » est correct. Il calcule le N^e terme d'une suite solution de la récurrence linéaire (1) à coefficients de degré au plus d en $O\big(r^\theta \mathsf{M}(\sqrt{dN})\log(dN)\big)$ opérations arithmétiques, où θ est un exposant faisable pour la complexité du produit de matrices, présenté au Chapitre 8.*

DÉMONSTRATION. L'algorithme calcule les produits $M = A(n-r)\cdots A(0)$ et $w = p_r(N-r)\cdots p_r(0)$ (pour ce dernier la récurrence est simplement $u_{n+1} = p_r(n)u_n$), puis le vecteur ${}^t(u_{N-r+1}, \dots, u_N)$ d'après la définition de A.

En ce qui concerne la complexité, elle se décompose étape par étape en :

1. le produit de matrices de polynômes en étape 1 a degré au plus md. Il est effectué récursivement par un arbre des sous-produits comme au chapitre 5. Son coût est donc borné par $O(\mathsf{MM}(r, md)\log m)$ opérations arithmétiques ;
2. l'évaluation en étape 2 est effectuée en réalisant séparément les évaluations multipoints des r^2 coefficients de la matrice P par la technique du Corollaire 5 page 68 en $O(r^2\frac{N}{m}\mathsf{M}(md)/(md))$ opérations arithmétiques ;
3. l'étape 3 effectue au plus m évaluations de matrices de degré au plus d, et est donc dominée par la précédente (qui calcule environ md évaluations de polynômes de degré md) ;
4. l'étape 4 multiple $O(m+N/m)$ matrices scalaires en $O(\mathsf{MM}(r)(m+N/m))$ opérations ;
5. l'étape suivante a une complexité au plus égale à celle des précédentes.

Au final, la complexité arithmétique de cet algorithme est donc majorée par

$$O\left(\mathsf{MM}(r, md)\log m + r^2\frac{N}{m}\frac{\mathsf{M}(md)}{md} + \mathsf{MM}(r)\sqrt{Nd}\right)$$
$$= O\left(r^\theta\mathsf{M}(\sqrt{Nd})\log(Nd) + r^2\sqrt{Nd}\frac{\mathsf{M}(\sqrt{Nd})}{\sqrt{Nd}} + r^\theta\sqrt{Nd}\right)$$
$$= O(r^\theta\mathsf{M}(\sqrt{Nd})\log(Nd)).$$

□

Exercice 1. Étant donné un polynôme $f \in \mathbb{K}[X]$ de degré 2 et un entier positif N, borner le nombre d'opérations dans $\mathbb{K}$ suffisantes pour déterminer le coefficient de X^N du polynôme f^N. (Indication : La solution consiste à calculer par exponentiation binaire tous les coefficients $u_0, \ldots, u_N$ de $f^N \bmod X^{N+1}$. Une approche encore plus rapide repose sur le fait que les u_n satisfont à une récurrence à coefficients polynomiaux d'ordre 2.)

2.1. Factorisation déterministe des entiers. Supposons que nous devons factoriser un entier N. Tester tous les diviseurs plus petits que $\sqrt{N}$ a une complexité binaire linéaire en $\sqrt{N}$. Afin d'accélérer ce calcul, on peut rassembler tous les entiers plus petits que $\sqrt{N}$ en $\sqrt[4]{N}$ blocs, chacun contenant $\sqrt[4]{N}$ entiers consécutifs. Soit c de l'ordre de $\sqrt[4]{N}$ et notons $f_0 = 1\cdots c \bmod N$, $f_1 = (c+1)\cdots(2c) \bmod N$, …, $f_{c-1} = (c^2-c+1)\cdots(c^2) \bmod N$. Si les valeurs $f_0, \ldots, f_{c-1}$ sont connues, alors il devient facile de déterminer un facteur de N, en prenant des pgcd de $f_0, \ldots, f_{c-1}$ avec N : la complexité binaire associée est $O\big(\sqrt[4]{N}\,\mathsf{M}_\mathbb{Z}(\log N)\log\log N\big)$.

Ainsi, la principale difficulté est de calculer les valeurs f_i. Pour ce faire, on prend $\mathbb{A} = \mathbb{Z}/N\mathbb{Z}$ et F le polynôme $(X+1)\cdots(X+c) \in \mathbb{A}[X]$, qu'on évalue en les points $0, c, 2c, \ldots, c(c-1)$ en $O(\mathsf{M}(c)\log c)$ opérations de $\mathbb{A}$. Par le théorème « évaluation interpolation », puisque c est d'ordre $\sqrt[4]{N}$, la complexité pour calculer les f_i est de $O(\mathsf{M}(\sqrt[4]{N})\log N)$ opérations modulo N, soit $O(\mathsf{M}(\sqrt[4]{N})\,\mathsf{M}_\mathbb{Z}(\log N)\log N)$ opérations bits. Ce terme est dominant et est donc aussi la complexité totale du calcul.

3. Scindage binaire

3.1. Cas de la factorielle. Pour exploiter la multiplication rapide, l'idée consiste à équilibrer les produits en calculant $P(a,b) = (a+1)(a+2)\cdots b$ récursivement par

$$P(a,b) = P(a,m)P(m,b) \quad \text{où} \quad m = \left\lfloor\frac{a+b}{2}\right\rfloor.$$

Appelons $C(a,b)$ le coût binaire du calcul de $P(a,b)$. Il résulte immédiatement de la méthode que ce coût vérifie l'inégalité

$$C(a,b) \leq C(a,m) + C(m,b) + \mathsf{M}_{\mathbb{Z}}\big(\log P(a,m), \log P(m,b)\big).$$

Sous l'hypothèse raisonnable que la complexité de la multiplication d'entiers est croissante avec la taille des entiers, le coût du calcul de $P(a,m)$ est inférieur au coût du calcul de $P(m,b)$, d'où la nouvelle inégalité

$$C(a,b) \leq 2C(m,b) + \mathsf{M}_{\mathbb{Z}}\big(P(m,b)\big).$$

À ce stade, le théorème « diviser pour régner » n'est pas suffisant pour conclure, mais l'utilisation de l'inégalité précédente donne les inégalités successives

$$\begin{aligned} C(0,n) &\leq 2C(n/2,n) + \mathsf{M}_{\mathbb{Z}}\big(\log P(n/2,n)\big) \\ &\leq 4C(3n/4,n) + 2\mathsf{M}_{\mathbb{Z}}\big(\log P(3n/4,n)\big) + \mathsf{M}_{\mathbb{Z}}\big(\log P(n/2,n)\big) \\ &\leq \cdots \\ &\leq 2^k C(n-2^{-k}n,n) + 2^k\mathsf{M}_{\mathbb{Z}}\big(\log P(n-2^{-k}n,n)\big) + \cdots + \mathsf{M}_{\mathbb{Z}}\big(\log P(n/2,n)\big), \end{aligned} \tag{3}$$

pour tout entier positif k. L'entier $P(n-2^{-k}n,n)$ est le produit de $2^{-k}n$ facteurs de taille bornée par $\log n$; il a donc une taille d'au plus $2^{-k}n\log n$. Par sous-additivité de la fonction $\mathsf{M}_{\mathbb{Z}}$, la dernière inégalité devient

$$C(0,n) \leq 2^k C(n-2^{-k}n,n) + k\mathsf{M}_{\mathbb{Z}}(n\log n).$$

En choisissant finalement d'arrêter la récursion lorsque $n-2^{-k}n$ et n diffèrent d'au plus un, ce qui impose k de l'ordre de $\log n$, on aboutit à la borne

$$C(0,n) \leq O(n) + \mathsf{M}_{\mathbb{Z}}(n\log n)\log n$$

donc à une complexité binaire pour le calcul de $N!$ dans $O\big(\mathsf{M}_{\mathbb{Z}}(N\log N)\log N\big)$.

Si la multiplication entière utilisée repose sur la FFT, cette complexité s'écrit $O(N\log^3 N\log\log N)$; si la multiplication entière utilisée est d'exposant α strictement plus grand que 1, alors elle s'écrit $O\big((N\log N)^\alpha\big)$: au lieu d'utiliser simplement la sous-additivité de la multiplication, on majore la somme des $\mathsf{M}_{\mathbb{Z}}$ dans l'inégalité (3) par une somme géométrique fois le dernier sommant.

3.2. Récurrences d'ordre 1. La factorielle de la section précédente suit la récurrence $u_{n+1} = (n+1)u_n$. On considère ici tout d'abord les solutions de récurrences de la forme

$$u_{n+1} = p(n)u_n, \qquad p \in \mathbb{Z}[X].$$

Si p a degré d, $\log p(N)$ est dans $O(d\log N)$, si bien que la taille de u_N est $O(dN\log N)$.

De même, pour toute récurrence de la forme

$$u_{n+1} = \frac{p(n)}{q(n)}u_n, \qquad p,q \in \mathbb{Z}[X],$$

on peut pour calculer le terme u_N appliquer séparément la même technique de scindage binaire sur le numérateur et le dénominateur. Si d est le maximum des degrés de p et q, le calcul produit deux entiers de taille $O(dN\log N)$ en un nombre d'opérations binaires en $O\big(\mathsf{M}_{\mathbb{Z}}(dN\log N)\log N\big)$.

À ce stade, on dispose d'un rationnel $u_N = P/Q$ dont le numérateur et le dénominateur sont des entiers potentiellement grands. Une évaluation numérique de ce rationnel à 2^{-e} près consiste à calculer $2^{-e}\lfloor 2^e P/Q\rfloor$. La méthode de Newton (Chapitre 4, théorème 2) permet de calculer l'inverse de Q a précision e en $O(\mathsf{M}_{\mathbb{Z}}(e))$ opérations. Il est donc possible de terminer le calcul par une approximation de u_N à précision e jusqu'à $O(dN\log N)$ sans affecter la complexité. Cette observation est

Scindage Binaire

Entrée : Une récurrence de la forme (1) avec $p_i \in \mathbb{Z}[X]$, $i = 0, \dots, r$, des conditions initiales rationnelles $u_0, \dots, u_{r-1}$, un entier N.

Sortie : Les termes $u_{N-r+1}, \dots, u_N$ de la solution.

1. Évaluer $A(N-r), \dots, A(1), A(0)$, où A est la matrice définie en page 215.
2. Calculer récursivement le produit $M_{N-r+1,0}$ de ces matrices par la formule
$$M_{i,j} = \begin{cases} A(j), & \text{si } i = j+1, \\ M_{i,k} \cdot M_{k,j}, & \text{avec } k = \lfloor (i+j)/2 \rfloor, \text{ sinon.} \end{cases}$$
3. Si $p_r(n) \neq 1$, calculer $w = p_r(N-r) \cdots p_r(0)$ par ce même algorithme ; sinon $w = 1$;
4. Renvoyer $\frac{1}{w} M \cdot {}^t(u_0, \dots, u_r)$.

FIGURE 2. Algorithme de scindage binaire.

exploitée dans les sections suivantes. (Le raisonnement est le même dans une base différente de 2 ; le changement de base a un coût légèrement plus que linéaire.)

3.3. Calcul de $e = \exp(1)$. Le point de départ est la suite (e_n) donnée par
$$e_n = \sum_{k=0}^{n} \frac{1}{k!}.$$
Cette suite converge vers e. Plus précisément, il est classique que le terme de reste dans $e - e_n$ se majore par une série géométrique de sorte que
$$0 \leq e - e_n < \frac{1}{(n+1)!} \left(1 + \frac{1}{n+1} + \frac{1}{(n+1)^2} + \cdots \right) = \frac{1}{n\, n!}.$$

Pour calculer N décimales de e par cette série, il suffit de rendre $\frac{1}{n\, n!}$ inférieur à 10^{-N}. Il s'ensuit qu'il suffit de prendre N de sorte que $n\, n!$ et 10^N soient du même ordre, c'est-à-dire d'avoir N proportionnel à $n \log n$. Il suffit donc de prendre $n = O(N/\log N)$ termes dans la série.

La suite $(e_n)_{n \geq 0}$ vérifie $e_n - e_{n-1} = 1/n!$ et donc
$$n(e_n - e_{n-1}) = e_{n-1} - e_{n-2}.$$
Cette récurrence se récrit
$$\begin{pmatrix} e_n \\ e_{n-1} \end{pmatrix} = \frac{1}{n} \underbrace{\begin{pmatrix} n+1 & -1 \\ n & 0 \end{pmatrix}}_{A(n)} \begin{pmatrix} e_{n-1} \\ e_{n-2} \end{pmatrix} = \frac{1}{n!} A(n) A(n-1) \cdots A(1) \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$
Le calcul du produit de matrices peut alors être effectué par scindage binaire. Le calcul de e_N par ce procédé demande donc $O(\mathsf{M}_{\mathbb{Z}}(N \log N) \log N)$ opérations binaires.

Pour calculer n décimales de e, la première étape ci-dessus construit deux entiers de taille $O(N \log N) = O(n)$ en $O(\mathsf{M}_{\mathbb{Z}}(n) \log n)$ opérations binaires. Il faut ensuite effectuer une division qui ne demande que $O(\mathsf{M}_{\mathbb{Z}}(n))$ opérations par l'algorithme de Newton. L'ensemble du calcul est donc quasi-optimal. (En outre, le $\log n$ n'est pas présent pour des multiplications comme celle de Karatsuba dont l'exposant de complexité est supérieur à 1.)

Tous les records récents de calculs de décimales de π reposent sur cette méthode (voir les notes en fin de chapitre).

3.4. Suites polynomialement récursives. Dans le cas général d'une suite P-récurrente vérifiant la récurrence (1) avec des coefficients polynomiaux p_i dont les coefficients sont entiers, l'algorithme est décrit en Figure 2.

THÉORÈME 2. *La complexité binaire de la méthode du scindage binaire pour calculer le N^e terme d'une suite solution de la récurrence (1), avec d une borne sur les degrés des polynômes p_i, et ℓ une borne sur la taille de leurs coefficients est en $O(r^\theta \mathsf{M}_\mathbb{Z}(dN\log N + \ell N + N\log r)\log N)$, où θ est un exposant faisable pour la complexité du produit de matrices, présenté au Chapitre 8.*

Comme précédemment, cette borne de complexité peut être améliorée à $O(r^\theta \mathsf{M}_\mathbb{Z}(dN\log N + \ell N + N\log r))$ si l'exposant α de la complexité $\mathsf{M}_\mathbb{Z}(m) = O(m^\alpha)$ est supérieur à 1.

EXERCICE 2. Vérifier les formules de ce théorème.

Exercices

EXERCICE 3 (Calcul rapide de factorielle et de coefficients binomiaux centraux). Cet exercice montre comment calculer certaines suites récurrentes linéaires plus vite que par la méthode de scindage binaire.

Soit $N \in \mathbb{N}$ et soit $Q = \sum_{i=0}^{2N} q_i X^i \in \mathbb{Z}[X]$ le polynôme $Q(X) = (1+X)^{2N}$.

1. Montrer que q_N peut être calculé en utilisant uniquement des additions d'entiers du triangle de Pascal, c'est-à-dire l'identité suivante sur les coefficients du binôme :
$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}, \qquad k \geq 1, n \geq 1.$$
Quelle est la complexité binaire de cet algorithme ?

On admet que le calcul de tous les nombres premiers inférieurs à N peut être effectué en $O(N\log N/\log\log N)$ opérations binaires, qu'il existe au plus $3N/\log(N)$ tels nombres premiers et que la multiplicité avec laquelle apparaît le nombre premier p dans la factorisation de $N!$ vaut
$$\mathrm{ind}(p,N) = \sum_{i=1}^{\infty} \left\lfloor \frac{N}{p^i} \right\rfloor,$$
où la notation $\lfloor x \rfloor$ représente la partie entière de x.

2. Montrer que le calcul de $\mathrm{ind}(p,N)$ peut être effectué en $O\big(\mathsf{M}_\mathbb{Z}(\log N)\log N\big)$ opérations binaires.
3. Montrer que la décomposition en facteurs premiers de $N!$ peut être effectuée en $O\big(\mathsf{M}_\mathbb{Z}(N)\log N\big)$ opérations binaires, ainsi que celle de q_N.
4. Montrer que $N!$ et q_N peuvent alors être reconstruits en respectivement $O\big(\mathsf{M}_\mathbb{Z}(N\log N)\log\log N\big)$ et $O\big(\mathsf{M}_\mathbb{Z}(N)\log N\big)$ opérations binaires.

EXERCICE 4. Soit $N \in \mathbb{N}$ et soit $P = \sum_{i=0}^{2N} p_i X^i \in \mathbb{Z}[X]$ le polynôme $P(X) = (1+X+X^2)^N$.

1. Montrer que $O(\mathsf{M}(N))$ opérations binaires suffisent pour déterminer la parité de tous les coefficients de P.
Indication : un entier n est pair si et seulement si $n = 0$ dans $\mathbb{K} = \mathbb{Z}/2\mathbb{Z}$.
2. Montrer que P vérifie une équation différentielle linéaire d'ordre 1 à coefficients polynomiaux. En déduire que les p_i suivent une récurrence d'ordre 2 que l'on précisera.
3. Donner un algorithme qui calcule p_N en $O(\mathsf{M}(N\log N)\log N)$ opérations binaires.

Notes

La méthode du scindage binaire est un grand classique en calcul formel. Elle a été redécouverte à plusieurs reprises dans la littérature, dans différents contextes : conversion d'une base à l'autre, calcul de décimales de constantes comme π ou e [5], calcul avec des récurrences linéaires [8], [6, §6].

L'une des premières références mentionnant le scindage binaire [1, §178] suggère déjà (sans preuve) qu'il permet l'évaluation numérique rapide des fonctions D-finies. L'article de synthèse de Bernstein [2, §12] est une bonne référence sur l'historique de la méthode.

L'algorithme par pas de bébés et pas de géants a été introduit par Strassen [9] et généralisé par les frères Chudnovsky [6] au problème du calcul d'un terme d'une suite récurrente linéaire à coefficients polynomiaux.

L'algorithme de factorisation déterministe en Section 2.1 est basé également sur l'article de Strassen [9]. Cet algorithme peut à son tour être un peu amélioré [4]. Notons qu'on ne connaît pas d'algorithme déterministe essentiellement meilleur ; l'existence d'un tel algorithme est un grand problème ouvert.

On trouve également dans [4] une application cryptologique du calcul du N^{e} terme d'une suite P-récurrente au problème du comptage de points sur une courbe hyperelliptique sur un corps fini. Le point de départ de cette application est l'exercice 1, inspiré de [7, Pb. 4].

Le résultat de la Section 3.3 est dû à Brent [5] ; le même raisonnement se généralise au calcul des sommes convergentes de suites hypergéométriques. En particulier, c'est ainsi que les systèmes de calcul formel calculent rapidement π par une formule découverte en 1989 par les frères Chudnovsky :

$$\frac{1}{\pi} = \frac{12}{C^{3/2}} \sum_{n=0}^{\infty} \frac{(-1)^n (6n)!(A+nB)}{(3n)!n!^3 C^{3n}}$$

où $A = 13591409$, $B = 545140134$ et $C = 640320$.

La partie de l'exercice 3 consacrée au calcul de $N!$ est due à P. Borwein [3].

Bibliographie

[1] Beeler, Michael, R. William Gosper et Richard Schroeppel (1972). *HAKMEM*. Artificial Intelligence Memo No. 239. Massachusetts Institute of Technology. URL : http://www.inwap.com/pdp10/hbaker/hakmem/hakmem.html.

[2] Bernstein, Daniel J. (2008). « Fast multiplication and its applications ». In : *Algorithmic number theory : lattices, number fields, curves and cryptography*. Vol. 44. Publications of the Research Institute for Mathematical Sciences. Cambridge University Press, p. 325–384.

[3] Borwein, Peter B. (1985). « On the complexity of calculating factorials ». In : *Journal of Algorithms*, vol. 6, n°3, p. 376–380.

[4] Bostan, Alin, Pierrick Gaudry et Éric Schost (2007). « Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator ». In : *SIAM Journal on Computing*, vol. 36, n°6, p. 1777–1806.

[5] Brent, Richard P. (1976). « Multiple-precision zero-finding methods and the complexity of elementary function evaluation ». In : *Analytic computational complexity*. Proceedings of a Symposium held at Carnegie-Mellon University, Pittsburgh, PA, 1975. Academic Press, p. 151–176.

[6] Chudnovsky, D. V. et G. V. Chudnovsky (1988). « Approximations and complex multiplication according to Ramanujan ». In : *Ramanujan revisited*. MA : Academic Press, p. 375–472.

[7] Flajolet, Philippe et Bruno Salvy (1997). « The SIGSAM Challenges : Symbolic Asymptotics in Practice ». In : *ACM SIGSAM Bulletin*, vol. 31, n°4, p. 36–47. doi : 10.1145/274888.274890.

[8] Kogge, P. et H. Stone (1973). « A parallel algorithm for the efficient solution of a general class of recurrence equations ». In : *IEEE Transactions on Computers*, vol. C-22, p. 786–793.

[9] Strassen, V. (1976/77). « Einige Resultate über Berechnungskomplexität ». In : *Jahresbericht der Deutschen Mathematiker-Vereinigung*, vol. 78, n°1, p. 1–8.

CHAPITRE 16

Résolution d'équations différentielles et de récurrences linéaires

Résumé

Les solutions polynomiales ou rationnelles d'équations différentielles linéaires ou de récurrences linéaires peuvent avoir une taille exponentiellement grande en la taille de l'équation. Néanmoins, elles s'obtiennent efficacement en passant des équations différentielles aux récurrences, en exploitant la structure de leurs espaces de solutions, en évitant de développer les polynômes concernés, et en exploitant le calcul rapide de factorielles de matrices.

Les chapitres précédents montrent que de nombreuses informations concernant les solutions d'équations différentielles ou de récurrences linéaires peuvent être calculées directement à partir de l'équation sans passer par une "résolution" de l'équation. Le cas particulier des solutions polynomiales et rationnelles mérite une attention spéciale : ces solutions, lorsqu'elles existent, permettent de nombreux calculs efficaces (notamment le prolongement analytique). D'autre part, décider de leur existence est une brique de base dans les algorithmes de sommation et d'intégration de la partie 6.

Trois types d'équations sont considérés ici :

— homogènes, c'est-à-dire selon le cas de l'une des deux formes

$$c_0(x)y^{(m)}(x) + \cdots + c_m(x)y(x) = 0, \tag{1}$$

$$a_0(n)u_{n+r} + \cdots + a_r(n)u_n = 0, \tag{2}$$

où les coefficients a_i et c_i sont des polynômes donnés à coefficients dans un corps $\mathbb{K}$;

— inhomogènes, c'est-à-dire du même type, mais avec un membre droit qui n'est plus simplement 0, mais un polynôme $b(x)$ ou une suite $b(n)$ qui peut être à support fini (voir ci-dessous) ou un polynôme ;

— inhomogènes paramétrées, où le membre droit est maintenant de la forme $\lambda_1 b_1 + \cdots + \lambda_k b_k$, où les b_i sont donnés, et les coefficients λ_i sont inconnus.

Les solutions étudiées sont soit

— à support fini (dans le cas des suites) ;

— polynomiales ;

— rationnelles ;

— séries (dans le cas des équations différentielles).

Enfin, les questions à aborder concernant ces solutions portent sur leur existence, leur valeur (et la structure de données à utiliser pour représenter cette valeur), la complexité arithmétique et binaire des calculs correspondants. En outre, dans le cas inhomogène paramétré, il s'agira de déterminer l'existence et la valeur des coefficients λ_i permettant l'existence d'une solution polynomiale ou rationnelle de l'équation.

1. Suites et séries

1.1. Espace des suites solutions de la récurrence linéaire homogène. On considère des suites (u_n) dont les valeurs sont reliées par (2) pour tout $n \in \mathbb{N}$.

LEMME 1. *L'ensemble des suites solutions de la récurrence* (2) *forme un espace vectoriel sur* $\mathbb{K}$.

DÉMONSTRATION. La combinaison linéaire de solutions reste une solution, il s'agit donc bien d'un sous-espace vectoriel de $\mathbb{K}^{\mathbb{N}}$. □

La dimension de cet espace est plus délicate, comme le montre l'exemple suivant.

EXEMPLE 1. Si la suite (u_n) obéit à la récurrence

$$n(n-1)u_{n+3} - (n-1)u_{n+2} + (n+1)u_{n+1} + 2nu_n = 0, \qquad n \in \mathbb{N},$$

alors les seules contraintes induites par la récurrence pour $n = 0$ et 1 sont

$$u_2 + u_1 = 0, \quad 2u_2 + 2u_1 = 0$$

alors qu'à partir de $n = 2$, la récurrence force la valeur de u_{n+3}. Autrement dit, les valeurs u_0, u_1, u_3, u_4 sont libres, et les autres s'en déduisent. La dimension de l'espace des solutions est donc 4 pour cette récurrence d'ordre 3.

PROPOSITION 1. *La dimension de l'espace des solutions de la récurrence* (2) *est comprise entre* r *et*

$$r + \operatorname{card} H \quad \text{où} \quad H := \{h \in \mathbb{N} \mid a_0(h) = 0\}.$$

DÉMONSTRATION. Si n n'appartient pas à l'ensemble $J := \{0, \dots, r-1\} \cup H + r$ (ici $H + r$ désigne la somme, c'est-à-dire $\{h \in \mathbb{N} \mid h \geq r \text{ et } a_0(h-r) = 0\}$), alors la valeur de u_n est fixée par les précédentes. À chaque $h \in J$ est associée une suite $(u_n^{(h)})$ définie par $u_h^{(h)} = 1$, $u_k^{(h)} = 0$ pour $k \in J \setminus \{h\}$, et l'équation (2) pour $n \notin H + r$. Dans l'espace vectoriel de dimension $r + \operatorname{card} H$ engendré par ces suites, ne sont solutions de la récurrence (2) que celles qui obéissent en outre aux contraintes linéaires imposées par (2) pour $n \in H + r$. Le nombre de telles relations linéairement indépendantes est compris entre 0 et $\operatorname{card} H$. □

1.2. La forme d'une suite P-récursive. Dans le cas fréquent où les coefficients de la récurrence (2) sont des polynômes à coefficients entiers, les tailles des éléments de la suite croissent avec leur indice dans le cas général. Ces questions de taille ont été traitées au chapitre 15. Il en ressort que la taille en bits du N^{e} élément de la suite croît en $O(dN \log N)$ où d est le degré des coefficients de la récurrence, et que la taille totale des N premiers éléments est bornée par $O(dN^2 \log N)$. Lorsque N est grand, ces tailles ont un impact sur les complexités, et il sera utile de concevoir des algorithmes qui opèrent sur ces suites sans nécessairement en représenter explicitement les éléments. En effet, les indices pertinents dans ce chapitre proviennent de solutions entières de polynômes comme dans la Proposition 1, et ces solutions peuvent être exponentiellement grandes en la taille binaire de la récurrence (le polynôme $n - N$ est un exemple frappant).

1.3. Séries solutions d'équations différentielles linéaires homogènes.

Traduction en récurrence. Le théorème 1 p. 200 montre que la suite des coefficients d'une série formelle solution de (1) vérifie une récurrence de type (2). Si les coefficients de l'équation différentielle sont

$$c_i(x) = \sum c_{ij} x^j,$$

le calcul explicite de la récurrence vérifiée par les coefficients (y_n) de la série $y(x)$ donne

$$\sum_{i,j} c_{ij}(n-j+1)\cdots(n-j+m-i)y_{n+m-i-j} = 0, \tag{3}$$

pour tout $n \in \mathbb{Z}$, avec $y_\ell = 0$ pour $\ell < 0$. Quitte à décaler les indices, cette récurrence se récrit sous la forme

$$a_0(n)y_{n+r} + \cdots + a_r(n)y_n = 0, \tag{4}$$

à laquelle il faut ajouter les équations

$$a_0(-1)y_{r-1} + \cdots + a_{r-1}(-1)y_0 = 0, \quad a_0(-2)y_{r-2} + \cdots = 0, \ldots, \quad a_0(-r)y_0 = 0 \tag{5}$$

correspondant aux contraintes $y_\ell = 0$ pour $\ell < 0$. (Voir l'exemple 4 p. 201.)

La complexité de cette traduction (donnée au Chapitre 14) est en $O((r+d)\mathsf{M}(d)\log d)$ opérations arithmétiques, quitte à utiliser des algorithmes rapides si r ou d sont un peu gros.

Definition 1. Le polynôme $a_0(n-r)$ est appelé *polynôme indiciel* de l'équation (1) en 0.

Exercice 1. Le polynôme $a_r(n)$ est appelé polynôme indiciel à l'infini. En changeant la variable x en $x+\alpha$ dans l'équation, le même calcul fournit deux polynômes. Le premier s'appelle polynôme indiciel en α. Montrer que le polynôme indiciel à l'infini est inchangé.

Séries solutions.

Proposition 2. *Si $F \in \mathbb{K}[[x]]$ est solution de l'équation différentielle (1) alors sa valuation annule le polynôme indiciel de (1) en 0.*

Démonstration. Il suffit d'évaluer la récurrence (4) en $n = \operatorname{val} F - r$. □

Corollaire 1. *Si le coefficient de tête c_0 de l'équation différentielle (1) est non-nul en 0, alors cette équation admet une base de m séries solutions ayant pour valuations $\{0, 1, \ldots, m-1\}$.*

Démonstration. L'équation (3) montre que lorsque $c_{00} \neq 0$, le polynôme indiciel en 0 est donné par

$$a_0(n-r) = c_{00}(n-m+1)\cdots(n).$$

Les équations (5) puis (4) permettent de conclure qu'il existe une unique solution avec les conditions initiales $y_i = \delta_{ij}$ pour $j = 0, \ldots, m-1$. D'après la proposition précédente, toutes les solutions s'obtiennent donc comme combinaisons linéaires de celles-ci, qui sont indépendantes puisque de valuations différentes. □

Séries généralisées solutions. Les calculs ci-dessus s'étendent facilement à une classe un peu plus grande de solutions.

DEFINITION 2. On appelle *série généralisée* une série de la forme $x^\alpha F$, où $F \in \mathbb{K}[[x]]$ et $\alpha \in \overline{\mathbb{K}}$.

PROPOSITION 3. *Si l'équation différentielle* (1) *admet une solution série généralisée alors* α *est racine du polynôme indiciel de* (1) *en 0. Si de plus,* $\alpha \notin \{0, 1, \dots, m-1\}$*, alors* $c_0(0) = 0$.

DÉMONSTRATION. L'équation

$$x^{-\alpha}\Big(c_0(x)(x^\alpha F)^{(m)} + \cdots + c_m(x)x^\alpha F\Big) = 0$$

est de même type que (1). De plus, la récurrence (4) qui lui correspond est obtenue en remplaçant n par $n-\alpha$ dans (4). Le résultat est alors un corollaire des deux résultats précédents. □

Translation. Les solutions séries généralisées de la forme

$$y(t) = (t-a)^\alpha \sum_{i=0}^{\infty} y_i (t-a)^i \qquad \text{avec } y_0 \neq 0$$

se ramènent au cas précédent en effectuant le changement d'inconnue $y(a+t) = \tilde{y}(t)$ qui fournit l'équation

$$c_0(a+t)\tilde{y}(t)^{(m)} + \cdots + c_m(a+t)\tilde{y}(t) = 0. \tag{6}$$

Les résultats obtenus jusqu'ici établissent donc la proposition suivante.

PROPOSITION 4. *Si l'équation différentielle* (1) *admet une solution de la forme* $(t-a)^\alpha \sum_{i=0}^{\infty} y_i(t-a)^i$ *avec* $y_0 \neq 0$ *et* $\alpha \notin \{0, \dots, m-1\}$ *alors* $c_0(a) = 0$ *et* α *est racine du polynôme indiciel de* (6).

DEFINITION 3. On appelle polynôme indiciel de l'équation différentielle (1) en a celui de l'équation (6) en 0.

DEFINITION 4. On dit que $\alpha \in \overline{\mathbb{K}}$ est un point *ordinaire* de l'équation (1) si $c_0(\alpha) \neq 0$. On dit qu'il est *singulier* dans le cas contraire. On dit que l'infini est un point singulier de l'équation si 0 est singulier pour l'équation satisfaite par $y(1/x)$.

EXEMPLE 2. La fraction rationnelle $y = 1/(1-x)$ est solution de l'équation

$$(1-x)y'(x) - y(x) = 0.$$

Le complexe 1 est singularité de la solution et donc nécessairement point singulier de l'équation, ce qui s'y traduit par l'annulation du coefficient de tête.

EXEMPLE 3. Le polynôme x^{10} est solution de l'équation

$$10xy'(x) - y(x) = 0.$$

La solution n'a pas de point singulier à distance finie, mais le complexe 0 est point singulier de l'équation ; le théorème de Cauchy ne s'y applique pas.

Preuves de non-D-finitude. Une conséquence utile de la Proposition 4 est que les fonctions D-finies ne peuvent avoir qu'un nombre fini de singularités (les racines du coefficient de tête). Il s'en déduit des résultats négatifs.

EXEMPLE 4. Les fonctions $1/\sin x$ et $\sqrt{\sin x}$ ne satisfont pas d'équation différentielle linéaire à coefficients polynomiaux.

EXEMPLE 5. La suite des nombres de Bernoulli, qui interviennent en particulier dans la formule d'Euler-Maclaurin, ne peut être solution d'une récurrence linéaire à coefficients polynomiaux, puisque la série génératrice exponentielle

$$\sum_{n=0}^{\infty} B_n \frac{z^n}{n!} = \frac{z}{\exp(z)-1}$$

a une infinité de pôles (aux $2ik\pi$, $k \in \mathbb{Z}^\star$).

D'autres types de preuves de non-D-finitude sont présentés p. 208.

2. Solutions à support fini

DEFINITION 5. Une suite (u_n) est *à support fini* s'il existe $N \in \mathbb{N}$ tel que $u_n = 0$ pour tout $n > N$. On appelle alors N le *degré* de (u_n) si $u_N \neq 0$. Pour la suite (0), on convient que le degré est infini.

2.1. Récurrences homogènes.

PROPOSITION 5. *Si (u_n) est une solution à support fini de la récurrence (2), alors son degré est racine du polynôme a_r. Réciproquement, si a_r a K racines dans $\mathbb{N}$, alors la récurrence possède un espace de solutions à support fini de dimension comprise entre 1 et K.*

DÉMONSTRATION. C'est la même idée que pour la Proposition 2 : il suffit d'évaluer la récurrence en $n = N$. La réciproque est similaire à la Proposition 1 : l'évaluation de la récurrence en une racine entière h de a_r ne contraint pas u_h, mais crée une contrainte linéaire sur les valeurs suivantes. Selon si cette contrainte est linéairement dépendante ou non des autres, elle permet ou non d'augmenter la dimension. □

EXEMPLE 6. Voici un cas extrême où toutes les solutions sont à support fini :

$$u_{n+2} + (n-7)u_{n+1} + (n-10)(n-5)u_n = 0,$$

(exercice : le vérifier) mais changer le coefficient 7 en 6 par exemple fait chuter la dimension des solutions à support fini à 1, et le degré à 5.

Algorithme naïf. La méthode la plus directe consiste donc à calculer d'abord l'ensemble

$$S := \{h \in \mathbb{N} \mid a_r(h) = 0\}.$$

Si cet ensemble est vide, alors il n'y a pas de solution à support fini. Sinon, $N = \max S$ est une borne sur le degré et une méthode de coefficients indéterminés mène à un système linéaire dont les équations sont les évaluations de la récurrence (2) en $n = 0, \ldots, N$ ainsi que $u_{N+i} = 0$ pour $i = 1, \ldots, r$, et les inconnues sont les u_i pour $i = 0, \ldots, N + r$.

Le calcul des équations requiert l'évaluation des coefficients de la récurrence, ce qui prend $O(Nrd)$ opérations par la méthode naïve si les coefficients ont un degré borné par d, puis la résolution peut être effectuée par exemple en considérant les suites (u_n^h) pour $h \in S$ définies par

$$u_n^h = \begin{cases} 0 & \text{si } n > h \text{ ou si } n \neq h \text{ et } n \in S \\ 1 & \text{si } n = h \\ \text{est défini par la valeur de (2) sinon.} \end{cases}$$

Alors les solutions à support fini sont de la forme $(\sum_{h\in S} \lambda_h u_n^h)$. Les coefficients λ_h sont solutions du système linéaire fourni par les évaluations en $n \in S$ de la récurrence. La complexité de cette étape est en $O(K^\theta)$. Au final, cette méthode naïve

demande donc $O(Nrd + K^\theta)$ opérations arithmétiques. Si K est plus grand que la dimension fournie par la Proposition 1, alors on définit les suites en tenant compte de a_0 plutôt que de a_r, ce qui revient à dérouler la récurrence dans le sens direct. La complexité s'exprime de la même façon, avec K la dimension de l'espace des suites solutions.

Complexité. L'entier N peut être au pire de taille exponentielle en la taille binaire de l'équation. La dépendance linéaire en N de la complexité ci-dessus peut donc devenir problématique.

Dans le cas où la récurrence n'a pas de singularité (le coefficient de tête a_0 n'a pas de racine dans $\mathbb{N}$), alors la complexité arithmétique peut être abaissée facilement en utilisant l'algorithme « Pas de bébés, pas de géants » du Chapitre 15, ce qui permet de calculer la matrice de passage des conditions initiales aux valeurs de la suite aux indices $N+1, \dots, N+r$ en $O(r^\theta \mathsf{M}(\sqrt{d(N+r)}) \log(d(N+r)))$ opérations arithmétiques. Le noyau de cette matrice, calculé en $O(r^\theta)$ opérations arithmétique supplémentaires, donne les conditions initiales des solutions à support fini. Cette méthode passe donc d'une complexité linéaire en N à une complexité quasi-linéaire en $\sqrt{N}$.

Conditions initiales généralisées. Cette méthode s'étend au cas où le coefficient de tête a_0 a des racines dans $\{0, \dots, N-r\}$. La dimension de l'espace de travail augmente alors en introduisant les suites (v_n^h) définies pour

$$h \in H := \{0, \dots r\} \cup \{h \in \{r, \dots, N\} \mid a_0(h-r) = 0\}$$

par

$$v_n^h = \begin{cases} 0 & \text{si } n < h \text{ ou si } n \neq h \text{ et } a_0(n-r) = 0 \\ 1 & \text{si } n = h \\ \text{est défini par la valeur de (2) évalué en } n-r & \text{sinon.} \end{cases}$$

Il s'agit ensuite de trouver les combinaisons linéaires de ces suites (v_n^h) qui ont support fini.

Definition 6. Soient $0 < 1 < \cdots < r-1 < h_1 < \cdots < h_M$ les éléments de H. On appellera *conditions initiales généralisées* de la récurrence (2) pour le support $\{0, \dots, N\}$ les valeurs μ_i, $i \in H$ telles qu'il existe au plus une solution de la récurrence sur l'intervalle $\{0, \dots, N\}$ avec $u_i = \mu_i$ pour tout $i \in H$.

Des équations sur ces conditions initiales généralisées s'obtiennent de proche en proche. La matrice de passage $M_0^{h_1}$ de $0, \dots, r-1$ à $h_1 - r, \dots, h_1 - 1$ permet de calculer les valeurs $v_{h_1-1}^0, \dots, v_{h_1-1}^{r-1}$. Les valeurs en h_1 sont fixées par la définition des v_n^h. La valeur en $(h_1 - r + 1, \dots, h_1)$ de la suite $(\mu_0 v_n^0 + \cdots + \mu_{r-1} v_n^{r-1} + \mu_{h_1} v_n^{h_1})$ est donc donnée par

$$(0|\operatorname{Id}_r) \begin{pmatrix} M_0^{h_1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mu_0 \\ \vdots \\ \mu_{r-1} \\ \mu_{h_1} \end{pmatrix}.$$

On peut ensuite multiplier la matrice $(r, r+1)$ ainsi définie par la matrice de passage $M_{h_1}^{h_2}$ et construire une matrice $(r, r+2)$ de la même façon, et ainsi de suite jusqu'en N pour obtenir une matrice $(r, r+M)$ dont on cherche un noyau. La dépendance en N de la complexité des calculs et multiplications de matrices est bornée par $O(Mr^\theta \mathsf{M}(\sqrt{dN}) \log(dN))$ opérations arithmétiques (et $M \leq d$). On peut remplacer ce facteur M par K ci-dessus s'il est plus petit, en déroulant la

récurrence des grands indices vers les petits. Ainsi, dans le cas singulier aussi, la complexité arithmétique est quasi-linéaire en $\sqrt{N}$.

Si les coefficients des polynômes sont eux-mêmes des entiers, la complexité binaire peut aussi être évaluée. Il vaut mieux d'abord effectuer un calcul modulo un nombre premier (pas trop petit) pour s'assurer qu'il peut exister une solution à support fini, en complexité quasi-linéaire en $\sqrt{N}$, puis utiliser la méthode de scindage binaire présentée au Chapitre 15 pour obtenir les coefficients μ_i correspondant aux solutions à support fini, ce qui donne une complexité binaire quasi-linéaire en N. Les tailles des éléments de la matrice à résoudre sont en $O(dN \log N)$, ce qui est donc aussi l'ordre de grandeur de la taille des conditions initiales lorsqu'existent des solutions à support fini. La taille totale de la solution reste donc bornée par $O(dN^2 \log N)$, mais le calcul des conditions initiales est en $O(r^\theta \mathsf{M}_\mathbb{Z}(dN \log N)$ opérations binaires seulement.

2.2. Récurrences inhomogènes. Si N est le degré d'une suite (u_n) à support fini solution de la récurrence linéaire inhomogène

$$a_0(n)u_{n+r} + \cdots + a_r(n)u_n = b(n),$$

alors pour $n > N$ nécessairement $b(n) = 0$. Donc cette récurrence ne peut avoir de solution à support fini que si la suite $(b(n))$ est elle-même à support fini, et le degré de (u_n) est majoré par le maximum des degrés des solutions à support fini de la partie homogène, et par le degré de la suite $(b(n))$. Une fois cette borne obtenue, la méthode naïve du cas homogène s'applique sans modification pour calculer les solutions à support fini. Elle mène alors à un système linéaire inhomogène pour les conditions initiales, et la complexité est encore en $O(Nrd + M^\theta)$.

Si l'entier N est très grand par rapport au degré de la suite $(b(n))$, on peut d'abord dérouler la récurrence comme dans la méthode naïve jusqu'à ce degré, puis utiliser les techniques du cas homogène (« pas de bébés, pas de géants » ou scindage binaire) pour résoudre en bonne complexité vis-à-vis de N.

2.3. Récurrences inhomogènes paramétrées. À nouveau, le degré d'une solution à support fini est majoré par le maximum des degrés des suites $(b_i(n))$ du membre droit et la borne de la partie homogène. Tout se déroule donc comme dans le cas inhomogène. La différence provient du système linéaire à résoudre à la fin, qui est cette fois-ci linéaire non seulement en les conditions initiales mais aussi en les coefficients λ_i.

3. Solutions polynomiales

3.1. Équations différentielles. Les solutions polynomiales de l'équation différentielle linéaire (1) ont des suites de coefficients à support fini solutions de la récurrence linéaire (4) qui s'en déduit par traduction, avec les contraintes sur les conditions initiales (5). Elles peuvent donc être calculées par les techniques de la section précédente. Les cas inhomogènes et inhomogènes paramétrés avec des membres droits polynomiaux passent à leurs analogues sur les suites dans cette traduction.

3.2. Récurrences linéaires. Le principe est le même : on commence par trouver une borne sur le degré des solutions, puis la solution se ramène par coefficients indéterminés à un système linéaire.

Borne sur le degré.

Exemple 7. Si la récurrence

$$(n-3)u_{n+2} - (2n-3)u_{n+1} + nu_n = 0$$

a une solution qui se comporte pour n grand comme $u_n \sim n^k$, alors le membre gauche est un polynôme de degré $k+1$. L'extraction des coefficients de degré $k+1, k, k-1, \dots$ fournit des équations qui doivent être satisfaites par une solution polynomiale. Il s'agit de

$$\begin{aligned} \text{degré } k+1 : &\quad 1-2+1=0, \\ \text{degré } k : &\quad 2k-3-2k+3=0, \\ \text{degré } k-1 : &\quad 2k(k-1)-6k-k(k-1)+3k = k(k-4) = 0, \end{aligned}$$

donc une solution polynomiale ne peut avoir que degré 0 ou 4.

Un algorithme pour ce calcul s'obtient un peu plus facilement en récrivant les décalages $u(n+k)$ de la suite initiale en terme de différences finies. Ainsi, on note $(\Delta \cdot u)(n) = u(n+1) - u(n)$ et, par récurrence, $(\Delta^{k+1} \cdot u)(n) = (\Delta^k \cdot u)(n+1) - (\Delta^k \cdot u)(n)$. La récurrence à annuler prend la forme

$$L \cdot u = \sum_{k=0}^{m} b_k(n) \Delta^k u = 0$$

pour de nouveaux polynômes b_k. L'opérateur Δ fait décroître de 1 exactement le degré des polynômes. Ainsi,

$$\deg(\Delta^k P) \leq \max(\deg P - k, 0),$$

avec égalité tant que $\Delta^k P$ n'est pas nul, et donc $\deg(L \cdot P) \leq \deg P + \max_k \{\deg(b_k) - k\}$. Soient alors les quantités

$$b := \max_k \{\deg(b_k) - k\}, \qquad E := \{\, k \mid \deg(b_k) - k = b \,\},$$

qui vont servir à fournir une borne sur le degré des solutions.

Exemple 8. La récurrence de l'exemple précédent se récrit

$$(n-3)(u_{n+2} - 2u_{n+1} + u_n) + (2(n-3) - (2n-3))u_{n+1} + (3-n-3+n)u_n = 0,$$

qui se simplifie en

$$\big((n-3)\Delta^2 - 3\Delta\big)\,(u_n) = 0$$

l'entier b vaut 0 et l'ensemble E est $\{1, 2\}$.

Soit D le degré d'une solution. La discussion distingue deux cas :

— soit $D + b < 0$, et alors $-(b+1)$ est une borne sur D ;

— sinon le coefficient de degré $D+b$ dans $L(n^D + \cdots)$ vaut

$$\sum_{k \in E} \mathrm{lc}(b_k) D(D-1) \cdots (D-k+1),$$

où lc désigne le coefficient de tête (*leading coefficient*). Cette expression, vue comme un polynôme en D, s'appelle *le polynôme indiciel* de la récurrence, lequel est non nul.

Cette discussion mène au résultat suivant.

Proposition 6. *Une borne sur le degré des solutions polynomiales de l'opérateur $L = \sum b_k(n)\Delta^k$ est donnée par le maximum de $-(b+1)$ et de la plus grande racine entière positive du polynôme indiciel de L.*

Exemple 9. Dans l'exemple précédent, le polynôme indiciel vaut

$$D(D-1) - 3D = D(D-4),$$

ce qui redonne bien le résultat espéré.

Algorithme naïf. Un algorithme simple consiste alors à calculer cette borne et à rechercher ensuite les solutions par un calcul d'algèbre linéaire.

EXERCICE 2. Trouver les solutions polynomiales de la récurrence

$$3u(n+2) - nu(n+1) + (n-1)u(n) = 0.$$

Base binomiale. La borne sur le degré peut être exponentiellement grande en la taille binaire de la récurrence.

EXEMPLE 10. La récurrence

$$nu(n+1) - (n+100)u(n) = 0$$

a pour solution le polynôme $u(n) = n(n+1)\cdots(n+99)$ de degré 100.

Comme dans les sections précédentes, il est alors souhaitable de disposer d'algorithmes permettant de détecter l'existence de solutions polynomiales en complexité maîtrisée par rapport à N. De plus, lorsqu'existent des solutions polynomiales de degré N, les développer utilise $O(N)$ coefficients, alors qu'elles peuvent être représentées de manière compacte par la récurrence et les conditions initiales correspondantes.

Dans le cas différentiel, ces conditions initiales étaient obtenues efficacement en utilisant la récurrence sur les coefficients des séries solutions. Cette propriété ne subsiste pas pour les solutions polynomiales de récurrences linéaires : leurs coefficients dans la base $1, n, n^2, \ldots$ ne vérifient pas une récurrence linéaire d'ordre borné a priori. En revanche, cette propriété réapparaît si l'on développe les polynômes dans la base binomiale.

PROPOSITION 7. *Si (u_n) est une solution polynomiale de la récurrence linéaire (2), alors ses coefficients dans la base binomiale $\{\binom{n}{k}, k \in \mathbb{N}\}$ sont solution à support fini d'une récurrence linéaire.*

DÉMONSTRATION. Le cœur de la preuve se résume au calcul suivant.

LEMME 2. *Si $u_n = \sum_{k=0}^{d} c_k \binom{n}{k}$ alors*

$$nu_n = \sum_{k=0}^{d+1} k(c_k + c_{k-1})\binom{n}{k},$$

$$u_{n+1} = \sum_{k=0}^{d} (c_k + c_{k+1})\binom{n}{k}.$$

DÉMONSTRATION. La relation

$$\binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k}$$

se récrit en

$$n\binom{n}{k} = (k+1)\binom{n}{k+1} + k\binom{n}{k}.$$

Le premier résultat s'obtient en injectant cette réécriture dans la somme nu_n et en extrayant le coefficient de $\binom{n}{k}$.

La seconde propriété découle directement du triangle de Pascal, qui s'obtient lui-même en extrayant le coefficient de X^k dans l'identité $(1+X)^{n+1} = (1+X)^n(1+X)$, ce qui donne

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n-1}{k-1}.$$

□

Algorithme de Liouville

Entrée : Une équation différentielle linéaire de la forme (1).

Sortie : Une base de ses solutions rationnelles.

1. En toute racine a de c_0 :
   - calculer le polynôme indiciel $p_a(n)$;
   - calculer la plus petite racine entière négative N_a de p_a, s'il n'en existe pas, faire $N_a := 0$;
2. Former le polynôme $P = \prod_{c_0(a)=0}(x-a)^{-N_a}$;
3. Effectuer le changement de fonction inconnue $y = Y/P$ et réduire au même dénominateur ;
4. Chercher une base B des solutions polynomiales de cette nouvelle équation. Une base des solutions rationnelles est formée des fractions $\{b/P, b \in B\}$.

FIGURE 1. Algorithme de Liouville pour les solutions rationnelles des équations différentielles linéaires.

En utilisant répétitivement ces deux propriétés, tout terme $n^i u_{n+j}$ apparaissant dans une récurrence se traduit en une portion de récurrence pour les coefficients, d'où le résultat de la proposition. □

Les algorithmes qui découlent de cette observation sont donc exactement les mêmes que pour les équations différentielles linéaires, y compris pour les cas inhomogènes et paramétrés.

4. Solutions rationnelles

4.1. Équations différentielles : l'algorithme de Liouville. En un pôle a d'une solution rationnelle (c'est-à-dire un zéro de son dénominateur), la solution admet un développement en série généralisée de la forme $(t-a)^\alpha \sum y_i (t-a)^i$ avec $a \in \mathbb{Z} \setminus \mathbb{N}$. D'après la Proposition 4, ces pôles sont donc faciles à localiser : ce sont des racines du coefficient de tête de l'équation différentielle, et leur multiplicité est bornée par la plus petite racine entière négative du polynôme indiciel en ces points.

Ceci conduit à l'algorithme présenté en Figure 1, essentiellement dû à Liouville, pour calculer les solutions rationnelles de (1). La preuve de l'algorithme se réduit à observer que le polynôme P qu'il calcule est un multiple du dénominateur de toute solution rationnelle. Du point de vue de la complexité, il faut noter que bien que les exposants N_a puissent avoir une valeur exponentielle en la taille de l'équation différentielle, le polynôme P n'a pas besoin d'être développé, et l'équation obtenue par changement de fonction inconnue grossit indépendamment des N_a, puisque la dérivée logarithmique de P est une fraction rationnelle avec un numérateur et un dénominateur de degré au plus d.

EXERCICE 3 (Système et équation). Ramener la résolution du système

$$Y'(x) = A(x)Y(x), \tag{7}$$

où $A(x)$ est une matrice de fractions rationnelles de $\mathbb{K}(x)$ et Y un vecteur à la résolution d'équations par la méthode de Liouville.

EXERCICE 4. Trouver une équation différentielle linéaire satisfaite par y_1 solution de

$$y_1' = xy_1 - y_2, \quad y_2' = y_1 - xy_2.$$

Le cas inhomogène. Lorsque le membre droit est un polynôme, le polynôme P trouvé par l'algorithme 1 reste un multiple du dénominateur des solutions rationnelles. En revanche, l'étape de réduction au même dénominateur s'écrit

$$A(x, Y(x)) = P(x) \sum_i c_i \frac{d^{m-i}}{dx^{m-i}} \frac{Y(x)}{P(x)} = P(x)b(x)$$

et risque de faire exploser le degré des coefficients. Pour éviter cette explosion, il suffit de dériver encore une fois l'équation, et d'effectuer une combinaison linéaire :

$$\begin{aligned} b(x)\frac{d}{dx}A(x, Y(x)) - \left(b(x)\frac{P'(x)}{P(x)} + b'(x)\right) A(x, Y(x)) \\ = b(x)(P(x)b(x))' - \left(b(x)\frac{P'(x)}{P(x)} + b'(x)\right) P(x)b(x) = 0. \end{aligned}$$

Cette méthode s'applique aussi au cas inhomogène paramétré, et l'équation demeure linéaire en les paramètres.

4.2. Récurrences : l'algorithme d'Abramov. Le principe est le même que pour les équations différentielles : trouver d'abord un multiple du dénominateur, puis effectuer un changement de suite inconnue et chercher une solution polynomiale à la nouvelle équation ainsi obtenue pour le numérateur.

Pour déterminer un multiple du dénominateur, il faut d'abord repérer les positions des pôles éventuels d'une solution rationnelle. L'intuition de départ s'obtient en considérant le cas de fractions dans le plan complexe. Si $u(n) = P(n)/Q(n)$ est solution de la récurrence inhomogène

$$a_0(n)\frac{P(n+r)}{Q(n+r)} + \cdots + a_r(n)\frac{P(n)}{Q(n)} = b(n) \tag{8}$$

alors chaque racine a de $Q(n)$ donne lieu à une racine $a-1$ de $Q(n+1)$, une racine $a-2$ de $Q(n+2)$ et ainsi de suite jusqu'à une racine $a-r$ de $Q(n+r)$. Pourtant, le membre gauche de la récurrence doit rester un polynôme. Cela signifie que ces racines doivent correspondre à des racines des coefficients correspondants. Si aucune paire de racines de Q ne différait d'un entier, alors le dénominateur serait nécessairement un multiple du pgcd des polynômes $a_0(n-r), \ldots, a_{r-1}(n-1), a_r(n)$. La situation est plus complexe si deux racines de Q diffèrent d'un entier. Cependant, si α et β sont deux racines de Q telles que $\alpha - \beta = k \in \mathbb{N}$ est maximal, alors nécessairement $a_r(\alpha) = 0$ et $a_0(\beta - r) = 0$. La plus grande différence entière H entre les racines de Q peut donc être bornée étant donnés a_0 et a_r.

Cette observation peut être raffinée et abstraite du contexte du plan complexe. Soit

$$M(n) = \operatorname{ppcm}(Q(n+1), \ldots, Q(n+m)).$$

La récurrence multipliée par $M(n)$ s'écrit

$$a_r(n)P(n+r)\frac{M(n)}{Q(n+r)} + \cdots + a_r(n)P(n)\frac{M(n)}{Q(n)} = 0.$$

Dans cette récurrence, tous les coefficients $M(n)/Q(n+k)$, $k = 1, \ldots, r$ sont des polynômes. Par conséquent Q divise $a_r M$, ce qui se récrit

$$Q(n) \mid a_r(n) \operatorname{ppcm}(Q(n+1), \ldots, Q(n+m)).$$

En décalant n et en reportant, il vient

$$\begin{aligned} Q(n) &\mid a_r(n) \operatorname{ppcm}(a_r(n+1) \operatorname{ppcm}(Q(n+2), \ldots, Q(n+m+1)), Q(n+2), \ldots, Q(n+m)) \\ &\mid a_r(n)a_r(n+1) \operatorname{ppcm}(Q(n+2), \ldots, Q(n+m+1)) \\ &\mid a_r(n) \cdots a_r(n+j) \operatorname{ppcm}(Q(n+j+1), \ldots, Q(n+j+m)). \end{aligned}$$

Multiple du dénominateur

Entrée : une récurrence linéaire de la forme (8).

Sortie : un multiple du dénominateur des solutions rationnelles.

1. Calculer le polynôme
$$R(h) = \operatorname{Res}_n\big(a_r(n+h), a_0(n-r)\big).$$
2. Si R n'a pas de racines dans $\mathbb{N}$, alors renvoyer le polynôme 1 ; sinon, soit $h_1 > h_2 > \cdots > h_m \geq 0$ ses racines entières positives. Initialiser Q à 1, A à $a_r(n)$, B à $a_0(n-r)$;
3. Pour $i = 1, \ldots, m$ faire
$$g(n) := \operatorname{pgcd}\big(A(n+h_i), B(n)\big) ;$$
$$Q(n) := g(n)g(n-1)\cdots g(n-h_i)Q(n) ;$$
$$A(n) := A(n)/g(n-h_i) ;$$
$$B(n) := B(n)/g(n).$$
4. Renvoyer Q.

FIGURE 2. Algorithme d'Abramov pour les multiples du dénominateur

Cependant, pour $j > H$, $\operatorname{pgcd}(Q(n), Q(n+j)) = 1$ et donc nous avons obtenu
$$Q(n) \mid a_r(n)\cdots a_r(n+H).$$
De la même manière, multiplier le polynôme
$$\operatorname{ppcm}(Q(n), \ldots, Q(n+m-1))$$
par la récurrence permet de voir que
$$Q(n) \mid a_0(n-r)\cdots a_0(n-r-H)$$
et finalement
$$Q(n) \mid \operatorname{pgcd}(a_r(n)\cdots a_r(n+H), a_0(n-r)\cdots a_0(n-r-H))$$
fournit un multiple de Q. Il est possible de calculer ce multiple efficacement en évitant de calculer des produits de degré trop élevé, et c'est ce que réussit l'algorithme d'Abramov en Figure 2.

Le polynôme dans l'étape (1) est un résultant particulier qui peut se calculer efficacement comme une somme composée. Ce calcul est présenté dans le Chapitre 3 comme application du calcul rapide de l'exponentielle des séries.

Du point de vue de la complexité, il faut noter qu'il n'est pas nécessaire de développer le polynôme $Q(n)$, et qu'une représentation sous la forme
$$Q(n) = \prod_i \prod_{j=1}^{h_i} g_i(n-j),$$
est suffisante, où g_i désigne le pgcd à l'étape i.

EXERCICE 5. Trouver les solutions rationnelles des récurrences suivantes :
$$(n+1)u_{n+1} - nu_n = 0,$$
$$(n+1)(n+3)u_{n+2} - 2(n+2)nu_{n+1} + (n-1)(n+1) = 0,$$
$$(n+3)(n+2)(n^2+6n+4)u(n+2) - (n+1)(3n^3+27n^2+64n+48)u(n+1) + 2n^2(n^2+8n+11)u(n) = 0.$$

Numérateur. Une fois ce multiple $Q(n)$ du dénominateur trouvé, le changement d'inconnue $u(n) = P(n)/Q(n)$ dans l'équation (8) peut être réalisé efficacement, même si les h_i sont très grands. C'est clair dans le cas où l'équation est homogène ($b(n) = 0$) puisqu'alors il suffit de multiplier le membre gauche par $Q(n)$ et d'observer que

$$\frac{Q(n)}{Q(n+k)} = \prod_i \prod_{j=1}^{h_i} \frac{g_i(n-j)}{g_i(n+k-j)}$$
$$= \prod_i \frac{g_i(n+k-h_i-1)g_i(n+k-h_i-2)\cdots g_i(n-h_i)}{g_i(n+k-1)g_i(n+k-2)\cdots g_i(n)}$$

est une fraction dont le numérateur et le dénominateur ont degré borné par $k\sum \deg(g_i) \leq k\max(\deg a_0, \deg a_m)$. Dans le cas inhomogène, ce changement d'inconnue mène à un membre droit de grand degré. Le même raisonnement que dans le cas différentiel s'applique : si le changement d'inconnue donne

$$A(n, P(n)) = Q(n)\sum a_i(n)\frac{P(n+r-i)}{Q(n+r-i)} = Q(n)b(n)$$

alors

$$b(n)A(n+1, P(n+1)) - b(n+1)\frac{Q(n+1)}{Q(n)}A(n, P(n)) = 0$$

est une équation homogène linéaire en les paramètres et dont les degrés des coefficients restent modérés.

Exercice 6. Résoudre en (u, λ, μ) la récurrence

$$3u(n+2) - nu(n+1) + (n-1)u(n) = \lambda n^3 + \mu n^2.$$

L'ordre 1. Lorsque la récurrence est d'ordre 1, il est en outre possible de prédire un facteur du numérateur, ce qui gagne en efficacité pratique. En effet, dans la récurrence

$$a(n)u(n+1) + b(n)u(n) = c(n),$$

si b et c ont une racine commune qui n'est ni un pôle de u ni une racine de a, alors celle-ci est nécessairement racine de $u(n+1)$. De même, si a et c ont une racine commune qui n'est ni un pôle de $u(n+1)$ ni une racine de b, alors elle est racine de u.

Ces calculs préalables permettent de réduire le degré des coefficients de l'équation dont on recherche ensuite les solutions polynomiales. Ils sont utilisés dans l'algorithme de Gosper présenté au chapitre 27.

5. Exercices

Ce chapitre montre qu'il est possible de déterminer les solutions à support fini de récurrences du type (2), puis par extension les solutions polynomiales de récurrences ou d'équations différentielles linéaires sous une *forme compacte* donnée par : la récurrence elle même à coefficients dans $\mathbb{Z}[X]$ de degré borné par d ; une borne N sur le degré ; des conditions initiales généralisées (voir p. 228) rationnelles de taille $O(dN \log N)$. Ceci n'est utile que si cette forme compacte se prête bien au calcul. Les exercices suivants explorent cette question.

Exercice 7 (Forme compacte dans la base monomiale).
1. Si P est donné en forme compacte dans la base (X^k), montrer comment obtenir efficacement son pgcd avec X^N.

2. Montrer que si (u_k) est solution d'une récurrence linéaire de la forme (2) et $\alpha \in \mathbb{Q} \setminus \{0\}$, alors la suite $(u_k \alpha^k)$ est solution d'une récurrence linéaire de la même forme, qui se calcule efficacement, dont le coefficient de tête a les mêmes racines entières positives que a_0, et dont les conditions initiales généralisées se calculent efficacement à partir de celles de (u_k).
3. Pour la suite $s_k := \sum_{i=0}^{k} u_i a^i$, montrer qu'on a encore facilement une récurrence linéaire, ainsi que les r premières conditions initiales, et que les racines du coefficient de tête sont encore inchangées.
4. Si (u_k) est donnée sous forme compacte, montrer comment obtenir les conditions initiales généralisées pour cette suite (s_k).
5. Montrer que si P est un polynôme donné sous forme compacte dans la base monomiale alors $P(\alpha)$ peut être calculé en un nombre d'opérations binaires quasi-linéaire en N, pour a comme ci-dessus.
6. Même question pour la dérivée $P^{(\ell)}(\alpha)$, pour $\ell = O(N)$.
7. Montrer comment obtenir les mêmes résultats pour $(\sum_{i=0}^{k} u_i \alpha^i)$ lorsque α est un nombre algébrique, donné par un polynôme $Q \in \mathbb{Q}[X]$ tel que $Q(\alpha) = 0$. (On admettra que les racines du terme de tête sont inchangées.)
8. En déduire que si P est donné en forme compacte dans la base (X^k), on peut en obtenir une forme compacte dans la base $((X-\alpha)^k)$ pour α donné un par un polynôme $Q \in \mathbb{Q}[X]$ tel que $Q(\alpha) = 0$.
9. Décrire un algorithme permettant de diviser efficacement le numérateur obtenu sous forme compacte dans l'algorithme de Liouville par son pgcd avec le polynôme P multiple du dénominateur. Estimer sa complexité.

EXERCICE 8 (Forme compacte en base binomiale).

1. Montrer que dans les mêmes conditions qu'à l'exercice précédent, lorsque α n'est pas un entier négatif, la suite $(\sum_{i=0}^{k} u_i \binom{\alpha}{i})$ est solution d'une récurrence linéaire à coefficients polynomiaux que l'on peut calculer efficacement, ainsi que les r premières conditions initiales, dont le terme de tête n'a pas de nouvelles racines entières positives.
2. Montrer que si P est donné sous forme compacte dans la base binomiale, on peut calculer $P(a)$ efficacement.
3. Même question pour le polynôme $\Delta^\ell P(a)$ (ici Δ est l'opérateur aux différences est le polynôme $\Delta : P(X) \mapsto P(X+1) - P(X)$, et Δ^ℓ dénote l'itérée ℓ^{e} de cet opérateur).

Notes

Les idées de base de l'algorithme de recherche de solutions rationnelles sont dues à Liouville [9] qui donne également une méthode par coefficients indéterminés pour trouver les solutions polynomiales. La présentation qui utilise les récurrences donne un algorithme de même complexité mais fait mieux ressortir la structure du calcul [3].

La recherche de solutions d'équations différentielles linéaires ne s'arrête pas aux solutions rationnelles. En utilisant la théorie de Galois différentielle, il existe une algorithmique sophistiquée de recherche de solutions *liouvilliennes* (c'est-à-dire formées par l'application répétée d'exponentielles, d'intégrales et de prise de racines de polynômes). Les calculs se ramènent à la recherche présentée ici de solutions rationnelles pour des équations (les puissances symétriques) formées à partir de l'équation de départ [10, 11, 12].

L'utilisation des algorithmes rapides pour la factorielle de matrice dans le contexte de la résolution d'équations différentielles et de récurrences, les adaptations au cas inhomogène et inhomogène paramétré, et les exercices 7 et 8 proviennent de [4, 5].

Notre présentation de la preuve de l'algorithme d'Abramov est due à Chen, Paule et Saad [6]. Abramov a proposé plusieurs raffinements de son algorithme pour obtenir des multiples du dénominateur de degré moindre [2] et aussi dans certains cas en tenant compte de tous les coefficients a_i [1]. Une manière plus directe d'aboutir à ces raffinements à été donnée par van Hoeij [7].

Bibliographie

[1] Abramov, S. A. (1989). « Rational solutions of linear differential and difference equations with polynomial coefficients ». In : *USSR Computational Mathematics and Mathematical Physics*, vol. 29, n°11, p. 1611–1620.

[2] — (1995). « Rational solutions of linear difference and q-difference equations with polynomial coefficients ». In : *ISSAC'95 : International Symposium on Symbolic and Algebraic Computation*. Éd. par A. H. M. Levelt. New York : ACM Press, p. 285–289.

[3] Abramov, Sergei A., Manuel Bronstein et Marko Petkovšek (1995). « On Polynomial Solutions of Linear Operator Equations ». In : *ISSAC'95 : International Symposium on Symbolic and Algebraic Computation*. Éd. par A. H. M. Levelt. New York : ACM Press, p. 290–296.

[4] Bostan, A., F. Chyzak, T. Cluzeau et B. Salvy (2006). « Low complexity algorithms for linear recurrences ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 31–38.

[5] Bostan, Alin, Thomas Cluzeau et Bruno Salvy (2005). « Fast Algorithms for Polynomial Solutions of Linear Differential Equations ». In : *ISSAC'05 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 45–52. doi : 10.1145/1073884.1073893.

[6] Chen, William Y.C., Peter Paule et Husam L. Saad (2008). « Converging to Gosper's algorithm ». In : *Advances in Applied Mathematics*, vol. 41, n°3, p. 351–364. doi : 10.1016/j.aam.2007.11.004.

[7] Hoeij, Mark van (1998). « Rational solutions of linear difference equations ». In : *ISSAC'98 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 120–123.

[8] Ince, E. L. (1956). *Ordinary differential equations*. Reprint of the 1926 edition. Dover Publications, viii+558 pages.

[9] Liouville, Joseph (1833). « Second mémoire sur la détermination des intégrales dont la valeur est algébrique ». In : *Journal de l'École polytechnique*, vol. 14, p. 149–193.

[10] Marotte, F. M. (1898). « Les équations différentielles linéaires et la théorie des groupes ». In : *Annales de la Faculté des Sciences de Toulouse*, vol. 12, n°4, p. 1–92. url : http://www.numdam.org/item?id=AFST_1898_1_12_4_H1_0.

[11] Singer, Michael F. (1981). « Liouvillian solutions of n-th order homogeneous linear differential equations ». In : *American Journal of Mathematics*, vol. 103, n°4, p. 661–682.

[12] Singer, Michael F. et Felix Ulmer (1997). « Linear differential equations and products of linear forms ». In : *Journal of Pure and Applied Algebra*, vol. 117/118, p. 549–563.

Quatrième partie

Factorisation des polynômes

CHAPITRE 17

Factorisations sans carré et séparable

Résumé

Les algorithmes de factorisation des polynômes sont constitués de plusieurs étapes de natures bien différentes. Pour des polynômes à une variable, la première d'entre elles consiste à décomposer le polynôme en facteurs dont les racines sont de même multiplicité. On parle alors de factorisation séparable. Lorsque les coefficients appartiennent à un corps de caractéristique zéro ou plus grande que le degré, alors la factorisation séparable coïncide avec la factorisation sans carré. Le présent chapitre présente des algorithmes pour ces tâches.

1. Introduction

Soit $\mathbb{K}$ un corps, et soit F un polynôme unitaire de $\mathbb{K}[X]$. La *factorisation en irréductibles* de F, notée $\mathrm{irr}(F) = \{(F_1, m_1), \ldots, (F_r, m_r)\}$ avec les F_i irréductibles unitaires, correspond à l'identité

$$F = F_1^{m_1} \cdots F_r^{m_r}.$$

L'entier m_i est appelé la *multiplicité* du facteur F_i.

On dit que F est *sans carré* s'il n'existe pas de polynôme P de $\mathbb{K}[X] \setminus \mathbb{K}$ tel que P^2 divise F. Autrement dit, toutes les multiplicités des facteurs irréductibles de F sont égales à 1. Si l'on note $\{\mu_1, \ldots, \mu_s\}$ l'ensemble des multiplicités des facteurs irréductibles (avec $s \leq r$), et Q_i le produit des F_j de multiplicité $\mu_i = m_j$, alors la *factorisation sans carré* de F, notée $\mathrm{sqr}(F)$, est l'ensemble des paires (Q, m) où Q est le produit des facteurs irréductibles de F de multiplicité exactement $m \geq 1$. La *partie sans carré* de F est alors définie comme le produit $\prod_{(Q,m)\in \mathrm{sqr}(F)} Q$; il s'agit manifestement d'un polynôme sans carré.

Par exemple, si

$$F = (X+1)(X+2)(X+3)^2(X+4)^2(X+5)^3(X+6)^3,$$

sa partie sans carré est le produit de ses facteurs irréductibles, c'est-à-dire

$$(X+1)(X+2)(X+3)(X+4)(X+5)(X+6).$$

La factorisation sans carré de F est ici :

$$\mathrm{sqr}(F) = \{((X+1)(X+2), 1), ((X+3)(X+4), 2), ((X+5)(X+6), 3)\}.$$

On voit bien que le calcul de factorisation sans carré est l'amorce de la factorisation irréductible ; en outre, c'est un calcul sensiblement plus facile. En effet, nous verrons plus loin un algorithme de calcul de la partie sans carré, de complexité quasi-linéaire en le degré, qui repose uniquement sur des calculs de pgcd bien choisis lorsque $\mathbb{K}$ est de caractéristique zéro ou suffisamment grande.

D'une façon générale, la factorisation irréductible d'un polynôme est l'une des premières questions difficiles du calcul formel. Nous présenterons ces difficultés et les solutions connues pour traiter les cas usuels, rencontrés en pratiques, dans le chapitre 20. Dans un premier temps nous exposerons les idées classiques en caractéristique zéro pour calculer la factorisation sans carré. En caractéristique positive

la situation est nettement plus complexe et il existe des corps effectifs où l'extraction des racines p^{e} n'est pas calculable. La factorisation sans carré n'est alors pas calculable en général et nous verrons alors l'intérêt d'introduire la factorisation séparable.

Tout au long de ce chapitre, $\mathbb{K}$ représente un corps effectif, p sa caractéristique, et F un polynôme unitaire de $\mathbb{K}[X]$ de degré $n \geq 1$.

2. Factorisation sans carré

Dans cette section nous considérons le cas de la caractéristique zéro, et nous nous intéressons d'abord au calcul de la partie sans carré de F. Nous verrons qu'il n'est pas difficile d'en déduire un algorithme pour la factorisation sans carré : on divise F par sa partie sans carré, on calcule la partie sans carré du résultat, etc. Ce faisant, on n'obtient pas une borne de complexité quasi-linéaire. L'algorithme rapide, dû à Yun, est présenté en fin de section.

2.1. Partie sans carré. Commençons par le lemme suivant, qui nous sera plusieurs fois dans la suite :

LEMME 1. *Supposons $\mathbb{K}$ de caractéristique $p = 0$. Soient $G_1, \ldots, G_s$ des polynômes sans carré de $\mathbb{K}[X] \setminus \mathbb{K}$, soit $G = G_1 \cdots G_s$, et soit $(c_1, \ldots, c_s) \in \mathbb{K}^s$. Si $G_1, \ldots, G_s$ sont premiers entre eux deux à deux, alors nous avons l'égalité suivante :*

$$\operatorname{pgcd}\left(G, \sum_{i=1}^{s} c_i \frac{G}{G_i} G_i'\right) = \prod_{c_i=0} G_i.$$

DÉMONSTRATION. La preuve découle du calcul suivant :

$$\operatorname{pgcd}\left(G, \sum_{i=1}^{s} c_i \frac{G}{G_i} G_i'\right) = \prod_{j=1}^{s} \operatorname{pgcd}\left(G_j, \sum_{i=1}^{s} c_i \frac{G}{G_i} G_i'\right) = \prod_{j=1}^{s} \operatorname{pgcd}\left(G_j, c_j \frac{G}{G_j} G_j'\right).$$

Ce dernier pgcd est égal à G_j si $c_j = 0$ et 1 si $c_j \neq 0$ puisque G_j' est premier avec G_j. □

PROPOSITION 1. *Si $\mathbb{K}$ est de caractéristique zéro, alors $F \in \mathbb{K}[X]$ est sans carré si, et seulement si,* $\operatorname{pgcd}(F, F') = 1$. *De plus la partie sans carré de $F \in \mathbb{K}[X]$ est donnée par la formule $F/\operatorname{pgcd}(F, F')$ et peut être calculée au moyen de $O(\mathsf{M}(n) \log n)$ opérations dans $\mathbb{K}$.*

DÉMONSTRATION. Écrivons la factorisation en irréductibles unitaires de F :

$$F = F_1^{m_1} \cdots F_r^{m_r},$$

où tous les m_i sont des entiers ≥ 1. En dérivant, on obtient

$$F' = \sum_{i=1}^{r} m_i F_i' \frac{F}{F_i}.$$

En posant $E = \prod_{i=1}^{r} F_i$ la partie sans carré de F, on en déduit que

$$\begin{aligned} \operatorname{pgcd}(F, F') &= \operatorname{pgcd}\left(\prod_{i=1}^{r} F_i^{m_i}, \sum_{i=1}^{r} m_i \frac{F}{F_i} F_i'\right) \\ &= \operatorname{pgcd}\left(E, \sum_{i=1}^{r} m \frac{E}{F_i} F_i'\right) \prod_{i=1}^{r} F_i^{m_i - 1} = \prod_{i=1}^{r} F_i^{m_i - 1}, \end{aligned}$$

où la dernière égalité provient du lemme 1. La partie sans carré E de F s'obtient alors comme $F/\operatorname{pgcd}(F, F')$. Le coût du calcul est essentiellement une conséquence de la proposition 7 du chapitre 6 qui donne le coût du pgcd. □

Factorisation sans carré de F

Entrée : F dans $\mathbb{K}[X]$, avec $\mathbb{K}$ de caractéristique zéro.

Sortie : $\mathrm{sqr}(F)$.

1. Initialiser L avec la liste vide et i à l'entier 1.
2. Calculer $C = \mathrm{pgcd}(F, F')$ et $G = F/C$.
3. Tant que $\deg(G) \geq 1$ faire :
   a. calculer $P = \mathrm{pgcd}(C, G)$ et $C = C/P$,
   b. si $\deg(G) > \deg(P)$ alors adjoindre $(G/P, i)$ à L,
   c. $G = P$, et $i = i + 1$.
4. Retourner L.

FIGURE 1. Algorithme de Musser, en caractéristique zéro.

2.2. Algorithme de Musser. Le premier algorithme de factorisation sans carré, que nous présentons dans la Figure 1, commence par calculer la partie sans carré.

PROPOSITION 2. *L'algorithme de factorisation sans carré de la Figure 1 est correct et son coût arithmétique est en* $O(n\mathsf{M}(n)\log n)$.

DÉMONSTRATION. Notons $(G_1, m_1), \ldots, (G_s, m_s)$ la factorisation sans carré de F normalisée de sorte que

$$F' = \sum_{i=1}^{s} m_i G_i' G_i^{m_i - 1} \frac{F}{G_i^{m_i}}.$$

Alors, à l'étape 2, nous avons $C = \prod_{i=1}^{s} G_i^{m_i-1}$, et $G = \prod_{i=1}^{s} G_i$. Dans la première étape de la boucle nous obtenons $P = \prod_{i=1, m_i \geq 2}^{s} G_i$, $C = \prod_{i=1, m_i \geq 2}^{s} G_i^{m_i-2}$, et G/P correspond aux G_i pour lesquels $m_i = 1$. À la deuxième étape de la boucle nous avons $P = \prod_{i=1, m_i \geq 3}^{s} G_i$, $C = \prod_{i=1, m_i \geq 3}^{s} G_i^{m_i-3}$, et G/P correspond aux G_i pour lesquels $m_i = 2$. Par récurrence nous déduisons que l'algorithme est correct.

Le nombre d'étapes de l'algorithme est borné par n, le coût de chaque étape est dominé par celui du pgcd qui est, rappelons le, en $O(\mathsf{M}(n)\log n)$. □

2.3. Algorithme de Yun. L'algorithme suivant de factorisation sans carré, présenté dans la Figure 2, permet d'atteindre un coût arithmétique essentiellement linéaire.

PROPOSITION 3. *L'algorithme de factorisation sans carré de la Figure 2 est correct et son coût arithmétique est en* $O(\mathsf{M}(n)\log n)$ *opérations dans* $\mathbb{K}$.

DÉMONSTRATION. Notons V_l et W_l les valeurs respectives de V et W au départ de l'étape l de la boucle « tant que », et notons H_l la valeur de H calculée durant l'étape l. Notons ℓ la valeur de l à la sortie de la boucle. Définissons aussi V_ℓ et W_ℓ comme les valeurs respectives de V et W à la sortie de la boucle. Nous allons prouver la propriété suivante par récurrence sur l de 1 à ℓ :

$$V_l = \prod_{\substack{(G,m)\in \mathrm{sqr}(F) \\ m \geq l}} G, \qquad W_l = \sum_{\substack{(G,m)\in \mathrm{sqr}(F) \\ m \geq l}} (m - l + 1)\frac{V_l}{G} G'.$$

À l'étape 2 nous avons :

$$U = \mathrm{pgcd}(F, F'), \qquad V = \frac{F}{\mathrm{pgcd}(F, F')}, \qquad W = \frac{F'}{\mathrm{pgcd}(F, F')}.$$

Factorisation sans carré de F

Entrée : F dans $\mathbb{K}[X]$, avec $\mathbb{K}$ de caractéristique zéro.

Sortie : $\mathrm{sqr}(F)$.

1. Poser $l = 1$ et initialiser L avec la liste vide.
2. Calculer $U = \mathrm{pgcd}(F, F')$, $V = F/U$, et $W = F'/U$.
3. Tant que $\deg(V) \geqslant 1$ faire :
   a. Calculer $H = \mathrm{pgcd}(V, W - V')$, $W = (W - V')/H$, et $V = V/H$.
   b. Si $\deg(H) \geqslant 1$ alors adjoindre (H, l) à L.
   c. Incrémenter l de 1.
4. Retourner L.

FIGURE 2. Algorithme de Yun, en caractéristique zéro.

Comme nous avons vu précedemment que $\mathrm{pgcd}(F, F') = \prod_{(G,m)\in\mathrm{sqr}(F)} G^{m-1}$, l'hypothèse de récurrence est satisfaite pour $l = 1$. Supposons l'hypothèse de récurrence satisfaite pour une certaine valeur de l dans $\{1, \dots, \ell - 1\}$. En utilisant le lemme 1 nous déduisons :

$$H_l = \mathrm{pgcd}(V_l, W_l - V_l') = \mathrm{pgcd}\left(\prod_{\substack{(G,m)\in\mathrm{sqr}(F)\\ m\geq l}} G, \sum_{\substack{(G,m)\in\mathrm{sqr}(F)\\ m\geq l}} (m-l)\frac{V_l}{G}G'\right)$$
$$= \prod_{\substack{(G,m)\in\mathrm{sqr}(F)\\ m=l}} G,$$

ce qui conduit aux formules attendues pour V_{l+1} et W_{l+1}. Puisque ℓ est le premier entier tel que $\deg(V_\ell) = 0$, nous obtenons que $\ell-1$ est la plus grande des multiplicités des facteurs de F. Finalement L contient tous les facteurs sans carré de F à la fin de la boucle.

L'étape 2 utilise $O(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$. L'étape l de la boucle coûte au plus $O(\mathsf{M}(\deg(V_l))\log(\deg(V_l)))$ puisque $\deg(W_l) \leq \deg(V_l) - 1$. Par la super-additivité de M, l'étape 3 totalise au plus $O\left(\mathsf{M}\left(\sum_{l=1}^{\ell-1}\deg(V_l)\right)\log n\right)$ opérations dans $\mathbb{K}$, ce qui conduit à la borne de complexité attendue puisque $\prod_{l=1}^{\ell-1} V_l = F$. □

3. Séparabilité et déflation

Si $\mathbb{K}$ est le corps des fractions $\mathbb{F}_2(T)$, et si $F(X) = T^2 + X^2 = (T + X)^2$, alors $F'(X) = 0$, et ni l'algorithme de Musser ni celui de Yun ne calcule la factorisation sans carré de F. Nous allons voir que le « bon concept » en caractéristique positive n'est pas celui d'être sans carré, mais plutôt d'être séparable :

DEFINITION 1. Un polynôme $F \in \mathbb{K}[X]$ est dit *séparable* s'il n'a aucune racine multiple dans une clôture algébrique $\bar{\mathbb{K}}$ de $\mathbb{K}$.

PROPOSITION 1. *Un polynôme F qui n'est pas constant est séparable si, et seulement si, son discriminant est non nul.*

DÉMONSTRATION. Notons $\alpha_1, \dots, \alpha_r$ les racines distinctes de F dans $\bar{\mathbb{K}}$ de multiplicités respectives $m_1, \dots, m_r$. On peut supposer F unitaire. Si toutes les multiplicités valent 1, alors $F'(\alpha_i) = \prod_{j\neq i}(\alpha_i - \alpha_j) \neq 0$. Le discriminant étant égal

au produit des $F'(\alpha_i)$, il est non nul. Si l'une des multiplicités, disons $m_1 \geq 2$ alors $F'(\alpha_1) = 0$ est le discriminant s'annule. □

Par exemple, si $\mathbb{K}$ est le corps des fractions $\mathbb{F}_2(T)$, alors $F(X) = T + X^2$ est sans carré mais n'est pas séparable pour autant puisque $F'(X)$ est identiquement nul. Néanmoins si $p = 0$, alors la condition de séparabilité est équivalente à être sans carré.

Dans la suite nous noterons $\mathcal{B} = \{1, p, p^2, p^3, \ldots\}$ l'ensemble des puissances de p si $p > 0$, et $\mathcal{B} = \{1\}$ si $p = 0$. Les problèmes liés à la séparabilité concernent les cas de caractéristique positive. Nous aurons besoin des définitions et propositions suivantes.

Definition 2. Le *degré d'inséparabilité* de F, noté $\deg^i(F)$, est le plus grand élément q de $\mathcal{B}$ tel que $F \in \mathbb{K}[X^q] \setminus \mathbb{K}[X^{pq}]$.

Proposition 2. *Soient F et G deux polynômes de $\mathbb{K}[X] \setminus \mathbb{K}$.*

1. *Le degré d'inséparabilité de F est la plus grande puissance de p qui divise la multiplicité de chaque racine de F.*
2. *$\deg^i(FG) \geq \min(\deg^i(F), \deg^i(G))$, l'inégalité devient une égalité dès lors que F et G sont premiers entre eux.*

Démonstration. Si $q \in \mathcal{B}$ divise la multiplicité de chaque racine de F, alors F appartient bien à $\mathbb{K}[X^q]$. Réciproquement, si F peut s'écrire $G(X^q)$ avec $q \in \mathcal{B}$, alors la multiplicité d'une racine de F est bien un multiple de q. La partie (2) est une conséquence directe de la partie (1). □

Definition 3. Le degré de séparabilité de $F \in \mathbb{K}[X]$, noté $\deg^s(F)$, est défini comme suit :

$$\deg^s(F) = \sum_{(G,m)\in \mathrm{irr}(F)} \deg(G)/\deg^i(G) = \sum_{(G,m)\in \mathrm{irr}(F)} \deg^s(G).$$

Proposition 3. *Soient F et G des polynômes de $\mathbb{K}[X] \setminus \mathbb{K}$.*

1. *$\deg^s(F)$ est égal au nombre de racines de F dans $\bar{\mathbb{K}}$ sans compter les multiplicités.*
2. *F est séparable si, et seulement si, $\deg^s(F) = \deg(F)$.*
3. *Pour tout $q \in \mathcal{B}$, $\deg^s(F(X^q)) = \deg^s(F)$.*
4. *$\deg^s(FG) \leq \deg^s(F) + \deg^s(G)$, avec égalité si, et seulement si, F et G sont premiers entre eux.*

Démonstration. Ces propriétés découlent presque immédiatement des définitions et nous laissons au lecteur le soin de les vérifier à titre d'exercice. □

Lorsque qu'un polynôme F appartient à $\mathbb{K}[X^p]$, il est utile de le voir comme un polynôme en la variable X^p. Ceci motive la définition suivante :

Definition 4. Si $F \in \mathbb{K}[X]\setminus\mathbb{K}$, le *polynôme déflaté* de F est l'unique polynôme $\tilde{F}$ défini par

$$\tilde{F}(X^{\deg^i(F)}) = F(X).$$

Les multiplicités des racines de $\tilde{F}$ sont celles de F divisées par $\deg^i(F)$, d'où la terminologie « déflation ». Puisque $\tilde{F}$ appartient à $\mathbb{K}[X] \setminus \mathbb{K}[X^p]$, la fonction suivante Φ de déflation est bien définie :

$$\Phi : \mathbb{K}[X] \setminus \mathbb{K} \to (\mathbb{K}[X] \setminus \mathbb{K}[X^p]) \times \mathcal{B}$$
$$F \mapsto (\tilde{F}, \deg^i(F)).$$

Notons $\mathcal{Q}$ l'ensemble des puissances q^{es} des polynômes irréductibles de $\mathbb{K}[X] \setminus \mathbb{K}$ pour $q \in \mathcal{B}$, et $\mathcal{S}$ l'ensemble des polynômes séparables irréductibles de $\mathbb{K}[X] \setminus \mathbb{K}$.

PROPOSITION 4. *Φ est une bijection qui envoie $\mathcal{Q}$ surjectivement sur $\mathcal{S} \times \mathcal{B}$.*

DÉMONSTRATION. Le fait que Φ est une bijection découle clairement des définitions. Si $F \in \mathbb{K}[X]$ est irréductible, alors $\tilde{F}$ est nécessairement irréductible et donc séparable car $\tilde{F} \notin \mathbb{K}[X^p]$. Puisque le polynôme déflaté de F^q coincide avec celui de $\tilde{F}^q$ pour tout $q \in \mathcal{B}$, le lemme 2 ci-dessous implique que $\Phi(\mathcal{Q})$ appartient bien à $\mathcal{S} \times \mathcal{B}$.

Réciproquement, si $(G, q) \in \mathcal{S} \times \mathcal{B}$. Nous devons montrer que $F(X) = G(X^q)$ appartient à $\mathcal{Q}$, et que $\deg^i(F) = q$. Cette dernière égalité est claire puisque G est séparable. Soit $h \leq q$ le plus grand élément de $\mathcal{B}$ tel que $G(X^h)$ est une puissance h^{e}, et soit $\tilde{G}$ la racine h^{e} correspondante, de sorte que $\tilde{G}^h(X) = G(X^h)$. Par le lemme 2 ci-dessous, $\tilde{G}$ et donc $\tilde{G}(X^{q/h})$ sont irréductibles et séparables, ce qui prouve que $F(X) = \tilde{G}(X^{q/h})^h \in \mathcal{Q}$. □

LEMME 2. *Soient F et G deux polynômes de $\mathbb{K}[X]\setminus\mathbb{K}$ tels que $G(X^p) = F(X)^p$.*

1. *G est séparable si, et seulement si, F est séparable.*
2. *Si G est irréductible alors F est irréductible. La réciproque est vraie si F ou G est séparable.*

DÉMONSTRATION. La preuve de la partie (a) est immédiate. Les multiplicité de $G(X^p)$ (resp. de $F(X)^p$) sont toutes p si, et seulement si, G (resp. F) est séparable. La partie (b) est donc une conséquence de la proposition 3. Concernant la partie (c), si G est irréductible, alors pour n'importe quel facteur irréductible A de F, le polynôme A^p divise F^p, et donc B divise G, si nous définissons B par $B(X^p) = A(X)^p$. Par conséquent B et donc A sont des unités de $\mathbb{K}$. Réciproquement, supposons F irréductible, et soit A un facteur irréductible de G. Puisque $A(X^p)$ divise $F(X)^p$, le polynôme $A(X^p)$ peut s'écrire $F(X)^\alpha$ pour un certain $\alpha \in \{0, \ldots, p-1\}$. En dérivant, nous obtenons $\alpha F' F^{\alpha-1} = 0$, d'où $\alpha = 0$ dès que F est séparable. □

4. Factorisation séparable

Nous pouvons maintenant définir un nouveau type de factorisation correspondant à écrire F essentiellement comme un produit de facteurs séparables :

DEFINITION 5. La *factorisation séparable* d'un polynôme $F \in \mathbb{K}[X] \setminus \mathbb{K}$, notée $\mathrm{sep}(F)$, est l'ensemble des triplets $(G_1, q_1, m_1), \ldots, (G_s, q_s, m_s)$ de $\mathbb{K}[X] \times \mathcal{B} \times \mathbb{N}^*$ vérifiant les propriétés suivantes :

- (S_1) $F(X) = \prod_{i=1}^{s} G_i^{m_i}(X^{q_i})$,
- (S_2) les $G_i(X^{q_i})$ sont premiers entre eux deux à deux,
- (S_3) les m_i ne sont pas divisibles par p,
- (S_4) les G_i sont séparables de degré au moins 1,
- (S_5) les couples (q_i, m_i) sont deux à deux distincts.

Notons que dans le cas où $p = 0$ la factorisation séparable n'est rien d'autre que la factorisation sans carré.

THÉORÈME 1. *Tout polynôme F de $\mathbb{K}[X]$ admet une unique factorisation séparable (à permutation et unités près dans $\mathbb{K}$).*

DÉMONSTRATION. Soient $(F_1, e_1), \ldots, (F_r, e_r)$ la factorisation irréductible de F dans $\mathbb{K}[X]$. Pour chaque $i \in \{1, \ldots, r\}$, soit a_i la plus grande puissance de p qui divise e_i, et soit $m_i = e_i/a_i$. En définissant $(G_i, q_i) = \Phi(F_i^{a_i})$, nous obtenons des triplets

$$(G_1, q_1, m_1), \ldots, (G_s, q_s, m_s)$$

qui satisfont (S_1), (S_2) et (S_3). La propriété (S_4) est aussi satisfaite par la proposition 4.

Pour obtenir (S_5) il suffit de regrouper les triplets qui partagent les mêmes deuxièmes et troisièmes coordonnées comme suit : si $(A_1, q, m), \ldots, (A_t, q, m)$ sont de tels triplets alors nous les fusionnons en le triplet $(A_1 \cdots A_r, q, m)$. Les propriétés de (S_1) à (S_4) sont préservées, ce qui prouve l'existence de la factorisation séparable.

Supposons maintenant que nous disposons de triplets

$$(G_1, q_1, m_1), \ldots, (G_s, q_s, m_s)$$

qui satisfont les propriétés de (S_1) à (S_5). Pour chaque $i \in \{1, \ldots, s\}$, n'importe quelle racine de $G_i(X^{q_i})$ dans $\overline{\mathbb{K}}$ admet q_i comme multiplicité, et donc $q_i m_i$ comme multiplicité dans F. La propriété (S_5) implique alors que les racines de $G_i(X^{q_i})$ sont exactement celles de F de multiplicité $q_i m_i$, ce qui conduit à l'unicité de la factorisation séparable. □

Exemple 1. Lorsque $\mathbb{K} = \mathbb{F}_3$ et $F = X^2(X+1)^3(X+2)^4 = X^9 + 2X^8 + 2X^3 + X^2$, nous avons $\mathrm{sep}(F) = \{(X, 1, 2), (X+1, 3, 1), (X+2, 1, 4)\}$.

Exemple 2. Lorsque $\mathbb{K} = \mathbb{F}_3[T]$ et $F = (X+2T)^7(X^3+2T)^3(X^6+T)$, nous avons $\mathrm{sep}(F) = \{(X+2T, 1, 7), (X+2T^3, 9, 1), (X^2+T, 3, 1)\}$.

4.1. Algorithme de Gianni et Trager. Les algorithmes de factorisation séparable connus utilisent ceux de Musser ou de Yun vus précédemment. Dans le cas de l'algorithme de Musser, le point de départ est la propriété suivante :

Proposition 5. *Si l'algorithme de la Figure 1 est exécuté en caractéristique positive, et si $L = \{(G_1, m_1), \ldots, (G_s, m_s)\}$ est la liste retournée, alors*

$$(G_1, 1, m_1), \ldots, (G_s, 1, m_s)$$

sont exactement les facteurs séparables de F de la forme $(G, 1, m)$.

Démonstration. Notons F_0 le facteur de F composé des racines de multiplicité qui ne sont pas multiples de p et posons $F_1 = F/F_0$. Notons justement $(G_1, 1, m_1), \ldots, (G_s, 1, m_s)$ la factorisation séparable de F_0. Nous allons montrer que l'algorithme retourne bien $\{(G_1, m_1), \ldots, (G_s, m_s)\}$.

Comme $F_1' = 0$ nous obtenons

$$F' = F_1 \sum_{i=1}^{s} m_i G_i' G_i^{m_i - 1} \frac{F_0}{G_i^{m_i}},$$

alors, à l'étape 2 de l'algorithme, nous avons $C = F_1 \prod_{i=1}^{s} G_i^{m_i - 1}$, et $G = \prod_{i=1}^{s} G_i$. Dans la première étape de la boucle nous obtenons $P = \prod_{i=1, m_i \geq 2}^{s} G_i$, $C = F_1 \prod_{i=1, m_i \geq 2}^{s} G_i^{m_i - 2}$, et G/P correspond aux G_i pour lesquels $m_i = 1$. À la deuxième étape de la boucle nous avons $P = \prod_{i=1, m_i \geq 3}^{s} G_i$, $C = F_1 \prod_{i=1, m_i \geq 3}^{s} G_i^{m_i - 3}$, et G/P correspond aux G_i pour lesquels $m_i = 2$. La preuve s'achève ainsi par récurrence comme pour l'algorithme de Musser. □

En conservant les notations de la preuve, notons F_0 le facteur de F composé des racines de multiplicité qui ne sont pas multiples de p et posons $F_1 = F/F_0$. En introduisant $\tilde{F}_1(X^p) = F_1(X)$, le degré de $\tilde{F}_1$ est au plus celui de F divisé par p, et nous pouvons donc calculer récursivement la factorisation séparable de $\tilde{F}_1$. Cette approche conduit à un algorithme dû à Gianni et Trager, présenté dans la Figure 3.

Proposition 6. *L'algorithme de la Figure 3 est correct. Si $\mathbb{K}$ est un corps, alors son coût est borné par $O(n\mathsf{M}(n)\log n)$.*

Factorisation séparable de F

Entrée : un polynôme $F \in \mathbb{K}[x]$ de degré $n \geq 1$.

Sortie : la factorisation séparable de F.

a. Appeler l'algorithme de la Figure 1 pour obtenir l'ensemble $L = \{(G_1, 1, m_1), \ldots, (G_r, 1, m_r)\}$ des facteurs séparables de F contenant toutes les racines de F de multiplicité non divisible par p.

b. Calculer $C = F/(G_1^{m_1} \cdots G_s^{m_s})$ et construire $\tilde{C}$ tel que $\tilde{C}(X^p) = C(X)$.

c. Appeler récursivement le présent algorithme pour obtenir la factorisation séparable $\tilde{L} = \{(G_{r+1}, q_{r+1}, m_{r+1}), \ldots, (G_s, q_s, m_s)\}$ de $\tilde{C}$.

d. Retourner $\{(G_1, 1, m_1), \ldots, (G_r, 1, m_r)\}$
$\cup \{(G_{r+1}, pq_{r+1}, m_{r+1}), \ldots, (G_s, pq_s, m_s)\}$.

FIGURE 3. Algorithme de Gianni et Trager.

DÉMONSTRATION. Par construction, les racines de C ont une multiplicité divisible par p donc $\tilde{C}$ est bien défini à l'étape 2. Comme $\tilde{L}$ est la factorisation séparable de $\tilde{C}$, alors $\{(G_{r+1}, pq_{r+1}, m_{r+1}), \ldots, (G_s, pq_s, m_s)\}$ est la factorisation séparable de C. Comme le degré de $\tilde{C}$ est borné par n/p, et que le calcul de C demande $O(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$ par la proposition 2 du Chapitre 5, le coût total est borné par

$$O\left(\sum_{i\geq 1} \frac{n}{p^i} \mathsf{M}\left(\frac{n}{p^i}\right) \log \frac{n}{p^i}\right) \subseteq O(n\mathsf{M}(n)\log n). \qquad \square$$

4.2. ⋆ Algorithme de Lecerf ⋆. L'extension de l'algorithme de Yun pour effectuer la factorisation séparable conduit à un coût arithmétique quasi-linéaire. Cette fin de section est assez technique et peut être omise en première lecture. Le point de départ est le lemme suivant :

LEMME 3. *Supposons $p > 0$, et soit F un polynôme unitaire de $\mathbb{K}[X]$. Il existe des uniques polynômes unitaires S_0 et S_1 de $\mathbb{K}[X]$ avec les propriétés suivantes :*

— *les facteurs irréductibles de S_0 sont séparables de multiplicité au plus $p-1$,*

— *$F(X) = S_0(X)S_1(X^p)$.*

DÉMONSTRATION. Le polynôme S_0 est défini par sa factorisation en irréductibles :

$$\mathrm{irr}(S_0) = \{(G, m \bmod p) \mid (G, m) \in \mathrm{irr}(F), G \text{ est séparable et } m \bmod p \neq 0\}.$$

Un facteur irréductible G de F/S_0 est soit inséparable soit de multiplicité m dans F divisible par p. Par conséquent $F/S_0 \in \mathbb{K}[X^p]$ et nous définissons $S_1(X^p) = F(X)/S_0(X)$. $\square$

EXEMPLE 3. Avec l'exemple 1 nous avons $S_0 = (X+2)X^2$ et $S_1 = (X+1)(X+2)$.

EXEMPLE 4. Avec l'exemple 2 nous avons $S_0 = X + 2T$ et $S_1 = (X^2+T)(X+2T^3)^2(X+2T)^3$.

Pour calculer la factorisation séparable de F nous commençons par celle de S_0.

LEMME 4. *Si l'algorithme de la Figure 2 est utilisé en caractéristique positive p, alors il retourne la factorisation sans carré de S_0 tel que défini dans le lemme 3. De plus sa complexité arithmétique reste en $O(\mathsf{M}(n)\log n)$.*

Fusion de $\mathrm{sqr}(S_0)$ et $\mathrm{sep}(S_1)$

Entrée : $F \in \mathbb{K}[X]$ de degré n, $\mathrm{sqr}(S_0)$ et $\mathrm{sep}(S_1)$.

Sortie : $\mathrm{sep}(F)$.

1. Si $S_1 \in \mathbb{K}$ alors retourner $\{(G_1, 1, m_1), \ldots, (G_r, 1, m_r)\}$.
   Sinon initialiser L avec la liste vide.
2. Pour tout $i \in \{1, \ldots, r\}$ faire
   a. Pour tout $j \in \{1, \ldots, s\}$ calculer $R_{i,j}(X) = \tilde{G}_i(X) \bmod H_j(X^{q_j})$, où $\tilde{G}_i$ est le polynôme défini par $\tilde{G}(X^p) = G_i(X)^p$.
   b. Pour tout $j \in \{1, \ldots, s\}$ calculer $S_{i,j}(X) = \gcd(R_{i,j}(X), H_j(X^{q_j}))$.
   c. Pour tout $j \in \{1, \ldots, s\}$ calculer $T_{i,j}(X) = G_i(X) \bmod S_{i,j}(X^p)$.
   d. Pour tout $j \in \{1, \ldots, s\}$ calculer $U_{i,j}(X) = \gcd(T_{i,j}(X), S_{i,j}(X^p))$ et adjoindre $(U_{i,j}, 1, m_i + pq_jn_j)$ à L si $\deg(U_{i,j}) \geq 1$.
   e. Calculer $V_i(X) = \prod_{j=1}^{s} U_{i,j}(X)$.
   f. Calculer $U_{i,0}(X) = G_i(X)/V_i(X)$ et adjoindre $(U_{i,0}, 1, m_i)$ à L si $\deg(U_{i,0}) \geq 1$.
3. Pour tout $j \in \{1, \ldots, s\}$ faire
   a. Calculer $W_j(X^{q_j}) = \prod_{i=1}^{r} S_{i,j}(X)^{q_j}$.
   b. Calculer $U_{0,j}(X) = H_j(X)/W_j(X)$ et adjoindre $(U_{0,j}, pq_j, n_j)$ à L si $\deg(U_{0,j}) \geq 1$.
4. Retourner L.

FIGURE 4. Algorithme rapide de fusion des factorisations séparables.

DÉMONSTRATION. À l'étape 2 de l'algorithme nous avons :

$$U = \gcd(S_0, S_0')S_1(X^p), \qquad V = \frac{S_0}{\gcd(S_0, S_0')}, \qquad W = \frac{S_0'}{\gcd(S_0, S_0')}.$$

Nous avons les mêmes valeurs de V et W en appelant l'algorithme directement avec S_0. Ensuite, nous laissons le soin au lecteur de vérifier à titre d'exercice que l'algorithme Yun retourne bien la factorisation sans carré de S_0, qui coïncide par ailleurs avec la factorisation séparable. Finalement l'analyse de complexité reste inchangée. □

Pour obtenir la factorisation séparable de F, nous commençons par calculer $\mathrm{sqr}(S_0)$ et S_1 par l'algorithme précédent. Ensuite nous calculons récursivement $\mathrm{sep}(S_1)$, pour finalement fusionner ces factorisations et obtenir $\mathrm{sep}(F)$. Expliquons maintenant comment cette fusion peut être réalisée rapidement. Si

$$\mathrm{sqr}(S_0) = \{(G_1, m_1), \ldots, (G_r, m_r)\},$$
$$\mathrm{sep}(S_1) = \{(H_1, q_1, n_1), \ldots, (H_s, q_s, n_s)\},$$

alors nous commençons par calculer tous les $U_{i,j} = \gcd(G_i, H_j(X^{pq_j}))$ pour $i \in \{1, \ldots, r\}$ et $j \in \{1, \ldots, s\}$. Nous verrons que $(U_{i,j}, 1, m_i + pq_jn_j)$ est un élément de $\mathrm{sep}(F)$ dès que $\deg(U_{i,j}) \geq 1$.

LEMME 5. *L'algorithme de la Figure 4 est correct et utilise $O(\mathsf{M}(n)\log n)$ opérations arithmétiques dans $\mathbb{K}$.*

DÉMONSTRATION. Nous devons vérifier que la liste L retournée par l'algorithme satisfait les propriétés (S_1) à (S_5) de la définition 5. Pour tout $i \in \{1, \ldots, r\}$

et tout $j \in \{1, \dots, s\}$ nous avons

$$R_{i,j}(X^p) = G_i(X)^p \bmod H_j(X^{pq_j}), \quad S_{i,j}(X^p) = \gcd(G_i(X)^p, H_j(X^{pq_j})),$$

et par conséquent

$$\begin{aligned} U_{i,j}(X) &= \gcd(G_i(X), S_{i,j}(X^p)) = \gcd(G_i(X), \gcd(G_i(X)^p, H_j(X^{pq_j}))) \\ &= \gcd(G_i(X), H_j(X^{pq_j})). \end{aligned}$$

À la fin de l'étape 2, pour tout $i \in \{1, \dots, r\}$, il est clair que $G_i = \prod_{j=0}^{s} U_{i,j}$, et que les $U_{i,j}$ sont séparables et premiers entre eux. Dans l'étape 3 nous avons

$$S_{i,j}(X)^{q_j} = \gcd(\tilde{G}_i(X)^{q_j}, H_j(X^{q_j})^{q_j}) = \gcd(\tilde{G}_i(X)^{q_j}, H_j(X^{q_j}))$$

puisque les $\tilde{G}_i$ sont séparables par le lemme 2. Par conséquent les $U_{0,j}$ sont bien définis et nous avons

$$W_j(X^{pq_j}) = \prod_{i=1}^{r} S_{i,j}(X^p)^{q_j} = \prod_{i=1}^{r} \gcd(G_i(X)^{pq_j}, H_j(X^{pq_j})) = \prod_{i=1}^{r} U_{i,j}(X)^{pq_j}.$$

Nous en déduisons que $H_j(X^{pq_j}) = U_{0,j}(X^{pq_j}) \prod_{i=1}^{r} U_{i,j}(X)^{pq_j}$ est vrai pour tout $j \in \{1, \dots, s\}$. Nous pouvons réécrire F en :

$$\begin{aligned} F(X) = S_0(X)S_1(X^p) &= \left(\prod_{i=1}^{r} \prod_{j=0}^{s} U_{i,j}(X)^{m_i} \right) \prod_{j=1}^{s} \left(U_{0,j}(X^{pq_j})^{n_j} \prod_{i=1}^{r} U_{i,j}(X)^{pq_j n_j} \right) \\ &= \left(\prod_{i=1}^{r} \prod_{j=1}^{s} U_{i,j}(X)^{m_i + pq_j n_j} \right) \left(\prod_{i=1}^{r} U_{i,0}(X)^{m_i} \right) \left(\prod_{j=1}^{s} U_{0,j}(X^{pq_j})^{n_j} \right). \end{aligned}$$

La factorisation retournée dans L satisfait bien (S_1). Les autres propriétés sont immédiatement satisfaites par construction.

Si $p \geq n+1$ alors $S_1 \in \mathbb{K}$, et l'algorithme ne fait rien. Nous pouvons donc supposer à partir de maintenant que $p \leq n$. Le coût suivant est une conséquence directe de la super-additivité de M, de l'inégalité $r \leq p-1$, et de $\deg(S_0) + p\deg(S_1) = n$. L'étape a calcule des puissances p^{es} et des réductions simultanées, totalisant ainsi

$$\begin{aligned} &O\left(\sum_{i=1}^{r} \deg(G_i) \log p + \sum_{i=1}^{r} (\mathsf{M}(\deg(S_1)) \log s + \mathsf{M}(\max(\deg(G_i), \deg(S_1)))) \right) \\ &\quad \subseteq O(\deg(S_0) \log d + p\mathsf{M}(S_1) \log d + \mathsf{M}(\deg(S_0))) \subseteq O(\mathsf{M}(n) \log n). \end{aligned}$$

Le coût total des pgcd dans les étapes b tombe dans

$$\begin{aligned} &O\left(\sum_{i=1}^{r} \sum_{j=1}^{s} \mathsf{M}(\deg(H_j(X^{q_j}))) \log(\deg(H_j(X^{q_j}))) \right) \\ &\quad \subseteq O(p\mathsf{M}(\deg(S_1)) \log n) \subseteq O(\mathsf{M}(n) \log n). \end{aligned}$$

Le coût total des réductions simultanées des étapes c tombe dans

$$\begin{aligned} &O\left(\sum_{i=1}^{r} \left(\mathsf{M}(\deg(G_i)) + \mathsf{M}\left(p \sum_{j=1}^{s} \deg(S_{i,j}) \right) \log n \right) \right) \\ &\quad \subseteq O(\mathsf{M}(\deg(S_0)) + \mathsf{M}(p\deg(S_1)) \log n) \subseteq O(\mathsf{M}(n) \log n). \end{aligned}$$

Le coût total des étapes d est borné par

$$O\left(\sum_{i=1}^{r} \sum_{j=1}^{s} \mathsf{M}(p\deg(S_{i,j})) \log n \right) \subseteq O(\mathsf{M}(n) \log n).$$

$\Phi^{-1}(G, q)$

Entrée : $(G, q) \in \mathcal{S} \times \mathcal{B}$.

Sortie : $(H, h) \in \mathbb{K}[X] \times \mathcal{B}$ avec H irréductible tel que $H^h = \Phi^{-1}(G, q)$.

1. Poser $\tilde{G} = G$ et $h = 1$.
2. Tant que $\tilde{G}(X^p)$ est une puissance p^{e} et tant que $h < q$ faire :
   Remplacer $\tilde{G}$ par la racine p^{e} de $\tilde{G}(x^p)$ et multiplier h par p.
3. Retourner $(\tilde{G}(X^{q/h}), h)$.

FIGURE 5. Calcul de l'inverse de la fonction de déflation.

Les étapes e et f coûtent

$$O\left(\sum_{i=1}^{r} \mathsf{M}(\deg(G_i)) \log n\right) \subseteq O(\mathsf{M}(n) \log n).$$

Finalement le coût de l'étape 3 est au plus

$$O\left(\sum_{j=1}^{s} \mathsf{M}(\deg(H_j(X^{q_j}))) \log n\right) \subseteq O(\mathsf{M}(n) \log n). \qquad \square$$

EXEMPLE 5. Avec l'exemple 1 nous entrons dans l'algorithme de la Figure 4 avec $\mathrm{sqr}(S_0) = \{(X+2, 1), (X, 2)\}$ et $\mathrm{sep}(S_1) = \{((X+1)(X+2), 1, 1)\}$, et obtenons les valeurs suivantes pour les $U_{i,j}$:

$i \setminus j$	0	1
0		$X+1$
1	1	$X+2$
2	X	1

EXEMPLE 6. Avec l'exemple 2 nous entrons dans l'algorithme de la Figure 4 avec $\mathrm{sqr}(S_0) = \{(X + 2X, 1)\}$ et $\mathrm{sqr}(S_1) = \{(X^2 + T, 1, 1), (X + 2T^3, 1, 2), (X + 2T^3, 3, 1)\}$, et obtenons les valeurs suivantes pour les $U_{i,j}$:

$i \setminus j$	0	1	2	3
0		X^2+T	1	$X+2T^3$
1	1	1	$X+2T$	1

Finalement les résultats de cette section se résument comme suit :

PROPOSITION 7. *La factorisation séparable d'un polynôme $F \in \mathbb{K}[X]$ de degré n peut être calculée au moyen de $O(\mathsf{M}(n) \log n)$ opérations dans $\mathbb{K}$.*

DÉMONSTRATION. La preuve est une conséquence directe des lemmes 4 et 5, et de la borne $\sum_{k \geq 0} \mathsf{M}(n/p^k) \log(n/p^k) \in O(\mathsf{M}(n) \log n)$. $\square$

5. Réduction à la factorisation des polynômes séparables

Une fois la factorisation séparable effectuée nous verrons dans les chapitres suivants comme réaliser la factorisation irréductible de chacun des facteurs séparables dans les situations courantes. Nous finissons donc ce chapitre en expliquant comment le problème de la factorisation irréductible se réduit au cas des polynômes séparables.

Nous supposons que $\mathbb{K}$ dispose d'une opération élémentaire permettant d'extraire une racine p^{e}, et expliquons comment implémenter l'inverse de la fonction de déflation Φ^{-1}.

Factorisation irréductible de F

Entrée : un polynôme $F \in \mathbb{K}[x]$ de degré $n \geq 1$.

Sortie : $\mathrm{irr}(F)$.

1. Calculer la factorisation séparable $\mathrm{sep}(F)$ de F.
2. Pour chaque $(G, q, m) \in \mathrm{sep}(F)$ calculer la factorisation irréductible de G.
3. Retourner

$$\bigcup_{(G,q,m)\in \mathrm{sep}(F)} \{(H(x^{q/h}), hm) \,|H^h = \Phi^{-1}(\bar{G}, q),$$

pour chaque facteur irréductible $\bar{G}$ de $G\}$.

FIGURE 6. Réduction de la factorisation au cas séparable.

LEMME 6. *L'algorithme de la Figure 5 est correct et coûte* $O(\deg(G) \log_p q)$ *extractions de racines* p^{es} *de* $\mathbb{K}$.

DÉMONSTRATION. Les quantités $\tilde{G}$ est h calculées par l'algorithme coïncident avec celles de la preuve de la proposition 4. □

Dans l'algorithme de la Figure 6 nous supposons disposer d'un algorithme de factorisation irréductible pour les polynômes séparables.

PROPOSITION 8. *L'algorithme de la Figure 6 est correct. Il effectue des factorisations irreducibles de polynômes séparables de* $\mathbb{K}[x]$ *dont la somme des degrés est au plus* n*, ainsi que* $O(\mathsf{M}(n) \log n)$ *opérations arithmétiques dans* $\mathbb{K}$ *et* $O(n)$ *extractions de racines* p^{es} *dans* $\mathbb{K}$.

DÉMONSTRATION. Le coût de la première étape est donné par la proposition 7 et le coût de la dernière étape par le lemme précédent. □

Notes

Cette partie sur la factorisation des polynômes est adaptée des notes d'un cours donné à l'occasion des *Journées Nationales de Calcul Formel* en 2013 [10].

La recherche des racines d'un polynôme à une variable et plus généralement la factorisation de tels polynômes en irréductibles trouve ses origines dans des temps très anciens des mathématiques. Un bref historique se trouve par exemple dans [4]. Une présentation plus détaillée sur la factorisation des polynômes se trouve dans les sections « Notes » des chapitres de la partie III du livre [5]. Nous fournirons des éclairages historiques plus précis dans les chapitres suivants. Précisons néanmoins dès maintenant que la plupart des logiciels de calcul formel offrent des fonctions pour factoriser les entiers et les polynômes. Le logiciel Magma propose une implémentation efficace couvrant tous les cas habituels [16]. Parmi les logiciels libres, de nombreux cas particuliers de factorisation sont disponibles dans NTL [15], Pari/gp [13], Sage [14], Singular [1], et plus récemment dans Mathemagix [7].

Concernant la factorisation sans carré, l'algorithme original de Musser peut être trouvé dans [12]. Son extension à la factorisation séparable par Gianni et Trager est publiée dans [6].

L'algorithme de Yun a été publié dans [17], et son extension à la factorisation séparable dans [9]. Par ailleurs, lorsque $\mathbb{K}$ est parfait, cette extension coïncide essentiellement avec l'algorithme de factorisation sans carré proposé dans [8, Section 4.6.3, Exercise 36] et [5, Exercise 14.30].

La preuve du théorème 1 utilise l'existence de la factorisation irréductible. Pour une autre preuve constructive le lecteur pourra consulter celle du livre [11, Chapter VI, Theorem 6.3].

Le fait qu'il existe des corps effectifs de caractéristique positive tels que l'extraction de racines p^{es} ne soit pas calculable a été établi par Fröhlich et Shepherdson en 1956 [2, Section 7]. Une autre preuve est esquissée dans [3, Remark 5.10].

Bibliographie

[1] Decker, W., G.-M. Greuel, G. Pfister et H. Schönemann (2012). Singular *3-1-6 — A computer algebra system for polynomial computations*. http://www.singular.uni-kl.de.

[2] Fröhlich, A. et J. C. Shepherdson (1956). « Effective procedures in field theory ». In : *Philosophical Transactions of the Royal Society A*, vol. 248, p. 407–432.

[3] Gathen, J. von zur (1984). « Hensel and Newton methods in valuation rings ». In : *Mathematics of Computation*, vol. 42, n°166, p. 637–661.

[4] — (2006). « Who was who in polynomial factorization ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computationt*. ACM Press, p. 1–2.

[5] Gathen, J. von zur et J. Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press.

[6] Gianni, P. et B. Trager (1996). « Square-free algorithms in positive characteristic ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 7, n°1, p. 1–14.

[7] Hoeven, J. van der, G. Lecerf, B. Mourrain et al. (2002). *Mathemagix*. http://www.mathemagix.org.

[8] Knuth, Donald E. (1997). *The Art of Computer Programming*. 3e éd. Vol. 2 : Seminumerical Algorithms. Computer Science and Information Processing. Addison-Wesley Publishing Co., xiv+762 pages.

[9] Lecerf, G. (2008). « Fast Separable Factorization and Applications ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 19, n°2, p. 135–160.

[10] — (2013). « Factorisation des polynômes à plusieurs variables (Cours no. II) ». In : *Journées nationales de calcul formel*. Éd. par G. Chèze, P. Boito, C. Pernet et M. Safey el Din. Vol. 3. Les cours du CIRM n°1. http://ccirm.cedram.org/ccirm-bin/fitem?id=CCIRM_2013__3_1_A2_0. Cedram, p. 1–85.

[11] Mines, R., F. Richman et W. Ruitenburg (1988). *A course in constructive algebra*. Universitext. Springer-Verlag.

[12] Musser, D. R. (1971). « Algorithms for Polynomial Factorization ». Thèse de doct. Univ. of Wisconsin : C.S. Department.

[13] *PARI/GP* (2012). Software available from http://pari.math.u-bordeaux.fr. The PARI Group. Bordeaux.

[14] Stein, W. A. et al. (2004). *Sage Mathematics Software*. http://www.sagemath.org. The Sage Development Team.

[15] Shoup, V. (from 1996). *NTL : A Library for doing Number Theory*. http://www.shoup.net.

[16] Steel, A. (2005). « Conquering inseparability : primary decomposition and multivariate factorization over algebraic function fields of positive characteristic ». In : *Journal of Symbolic Computation*, vol. 40, n°3, p. 1053–1075.

[17] Yun, David Y.Y. (1976). « On square-free decomposition algorithms ». In : *SYMSAC'76*. Yorktown Heights, New York, United States : ACM, p. 26–35. doi : 10.1145/800205.806320.

CHAPITRE 18

Factorisation des polynômes à une variable sur un corps fini

Résumé

Les algorithmes de factorisation des polynômes dépendent fortement du corps de leurs coefficients. Il est alors naturel de commencer avec le cas des corps finis. Dans ce chapitre nous présentons les algorithmes historiques qui sont encore très largement utilisés en pratique. Grâce aux résultats du chapitre précédent nous pouvons supposer que le polynôme à factoriser est séparable.

1. Corps finis, quelques rappels

Nous commençons par rappeler quelques résultats élémentaires de théorie des groupes dont nous avons besoin.

1.1. Préliminaires : groupes commutatifs finis, arithmétique. Soit G un groupe commutatif fini, noté multiplicativement (dans la suite, G sera le groupe $F_q^* = \mathbb{F}_q \setminus \{0\}$, $\mathbb{F}_q$ étant un corps fini à q éléments). L'*ordre* d'un élément a de G est le plus petit entier positif non nul γ tel que $a^\gamma = 1$. Dans la suite nous utiliserons les résultats suivants :

LEMME 1. *L'ordre de tout élément divise le cardinal* $\mathrm{card}(G)$ *de* G*, et donc* $a^{\mathrm{card}(G)} = 1$ *pour tout* a *de* G*.*

DÉMONSTRATION. L'ensemble des puissances de a est un sous-groupe de G, de cardinal γ. On utilise alors le théorème de Lagrange, qui dit que le cardinal d'un sous-groupe de G divise $\mathrm{card}(G)$. □

LEMME 2. *Soit* γ *le ppcm des ordres des éléments de* G*. Alors, il existe un élément d'ordre* γ *; en particulier,* γ *divise* $\mathrm{card}(G)$*.*

LEMME 3. *Soient* p, m, n *trois entiers positifs, avec* $p \geq 2$*. Si* $p^m - 1$ *divise* $p^n - 1$ *alors* m *divise* n*.*

DÉMONSTRATION. En écrivant $n = qm + r$, avec $0 \leq r < m$, nous avons $p^r = p^n \mod (p^m - 1)$, donc $p^r = 1 \mod (p^m - 1)$. Puisque $r < m$, nous déduisons que $p^r = 1$ et donc $r = 0$. □

1.2. Corps finis. L'exemple le plus simple de corps fini, noté $\mathbb{F}_p$, est un quotient de la forme $\mathbb{Z}/p\mathbb{Z}$, où p est un nombre premier. Attendu qu'un tel corps ne contient pas de sous-corps propre, il est dit *premier*.

Soit ensuite P un polynôme irréductible de $\mathbb{F}_p[X]$, de degré d. Alors, $\mathbb{K} = \mathbb{F}_p[X]/(P)$ est également un corps ; puisque $1, X, \ldots, X^{d-1}$ forment une base de $\mathbb{K}$ comme espace vectoriel sur $\mathbb{F}_p$, le cardinal de $\mathbb{K}$ est p^d. Si nous prenons un autre polynôme irréductible P de degré d nous obtenons un autre corps fini isomorphe à $\mathbb{K}$. Un corps fini à $q = p^d$ éléments est noté $\mathbb{F}_q$. En admettant l'existence de polynômes irréductibles de n'importe quel degré à coefficients dans $\mathbb{F}_p$, il est aisé

de construire n'importe quel corps fini de cardinal p^d quelle que soit la valeur de d. La proposition suivante montre que $\mathbb{F}_q$ correspond à l'ensemble des racines de $X^q - X$. Le *groupe multiplicatif* des éléments non nuls de $\mathbb{F}_q$ est noté $\mathbb{F}_q^*$. Son cardinal est $q-1$.

PROPOSITION 1. *Pout tout a dans $\mathbb{F}_q^*$, nous avons $a^{q-1} = 1$ et $X^q - X = \prod_{a\in\mathbb{F}_q}(X-a)$.*

DÉMONSTRATION. Pour tout a dans $\mathbb{F}_q^*$, le Lemme 1 nous donne $a^{q-1} = 1$. Par conséquent $a^q = a$ pout tout a de $\mathbb{F}_q$, nul ou pas. Ceci montre que le polynôme $X^q - X$ s'annule sur tous les éléments de $\mathbb{F}_q$, donc $\prod_{a\in\mathbb{F}_q}(X-a)$ divise $X^q - X$. Comme les deux polynômes sont unitaires et de même degré, ils sont égaux. □

COROLLAIRE 1. *Pour tout $i = 1, \ldots, p-1$, le coefficient binomial $\binom{p}{i}$ est divisible par p.*

DÉMONSTRATION. Rappelons que ce coefficient binomial est le coefficient de X^i dans $(X+1)^p$. La proposition précédente entraîne les égalités

$$\begin{array}{rcl} X^p - X & = & \prod_{a\in\mathbb{F}_p}(X-a) \;=\; \prod_{a\in\mathbb{F}_p}(X-a+1) \\ & = & \prod_{a\in\mathbb{F}_p}((X+1)-a) \;=\; (X+1)^p - (X+1). \end{array}$$

On en déduit que $(X+1)^p = X^p + 1$, qui est ce que l'on voulait montrer. □

Le corollaire suivant est, d'une manière ou d'une autre, à la base de la plupart des résultats « arithmétiques » sur les corps finis.

COROLLAIRE 2. *Soit F un polynôme quelconque dans $\mathbb{F}_q[X]$ et soit A le quotient $\mathbb{F}_q[X]/(F)$ (qui n'est pas nécessairement un corps). L'application, dite de* Frobenius,

$$\phi_F : \begin{array}{ccc} A & \to & A \\ a & \mapsto & a^q \end{array}$$

est $\mathbb{F}_q$-linéaire.

DÉMONSTRATION. Soient λ dans $\mathbb{F}_q$ et a, b dans A. L'égalité $\phi_F(\lambda a) = \lambda\phi(a)$ est une conséquence de la Proposition 1, appliquée à $\mathbb{F}_q$. L'égalité $\phi_F(a+b) = \phi_F(a)+\phi_F(b)$ est une conséquence du corollaire précédent, en développant le produit $(a+b)^q$. □

Voyons quelques conséquences de ce corollaire.

PROPOSITION 2. *Pour tous $c_0, \ldots, c_n$ dans $\mathbb{F}_q$, nous avons l'égalité suivante :*

$$\left(\sum_{i=0}^{n} c_i X^i\right)^q = \sum_{i=0}^{n} c_i X^{iq}.$$

DÉMONSTRATION. Il s'agit d'une conséquence facile de la Proposition 1 et du Corollaire 1. □

PROPOSITION 3. *Pour tout $e \geq 1$, le polynôme $X^{q^e} - X$ est le produit des polynômes unitaires et irréductibles de $\mathbb{F}_q[X]$ dont le degré divise e.*

DÉMONSTRATION. Soit F un polynôme irréductible de $\mathbb{F}_q[X]$ de degré n divisant e, et soit $\mathbb{K}$ le quotient $\mathbb{F}_q[X]/(F)$. Soit ensuite x la classe de X dans $\mathbb{K}$; la Proposition 1 appliquée à x montre que $x^{q^n-1} = 1$ donc $x^{q^n} = x$, d'où $x^{q^e} = x$. Par conséquent F divise $X^{q^e} - X$.

Réciproquement, soit F un facteur irréductible de degré n de $X^{q^e} - X$; il faut montrer que n divise e. Soit $\mathbb{K}$ le quotient $\mathbb{F}_q[X]/(F)$ et soit x l'image de X dans

$\mathbb{K}$. Puisque $F(x) = 0$, alors $x^{q^e} - x$ est nul. Pour tous λ_i dans $\mathbb{F}_q$, nous avons alors les égalités suivantes :

$$(\sum_i \lambda_i x^i)^{q^e} = \sum_i \lambda_i (x^i)^{q^e} = \sum_i \lambda_i (x^{q^e})^i = \sum_i \lambda_i x^i,$$

la première égalité s'obtenant en appliquant n fois le Corollaire 2. Donc tout élément de $\mathbb{K}$ est racine de $X^{q^e} - X$, de sorte que tout élément non nul de $\mathbb{K}$ est racine de $X^{q^e-1} - 1$.

Soit maintenant γ l'exposant du groupe $\mathbb{K} \setminus \{0\}$; d'après le Lemme 2, γ divise $\mathrm{card}(\mathbb{K} \setminus \{0\}) = q^n - 1$. Par ailleurs, par définition, tout élément non nul de $\mathbb{K}$ annule $X^\gamma - 1$, ce qui implique $\gamma \geq q^n - 1$, et donc $\gamma = q^n - 1$. Le Lemme 2 montre ensuite qu'il existe un élément a d'ordre $q^n - 1$ dans $\mathbb{K} \setminus \{0\}$. De $a^{q^e-1} = 1$, on déduit que $q^n - 1$ divise $q^e - 1$. Le Lemme 3 permet alors de conclure que n divise e.

À ce stade, on sait donc que les facteurs irréductibles de $X^{q^e} - X$ sont exactement les polynômes irréductibles de degré divisant e. Il reste à montrer qu'il n'existe pas de facteur multiple. Pour cela, il suffit de constater que la dérivée de $X^{q^e} - X$ vaut -1, et en particulier que son pgcd avec $X^{q^e} - X$ vaut 1. Il suffit pour finir de faire appel à la Proposition 1. □

2. Algorithme de Berlekamp

Nous commençons par présenter des algorithmes de factorisation, attribués à Berlekamp. À partir de maintenant, le polynôme F à factoriser est supposé unitaire, séparable, de degré n, et a ses coefficients dans le corps fini $\mathbb{F}_q$, avec $q = p^d$. Nous noterons $F_1, \ldots, F_r$ les facteurs irréductibles unitaires de F, que nous cherchons à calculer, de sorte que $F = F_1 \cdots F_r$.

2.1. Réduction à l'algèbre linéaire.

En vertu du Corollaire 2, l'application

$$\begin{array}{rccc} \psi_F : & \mathbb{F}_q[X]/(F) & \to & \mathbb{F}_q[X]/(F) \\ & G & \mapsto & G^q - G \end{array}$$

est une application $\mathbb{F}_q$-linéaire. L'étude du noyau de ψ_F va permettre de factoriser F, à partir de la proposition suivante :

Proposition 4. *Un polynôme G de degré au plus $n-1$ est dans le noyau de ψ_F si, et seulement si, G est une constante modulo chaque F_i.*

Démonstration. Par la Proposition 1, le polynôme $X^q - X$ se factorise dans $\mathbb{F}_q$, et donc dans $\mathbb{F}_q[X]/(F_i)$, sous la forme $\prod_{a \in \mathbb{F}_q}(X - a)$. Puisque $\mathbb{F}_q[X]/(F_i)$ est un corps contenant $\mathbb{F}_q$, un élément $b \in \mathbb{F}_q[X]/(F_i)$ est une racine de $X^q - X$ si et seulement s'il annule un des facteurs $X - a$, donc si, et seulement si, $a_i \in \mathbb{F}_q$. □

En notant $\hat{F}_i = F/F_i$, nous pouvons maintenant observer que les $B_i = (\hat{F}_i F_i'/F') \mod F$ pour i de 1 à r satisfont la propriété suivante : $B_i \mod F_j$ vaut 1 si $j = i$ et 0 sinon. Il s'en suit que les B_i forment une base du noyau de ψ_F. Un polynôme G dans le noyau de ψ_F est donc nécessairement une combinaison linéaire sur $\mathbb{F}_q$ de la forme $G = \sum_{i=1}^n b_i B_i$. La proposition précédente donne déjà un moyen de calculer le nombre r de facteurs irréductibles de F. La proposition suivante donnera une méthode pour en déduire un scindage non trivial de F.

Proposition 5. *Si G est non constant dans le noyau de ψ_F, alors les constantes $G \mod F_i$ se sont pas toutes identiques.*

Démonstration. Puisque tous les F_i sont distincts et irréductibles, par le théorème des restes chinois, nous avons un isomorphisme

$$\mathbb{F}_q[X]/(F) = \mathbb{F}_q[X]/(F_1) \times \cdots \times \mathbb{F}_q[X]/(F_r),$$

Scindage de F

Entrée : F de degré n dans $\mathbb{F}_q[X]$.

Sortie : un facteur non trivial de F si F n'est pas irréductible.

1. Calculer la matrice de l'application ψ_F.
2. Si le rang de ψ_F est $n-1$ alors retourner le fait que F est irréductible.
3. Choisir un polynôme G de degré au plus $n-1$ et non constant dans le noyau de ψ_F.
4. Calculer tous les $\gcd(F, G-a)$ pour toutes les valeurs de a dans $\mathbb{F}_q$, jusqu'à trouver un scindage non trivial de F.

FIGURE 1. Algorithme déterministe de Berlekamp.

donné par

$$G \mod F \mapsto (G \mod F_1, \ldots, G \mod F_r).$$

Les F_i étant irréductibles, chaque $\mathbb{F}_q[X]/(F_i)$ est un corps. Dans la représentation « produit de corps », l'application ψ_F s'écrit bien sûr :

$$(G_1, \ldots, G_r) \mapsto (G_1^q - G_1 \mod F_1, \ldots, G_r^q - G_r \mod F_r).$$

Si tous les $G_i = G \mod F_i$ sont les mêmes, alors nécessairement $G = G_i$ pout tout i, car le degré de G est plus petit que F. En particulier, G devrait être constant, ce qui est contraire à nos hypothèses. □

2.2. Algorithme déterministe. La clé de l'algorithme déterministe de Berlekamp peut se formuler comme suit :

PROPOSITION 6. *Soit G comme dans la proposition précédente. Il existe a dans $\mathbb{F}_q$ tel que $\gcd(F, G-a)$ est un diviseur non trivial de F.*

DÉMONSTRATION. Pour a quelconque dans $\mathbb{F}_q$, nous avons l'égalité $\gcd(F, G-a) = \prod_i F_i$, où le produit est pris sur tous les F_i tels que F_i divise $G-a$, c'est-à-dire tels que $G \mod F_i = a$. Puisqu'il existe au moins deux valeurs distinctes pour les $G \mod F_i$ et que chacune d'entre elles donne un diviseur de F, ces diviseurs sont stricts. □

Dans la suite de ce chapitre nous notons $\omega > 2$ un exposant faisable pour la multiplication de matrices sur $\mathbb{F}_q$, tel que défini dans le Chapitre 8.

PROPOSITION 7. *L'algorithme de la Figure 1 calcule un facteur non trivial de F au moyen d'au plus $O(n^\omega + q\mathsf{M}(n)\log n)$ opérations arithmétiques dans $\mathbb{F}_q$.*

DÉMONSTRATION. Le premier pas consiste à calculer la matrice de ψ_F dans la base $1, X, \ldots, X^n - 1$. On commence par calculer $X^q \mod F$: cela demande $O(\log q)$ opérations modulo F, soit $O(\mathsf{M}(n)\log q)$ opérations dans $\mathbb{F}_q$. Ensuite, il faut calculer $(X^2)^p = (X^p)^2, (X^3)^p = (X^p)^3, \ldots$, ce qui se fait en n multiplications modulo F, soit $O(n\mathsf{M}(n))$ opérations en tout.

L'algèbre linéaire coûte $O(n^\omega)$. Une fois obtenu un vecteur du noyau, chaque essai de pgcd coûte $O(\mathsf{M}(n)\log n)$ opérations, et il faut en faire au plus q. □

Pour trouver la factorisation complète, il suffit d'itérer le procédé, ce qui entraîne *a priori* un surcoût de n. En fait, il n'y a pas besoin de refaire de l'algèbre linéaire, mais seulement de parcourir tous les vecteurs du noyau de ψ_F, de sorte que le coût total de la factorisation est

$$O(n^\omega + qn\mathsf{M}(n)\log n)$$

opérations de $\mathbb{F}_q$. Nous avons alors montré le résultat suivant :

PROPOSITION 8. *Un polynôme F séparable de $\mathbb{F}_q[X]$ de degré n peut être factorisé en effectuant au plus $O(n^\omega + qn\mathsf{M}(n)\log n)$ opérations de $\mathbb{F}_q$.*

La complexité de cet algorithme est linéaire en q, ce qui le rend rapidement impraticable, dès lors que q atteint quelques milliers. Par ailleurs, supposer F séparable n'est pas restrictif ici puisque les techniques du chapitre précédent permettent de se ramener facilement à ce cas.

À l'heure actuelle, on ne connaît pas d'algorithme de factorisation déterministe de complexité polynomiale, c'est-à-dire en n et $\log q$. Nous présenterons dans la suite des algorithmes probabilistes de coûts moyens polynomiaux.

2.3. Algorithme probabiliste. L'algorithme probabiliste de Berlekamp repose essentiellement sur les même ingrédients que la version déterministe précédente. Supposons p impair, c'est-à-dire autre que 2. L'entier $q-1$ est alors divisible par 2 et la puissance $(q-1)/2$ d'un élément de $\mathbb{F}_q$ vaut soit -1 soit 1. Sur l'ensemble des éléments non nuls il y a autant de telles puissances valant -1 que 1. Donc si G est pris comme une combinaison linéaire non nulle des vecteurs d'une base fixe du noyau de ψ_F, alors on peut attendre que

$$G^{(q-1)/2} = \sum_{i=1}^{r} b_i^{(q-1)/2} B_i^{(q-1)/2}$$

ait quelques possibles valeurs $b_i^{(q-1)/2}$ qui valent 0 et la plupart des autres qui valent -1 ou 1 à peu près en même nombre. Du coup, il est possible que $\operatorname{pgcd}(G, F)$ soit un facteur non trivial de F, mais en général ce n'est pas le cas et on examine $\operatorname{pgcd}(G^{(q-1)/2} - 1, F)$. Ce dernier polynôme est souvent un facteur non trivial de F contenant une quantité proche de $r/2$ facteurs irréductibles en moyenne. Nous formaliserons d'avantage cette description de l'algorithme dans les paragraphes suivants. Lorsque $p = 2$, l'entier $q-1$ n'est pas divisible par 2 et la technique que nous venons d'ébaucher ne s'applique plus. Il est alors d'usage de faire appel à la construction suivante un peu plus coûteuse en calculs :

DÉFINITION 1. *Si $p = 2$, le polynôme de la d-ième trace sur $\mathbb{F}_2$ est* $\operatorname{Tr}_d(X) = X^{2^{d-1}} + X^{2^{d-2}} + \cdots + X^4 + X^2 + X + 1 \in \mathbb{F}_{2^d}[X]$.

LEMME 4. *Pour tout $a \in \mathbb{F}_{2^d}$ nous avons $\operatorname{Tr}_d(a) \in \mathbb{F}_2$ et $\operatorname{card}(\operatorname{Tr}_d^{-1}(0)) = \operatorname{card}(\operatorname{Tr}_d^{-1}(1)) = 2^{d-1}$.*

DÉMONSTRATION. Puisque $X^{2^d} + X = \operatorname{Tr}_d^2(X) + \operatorname{Tr}_d(X) = \operatorname{Tr}_d(X)(\operatorname{Tr}_d(X)+1)$ nous avons bien $\operatorname{Tr}_d(a) \in \mathbb{F}_2$. Nous obtenons ainsi le fait que Tr_d a toutes ses 2^{d-1} racines dans $\mathbb{F}_{2^d}$. □

La première partie de l'algorithme probabiliste de Berlekamp est le calcul d'une base $G_1, \ldots, G_r$ du noyau de ψ_F. La deuxième partie, résumée dans la Figure 2, consiste à calculer tous les facteurs irréductibles de F à partir de cette base. Rappelons que le coût moyen d'un algorithme pour une entrée donnée est la moyenne des coûts sur toutes les exécutions possibles.

PROPOSITION 9. *L'algorithme de la Figure 2 est correct et nécessite au plus $O(nr^{\omega-1}\log r + (r + \log n + \log q)\mathsf{M}(n)\log r)$ opérations dans $\mathbb{F}_q$ en moyenne.*

DÉMONSTRATION. Si $\operatorname{pgcd}(F, G)$ à l'étape 2 n'est pas constant alors il s'agit d'un facteur non trivial de F. Si p est impair, les facteurs irréductibles de H_1 à l'étape 5 sont les F_i tels que $G^{(q-1)/2} \mod F_i$ vaut 1.

Facteurs irréductibles de F à partir d'une base du noyau de ψ_F

Entrée : $F \in \mathbb{F}_q[x]$ séparable de degré n, et une base $G_1, \ldots, G_r$ du noyau de ψ_F.

Sortie : les facteurs irréductibles $F_1, \ldots, F_r$ de F.

1. Si $r = 1$ alors retourner F.
2. Choisir des éléments $g_1, \ldots, g_r$ dans $\mathbb{F}_q$ à l'aide d'un générateur aléatoire, et calculer $G = g_1 G_1 + \cdots + g_r G_r$.
3. Si $H_1 = \gcd(F, G)$ n'est pas constant alors aller à l'étape 6.
4. Calculer $H = G^{\frac{q-1}{2}} - 1 \mod F$ si p est impair, et $H = \mathrm{Tr}_d(G) \mod F$ si $p = 2$.
5. Si $H_1 = \gcd(F, H)$ est constant alors retourner à l'étape 2.
6. Calculer $H_2 = F/H_1$.
7. Calculer une base $G_{1,1}, \ldots, G_{r_1,1}$ des $(G_i F' \mod F)/(H_1' H_2) \mod H_1$ pour $i \in \{1, \ldots, r\}$.
8. Calculer une base $G_{1,2}, \ldots, G_{r_2,2}$ des $((G_i F' \mod F)/(H_1 H_2') \mod H_2$ pour $i \in \{1, \ldots, r\}$.
9. Retourner la réunion des facteurs obtenus par l'appel récursif avec $H_1, G_{1,1}, \ldots, G_{r_1,1}$, d'une part, puis $H_2, G_{1,2}, \ldots, G_{r_2,2}$ d'autre part.

FIGURE 2. Deuxième partie de l'algorithme probabiliste de Berlekamp

Nous avons vu que chaque r-uplet $(c_1, \ldots, c_r)$ est en bijection avec un r-uplet $(b_1, \ldots, b_r)$ de $\mathbb{F}_q^r$ satisfaisant $G = \sum_{i=1}^r b_i B_i$. Sur les q^r valeurs possibles du r-uplet $(b_1, \ldots, b_r)$, $(q-1)^r$ d'entre elles sont dans $(\mathbb{F}_q^*)^r$. Donc pour $q^r - 1 - (q-1)^r$ un scindage non trivial de F est trouvé dès l'étape 3. Si p est impair, les r-uplets $(b_1^{(q-1)/2}, \ldots, b_r^{(q-1)/2})$ sont équi-distribués dans $\{-1, 1\}^r$. Seuls les deux tels uplets constants sur les 2^r possibles ne conduisent pas à un scindage non trivial de F à l'étape 5. La proportion d'uplets qui ne conduisent pas à un scindage non trivial est donc

$$\frac{1 + (q-1)^r/2^{r-1}}{q^r} \leq 1/2.$$

Le nombre moyen d'uplets à essayer est donc borné par une constante. Autrement dit, l'algorithme arrive rapidement à l'étape 6 en moyenne.

À partir de l'identité

$$(B_i F' \mod F)/(H_1' H_2) \equiv (F/F_i)/(H_1' H_2) \equiv (H_1/F_i) H_1' \mod H_1,$$

nous voyons que $(B_i F' \mod F)/(H_1' H_2)$ est dans le noyau de ψ_{H_1} pour tout $i \in \{1, \ldots, r\}$, et que les $G_{i,1}$ forment bien un ensemble générateur du noyau de ψ_{H_1}. Il en va de même concernant H_2. Par conséquent l'algorithme fonctionne correctement. Nous laissons le soin au lecteur de traiter le cas particulier où $p = 2$.

L'étape 2 coûte $O(rn)$ opérations. L'étape 4 coûte $O((\log n + \log q)\mathsf{M}(n))$ opérations. Les étapes 3, 5, et 6 utilisent $O(\mathsf{M}(n) \log n)$ opérations. Les étapes 7 et 8 nécessitent $O((r + \log n)\mathsf{M}(n) + nr^{\omega-1})$ opérations. Le nombre d'opérations total en moyenne avant d'entrer dans les appels récursifs est donc $O(nr^{\omega-1} + (r + \log n + \log q)\mathsf{M}(n))$.

Considérons maintenant l'arbre des appels récursifs. Au début, à la racine, le degré du polynôme en entrée est n. Si $r \geq 2$, à la profondeur 1 de cet arbre, il y a deux nœuds correspondant à H_1 et H_2 trouvés par l'algorithme. La somme des degrés de H_1 et H_2 est n. Il est aisé de constater qu'à une profondeur donnée la

Factorisation irréductible de F

Entrée : F séparable dans $\mathbb{F}_q[X]$.

Sortie : factorisation irréductible de F.

1. Calculer, dans la base $1, X, \ldots, X^{n-1}$, la matrice de l'endomorphisme ψ_F.
2. Calculer une base $G_1, \ldots, G_r$ du noyau de ψ_F.
3. Appeler l'algorithme de la Figure 2 avec $F, G_1, \ldots, G_r$ et retourner la factorisation obtenue.

FIGURE 3. Algorithme probabiliste de Berlekamp.

somme des degrés de tous les polynômes à factoriser est toujours au plus n. De part la super-additivité de M, le coût total des calculs propres à tous les nœuds d'une même profondeur est donc $O(nr^{\omega-1} + (r + \log n + \log q)\mathsf{M}(n))$ opérations dans $\mathbb{F}_q$. Il reste à montrer que la profondeur moyenne de l'arbre des appels récursifs est $O(\log r)$ pour conclure l'analyse de complexité.

Considérons le cas où p est impair. Notons ℓ la variable aléatoire représentant la profondeur de l'arbre, et $E(\ell)$ la moyenne des profondeurs. Si $E(\ell \geq h)$ représente la probabilité que la profondeur soit au moins h, alors

$$E(\ell) = \sum_h h(E(\ell \leq h) - E(\ell \leq h+1)) = \sum_h E(\ell \geq h).$$

La probabilité que deux facteurs $i < j$ qui sont dans le même nœud à la profondeur $h-1$ le reste à la profondeur h est $1/q^2 + 1/2$. Elle correspond aux possibilités suivantes pour (b_i, b_j) : $(0,0)$, $(b_i, b_j) \in (\mathbb{F}_q^*)^2$ avec $b_i^{(q-1)/2} = b_j^{(q-1)/2}$. Par conséquent, la probabilité que deux facteurs $i < j$ soient dans le même nœud jusqu'à la profondeur h est $(1/q^2 + 1/2)^h$. En prenant la somme de toutes ces probabilités sur toutes les paires $i < j$, on déduit que la probabilité $E(\ell \geq h)$ que tous les facteurs de F n'aient pas été découverts à la profondeur $h-1$ est au plus $r^2(1/q^2 + 1/2)^{h-1} \leq r^2(3/4)^{h-1}$. Finalement nous déduisons la borne souhaitée :

$$\sum_h E(\ell \leq h) \leq \sum_{h \leq (2\log h)/\log(3/4)} E(\ell \leq h) + \sum_{h > (2\log h)/\log(3/4)} E(\ell \leq h) = O(\log r).$$

À nouveau, nous laissons l'étude du cas $p = 2$ en exercice. □

Nous obtenons finalement une version probabiliste de l'algorithme de Berlekamp dans la Figure 3.

PROPOSITION 10. *L'algorithme de la Figure 3 est correct et utilise en moyenne* $O(n^\omega \log n + (n + \log q)\mathsf{M}(n) \log n)$ *opérations dans* $\mathbb{F}_q$.

DÉMONSTRATION. Le fait que l'algorithme est correct provient de la Proposition 9. Le coût de la première étape est dans $O((n + \log q)\mathsf{M}(n))$. L'étape 2 effectue $O(n^\omega)$ opérations. Le reste du coût provient essentiellement de la proposition précédente. □

3. Algorithme de Cantor et Zassenhaus

Une composante commune à de nombreux algorithmes de factorisation sur les corps finis est la suite des itérés de Frobenius $(\Phi_i)_{i \geq 0}$, relative à F définie par $\Phi_i = X^{q^i} \mod F$. D'une façon naïve $\Phi_0, \ldots, \Phi_n$ peuvent être calculés en utilisant $O(n\mathsf{M}(n) \log q)$ opérations dans $\mathbb{F}_q$.

La dépendance du coût du calcul des itérés de Frobenius peut être diminuée en considérant l'opération de composition modulaire définie comme le calcul de $G \circ H$

Factorisation par degrés de F

Entrée : F dans $\mathbb{F}_q[X]$ de degré n.

Sortie : la factorisation $E_1, \ldots, E_n$ par degrés des facteurs irréductibles de F.

1. Calculer $\Phi_0, \ldots, \Phi_n$.
2. Pour chaque i de 1 à n : calculer $E_i = \gcd(\Phi_i - X, F)$ puis remplacer F par F/E_i.
3. Retourner $E_1, \ldots, E_n$.

FIGURE 4. Algorithme de factorisation par degrés.

mod F pour G et H deux polynômes de degré au plus $n-1$. Nous pouvons calculer Φ_1 en $O(\mathsf{M}(n)\log q)$ opérations puis obtenir Φ_{i+1} comme $\Phi_1 \circ \Phi_i \mod F$. Avec cette idée, pour calculer calculer les $n+1$ premiers itérés, il existe une technique spécifique exploitant l'évaluation et interpolation rapide d'un polynôme en plusieurs points du Chapitre 5, et donnant le résultat suivant :

PROPOSITION 11. *Avec les notations précédentes, une fois Φ_1 connu, $\Phi_0, \ldots, \Phi_l$, pour $l \leq n$, peuvent être calculés avec $O(\mathsf{M}(n)^2 \log n \log l)$ opérations dans $\mathbb{F}_q$.*

DÉMONSTRATION. L'évaluation de Φ_i en $\Phi_1, \ldots, \Phi_i$ modulo F donne $\Phi_{i+1}, \ldots, \Phi_{2i}$. Le coût total découle alors de la Proposition 3 du Chapitre 5. □

L'algorithme de Cantor et Zassenhaus comporte deux étapes. La première consiste à décomposer F en un produit de polynômes $\prod_{i\geq 1} E_i$ où chaque E_i ne comporte que des facteurs irréductibles de degré exactement i. Nous parlons alors de *factorisation par degrés* des facteurs. La deuxième étape consiste à factoriser individuellement chaque E_i en irréductibles.

PROPOSITION 12. *L'algorithme de la Figure 4 est correct et nécessite $O(\mathsf{M}(n)(\mathsf{M}(n)\log^2 n + \log q))$ opérations dans $\mathbb{F}_q$.*

DÉMONSTRATION. La formule utilisée pour calculer E_i repose sur la Proposition 3. Le coût total provient directement de la proposition précédente. □

Notons que l'algorithme précédent ne fait pas intervenir de valeurs aléatoires. L'étape suivante consiste à supposer que F est le produit de r polynômes irréductibles de degré exactement n/r.

PROPOSITION 13. *L'algorithme de la Figure 5 est correct et effectue $O(n\mathsf{M}(n)\log q \log n)$ opérations dans $\mathbb{F}_q$ en moyenne. Ce coût tombe dans $O(\mathsf{M}(n)^2 \log^3 n + \mathsf{M}(n)\log q \log n)$ lorsque p est impair.*

DÉMONSTRATION. En notant $\hat{F}_i = F/F_i$, nous avons précédemment observé que les $B_i = (\hat{F}_i F_i'/F') \mod F$ pour i de 1 à r satisfont la propriété suivante : $B_i \mod F_j$ vaut 1 si $j = i$ et 0 sinon.

Si G est un polynôme de degré au plus $n-1$, il s'écrit d'une unique façon sous la forme $\sum_{i=1}^{r} C_i(X)B_i(X)$ avec $C_i(X) \in \mathbb{F}_q[X]$ de degré au plus $n/r - 1$, par le théorème des restes chinois. Du coup $G^{(q^{n/r}-1)/2} \equiv C_i^{(q^{n/r}-1)/2} \mod F_i$ vaut assez rarement 0, et plus fréquemment soit -1 soit 1 dans la même proportion. Donc H_1 définit bien un scindage non trivial de F en général. Le résultat retourné par l'algorithme est donc correct. Nous laissons le soin au lecteur d'adapter la preuve de l'algorithme probabiliste de Berlekamp, et en particulier de prouver que la profondeur moyenne des appels récursifs est dans $O(\log n)$.

Factorisation de F

Entrée : F séparable de degré n ayant tous ses facteurs irréductibles de degré n/r.

Sortie : la factorisation irréductible $F_1, \ldots, F_r$ de F.

1. Si $r = 1$ alors retourner F.
2. Répéter :
   a. Choisir un polynôme G de degré au plus $n-1$ d'une façon aléatoire.
   b. Calculer $H_1 = \mathrm{pgcd}(F, G)$. Si H_1 n'est pas constant alors aller à l'étape 3.
   c. Calculer $H = G^{\frac{q^{n/r}-1}{2}} \mod F$ si p est impair et $H = \mathrm{Tr}_{kn/r}(G) \mod F$ si $p = 2$.
   d. Calculer $H_1 = \gcd(F, H-1)$.
   e. Si H_1 n'est pas constant alors aller à l'étape 3.
3. Calculer $H_2 = F/H_1$ et appeler récursivement la fonction sur H_1 et H_2 et retourner la réunion des facteurs.

FIGURE 5. Algorithme de factorisation en équi-degré.

Le coût du calcul de H à l'étape 2 peut se faire en $O(n\mathsf{M}(n)\log q)$. Les autres opérations totalisent un coût dans $O(\mathsf{M}(n)\log n)$. Lorsque p est impair il est intéressant de procéder comme suit. En notant $m = n/r$ et $\alpha = H \mod F$, comme

$$\frac{q^m - 1}{2} = (q^{m-1} + q^{m-2} + \cdots + q + 1)\frac{q-1}{2},$$

nous pouvons calculer $\alpha^{\frac{q^m-1}{2}} = (\alpha^{q^{m-1}} \cdots \alpha^q \alpha)^{\frac{m-1}{2}}$ grâce à la proposition 11. □

Nous arrivons au résultat principal de cette section :

THÉORÈME 1. *Un polynôme de degré n dans $\mathbb{F}_q[x]$ peut être factorisé avec un nombre moyen d'opérations dans $\mathbb{F}_q$ dans $\tilde{O}(n^2 \log q)$. Ce coût tombe dans $\tilde{O}(n^2 + n\log q)$ si p est impair.*

DÉMONSTRATION. Il s'agit d'une conséquence des Propositions 12 et 13. □

Notes

La présentation choisie ici est inspirée de [8, Chapter 14]. Le calcul des itérés de Frobenius constitue une brique importante en terme des performances pratiques. Les algorithmes décrits dans ce chapitre sont essentiellement ceux utilisés en pratique à ce jour. Ils peuvent être encore un peu améliorés et adaptés pour la recherche des racines, pour tester l'irréductibilité, ou bien encore pour construire des polynômes irréductibles de degré donné. Des analyses détaillées des aspects probabilistes ont été présentées dans l'article [6]. Comme autres livres sur les corps finis et leurs applications, le lecteur pourra être intéressé par consulter [14, 20].

Les premières techniques connues de factorization des polynômes à coefficients dans un corps fini remontent aux travaux de Gauss en 1798, Galois en 1830, et Arwins en 1918. Les premiers algorithmes performants ont été formalisés par Berlekamp [1, 2], Zassenhaus [21], puis Cantor et Zassenhaus [4]. Ils regroupent les principales idées mathématiques que nous avons présentées ici.

Dans l'article [10], Kaltofen et Shoup ont relié le problème de factorisation au problème de composition modulaire et ont obtenu une borne de complexité dans $O(d^{1,815}\log q)$ en terme du nombre d'opérations effectuées dans $\mathbb{F}_q$. Sur

un corps fini, Kedlaya et Umans ont ensuite conçu un algorithme de composition modulaire ayant un coût binaire quasi-linéaire [11], permettant de déduire de [10] un algorithme de factorisation sur $\mathbb{F}_q[X]$ ayant un coût binaire moyen dans $(d^{1,5} + d \log q)^{1+o(1)} \log q$, ce qui représente la meilleure borne de complexité asymptotique connue à ce jour. Malheureusement cette approche ne semble pas plus efficace que l'algorithme de Cantor et Zassenhaus pour les tailles de problèmes que l'on peut traiter avec les ordinateurs actuels.

La conception d'algorithmes déterministes en temps polynomial est toujours un sujet de recherche actif. Des solutions partielles sont données dans des cas particuliers, selon les propriétés du polynômes ou bien du corps fini [3, 7, 9, 12, 13, 15, 16, 17, 18, 19, 22]. Un algorithme de coût sous-exponentiel a été proposé par Evdokimov [5].

Bibliographie

[1] Berlekamp, E. R. (1967). « Factoring polynomials over finite fields ». In : *Bell System Technical Journal*, vol. 46, p. 1853–1859.

[2] — (1970). « Factoring polynomials over large finite fields ». In : *Math. Comp.* vol. 24, p. 713–735.

[3] Camion, P. (1983). « A deterministic algorithm for factorizing polynomials of $\mathbf{F}_q[X]$ ». In : *Combinatorial mathematics (Marseille-Luminy, 1981)*. Vol. 75. North-Holland Math. Stud. North-Holland, Amsterdam, p. 149–157.

[4] Cantor, D. G. et H. Zassenhaus (1981). « A new algorithm for factoring polynomials over finite fields ». In : *Mathematics of Computation*, vol. 36, n°154, p. 587–592.

[5] Evdokimov, S. (1994). « Factorization of polynomials over finite fields in subexponential time under GRH ». In : *Algorithmic number theory (Ithaca, NY, 1994)*. Vol. 877. Lecture Notes in Computer Science. Springer-Verlag, p. 209–219.

[6] Flajolet, Ph. et J.-M. Steyaert (1982). « A branching process arising in dynamic hashing, trie searching and polynomial factorization ». In : *Automata, Languages and Programming*. Éd. par M. Nielsen et E. Schmidt. Vol. 140. Lecture Notes in Computer Science. Springer-Verlag, p. 239–251.

[7] Gathen, J. von zur (1987). « Factoring polynomials and primitive elements for special primes ». In : *Theoretical Computer Science*, vol. 52, n°1-2, p. 77–89.

[8] Gathen, J. von zur et J. Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press.

[9] Huang, Ming-Deh A. (1991). « Generalized Riemann hypothesis and factoring polynomials over finite fields ». In : *Journal of Algorithms*, vol. 12, n°3, p. 464–481.

[10] Kaltofen, E. et V. Shoup (1998). « Subquadratic-time factoring of polynomials over finite fields ». In : *Mathematics of Computation*, vol. 67, n°223, p. 1179–1197.

[11] Kedlaya, Kiran S. et Christopher Umans (2008). « Fast Modular Composition in any Characteristic ». In : *FOCS'08 : IEEE Conference on Foundations of Computer Science*. IEEE Computer Society, p. 146–155. doi : 10.1109/FOCS.2008.13.

[12] Mignotte, M. et C. Schnorr (1988). « Calcul déterministe des racines d'un polynôme dans un corps fini ». In : *Comptes Rendus des Séances de l'Académie des Sciences. Série I. Mathématique*, vol. 306, n°12, p. 467–472.

[13] Moenck, R. T. (1977). « On the efficiency of algorithms for polynomial factoring ». In : *Mathematics of Computation*, vol. 31, p. 235–250.

[14] Mullen, G. L. et D. Panario (2013). *Handbook of Finite Fields*. Discrete Mathematics and Its Applications. Chapman et Hall/CRC.

[15] Rónyai, L. (1989). « Factoring polynomials modulo special primes ». In : *Combinatorica. An International Journal on Combinatorics and the Theory of Computing*, vol. 9, n°2, p. 199–206.

[16] — (1992). « Galois Groups and Factoring Polynomials over Finite Fields ». In : *SIAM Journal on Discrete Mathematics*, vol. 5, n°3, p. 345–365.

[17] Schoof, R. (1985). « Elliptic curves over finite fields and the computation of square roots mod p ». In : *Mathematics of Computation*, vol. 44, n°170, p. 483–494.

[18] Shoup, V. (1990). « On the deterministic complexity of factoring polynomials over finite fields ». In : *Information Processing Letters*, vol. 33, n°5, p. 261–267.

[19] — (1991). « A Fast Deterministic Algorithm for Factoring Polynomials over Finite Fields of Small Characteristic ». In : *ISSAC'91 : International Symposium on Symbolic and Algebraic Computation*. Éd. par S. M. Watt. ACM Press, p. 14–21.

[20] — (2009). *A Computational Introduction to Number Theory and Algebra*. 2e éd. Cambridge University Press.

[21] Zassenhaus, H. (1969). « On Hensel Factorization I ». In : *J. Number Theory*, vol. 1, n°1, p. 291–311.

[22] Źrałek, B. (2010). « Using partial smoothness of $p-1$ for factoring polynomials modulo p ». In : *Mathematics of Computation*, vol. 79, p. 2353–2359.

CHAPITRE 19

Factorisation des polynômes à une variable sur les rationnels

CHAPITRE 20

Factorisation à plusieurs variables

Cinquième partie

Systèmes polynomiaux

CHAPITRE 21

Bases standard

Résumé

Les bases standard de la théorie d'Hironaka pour l'étude de systèmes de fonctions analytiques s'avèrent être un outil central pour rendre effectifs les manipulations de systèmes d'équations polynomiales. Elle sont alors le plus souvent connues sous la dénomination de *bases de Gröbner*.

MOTIVATION 1. Essayons de prouver le théorème d'Apollonius, qui affirme que les milieux des côtés d'un triangle rectangle, le sommet de l'angle droit et le pied de la hauteur relative à l'hypoténuse sont sur un même cercle. Ce n'est qu'une version simplifiée du théorème du cercle des 9 points d'un triangle. Soit donc OAB

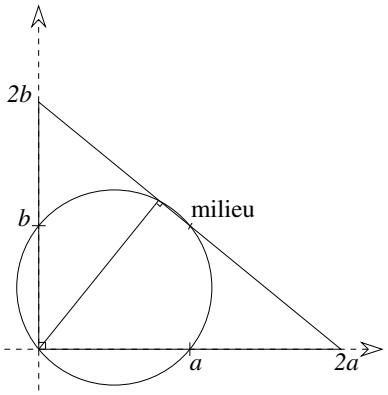


FIGURE 1. Le théorème d'Apollonius

un triangle rectangle de sommet O. Nous pouvons toujours choisir un système de coordonnées tel que l'origine soit en O, et les deux autres sommets les points $(2a, 0)$ et $(0, 2b)$. Le cercle (C) passant par le sommet et les milieux $(a, 0)$ et $(0, b)$ des côtés a pour équation $C(X, Y) = X^2 + Y^2 - aX - bY = 0$. Vérifier que le milieu (a, b) de l'hypoténuse appartient à ce cercle revient à une évaluation, à savoir tester si $C(a, b)$ est nul, ce qui est immédiat. Le pied de la hauteur, $H = (x, y)$, est défini par des relations linéaires exprimant son appartenance à AB et l'orthogonalité entre OH et AB. Il est donc solution du système linéaire $H_1(X, Y) = H_2(X, Y) = 0$, avec $H_1 = aY + bX - 2ab$ et $H_2 = bY - aX$.

Il s'agit donc de démontrer que les hypothèses $H_1(x, y) = H_2(x, y) = 0$ impliquent $C(x, y) = 0$, soit H appartient à (C). Une voie royale consisterait à prouver que le polynôme conclusion C est une combinaison linéaire (à coefficients polynomiaux) de H_1 et H_2, autrement dit qu'il appartient à l'idéal engendré par H_1 et H_2. Est-ce le cas ?

MOTIVATION 2. Dans un autre ordre d'idées, considérons la courbe classique donnée par la paramétrisation

$$x = \frac{1-t^2}{1+t^2} \quad \text{et} \quad y = \frac{2t}{1+t^2},$$

et pour laquelle nous aimerions savoir si elle vérifie une équation indépendante de t. Là encore, nous aimerions une recombinaison algorithmique des polynômes

$$(1+T^2)\,X - (1-T^2) \quad \text{et} \quad (1+T^2)\,Y - 2T$$

avec des coefficients polynomiaux en X, Y et T qui fournisse un polynôme en X et Y seulement. Comment l'obtenir ?

Un anneau de polynômes sur un corps effectif est lui-même effectif d'après le Chapitre 1. Les ennuis commencent en présence d'un idéal de cet anneau, quand il s'agit d'étendre la reconnaissance de zéro à l'anneau quotient, les autres opérations arithmétiques ne posant pas de problème. Autrement dit, comment reconnaître l'appartenance à un idéal d'un anneau de polynômes ? Deux cas particuliers permettent de dégager les concepts qui mènent au traitement général de cette question.

Dans le cas particulier des polynômes à une variable, considérons un idéal donné par un nombre fini de générateurs. L'application successive du vénérable algorithme d'Euclide permet de calculer de proche en proche leur plus grand commun diviseur ; moyennant quoi la question de l'appartenance d'un polynôme candidat à l'idéal en question se résume à sa divisibilité par ledit PGCD, ce que résout une simple division euclidienne, brique de base de l'algorithme d'Euclide.

D'autre part, le non moins ancien et respectable algorithme d'élimination linéaire (bien antérieur à Gauss, voir Chapitre 6) aboutit à la solution si les générateurs sont des formes affines en un nombre quelconque de variables.

Dans ce chapitre, nous allons généraliser à plusieurs variables à la fois l'algorithme d'Euclide et l'algorithme de Gauss, sous le vocable de « bases standard », bien que, dans le cas des algèbres de polynômes, elles soient plus souvent appelées « bases de Gröbner ». La notion de base standard a été introduite par Hironaka pour calculer des générateurs de l'ensemble des termes initiaux des fonctions analytiques au voisinage d'un point [1]. Dans ce cours, nous parlons de base standard pour suggérer une forme de généralisation. Cependant nous resterons dans le cadre polynomial des bases de Gröbner.

Les systèmes d'équations polynomiales sont un outil pour modéliser. La question de l'effectivité des anneaux quotients intervient dans un certain nombre de problèmes d'applications. Par exemple en robotique, la position de la main d'un robot en fonction de la longueur de ses bras est caractérisée par des équations polynomiales. D'autres domaines d'application sont le traitement du signal, la vision par ordinateur, la cryptologie ainsi que la chimie ou la biologie. Pour ces deux derniers domaines, on imagine un système dont l'évolution est régie par un ensemble d'équations différentielles à coefficients polynomiaux. Si on s'intéresse aux états d'équilibres, dans lesquels toutes les dérivées s'annulent, on est donc amené à résoudre un système d'équations polynomiales.

1. Lien entre algèbre et géométrie

La manipulation des systèmes d'équations polynomiales prend tout son sens quand elle donne des informations sur l'ensemble des solutions, à savoir le lieu des zéros communs aux équations polynomiales. Ainsi, un des grands intérêts de la combinatoire des anneaux polynomiaux réside dans la description de son substrat géométrique.

1.1. Le cas affine. Notons n le nombre de variables et $R = k[X_1, \dots, X_n]$ l'anneau de polynômes sur un corps effectif k (le plus souvent $\mathbb{Q}$ mais aussi $\mathbb{R}$ ou $\mathbb{C}$). Notons $\mathbb{A}^n_k$, ou $\mathbb{A}^n$, l'*espace affine* à n dimensions sur k, c'est-à-dire l'ensemble k^n. Posons aussi $f_1, \dots, f_p \in R$ des polynômes.

DÉFINITION 1. *La* variété algébrique $\mathbf{V}_k(J)$ associée à un ensemble de polynômes $J \subseteq R$ *est l'ensemble des racines dans* k *communes aux polynômes de* J, *i.e.*

$$\mathbf{V}_k(J) = \big\{(x_1, \dots, x_n) \in \mathbb{A}^n \mid \forall f \in J, f(x_1, \dots, x_n) = 0\big\}.$$

S'il n'y a pas d'ambiguïté sur le corps k, *nous noterons* $\mathbf{V}(J) := \mathbf{V}_k(J)$.

Pour ne pas alourdir la formulation, on appelle « variété algébrique » tout sous-ensemble de $\mathbb{A}^n$ obtenu de la manière précédente, bien que ce cas ne soit qu'un cas particulier de la définition générale en géométrie algébrique.

DÉFINITION 2. *Un* idéal I *d'un anneau* R *est un sous-groupe de* R *stable par multiplication par les éléments de* R. *C'est-à-dire que* $I \neq \varnothing$ *et* $\forall f, g \in I,\ f - g \in I$ *et* $\forall f \in I, \forall h \in R,\ hf \in I$.

L'idéal $(f_1, \dots, f_s)$ *engendré par une partie* $\{f_1, \dots, f_s\}$ *de* R *est l'ensemble* $\sum_{i=1}^{s} Rf_i$.

Le résultat suivant découle de ce que l'opération $\mathbf{V}$ met en place une sorte de dualité.

PROPOSITION 1. *Pour deux ensembles* J_1 *et* J_2 *de polynômes, si* $J_1 \subseteq J_2$, *alors* $\mathbf{V}(J_1) \supseteq \mathbf{V}(J_2)$.

Remarquons que l'on a $\mathbf{V}\big(\{f_1, \dots, f_s\}\big) = \mathbf{V}(I)$ avec $I = (f_1, \dots, f_s)$, et que si deux ensembles de polynômes engendrent le même idéal alors ils définissent la même variété. Ainsi l'idéal $(f_1, \dots, f_s)$ est l'objet naturel à associer au système d'équations $f_1 = 0, \dots, f_s = 0$. De même que nous pouvons associer à un idéal sa variété, nous pouvons faire l'opération inverse.

DÉFINITION 3. *L'idéal* $\mathbf{I}(E)$ *associé à un sous-ensemble* E *de* $\mathbb{A}^n$ *est l'ensemble des polynômes de* R *s'annulant identiquement sur* E, *c'est-à-dire*

$$\mathbf{I}_k(E) = \{f \in R \mid \forall x \in E, f(x) = 0\}.$$

Si il n'y a pas d'ambiguïté sur le corps k, *nous noterons* $\mathbf{I}(E) := \mathbf{I}_k(E)$.

On vérifie facilement que $\mathbf{I}(E)$ est un idéal.

EXEMPLE 1. L'hyperbole d'équation $XY = 1$ est la variété $\mathbf{V}(XY - 1)$. De même, le cercle d'équation $X^2 + Y^2 = 1$ est une variété.

EXEMPLE 2. La variété réelle $\mathbf{V}_{\mathbb{R}}(X^2 + Y^2 + 1)$ est vide. Cependant la variété complexe $\mathbf{V}_{\mathbb{C}}(X^2 + Y^2 + 1)$ est non vide.

La dualité évoquée plus haut se prolonge avec les résultats suivants.

PROPOSITION 2. *Pour deux sous-ensembles* E_1 *et* E_2 *de* $\mathbb{A}^n$, *si* $E_1 \subseteq E_2$, *alors* $\mathbf{I}(E_1) \supseteq \mathbf{I}(E_2)$.

Si J *est un ensemble de polynômes et* E *une partie de* $\mathbb{A}^n$, *alors* $J \subseteq \mathbf{I}\big(\mathbf{V}(J)\big)$ *et* $E \subseteq \mathbf{V}\big(\mathbf{I}(E)\big)$.

EXEMPLE 3. Si $R = k[X]$ et $J = (X^2)$ alors $\mathbf{V}(J) = \mathbf{V}((X))$ et $\mathbf{I}\big(\mathbf{V}(J)\big) = (X)$ n'est pas égal à J.

EXEMPLE 4. Si $R = \mathbb{Q}[X, Y]$ et $E = (k \setminus \{0\}) \times \{0\}$, alors $\mathbf{I}(E) = (Y)$ et $\mathbf{V}(\mathbf{I}(E)) = k \times \{0\}$ n'est pas égal à E.

Les variétés algébriques sont des objets géométriques et il est fructueux de les manipuler en suivant une intuition géométrique. La démonstration algébrique fonctionne alors (souvent) comme on le pense.

Exercice 1. Montrer que l'union de deux variétés algébriques est une variété algébrique. Montrer que $V \subseteq \mathbb{A}^n$ est une variété algébrique si, et seulement si, $\mathbf{V}(\mathbf{I}(V)) = V$.

1.2. Le cas projectif. Le passage à la géométrie projective est motivé par la simplification des énoncés et de certains algorithmes dans ce cadre. Posons $R = k[X_0, \ldots, X_n]$.

Définition 4. *Nous notons $\mathbb{P}^n_k$, ou $\mathbb{P}^n$, l'espace projectif de dimension n sur le corps k, c'est-à-dire l'ensemble défini comme le quotient de $k^{n+1} \setminus \{0\}$ par la relation d'équivalence $\sim$ elle-même définie par $(x_0, \ldots, x_n) \sim (y_0, \ldots, y_n)$ si $\exists \lambda \in k,\ \forall i,\ x_i = \lambda y_i$.*

On note $(x_0 : \ldots : x_n)$ la classe d'équivalence de $(x_0, \ldots, x_n)$, que l'on appelle aussi un point (projectif).

L'espace projectif $\mathbb{P}^n$ est une variété algébrique d'une nature plus générale que les $\mathbf{V}(J)$ de la définition 1. Précisons ce que cela signifie pour nous. Posons $U_i = \big\{(a_0 : \ldots : a_n) \mid a_i \neq 0\big\}$ pour $i \in \{0, \ldots, n\}$. Les U_i sont des ouverts de $\mathbb{P}^n$ qui sont en bijection avec l'espace affine $\mathbb{A}^n$ via

$$\begin{array}{ccc} \mathbb{A}^n & \to & U_i \\ (x_1, \ldots, x_n) & \mapsto & (x_0 : \ldots : x_{i-1} : 1 : x_{i+1} : \ldots : x_n) \end{array}.$$

Ces U_i recouvrent l'espace projectif. Nous appellerons U_i la *i-ème partie affine* de $\mathbb{P}^n$.

Définition 5. *On appelle* points à l'infini *de $\mathbb{P}^n$ les points de $\mathbb{P}^n \setminus U_0$. Ils correspondent aux directions de droites dans $\mathbb{A}^n$.*

Le formalisme ne doit pas faire oublier que l'espace projectif est un objet naturel : c'est l'espace affine auquel on a ajouté des points à l'infini. Ces points à l'infini ne sont autres que les points $(0 : x_1 : \ldots : x_n)$, alors que les points affines sont de la forme $(1 : x_1 : \ldots : x_n)$. On a donc

$$\mathbb{P}^n = \big\{(1 : x_1 : \ldots : x_n) \;\big|\; x \in \mathbb{A}^n\big\} \cup \big\{(0 : x_1 : \ldots : x_n) \;\big|\; x \in \mathbb{A}^n \setminus \{0\}\big\}.$$

Dans un espace projectif, construire des équations à partir de polynômes est plus délicat que dans le cas affine. Soit $f \in R$ un polynôme homogène, c'est-à-dire tel que tous ses monômes soient de même degré total. Alors $f(x_0 : \ldots : x_n) = 0$ a un sens car f s'annule soit sur tout représentant de $(x_0 : \ldots : x_n)$, soit sur aucun d'entre eux. Ainsi nous pouvons associer à tout polynôme homogène f de R une équation $f = 0$ sur $\mathbb{P}^n$. Notons bien que R a $n+1$ variables, où n est la dimension de l'espace projectif.

Définition 6. *La* variété projective $\mathbf{V}(J)$ associée à l'ensemble de polynômes homogènes $J \subseteq R$ *est l'ensemble des solutions communes aux équations homogènes, c'est-à-dire*

$$\big\{(x_0 : \ldots : x_n) \in \mathbb{P}^n_k \mid \forall j \in J,\ j(x_0 : \ldots : x_n) = 0\big\}.$$

Le bon objet à associer à un système d'équations homogènes est un idéal homogène.

Définition 7. *Un idéal de R est dit* homogène *s'il est engendré par des polynômes homogènes.*

PROPOSITION 1. *Un idéal I de R est homogène si, et seulement si, l'espace vectoriel I est la somme directe $\bigoplus_{d\geq 0} I_d$ où I_d est l'espace vectoriel des polynômes de I homogènes de degré d.*

DÉMONSTRATION. Supposons que I soit engendré par des polynômes homogènes $f_1, \ldots, f_s$ et soit h un polynôme de I. Il existe alors des polynômes $g_1, \ldots, g_s$ tels que $h = g_1 f_1 + \cdots + g_s f_s$. En notant $g_{i,d}$ la composante homogène de degré d de g_i, la précédente égalité se réécrit

$$h = \sum_{d\geq 0} g_{1,d} f_1 + \cdots + \sum_{d\geq 0} g_{s,d} f_s,$$

et la composante homogène h_d de h s'obtient alors comme

$$h_d = g_{1,d-deg f_1} f_1 + \cdots + g_{s,d-deg f_s} f_s.$$

Autrement dit, les composantes homogènes de h sont aussi dans I, d'où $I = \bigoplus_{d\geq 0} I_d$.

Réciproquement, si $I = \bigoplus_{d\geq 0} I_d$, et si I est engendré par les polynômes $f_1, \ldots, f_s$ alors les composantes homogènes des f_i sont dans I et par conséquent l'engendrent. □

L'homogénéisation des polynômes et des idéaux permet de plonger naturellement une variété affine dans l'espace projectif.

DÉFINITION 8. *L'*homogénéisé $f^\sharp \in R$ *d'un polynôme $f \in k[X_1, \ldots, X_n]$ est le polynôme $X_0^{\deg(f)} f(X_1/X_0, \ldots, X_n/X_0)$.*

*L'*idéal homogénéisé $I^\sharp$ *d'un idéal I de $k[X_1, \ldots, X_n]$ est l'idéal de R engendré par $\{f^\sharp \mid f \in I\}$.*

EXERCICE 2. Montrer que le plongement du cercle $\mathbf{V}_{\mathbb{C}}(X^2 + Y^2 - 1)$ dans l'espace projectif $\mathbb{P}^2_{\mathbb{C}}$ contient deux points à l'infini.

L'opération de déshomogénéisation associe au polynôme $g \in k[X_0, \ldots, X_n]$ le polynôme $g^\flat = g(1, X_1, \ldots, X_n) \in k[X_1, \ldots, X_n]$. On vérifie facilement que pour $g \in k[X_1, \ldots, X_n]$ on a $(g^\sharp)^\flat = g$. Cependant, la composition inverse n'est pas l'identité : $(X_0^\flat)^\sharp = 1$. La proposition suivante fait le lien entre $(g^\flat)^\sharp$ et g.

PROPOSITION 2. *Soit $g \in k[X_0, \ldots, X_n]$ homogène, alors il existe $e \in \mathbb{N}$ tel que $X_0^e (g^\flat)^\sharp = g$.*

DÉMONSTRATION. Soit $d = \deg g$. Voyons g comme un polynôme en X_0 et notons e sa valuation en X_0 : g s'écrit $g = \sum_{i=0}^{d-e} g_{d-e-i} X_0^{e+i}$ avec $g_i \in k[X_1, \ldots, X_n]$ de degré i. Ainsi $g^\flat = \sum_{i=0}^{d-e} g_{d-e-i}$ et $d = \deg g^\flat + e$. Finalement $X_0^e (g^\flat)^\sharp = X_0^e \sum_{i=0}^{d-e} g_{d-e-i} X_0^i = g$. □

2. Ordres totaux admissibles sur le monoïde des monômes

Notons $R = k[X_1, \ldots, X_n]$. À un élément $a = (a_1, \ldots, a_n)$ de $\mathbb{N}^n$ est associé le monôme ou produit de puissances $X = X_1^{a_1} \cdots X_n^{a_n} \in R$. Ainsi les monômes de l'anneau R forment un monoïde multiplicatif $\mathcal{M}_n$ isomorphe au monoïde additif $\mathbb{N}^n$. Rappelons qu'un monoïde $\mathcal{M}$ vérifie les axiomes des groupes à l'exception de l'existence de l'inverse pour un élément quelconque. Dans notre exemple, l'élément neutre $1 \in \mathcal{M}_n$ correspond au point de coordonnées toutes nulles dans $\mathbb{N}^n$. Nous nous autoriserons à utiliser indifféremment la notation additive ou multiplicative.

DÉFINITION 9. *Un ordre total $<$ sur les monômes est dit* compatible *s'il est compatible avec la structure de monoïde, au sens où la multiplication par un monôme est croissante.*

Définition 10. *Un ordre compatible est dit être un* ordre admissible *si l'élément neutre est plus petit que tous les autres monômes.*

Définition 11. *Un ordre compatible est dit être un* bon ordre *si toute suite décroissante de monômes stationne. En résumé, un ordre $<$ sur les monômes est un bon ordre si :*

1. *c'est un ordre total ;*
2. *pour tout m, m' et m'', $m' < m''$ implique $mm' < mm''$.*
3. *pour toute suite décroissante de monômes $(m_i)_{i\in\mathbb{N}}$, il existe un rang N tel que pour tout $n \geq N$, on ait $m_n = m_N$.*

Un bon ordre est toujours admissible car on peut construire une suite strictement décroissante à partir des puissances d'un monôme $m < 1$. Nous verrons au Corollaire 1 qu'un ordre admissible est un bon ordre.

Exemple 5. Un exemple est l'ordre fondamental $<_{\text{lex}}$: pour ordonner deux points différents de $\mathbb{N}^n$, on regarde la première coordonnée qui diffère. En fait, on devrait plutôt parler de l'ordre lexicographique induit par un ordre total sur les variables. En Maple, on écrit `plex(x[1], ..., x[n])` pour l'ordre lexicographique avec $X_1 >_{\text{lex}} X_2 >_{\text{lex}} \cdots >_{\text{lex}} X_n$. Intuitivement c'est ainsi que l'ordre des lettres de l'alphabet induit un ordre sur les mots du dictionnaire avec $a <_{\text{lex}} b <_{\text{lex}} c <_{\text{lex}} \ldots$

Lemme 1. *L'ordre lexicographique est un bon ordre.*

Démonstration. L'ordre lexicographique est facilement compatible. Montrons la propriété de bon ordre par récurrence sur le nombre n de variables. Ceci est trivial pour $n = 1$. Soit une suite $(a_j)_{j\in\mathbb{N}}$ décroissante pour $<_{\text{lex}}$ dans $\mathbb{N}^n$. Écrivons chaque a_j sous la forme (α_j, b_j) pour $\alpha_j \in \mathbb{N}$ et $b_j \in \mathbb{N}^{n-1}$. La suite des α_j est alors décroissante et stationne, par le cas $n = 1$. La suite des b_j est donc décroissante à partir d'un certain rang, et donc stationne. En conclusion, la suite $(a_j)_{j\in\mathbb{N}}$ stationne elle aussi. □

Exemple 6. Il existe aussi pour les rimailleurs des dictionnaires de rimes : les mots sont ordonnés d'après les dernières lettres ; et pour les amateurs de *Scrabble*, des listes de mots rangés par longueur croissante. On peut lier ces dictionnaires à des ordres sur les monômes.

Exemple 7. L'ordre diagonal induit par un ordre compatible $<$ est l'ordre noté $a <_d b$ pour $a = (a_1, \ldots, a_n)$ et $b = (b_1, \ldots, b_n)$ si $\sum a_i < \sum b_i$ ou $\big(\sum a_i = \sum b_i$ et $a < b\big)$. En Maple, nous avons `grlex(x[1], ..., x[n])` (« graded lexicographic order ») pour l'ordre diagonal induit par l'ordre lexicographique classique. Une autre terminologie pour les ordres diagonaux est celle d'« ordres gradués par le degré ».

Exemple 8. Un autre ordre usuel est l'ordre gradué lexicographique inverse (`tdeg(x[1], ..., x[n])` en Maple). Soit $<_{\text{lex}'}$ l'ordre lexicographique inverse induit par $X_n > \cdots > X_1$. L'ordre $<_r$ est défini comme l'ordre diagonal induit par le renversement de l'ordre $<_{\text{lex}'}$. Ainsi, on a $a <_r b$ pour $a = (a_1, \ldots, a_n)$ et $b = (b_1, \ldots, b_n)$ si $\sum a_i < \sum b_i$ ou $\big(\sum a_i = \sum b_i$ et $a >_{\text{lex}'} b\big)$. Cela consiste donc à ordonner par degré total puis à prendre le moins de X_n puis de X_{n-1} et ainsi de suite. Notons que la notation Maple est justifiée par $X_1 >_r \cdots >_r X_n$.

Remarquons que la propriété de bon ordre est triviale pour les ordres diagonaux, par finitude du nombre de monômes de même degré. Comparons maintenant les ordres qui viennent d'être introduits sur quelques exemples qui permettent de les

distinguer. Commençons par les ordres $<_{\text{lex}}$ et $<_{\text{lex}'}$ induits par $X_1 > X_2 > X_3 > X_4$. Nous avons alors :

$$X_1 >_{\text{lex}} X_2X_4 >_{\text{lex}} X_3 \quad \text{et} \quad X_1 <_{\text{lex}'} X_3 <_{\text{lex}'} X_2X_4.$$

Pour observer une différence entre l'ordre gradué lexicographique inverse $<_t :=$ `tdeg(X, Y, Z)` et l'ordre $<_g :=$ `grlex(X, Y, Z)`, tous deux pour $X > Y > Z$, il convient d'aller jusqu'aux monômes de degré 3. Ainsi,

$$X^3 >_t X^2Y >_t XY^2 >_t Y^3 >_t X^2Z >_t XYZ >_t Y^2Z >_t XZ^2 >_t YZ^2 >_t Z^3$$

alors que

$$X^3 >_g X^2Y >_g X^2Z >_g XY^2 >_g XYZ >_g XZ^2 >_g Y^3 >_g Y^2Z >_g YZ^2 >_g Z^3.$$

Nous raccourcirons par la suite en `plex`, `grlex`, `tdeg` les ordres précédents quand ils sont induits par $X_1 > \cdots > X_n$.

Exemple 9 (Ordre d'élimination). Soit $i \in \{1, \ldots, n\}$ et $<_{\text{lex}}$ l'ordre lexicographique avec $x_1 > \cdots > x_n$. Le i^{e} ordre d'élimination $<_i$ de $\mathbb{N}^n$ est l'ordre défini par $a <_i b$ si $\sum_{j=1}^{i} a_j < \sum_{j=1}^{i} b_j$ ou $\left(\sum_{j=1}^{i} a_j = \sum_{j=1}^{i} b_j \text{ et } a <_{\text{lex}} b\right)$.

On retrouve pour $i = 1$ l'ordre lexicographique usuel et pour $i = n$ l'ordre lexicographique gradué.

Exemple 10. Une dernière méthode pour créer des ordres compatibles à partir de l'ordre lexicographique est de prendre une matrice $M \in \mathcal{M}_{m\times n}(\mathbb{R})$ et de poser $a <_M b$ si $Ma <_{\text{lex}} Mb$. Si M est injective alors l'ordre est total. L'hypothèse supplémentaire que toutes les colonnes C de M vérifient $0 <_{\text{lex}} C$ est nécessaire pour que l'ordre devienne admissible. Ce changement d'ordre permet toujours de se ramener à l'ordre lexicographique via le théorème suivant.

Théorème 1 (Robbiano 1985). *Soit $<$ un ordre total compatible sur $\mathbb{N}$. Alors il existe $M \in \mathcal{M}_{m\times n}(\mathbb{R})$ telle que $a < b \iff Ma <_{lex} Mb$.*

Exercice 3. Trouver les matrices du théorème précédent pour les ordres présentés précédemment.

3. Exposants privilégiés et escaliers

Définition 12. *À tout polynôme homogène non nul :*

$$f = \sum_{a\in\mathbb{N}^{n+1}} f_a\, X^a$$

est associé son support $\{a \in \mathbb{N}^{n+1} \mid f_a \neq 0\}$ *dans* $\mathbb{N}^{n+1}$.

Définition 13. *L'*exposant privilégié *d'un polynôme f* non nul*, noté* $\exp(f)$ *pour un ordre donné $<$ est le plus grand $(n+1)$-uplet du support de f pour l'ordre $<$. Le* monôme de tête *est* $\operatorname{lm}(f) := X^{\exp(f)}$. *Le* terme de tête *de f est* $\operatorname{lt}(f) := f_{\exp(f)}X^{\exp(f)}$*, sachant qu'un terme est le produit d'un monôme par un coefficient, c'est-à-dire est de la forme $c_\alpha X^\alpha$. Le coefficient du terme de terme est noté* $\operatorname{lc}(f)$.

Notons bien que la notion d'exposant privilégié et celles qui en découlent ne sont pas définies pour le polynôme nul.

Définition 14. *Un* idéal de monômes *est une partie de $\mathcal{M}_n$ stable par multiplication externe : tout multiple d'un élément de cet idéal de monômes par un monôme quelconque appartient encore à cet idéal de monômes.*

De manière analogue, une partie stable *de $\mathbb{N}^n$ est stable par addition de quadrant : tout translaté d'un point de cette partie stable par un élément de $\mathbb{N}^n$ est encore dans cette partie stable.*

On fera attention à cette terminologie : au sens de ce cours, un idéal de monômes n'est pas un idéal. Certains auteurs parlent d'ailleurs de « monoïdéal » pour marquer la distinction ; d'autres auteurs réservent le vocable d'« idéal de monômes » pour les idéaux (polynomiaux) engendrés par les idéaux de monômes de ce cours.

Plutôt que d'idéal de monômes, on parle le plus souvent de manière imagée d'un *escalier*.

DÉFINITION 15. *L'*escalier $E(I)$ *associé à l'idéal* I *pour un ordre* $<$ *est la partie stable de* $\mathbb{N}^n$ *donnée par* $\{\exp_<(f) \mid f \in I\}$.

Par convention, et bien que le polynôme nul ne possède pas d'exposant privilégié, $E(0)$ est l'ensemble vide $\varnothing$; tandis que plus naturellement $E(R)$ est $\mathbb{N}^n$ tout entier puisque $R = (1)$.

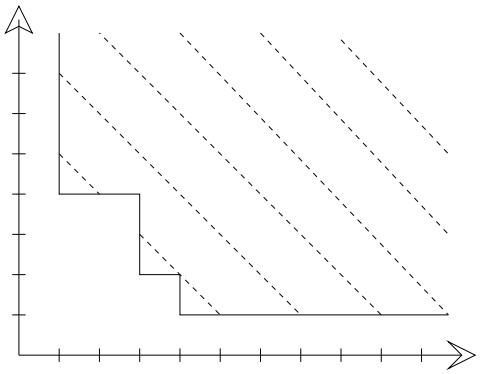


FIGURE 2. Un exemple d'escalier dans $\mathbb{N}^2$

4. Noethérianité du monoïde des monômes

PROPOSITION 3. *Les propriétés suivantes sont équivalentes :*

1. *Toute partie stable de* $\mathbb{N}^n$ *est engendrée par un nombre fini d'éléments, c'est-à-dire :*

$$E = \bigcup_{i=1}^{q} \big(a_i + \mathbb{N}^n\big); \tag{1}$$

2. *Toute suite croissante de parties stables de* $\mathbb{N}^n$ *stationne.*

DÉMONSTRATION. Montrons que (1) implique (2). Soit $(E_j)_{j\in\mathbb{N}}$, une suite croissante de parties stables. La réunion $E = \bigcup_{j\in\mathbb{N}} E_j$ est encore stable, donc finiment engendrée sous la forme (1). Chacun des générateurs a_i appartient à un membre E_{j_i} de la suite. Considérons la valeur $\hat{j}$ maximale des j_i. Comme $E_{\hat{j}}$ contient tous les E_j précédents, $E = \bigcup_{j=1}^{q}(a_i + \mathbb{N}^n) \subseteq \sum_{j=1}^{\hat{j}} E_j \subseteq E_{\hat{j}} \subseteq E$, d'où $E = E_{\hat{j}}$. Par suite, pour tout $h \in \mathbb{N}$, $E = E_{\hat{j}} \subseteq E_{\hat{j}+h} \subseteq E$, et la suite stationne donc sur la valeur $E = E_{\hat{j}}$.

Montrons maintenant que (2) implique (1). Supposons que E soit engendré par une suite dénombrable $a_1, a_2, a_3, \ldots$ On pose $E_i := \cup_{j=1}^{i}(a_j + \mathbb{N}^n)$, pour tout $i \geq 1$. La suite des E_i est croissante et donc stationne à partir d'un certain entier $\hat{i}$. On en déduit $E = \cup_{j\geq 1} E_j = \cup_{j=1}^{\hat{i}} E_j$. □

LEMME 2 (Lemme de Dickson). *Toute partie stable E de $\mathbb{N}^n$ est finiment engendrée, c'est-à-dire qu'il existe une famille $a_1, \ldots, a_q$ telle que :*

$$E = \bigcup_{i=1}^{q} (a_i + \mathbb{N}^n).$$

DÉMONSTRATION. La démonstration se fait par récurrence sur n. L'assertion est immédiate pour $n = 1$. Supposons l'assertion vraie pour $n \geq 1$. On introduit $\pi : \mathbb{N}^{n+1} \to \mathbb{N}^n$ la projection canonique sur $\mathbb{N}^n$, identifié à $\mathbb{N}^n \times \{0\}$. Chacune des sections E_j de E par l'hyperplan de coordonnées $X_{n+1} = j$ s'identifie aussi par π à une partie stable de $\mathbb{N}^n$, engendrée par une famille finie, par récurrence. Les $\pi(E_j)$ forment une suite croissante de parties stables qui stationne sur une valeur E_∞ à partir d'un certain indice η, par la proposition précédente. Une famille de générateurs de E est obtenue en prenant la réunion des générateurs des E_j pour $j \leq \eta$. □

La valeur E_∞ de la démonstration précédente est appelée *section à l'infini*. Cette notion interviendra de nouveau dans le chapitre suivant.

COROLLAIRE 1. *Tout ordre total admissible est un* bon ordre*, c'est-à-dire que toute chaîne descendante (c'est-à-dire une suite décroissante) stationne.*

DÉMONSTRATION. Soit une suite $(a_j)_{j\in\mathbb{N}}$ décroissante de $\mathbb{N}^n$ pour un ordre $<$ admissible vu dans le monoïde des exposants. Posons $E_i := \cup_{j=0}^{i}(a_j + \mathbb{N}^n)$ pour tout $i \geq 0$. La suite des E_i est croissante et donc stationne à partir d'un indice N, par le lemme de Dickson. Si $k \geq N$ alors a_k appartient à l'un des $a_j + \mathbb{N}^n$ pour un indice $j \leq N$ et donc $a_k \geq a_j \geq a_N$. On en déduit $a_k = a_N$. □

DÉFINITION 16. *Une* base standard *d'un idéal de l'anneau de polynômes est un ensemble fini de polynômes de l'idéal dont les exposants privilégiés engendrent l'escalier de l'idéal.*

Le lemme de Dickson nous donne l'existence d'une base standard pour tout idéal I de R. Notons bien que les éléments doivent appartenir à l'idéal mais que rien pour l'instant hormis la terminologie ne prouve qu'ils l'engendrent. C'est le paragraphe suivant qui va nous le démontrer, grâce à une généralisation à plusieurs variables de la notion de division.

Notons aussi qu'une base standard ne sera pas une base, mais seulement un système de générateurs : elle ne fournira pas une unicité de l'écriture, au sens où les cofacteurs ne sont pas définis uniquement.

5. Divisions

Dans l'algorithme de division d'Euclide, une étape élémentaire consiste à tuer le terme de plus haut degré du dividende par le terme de plus haut degré du diviseur. Nous pouvons étendre ce principe aux polynômes à plusieurs variables une fois choisi un ordre admissible $<$ sur les monômes.

Pour ce faire, commençons par définir une *division élémentaire faible* (resp. *forte*) d'un polynôme dividende f par un seul diviseur g non nul. Posons $r := \mathrm{lt}(g) - g$ et associons à notre diviseur la règle de réécriture $\mathrm{lt}(g) \to_g r$. Elle consiste à remplacer une occurrence éventuelle de $\mathrm{lt}(g)$ comme facteur du monôme privilégié (resp. de tout monôme) du dividende par le polynôme r.

Pour être plus explicite, soit un dividende (non nul) exprimé sous la forme $f = c_1 m_1 + \cdots + c_\ell m_\ell$ pour une suite strictement décroissante de monômes m_i et des coefficients non nuls c_i. Si $\mathrm{lm}(g)$ divise m_1 pour le cas de la division faible (resp.

s'il divise l'un des m_i, que l'on choisit être le plus grand, pour le cas de la division forte), alors la division consiste à réécrire f en le polynôme

$$f - \frac{\mathrm{lt}(f)}{\mathrm{lt}(g)}g = \frac{\mathrm{lt}(f)}{\mathrm{lt}(g)}r + c_2m_2 + \cdots + c_\ell m_\ell \tag{2}$$

dans le cas de la division faible (resp. en le polynôme

$$f - \frac{c_im_i}{\mathrm{lt}(g)}g = c_1m_1 + \cdots + c_{i-1}m_{i-1} + \frac{c_im_i}{\mathrm{lt}(g)}r + c_{i+1}m_{i+1} + c_\ell m_\ell \tag{3}$$

dans le cas de la division forte). Dans toutes ces écritures, chaque quotient est en fait un terme.

La division faible consiste à appliquer cette règle de réécriture à $\mathrm{lm}(f)$ (si possible) et à itérer la réécriture au nouveau polynôme tant que possible. La division forte consiste à appliquer la règle de réécriture à tout terme de f et de s'arrêter quand on ne peut plus l'appliquer à aucun terme.

L'itération du processus de division faible (resp. forte) s'arrête (car $<$ est un bon ordre par le corollaire 1) sur un reste dont le monôme privilégié n'est plus divisible par $\mathrm{lm}(g)$ (resp. dont aucun monôme n'est divisible par $\mathrm{lm}(g)$).

La division faible (resp. forte) par une famille $g_1, \ldots, g_s$ consiste à appliquer successivement l'une des règles de réécriture $\mathrm{lt}(g_i) \to_{g_i} r_i, \quad i \in \{1, \ldots, s\}$, à un dividende f jusqu'à ce qu'on ne puisse plus l'appliquer au terme de tête $\mathrm{lt}(f)$ de f (resp. à aucun terme de f).

Notons qu'en rassemblant de façon adéquate les monômes $\mathrm{lt}(f)/\mathrm{lt}(g_j)$ intervenant dans les étapes (2) d'une division faible (resp. les monômes $c_im_i/\mathrm{lt}(g_j)$ dans les étapes (3) d'une division forte), on obtient des quotients q_j, un reste final r, ainsi qu'une écriture de la division sous la forme

$$f = q_1g_1 + \cdots + q_sg_s + r.$$

Cette sortie du processus de division (faible ou forte) dépend de l'ordre dans lequel sont faites les réécritures, ce que montre l'exemple suivant.

Exemple 11. Soient les diviseurs $g_1 = YX - 1$, $g_2 = X^2$ et le dividende $f = YX^2 - X$. Pour l'ordre lexicographique avec $X < Y$, on observe d'un côté $f \to_{g_1} 0$ et de l'autre $f \to_{g_2} -X$ et on ne peut plus récrire X. Ceci signifie que les règles de réécriture ne sont pas complètes (au sens de la théorie de la réécriture telle que proposée par [2]). Il convient de rajouter $X \to 0$ pour (tenter de) compléter le jeu de règles.

Cependant, les bases standard jouissent de propriétés privilégiées vis à vis de la réécriture ; on a les résultats suivants.

Proposition 4. *Si un reste par une division, faible ou forte, d'un dividende f quelconque par une base standard $(g_1, \ldots, g_s)$ est nul, le reste par toute autre division faible ou forte l'est aussi. De plus, l'exposant privilégié du reste par division ne dépend ni de l'ordre des réécritures ni de la nature faible ou forte de la division.*

Démonstration. Posons $I = (g_1, \ldots, g_s)$. Le procédé de réécriture faible ou forte change de représentant mais pas de classe modulo I. Soient r et r' deux restes de divisions faibles ou fortes par $(g_1, \ldots, g_s)$. On a $f \equiv r$ et $f \equiv r'$ modulo I, donc $r - r' \in I$. Si l'un des restes est nul, disons r', alors l'autre, r, est dans I. Comme il n'est pas réductible, il est lui aussi nul, ce qui prouve la première assertion.

Dans le cas contraire, $rr' \neq 0$. Si $\mathrm{lm}(r) \neq \mathrm{lm}(r')$, on peut supposer $\mathrm{lm}(r) > \mathrm{lm}(r')$ auquel cas $\mathrm{lm}(r) = \mathrm{lm}(r - r') \in E(I)$. Ceci est absurde car on peut alors encore réduire r. Ainsi, r et r' ont même exposant privilégié. □

Proposition 5. *Le reste de toute division, faible ou forte, d'un élément f d'un idéal par une base standard de celui-ci est nul.*

DÉMONSTRATION. Le reste est dans l'idéal. S'il n'est pas nul, son monôme de tête est multiple d'un des monômes de tête de la base standard, par définition, ce qui contredit que ce soit un reste. □

PROPOSITION 6. *Le reste d'une division forte d'un dividende f par une base standard $(g_1, \ldots, g_s)$ est unique.*

DÉMONSTRATION. Avec les mêmes notations que dans la preuve précédente, si $r - r' \neq 0$ alors $\mathrm{lm}(r - r') \in E(I)$ provient de l'un des termes de r ou r'. Ceci est absurde car aucun monôme de r ni r' n'est dans $E(I)$ sinon on pourrait le récrire. Donc $r = r'$, ce qui prouve la proposition. □

COROLLAIRE 2 (Théorème de division d'Hironaka). *Soient I un idéal de R, $>$ un ordre admissible et $E(I)$ l'escalier de I pour cet ordre. Tout polynôme de R est congru modulo I à un unique polynôme appelé reste de la division par l'idéal qui soit est nul, soit a son support à l'extérieur à $E(I)$, c'est-à-dire (strictement) sous l'escalier.*

DÉMONSTRATION. L'opération de division forte par une base standard donne l'existence d'un tel reste. L'unicité est similaire à la preuve de la proposition précédente. □

COROLLAIRE 3 (Noethérianité de l'anneau des polynômes). *Toute base standard d'un idéal engendre cet idéal.*

DÉMONSTRATION. Le reste de la division d'un élément f de l'idéal par une base standard $(g_1, \ldots, g_s)$ ne peut être que nul puisque sinon, son monôme dominant devrait être à la fois à l'extérieur de $E(I)$ (par construction du reste) et dans $E(I)$ (par définition de ce dernier). Il existe donc des quotients q_1, …, q_s tels que $f = q_1 g_1 + \cdots + q_s g_s$, et f appartient donc à l'idéal engendré par la base standard. □

Ce corollaire établit la noethérianité de l'anneau de polynômes.

COROLLAIRE 4 (Décomposition du quotient). *En tant que k-espace vectoriel, on a la décomposition $R = I \oplus R/I$ où le quotient R/I est isomorphe au sous-espace vectoriel de R donné comme la somme directe*

$$R/I = \bigoplus_{a \notin E(I)} kX^a.$$

Nous avons vu dans ce chapitre que pour effectuer le test à zéro effectivement, il nous suffit de disposer d'une base standard. Le calcul d'une telle base est le sujet du chapitre suivant.

Bibliographie

[1] HIRONAKA, Heisuke (1964). « Resolution of singularities of an algebraic variety over a field of characteristic zero. I, II ». In : *Annals of Mathematics. Second Series*, vol. 79, p. 205–326.

[2] KNUTH, D. et P. BENDIX (1970). « Computational problems in abstract algebra ». In : éd. par John LEECH. Pergamon Press. Chap. Simple word problems in universal algebras, p. 263–297.

CHAPITRE 22

Construction de bases standard

Résumé

Construire une base standard d'un idéal revient à trouver des générateurs de l'idéal monomial de tête de cet idéal. L'étude des relations entre générateurs d'une présentation de l'idéal donne la manière de compléter la présentation de l'idéal monomial correspondant, et par relèvement de compléter le système de générateurs de l'idéal en une base standard.

1. Algorithme naïf par algèbre linéaire

Dans le cadre homogène, l'algorithme naïf que nous allons introduire revient à faire de l'algèbre linéaire en se ramenant à des espaces vectoriels de dimension finie.

DÉFINITION 1. *Un anneau R est gradué s'il existe des sous-groupes R_d de $(R,+)$ tels que $R=\bigoplus_{d\in\mathbb{N}} R_d$ et $R_dR_{d'}\subset R_{d+d'}$.*

L'anneau $R=k[X_0,\dots,X_n]$ est muni d'une graduation $R=\bigoplus_{d\in\mathbb{N}} R_d$ où R_d est l'ensemble des polynômes homogènes de degré d. Notons que chaque R_d est un k-espace vectoriel de dimension finie. Soit $I=(f_1,\dots,f_s)$ un idéal homogène. Par la Proposition 1 du Chapitre 21, l'anneau R/I est gradué en posant $(R/I)_d=R_d/I_d$ pour d entier.

Notons d_i le degré de g_i. Pour tout $u\in\mathbb{N}$, introduisons l'application

$$\phi_u : \left\{ \begin{array}{ccc} R_{u-d_1}\times\cdots\times R_{u-d_s} & \to & R_u \\ (g_1,\dots,g_s) & \mapsto & \sum_i g_i f_i \end{array} \right.$$

d'image I_u. Adoptons un ordre admissible $<$ et, pour tout v, posons B_v la base de R_v formée des monômes homogènes de degré v ordonnés par $<$. Identifions chaque R_v à un espace vectoriel de vecteurs colonnes. Les colonnes de la matrice de ϕ_u dans ces bases engendrent alors I_u comme k-espace vectoriel. Par opérations élémentaires sur les colonnes de la matrice, on peut la mettre sous forme échelonnée (voir Chapitre 8). Les colonnes de la nouvelle matrice engendrent encore I_u. Les premiers coefficients non nuls des colonnes donnent les monômes de tête des polynômes de I_u.

Par noethérianité, la réunion des colonnes obtenues en itérant le processus pour $u\in\{1,\dots,D(E)\}$ avec $D(E)$ assez grand forme une base standard. Pour transformer cette idée en algorithme il faudrait assurer effectivement la terminaison par une borne supérieure du degré maximal $D(E)$ à atteindre. Une telle borne existe mais est disproportionnée en général. Ce qui précède consiste à considérer la structure k-vectorielle de I. Du point de vue des R-modules, le noyau de la même application est par définition le *module des relations* entre $f_1,\dots,f_s$, ou *(premier) module de syzygies*. On peut le construire par algèbre linéaire sur k en degré donné car cela revient à trouver le noyau de ϕ_u.

Il se trouve que le module des relations entre les éléments d'une base standard se construit facilement ; en retour ceci nous conduira à un algorithme de construction d'une base standard à partir d'un système donné de générateurs.

Ce qui précède se transpose aisément au cas affine, en remplaçant la graduation $(R_d)_{d\in\mathbb{N}}$ par la *filtration* $(S_d)_{d\in\mathbb{N}}$ donnée par les espaces vectoriels S_d constitués des polynômes de degré au plus d, au sens de la définition suivante :

DÉFINITION 2. *Un anneau R est filtré s'il existe des sous-groupes S_d de $(R,+)$ tels que $R=\bigcup_{d\in\mathbb{N}} S_d$ et $S_dS_{d'}\subset S_{d+d'}$.*

2. Polynôme de syzygie

DÉFINITION 3. *Soit $(f_1,\dots,f_s)$ une famille de polynômes non nuls. Pour i différent de j, le* polynôme de syzygie*, ou* S-polynôme*, est le polynôme*

$$S(f_i,f_j)=\frac{\operatorname{lt}(f_j)f_i-\operatorname{lt}(f_i)f_j}{\operatorname{pgcd}\big(\operatorname{lm}(f_i),\operatorname{lm}(f_j)\big)}. \tag{1}$$

Sous l'hypothèse additionnelle que les f_i constituent une base standard, en divisant, faiblement ou fortement, $S(f_i,f_j)$ par $(f_1,\dots,f_s)$, on obtient un reste nul et donc une relation entre les polynômes de la base standard ; nous conviendrons d'appeler une telle relation une relation bilatérale évidente*. Elle est de la forme $S(f_i,f_j)=q_1f_1+\cdots+q_sf_s$, avec pour chaque ℓ entre 1 et s, $\operatorname{lm}(q_\ell f_\ell)<\operatorname{lm}\big(\operatorname{lm}(f_j)\,f_i\big)=\operatorname{lm}\big(\operatorname{lm}(f_i)\,f_j\big)$.*

Une première remarque est que le choix de normalisation du coefficient dominant du polynôme (1) n'a pas d'incidence sur la théorie, vu que seul les espaces vectoriels engendrés comptent. Par ailleurs, une définition équivalente des polynômes de syzygie est

$$S(f_i,f_j)=c_j\frac{\operatorname{ppcm}\big(\operatorname{lm}(f_i),\operatorname{lm}(f_j)\big)}{\operatorname{lm}(f_i)}f_i-c_i\frac{\operatorname{ppcm}\big(\operatorname{lm}(f_i),\operatorname{lm}(f_j)\big)}{\operatorname{lm}(f_j)}f_j,$$

après avoir posé $\operatorname{lt}(f_i)=c_i\operatorname{lm}(f_i)$ et $\operatorname{lt}(f_j)=c_j\operatorname{lm}(f_j)$. Cette formulation est le calcul effectué en pratique.

EXEMPLE 1. La base standard $g_1=2XY-XZ, g_2=X^2, g_3=X^2Z$ pour l'ordre `plex` avec $X>Y>Z$ donne les relations bilatérales évidentes : $Xg_1-2Yg_2=-g_3$, $XZg_1-2Yg_3=-Zg_3$, $Zg_2-g_3=0$.

L'ensemble des relations bilatérales évidentes n'est pas aussi particulier qu'on pourrait le penser, en raison du théorème suivant.

THÉORÈME 1 (Spear–Schreyer). *Les relations bilatérales évidentes engendrent le module des relations entre les éléments d'une base standard.*

DÉMONSTRATION. Donnons-nous une relation $\sum_{i=1}^s g_if_i=0$ entre les générateurs. Posons $m_i=\operatorname{lm}(g_if_i)$. Soient m le maximum des m_i et T l'ensemble des indices i tels que m_i soit égal à m. Remarquons que T ne peut pas être réduit à un seul élément. Soient r et t deux éléments distincts de T ; m est un multiple commun de $\operatorname{lm}(f_r)$ et $\operatorname{lm}(f_t)$, donc du ppcm de ces deux monômes, et à l'aide d'un multiple de la relation bilatérale évidente entre f_r et f_t, on peut récrire la relation initiale en une nouvelle pour laquelle soit T contient moins d'éléments, soit le monôme m est plus petit. Il suffit alors de réappliquer le procédé aussi longtemps que possible pour conclure. Nous renvoyons au livre [2, section 2.6, théorème 6] pour plus de détails. □

3. L'algorithme de construction de Buchberger

En s'inspirant de ce qui précède, on peut imaginer un algorithme de construction d'une base standard de I à partir d'un système donné F de générateurs, algorithme procédant par adjonctions successives de restes non nuls de divisions de polynômes de syzygies. Cette idée revient historiquement à [1] qui l'a exprimée dans

le cas affine, en appelant le résultat *base de Gröbner* (du nom de son directeur de thèse). Cet algorithme est détaillé dans la figure 1, la division utilisée est, au choix, faible ou forte.

Algorithme de Buchberger

Entrée : Un système de générateurs F de l'idéal I.

Sortie : Une base standard G de l'idéal I.

$G \leftarrow F$
Répéter
  $G' \leftarrow G$
  Pour toute paire p, q $(p \neq q)$ de G' faire
    Diviser $S(p,q)$ par G'
    Si le reste r est non nul alors $G \leftarrow G \cup \{r\}$
Jusqu'à $G = G'$.

FIGURE 1. Algorithme de construction de bases standard

Nous allons donner une preuve rapide de la correction et de la terminaison de l'algorithme. Pour plus de détail, nous renvoyons à [2, section 2.7].

THÉORÈME 1. *Une famille $(g_1, \dots, g_r)$ est une base standard de l'idéal qu'ils engendrent si et seulement si pour tout $1 \leq i, j \leq r$ le polynôme de syzygie $S(g_i, g_j)$ se réduit à zéro après division faible par $(g_1, \dots, g_r)$.*

DÉMONSTRATION. L'implication directe est conséquence de la Proposition 5 du chapitre précédent. La réciproque est de même nature que la preuve du théorème de Spear–Schreyer : soit $f = \sum_i f_i g_i$ un polynôme non nul de l'idéal, et montrons que son monôme de tête est dans l'idéal monomial engendré par $\mathrm{lm}(g_1), \dots, \mathrm{lm}(g_r)$. Pour cela, soient $m_i = \mathrm{lm}(g_i f_i)$, m le maximum des m_i, T l'ensemble des indices i tels que m_i soit égal à m. Si $m \neq \mathrm{lm}(f)$, alors T a au moins deux éléments r et t, et on utilise la réécriture donnée par $S(g_r, g_t)$ pour diminuer T jusqu'à ce que $m = \mathrm{lm}(f)$ comme suit. Notons $m_{r,t}$ le ppcm de $\mathrm{lm}(g_r)$ et $\mathrm{lm}(g_t)$, de sorte que

$$S(g_r, g_t) = \mathrm{lc}(g_t)\frac{m_{r,t}}{\mathrm{lm}(g_r)} g_r - \mathrm{lc}(g_r)\frac{m_{r,t}}{\mathrm{lm}(g_t)} g_t.$$

On obtient alors l'égalité suivantes :

$$f_r g_r + f_t g_t - \frac{\mathrm{lc}(f_r) m}{\mathrm{lc}(g_t) m_{r,t}} S(g_r, g_t) = (f_r - \mathrm{lt}(f_r)) g_r + \left(f_t + \frac{\mathrm{lc}(g_r)\,\mathrm{lc}(f_r) m}{\mathrm{lc}(g_t)\,\mathrm{lm}(g_t)} \right) g_t.$$

Comme $\frac{\mathrm{lc}(f_r) m}{\mathrm{lc}(g_t) m_{r,t}} S(g_r, g_t)$ est divisible par $g_1, \dots, g_r$ et ne contient que des monômes strictement inférieurs à m, alors on obtient une nouvelle écriture de la forme $f = \sum_i f_i g_i$ avec un ensemble T strictement plus petit.

En itérant le procédé, on finit bien par trouver une écriture telle que $\mathrm{lm}(f)$ est multiple d'un $\mathrm{lm}(g_i)$. Les $\mathrm{lm}(g_i)$ engendrent donc l'escalier de l'idéal. Par la Définition 16 du chapitre précédent, $(g_1, \dots, g_r)$ est une base standard. □

THÉORÈME 2. *L'algorithme de Buchberger est correct.*

DÉMONSTRATION. Pour la terminaison, il suffit de considérer l'idéal monomial $\mathrm{lm}(G)$ engendré par $\mathrm{lm}(g_1), \dots, \mathrm{lm}(g_s)$. À chaque tour de la boucle l'idéal croît au sens large. Il est constant à partir d'un certain rang par noethérianité. Or, si $\mathrm{lm}(G)$ stagne, G stagne. En effet un reste r non nul agrandirait l'idéal monomial. Autrement son monôme de tête serait réductible et la division continuerait après r.

La correction de l'algorithme de Buchberger est un corollaire du théorème précédent : si l'algorithme s'arrête, c'est que tous les polynômes de syzygies se réduisent à zéro. □

EXERCICE 1 (Critères de Buchberger). Les deux critères suivants sont dus à Buchberger. Ils permettent de savoir sans calcul que certains S-polynômes se réduisent à zéro, ce qui permet d'accélérer l'algorithme de Buchberger. Soient $R = k[X_1 \dots, X_n]$ et $<$ un ordre monomial sur R. Montrer les deux critères suivants :

1. Premier critère : Soient $f, g \in R$ tels que leurs monômes de tête n'aient aucune variable en commun. Alors le reste de la division, faible ou forte, de $S(f, g)$ par la famille $\{f, g\}$ est nul.
2. Second critère : Soient $F \subset R$ une famille finie de R, $p, f, g, \in R$ des polynômes tels que :
   (a) $\mathrm{lm}(p) \mid \mathrm{ppcm}\big(\mathrm{lm}(f), \mathrm{lm}(g)\big)$;
   (b) les restes de la division forte de $S(f, p)$ et $S(g, p)$ par F sont nuls.

   Alors le reste de la division forte de $S(f, g)$ par F est nul.

EXERCICE 2 (Appartenance au radical, astuce de Rabinowitsch). Le radical d'un idéal $I \subseteq R$, noté $\sqrt{I}$ est défini par $\{f \in R \mid \exists r \in \mathbb{N},\ f^r \in I\}$. Soit $I = (f_1, \dots, f_s)$ un idéal de $k[X_1, \dots, X_n]$, et soit h un polynôme de $k[X_1, \dots, X_n]$. Soit T une nouvelle variable et on considère l'idéal $K := I + (Th - 1)$ de $k[T, X_1, \dots, X_n]$.

1. Montrer que h appartient au radical $\sqrt{I}$ de I si, et seulement si, 1 appartient à K.
2. En déduire un algorithme pour tester l'appartenance de h au radical de I.

REMARQUE. En fait, l'algorithme de Buchberger (Figure 1) en homogène est mieux compris car le degré des polynômes dans l'algorithme est croissant. Dans le cas affine, l'évolution du degré des polynômes dans l'algorithme est erratique.

4. Exemple de calcul

Nous donnons ici le calcul d'une base standard qui resservira dans l'exercice 3 : celui pour l'idéal $(x+y-z, x^2-2t^2, y^2-5t^2)$ et l'ordre lexicographique induit par $x > y > z > t$. Nous présentons tous les polynômes du calcul par monômes décroissants, en soulignant le monôme dominant. Dans ce qui suit, nous ajoutons une règle de réécriture dont le lecteur se convaincra qu'elle ne change rien à la théorie, en se permettant de remplacer un polynôme par tout multiple non nul par un entier, la fonction de cette réécriture étant d'éviter l'apparition de dénominateurs. Nous utiliserons le symbole $\sim$ pour une telle réécriture vide.

Nous débutons avec la famille suivante :

$$p_1 := \underline{x} + y - z, \qquad p_2 := \underline{x^2} - 2t^2, \qquad p_3 := \underline{y^2} - 5t^2.$$

Par le premier critère de Buchberger, $S(p_1, p_3)$ et $S(p_2, p_3)$ vont se réduire à zéro. Nous ne réduisons donc que :

$$\begin{aligned} S(p_1, p_2) &= xp_1 - p_2 = \underline{xy} - xz + 2t^2 \\ &\to_{p_1} -\underline{xz} - y^2 + yz + 2t^2 \to_{p_1} -\underline{y^2} + 2yz - z^2 + 2t^2 \to_{p_3} 2\underline{yz} - z^2 - 3t^2, \end{aligned}$$

ce qui nous fait définir un nouveau polynôme dans la base standard en construction :

$$p_4 := 2\underline{yz} - z^2 - 3t^2.$$

Comme précédemment, les polynômes de syzygies $S(p_1, p_4)$ et $S(p_2, p_4)$ vont se réduire à zéro, si bien que nous ne considérons que la réduction suivante, qui mène à un nouveau polynôme p_5 :

$$\begin{aligned} S(p_3, p_4) = 2zp_3 - yp_4 = \underline{yz^2} + 3yt^2 - 10zt^2 \\ \sim 2\underline{yz^2} + 6yt^2 - 20zt^2 \to_{p_4} 6\underline{yt^2} - 17zt^2 + z^3 =: p_5. \end{aligned}$$

Ne subsistent ensuite que deux réductions intéressantes, celles de $S(p_3, p_5)$ et de $S(p_4, p_5)$:

$$\begin{aligned} S(p_3, p_5) = 6t^2p_3 - yp_5 = -\underline{yz^3} + 17yzt^2 - 30t^4 \sim -2\underline{yz^3} + 34yzt^2 - 60t^4 \\ \to_{p_4} 34\underline{yzt^2} - z^4 - 3z^2t^2 + 60t^4 \to_{p_4} -\underline{z^4} + 14z^2t^2 - 9t^4 := p_6, \\ S(p_4, p_5) = 3t^2p_4 - zp_5 = -\underline{z^4} + 14z^2t^2 - 9t^4 \to_{p_6} 0, \end{aligned}$$

après lesquelles p_6 ne produit aucun polynôme de syzygie intéressant.

Nous avons donc obtenu que $G_1 = \{p_1, \dots, p_6\}$ est une base standard de l'idéal considéré. Il s'avère qu'il est en de même de $G_2 = \{p_1, p_3, \dots, p_6\}$: en effet, les deux systèmes engendrent le même idéal et comme nous n'avons jamais utilisé p_2 dans les divisions ci-dessus, les polynômes de syzygie de G_2 se réduisent tous à 0 par G_2, qui est donc aussi une base standard.

EXERCICE 3.

1. Déduire des calculs précédents que $\sqrt{2} + \sqrt{5}$ est algébrique sur le corps des rationnels $\mathbb{Q}$, en exhibant un polynôme à une variable à coefficients rationnels dont il est racine.
2. Quel est le résultant de $(y - z)^2 - 2$ et $y^2 - 5$ par rapport à y ?
3. Déduire des calculs précédents que $\mathbb{Q}(\sqrt{2}, \sqrt{5}) = \mathbb{Q}(\sqrt{2} + \sqrt{5})$. Exprimer $\sqrt{2}$ et $\sqrt{5}$ en fonction de $\sqrt{2} + \sqrt{5}$.

5. Propriétés des bases standard pour quelques ordres

Dans cette section, nous étudions les propriétés des bases standard pour l'ordre `plex` dans le cas affine et `tdeg` dans le cas projectif.

5.1. Ordre `plex` — Projection et implicitation. Nous rappelons que `plex` est l'ordre lexicographique induit par $X_1 > X_2 > \cdots > X_n$.

PROPOSITION 1. *Soit $g \in k[X_1, \dots, X_n]$. Si X_1 ne divise pas le monôme de tête pour* `plex` *de g alors $g \in k[X_2, \dots, X_n]$.*

DÉMONSTRATION. Le monôme de tête est l'un des monômes qui contient le plus de X_1. Si un monôme de g contient du X_1 alors son monôme de tête en contient. □

PROPOSITION 2. *Soit $g_1, \dots, g_u, g_{u+1}, \dots, g_s$ une base standard d'un idéal I pour* `plex`*. Supposons que X_1 divise les monômes de tête de $g_1, \dots, g_u$ uniquement. Alors la famille $g_{u+1}, \dots, g_s$ engendre $I \cap k[X_2, \dots, X_n]$ et en forme une base standard.*

DÉMONSTRATION. La proposition précédente nous donne que $g_{u+1}, \dots, g_s \in k[X_2, \dots, X_n]$ donc $(g_{u+1}, \dots, g_s) \subseteq I \cap k[X_2, \dots, X_n]$. Soit $f \in I \cap k[X_2, \dots, X_n]$. Le terme monôme tête $\mathrm{lm}(f)$ est dans l'escalier engendré par $\mathrm{lm}(g_1), \dots, \mathrm{lm}(g_s)$ donc c'est un multiple d'un $\mathrm{lm}(g_i)$ pour un certain $i \in \{1, \dots, s\}$. On a nécessairement $i \notin \{1, \dots, u\}$ car X_1 ne divise pas $\mathrm{lm}(f)$. Nous venons de montrer que $g_{u+1}, \dots, g_s$ est une base standard de $I \cap k[X_2, \dots, X_n]$ et donc que cette famille l'engendre. □

Le pendant géométrique de l'élimination de X_1 dans l'idéal I est une projection selon X_1 sur les autres coordonnées. Nous prouverons ce résultat dans le chapitre suivant qui nous donnera plus d'outils pour faire le lien entre l'algèbre et la géométrie.

Par l'une ou l'autre des interprétations, on observe qu'un polynôme de I qui ne fait pas intervenir X_1 fournit une équation vérifiée par les coordonnées $(x_2, \ldots, x_n)$ d'une solution x de I *quelle que soit la valeur de sa coordonnée* x_1. L'élimination est donc bien l'outil pour répondre au problème d'implicitation donné en motivation au début de ce chapitre.

EXERCICE 4. Soient $I = (f_1, \ldots, f_s)$ et $J = (g_1, \ldots, g_r)$ deux idéaux de $k[X_1, \ldots, X_n]$. Soit T une nouvelle variable et soit

$$K := (Tf_1, \ldots, Tf_s, (1-T)g_1, \ldots, (1-T)g_r)$$

vu comme un idéal de $k[T, X_1, \ldots, X_n]$

1. Montrer que $I \cap J = K \cap k[X_1, \ldots, X_n]$.
2. En déduire un algorithme pour calculer un ensemble de polynômes générateurs de $I \cap J$.

5.2. Ordre `tdeg` en homogène — Section par un hyperplan. Nous rappelons que `tdeg` est l'ordre gradué lexicographique inverse.

PROPOSITION 3. *Soit g un polynôme homogène de $k[X_0, \ldots, X_n]$. Si X_n divise le monôme de tête de g pour* `tdeg`*, alors X_n divise g.*

DÉMONSTRATION. Comme g est homogène, son monôme de tête est l'un des monômes contenant le moins de X_n. S'il y a du X_n dans le monôme de tête de g, il y en a dans tous les autres monômes et ainsi X_n divise g. □

PROPOSITION 4. *Soit $g_1, \ldots, g_u, g_{u+1}, \ldots, g_s$ une base standard homogène d'un idéal I pour* `tdeg` *ordonnée telle que X_n divise uniquement les monômes de tête de $g_{u+1}, \ldots, g_s$. Alors la famille $g_1, \ldots, g_u, X_n$ engendre $I + (X_n)$ et en forme une base standard.*

DÉMONSTRATION. La proposition précédente nous donne que X_n divise g_{u+1}, $\ldots, g_s$. On en déduit facilement que $(g_1, \ldots, g_u, X_n) = I + (X_n)$. Soit $f \in I + (X_n)$. Montrons que son terme monôme tête $\mathrm{lm}(f)$ est un multiple de l'un des monômes de tête $\mathrm{lm}(g_1), \ldots, \mathrm{lm}(g_u)$, ou de X_n. Si X_n divise le monôme de tête $\mathrm{lm}(f)$ de f alors c'est bien le cas. Sinon, soit $h \in (X_n)$ tel que $f - h \in I$. On vérifie que $\mathrm{lm}(f) = \mathrm{lm}(f-h)$ et donc $\mathrm{lm}(f)$ est multiple de l'un des $\mathrm{lm}(g_i)$ pour $i \in \{1, \ldots, s\}$. Cependant, $i \notin \{u+1, \ldots, s\}$, car X_n ne divise pas $\mathrm{lm}(f)$. Nous venons de montrer que $(g_1, \ldots, g_u, X_n)$ est une base standard de l'idéal $I + (X_n)$. □

L'opération de rajouter l'équation X_n à l'idéal I correspond à l'opération de section de $\mathbf{V}(I)$ avec l'hyperplan X_n.

Notes

L'étude des propriétés du module de syzygies d'un idéal a été réalisée indépendamment par Spear [8] et Schreyer [7], et a mené au théorème 1 et à un algorithme de calcul d'un système de générateurs de ce module. Ces résultats trouvent néanmoins leurs racines dans les travaux de Janet [3]. Cette approche algorithmique a pu être ensuite prolongée par Möller et Mora [5]. Pour d'avantage de détails nous renvoyons le lecteur à l'ouvrage [6, Chapter 23.8].

L'étude des bases standard a conduit à une vaste littérature concernant leurs propriétés et les algorithmes pour les calculer. Il a été prouvé que les degrés des polynômes d'une base standard minimale de $(f_1, \ldots, f_s) \subseteq k[X_1, \ldots, X_n]$ sont au

plus $2(d^2/2+d)^{2^{n-1}}$ lorsque $\deg f_i \leq d$ pour tout i. Cette borne est doublement exponentielle dans le nombre de variables. En outre, il existe des idéaux pour lesquels les bases standard contiennent au moins $2^{2^{cn}}$ éléments et des éléments de degré au moins $2^{2^{cn}}$ pour une certaine constante $c \in \mathbb{R}$ [4].

Bibliographie

[1] Buchberger, B. (1965). « Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal) ». Thèse de doct. Mathematical Institute, University of Innsbruck, Austria.

[2] Cox, David, John Little et Donal O'Shea (1996). *Ideals, varieties, and algorithms*. 2e éd. Springer-Verlag, xiv+536 pages.

[3] Janet, M. (1920). « Sur les systèmes d'équations aux dérivées partielles ». In : *J. Math. Pure et Appl.* vol. 3, p. 65–151.

[4] Mayr, Ernst W. et Albert R. Meyer (1982). « The complexity of the word problems for commutative semigroups and polynomial ideals ». In : *Advances in Mathematics*, vol. 46, n°3, p. 305–329.

[5] Möller, H. Michael et Ferdinando Mora (1986). « New constructive methods in classical Ideal theory ». In : *J. Algebra*, vol. 100, n°1, p. 138–178.

[6] Mora, Teo (2005). *Solving Polynomial Equation Systems II : Macaulay's Paradigm and Gröbner Technology*. Vol. 99. Encyclopedia of Mathematics and its Applications. Cambridge Univ. Press.

[7] Schreyer, F. (1980). « Die Berechnung Syzygien mit dem verallgemeinerten Weierstrasschen Divisionsatz ». Diplomarbeit. Hamburg.

[8] Spear, D. (1978). « A constructive approach to commutative ring theory ». In : *1977 MACSYMA User's Conference*, p. 369–376.

CHAPITRE 23

Nullstellensatz et applications

Résumé

Le Nullstellensatz (en allemand, « théorème du lieu des zéros ») est le théorème qui fait le lien entre les variétés algébriques et les idéaux. Il en découle que certaines opérations géométriques se traduisent aisément en algorithmes sur les équations. L'exemple de la projection d'une variété algébrique sur un hyperplan sera traité en application.

Si I est un idéal *propre*, *i.e* $I \neq (1)$, de $k[X_1, \ldots, X_n]$, et si k est algébriquement clos, alors l'objet de ce chapitre est de montrer que $\mathbf{V}(I) \neq \varnothing$.

1. Nullstellensatz affine

La preuve dans le cas affine que nous proposons ici repose tout d'abord sur l'utilisation de variables particulières.

1.1. Changement des variables.

LEMME 1. *Si k est infini et si $f \in k[X_1, \ldots, X_n]$ est non nul, alors $k^n \setminus \mathbf{V}(f)$ est non vide.*

DÉMONSTRATION. La preuve s'effectue par récurrence sur le nombre de variables. Si $n = 1$ le lemme est vrai. Supposons qu'il soit vrai pour $n \geq 1$. En appliquant l'hypothèse de récurrence, il existe $(a_1, \ldots, a_{n-1}) \in k^{n-1}$ qui n'annule pas le terme constant de f vu comme polynôme en X_n à coefficients dans $k[X_1, \ldots, X_{n-1}]$. Ainsi $f(a_1, \ldots, a_{n-1}, X_n)$ est un polynôme non nul. Il a donc un nombre fini de racines. □

PROPOSITION 1. *Si k est infini, alors pour tout $f \in k[X_1, \ldots, X_n]$ non nul, il existe des constantes $a_1, \ldots, a_{n-1}$ dans k telles que le changement de variables inversible*

$$(1) \qquad \begin{cases} X_1 & \leftarrow & X_1 + a_1 X_n \\ X_2 & \leftarrow & X_2 + a_2 X_n \\ \vdots & \leftarrow & \vdots \\ X_{n-1} & \leftarrow & X_{n-1} + a_{n-1} X_n \end{cases}$$

rende f unitaire en X_n.

DÉMONSTRATION. On peut supposer f homogène de degré d. Comme le coefficient de tête de $f(X_1 + a_1 X_n, X_2 + a_2 X_n, \ldots, X_{n-1} + a_{n-1} X_{n-1}, X_n)$ comme polynôme en X_n est $f(a_1, \ldots, a_{n-1}, 1) \neq 0$, la conclusion découle du lemme précédent. □

1.2. Spécialisation du résultant. Le résultat suivant fournit une condition simple pour que le résultant de la spécialisation de deux polynômes corresponde avec la spécialisation de leur résultant.

PROPOSITION 2. *Soient A et B deux anneaux commutatifs avec des unités respectives 1_A et 1_B. Soit φ un morphisme de A dans B qui envoie 1_A sur 1_B, et que l'on étend naturellement à $A[X]$ en l'appliquant coefficient par coefficient. Soient f et g deux polynômes de $A[X]$. Si f est unitaire alors $\varphi(\mathrm{Res}(f,g)) = \mathrm{Res}(\varphi(f),\varphi(g))$.*

DÉMONSTRATION. Notons $f = 1_A X^m + f_{m-1}X^{m-1} + \cdots + f_0$, et $g = g_n X^n + \cdots + g_0$, avec $g_n \neq 0$. La matrice de Sylvester $\mathsf{Syl}(f,g)$ s'écrit :

$$\mathsf{Syl}(f,g) = \begin{pmatrix} 1_A & f_{m-1} & \cdots & f_0 & & & \\ & 1_A & f_{m-1} & \cdots & f_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & 1_A & f_{m-1} & \cdots & f_0 \\ g_n & g_{n-1} & \cdots & g_0 & & & \\ & g_n & g_{n-1} & \cdots & g_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & g_n & g_{n-1} & \cdots & g_0 \end{pmatrix}$$

En notant $r_i = r_{i,m-1}X^{m-1} + \cdots + r_{i,0}$ le reste de la division euclidienne de $X^i g$ par f, et en effectuant les opérations sur les lignes de $\mathsf{Syl}(f,g)$ correspondant à ces divisions, on obtient que le résultant de f et g est égal au déterminant de la matrice

$$M(f,g) := \begin{pmatrix} r_{m-1,m-1} & r_{m-1,m-2} & \cdots & r_{m-1,0} \\ r_{m-2,m-1} & r_{m-2,m-2} & \cdots & r_{m-2,0} \\ \vdots & \vdots & \cdots & \vdots \\ r_{0,m-1} & r_{0,m-2} & \cdots & r_{0,0} \end{pmatrix}$$

Comme f est unitaire, les coefficients de r_i sont de polynômes en les coefficients de f et g. Par conséquent $\varphi(r_i)$ est égal au reste de la division euclidienne de $\varphi(X^i g)$ par $\varphi(f)$, et donc $\varphi(M(f,g)) = M(\varphi(f),\varphi(g))$. La conclusion vient du fait que $\varphi(\det(M(f,g)) = \det(\varphi(M(f,g)))$. □

1.3. Nullstellensatz faible.

THÉORÈME 1 (Nullstellensatz affine faible). *Si k est algébriquement clos, et si I est un idéal de $k[X_1,\ldots,X_n]$ tel que $\mathbf{V}(I) = \varnothing$, alors $I = k[X_1,\ldots,X_n]$.*

DÉMONSTRATION. Cette démonstration est adaptée de [1]. Montrons, par récurrence sur la dimension n de l'espace ambiant, que si $I = (f_1,\ldots,f_s)$ est un idéal propre alors $\mathbf{V}(I) \neq \varnothing$. Si $n = 1$, alors comme $k[X_1]$ est principal, $I = (f)$ avec $f \neq 1$, et f a une racine car k est algébriquement clos.

Supposons le résultat vérifié en dimension $n-1$. Quitte à faire le changement de coordonnées de la Proposition 1, on peut supposer f_1 unitaire en X_n (rappelons qu'un corps algébriquement clos est nécessairement infini). On pose $J = I \cap k[X_1,\ldots,X_{n-1}]$. Comme J ne contient pas 1, il est propre. Par récurrence, il existe $(a_1,\ldots,a_{n-1}) \in \mathbf{V}(J)$.

Soit $f \in I$, considérons $g := \mathrm{Res}_{X_n}(f_1, f)$. Par la Proposition 3 du Chapitre 6 on a $g \in I \cap k[X_1,\ldots,X_{n-1}]$ et donc $g(a_1,\ldots,a_{n-1}) = 0$. Par ailleurs, comme f_1 est unitaire en X_n, la Proposition 2 nous donne

$$g(a_1,\ldots,a_{n-1}) = \mathrm{Res}_{X_n}(f_1(a_1,\ldots,a_{n-1},X_n), f(a_1,\ldots,a_{n-1},X_n)),$$

ce qui implique que $f(a_1,\ldots,a_{n-1},X_n)$ n'est pas constant. Finalement l'idéal

$$H := \{f(a_1,\ldots,a_{n-1},X_n) \mid f \in I\}$$

de $k[X_n]$ est propre, ce qui nous permet de conclure, comme pour $n = 1$, qu'il y a un zéro dans $\mathbf{V}(H)$ et donc un zéro dans $\mathbf{V}(I)$. □

EXEMPLE 1. Il est nécessaire de regarder les zéros sur un corps k algébriquement clos. En effet soit $I = (X^2+1)$. Alors $\mathbf{V}_{\mathbb{R}}(I) = \varnothing$ mais $1 \notin I$. Ceci est cohérent avec le résultat précédent car la variété $\mathbf{V}_{\mathbb{C}}(I)$ se réduit aux points complexes $\pm i$.

1.4. Nullstellensatz fort. Nous allons montrer que la version faible du Nullstellensatz implique une version plus forte.

PROPOSITION 3. *Supposons k algébriquement clos, et soient $I = (f_1, \ldots, f_s)$ un idéal de $k[X_1, \ldots, X_n]$ et g un polynôme qui s'annule identiquement sur $\mathbf{V}(I)$. Alors la variété de l'idéal $I' := I + (Yg - 1)$ de $k[X_1, \ldots, X_n, Y]$ est vide.*

DÉMONSTRATION. D'une part, $I \subset I'$ implique $\mathbf{V}(I') \subset \mathbf{V}(I)$ dans $\mathbb{A}^{n+1}$. Mais d'autre part, puisque $y\, g(x_1, \ldots, x_n) - 1 = 0$ implique $g(x_1, \ldots, x_n) \neq 0$, $\mathbf{V}(I') \subset \mathbb{A}^{n+1} \setminus \mathbf{V}(g) \subset \mathbb{A}^{n+1} \setminus \mathbf{V}(I)$. □

DÉFINITION 1. *Soit I un idéal de $R = k[X_1, \ldots, X_n]$. Le* radical *de I, noté $\sqrt{I}$, est l'idéal $\{f \in R \mid \exists r \in \mathbb{N},\ f^r \in I\}$; il contient I. Un idéal I est dit* radical *si $I = \sqrt{I}$.*

THÉORÈME 2 (Nullstellensatz affine fort). *Si k est algébriquement clos, si I est un idéal de $R = k[X_1, \ldots, X_n]$, alors tout polynôme $g \in R$ qui s'annule identiquement sur $\mathbf{V}(I)$ appartient à $\sqrt{I}$.*

DÉMONSTRATION. Par la Proposition 3, $\mathbf{V}(I') = \varnothing$ pour $I' = I + (Yg - 1)$, si bien que I' contient 1 par le Théorème 1. Il existe donc des polynômes p_i et un polynôme p dans $k[X_1, \ldots, X_n, Y]$ tels que

$$1 = \sum_{i=1}^{s} p_i(X_1, \ldots, X_n, Y) f_i(X_1, \ldots, X_n) + p(X_1, \ldots, X_n, Y)(Yg(X_1, \ldots, X_n) - 1).$$

Substituons $1/g(X_1, \ldots, X_n)$ à Y dans cette égalité. Ainsi,

$$1 = \sum_{i=1}^{s} p_i(X_1, \ldots, X_n, 1/g) f_i(X_1, \ldots, X_n).$$

En multipliant par une puissance suffisante de g nous tuons les dénominateurs. Plus précisément, pour $r = \max(\deg_Y p_i)$, nous avons

$$g^r = \sum_{i=1}^{s} g^r p_i(X_1, \ldots, X_n, 1/g) f_i(X_1, \ldots, X_n)$$

avec $g^r p_i(X_1, \ldots, X_n, 1/g) \in k[X_1, \ldots, X_n]$. C'est-à-dire $g^r \in I$. □

COROLLAIRE 1. *Soit I un idéal, alors $\mathbf{I}(\mathbf{V}(I)) = \sqrt{I}$.*

DÉMONSTRATION. Le Nullstellensatz affine fort donne $\mathbf{I}(\mathbf{V}(I)) \subseteq \sqrt{I}$. L'inclusion inverse est immédiate. □

Les théorèmes précédents font le lien entre les variétés affines et les idéaux qui les définissent :

Algèbre		Géométrie
Idéal radical		Variété
I	$\rightarrow$	$\mathbf{V}(I)$
$\mathbf{I}(V)$	$\leftarrow$	V

2. Nullstellensatz projectif

Nous avons des théorèmes analogues aux précédents pour les variétés projectives et les idéaux homogènes.

DÉFINITION 2. *Le cône affine $\tilde{V}$ d'une variété projective V est l'ensemble des représentants des éléments de V :*

$$\tilde{V} := \{(x_0, \ldots, x_n) \in \mathbb{A}^{n+1} \mid (x_0, \ldots, x_n) = (0, \ldots, 0) \text{ ou } (x_0 : \ldots : x_n) \in V\}.$$

Remarquons que si I est un idéal homogène de $k[X_0, \ldots, X_n]$ alors le cône affine de la variété projective de I est égal à la variété affine de I.

THÉORÈME 3 (Nullstellensatz projectif faible). *Si k est algébriquement clos, si I est un idéal homogène de $k[X_0, \ldots, X_n]$ tel que $\mathbf{V}(I) = \varnothing$, alors il existe $r \in \mathbb{N}$ tel que $J^r \subseteq I$, où J est l'idéal $\mathbf{I}((0, \ldots, 0)) = (X_0, \ldots, X_n)$.*

DÉMONSTRATION. Le cône affine $\widetilde{\mathbf{V}(I)}$ de $\mathbf{V}(I)$ est réduit au point $(0, \ldots, 0)$. Pour tout $i \in \{1, \ldots, n\}$, le polynôme X_i s'annule identiquement sur $\widetilde{\mathbf{V}(I)}$. Par le Nullstellensatz affine fort, il existe $r_i \in \mathbb{N}$ tel que $X_i^{r_i} \in I$. Soient $m = \max(r_i)$ et $r = (n+1)(m-1)+1$. Montrons que $(X_0, \ldots, X_n)^r \subset I$.

L'idéal J^r est l'idéal engendré par les monômes de degré r. Soit X^α un tel monôme avec $\alpha = (\alpha_0, \ldots, \alpha_n) \in \mathbb{N}^{n+1}$ et $|\alpha| = \sum_i \alpha_i = r$. Alors il existe i tel que $\alpha_i \geq m$ donc $X^\alpha = X_i^r X^\beta \in I$ pour un certain $\beta \in \mathbb{N}^{n+1}$. Ainsi chaque générateur de J^r est dans I et $J^r \subseteq I$. □

THÉORÈME 4 (Nullstellensatz projectif fort). *Si k est algébriquement clos et si I est un idéal homogène de $R = k[X_0, \ldots, X_n]$, alors tout polynôme $g \in R$ homogène qui s'annule identiquement sur $\mathbf{V}(I)$ appartient à $\sqrt{I}$.*

DÉMONSTRATION. Comme dans la démonstration précédente, nous nous ramenons au cas affine en considérant le cône affine $\widetilde{\mathbf{V}(I)}$ dans $\mathbb{A}^{n+1}$ défini par I. Le polynôme g s'annule alors identiquement sur $\widetilde{\mathbf{V}(I)}$, ce qui implique par le Nullstellensatz fort que $g \in \sqrt{I}$. □

COROLLAIRE 2. *Si k est algébriquement clos et si I est un idéal homogène, alors $\mathbf{I}(\mathbf{V}(I)) = \sqrt{I}$. En particulier, $\sqrt{I}$ est un idéal homogène.*

3. Idéaux de dimension zéro

Comme application du Nullstellensatz, nous exprimons ici un premier lien entre l'escalier et la variété affine de certains idéaux.

PROPOSITION 4. *Soit I un idéal de $k[X_1, \ldots, X_n]$. Les assertions suivantes sont équivalentes :*

i) le complémentaire de l'escalier $E(I)$ est fini pour tout ordre admissible ;

ii) il existe un ordre admissible pour lequel le complémentaire de $E(I)$ est fini ;

iii) $\dim_k k[X_1, \ldots, X_n]/I$ est finie ;

iv) la variété affine $\mathbf{V}_{\bar{k}}(I)$ contient un nombre fini de points (où $\bar{k}$ représente la clôture algébrique de k).

DÉMONSTRATION. Il est clair que (i) implique (ii). Comme $R/I = \bigoplus_{a \notin E(I)} kx^a$ alors $\dim_k R/I$ est le cardinal du complémentaire de l'escalier, et donc (ii) implique (iii).

Si (iii) est vrai alors notons $d = \dim_k R/I < \infty$. Pour tout $j \in \{1, \ldots, n\}$, la famille $1, X_j, \ldots, X_j^d$ est liée dans R/I. Donc il existe $P_j \in k[X_j]$ qui appartient à l'idéal I. Par conséquent on a $\mathbf{V}_{\bar{k}}(I) \subset \bigcap_{j=1}^n \mathbf{V}_{\bar{k}}(P_j)$, ce qui implique (iv).

Si (iv) est vrai, alors pour tout $j \in \{1, \ldots, n\}$, la projection de $\mathbf{V}_{\bar{k}}(I)$ sur l'axe porté par X_j est finie. Il existe donc $P_j \in k[X_j]$ qui s'annule sur cette projection. Par le Nullstellensatz affine fort, on déduit de $\mathbf{V}_{\bar{k}}(I) \subseteq \mathbf{V}_{\bar{k}}(P_j)$ l'existence de $r_j \in \mathbb{N}$ tel que $P_j^{r_j} \in I \cap k[X_j]$. Donc l'escalier touche chaque axe et son complémentaire est fini quel que soit l'ordre admissible considéré, ce qui implique (i). □

DÉFINITION 3. *Les idéaux vérifiant la proposition précédente sont appelés les* idéaux zéro-dimensionnels.

4. Projection des variétés affines

Nous allons voir maintenant que le pendant géométrique de l'élimination de la variable X_n correspond à la projection

$$\begin{array}{rccc} \pi : & k^n & \to & k^{n-1} \\ & (x_1, \ldots, x_n) & \mapsto & (x_1, \ldots, x_{n-1}). \end{array}$$

THÉORÈME 5. *Soit I un idéal de $k[X_1, \ldots, X_n]$. La projection $\pi(\mathbf{V}(I))$ de la variété affine $\mathbf{V}(I)$ est contenue dans la variété affine $\mathbf{V}(I \cap k[X_1, \ldots, X_{n-1}])$.*

De plus, si k est algébriquement clos, alors $\mathbf{V}(I \cap k[X_1, \ldots, X_{n-1}])$ est la plus petite variété contenant $\pi(\mathbf{V}(I))$. On dit que c'est la clôture de Zariski *de $\pi(\mathbf{V}(I))$.*

DÉMONSTRATION. Soient $x = (x_1, \ldots, x_{n-1}) \in \pi(\mathbf{V}(I))$ et $f \in I \cap k[X_1, \ldots, X_{n-1}]$. Il existe $x_n \in k$ tel que $x' = (x_1, \ldots, x_n) \in \mathbf{V}(I)$ donc $f(x') = 0$. Comme on peut aussi voir f comme un polynôme de $k[X_1, \ldots, X_{n-1}]$, on peut écrire $f(x) = 0$. Par conséquent, $\pi(\mathbf{V}(I)) \subseteq \mathbf{V}(I \cap k[X_1, \ldots, X_{n-1}])$.

Soit $f \in k[X_1, \ldots, X_{n-1}]$ s'annulant identiquement sur tout $\pi(\mathbf{V}(I)) \subseteq \mathbb{A}^{n-1}$. Puisque f ne dépend pas de X_n alors f, vu comme un polynôme de $k[X_1, \ldots, X_n]$, s'annule identiquement sur $\pi(\mathbf{V}(I)) \times k \subseteq \mathbb{A}^n$. Comme $\mathbf{V}(I) \subseteq \pi(\mathbf{V}(I)) \times k$, le Nullstellensatz fort nous donne $f \in \sqrt{I}$. Finalement f s'annule bien sur $\mathbf{V}(I \cap k[X_1, \ldots, X_{n-1}])$, ce qui prouve que cette dernière est bien la plus petite variété algébrique contenant $\pi(\mathbf{V}(I))$. □

EXEMPLE 2. Soit la variété algébrique $X_1X_2 - 1 = 0$. Alors la projection de l'hyperbole sur l'axe des X_1 est toute la droite privée de l'origine. La plus petite variété la contenant est bien $\mathbf{V}((0))$ car $(X_1X_2 - 1) \cap k[X_1] = (0)$. On voit qu'il peut manquer des points lorsque l'on projette des variétés algébriques affines.

On peut préciser le théorème précédent en étudiant le lieu des points qui n'appartiennent pas à la projection.

PROPOSITION 5. *Supposons k algébriquement clos, et soit $I = (f_1, \ldots, f_s)$ et $I' = I \cap k[X_1, \ldots, X_{n-1}]$ son premier idéal éliminant. Considérons les polynômes f_i comme des polynômes en X_n à coefficients dans $k[X_1, \ldots, X_{n-1}]$ et notons $g_i \in k[X_1, \ldots, X_{n-1}]$ le terme de tête de f_i. Alors*

$$\mathbf{V}(I') = \pi(\mathbf{V}(I)) \cup (\mathbf{V}(g_1, \ldots, g_s) \cap \mathbf{V}(I')).$$

En d'autres termes, tout point de $\mathbf{V}(I')$ qui n'annule pas tous les termes de tête se remonte en un point de $\mathbf{V}(I)$.

DÉMONSTRATION. Il suffit de montrer que $\mathbf{V}(I') \setminus \mathbf{V}(g_1, \ldots, g_s) \subseteq \pi(\mathbf{V}(I))$. Soit $a = (a_1, \ldots, a_{n-1}) \in \mathbf{V}(I')$. Sans perte de généralité on peut supposer que $g_1(a) \neq 0$. Soit $f \in I$ et considérons $g := \mathrm{Res}_{X_n}(f_1, f)$. Par la proposition 3 du Chapitre 6 on a $g \in I'$ et donc $g(a) = 0$. Par ailleurs, comme $g_1(a) \neq 0$, on peut rendre f_1 unitaire en X_n en le voyant comme un polynôme à coefficients dans les séries formelles $k[[X_1 - a_1, \ldots, X_{n-1} - a_{n-1}]]$. La proposition 2 nous donne alors que $g(a)$ est proportionnel sur k à

$$\mathrm{Res}_{X_n}(f_1(a_1, \ldots, a_{n-1}, X_n), f(a_1, \ldots, a_{n-1}, X_n)),$$

ce qui implique que $f(a_1, \ldots, a_{n-1}, X_n)$ n'est pas constant. Finalement l'idéal

$$H := \{f(a_1, \ldots, a_{n-1}, X_n) \mid f \in I\}$$

de $k[X_n]$ est propre, ce qui nous permet de conclure que $\pi^{-1}(a) \cap \mathbf{V}(I)$ est non vide. □

COROLLAIRE 3. *Supposons qu'il existe un polynôme unitaire en X_n dans I. Alors $\mathbf{V}(I') = \pi(\mathbf{V}(I))$.*

EXEMPLE 3. Reprenons l'exemple précédent de l'hyperbole d'équation $X_1X_2 - 1 = 0$ projetée sur l'axe des X_1. Seul le point 0 de l'axe des X_1 n'est pas un projeté. Appliquons le changement de variables $X_1 \leftarrow X_1 + X_2$. Alors f devient $X_2^2 + X_1X_2 - 1$, et on se trouve dans la situation du corollaire, où la projection est bien surjective.

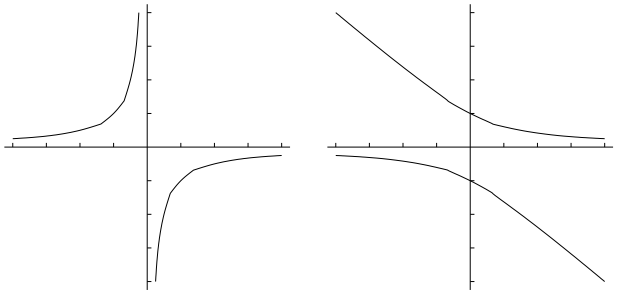


FIGURE 1. Les projections de l'hyperbole

5. Projection des variétés projectives

Avant de pouvoir définir une projection dans un contexte projectif, commençons par quelques définitions et propriétés élémentaires sur les espaces linéaires projectifs.

DÉFINITION 4. *Un* espace linéaire projectif *E est un sous-ensemble de points $(x_0 : \ldots : x_n)$ de l'espace projectif $\mathbb{P}^n$ vérifiant un système d'équations linéaires*

$$\begin{cases} a_{1,0}x_0 & + & \cdots & + & a_{1,n}x_n & = & 0 \\ \vdots & & & & \vdots & & \vdots \\ a_{s,0}x_0 & + & \cdots & + & a_{s,n}x_n & = & 0 \end{cases}.$$

Un hyperplan *projectif est un espace linéaire projectif donné par une seule équation non triviale.*

DÉFINITION 5. *La somme $E + F$ de deux sous-espaces linéaires projectifs de $\mathbb{P}^n$ est définie par $\{(x_0 : \ldots : x_n) \in \mathbb{P}^n \mid (x_0, \ldots, x_n) \in \tilde{E} + \tilde{F}\}$.*

On étend naturellement la notion de dimension aux espaces linéaires projectifs.

DÉFINITION 6. *La dimension de l'espace linéaire projectif E, noté* $\dim E$*, est* $\dim \tilde{E} - 1$*, où $\tilde{E}$ est le cône affine de E.*

On vérifie facilement que $\mathbb{P}^n$ est de dimension n, que les hyperplans sont de dimension $n-1$ et ainsi de suite. Une propriété voulue du plan projectif est que deux droites se coupent toujours. Ceci est une conséquence de la proposition suivante en dimension quelconque.

PROPOSITION 6. *Soient E et F deux sous-espaces linéaires de $\mathbb{P}^n$. Alors*

$$\dim(E + F) + \dim(E \cap F) = \dim E + \dim F.$$

DÉMONSTRATION. Les espaces vectoriels associés $\tilde{E}$ et $\tilde{F}$ sont respectivement de dimension $\dim E + 1$ et $\dim F + 1$. Ainsi on a $\dim(\tilde{E} \cap \tilde{F}) + \dim(\tilde{E} + \tilde{F}) = \dim \tilde{E} + \dim \tilde{F}$. Comme $\tilde{E} \cap \tilde{F} = \widetilde{E \cap F}$ et $\tilde{E} + \tilde{F} = \widetilde{E + F}$, on déduit $\dim(E + F) + \dim(E \cap F) = \dim E + \dim F$. □

COROLLAIRE 4. *Soient E et F deux sous-espaces linéaires de $\mathbb{P}^n$ tels que $\dim E + \dim F \geq n$. Alors $E \cap F \neq \varnothing$.*

DÉMONSTRATION. Par la proposition précédente, $\dim E \cap F \geq 0$. Donc $E \cap F$ n'est pas vide. □

Nous allons définir les projections de l'espace projectif et vérifier que cela correspond bien à la notion habituelle affine.

DÉFINITION 7. *Soient L et B deux sous-espaces linéaires propres de $\mathbb{P}^n$ qui ne se coupent pas et tels que $\dim L + \dim B = n - 1$. Alors la projection de* base B *avec L comme* centre de projection *est l'application*

$$\pi : \begin{cases} \mathbb{P}^n \setminus L & \to & \mathbb{P}^n \\ x & \mapsto & (L + x) \cap B. \end{cases}$$

Le fait que l'application de projection π soit bien définie découle du Corollaire 4. Remarquons que cette définition généralise bien les projections affines usuelles. En effet soit la projection affine

$$\pi_n : \begin{cases} \mathbb{A}^n & \to & \mathbb{A}^{n-1} \\ (x_1, \ldots, x_n) & \mapsto & (x_1, \ldots, x_{n-1}). \end{cases}$$

Considérons l'espace affine $\mathbb{A}^n$ comme une partie de $\mathbb{P}^n$ via l'inclusion

$$\begin{cases} \mathbb{A}^n & \hookrightarrow & \mathbb{P}^n \\ (x_1, \ldots, x_n) & \mapsto & (1 : x_1 : \cdots : x_n). \end{cases}$$

On vérifie que la projection affine π_n est la restriction à $\mathbb{A}^n$ de la projection sur la base $B = \mathbf{V}(X_n)$ et de centre $L = (0 : \ldots : 0 : 1)$: en effet $(L + (1 : x_1 : \cdots : x_n)) \cap B$ n'est autre que $(1 : x_1 : \cdots : x_{n-1} : 0)$.

Contrairement au cas affine, le résultat suivant assure que les variétés algébriques projectives sont stables par projection.

THÉORÈME 6. *Supposons k algébriquement clos, et soient L et B deux sous-espaces linéaires propres de $\mathbb{P}^n$ tels que $\dim L + \dim B = n - 1$. Le projeté d'une variété algébrique disjointe de L par la projection de base B et de centre de projection L est encore une variété algébrique.*

DÉMONSTRATION. Supposons que B soit de dimension r engendré par les points $l_0, \ldots, l_r$ de $\mathbb{P}^n$, et que L soit engendré par $l_{r+1}, \ldots, l_n$. Pour tout $i \in \{r, \ldots, n-1\}$, on pose $B_i := \langle l_0, \ldots, l_i \rangle$ et $L_i := \langle l_{i+1}, \ldots, l_n \rangle$. Comme $\dim L_i = n - i - 1$ et $\dim B_i = i$ et que B_i et L_i ne se coupent pas alors on peut considérer la projection π_i de base B_i et de centre L_i.

π_{n-1} est la projection sur l'hyperplan B_{n-1} de centre l_n. La restriction de π_{n-2} à B_{n-1} est la projection de base B_{n-2} et de centre l_{n-1}. Plus généralement, la restriction de π_{n-i} à B_{n-i+1} est la projection de base B_{n-i} et de centre l_{n-i+1}. Comme $\pi_r = \pi_r \circ \cdots \circ \pi_{n-1}$, il suffit de prouver le théorème lorsque $r = n - 1$, ce que nous supposons désormais.

Sans perte de généralité nous pouvons effectuer un changement de variables linéaires pour nous ramener au cas où $B = \mathbf{V}(X_n)$ et $L = (0 : \cdots : 0 : 1)$. On note π la projection correspondante. Comme L n'est pas dans $\mathbf{V}(I)$, il existe un polynôme homogène $f_1 \in I$ ne s'annulant pas sur L. On pose $I' = I \cap k[X_0, \ldots, X_{n-1}]$ et on considère un point $a = (a_0 : \cdots : a_{n-1}) \in \mathbf{V}(I')$.

Soit $f \in I$ et considérons $g := \mathrm{Res}_{X_n}(f_1, f)$. Par la proposition 3 du Chapitre 6 on a $g \in I'$ et donc $g(a) = 0$. Par ailleurs, comme $f_1(L) \neq 0$, le degré partiel de f_1 en X_n coïncide avec son degré total et la Proposition 2 nous donne alors que $g(a)$ est proportionnel sur k à

$$\mathrm{Res}_{X_n}(f_1(a_0, \ldots, a_{n-1}, X_n), f(a_0, \ldots, a_{n-1}, X_n)),$$

ce qui implique que $f(a_0, \ldots, a_{n-1}, X_n)$ n'est pas constant. Finalement l'idéal

$$H := \{f(a_0, \ldots, a_{n-1}, X_n) \mid f \in I\}$$

de $k[X_n]$ est propre, ce qui nous permet de conclure que $\pi^{-1}(a) \cap \mathbf{V}(I)$ est non vide. □

Bibliographie

[1] Arrondo, Enrique (2006). « Another elementary proof of the Nullstellensatz ». In : *The American Mathematical Monthly*, vol. 113, n°2, p. 169–171.

CHAPITRE 24

Fonction et polynôme de Hilbert, dimension, degré

Résumé

Le polynôme de Hilbert est un objet combinatoire lié à un idéal. On peut lire sur ce polynôme des informations géométriques sur la variété correspondant à l'idéal comme sa dimension et son degré.

Sauf indication contraire, nous sommes dans ce chapitre dans le *contexte projectif*. L'anneau R est donc $k[X_0, \ldots, X_n]$.

1. Fonction et polynôme de Hilbert

Le *degré* d'un point $a = (a_0, \ldots, a_n)$ de $\mathbb{N}^{n+1}$ est l'entier $|a| = a_0 + \cdots + a_n$.

Définition 1 (Fonction de Hilbert). *Soit E une partie stable de $\mathbb{N}^{n+1}$; la fonction HF_E, qui associe à tout entier u le nombre de points de degré u n'appartenant pas à E, est appelée* fonction de Hilbert *du complémentaire de E :*

$$\mathrm{HF}_E(u) = \#\{a \in \mathbb{N}^{n+1} \mid a \notin E, |a| = u\}.$$

On étend HF_E aux entiers négatifs par la valeur 0.

Théorème 1 (Hilbert). *Pour u assez grand, la fonction HF_E est égale à un polynôme HP_E (dit de Hilbert). De plus, il existe des entiers $c_0, \ldots, c_d$ tels que :*

$$\mathrm{HP}_E(u) = \sum_{i=0}^{d} c_i \binom{u}{d-i}$$

où $\binom{u}{r} := \dfrac{u(u-1)(u-2)\cdots(u-r+1)}{r!}$ est la fonction binomiale.

Si u est strictement négatif, alors on a $\binom{u}{r} := (-1)^r \frac{(-u)(-u+1)(-u+2)\cdots(-u+r-1)}{r!} = (-1)^r \binom{-u+r-1}{r}$. Par conséquent $\binom{u}{r} \in \mathbb{Z}$, et donc $\mathrm{HP}_E(u) \in \mathbb{Z}$ dès que $u \in \mathbb{Z}$.

Définition 2. *Avec les notations précédentes, d est la* dimension *de E, et c_0 son* degré. *Nous attribuons par convention le degré -1 au polynôme nul.*

Remarquons que le coefficient de tête du polynôme HP_E est $c_0/d!$.

Définition 3 (Régularité et degré maximum). *La* régularité *$H(E)$ de la fonction de Hilbert est le plus petit entier à partir duquel la fonction de Hilbert coïncide avec le polynôme de Hilbert. Si E est non vide, nous notons $D(E)$ le degré maximum des éléments engendrant minimalement E.*

Toutes ces définitions s'étendent directement aux idéaux homogènes et aux variétés algébriques projectives.

Définition 4. *Soient $R = k[X_0, \ldots, X_n]$, $<$ un ordre monomial admissible sur R. Soit $R' = R/I$ un quotient de R par un idéal I homogène. On définit alors $\mathrm{HF}_{R'} := \mathrm{HF}_{E_<(I)}$ la* fonction de Hilbert *du quotient R'. Cette définition est indépendante de l'ordre choisi par le Corollaire 4 du Chapitre 21.*

2. Démonstrations et borne supérieure de la régularité

Dans cette section, pour simplifier les notations, on pose $e := D(E)$, et nous prouvons le Théorème 1, ainsi que la borne supérieure suivante par récurrence sur n :

$$H(E) \leq (n+1)(e-1)+1. \tag{1}$$

L'assertion est claire pour $n = 0$. Supposons l'hypothèse de récurrence satisfaite jusqu'à $n-1$. La section E_i de E par l'hyperplan $\{x_n = i\}$ est constante dès que $i \geq e$. Considérons le développement suivant de la fonction de Hilbert :

$$\mathrm{HF}_E(u) = \sum_{i=0}^{u} \mathrm{HF}_{E_i}(u-i).$$

Cette dernière somme se décompose en trois termes de la façon suivante :

$$\begin{aligned}
\mathrm{HF}_E(u) &= \sum_{0\leq i\leq e-1} \mathrm{HF}_{E_i}(u-i) + \sum_{e\leq i\leq u-H(E_e)} \mathrm{HF}_{E_e}(u-i) \\
&\quad + \sum_{u-H(E_e)+1\leq i\leq u} \mathrm{HF}_{E_e}(u-i) \\
&= \sum_{0\leq i\leq e-1} \mathrm{HF}_{E_i}(u-i) + \sum_{H(E_e)\leq i\leq u-e} \mathrm{HF}_{E_e}(i) + \sum_{0\leq i\leq H(E_e)-1} \mathrm{HF}_{E_e}(i) \\
&= \sum_{0\leq i\leq e-1} \mathrm{HF}_{E_i}(u-i) + \sum_{H(E_e)\leq i\leq u-e} \mathrm{HP}_{E_e}(i) + \sum_{0\leq i\leq H(E_e)-1} \mathrm{HF}_{E_e}(i),
\end{aligned}$$

où la dernière égalité provient du fait que $H(E_e) \leq i$ dans la deuxième somme. La troisième somme se comporte comme une constante. Il suffit donc d'analyser le comportement des deux premières.

L'hypothèse de récurrence implique $H(E_i) \leq n(e-1)+1$. Par conséquent, dès que $u \geq (n+1)(e-1)+1 \geq \max\{i + H(E_i) \mid 0 \leq i \leq e-1\}$, on a $\mathrm{HF}_{E_i}(u-i) = \mathrm{HP}_{E_i}(u-i)$, et donc la première somme coïncide avec le polynôme $\sum_{0\leq i\leq e-1} \mathrm{HP}_{E_i}(u-i)$, qui envoie $\mathbb{Z}$ dans $\mathbb{Z}$.

Par ailleurs, le Lemme 1 ci-dessous nous assure que la deuxième somme est un polynôme qui envoie $\mathbb{Z}$ dans $\mathbb{Z}$. On en déduit que $\mathrm{HF}_E(u)$ coïncide avec une fonction polynomiale dès que $u \geq (n+1)(e-1)+1$.

Lemme 1. *Soit f un polynôme de $k[X]$ de degré $d \geq 0$. La fonction F de $\mathbb{Z}$ dans k définie par $F(i) := \sum_{j=0}^{i} f(j)$ si $i \geq 0$, et $F(i) := -\sum_{j=i+1}^{-1} f(j)$ si $i < 0$, coïncide avec un polynôme de degré $d+1$.*

Démonstration. La preuve peut se faire par récurrence sur d. Si $d = 0$ alors f est une constante et le lemme est vrai. Supposons le lemme vrai jusqu'un degré $d-1 \geq 0$. On peut supposer que f est le monôme X^d. Comme

$$F(i) - \frac{(i+1)^{d+1}}{d+1} = \sum_{j=0}^{i} \left(j^d - \frac{(j+1)^{d+1}}{d+1} + \frac{j^{d+1}}{d+1} \right), \text{ si } i \geq 0, \text{ et}$$

$$F(i) - \frac{(i+1)^{d+1}}{d+1} = -\sum_{j=i+1}^{-1} \left(j^d - \frac{(j+1)^{d+1}}{d+1} + \frac{j^{d+1}}{d+1} \right), \text{ si } i < 0,$$

et comme $j^d - \frac{(j+1)^{d+1}}{d+1} + \frac{j^{d+1}}{d+1}$ est un polynôme en j de degré au plus $d-1$, l'hypothèse de récurrence implique que les membres droits des précédentes égalités coïncident avec un même polynôme de degré au plus d. Par conséquent F coïncide bien avec un polynôme de degré $d+1$. □

Comme $HP_E(0) = c_d$, alors c_d est entier. Comme $HP_E(1) = c_d + c_{d-1}$, alors c_{d-1} est entier. Comme $HP_E(2) = c_d + 2c_{d-1} + c_{d-2}$, alors c_{d-2} est entier. Plus généralement, si i est un entier inférieur à d on a $\mathrm{HP}_E(i) = \sum_{j=d-i+1}^{d} c_j \binom{i}{d-j} + c_{d-i}$, et donc, par récurrence, tous les c_i sont entiers.

EXEMPLE 1. Étant donnés $n+1$ entiers positifs $a_0, \ldots, a_n$, considérons l'idéal $(X_0^{a_0}, \ldots, X_n^{a_n})$, dont ces générateurs monomiaux forment évidemment une base standard. La partie stable associée E est le complément d'un parallélépipède dont l'élément de degré maximal est $(a_0 - 1, \ldots, a_n - 1)$. La régularité est donc $1 + \sum_{i=0}^{n}(a_i - 1)$; dans le cas où tous les a_i sont égaux, ceci prouve que la borne ci-dessus peut être atteinte.

3. Suites régulières

DÉFINITION 5. *Soit R un anneau. On dit qu'un élément $a \in R$ est un* diviseur de zéro *dans R s'il existe $d \in R \setminus \{0\}$ tel que $ad = 0$. Ceci est équivalent au fait que l'endomorphisme de multiplication par a dans R ne soit pas injectif.*

THÉORÈME 2. *Soit I un idéal homogène de dimension r, de degré δ, et de régularité e. Soit f un polynôme homogène de degré d non diviseur de zéro dans R/I. Alors $I + (f)$ a pour dimension $r-1$, pour degré $d\delta$, et sa régularité est bornée par $e + d$.*

DÉMONSTRATION. Notons $J = I + (f)$, et considérons la suite exacte (l'image de chaque morphisme d'anneaux est le noyau du suivant) suivante :

$$0 \to R/I \xrightarrow{\varphi} R/I \to R/J \to 0,$$

où φ représente l'endomorphisme de multiplication par f. Comme I est homogène, l'anneau R/I hérite de la graduation de R. La suite exacte respecte cette graduation, *i.e.* pour tout entier j la suite suivante est exacte :

$$0 \to (R/I)_{j-d} \xrightarrow{\varphi} (R/I)_j \to (R/J)_j \to 0.$$

Ainsi on a $\mathrm{HF}_{R/J}(u) = \mathrm{HF}_{R/I}(u) - \mathrm{HF}_{R/I}(u-d)$. On en déduit que $\mathrm{HP}_{R/J}(u) = \mathrm{HP}_{R/I}(u) - \mathrm{HP}_{R/I}(u-d)$ et que la régularité de R/J est bornée par $e+d$. À partir de $\mathrm{HP}_{R/I}(u) = \delta \frac{u^r}{r!} + cu^{r-1} + \cdots$, pour un certain $c \in \mathbb{Q}$, on obtient :

$$\begin{aligned}\mathrm{HP}_{R/J}(u) &= \frac{\delta}{r!}\left[u^r - (u-d)^r\right] + c\left[u^{r-1} - (u-d)^{r-1}\right] + \cdots \\ &= d\delta\frac{u^{r-1}}{(r-1)!} + c'u^{r-2} + \cdots\end{aligned}$$

pour un certain $c' \in \mathbb{Q}$, ce qui conclut la preuve. □

DÉFINITION 6. *Soit $f_1, \ldots, f_s$ une suite de polynômes. La suite est dite* régulière *si pour tout $i \in \{1, \ldots, s\}$, la classe de f_i dans $k[X_0, \ldots, X_n]/(f_1, \ldots, f_{i-1})$ n'est pas un diviseur de zéro.*

COROLLAIRE 1. *Soit I un idéal homogène engendré par une suite régulière homogène $f_1, \ldots, f_s$ de degrés respectifs $d_1, \ldots, d_s$. Alors I a pour dimension $n-s$, pour degré $\prod_{i=1}^{s} d_i$ et une régularité majorée par $(\sum_{i=1}^{s} d_i) - n$.*

DÉMONSTRATION. Si $s = 0$ alors $R/I = R$ et

$$\mathrm{HF}_R(u) = \begin{cases} \binom{u+n}{n} & \text{pour } u \text{ positif} \\ 0 & \text{pour } u \text{ négatif.} \end{cases}$$

La dimension de R est donc n, son degré est 1, et sa régularité est $-n$ car HF_R coïncide avec $\mathrm{HP}_R(u) = \binom{u+n}{n}$ si, et seulement si, $u \geq -n$. Le résultat est finalement une conséquence du théorème précédent. □

4. Autres définitions de la dimension

Si $V = \mathbf{V}(I)$ est une variété projective alors sa dimension et son degré sont définis comme étant ceux de I. Dans cette section, nous supposons que k est algébriquement clos, et nous donnons des interprétations de la dimension en termes géométriques et calculatoires.

DÉFINITION 7. *La* dimension de section *est le plus petit entier σ tel qu'il existe une sous-variété linéaire de codimension $\sigma+1$ n'intersectant pas $\mathbf{V}(I)$. La* dimension de projection *est le plus grand entier π tel qu'il existe une projection envoyant surjectivement $\mathbf{V}(I)$ sur une base de dimension π.*

DÉFINITION 8 (Dimension lexicographique inférieure (resp. supérieure)). *Soit x un système de coordonnées de $\mathbb{P}^n$. Nous appellerons $linf_x$ (resp. $lsup_x$) le plus petit (resp. le plus grand) entier e tel que $E_x(I)$ relativement à l'ordre* `tdeg` *(resp.* `plex`*) rencontre les derniers $n-e$ axes de coordonnées de $\mathbb{N}^{n+1}$ (resp. ne rencontre pas le plan des $e+1$ premières coordonnées).*

Le minimum linf des $linf_x$ (resp. le maximum lsup des $lsup_x$) pris sur tous les systèmes de coordonnées est appelé la dimension lexicographique inférieure (resp. supérieure).

THÉORÈME 3 (Dimension). *Pour tout idéal homogène I de R, la dimension de I coïncide avec sa dimension de section, sa dimension de projection, sa dimension lexicographique inférieure, et sa dimension lexicographique supérieure.*

Remarquons que le théorème peut-être vérifié aisément lorsque I est engendré par des polynômes de degré 1, par des techniques classiques d'algèbre linéaire, telles que rappelées au début du prochain chapitre. Par ailleurs, la Proposition 4 du Chapitre 23 implique le théorème lorsque I est zéro-dimensionnel.

DÉMONSTRATION. Notons r la dimension de I, et prouvons tout d'abord que $r \geq lsup$. Supposons qu'il existe des coordonnées de $\mathbb{P}^n$ pour lesquelles, relativement à un ordre monomial admissible, par exemple l'ordre lexicographique, $E(I)$ ne rencontre pas le plan des $e+1$ premières coordonnées. Alors la fonction de Hilbert $\mathrm{HF}_{R/I}(u)$ est minorée par le nombre de monômes de degré u en $e+1$ variables, qui est équivalent à $u^e/e!$ quand u tend vers l'infini.

Prouvons maintenant que $lsup \geq \pi$. Supposons que $\mathbf{V}(I)$ puisse être projetée surjectivement sur une base B de dimension b. Nous pouvons choisir des coordonnées telles que B soit définie par les équations $X_{b+1} = \cdots = X_n = 0$. Ainsi l'idéal $I \cap k[X_0, \ldots, X_b]$ se réduit à 0 ; car sinon il existerait un polynôme non nul $f(X_0, \ldots, X_b)$ dans I qui ne s'annulerait pas sur tout B, et $\mathbf{V}(I)$ ne pourrait pas se projeter surjectivement sur B. Par rapport à de telles coordonnées, la partie stable $E(I)$ relativement à l'ordre `plex` ne peut rencontrer le plan des $b+1$ premières variables d'après la Proposition 2 du Chapitre 22.

Montrons maintenant que $\pi \geq \sigma$. Par définition de σ, il existe une sous-variété linéaire L de codimension $\sigma+1$ évitant $\mathbf{V}(I)$. Par ailleurs nous pouvons choisir une sous-variété linéaire B de dimension σ évitant L. La projection de centre L envoie surjectivement $\mathbf{V}(I)$ sur B, puisque σ est minimal ; car si b est un point de B, la sous-variété linéaire qu'il engendre avec L est de codimension σ, donc coupe $\mathbf{V}(I)$ en au moins un point qui se projette sur b.

Montrons maintenant que $\sigma \geq linf$. Par définition de σ, il existe une sous-variété linéaire L de codimension $\sigma+1$ évitant $\mathbf{V}(I)$. Nous pouvons choisir des coordonnées telles que L soit définie par les équations $X_0 = \cdots = X_\sigma = 0$. L'idéal $J = I + (X_0, \ldots, X_\sigma)$ définit la variété vide, donc il contient une puissance de l'idéal maximal $(X_0, \ldots, X_n)$ par le Théorème 3 du Chapitre 23. Quelque soit l'ordre, en

particulier l'ordre `tdeg`, l'escalier $E(J)$ coupe tous les axes de coordonnées. Considérons un polynôme dont le monôme dominant est sur un des axes $X_{\sigma+1}, \ldots, X_n$; il est congru modulo $(X_0, \ldots, X_\sigma)$ à un polynôme de I, non nul et de même monôme dominant. Par conséquent la section de $E(I)$ par le plan des $n - \sigma$ dernières coordonnées recoupe tous les axes.

Montrons maintenant que $linf \geq r$. Supposons qu'il y ait des coordonnées de $\mathbb{P}^n$ telles que, relativement à l'ordre `tdeg`, l'escalier $E(I)$ rencontre les derniers $n - linf$ axes. En degré u suffisamment grand, le complémentaire de $E(I)$ ne contient que des monômes dont les $n - linf$ derniers exposants sont majorés par un entier fixé. À une constante multiplicative près, la fonction de Hilbert $\mathrm{HF}(u)$ est donc majorée par le nombre de monômes en $linf + 1$ variables, soit par $O(u^{linf})$ quand u tend vers l'infini. □

Notes

Soit I un idéal engendré par des polynômes de degré inférieur à d. Une fois construite une base standard, on peut calculer la dimension de Hilbert. Mais il existe une famille d'idéaux, donnés par des générateurs de degré au plus d, dont le degré des éléments d'une base standard est minoré par Cd^{a^n}, $C > 0$, $a > 1$ (exemple de [3], voir aussi [1], ou [2, Chapitre 21] pour un résumé de tous les pires cas). Cependant, il y a des classes d'idéaux qui n'atteignent pas cette borne de pire complexité. Par exemple, si I est un idéal engendré par une suite régulière $(f_1, \ldots, f_s)$ de degrés majorés par d, alors on peut montrer qu'il existe un changement de coordonnées tel que la base standard de I pour `tdeg` vérifie $D(E(I)) \leq (n+1)d - n$. On peut donc attendre des bases standard pour `tdeg` que leur degré maximal soit généralement linéaire en n, ce qui est bien moins que la borne précédente de pire degré.

Bibliographie

[1] Demazure, M. (1987). « Le théorème de complexité de Mayr et Meyer ». In : *Géométrie algébrique et applications, I (La Rábida, 1984)*. Vol. 22. Travaux en Cours. Paris : Hermann, p. 35–58.

[2] Gathen, Joachim von zur et Jürgen Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press, xiv+785 pages.

[3] Mayr, Ernst W. et Albert R. Meyer (1982). « The complexity of the word problems for commutative semigroups and polynomial ideals ». In : *Advances in Mathematics*, vol. 46, n°3, p. 305–329.

CHAPITRE 25

Normalisation de Noether

Résumé

Étant donné un système d'équations linéaires homogènes, les formules de Cramer permettent de paramétrer les solutions en fonction d'un certain nombre de variables que l'on peut choisir arbitrairement. Le lemme de normalisation de Noether fournit un résultat analogue pour des systèmes d'équations polynomiales.

1. La situation linéaire

Soient $f_1, \dots, f_s$ des formes linéaires homogènes à $n+1$ variables $X_0, \dots, X_n$, à coefficients dans un corps k :

$$f_i(X_0, \dots, X_n) = \sum_{j=0}^{n} f_{ij} X_j.$$

1.1. Point de vue de l'algèbre. On peut supposer, quitte à en enlever certaines, que les formes linéaires $f_1, \dots, f_s$ sont linéairement indépendantes (et donc forment une suite régulière). Comme la $s \times (n+1)$ matrice F qui leur est associée est de rang s, il existe un mineur M non nul de taille s. On pose $r = n - s$. Quitte à renuméroter les variables, nous pouvons supposer que F s'écrit

$$F = \begin{array}{c} \\ 1 \\ \vdots \\ s \end{array} \begin{array}{c} \begin{array}{cc} 0 \dots r & r+1 \dots n \end{array} \\ \begin{pmatrix} N & M \end{pmatrix} \end{array}$$

avec $\det M \neq 0$. Alors nous avons l'équivalence

$$\begin{aligned} F \begin{pmatrix} X_0 \\ \vdots \\ X_n \end{pmatrix} = 0 \quad &\Longleftrightarrow \quad M \begin{pmatrix} X_{r+1} \\ \vdots \\ X_n \end{pmatrix} = -N \begin{pmatrix} X_0 \\ \vdots \\ X_r \end{pmatrix} \\ &\Longleftrightarrow \quad \begin{pmatrix} X_{r+1} \\ \vdots \\ X_n \end{pmatrix} = -M^{-1} N \begin{pmatrix} X_0 \\ \vdots \\ X_r \end{pmatrix}. \end{aligned}$$

Ainsi, les $n - r$ dernières coordonnées, que nous appellerons *variables liées*, sont données en fonction des $r+1$ premières, que nous appellerons *variables libres*. L'espace vectoriel des solutions devient ainsi le graphe d'une application linéaire $k^{r+1} \to k^s$.

1.2. Point de vue de la géométrie. L'interprétation géométrique des formules de Cramer est la suivante : si le rang de la matrice F est s, nous pouvons partitionner l'ensemble des variables en deux paquets, $X_0, \dots, X_r$ et $X_{r+1}, \dots, X_n$ après renumérotation. Définissons les deux sous-variétés linéaires projectives B d'équations $X_{r+1} = \cdots = X_n = 0$ et L d'équations $X_0 = \cdots = X_r = 0$. La sous-variété

linéaire d'équations $f_1 = \cdots = f_s = 0$ ne coupe pas L, et la projection de centre L l'envoie surjectivement sur B.

1.3. Point de vue de l'algorithmique. À partir de la matrice F, l'algorithme du pivot de Gauss permet d'exhiber un mineur non nul de taille maximale, avec un nombre d'opérations arithmétiques polynomial en s, n. Ces questions ont été abordées au Chapitre 8.

2. La situation générale

Notons $R := k[X_0, \ldots, X_n]$, et soient $f_1, \ldots, f_s$ des polynômes homogènes de R de degrés $d_1, \ldots, d_s$:

$$f_i(X_0, \ldots, X_n) = \sum_{|a|=d_i} f_{i,a} X_0^{a_0} \cdots X_n^{a_n}$$

où $a = (a_0, \ldots, a_n)$ représente un élément de $\mathbb{N}^{n+1}$ de degré $|a| = a_0 + \cdots + a_n$.

2.1. Point de vue de l'algèbre. L'anneau $R = k[X_0, \ldots, X_n]$ peut être muni d'une structure d'anneau gradué par la décomposition canonique $R = \bigoplus_{s\geq 0} R_s$ où R_s est l'ensemble des polynômes homogènes de degré s. L'idéal I de R engendré par les formes $f_1, \ldots, f_s$ est *homogène* et est donc gradué par $I = \bigoplus_{s\geq 0}(I \cap R_s)$, d'après la Proposition 1 du Chapitre 21.

Définition 1. *Soient $A \subseteq B$ deux anneaux. On dit que $b \in B$ est* entier *sur A si il existe $P \in A[X]$ unitaire tel que $P(b) = 0$. L'extension $A \subseteq B$ est en* entière *si tout $b \in B$ est entier sur A.*

Proposition 1. *Soient $A \subseteq B$ et $b, c \in B$. Si b est entier sur A et que c est entier sur l'anneau $A[b]$, alors c est entier sur A.*

Démonstration. Considérons les relations de dépendance intégrale $P(b) = b^q + \sum_{i=0}^{q-1} a_i b^i = 0$ et $Q(c) = c^t + \sum_{j=0}^{t-1} A_j(b) c^j = 0$ avec $a_i \in A$ et $A_j \in A[X]$. Considérons ces relations comme des polynômes à coefficients dans $A[c]$ évalués en b. Leur résultant $\mathrm{Res}(P, Q)$ est nul car ils ont b comme racine commune. Dans le développement du déterminant de la matrice de Sylvester, la diagonale produit le terme de plus grand degré, en l'occurrence tq, en c. Ce résultant est donc une relation unitaire à coefficients dans A. □

Proposition 2 (Version algébrique du lemme de normalisation de Noether). *Soient I un idéal homogène de R. Si k est infini, alors il existe un entier r et un changement de variables linéaire inversible tel que, une fois appliqué à I, on obtienne les propriétés suivantes :*

- $I \cap k[X_0, \ldots, X_r] = (0)$,
- *R/I est une extension entière de $k[X_0, \ldots, X_r]$.*

Dit autrement, nous pouvons, après changement de variables, partitionner les variables en deux paquets : les $r+1$ premières seront dites *libres* et les $n-r$ dernières *liées*.

Démonstration. Pour $i \in \{1, \ldots, n\}$, notons M_i la matrice carrée de taille $i+1$ définie par

$$M_i := \begin{pmatrix} 1 & 0 & \cdots & 0 & a_{i,0} \\ 0 & 1 & \cdots & 0 & a_{i,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & a_{i,i-1} \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Posons $A_i := k[X_0, \ldots, X_i]$, et $J_i := I \cap A_i$. Nous allons prouver par récurrence descendante sur i, à partir de $i = n$, qu'il existe un changement de variables linéaire inversible qui ou bien rend R/I entier sur la projection de A_{i-1} dans R/I, ou bien donne $J_i = (0)$.

Si $J_n = (0)$ alors l'hypothèse de récurrence est bien vérifiée pour $i = n$. Sinon, il existe un polynôme homogène f non nul dans J_n. Par la Proposition 1 du Chapitre 23, il existe des valeurs $a_{n,0}, \ldots, a_{n,n-1}$ de k telles que le changement de variables $(X_0, \ldots, X_n)^t \leftarrow M_n(X_0, \ldots, X_n)^t$ rende f unitaire en X_n. Supposons un tel changement de variable désormais effectué. L'hypothèse de récurrence est bien vérifiée pour $i = n$. Supposons l'hypothèse de récurrence vérifiée pour un certain entier $i + 1$ tel que $1 \leq i + 1 \leq n$.

Si $J_i = (0)$, alors l'hypothèse de récurrence est bien satisfaite pour i. Sinon, il existe un polynôme homogène f non nul dans J_i. Par la Proposition 1 du Chapitre 23, il existe des valeurs $a_{i,0}, \ldots, a_{i,i-1}$ de k telles que le changement de variables $(X_0, \ldots, X_i)^t \leftarrow M_i(X_0, \ldots, X_i)^t$ rende f unitaire en X_i. Supposons un tel changement de variable désormais effectué. Comme le projeté de X_i dans R/I est entier sur A_{i-1}, et que R/I est entier sur le projeté de $A_i = A_{i-1}[X_i]$ dans R/I, la Proposition 1 nous donne que R/I est entier sur le projeté de A_{i-1} dans R/I. L'hypothèse de récurrence est alors aussi bien satisfaite pour i.

L'entier r de la proposition le plus grand entier i de cette récurrence tel que $I \cap A_i = (0)$. □

2.2. Point de vue de la géométrie. La version géométrique du lemme de normalisation de Noether dans le cadre projectif est la suivante :

PROPOSITION 3 (Version géométrique du lemme de normalisation de Noether). *Supposons k algébriquement clos. Pour tout idéal homogène I de dimension r, il existe deux sous-variétés linéaires L de codimension $r+1$ et B de dimension r telles que :*

- *$\mathbf{V}(I)$ et L ne se coupent pas,*
- *la restriction à $\mathbf{V}(I)$ de la projection π de centre L et de base B est surjective et finie, c'est-à-dire que pour tout point b de B, la fibre $\pi^{-1}(b)$ est constituée d'un nombre fini non nul de points.*

DÉMONSTRATION. Par le Théorème 3 du Chapitre 24, il existe une sous-variété L disjointe de $\mathbf{V}(I)$ de codimension minimale égale à $r + 1$. Choisissons pour B un espace de dimension r ne coupant pas L. Soit π la projection de centre L et de base B.

Montrons d'abord, par l'absurde, que la restriction de π à $\mathbf{V}(I)$ est surjective. Si un point $b \in B$ n'est pas une projection alors l'espace linéaire projectif $L + b$ ne coupe pas $\mathbf{V}(I)$. Cet espace est de codimension r ce qui contredit le Théorème 3 du Chapitre 24.

Montrons enfin que la fibre $\pi^{-1}(b) = \mathbf{V}(I) \cap (L + b)$ est finie, c'est à dire de dimension zéro. Définissons W comme $\pi^{-1}(b)$ en tant que sous-variété de $L + b$. Comme W n'intersecte par L, et que L est de codimension 1 dans $L + b$, le Théorème 3 du Chapitre 24 implique que la dimension de W est au plus zéro. □

La proposition suivante montre que la version algébrique de la normalisation de Noether implique la version géométrique.

PROPOSITION 4. *Supposons k algébriquement clos. Soient I un idéal homogène de R et r un entier, tels que $I \cap k[X_0, \ldots, X_r] = (0)$ et R/I est une extension entière de $k[X_0, \ldots, X_r]$. Soient $L := \mathbf{V}(X_0, \ldots, X_r)$, $B := \mathbf{V}(X_{r+1}, \ldots, X_n)$, et π la projection de centre L et de base B. Alors $\mathbf{V}(I)$ n'intersecte pas L, la restriction de π à $\mathbf{V}(I)$ est surjective, et r est égal à la dimension de I.*

DÉMONSTRATION. Comme R/I est entier sur $k[X_0,\dots,X_r]$, pour tout $j \geq r+1$, il existe un entier m_j tel que $X_j^{m_j} \in I + (X_0,\dots,X_r)$. Par conséquent $\mathbf{V}(I)$ n'intersecte pas L. Comme $X_0,\dots,X_r$ forment une suite régulière, la dimension de L est $n-r-1$ par le Corollaire 1 du Chapitre 24. Le Théorème 3 du Chapitre 24 implique que I est de dimension au plus r.

Afin de montrer que π envoie $\mathbf{V}(I)$ surjectivement sur B nous pouvons nous restreindre à prouver que $\pi^{-1}(b)$ est non vide pour $b = (0 : \cdots : 0 : 1)$, quitte à effectuer un changement de variables linéaire inversible à $X_0,\dots,X_r$. Soit J l'idéal de $\mathbf{V}(I) \cap (L+b)$, et soit f un polynôme de J. Notons f_j une relation entière pour l'image de X_j dans R/I, à coefficients dans $k[X_0,\dots,X_{j-1}]$, pour chaque $j \geq r+1$, et considérons

$$g := \mathrm{Res}_{X_{r+1}}(f_{r+1}, \mathrm{Res}_{X_{r+2}}(f_{r+2}, \mathrm{Res}_{X_{r+3}}(\dots, \mathrm{Res}_{X_n}(f_n, f)))).$$

En appliquant plusieurs fois la Proposition 3 du Chapitre 6, on a $g \in I \cap k[X_0,\dots,X_r]$, ce qui implique que g est nul. Par ailleurs, en appliquant plusieurs fois la Proposition 2 du Chapitre 23, on déduit que $f(b, X_{r+1},\dots,X_n)$ est non constant. Par conséquent J ne contient pas de puissance de l'idéal $(X_0,\dots,X_r)$, et le Théorème 3 du Chapitre 23 implique que $\mathbf{V}(J) \neq \varnothing$. La restriction de π à $\mathbf{V}(I)$ est donc bien surjective et le Théorème 3 du Chapitre 24 implique que I est de dimension au moins r. □

2.3. Point de vue de l'algorithmique. Si M est une matrice carrée de taille $n+1$ à coefficients dans k, et si $f \in k[X_0,\dots,X_n]$, on définit $f \circ M(X_0,\dots,X_n) := f(M(X_0,\dots,X_n)^t)$. Si I est un idéal alors on note $I \circ M := (f \circ M \mid f \in I)$. L'ensemble des matrices carrées de taille $n+1$ sur k qui sont triangulaires supérieures avec des 1 sur la diagonale est noté $\mathcal{T}_{n+1}(k)$. Dans la suite nous utiliserons définition suivante :

DÉFINITION 2. *Un idéal I homogène de R de dimension r est dit en* position de Noether *si $I \cap k[X_0,\dots,X_r] = (0)$, et R/I est une extension entière de $k[X_0,\dots,X_r]$.*

En suivant la preuve de la Proposition 2 on obtient l'algorithme de la Figure 1.

Mise en position de Noether

Entrée : un idéal I donné par un système de générateurs F.

Sortie : une matrice $M \in \mathcal{T}_{n+1}(k)$ telle que $I \circ M$ est en position de Noether, ainsi que la dimension r de I.

Initialiser i à n et M avec la matrice identité.

Tant que $(I \circ M) \cap k[X_0,\dots,X_i] \neq (0)$ répéter :

choisir $f \in (I \circ M) \cap k[X_0,\dots,X_i]$ non nul,

trouver $(a_{0,i},\dots,a_{i-1,i}) \in k^i \setminus \{0\}$ qui n'annule pas f,

remplacer M par MM_i et i par $i-1$.

Retourner M et i.

FIGURE 1. Algorithme de mise en position de Noether

PROPOSITION 5. *Si k est infini, l'algorithme de la Figure 1 termine et le résultat retourné est correct.*

DÉMONSTRATION. L'algorithme est calqué sur la preuve de la Proposition 2. Si k est infini alors il termine et retourne bien un changement de variable qui met l'idéal I en position de Noether. Le fait que l'entier i est égal à la dimension de I vient de la Proposition 4. □

La partie calculatoire coûteuse de l'algorithme de mise en position de Noether est le test $(I \circ M) \cap k[X_0, \ldots, X_i] \neq (0)$. Ce test peut être fait par un calcul de base standard en utilisant la Proposition 4 du Chapitre 22. Mentionnons qu'une technique plus performante, proposée dans [3], permet la mise en position de Noether avec un nombre d'opérations arithmétiques polynomial en d^{n^2}, où d est le maximum des degrés des générateurs F de I.

Une autre façon d'aborder la mise en position de Noether est d'adopter un point de vue probabiliste, à partir de la proposition suivante :

PROPOSITION 6. *Soit I un idéal homogène de R. Il existe une variété algébrique affine propre W de $\mathcal{T}_{n+1}(k)$ telle que pour toute matrice $M \notin W$, l'idéal $I \circ M$ est en position de Noether.*

DÉMONSTRATION. Le changement de variables M retourné par l'algorithme de la Figure 1 est $M = M_n M_{n-1} \cdots M_{r+1}$. Quels que soient les valeurs $a_{n,0}, \ldots, a_{n,n-1}$ n'annulant pas un polynôme homogène non nul de I, les valeurs $a_{n-1,0}, \ldots, a_{n-1,n-2}$ n'annulant pas un polynôme homogène non nul de $I \cap k[X_1, \ldots, X_{n-1}]$, …, les valeurs $a_{r+1,0}, \ldots, a_{r+1,r}$ n'annulant pas un polynôme homogène non nul de $I \cap k[X_1, \ldots, X_{r+1}]$, l'idéal $I \circ M$ est en position de Noether. Ensuite, il suffit de remarquer que pour de telles valeurs et pour n'importe quelles valeurs de $a_{r,0}, \ldots, a_{r,r-1}$, …, $a_{2,0}, a_{2,1}$, $a_{1,0}$ de k, l'idéal $I \circ M_n M_{n-1} \cdots M_1$ est toujours en position de Noether. □

COROLLAIRE 1. *Soit I un idéal homogène de R. Il existe une variété algébrique affine propre W de $\mathcal{M}_{n+1}(k)$ telle que pour toute matrice $M \notin W$, M est inversible et l'idéal $I \circ M$ est en position de Noether.*

DÉMONSTRATION. L'ensemble des matrices carrées de taille $n+1$ ayant un mineur principal nul est contenu dans une variété algébrique propre de $\mathcal{M}_{n+1}(k)$. Par ailleurs, il est classique que toute matrice M en dehors de cette dernière variété peut se décomposer en $M = LU$, où L est une matrice triangulaire inférieure et U est triangulaire supérieure avec des 1 sur la diagonale. Comme $I \circ L$ est en position de Noether si, et seulement si, I l'est, la conclusion provient de la proposition précédente. □

Si k est infini, ou en tout cas suffisamment grand, on peut tirer au hasard un changement de variables qui met I en position de Noether avec une forte probabilité. L'étude de cette probabilité dépasse le cadre de ce cours et nous renvoyons par exemple à l'article [4] pour plus de détails.

3. Test à zéro d'un polynôme

Pour pouvoir tourner les coordonnées dans l'algorithme de normalisation de Noether de la précédente section, il faut trouver un point qui n'annule pas un polynôme non nul. Le problème inverse consiste à tester la nullité d'un polynôme en l'évaluant en différents points. Nous allons nous intéresser à la complexité de ces problèmes.

3.1. Ensemble questeur. Nous mentionnons ici quelques résultats connus pour tester si un polynôme est identiquement nul en l'évaluant sur plusieurs points.

DÉFINITION 3. *Soient $P \subseteq k[X_1, \ldots, X_n]$ un ensemble de polynômes et $Q \subseteq k^n$. On dit que l'ensemble Q est* questeur *pour P si aucun polynôme de P non nul ne s'annule identiquement sur Q. Ceci revient à dire que l'évaluation sur Q est un test à zéro pour les polynômes de P.*

PROPOSITION 7. *Soient $E_1, \ldots, E_n \subseteq k$ des parties de cardinal supérieur à $d+1$. Alors $Q := E_1 \times \cdots \times E_n \subseteq k^n$ est un ensemble questeur pour les polynômes de $k[X_1, \ldots, X_n]$ de degré au plus d.*

DÉMONSTRATION. Procédons par récurrence sur n. Pour $n = 1$, un polynôme de degré au plus d qui admet $d+1$ racines est nul. Remarquons que ceci est vrai généralement pour les polynômes à coefficients dans un anneau unitaire car la démonstration repose sur la division euclidienne par $X - x$ si x est une racine. Supposons le résultat vrai au rang $n-1$. Soit $p \in P$, écrivons-le $p = \sum p_i X_n^i$ avec $p_i \in k[X_0, \ldots, X_{n-1}]$. Soit $x \in E_n$, alors $p(X_1, \ldots, X_{n-1}, x)$ est nul car il s'annule sur $E_1 \times \cdots \times E_{n-1}$. Le polynôme p a au moins $d+1$ racines en X_n donc il est nul. □

Le problème de l'ensemble questeur de la précédente proposition est sa taille exponentielle en n. On peut trouver d'autres ensembles questeurs plus adaptés à la structure d'un polynôme. Par exemple, soit $A \subseteq \mathbb{N}^n$, et considérons $W_A := \{\sum_{a \in A} c_a X^a \mid c_a \in k\}$, l'ensemble des polynômes de $k[X_1, \ldots, X_n]$ à support dans A. On a le résultat suivant :

THÉORÈME 1 (Risler, Ronga [8]). *L'ensemble*

$$2^{\{A\}} = \{(2^{a_1}, \ldots, 2^{a_n}) \mid (a_1, \ldots, a_n) \in A\}$$

est questeur pour W_A.

Par ailleurs, mentionnons que l'on conjecture que A est un ensemble questeur pour W_A.

3.2. Structures de données pour les polynômes. La représentation dense consiste à se donner un polynôme par son degré et par la liste de tous les coefficients des monômes de degré plus petits. Dans cette représentation, le test à zéro s'effectue en testant à zéro chaque coefficient. Le coût du test est donc proportionnel à la taille du codage du polynôme. Par exemple, pour un polynôme de degré d en n variables, on devra coder $\binom{n+d}{d-1}$ coefficients.

Or, même dans le cas simple des suites régulières, si on a en entrée n polynômes de degré d, un calcul de base standard fait intervenir des polynômes de degré de l'ordre de grandeur de d^n. Le nombre de monômes en ce degré est de l'ordre de grandeur de d^{n^2}. Pour pallier ce problème nous allons introduire une autre structure de données. L'objectif est d'avoir des algorithmes de complexité polynomiale en d^n pour les tâches usuelles de la géométrie algébrique.

Nous allons considérer un modèle théorique de calcul. Les DAG, pour *Directed Acyclic Graphs* en anglais et graphes acycliques orientés en français, permettent de représenter un calcul en partageant les sous-expressions communes. Les nœuds sont les opérations arithmétiques et les feuilles sont des constantes ou des entrées. Nous renvoyons à [2] pour les définitions précises et plus de détails sur le sujet. Soit donc un DAG ayant les propriétés suivantes : les feuilles de l'arbre sont soit des éléments de k soit des variables X_i. Les noeuds possibles du graphe sont des noeuds binaires qui effectuent une opération arithmétique $(+, -, \times, /)$. Les noeuds sans parent sont les sorties du calcul. La taille du graphe définit la *complexité d'évaluation* du calcul.

EXEMPLE 1. Tout polynôme peut être représenté par un DAG. Par exemple, le schéma de Horner consiste à calculer $P = \sum a_i X^i$ par $P = ((a_n X + a_{n-1})X +$

$a_{n-2}\dots)X + a_0$. Il code tout polynôme à une variable de degré d par un DAG de complexité d'évaluation en $O(d)$. À l'inverse, le polynôme $(X+1)^{2^n}$ se code beaucoup mieux par le DAG $(\cdots(((X+1)^2)^2)^2\cdots)^2$ que par sa représentation sur la base monomiale.

Définition 4. *La complexité d'évaluation d'un polynôme est le minimum des complexités d'évaluation des DAG qui le représentent.*

Exemple 2. Soit $M = (X_{i,j})$ la matrice générique carré de taille n. Alors le polynôme déterminant de M se code mal par sa représentation sur la base monomiale car il a $n!$ monômes. Par contre, il se code en complexité d'évaluation $O(n^4)$ par l'algorithme de Berkowitz [1]. Cette exemple est important car il illustre que les polynômes d'élimination, dont le déterminant est un bon représentant, s'évaluent bien.

Le problème des DAG est le test à zéro. Nous allons voir qu'il existe des ensembles questeurs de tailles polynomiales. Cependant ils peuvent être difficiles à construire.

Théorème 2 (Heintz, Schnorr [7]). *Soit $W(D,n,L)$ l'ensemble des polynômes en n variables de degré au plus D et de complexité d'évaluation au plus L. Fixons $\Gamma \subset k$ de cardinal $2L(D+1)^2$ et $m := 6(L+1)(L+n+1)$. Soient $\tau(D,n,L,\Gamma)$ l'ensemble des m-uplets de n-uplets d'éléments de Γ et $\xi(D,n,L,\Gamma)$ celui des éléments de τ qui ne sont pas questeurs. Alors on a :*

$$\frac{|\xi(D,n,L,\Gamma)|}{|\Gamma|^{nm}} \leq \frac{1}{|\Gamma|^{m/6}}.$$

Le membre droit est majoré par 1/264000 dans les cas non triviaux.

Les algorithmes utilisant des DAG font alors intervenir deux types de complexité. D'une façon informelle, la complexité non uniforme revient à dire, « au prix d'un précalcul, la complexité est … ». Ce précalcul revient dans le cas présent à trouver un ensemble questeur. L'autre complexité, dite probabiliste, revient à dire « si on choisit un ensemble questeur au hasard … ». Dans l'article [5], il est décrit un algorithme de mise en position de Noether de complexité non uniforme polynomiale en d^n. Une implémentation est proposée dans [4].

Finalement une implémentation plus générale, connue sous le nom d'algorithme de résolution géométrique, se trouve dans [6]. En utilisant la mise en position de Noether, cet algorithme permet de résoudre des systèmes d'équations polynomiales sans calculer de bases standard. Nous en présentons une version simplifiée dans le chapitre suivant.

Notes

Le point de vue utilisé dans ce chapitre pour la normalisation de Noether provient essentiellement des deux articles [3] et [5].

Bibliographie

[1] Berkowitz, Stuart J. (1984). « On Computing the determinant in small parallel time using a small number of processors ». In : *Information Processing Letters*, vol. 18, p. 147–150.

[2] Bürgisser, Peter, Michael Clausen et M. Amin Shokrollahi (1997). *Algebraic complexity theory*. Vol. 315. Grundlehren der Mathematischen Wissenschaften. Springer-Verlag, xxiv+618 pages.

[3] Giusti, Marc (1988). « Combinatorial dimension theory of algebraic varieties ». In : *Journal of Symbolic Computation*, vol. 6, n°2-3. Special issue on Computational aspects of commutative algebra, p. 249–265.

[4] Giusti, Marc, Klemens Hägele, Grégoire Lecerf, Joël Marchand et Bruno Salvy (2000). « The projective Noether Maple package : computing the dimension of a projective variety ». In : *J. Symbolic Comput.* vol. 30, n°3, p. 291–307.

[5] Giusti, Marc et Joos Heintz (1993). « La détermination des points isolés et de la dimension d'une variété algébrique peut se faire en temps polynomial ». In : *Computational algebraic geometry and commutative algebra (Cortona, 1991)*. Vol. XXXIV. Sympos. Math. Cambridge : Cambridge Univ. Press, p. 216–256.

[6] Giusti, Marc, Grégoire Lecerf et Bruno Salvy (2001). « A Gröbner free alternative for polynomial system solving ». In : *Journal of Complexity*, vol. 17, n°1, p. 154–211.

[7] Heintz, J. et C.-P. Schnorr (1982). « Testing polynomials which are easy to compute ». In : *Logic and algorithmic (Zurich, 1980)*. Vol. 30. Monograph. Enseign. Math. Geneva : Univ. Genève, p. 237–254.

[8] Risler, Jean-Jacques et Felice Ronga (1990). « Testing polynomials ». In : *Journal of Symbolic Computation*, vol. 10, n°1, p. 1–5.

CHAPITRE 26

Résolution géométrique

Résumé

Dans ce chapitre nous présentons une version simplifiée de l'algorithme de résolution géométrique pour résoudre une suite régulière réduite de polynômes à coefficients dans un corps de caractéristique zéro où suffisamment grande.

Tout au long de ce chapitre le corps k est supposé être de caractéristique zéro (ou suffisamment grande, mais nous n'entrerons pas dans ces détails), et $f_1, \ldots, f_s$ est une suite de polynômes homogènes de $k[X_0, \ldots, X_n]$ formant une *suite régulière* de degrés respectifs $d_1, \ldots, d_s$.

Pour $i \in \{0, \ldots, s\}$, on pose $I_i := (f_1, \ldots, f_i)$, le i-ème *idéal intermédiaire*. Par le Corollaire 1 du Chapitre 24, la dimension (projective) de I_i est $n - i$. Afin de simplifier les notations, la quantité r représentera systématiquement l'expression $n - i$. On pose $A_i := k[X_0, \ldots, X_r]$, $B_i := k[X_0, \ldots, X_n]/I_i$, $A'_i := k(X_0, \ldots, X_r)$ et $B'_i := A'_i[X_{r+1}, \ldots, X_n]/I_i$. On définit aussi $\delta_i := d_1 \cdots d_i$.

Exemple 1. Tout au long de ce chapitre nous utiliserons plusieurs fois l'exemple suivant, avec $n = 3$ et $s = 3$, pour illustrer les résultats :

$$\begin{aligned} f_1 &:= X_1^2 + X_2^2 + X_3^2 - 2X_0^2, \\ f_2 &:= X_1^2 + X_2^2 - X_0^2, \\ f_3 &:= X_1 - X_2 + 3X_3. \end{aligned}$$

Pour l'analyse de complexité, nous supposerons que les polynômes $f_1, \ldots, f_s$ sont représentés par un DAG tel que décrit à la fin du Chapitre 25 permettant de les évaluer en L opérations.

1. Polynômes caractéristiques et minimaux

On rappelle qu'un idéal homogène I de dimension r est dit en *position de Noether* si $B := k[X_0, \ldots, X_n]/I$ est une *extension entière* de $A := k[X_0, \ldots, X_r]$. En d'autres termes, la position de Noether de I correspond à $I \cap A = (0)$ et au fait que tous les éléments de B sont entiers sur A. L'anneau A peut alors être vu comme un sous-anneau de B, et $B' := k(X_0, \ldots, X_r)[X_{r+1}, \ldots, X_n]/I$ est un $k(X_0, \ldots, X_r)$-espace vectoriel de dimension finie, notée δ.

Definition 1. Un polynôme $q \in k[X_0, \ldots, X_r, T]$ est dit *quasi-homogène* pour le poids d pour T et 1 pour $X_0, \ldots, X_r$ si $q(X_0, \ldots, X_r, T^d)$ est homogène. Le *quasi-degré* d'un tel polynôme q pour ces poids est le degré de $q(X_0, \ldots, X_r, T^d)$.

Proposition 1. *Soit I un idéal homogène en position de Noether, et soit $f \in k[X_0, \ldots, X_n]$ homogène de degré d. Avec les précédentes notations, le polynôme minimal $\mu(T)$ et le polynôme caractéristique $\chi(T)$ de l'endomorphisme de multiplication par f dans B' appartiennent à $A[T]$, et sont quasi-homogènes de quasi-degré $d\delta$, si on associe le poids d à la variable T, et 1 aux variables $X_0, \ldots, X_r$.*

DÉMONSTRATION. Puisque f est entier sur A, il existe un polynôme unitaire $q \in A[T]$ tel que $q(f) = 0$ dans B. On peut même supposer que q est quasi-homogène de quasi-degré $d \deg q$. Par le lemme de Gauss [17, Chapter 4, Theorem 2.1] tous les facteurs irréductibles de q ont leurs coefficients dans A et sont quasi-homogènes. Comme μ divise q et est unitaire alors tous ses facteurs irréductibles sont quasi-homogènes et appartiennent à $A[T]$, il en va de même pour χ. □

DEFINITION 2. Soit A un anneau intègre, et soit B un A-module. Un élément $b \in B$ est de *torsion* s'il existe un élément $a \in A$ non nul tel que $ab = 0$. Le module B est dit *sans torsion* si zéro est son seul élément de torsion.

Le Corollaire 1 du Chapitre 25 nous donne l'existence d'une sous-variété propre W de l'espace des changements de variables linéaires telle que tout changement de variables choisi en dehors de W soit inversible et mette tous les idéaux I_i en position de Noether. À partir de maintenant nous pourrons supposer que :

(H_1) Pour tout $i \in \{1, \dots, s\}$, l'idéal I_i est en position de Noether.

LEMME 1. *Sous l'hypothèse* (H_1)*, pour tout* $i \in \{0, \dots, s-1\}$*, le polynôme* $\mu_{i+1}(0)$ *est unitaire en* X_r*.*

DÉMONSTRATION. Soit $\rho \in k[X_0, \dots, X_r]$ un polynôme homogène non nul, unitaire en X_r, appartenant à I_{i+1}. Il existe un polynôme homogène g tel que $\rho = gf_{i+1}$ dans B_i. À partir du polynôme minimal $\mu_{i+1}(T) = T^m + \mu_{i+1,m-1}T^{m-1} + \cdots + \mu_{i+1,0}$ on obtient que

$$\mu_{i+1,0}g^m + \cdots + \mu_{i+1,m-1}\rho^{m-1}g + \rho^m$$

vaut 0 dans B_i' et donc que le polynôme minimal de g est

$$T^m + \cdots + \frac{\mu_{i+1,m-1}\rho^{m-1}}{\mu_{i+1,0}}T + \frac{\rho^m}{\mu_{i+1,0}}.$$

Par la Proposition 1, $\mu_{i+1}(0) = \mu_{i+1,0}$ divise ρ^m, ce qui implique en particulier que $\mu_{i+1}(0)$ est unitaire en X_r. □

LEMME 2. *Sous l'hypothèse* (H_1)*, si* f *est un diviseur de zéro dans* B_s *alors le coefficient constant du polynôme minimal de l'endomorphisme de multiplication par* f *dans* B_s' *est égal à zéro.*

DÉMONSTRATION. Il existe un polynôme homogène g tel que $0 = gf$ dans B_s. La conclusion provient des calculs de la preuve précédente avec $r = s$ et $\rho = 0$. □

DÉFINITION 1. *Des polynômes* $a_1, \dots, a_m$ *de* R *sont dits* algébriquement indépendants *modulo un idéal* I *de* R *s'il n'existe pas de polynôme* E *non nul tel que* $E(a_1, \dots, a_m) \in I$*.*

LEMME 3. *Sous l'hypothèse* (H_1)*, pour tout* $i \in \{0, \dots, s\}$*,* $X_1, \dots, X_{r-1}, f_{i+1}$ *sont algébriquement indépendants modulo* I_i*, et* B_i *est une extension entière de* $k[X_0, \dots, X_{r-1}, f_{i+1}]$*.*

DÉMONSTRATION. Soit $E \in k[X_1, \dots, X_{r-1}, T]$ tel que $E(X_1, \dots, X_{r-1}, f) \in I_i$. Par la Proposition 1, $\mu_{i+1}(0)$ divise $E(0)$ dans A_i, ce qui implique que $E(0) = 0$ à l'aide du Lemme 1. Puisque f_{i+1} est non diviseur de zéro, on en déduit que $E = 0$. En d'autres termes, $X_1, \dots, X_{r-1}, f_{i+1}$ sont algébriquement indépendants modulo I_i. Par ailleurs, le Lemme 1 assure que μ_{i+1} définit une relation entière pour X_r sur $k[X_1, \dots, X_{r-1}, f_{i+1}]$ modulo I_i, on peut voir B_i comme une extension entière de $k[X_1, \dots, X_{r-1}, f_{i+1}]$. □

PROPOSITION 2. *Sous l'hypothèse* (H_1)*, pour tout* $i \in \{0, \dots, s\}$*, le* A_i*-module* B_i *est sans torsion.*

DÉMONSTRATION. La preuve se fait par récurrence sur i. Quand $i = 0$, $B_0 = A_0$ et donc la proposition est vérifiée. Supposons la proposition vérifiée pour un certain entier $i \geq 0$. On utilise le lemme précédent, et on note $B_{f_{i+1}}$ pour B_i vu comme une extension entière de $k[X_1, \ldots, X_{r-1}, f_{i+1}]$. Le $k(X_1, \ldots, X_{r-1}, f_{i+1})$-espace vectoriel correspondant est noté $B'_{f_{i+1}}$.

Soient $m \in \mathbb{N}$, $b \in k[X_0, \ldots, X_n]$ et $a \in k[X_0, \ldots, X_{r-1}] \setminus \{0\}$ tels que $a^m b^m - gf_{i+1} \in I_i$. Par la Proposition 1, les polynômes minimaux des endomorphismes de multiplication par g et par b^m dans $B'_{f_{i+1}}$ appartiennent à $k[X_0, \ldots, X_{r-1}, f_{i+1}][T]$. Soit $\rho(T) = T^\alpha + \rho_{\alpha-1}T^{\alpha-1} + \cdots + \rho_0$ le polynôme minimal de g dans $B'_{f_{i+1}}$. Par l'hypothèse de récurrence a est non diviseur de zéro dans $B'_{f_{i+1}}$, on peut donc écrire le polynôme minimal de b^m dans $B'_{f_{i+1}}$ comme

$$\left(\frac{f_{i+1}}{a^m}\right)^\alpha \rho\left(\frac{a^m T}{f_{i+1}}\right) = T^\alpha + \rho_{\alpha-1}\left(\frac{f_{i+1}}{a^m}\right)T^{\alpha-1} + \cdots + \left(\frac{f_{i+1}}{a^m}\right)^\alpha \rho_0.$$

Par conséquent $(a^m)^j$ divise $f_{i+1}^j \rho_{\alpha-j}$ dans $k[X_0, \ldots, X_{r-1}, f_{i+1}]$, pour tout $j \in \{0, \ldots, \alpha - 1\}$. Comme $X_1, \ldots, X_{r-1}, f_{i+1}$ sont algébriquement indépendants modulo I_i, et que $a \in k[X_0, \ldots, X_{r-1}]$, on déduit que $(a^m)^j$ divise $\rho_{\alpha-j}$, et donc que $(b^m)^\alpha \in I_i + (f_{i+1})$. Par conséquent B_{i+1} est bien un A_{i+1}-module sans torsion. □

COROLLAIRE 1. *Sous l'hypothèse (H_1), pour tout $i \in \{1, \ldots, s\}$, pour tout polynôme f homogène de polynôme caractéristique χ et minimal μ dans B'_i, on a $\mu(f) = \chi(f) = 0$ dans B_i.*

DÉMONSTRATION. Comme il existe un polynôme non nul $c \in A_i$ tel que $c\mu(f) = 0$ dans B_i et comme B_i est sans torsion par la proposition précédente, on en déduit $\mu(f) = 0$ dans B_i. Il en va de même pour χ. □

EXEMPLE 2. Revenons à l'Exemple 1. Comme f_1 est non nul, il n'est pas diviseur de zéro dans $B_0 = k[X_0, \ldots, X_n]$. De plus, on observe que I_1 est bien en position de Noether. Une base de B'_1 est $1, X_3$. Dans cette base, on peut calculer la matrice de multiplication par f_2 et obtenir ainsi le coefficient constant du polynôme caractéristique de f_2 dans B'_1 :

$$\chi_2(0) = X_2^4 + 2X_1^2X_2^2 - 2X_0^2X_2^2 + X_1^4 - 2X_0^2X_1^2 + X_0^4.$$

Par le Lemme 2, on en déduit que f_2 n'est pas diviseur de zéro dans B_1 et que I_2 est en position de Noether.

2. Élément primitif

Dans cette section nous étudions des bases particulières pour les B'_i, dans lesquelles les calculs sont plus faciles que dans d'autres bases en général.

DEFINITION 3. Une forme linéaire $u = \lambda_0 X_0 + \cdots + \lambda_n X_n$ est dite un *élément primitif* pour B'_i si $1, u, \ldots, u^{\delta_i - 1}$ forment une base de B'_i.

PROPOSITION 3. *Si u est un élément primitif et q le polynôme caractéristique de l'endomorphisme de multiplication par u dans B'_i, alors B'_i est isomorphe à $A'_i[T]/(q(T))$:*

$$\begin{array}{rcl} B'_i & \to & A'_i[T]/(q(T)) \\ X_j & \mapsto & v_j(T), \end{array}$$

où les $v_j \in A'_i[T]$ sont de degré au plus $\delta_i - 1$. On dit que les polynômes $q, v_{r+1}, \ldots, v_n$ constituent une résolution géométrique *de I_i.*

DÉMONSTRATION. L'image de chaque X_j dans B'_i s'exprime comme une combinaison linéaire de $1, u, \ldots, u^{\delta_i - 1}$. □

EXEMPLE 3. Si f_1 est unitaire en X_n et sans carré, alors la forme linéaire $u = X_n$ est un élément primitif pour B'_1.

LEMME 4. *Si I_i est radical, alors I_i vu comme l'idéal I'_i de $A'_i[X_{r+1}, \dots, X_n]$ est aussi radical. En particulier B'_i est une A'_i-algèbre réduite.*

DÉMONSTRATION. Soit $f \in \sqrt{I'_i}$, il existe un entier n et un polynôme $c \in A_i$ tels que $(cf)^n \in I'_i$ et $cf \in k[X_0, \dots, X_n]$. Par conséquent $cf \in I_i$, et donc $f \in I'_i$. □

Soient $\Lambda_0, \dots, \Lambda_n$ des nouvelles variables. On note $k_\Lambda := k(\Lambda_0, \dots, \Lambda_n)$, $A_{\Lambda,i} := k[\Lambda_0, \dots, \Lambda_n, X_0, \dots, X_r]$, et $A'_{\Lambda,i} := k(\Lambda_0, \dots, \Lambda_n, X_0, \dots, X_r)$. On introduit aussi l'algèbre $B'_{\Lambda,i} := A'_{\Lambda,i}[X_{r+1}, \dots, X_n]/I'_{\Lambda,i}$, où $I'_{\Lambda,i}$ représente l'extension de I_i dans $A'_{\Lambda,i}[X_{r+1}, \dots, X_n]$. L'extension de I_i dans $k[\Lambda_0, \dots, \Lambda_n, X_0, \dots, X_n]$ est notée $I_{\Lambda,i}$. Soit $u_\Lambda := \Lambda_0 X_0 + \cdots + \Lambda_n X_n$, et soit χ_{u_Λ} le polynôme caractéristique de l'endomorphisme de multiplication par u_Λ dans $B'_{\Lambda,i}$.

PROPOSITION 4. *Sous l'hypothèse (H_1), si I_i est radical, alors le polynôme χ_{u_Λ} est sans carré et $\chi_{u_\Lambda}(u_\Lambda)$ appartient à $I_{\Lambda,i}$. De plus on a $\frac{\partial \chi_{u_\Lambda}}{\partial T}(u_\Lambda)X_j + \frac{\partial \chi_{u_\Lambda}}{\partial \Lambda_j}(u_\Lambda) \in I_{\Lambda,i}$ pout tout $j \in \{0, \dots, n\}$.*

DÉMONSTRATION. $\chi_{u_\Lambda} \in A_{\Lambda,i}[T]$ et $\chi_{u_\Lambda}(u_\Lambda) \in I_{\Lambda,i}$ par le Corollaire 1. En dérivant cette dernière expression par rapport à Λ_j, on obtient $\chi'_{u_\Lambda}(u_\Lambda)X_j + \frac{\partial \chi_{u_\Lambda}}{\partial \Lambda_j}(u_\Lambda) \in I_{\Lambda,i}$. Comme $I'_{\Lambda,i}$ est radical par le lemme précédent, la partie sans carré μ_{u_Λ} de χ_{u_Λ} satisfait $\mu_{u_\Lambda}(u_\Lambda) \in I_{\Lambda,i}$. Comme on a aussi $\mu'_{u_\Lambda}(u_\Lambda)X_j + \frac{\partial \mu_{u_\Lambda}}{\partial \Lambda_j}(u_\Lambda) \in I_{\Lambda,i}$, on obtient que B'_i est isomorphe à $A'_{\Lambda,i}[T]/(\mu_{u_\Lambda})$. Par conséquent on a $\mu_{u_\Lambda} = \chi_{u_\Lambda}$. □

COROLLAIRE 2. *Si I_i est radical, alors u est un élément primitif de B'_i si, et seulement si, le polynôme caractéristique χ_u de l'endomorphisme de multiplication par u dans B'_i est sans carré.*

DÉMONSTRATION. Si u est primitif alors χ_u est sans carré car I_i est radical. Inversement supposons χ_u sans carré. La proposition précédente donne l'existence de polynômes $w_j \in A'_i[T]$ tels que $\chi'_u(u)X_j - w_j(u) \in I_i$ pour tout $j \in \{0, \dots, n\}$. On en déduit que les puissances de u engendrent bien B'_i. □

PROPOSITION 5. *Sous l'hypothèse (H_1), si le polynôme χ_{u_Λ} est sans carré alors I_i est radical.*

DÉMONSTRATION. $\chi_{u_\Lambda} \in A_{\Lambda,i}[T]$ et $\chi_{u_\Lambda}(u_\Lambda) \in I_{\Lambda,i}$ par le Corollaire 1. En dérivant cette dernière expression par rapport à Λ_j, on obtient $\chi'_{u_\Lambda}(u_\Lambda)X_j + \frac{\partial \chi_{u_\Lambda}}{\partial \Lambda_j}(u_\Lambda) \in I_{\Lambda,i}$. Si χ_{u_Λ} est sans carré, alors u_Λ est bien un élément primitif pour $B'_{\Lambda,i}$ ce qui implique de I'_i est radical. Si f est un polynôme homogène de $k[X_0, \dots, X_n]$ tel que $f^m \in I_i$, alors $f \in I'_i$ et il existe un polynôme $c \in A_i$ tel que $cf \in I_i$. Par la Proposition 2 on en déduit que $f \in I_i$ et donc que I_i est radical. □

PROPOSITION 6. *Sous l'hypothèse (H_1), si I_i est radical, si u est un élément primitif pour B'_i, et si $q \in A_i[T]$ en est le polynôme caractéristique, alors, pour tout α dans la clôture algébrique $\overline{A'_{i+1}}$ de A'_{i+1}, le spécialisé $q(X_1, \dots, X_{r-1}, \alpha, T)$ est le polynôme caractéristique de u dans $\overline{A'_{i+1}}[X_r, \dots, X_n]/(I_i + (X_r - \alpha))$.*

DÉMONSTRATION. Soit $\tilde{B}_i := \overline{A'_{i+1}}[X_r, \dots, X_n]/\tilde{I}_i$, où $\tilde{I}_i$ représente l'idéal I_i vu dans $\overline{A'_{i+1}}[X_r, \dots, X_n]$. Comme B_i est sans torsion il en va de même pour $\tilde{B}_i$, qui est par conséquent un $\overline{A'_{i+1}}[X_r]$-module libre de rang fini [17, Chapter III,

Theorem 7.3]. Comme une base de $\tilde{B}_i$ induit une base de B'_i, le polynôme caractéristique de l'endomorphisme de multiplication par u dans $\tilde{B}_i$ est q. Par ailleurs, une base de $\tilde{B}_i$ induit aussi une base de $\overline{A'_{i+1}}[X_r,\dots,X_n]/(\tilde{I}_i+(X_r-\alpha))$, ce qui implique que $q(X_1,\dots,X_{r-1},\alpha,T)$ est bien le polynôme caractéristique de u dans $\overline{A'_{i+1}}[X_r,\dots,X_n]/(I_i+(X_r-\alpha))$ □

À partir de maintenant nous supposons que les idéaux intermédiaires I_i sont tous *radicaux*. On dit alors que $f_1,\dots,f_s$ forment une suite *régulière réduite*.

PROPOSITION 7. *Il existe une variété algébrique W_i propre de $\mathbb{P}^n$ telle que pour tout $(\lambda_0:\cdots:\lambda_n)\notin W_i$ la forme linéaire $u=\lambda_0X_0+\cdots+\lambda_nX_n$ est un élément primitif pour B'_i.*

DÉMONSTRATION. W_i est donnée comme le lieu d'annulation du discriminant de χ_{u_Λ}. □

Tout comme nous avons changé les variables pour obtenir des positions de Noether, nous allons pouvoir prendre des nouvelles variables qui soient des éléments primitifs. L'ensemble des matrices carrées de taille $n+1$ sur k qui sont triangulaires supérieures avec des 1 sur la diagonale est noté $\mathcal{T}_{n+1}(k)$.

PROPOSITION 8. *Pour tout $i\in\{1,\dots,s\}$, il existe une variété algébrique affine propre W_i de $\mathcal{T}_{n+1}(k)$ telle que pour toute matrice $M\notin W_i$, une fois appliquée à $f_1,\dots,f_s$, l'hypothèse (H_1) est satisfaite et X_{r+1} est un élément primitif pour B'_i.*

DÉMONSTRATION. Soient $l_0,\dots,l_n$ les formes linéaires définies par

$$(l_0,\dots,l_n)^t:=M^{-1}(X_0,\dots,X_n)^t.$$

Fixons un entier $i\in\{1,\dots,s\}$. Par la Proposition 6 du Chapitre 25, on peut supposer que $l_0,\dots,l_r$ sont algébriquement indépendantes modulo I_i et $l_{r+1},\dots,l_n$ sont entières sur $k[l_0,\dots,l_r]$ modulo I_i. Comme $M\in\mathcal{T}_{n+1}(k)$, les variables $X_{r+1},\dots,X_n$ sont aussi entières sur $k[l_0,\dots,l_r]$ modulo I_i. La précédente proposition assure que si les coefficients de l_{r+1} sont en dehors d'une certaine variété algébrique alors l_{r+1} est bien un élément primitif pour B'_i. □

COROLLAIRE 3. *Pour tout $i\in\{1,\dots,s\}$, il existe une variété algébrique affine propre W_i de $\mathcal{M}_{n+1}(k)$ telle que pour toute matrice $M\notin W_i$, M soit inversible et, une fois appliquée à $f_1,\dots,f_s$, l'hypothèse (H_1) est satisfaite et X_{r+1} est un élément primitif pour B'_i.*

DÉMONSTRATION. L'ensemble des matrices carrées de taille $n+1$ ayant un mineur principal nul est contenu dans une variété algébrique propre de $\mathcal{M}_{n+1}(k)$. Par ailleurs, il est classique que toute matrice M en dehors de cette dernière variété peut se décomposer en $M=LU$, où L est une matrice triangulaire inférieure et U est triangulaire supérieure avec des 1 sur la diagonale. Comme $A'_i[X_{r+1},\dots,X_n]/(I_i\circ L)$ admet X_{r+1} comme élément primitif si, et seulement si, X_{r+1} est primitif pour B'_i, alors la conclusion provient de la proposition précédente et du Corollaire 1 du Chapitre 25. □

Nous utiliserons souvent l'hypothèse suivante concernant le choix des coordonnées :

(H_2) Pour tout $i\in\{1,\dots,s\}$, X_{r+1} est un élément primitif pour B'_i.

3. Intersection

Dans cette section on s'intéresse à calculer une résolution géométrique de I_{i+1} à partir de celle de I_i. Rappelons que $f_1,\dots,f_s$ est supposé être une suite régulière réduite.

PROPOSITION 9. *Sous l'hypothèse* (H_1)*, le polynôme* $\chi_{i+1}(0)$ *est proportionnel sur* k *au polynôme caractéristique de la multiplication par* X_r *dans* B'_{i+1}.

DÉMONSTRATION. Soit $\tilde{B}_i := k(X_0,\dots,X_{r-1})[X_r,\dots,X_n]/\tilde{I}_i$ où $\tilde{I}_i$ représente l'idéal I_i vu dans $A'_{i+1}[X_r,\dots,X_n]$. Comme B_i est sans torsion il en va de même pour $\tilde{B}_i$ qui est par conséquent un $k(X_0,\dots,X_{r-1})[X_r]$-module libre de rang fini noté δ [17, Chapter III, Theorem 7.3]. Par le théorème classique de réduction des matrices dont les entrées sont dans un anneau principal [17, Chapter III, Theorem 7.9], il existe deux bases $e_1,\dots,e_\delta$ et $e'_1,\dots,e'_\delta$ de $\tilde{B}_i$ et des polynômes $h_1,\dots,h_\delta$ de $k(X_0,\dots,X_{r-1})[X_r]$, tels que h_l divise h_{l+1} pour tout $l\in\{1,\dots,\delta-1\}$ et $f_{i+1}e_l = h_l e'_l$ dans $\tilde{B}_i$.

Comme une base de $\tilde{B}_i$ induit une base de B', on obtient que $\chi_{i+1}(0) = ah_1\cdots h_\delta$, pour un certain $a\in k(X_0,\dots,X_{r-1})$. D'autre part, on prétend que

$$\mathcal{B} := \{X_r^{\alpha_l} e'_l \mid 1\leqslant l\leqslant\delta, 0\leqslant\alpha_l\leqslant \deg(h_l)-1\}$$

est une base de $\tilde{B}_i/(f)$. Vérifions tout d'abord que $\mathcal{B}$ engendre bien $\tilde{B}_i/(f)$. Une préimage $\tilde{g}$ d'un élément $g\in\tilde{B}_i/(f)$ peut se réécrire en $g=\sum_{l=1}^{\delta} g_l e'_l$, où les g_l appartiennent à $k(X_0,\dots,X_{r-1})[X_r]$ et sont de degré strictement inférieur à $\deg(h_l)$. Si $\sum_{l=1}^{\delta} g_l e'_l = 0$ dans $\tilde{B}_i/(f)$, avec $\deg(g_l)\leqslant\deg(h_l)-1$, alors il existe des polynômes $q_1,\dots,q_\delta$ dans $k(X_0,\dots,X_{r-1})[X_r]$ tels que $\sum_{l=1}^{\delta} g_l e'_l + \sum_{l=1}^{\delta} q_l h_l e'_l = 0$ dans $\tilde{B}_i$. Ce qui implique $g_l + q_l h_l = 0$ et donc $q_l = g_l = 0$.

Dans la base $\mathcal{B}$, la matrice M de la multiplication par X_r dans $\tilde{B}_i/(f)$ est diagonale par bloc. Chacun de ces blocs est la matrice compagnon de h_l. Par conséquent $\chi_{i+1}(0)$ est bien proportionnel à $\det(M)$ sur $k(X_0,\dots,X_{r-1})$. Finalement, puisque I_{i+1} est en position de Noether, $\chi_{i+1}(0)$ est unitaire en X_r, par le Lemme 1, ce qui achève la preuve. □

EXEMPLE 4. Dans l'Exemple 2 nous avions obtenu $\chi_2(0) = (X_2^2+(X_1^2-X_0^2))^2$. Par le Corollaire 2 et la Proposition 9 on en déduit que ou bien X_2 n'est pas un élément primitif pour B'_2, ou bien I_2 n'est peut-être pas radical.

Remplaçons maintenant la variable X_2 par $X_2 - 2X_3$ dans f_1, f_2, f_3. Le coefficient constant du polynôme caractéristique de multiplication par f_2 dans B'_1 devient alors

$$\chi_2(0) = X_2^4 + 2X_1^2X_2^2 - 10X_0^2X_2^2 + X_1^4 + 6X_0^2X_1^2 + 9X_0^4,$$

qui est sans carré. Par la Proposition 5, on en déduit que I_2 est bien radical et que X_2 est devenu primitif pour B'_2. Une résolution géométrique de B'_2 est donnée par $u := X_2$, $q(T) := T^4 + 2X_1^2T^2 - 10X_0^2T^2 + X_1^4 + 6X_0^2X_1^2 + 9X_0^4$, $v_2 := T$, et $v_3 := (T^2 + X_1^2 + 3X_0^2)/(4T) \mod q(T) =$

$$-\frac{X_1^2T^3 + 3X_0^2T^3 + X_1^4T - 10X_0^2X_1^2T - 39X_0^4T}{4X_1^4 + 24X_0^2X_1^2 + 36X_0^4}.$$

On peut ainsi calculer la matrice de multiplication par f_3 et obtenir

$$\chi_3(0) = 25X_1^4 + 14X_0^2X_1^2 + 25X_0^4,$$

qui est sans carré, ce qui prouve que I_3 est aussi en position de Noether et est radical.

COROLLAIRE 4. *Sous les hypothèses* (H_1) *et* (H_2)*, la dimension de* B'_i *est* δ_i.

DÉMONSTRATION. La preuve se fait par récurrence sur i. Le résultat est vrai lorsque $i=0$. Par ailleurs, en combinant la Proposition 1 appliquée à f_{i+1} et la Proposition 9, on obtient que le degré de $\chi_{i+1}(0)$ est $d_{i+1}\delta_i = \delta_{i+1}$ et que ce degré coïncide avec la dimension de B'_{i+1}. □

La Proposition 9 donne un moyen d'obtenir le polynôme minimal de X_r. Pour calculer les expressions des autres variables X_j en fonction de X_r, nous allons utiliser une idée dérivée de la Proposition 4. À partir de maintenant, $q, v_{r+1}, \ldots, v_n$ représente une résolution géométrique de I_i pour l'élément primitif u. Soit t un élément de k, on note $U := X_r + tu$.

LEMME 5. *Sous les hypothèses (H_1) et (H_2), sauf pour un nombre fini de valeurs de t, la forme linéaire U est un élément primitif pour B'_i et B'_{i+1}, qui n'est pas diviseur de zéro dans B_i.*

DÉMONSTRATION. Notons μ_U le polynôme minimal de l'endomorphisme de multiplication de U dans B'_i. Comme u est primitif pour B'_i, le discriminant de μ_U est une expression polynomiale en t qui est non nulle par la Proposition 2 et le Corollaire 2. Le même argument donne que U est aussi primitif pour B'_{i+1} sauf pour un nombre fini de valeurs de t. Si $\mu_U(0)$ était nul pour toute valeur de t, alors X_r serait diviseur de 0 dans B_i, ce qui n'est pas possible par la Proposition 2. □

Nous utiliserons les polynômes w_j définis par $w_j := q'v_j \mod q$ dans $A'_i[T]$ pour chaque $j \in \{r+1, \ldots, n\}$. Par les Propositions 1 et 4, les w_j sont polynomiaux en $X_0, \ldots, X_r, T$ de degré totaux δ_i, et $q'(u)X_j - w_j(u) \in I_i$. Nous dirons que $q, w_{r+1}, \ldots, w_n$ représente une résolution géométrique de B'_i sous *forme de Kronecker*. On introduit une nouvelle variable Y et on définit les polynômes suivants :

$$\tilde{q}(X_0, \ldots, X_{r-1}, Y, T) := q(X_0, \ldots, X_{r-1}, Y - tT, T),$$
$$\tilde{p}(X_0, \ldots, X_{r-1}, Y, T) := q'(X_0, \ldots, X_{r-1}, Y - tT, T),$$
$$\tilde{w}_j(X_0, \ldots, X_{r-1}, Y, T) := w_j(X_0, \ldots, X_{r-1}, Y - tT, T), \text{ pour } j \geq r+1.$$

PROPOSITION 10. *Sous les hypothèses (H_1) et (H_2). Sauf pour un nombre fini de valeurs de t, $\tilde{p}$ est inversible modulo $\tilde{q}$ dans $k(X_0, \ldots, X_{r-1}, Y)[T]$. De plus, si on pose*

$$h := f_{i+1}\left(X_0, \ldots, X_{r-1}, Y - tT, \frac{\tilde{w}_j}{\tilde{p}}, \ldots, \frac{\tilde{w}_n}{\tilde{p}}\right),$$

alors le polynôme minimal de multiplication par U dans B'_{i+1} est égal à $Q_U := \mathrm{Res}_T(\tilde{q}, h)$.

DÉMONSTRATION. Le résultant $\mathrm{Res}_T(\tilde{q}, \tilde{p})$ dépend polynomialement de t et est non nul lorsque $t = 0$. Par la Proposition 2 du Chapitre 23, on en déduit que $\tilde{p}$ est bien inversible modulo $\tilde{q}$ dans $A'_i[T]$ sauf pour un nombre fini de valeurs de t.

Sauf pour un nombre fini de valeurs de t, par le lemme précédent, U est non-diviseur de zéro dans B_i. Comme $B_i/(X_r)$ est en position de Noether, $B_i/(U)$ est aussi en position de Noether sauf pour un nombre fini de valeurs de t. On peut alors appliquer la Proposition 9 à la suite régulière $f_1, \ldots, f_{i+1}$, par rapport aux variables $X_0, \ldots, X_{r-1}, U, X_{r+1}, \ldots, X_n$, et obtenir que $\mathrm{Res}_T(\tilde{q}, h)$ est bien le polynôme minimal de U dans B'_{i+1}. □

Nous dirons qu'un algorithme commute avec la spécialisation d'une variable X en la valeur a si la spécialisation de la sortie de l'algorithme coïncide avec la sortie de l'algorithme appliqué aux entrées spécialisées.

PROPOSITION 11. *Sous les hypothèses (H_1) et (H_2), sauf pour un nombre fini de valeurs de t, l'algorithme de la Figure 1 est correct. De plus, pour tout point $(a_0, \ldots, a_{r-1})$ en dehors d'une variété propre de $\mathbb{A}^r$, l'algorithme commute avec la spécialisation des variables $X_0, \ldots, X_{r-1}$ en $a_0, \ldots, a_{r-1}$.*

DÉMONSTRATION. La correction de l'algorithme vient de la proposition précédente. Les valeurs $(a_0, \ldots, a_{r-1})$ peuvent être prises de sorte à commuter avec tous les tests à zéro et à n'annuler aucun des dénominateurs intervenant dans les calculs. Ces conditions s'expriment de façon polynomiale. □

Polynôme minimal de $X_r + tu$ dans B'_{i+1}

Entrée : une résolution géométrique $u, q, w_{r+1}, \ldots, w_n$ de B'_i sous forme de Kronecker, et $t \in k$.

Sortie : le polynôme minimal Q_U de $U := X_r + tu$ dans B'_{i+1}.

1. Calculer $\tilde{q} := q(X_0, \ldots, X_{r-1}, Y - tT, T)$.
2. Calculer $\tilde{p} := q'(X_0, \ldots, X_{r-1}, Y - tT, T)$.
3. Calculer $\tilde{w}_j := w_j(X_0, \ldots, X_{r-1}, Y - tT, T)$, pour tout $j \geq r+1$.
4. Évaluer $h(T) := f_{i+1}(X_0, \ldots, X_{r-1}, Y - tT, \tilde{w}_{r+1}/\tilde{p}, \ldots, \tilde{w}_n/\tilde{p}) \mod \tilde{q}$ dans $A'_i[T]$.
5. Calculer $Q_U := \mathrm{Res}_T(\tilde{q}, h)$ et retourner Q_U.

FIGURE 1. Polynôme minimal après ajout d'une équation

Résolution géométrique de B'_{i+1}

Entrée : une résolution géométrique $u, q, w_{r+1}, \ldots, w_n$ de B'_i sous forme de Kronecker, et $t \in k$.

Sortie : une résolution géométrique $Q, V_r, \ldots, V_n$ de B'_{i+1} pour $U := X_r + tu$.

1. Appeler l'algorithme de la Figure 1 en remplaçant k par $k[[Z - t]]$, en prenant Z comme valeur pour t, et en calculant à la précision $O((Z-t)^2)$. On note $Q_Z = Q - W(Z - t) + O((Z - t)^2)$ le polynôme minimal de $U_Z = X_r + Zu$ ainsi obtenu.
2. Calculer $V = W/Q' \mod Q$, et $V_r := T - tV$.
3. Avec les notations de la Figure 1, calculer :
   $\rho := \tilde{p}(X_0, \ldots, X_{r-1}, T, V(T)) \mod Q(T)$, et
   $\omega_j := \tilde{w_j}(X_0, \ldots, X_{r-1}, T, V(T)) \mod Q(T)$, pour $j \geq r+1$.
4. Pour chaque $j \in \{r+1, \ldots, n\}$, calculer $V_j := \omega_j/\rho \mod Q$.
5. Retourner $Q, V_r, \ldots, V_n$.

FIGURE 2. Résolution géométrique après ajout d'une équation

PROPOSITION 12. *Sous les hypothèses (H_1) et (H_2), il existe une variété algébrique W_i propre de $\mathbb{P}^n$ telle que, pour tout $(\lambda_0 : \cdots : \lambda_n) \notin W_i$ et sauf pour un nombre fini de valeurs de t, l'algorithme de la Figure 2 est correct avec $u = \lambda_0 X_0 + \cdots + \lambda_n X_n$.*

De plus, pour tout point $(a_0, \ldots, a_{r-1})$ en dehors d'une variété propre de $\mathbb{A}^r$, l'algorithme commute avec la spécialisation des variables $X_0, \ldots, X_{r-1}$ en les valeurs $a_0, \ldots, a_{r-1}$. Pour une telle spécialisation, le calcul de

$$Q(a_0, \ldots, a_{r-1}, T), V_r(a_0, \ldots, a_{r-1}, T), \ldots, V_n(a_0, \ldots, a_{r-1}, T)$$

à partir de

$$q(a_0, \ldots, a_{r-1}, X_r, T), w_{r+1}(a_0, \ldots, a_{r-1}, X_r, T), \ldots, w_n(a_0, \ldots, a_{r-1}, X_r, T)$$

peut se faire avec $\tilde{O}((n + L)d_{i+1}\delta_i^2)$ opérations dans k.

DÉMONSTRATION. Si l'algorithme de la Figure 2 est exécuté dans $k(Z-t)$, alors on obtient le polynôme caractéristique de U_Z dans B'_{i+1} étendu avec Z. Lorsque Z vaut t on obtient bien le polynôme minimal de $U = X_r + tu$, sauf pour un nombre fini de valeurs de t par la proposition précédente.

Par ailleurs, la Proposition 4, nous donne que $Q'(U)u - W(U) \in I_{i+1}$. Sauf pour un nombre fini de valeurs de t, le Lemme 5 nous assure que U est primitif pour B'_{i+1} et donc que Q' est inversible modulo Q et $u = W(U)/Q'(U)$ dans B'_{i+1}.

Soit α une racine de Q dans $\overline{A'_{i+1}}$. Comme U est primitif, on note $(\alpha_r, \ldots, \alpha_n)$ l'unique solution de $I_{i+1} + (U - \alpha)$. Si $u = \lambda_0 X_0 + \cdots + \lambda_n X_n$, on note u_α la valeur $\lambda_0 X_0 + \cdots + \lambda_{r-1} X_{r-1} + \lambda_r \alpha_r + \cdots + \lambda_n \alpha_n$. Par la Proposition 6, le polynôme $q(X_0, \ldots, X_{r-1}, \alpha_r, T)$ est le polynôme caractéristique de la multiplication par u dans $\overline{A'_{i+1}}[X_r, \ldots, X_n]/(I_i + (X_r - \alpha_r))$. Comme X_r est supposé primitif pour B_{i+1}, l'idéal $I_{i+1} + (X_r - \alpha) = I_i + (X_r - \alpha) + f_{i+1}$ est radical. Si u est pris primitif pour $A'_{i+1}[X_r, \ldots, X_n]/\sqrt{I_{i+1}}$, alors on en déduit que u_α est une racine simple de $q(X_0, \ldots, X_{r-1}, \alpha_r, T)$. Par conséquent ρ est inversible modulo Q.

Le calcul de $\tilde{q}$ et des $\tilde{w}_j$ de la Figure 1 peut se faire avec $\tilde{O}(n\delta_i^2)$ opérations dans k en utilisant le Théorème 1 du Chapitre 5. Ensuite Q_U est interpolé à partir de $\delta_{i+1} + 1$ valeurs. Pour chacune de ces valeurs, h peut être obtenu avec $\tilde{O}(L\delta_i)$ opérations et Q_U avec $\tilde{O}(\delta_i)$ opérations en utilisant une variante de l'algorithme d'Euclide rapide du Chapitre 6. Le calcul de Q, V, ρ, et des ω_j de la Figure 1 utilise $\tilde{O}(nd_{i+1}\delta_i^2)$ opérations. □

4. Remontée de Hensel

Soit $q, v_{r+1}, \ldots, v_n$ une résolution géométrique de B'_i pour l'élément primitif u.

DEFINITION 4. Un point $(a_0 : \cdots : a_r)$ est un *point de remontée* s'il n'annule pas le discriminant de q.

PROPOSITION 13. *Il existe une variété algébrique propre W de $\mathbb{P}^r$ telle que tout $(a_0 : \cdots : a_r) \notin W$ est un point de remontée.*

DÉMONSTRATION. On prend pour W la variété définie par le discriminant de q. □

PROPOSITION 14. *La matrice jacobienne J de $(f_1, \ldots, f_i)$ par rapport aux variables $X_{r+1}, \ldots, X_n$ est inversible dans B'_i. La spécialisation de J en un point de remontée reste inversible.*

DÉMONSTRATION. En dérivant $\chi_{u_\Lambda}(u_\Lambda) = g_1 f_1 + \cdots + g_i f_i$ par rapport à X_j on obtient l'égalité suivante dans $B_{\Lambda,i}$:

$$\Lambda_j \chi'_{u_\Lambda}(u_\Lambda) = g_1 \frac{\partial f_1}{\partial X_j} + \cdots + g_i \frac{\partial f_i}{\partial X_j}.$$

Comme $\chi'_{u_\Lambda}(u_\Lambda)$ est inversible dans $B'_{\Lambda,i}$ alors la matrice jacobienne de $(f_1, \ldots, f_i)$ l'est.

Soit $a_0, \ldots, a_r$ un point de remontée. Par la Proposition 6, le polynôme caractéristique de u_Λ dans $k[X_0, \ldots, X_n]/(I_i + (X_0 - a_0, \ldots, X_r - a_r))$ est $\chi_{u_\Lambda}(a_0, \ldots, a_r, T)$. Comme ce dernier est sans carré, les arguments du paragraphe précédent s'adaptent pour montrer que J est inversible dans $k[X_0, \ldots, X_n]/(I_i + (X_0 - a_0, \ldots, X_r - a_r))$. □

PROPOSITION 15. *Sous l'hypothèse (H_1), si $(a_0, \ldots, a_r)$ est un point de remontée pour B'_i, alors l'algorithme de la Figure 3 est correct et commute avec la spécialisation des variables $X_1, \ldots, X_{r-1}$ en $a_1, \ldots, a_{r-1}$. Le calcul de*

$$q(a_0, \ldots, a_{r-1}, X_r, T), w_{r+1}(a_0, \ldots, a_{r-1}, X_r, T), \ldots, w_n(a_0, \ldots, a_{r-1}, X_r, T)$$

à partir de

$$q(a_0, \ldots, a_r, T), w_{r+1}(a_0, \ldots, a_r, T), \ldots, w_n(a_0, \ldots, a_r, T)$$

peut se faire avec $\tilde{O}(n^4\delta_i + (n^3 + nL)\delta_i^2)$ opérations dans k.

Remontée de Hensel

Entrée : $q(a_0,\dots,a_r,T), v_{r+1}(a_0,\dots,a_r,T),\dots,v_n(a_0,\dots,a_r,T)$.

Sortie : $q(a_0,X_1,\dots,X_r,T), w_{r+1}(a_0,X_1\dots,X_r,T),\dots,$
$w_n(a_0,X_0,\dots,X_r,T)$ tels que $w_j = \frac{\partial q}{\partial T} v_j \mod q$.

1. Initialiser $\hat{Q} := q(a_0,\dots,a_r,T)$, $\hat{V}_{r+1} := v_{r+1}(a_0,\dots,a_r,T)$, …, $\hat{V}_n := v_n(a_0,\dots,a_r,T)$ vus dans $k[[X_1-a_1,\dots,X_r-a_r]][T]$. Poser $l := 0$.
2. Tant que $2^l < \delta_i$ faire :
   (a) Calculer $\hat{v}_{r+1},\dots,\hat{v}_n$ à la précision $(X_1-a_1,\dots,X_r-a_r)^{2^{l+1}}$ modulo $\hat{Q}$ comme suit :

$$\begin{pmatrix} \hat{v}_{r+1} \\ \vdots \\ \hat{v}_n \end{pmatrix} = \begin{pmatrix} \hat{V}_{r+1} \\ \vdots \\ \hat{V}_n \end{pmatrix} - \left(J^{-1} \begin{pmatrix} f_1 \\ \vdots \\ f_i \end{pmatrix} \right) (a_0, X_1, \dots, X_r, \hat{V}_{r+1}, \dots, \hat{V}_n),$$

   où J^{-1} représente l'inverse de la matrice jacobienne J de $f_1,\dots,f_i$ par rapport aux variables $X_{r+1},\dots,X_n$.
   (b) Calculer $\Delta := u(a_0,X_1,\dots,X_r,\hat{v}_{r+1},\dots,\hat{v}_n) - T$.
   (c) Pour tout $j \in \{r+1,\dots,n\}$ faire :
      (i) Calculer $\Delta_j := \Delta \frac{\partial \hat{V}_j}{\partial T} \mod \hat{Q}$ à la précision $(X_1-a_1,\dots,X_r-a_r)^{2^{l+1}}$.
      (ii) Remplacer $\hat{V}_j$ par $\hat{v}_j - \Delta_j$.
   (d) Calculer $\Delta_Q := \Delta \frac{\partial \hat{Q}}{\partial T} \mod \hat{Q}$ à la précision $(X_1-a_1,\dots,X_r-a_r)^{2^{l+1}}$.
   (e) Remplacer $\hat{Q}$ par $\hat{Q} - \Delta_Q$.
   (f) Remplacer l par $l+1$.
3. Pour tout $j \in \{r+1,\dots,n\}$ calculer $\hat{W}_j := \hat{V}_j \frac{\partial \hat{Q}}{\partial T} \mod \hat{Q}$ à la précision $(X_1-a_1,\dots,X_r-a_r)^{2^{l+1}}$.
4. Retourner $\hat{Q}, \hat{W}_{r+1},\dots,\hat{W}_n$ vus comme des polynômes de $A_i[T]$.

FIGURE 3. Remontée d'une résolution géométrique

DÉMONSTRATION. Au début de l'étape l de la boucle principale de l'algorithme, on va prouver par récurrence que l'on a $f_j(a_0,X_1,\dots,X_r,\hat{V}_{r+1},\dots,\hat{V}_n) \mod \hat{Q} = 0$ et $u(a_0,X_1,\dots,X_r,\hat{V}_{r+1},\dots,\hat{V}_n) = T$ à la précision $(X_1-a_1,\dots,X_r-a_r)^{2^l}$ pour chaque $j \in 1,\dots,i$. Si $l=0$, l'hypothèse de récurrence est satisfaite. Supposons l'hypothèse vraie à l'étape $l \geq 0$.

Comme la matrice jacobienne J est inversible par la Proposition 14, les $\hat{v}_j$ ne sont rien d'autre que le résultat d'une itération de Newton à plusieurs variables généralisant celle vue dans la Section 3 du Chapitre 3. Par conséquent on obtient que $f_j(a_0,X_1,\dots,X_r,\hat{v}_{r+1},\dots,\hat{v}_n) \mod \hat{Q} = 0$ à la précision $(X_1-a_1,\dots,X_r-a_r)^{2^{l+1}}$ pour chaque $j \in \{1,\dots,i\}$.

Notons ensuite que Δ, Δ_Q et les Δ_j sont de valuation au moins 2^l. Par conséquent, à la précision 2^{l+1} on a $(\hat{Q}-\Delta_Q)(T+\Delta) = \hat{Q}(T+\Delta) - \Delta_Q(T) = \hat{Q}(T) + \hat{Q}'(T)\Delta - \Delta_Q(T) = 0$ modulo $\hat{Q}$. Pour $j \geq r+1$ on vérifie aussi que l'on a $(\hat{v}_j - \Delta_j)(T+\Delta) = \hat{v}_j(T+\Delta) - \Delta_j(T) = \hat{v}_j(T) + \hat{v}_j'(T)\Delta - \Delta_j(T) = 0$

Résolution géométrique

Entrée : des valeurs t, $a_0, \ldots, a_r$ dans k, un élément primitif u de B'_i, et les $q(a_0, \ldots, a_r, T), v_{r+1}(a_0, \ldots, a_r, T), \ldots, v_n(a_0, \ldots, a_r, T)$, où $q, v_{r+1}, \ldots, v_n$ est une résolution géométrique de B'_i.

Sortie : $Q(a_0, \ldots, a_{r-1}, T), V_r(a_0, \ldots, a_{r-1}, T), \ldots, V_n(a_0, \ldots, a_{r-1}, T)$ où $Q, V_r, \ldots, V_n$ est la résolution géométrique de B'_{i+1} pour l'élément primitif $U := X_r + tu$.

1. Appeler l'algorithme de la Figure 3 pour calculer $q(a_0, \ldots, a_{r-1}, X_r, T)$, $w_{r+1}(a_0, \ldots, a_{r-1}, X_r, T)$, …, $w_n(a_0, \ldots, a_{r-1}, X_r, T)$.
2. Appeler l'algorithme de la Figure 2 pour calculer $Q(a_0, \ldots, a_{r-1}, T)$, $V_r(a_0, \ldots, a_{r-1}, T)$, …, $V_n(a_0, \ldots, a_{r-1}, T)$.

FIGURE 4. Étape incrémentale de l'algorithme résolution

modulo $\hat{Q}$, et à la précision 2^{l+1}. Finalement à la fin de l'étape l on obtient bien l'hypothèse de récurrence de l'étape $l+1$.

Comme, pour un élément primitif fixé la résolution géométrique est unique, la correction de l'algorithme vient du fait que les polynômes $q(a_0, X_1, \ldots, X_r, T)$ et les $w_j(a_0, X_1, \ldots, X_r, T)$ coïncident respectivement avec $\hat{Q}$ et les $\hat{V}_j/\hat{Q}' \mod \hat{Q}$ jusqu'à la précision courante. La borne δ_i sur la précision nécessaire vient de la Proposition 1.

L'inversion de la matrice jacobienne à l'étape $l = 0$ doit se faire avec un algorithme sans division, ce qui coûte $\tilde{O}(n^4\delta_i)$ opérations dans k, par l'algorithme de Berkowitz [1]. L'amélioration de la précision de cet inverse à l'étape l requiert ensuite $\tilde{O}(n^3\delta_i 2^l)$ opérations par la méthode de la Section 2.7 du Chapitre 3.

L'évaluation des f_j et de J s'effectue avec $\tilde{O}(nL\delta_i 2^l)$ opérations dans k. Les calculs restant totalisent $\tilde{O}(n\delta_i 2^l)$ opérations dans k supplémentaires. □

5. Résolution géométrique

La première idée de l'algorithme de résolution géométrique est de calculer des résolutions géométriques successivement pour $B_1, B_2, B_3, \ldots$ en prenant des coordonnées suffisamment génériques pour que (H_1) et (H_2) soient vérifiées et en appliquant l'algorithme de la Figure 2. Se limiter à cette seule idée conduit à des calculs coûteux car les fractions rationnelles dans les variables libres $X_0, \ldots, X_r$ de l'étape i deviennent rapidement de plus en plus volumineuses.

La deuxième idée de l'algorithme de résolution géométrique est donc de s'affranchir de ces fractions volumineuses en calculant des résolutions géométriques pour $B_i/(X_0 - a_0, X_1 - a_1, \ldots, X_r - a_r)$ pour i de 1 à s, si les a_i sont pris suffisamment génériques. L'algorithme est résumé dans la Figure 4.

PROPOSITION 16. *Sous les hypothèses (H_1) et (H_2), il existe des variétés algébriques propres W_1 et W_2 de $\mathbb{P}^n$ telles que, pour tout $(\lambda_0 : \cdots : \lambda_n) \notin W_1$, tout $(a_0 : \cdots : a_n) \notin W_2$, et sauf pour un nombre fini de valeurs de t, l'algorithme de la Figure 2 est correct avec $u = \lambda_0 X_0 + \cdots + \lambda_n X_n$ et utilise $\tilde{O}((n^4 + nL)\delta_{i+1}^2)$ opérations dans k.*

DÉMONSTRATION. Il s'agit d'une conséquence des propositions 12 et 15. □

Nous pouvons enfin résumer le résultat principal de ce chapitre :

THÉORÈME 1. *Soit k un corps de caractéristique zéro, et soit I un idéal engendré par une suite régulière réduite de polynômes homogènes $f_1, \ldots, f_s$ de*

$k[X_0, \ldots, X_n]$ de degré au moins 2. Il existe une variété algébrique propre W_0 de $\mathcal{M}_{n+1}(k)$, et des variétés algébriques propres W_1 et W_2 de $\mathbb{P}^n$ telles que, pour tout $M \notin W_0$, tout $(\lambda_0 : \cdots : \lambda_n) \notin W_1$, tout $(a_0 : \cdots : a_n) \notin W_2$, et sauf pour un nombre fini de valeurs de t dans k, la matrice M est inversible et on peut calculer une résolution géométrique de $I \circ M + (X_0 - a_0, \ldots, X_{n-s} - a_{n-s})$ en utilisant $\tilde{O}((n^4 + nL)\delta_s^2)$ opérations dans k.

DÉMONSTRATION. La preuve découle de la proposition précédente, en remarquant que $\delta_{i+1} \geq 2\delta_i$ lorsque l'on somme les coûts des différentes étapes. □

Des exemples illustrant le déroulement de l'algorithme se trouvent dans l'article [3] et la thèse [2].

Notes

L'algorithme de résolution géométrique et aussi appelé parfois l'algorithme *Kronecker* en hommage aux travaux de L. Kronecker sur la résolution des systèmes algébriques [16]. Ce chapitre et une version simplifiée, adapté au seul cas projectif de l'article [3]. Une présentation plus détaillée dans le cas affine, comprenant des exemples et des illustrations, se trouve aussi dans la thèse [2]. L'analyse de complexité est adaptée de [10]. Une implémentation en C++ est disponible dans la librairie GEOMSOLVEX de MATHEMAGIX (www.mathemagix.org). Les résultats présentés dans ce chapitre sont en fait le fruit d'une séries de travaux initiés par Giusti, Heintz, Morais et Pardo au début des années 90 [4, 5, 6, 7, 8, 9, 15]. Une présentation historique est donnée dans l'introduction de [3].

Dans le cas ou $k = \mathbb{Q}$ et $(f_1, \ldots, f_s)$ est zéro dimensionnel, une bonne stratégie de calcul consiste à appliquer d'abord l'algorithme de résolution géométrique modulo un nombre premier suffisamment grand pour ne pas annuler les dénominateurs des nombres intervenant lors du calcul sur $\mathbb{Q}$, mais pas trop grand non plus pour que l'arithmétique modulaire soit rapide. Ensuite il est possible d'adapter l'algorithme de la Figure 3 pour remonter des entiers p-adiques et de récupérer la résolution géométrique sur $\mathbb{Q}$. Nous renvoyons à l'article [10] pour plus de détails.

L'algorithme de résolution est probabiliste mais dans le cas projectif d'une suite régulière réduite, on connaît les degrés des idéaux intermédiaires et on peut vérifier que si le choix du changement de variables et des autres paramètres ne sont pas bons alors le nombre de points solutions trouvé est strictement inférieur à celui attendu. Par ailleurs, si la suite n'est pas régulière, l'ensemble des solutions au problème doit être décomposé en une union de variétés algébriques de différentes dimensions. On parle alors de *décomposition équidimensionelle*. Si la suite n'est pas réduite, alors il faut utiliser des généralisations de la méthode de Newton en présence de racines multiples, comme par exemple des techniques dites de *déflation*. Plusieurs articles traitent de ces questions [11, 12, 13, 14, 18, 19, 20]. Une courte description de ces travaux de généralisation se trouve aussi dans l'introduction de l'article [3].

Bibliographie

[1] BERKOWITZ, Stuart J. (1984). « On Computing the determinant in small parallel time using a small number of processors ». In : *Information Processing Letters*, vol. 18, p. 147–150.

[2] DURVYE, Clémence (2008). « Algorithmes pour la décomposition primaire des idéaux polynomiaux de dimension nulle donnés en évaluation ». Thèse de doct. Université de Versailles Saint-Quentin-en-Yvelines.

[3] DURVYE, Clémence et Grégoire LECERF (2007). « A Concise Proof of the Kronecker Polynomial System Solver from Scratch ». In : *Expositiones Mathematicae*, vol. 26, n°2, p. 101–139.

[4] Fitchas, N., M. Giusti et F. Smietanski (1995). « Sur la complexité du théorème des zéros ». In : *Approximation and optimization in the Caribbean, II (Havana, 1993)*. Vol. 8. Approx. Optim. Frankfurt am Main : Lang, p. 274–329.

[5] Giusti, M., K. Hägele, J. Heintz, J. L. Montaña, J. E. Morais et L. M. Pardo (1997). « Lower bounds for Diophantine approximations ». In : *J. Pure Appl. Algebra*, vol. 117/118, p. 277–317.

[6] Giusti, M., J. Heintz, J. E. Morais, J. Morgenstern et L. M. Pardo (1998). « Straight-line programs in geometric elimination theory ». In : *J. Pure Appl. Algebra*, vol. 124, n°1-3, p. 101–146.

[7] Giusti, M., J. Heintz, J. E. Morais et L. M. Pardo (1995). « When polynomial equation systems can be "solved" fast ? » In : *Applied algebra, algebraic algorithms and error-correcting codes (Paris, 1995)*. Vol. 948. Lecture Notes in Computer Science. Springer-Verlag, p. 205–231.

[8] Giusti, M., J. Heintz et J. Sabia (1993). « On the efficiency of effective Nullstellensätze ». In : *Computational Complexity*, vol. 3, n°1, p. 56–95.

[9] Giusti, Marc et Joos Heintz (1993). « La détermination des points isolés et de la dimension d'une variété algébrique peut se faire en temps polynomial ». In : *Computational algebraic geometry and commutative algebra (Cortona, 1991)*. Vol. XXXIV. Sympos. Math. Cambridge : Cambridge Univ. Press, p. 216–256.

[10] Giusti, Marc, Grégoire Lecerf et Bruno Salvy (2001). « A Gröbner free alternative for polynomial system solving ». In : *Journal of Complexity*, vol. 17, n°1, p. 154–211.

[11] Jeronimo, G., T. Krick, J. Sabia et M. Sombra (2004). « The computational complexity of the Chow form ». In : *Foundations of Computational Mathematics. The Journal of the Society for the Foundations of Computational Mathematics*, vol. 4, n°1, p. 41–117.

[12] Jeronimo, G., S. Puddu et J. Sabia (2001). « Computing Chow forms and some applications ». In : *J. Algorithms*, vol. 41, n°1, p. 52–68.

[13] Jeronimo, G. et J. Sabia (2000). « Probabilistic equidimensional decomposition ». In : *Comptes Rendus des Séances de l'Académie des Sciences. Série I. Mathématique*, vol. 331, n°6, p. 485–490.

[14] — (2002). « Effective equidimensional decomposition of affine varieties ». In : *Journal of Pure and Applied Algebra*, vol. 169, n°2-3, p. 229–248.

[15] Krick, T. et L. M. Pardo (1996). « A computational method for Diophantine approximation ». In : *Algorithms in algebraic geometry and applications (Santander, 1994)*. Vol. 143. Progr. Math. Birkhäuser, p. 193–253.

[16] Kronecker, Leopold (1882). « Grundzüge einer arithmetischen Theorie der algebraischen Grössen ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 92, p. 1–122.

[17] Lang, Serge (2002). *Algebra*. 3e éd. Vol. 211. Graduate Texts in Mathematics. Springer-Verlag, xvi+914 pages.

[18] Lecerf, G. (2000). « Computing an equidimensional decomposition of an algebraic variety by means of geometric resolutions ». In : *ISSAC'00 : International Symposium on Symbolic and Algebraic Computation*. ACM, p. 209–216.

[19] — (2002). « Quadratic Newton Iteration for Systems with Multiplicity ». In : *Found. Comput. Math.* vol. 2, n°3, p. 247–293.

[20] — (2003). « Computing the equidimensional decomposition of an algebraic closed set by means of lifting fibers ». In : *J. Complexity*, vol. 19, n°4, p. 564–596.

Sixième partie

Sommation et intégration de suites et fonctions spéciales

CHAPITRE 27

Solutions hypergéométriques de récurrences et sommation hypergéométrique

Résumé

Le Chapitre 16 a indiqué comment un petit noyau d'algorithmes relativement simples permet de trouver les solutions polynomiales et rationnelles de récurrences linéaires, et les a appliqués à la recherche de sommes indéfinies de suites hypergéométriques. Nous adaptons et appliquons ici tout cet ensemble d'algorithmes à la recherche de solutions hypergéométriques de récurrences linéaires et pour trouver des sommes définies de suites hypergéométriques. La suite du cours prolongera les méthodes de ce chapitre aux questions de sommation et d'intégration de suites et fonctions spéciales plus générales, solutions de systèmes d'équations différentielles et de récurrence d'ordre quelconque.

Voici deux exemples de sommes dont le traitement algorithmique est détaillé dans ce cours ; pour la seconde, on adopte la convention que les binomiaux $\binom{a}{b}$ sont nuls lorsque $b < 0$ ou $b > a$:

$$ {}_2F_1\left(\begin{matrix}a, b\\ c\end{matrix}\middle| 1\right) = \sum_{k=0}^{\infty} \frac{(a)_k (b)_k}{(c)_k k!} = \frac{\Gamma(c-a-b)\,\Gamma(c)}{\Gamma(c-a)\,\Gamma(c-b)}, $$

$$ \sum_{k\in\mathbb{Z}} (-1)^k \binom{a+b}{a+k}\binom{a+c}{c+k}\binom{b+c}{b+k} = \frac{(a+b+c)!}{a!\,b!\,c!}. $$

Ces deux sommes sont classiques : la première est due à Gauss, la seconde à Dixon. Pour justifier de telle sommes *définies* — où toutes les valeurs possibles de l'indice de sommation sont parcourues —, il serait agréable de pouvoir faire appel, comme dans le cas de l'intégration où l'on recherche une primitive, à une somme *indéfinie* — c'est-à-dire vue comme fonction de sa borne supérieure — et de voir la somme définie comme différence de deux évaluations. Cependant, pour ces deux exemples il n'existe pas de somme indéfinie hypergéométrique, et c'est le cas général. Ces sommes définies peuvent néanmoins être calculées automatiquement, si l'on prend la peine de rechercher une suite auxiliaire qui, elle, admet une somme indéfinie hypergéométrique.

Une chaîne complète d'algorithmes permettant de trouver les membres droits des identités ci-dessus est détaillée dans ce chapitre : l'algorithme de Gosper de la section 1 décide de l'existence d'une somme indéfinie hypergéométrique pour un sommant hypergéométrique ; en modifiant convenablement un sommant hypergéométrique, l'algorithme de Zeilberger de la section 2 trouve, quand il en existe un, une récurrence vérifiée par une somme hypergéométrique définie. Pour ne pas s'arrêter sur une récurrence et revenir à une forme explicite, l'algorithme de Petkovšek de la section 3 obtient toutes les solutions d'une récurrence linéaire qui sont données comme combinaisons linéaires de termes hypergéométriques.

1. Sommation hypergéométrique indéfinie. Algorithme de Gosper

La recherche de formes closes pour la sommation est souvent naturellement posée en terme de suites hypergéométriques. Après quelques définitions et propriétés de ces suites, nous abordons leur sommation.

1.1. Suites hypergéométriques.

DEFINITION 1. On dit qu'une suite (u_n) est *hypergéométrique* s'il existe deux polynômes P et Q tels que pour tout $n \in \mathbb{N}$,

$$Q(n)u_{n+1} = P(n)u_n. \tag{1}$$

Autrement dit, elle est polynomialement récurrente et vérifie une récurrence linéaire *d'ordre 1*.

Cette récurrence étant d'ordre 1, il est facile d'en écrire la solution générale au moins lorsque Q n'a pas de racine entière positive :

$$u_n = u_0 \left(\frac{\mathrm{lc}(P)}{\mathrm{lc}(Q)}\right)^n \frac{\prod_{P(\alpha)=0}(-\alpha)(-\alpha+1)\cdots(-\alpha+n-1)}{\prod_{Q(\beta)=0}(-\beta)(-\beta+1)\cdots(-\beta+n-1)},$$

où $\mathrm{lc}(p)$ désigne le coefficient de tête du polynôme p et où chaque produit doit être compris avec répétition des racines selon leurs multiplicités.

Les suites hypergéométriques jouent un rôle important en analyse classique, où elles apparaissent comme coefficients de Taylor des séries introduites par la définition suivante.

DEFINITION 2. On appelle *série hypergéométrique généralisée* et on note

$${}_pF_q\left(\begin{matrix}a_1,\ldots,a_p\\ b_1,\ldots,b_q\end{matrix}\middle| x\right)$$

la série

$$\sum_{n\geq 0} \frac{(a_1)_n\cdots(a_p)_n}{(b_1)_n\cdots(b_q)_n}\frac{x^n}{n!},$$

où la notation $(a)_n$ (appelée *symbole de Pochhammer*) représente le produit $a(a-1)\cdots(a-n+1)$.

Des cas particuliers de séries qui s'expriment à l'aide de séries hypergéométriques généralisées sont les séries $\exp(x)$, $\log(1+x)$, les dilogarithmes et polylogarithmes, les fonctions J_ν de Bessel, les fonctions d'Airy, etc. Ainsi :

$$\exp(x) = {}_0F_0\left(\begin{matrix}-\\ -\end{matrix}\middle| x\right),$$

$$J_\nu(x) = \left(\frac{x}{2}\right)^\nu \frac{1}{\Gamma(\nu+1)}{}_0F_1\left(\begin{matrix}-\\ \nu+1\end{matrix}\middle| x\right),$$

$$\mathrm{Ai}(x) = -\frac{x\sqrt[6]{3}\Gamma(2/3)}{2\pi}{}_0F_1\left(\begin{matrix}-\\ 4/3\end{matrix}\middle| \frac{x^3}{9}\right) + \frac{\sqrt[3]{3}}{3\Gamma(2/3)}{}_0F_1\left(\begin{matrix}-\\ 2/3\end{matrix}\middle| \frac{x^3}{9}\right).$$

EXERCICE 1. Récrire l'identité de Dixon en terme de l'évaluation d'une série ${}_3F_2$ en 1.

1.2. Algorithme de Gosper. Étant donnée une suite hypergéométrique $u(n)$, le problème de sommation indéfinie hypergéométrique de $u(n)$ consiste à déterminer s'il existe une autre suite hypergéométrique $U(n)$ telle que $U(n+1) - U(n) = u(n)$, et si oui, la calculer.

Une observation simple est formulée dans le lemme suivant.

LEMME 1. *Si $U(n)$ est une suite hypergéométrique et L un opérateur de récurrence linéaire à coefficients polynomiaux, alors il existe une fraction rationnelle $r(n)$ telle que $(L \cdot U)(n) = r(n)U(n)$.*

DÉMONSTRATION. Si $U(n)$ est hypergéométrique, par définition, il existe une fraction rationnelle $R(n)$ telle que $U(n+1) = R(n)U(n)$ et donc par récurrence $U(n+k) = R(n+k-1)\cdots R(n)U(n)$ pour tout k. Le résultat s'en déduit en additionnant les contributions. □

Dans le langage du Chapitre 28, la fonction $r(n)$ n'est autre que le reste de la division euclidienne non commutative de L par le polynôme unitaire de degré 1 donnant la récurrence qui annule $U(n)$.

Ce lemme entraîne qu'une somme hypergéométrique $U(n)$ de $u(n)$ doit être le produit de $u(n)$ par une fraction rationnelle $R(n)$. Diviser la récurrence $U(n+1) - U(n) = u(n)$ par $u(n)$ réduit alors le calcul à celui de la recherche de solutions rationnelles de l'équation inhomogène

$$R(n+1)\frac{u(n+1)}{u(n)} - R(n) = 1, \tag{2}$$

d'ordre 1, problème traité dans au Chapitre 16, particulièrement en Section 4.2. L'algorithme ainsi obtenu est connu sous le nom d'algorithme de Gosper [3].

EXERCICE 2. Montrer qu'il y a unicité de la somme indéfinie hypergéométrique de $u = (u_n)$ si et seulement si u n'est pas rationnelle.

EXERCICE 3. Calculer une somme indéfinie hypergéométrique de

$$\frac{2^k(k-1)}{k(k+1)}.$$

EXEMPLE 1. L'algorithme de Gosper permet également de donner des réponses négatives. Voici en détail comment il permet de prouver par l'absurde que la suite des $S(n) := \sum_{k=1}^{n} 1/k!$ n'est pas hypergéométrique.

Supposons que $S(n)$ est hypergéométrique. Alors, elle doit être le produit de $1/n!$ par une fraction rationnelle $r(n)$. Cette fraction est donc solution de la récurrence

$$S(n+1) - S(n) = \frac{1}{(n+1)!} = \frac{r(n+1)}{(n+1)!} - \frac{r(n)}{n!}.$$

En chassant les dénominateurs, il reste

$$r(n+1) - (n+1)r(n) = 1.$$

Le résultat de l'algorithme « Multiple du dénominateur » d'Abramov p. 234 montre que r doit être un polynôme : les pôles de plus petite partie réelle de r doivent être des zéros du pgcd, trivial, de 1 et $n+1$. Enfin, r ne peut pas non plus être un polynôme : son terme de plus haut degré ne disparaît pas par évaluation de la récurrence.

1.3. Sommation paramétrée. Soient des suites $u_1(n), \ldots, u_s(n)$ hypergéométriques et deux à deux similaires, au sens où tous les rapports $u_i(n)/u_j(n)$ sont rationnels. Il s'agit de déterminer, si elles existent, une suite hypergéométrique $S(n)$ et des constantes $\lambda_1, \ldots, \lambda_s$ telles que

$$S(n+1) - S(n) = \lambda_1 u_1(n) + \cdots + \lambda_s u_s(n).$$

Comme toutes les u_i sont similaires à une même suite $u(n)$, le membre droit de (2) peut être remplacé par une combinaison linéaire des λ_i à coefficients des fractions rationnelles et la méthode du Chapitre 16 pour les récurrences linéaires inhomogènes paramétrées s'applique.

2. Sommation hypergéométrique définie. Algorithme de Zeilberger

L'algorithme considéré dans cette section s'applique à des suites u dites hypergéométriques en deux variables, c'est-à-dire pour lesquelles

$$\frac{u(n+1,k)}{u(n,k)} \quad \text{et} \quad \frac{u(n,k+1)}{u(n,k)}$$

sont deux fractions rationnelles en n et k.

Exercice 4. Soit une suite hypergéométrique $u = (u_{n,k})$, donnée par les rapports rationnels $a(n,k) = u_{n+1,k}/u_{n,k}$ et $b(n,k) = u_{n,k+1}/u_{n,k}$. Montrer que les fractions rationnelles a et b satisfont une relation que l'on explicitera.

Le problème de sommation définie hypergéométrique est de déterminer si pour une telle suite u, la nouvelle suite

$$U(n) = \sum_k u(n,k)$$

vérifie une récurrence linéaire et si oui, la calculer. L'absence de bornes explicites de sommation signifie que la somme porte en fait sur toutes les valeurs $k \in \mathbb{Z}$, étant entendu que, bien souvent, ces sommes ont en fait un support fini. Bien souvent aussi, l'intervalle de sommation s'arrête aux *bornes naturelles* de sommation, c'est-à-dire aux indices où par construction la suite et toutes ses décalées par rapport à l'autre indice, n, s'annulent.

Exemple 2. Des sommes simples sont

$$\sum_k \binom{n}{k} = 2^n \quad \text{et} \quad \sum_k \binom{n}{k}^2 = \binom{2n}{n}.$$

Dans ces deux exemples, pour k en dehors de $\{0,\dots,n\}$, les binomiaux sont nuls. Ainsi, les bornes 0 et n sont naturelles.

Un cas très favorable pour obtenir des sommations explicites est lorsque $U(n)$ est elle-même hypergéométrique. Mais pour bien des applications, il est tout aussi utile de trouver une récurrence linéaire à laquelle $U(n)$ obéit. C'est ce que réalise l'algorithme de Zeilberger, par une approche dite de *télescopage créatif*. Trouver si la somme est hypergéométrique se ramène alors à la recherche de solutions hypergéométriques de récurrences linéaires, problème qui est traité par l'algorithme de Petkovšek en Section 3.

2.1. Principe du télescopage créatif.

On cherche un opérateur

$$P(n,S_n) - (S_k - 1)Q(n,k,S_n,S_k) \tag{3}$$

qui annule la suite $u(n,k)$ à sommer (S_n et S_k désignent les opérateurs de décalage en n et k, $(S_n \cdot u)(n,k) = u(n+1,k)$ et $(S_k \cdot u)(n,k) = u(n,k+1)$). Une fois un tel opérateur trouvé, la sommation sur k est aisée, et, sous l'hypothèse de bornes naturelles, conduit à l'égalité

$$\big(P(n,S_n)\cdot U\big)(n) = 0. \tag{4}$$

Exemple 3. En suivant ce modèle, voici une preuve compliquée que

$$U(n) = \sum_k \binom{n}{k} = 2^n. \tag{5}$$

Le point de départ est l'identité du triangle de Pascal

$$\binom{n+1}{k+1} = \binom{n}{k+1} + \binom{n}{k},$$

qui se transpose en l'opérateur annulateur

$$S_n S_k - S_k - 1.$$

Ce dernier se récrit

$$(S_k - 1)(S_n - 1) + (S_n - 2),$$

opérateur qui traduit la relation explicite

$$\left(\binom{n+1}{k+1} - \binom{n+1}{k}\right) - \left(\binom{n}{k+1} - \binom{n}{k}\right) + \left(\binom{n+1}{k} - 2\binom{n}{k}\right) = 0.$$

En sommant sur $k \in \mathbb{Z}$, les premiers termes se téléscopent deux à deux, et il reste

$$((S_n - 2) \cdot U)(n) = U(n+1) - 2U(n) = 0.$$

La fin de la preuve se ramène à vérifier la condition initiale $S(0) = 1$.

En l'absence de bornes naturelles, (4) s'écrit plutôt

$$(P \cdot U)(n) = (Q \cdot u)(n, b+1) - (Q \cdot u)(n, a).$$

Les clôtures du Chapitre 14 fournissent alors un opérateur $\tilde{P}(n, S_n)$ qui annule le membre droit de l'égalité ci-dessus. On obtient ainsi une équation homogène, sous la forme

$$(\tilde{P}P) \cdot U = 0.$$

2.2. Algorithme de Zeilberger. Il reste à voir comment trouver les opérateurs P et Q de l'équation (3) dans le cas général. L'idée de Zeilberger est que cette équation,

$$\big(P(n, S_n) \cdot u\big)(n, k) = \big((S_k - 1)Q(n, k, S_n, S_k) \cdot u\big)(n, k),$$

signifie que $Q \cdot u$ est une somme hypergéométrique *indéfinie* de $P(n, S_n) \cdot u$ par rapport à k. Comme u est supposée hypergéométrique, $Q \cdot u$ est en fait de la forme $q(n, k)u(n, k)$ pour une fraction rationnelle q : si P était connu, l'algorithme de Gosper saurait donner q.

La méthode donnée par Zeilberger procède alors incrémentalement sur le degré de P en S_n. Pour $m = 0, 1, \ldots$, il cherche s'il existe des $\lambda_i(n)$ (les coefficients de P) tels que

$$\lambda_0(n)u(n, k) + \lambda_1(n)u(n+1, k) + \cdots + \lambda_m(n)u(n+m, k)$$

ait une somme hypergéométrique. L'algorithme pour cela est la variante paramétrée de l'algorithme de Gosper, présenté au Chapitre 16.

Exercice 5. Exécuter l'algorithme de Zeilberger sur la somme (5) de l'exemple 3. Comparer les résultats des calculs et les preuves de l'identité selon les deux approches.

Exercice 6. Évaluer en termes du symbole de Pochhammer $(x)_n = \Gamma(x+n)/\Gamma(x)$ la somme hypergéométrique

$${}_2F_1\left(\begin{matrix}-n, b\\ c\end{matrix}\,\middle|\, 1\right).$$

Exercice 7. On veut montrer, pour tout $n \in \mathbb{N}$, l'identité

$${}_2F_1\left(\begin{matrix}-2n, 2n+2\alpha+1\\ \alpha+1\end{matrix}\,\middle|\, \frac{1-x}{2}\right) = {}_2F_1\left(\begin{matrix}-n, n+\alpha+\frac{1}{2}\\ \alpha+1\end{matrix}\,\middle|\, 1-x^2\right).$$

Indiquer (sans faire les calculs) comment on peut utiliser l'algorithme de Zeilberger pour arriver à ce résultat.

2.3. Terminaison. La méthode ne termine pas en général. Wilf et Zeilberger ont été les premiers à délimiter une classe importante, celle des suites *proprement hypergéométriques* sur lesquelles la terminaison et par conséquent le succès de la méthode sont garantis. Ces suites sont de la forme

$$u(n,k) = P(n,k)Z^k \frac{\prod_{i=1}^{\ell}(a_i n + b_i k + c_i)!}{\prod_{i=1}^{m}(u_i n + v_i k + d_i)!},$$

où les a_i, b_i, u_i et v_i sont des *entiers*, ℓ et m sont des entiers positifs, P est un polynôme et Z une constante.

PROPOSITION 1. *L'algorithme de Zeilberger termine si u est proprement hypergéométrique.*

DÉMONSTRATION. L'idée de la preuve est de prouver un résultat légèrement plus fort, à savoir l'existence d'un polynôme $A(n, S_k, S_n)$ qui ne dépend pas de k et annule u. Une division euclidienne de A par $S_k - 1$ permet d'en déduire un couple (P, Q) tel que l'opérateur (3) annule u *et de plus* Q ne dépend pas de k. L'existence de A est obtenue en considérant les monômes en n, S_n, S_k par degré total croissant et leur action sur u. L'application de ces monômes sur u produit un multiple rationnel de u. Comme les coefficients a_i, b_i, u_i et v_i sont des entiers, le nombre de nouveaux facteurs de ces fractions rationnelles finit par croître moins vite que le nombre de monômes, et il s'ensuit l'existence d'une relation de liaison. La partie technique de la preuve se concentre donc sur l'étude soigneuse des facteurs de ces fractions rationnelles et de leur nombre. De plus, il convient de s'assurer que le P obtenu est non nul. Pour cela, on montre qu'un tel P peut toujours être obtenu non nul, moyennant au besoin une légère modification sur A. □

3. Solutions hypergéométriques. Algorithme de Petkovšek

Si $(L \cdot u)(n) = 0$ avec $u(n)$ hypergéométrique, il existe deux polynômes P et Q vérifiant

$$u(n+1) = \frac{P(n)}{Q(n)} u(n),$$

et on peut prendre Q unitaire. Il s'ensuit que

$$u(n+\ell) = \frac{P(n+\ell-1)}{Q(n+\ell-1)} \cdots \frac{P(n)}{Q(n)} u(n).$$

Pour une récurrence donnée par un opérateur $L = a_m S_n^m + \cdots + a_0$ une condition nécessaire et suffisante d'existence de solution hypergéométrique s'écrit

$$\begin{aligned}(6)\quad & a_m(n)P(n+m-1)P(n+m-2)\cdots P(n) \\ & \quad + a_{m-1}(n)Q(n+m-1)P(n+m-2)\cdots P(n) \\ & \qquad + \cdots + a_0(n)Q(n+m-1)\cdots Q(n) = 0.\end{aligned}$$

On peut voir cette expression comme une renormalisation du reste de la division euclidienne de L par $S_n - P/Q$, après réduction au même dénominateur. Si on savait prédire que $\operatorname{pgcd}\big(P(n), Q(n+i)\big) = 1$ pour $i = 0, \ldots, m-1$, alors on en déduirait facilement que $P(n)$ divise $a_0(n)$ et $Q(n+m-1)$ divise $a_m(n)$. Ceci nous donnerait un algorithme : toute paire de facteurs unitaires de $a_0(n)$ et $a_m(n-m+1)$ donnerait un candidat pour $Q(n)$ et un candidat pour P à une constante près, il suffirait alors d'injecter ces candidats dans (6) et de chercher s'il existe une constante satisfaisant à l'équation. En itérant sur les facteurs de a_0 et a_m, le travail serait terminé.

En général, il se peut très bien que $P(n)$ et $Q(n+i)$ aient des facteurs communs. La solution trouvée par Petkovšek consiste à recourir à une décomposition plus forte

de la fraction rationnelle. On cherche une solution telle que

$$\frac{u(n+1)}{u(n)} = Z\frac{A(n)}{B(n)}\frac{C(n+1)}{C(n)},$$

où Z est une constante, A, B, C sont des polynômes unitaires, $\operatorname{pgcd}(A,C) = 1$, $\operatorname{pgcd}\big(B(n), C(n+1)\big) = 1$ et $\operatorname{pgcd}\big(A(n), B(n+i)\big) = 1$ pour tout $i \in \mathbb{N}$. Cette décomposition des fractions rationnelles est appelée décomposition de Gosper–Petkovšek.

Exercice 8. Montrer que toute fraction rationnelle peut se décomposer sous la forme ci-dessus. Donner un algorithme calculant les polynômes A, B, C et les constantes Z correspondant à une fraction rationnelle donnée en entrée. On pourra représenter Z par son polynôme annulateur minimal. (Indice : s'inspirer de l'algorithme d'Abramov.)

Avec cette décomposition, l'équation (6) devient

$$\begin{aligned}(7)\quad & Z^m a_m(n)A(n+m-1)\cdots A(n)C(n+m)+\\ & Z^{m-1}a_{m-1}(n)B(n+m-1)A(n+m-2)\cdots A(n)C(n+m-1)\\ & \qquad +\cdots+a_0(n)B(n+m-1)\cdots B(n)C(n)=0.\end{aligned}$$

Les contraintes de la décomposition permettent de déduire immédiatement

$$A(n) \mid a_0(n), \qquad B(n+m-1) \mid a_m(n).$$

L'algorithme de Petkovšek s'en déduit : pour chaque paire (A, B) de facteurs de a_0 et a_m, le coefficient de tête du polynôme au membre gauche de (7) donne une équation polynomiale sur Z. Une fois Z ainsi fixé, il reste à chercher les solutions polynomiales C de l'équation, s'il en existe.

Exercice 9. Résoudre ainsi

$$(n-1)u(n+2) - (n^2+3n-2)u(n+1) + 2n(n+1)u(n) = 0,$$
$$u(n+2) - (2n+1)u(n+1) + (n^2-2)u(n) = 0.$$

On peut noter que la recherche de solutions hypergéométriques de récurrences linéaires *inhomogènes* n'est pas difficile : le membre droit doit être hypergéométrique et, outre les solutions hypergéométriques de la partie homogène, la solution doit être le produit du membre droit par une fraction rationnelle. Ceci ramène le problème à la recherche de solutions rationnelles.

Notes

Le livre de Petkovšek, Wilf et Zeilberger intitulé $A = B$ [4] est une bonne introduction aux questions abordées dans ce cours. L'algorithme de Petkovšek possède une extension intéressante [2], par réduction de l'ordre, à une plus grande classe de solutions, appelées *solutions d'Alembertiennes*.

La proposition 1 ne donne qu'une condition suffisante pour l'existence d'un $P(n, S_n)$ et conséquemment pour la terminaison de l'algorithme de Zeilberger. Il existe des suites qui ne sont pas proprement hypergéométriques et pour lesquelles la méthode fonctionne quand même. Cela a donné lieu à une série de travaux, et le dernier mot revient à un travail d'Abramov [1] qui donne un test algorithmique de terminaison de l'algorithme.

De nombreuses généralisations de l'algorithme de Zeilberger sont possibles : par exemple la suite à sommer peut ne pas être hypergéométrique, tout en étant P-récurrente, ou il peut s'agir d'une suite de fonctions différentiellement finies et l'on cherche une équation différentielle satisfaite par la somme définie, etc. Ces questions seront abordées au Chapitre 30.

Bibliographie

[1] Abramov, S. A. (2003). « When does Zeilberger's algorithm succeed ? » In : *Advances in Applied Mathematics*, vol. 30, n°3, p. 424–441.

[2] Abramov, Sergei A. et Marko Petkovšek (1994). « D'Alembertian solutions of linear differential and difference equations ». In : *ISSAC'94 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 169–174.

[3] Gosper, R. William (1978). « Decision procedure for indefinite hypergeometric summation ». In : *Proceedings of the National Academy of Sciences USA*, vol. 75, n°1, p. 40–42.

[4] Petkovšek, Marko, Herbert S. Wilf et Doron Zeilberger (1996). $A = B$. A. K. Peters, xii+212 pages.

CHAPITRE 28

Équations fonctionnelles linéaires et polynômes tordus

Résumé

Une certaine variété de polynômes non commutatifs fournit une représentation unifiée pour une large classe d'équations fonctionnelles linéaires. Celle-ci s'avère bien adaptée pour les calculs. Nous réinterprétons nombre des algorithmes vus dans ce cours dans ce point de vue.

1. Des polynômes non commutatifs pour calculer avec des opérateurs linéaires

Dans les années 1930, le mathématicien Oystein Ore (1899–1968) s'est intéressé à la résolution de systèmes linéaires liant des dérivées $f_i^{(j)}(x)$, des décalées $f_i(x+j)$, ou les substitutions $f_i(q^j x)$ de fonctions inconnues $f_i(x)$. À cette fin, il a introduit de nouvelles familles de polynômes en une variable ayant la propriété que cette variable ne commute pas avec les coefficients des polynômes. Ce défaut de commutativité reflète une sorte de loi de Leibniz.

Rappelons la relation de Leibniz pour deux fonctions quelconques f et g :

$$(fg)'(x) = f'(x)g(x) + f(x)g'(x).$$

En notant D l'opérateur de dérivation, M celui qui à une fonction f associe la fonction donnée par $M(f)(x) = xf(x)$, id l'opérateur identité sur les fonctions, et $\circ$ la composition d'opérateurs, la règle de Leibniz donne, pour $f(x) = x$ et g quelconque,

$$(D \circ M)(g) = D\big(M(g)\big) = M\big(D(g)\big) + g = (M \circ D + \mathrm{id})(g).$$

L'identité étant vérifiée par toute g, on obtient l'égalité $D \circ M = M \circ D + \mathrm{id}$ entre opérateurs linéaires différentiels. D'autres opérateurs vérifient des analogues de la règle de Leibniz : l'opérateur Δ de différence finie, donné par $\Delta(f)(x) = f(x+1) - f(x)$; l'opérateur $S = \Delta + \mathrm{id}$ de décalage, donné par $S(f)(x) = f(x+1)$; pour une constante q fixée autre que 0 et 1, l'opérateur H de dilatation, donné par $H(f)(x) = f(qx)$. On a les relations :

$$\Delta(fg)(x) = f(x+1)\Delta(g)(x) + \Delta(f)(x)g(x),$$
$$(fg)(x+1) = f(x+1)g(x+1), \qquad (fg)(qx) = f(qx)g(qx),$$

qui mènent aux relations $\Delta \circ M = (M + \mathrm{id}) \circ \Delta + \mathrm{id}$, $S \circ M = (M + \mathrm{id}) \circ S$, $H \circ M = Q \circ M \circ H$ entre opérateurs linéaires, après avoir introduit un nouvel opérateur Q donné par $Q(f)(x) = qf(x)$.

Le point de vue d'Ore est d'abstraire ces différents contextes d'opérateurs dans un même cadre algébrique.

Définition. *Soit A un anneau commutatif unitaire intègre de caractéristique zéro, que nous supposons muni d'un endomorphisme injectif σ et d'une σ-dérivation δ, au sens où pour tout a et tout b de A,*

$$\sigma(a+b)=\sigma(a)+\sigma(b),\qquad \sigma(ab)=\sigma(a)\sigma(b),\qquad \delta(ab)=\sigma(a)\delta(b)+\delta(a)b.$$

Pour une nouvelle variable ∂, on appelle anneau de polynômes tordus *l'algèbre sur A engendrée par ∂ et les relations, pour tout a de A,*

$$\partial a=\sigma(a)\partial+\delta(a).$$

On note cet anneau $A\langle\partial;\sigma,\delta\rangle$.

La terminologie « polynôme tordu » est la traduction de l'anglais « *skew polynomial* », où « *skew* » signifie « de biais », « oblique ». Certains auteurs ont proposé la traduction « polynôme gauche », où « gauche » a le sens de « voilé », par opposition à « plan ». Mais nous voulons éviter ici toute confusion avec des notions algébriques de multiple, module, fraction, etc, pour lesquelles « gauche » a le sens opposé de « droite ». De plus, la littérature note généralement les anneaux de polynômes tordus avec des crochets, par $A[\partial;\sigma,\delta]$, mais les chevrons nous semblent mieux évoquer que l'anneau en question est une algèbre sur A donnée par le générateur ∂ et des relations induites par σ et δ, et n'est en général pas commutatif.

Une conséquence immédiate de la définition est que tout élément f d'un anneau de polynômes tordus admet une réécriture canonique de la forme

$$f=a_r\partial^r+\cdots+a_0$$

pour des a_i dans A, avec a_r non nul sauf si la somme ne comprend aucun terme (et alors $f=0$). On montre que les a_i et l'entier r ainsi définis (sauf pour $f=0$) sont uniques et que r a les propriétés d'un degré. En particulier, le degré d'un produit est la somme des degrés de ses facteurs. Ceci résulte de l'injectivité de σ et de l'intégrité, par la formule

$$(a\partial^r+\dots)(b\partial^s+\dots)=a\sigma^r(b)\partial^{r+s}+\dots$$

car $a\sigma^r(b)$ est nul si et seulement si a ou b l'est.

Des choix adéquats de σ et δ nous font retrouver les quelques exemples donnés plus haut. Pour simplifier la notation, nous supposons que A peut s'identifier à un bon espace de fonctions. On a alors, en notant 0 l'application qui a toute fonction associe la fonction constante nulle :

- $\mathbb{Q}(x)\langle\partial;\mathrm{id},D\rangle$ représente l'algèbre des opérateurs différentiels linéaires ;
- $\mathbb{Q}(x)\langle\partial;S,0\rangle$ représente l'algèbre des opérateurs de récurrence ;
- $\mathbb{Q}(x)\langle\partial;S,\Delta\rangle$ représente l'algèbre des opérateurs de différence finie ;
- $\mathbb{Q}(x)\langle\partial;H,0\rangle$ pour $q\in\mathbb{Q}\setminus\{0,1\}$ représente l'algèbre des opérateurs de q-dilatation ;
- $\mathbb{Q}(x)\langle\partial;\mathrm{id},0\rangle$ n'est autre que l'anneau commutatif $\mathbb{Q}(x)[\partial]$ des polynômes usuels.

Toutes ces algèbres d'opérateurs sont à coefficients dans $\mathbb{Q}(x)$; on dispose aussi d'analogues pour $A=\mathbb{Q}[x]$ et, pour A ne faisant pas intervenir S, pour $A=\mathbb{Q}[x,x^{-1}]$.

On fera attention à la notation. Si la composition entre opérateurs est notée par $\circ$, nous ne ferons qu'une simple juxtaposition pour le produit de polynômes tordus, et nous noterons 1 l'élément neutre pour le produit de polynômes tordus. Néanmoins, on fera l'abus de notation de noter de la même façon, D_x, la dérivation par rapport à x quel que soit l'anneau A, et id, sans indice, pour l'identité de n'importe quel A. De plus, nous noterons simplement x pour M. Ainsi, on a :

— $\partial x = x\partial + 1$ dans $\mathbb{Q}(x)\langle\partial; \mathrm{id}, D\rangle$, car plus généralement, pour $u \in \mathbb{Q}(x)$,

$$\partial u = \sigma(u)\partial + \delta(u) = \mathrm{id}(u)\partial + D(u) = u\partial + u';$$

— $\partial x = (x+1)\partial$ dans $\mathbb{Q}(x)\langle\partial; S, 0\rangle$, car plus généralement, pour $u \in \mathbb{Q}(x)$,

$$\partial u = \sigma(u)\partial + \delta(u) = S(u)\partial + 0(u) = u(x+1)\partial;$$

— $\partial x = (x+1)\partial + 1$ dans $\mathbb{Q}(x)\langle\partial; S, \Delta\rangle$;

— $\partial x = qx\partial$ dans $\mathbb{Q}(x)\langle\partial; H, 0\rangle$;

— $\partial x = x\partial$ dans $\mathbb{Q}(x)\langle\partial; \mathrm{id}, 0\rangle = \mathbb{Q}(x)[\partial]$.

Le cas $\delta = 0$ est fréquent, et on écrit alors $A\langle\partial; \sigma\rangle$, sans référence au 0. De même que dans le cas commutatif on définit les polynômes de Laurent, dont l'algèbre est notée $A[X, X^{-1}]$, et dans laquelle $XX^{-1} = X^{-1}X = 1$, le cas où σ est inversible et autre que l'identité permet de représenter des opérateurs qui possèdent un inverse. Dans ce cas, on notera $A\langle\partial, \partial^{-1}; \sigma\rangle$ l'algèbre où $\partial a = \sigma(a)\partial$, $\partial^{-1}a = \sigma^{-1}(a)\partial^{-1}$, $\partial\partial^{-1} = \partial^{-1}\partial = 1$.

Pour finir de se détacher de la notation en termes d'opérateurs, on fait agir les anneaux de polynômes tordus sur les espaces de fonctions, au sens de l'action d'un anneau sur un module. Rappelons qu'un module M sur un anneau A est un ensemble non vide, muni d'une loi + en faisant un groupe additif, stable sous l'action d'une produit externe par les éléments de A, tel que l'action par produit externe par 1 soit l'identité, et vérifiant les formules $(PQ) \cdot f = P \cdot (Q \cdot f)$ et $(P + Q) \cdot f = (P \cdot f) + (Q \cdot f)$. Un anneau de polynômes tordus n'a pas d'action unique sur un espace de fonctions donné, aussi adoptons-nous quelques conventions.

Convention. Dans toute la suite du texte :

— un anneau de la forme $A\langle\partial; \sigma\rangle$ agit par $\partial \cdot f = \sigma(f)$ pour une extension convenable de σ,

— un anneau de la forme $A\langle\partial; \sigma, \delta\rangle$ agit par $\partial \cdot f = \delta(f)$ pour une extension convenable de δ,

— les coefficients dans A agissent par simple multiplication, $a \cdot f = af$.

Remarquons que l'algèbre des suites $A^{\mathbb{N}}$ ne peut pas être vue comme un module sur l'algèbre $A\langle\partial, \partial^{-1}; \sigma\rangle$ quand σ est le décalage avant. En effet, l'action de ∂ sur la suite valant 1 en 0 et nulle ensuite donne la suite nulle : l'action de ∂ ne peut donc pas être inversée. Ce problème technique peut être contourné en considérant les classes de suites identifiées lorsqu'elles sont égales après un certain rang (qui dépend des deux suites). (On parle de « germe de suite à l'infini ».) Mais on perd alors la possibilité d'exprimer certaines suites, celles à support fini, justement.

À l'aide de ces notations génériques, nous allons maintenant réexprimer des algorithmes déjà vus et petit à petit introduire de nouveaux calculs.

2. Clôtures par morphismes entre anneaux de polynômes tordus

Dans cette section, nous sommes amenés à considérer simultanément des fonctions de x et des fonctions d'une autre variable. Aussi indiquerons-nous en indice de D, S, ∂, σ, δ, etc, la variable à laquelle ces objets font référence. De plus, les anneaux A qui servent à construire les anneaux de polynômes tordus sont de la forme $\mathbb{Q}[x]$ ou $\mathbb{Q}[x, x^{-1}]$.

2.1. Récurrence sur les coefficients extraits d'une série différentiellement finie et série génératrice d'une suite polynomialement récurrente. On a déjà vu que lorsqu'une série $f = \sum_{n\geq 0} u_n x^n$ est différentiellement finie, ses coefficients vérifient une relation de récurrence finie. Autrement dit, la suite $u = (u_n)_{n\geq 0}$ est polynomialement récurrente. La preuve repose sur les identités

$$xf = \sum_{n\geq 1} u_{n-1}x^n = \sum_{n\geq 1} (\partial_n^{-1}\cdot u)(n)\, x^n$$

et

$$D_x(f) = f' = \sum_{n\geq 0}(n+1)u_{n+1}x^n = \sum_{n\geq 0}\big((n+1)\partial_n\cdot u\big)(n)\, x^n,$$

où nous avons introduit l'anneau $\mathbb{Q}[n]\langle \partial_n, \partial_n^{-1}; S_n\rangle$. Par récurrence, ceci donne

$$\begin{aligned} x^\alpha D_x^\beta(f) &= \sum_{n\geq \alpha}\Big(\partial_n^{-\alpha}\big((n+1)\partial_n\big)^\beta\cdot u\Big)(n)\, x^n \\ &= \sum_{n\geq \alpha}\big((n+1-\alpha)\cdots(n+\beta-\alpha)\partial_n^{\beta-\alpha}\cdot u\big)(n)\, x^n. \end{aligned}$$

Pour une série f solution de l'équation différentielle

$$a_r(x)f^{(r)}(x) + \cdots + a_0(x)f(x) = 0$$

où les a_i sont dans $\mathbb{Q}[x]$, nous obtenons ainsi une récurrence sur u, valable pour des n assez grands. Cette récurrence s'exprime en termes de polynômes tordus de la façon suivante. On représente l'opérateur différentiel associé à l'équation par le polynôme tordu $L = a_r(x)\partial_x^r + \cdots + a_0(x)$ dans $\mathbb{Q}[x]\langle \partial_x; \mathrm{id}, D_x\rangle$ sur $\mathbb{Q}$. De la sorte, l'équation différentielle s'écrit $L\cdot f = 0$. On introduit aussi l'algèbre $\mathbb{Q}[n]\langle \partial_n, \partial_n^{-1}; S_n\rangle$ et le morphisme d'algèbres μ défini par $\mu(x) = \partial_n^{-1}$ et $\mu(\partial_x) = (n+1)\partial_n$. Alors, la suite u des coefficients est solution pour n assez grand de la récurrence représentée par l'image $\mu(L)$. Pour comprendre pour quels n cette récurrence est valide, écrivons

$$\mu(L) = b_p(n)\partial_n^p + \cdots + b_q(n)\partial_n^q$$

pour $p \leq q$ et $b_p b_q \neq 0$, de sorte que la récurrence prend la forme

$$\big(\mu(L)\cdot u\big)(n) = b_p(n)u_{n+p} + \cdots + b_q(n)u_{n+q} = 0$$

et est vérifiée pour tout n si $p \geq 0$ et pour tout $n \geq -p$ si $p < 0$.

De façon duale, une suite polynomialement récurrente u a une série génératrice $f = \sum_{n\geq 0} u_n x^n$ différentiellement finie, ce que nous allons retrouver en termes de polynômes tordus. Pour ce point, nous supposons en fait que la suite u est prolongée aux indices négatifs par $u_n = 0$ pour $n < 0$, et qu'elle est P-récurrente sur $\mathbb{Z}$ tout entier. Ceci ne constitue aucune perte de généralité : une récurrence valable pour la suite initiale devient valable pour la suite prolongée après multiplication par un polynôme de la forme $(n+1)(n+2)\ldots(n+r)$. Les formules

$$\sum_{n\in\mathbb{Z}} nu_n x^n = x\partial_x\cdot f \qquad \text{et} \qquad \sum_{n\in\mathbb{Z}} u_{n+1}x^n = x^{-1}f$$

donnent par récurrence

$$\sum_{n\geq 0} n^\alpha u_{n+\beta}x^n = (x\partial_x)^\alpha x^{-\beta}\cdot f = x^{-\beta}(x\partial_x - \beta)^\alpha\cdot f,$$

et fournissent un autre morphisme, ν, de $\mathbb{Q}[n]\langle \partial_n; S_n\rangle$ dans $\mathbb{Q}[x, x^{-1}]\langle \partial_x; \mathrm{id}, D_x\rangle$, donné par $\nu(n) = x\partial_x$ et par $\nu(\partial_n) = x^{-1}$. Pour une suite u solution sur $\mathbb{Z}$ de

l'équation de récurrence

$$b_p(n)u_{n+p} + \cdots + b_0(n)u_n = 0$$

où les b_i sont dans $\mathbb{Q}[n]$, nous introduisons le polynôme tordu $P = b_p(n)\partial_n^p + \cdots + b_0(n)$ de $\mathbb{Q}[n]\langle\partial_n; S_n\rangle$. Pour obtenir une relation différentielle sur la série génératrice f, nous considérons $\nu(P)$ que nous écrivons

$$\nu(P) = a_0(x) + \cdots + a_r(x)\partial_x^r.$$

Alors, comme $\sum_{n\in\mathbb{Z}}(P\cdot u)(n)x^n = 0$, la série f vérifie la relation différentielle

$$a_0(x)f(x) + \cdots + a_r(x)f^{(r)}(x) = 0.$$

Algébriquement, les propriétés précédentes s'expriment par le fait que μ s'étend en un isomorphisme d'algèbres entre $\mathbb{Q}[x, x^{-1}]\langle\partial_x; \mathrm{id}, D_x\rangle$ et $\mathbb{Q}[n]\langle\partial_n, \partial_n^{-1}; S_n\rangle$, dont l'inverse étend ν. Ce morphisme vérifie $\mu(x^i) = \partial_n^{-i}$ et $\mu(\partial_x^i) = \big((n+1)\partial_n\big)^i = (n+1)\dots(n+i)\partial_n^i$. Les deux algèbres isomorphes peuvent être vues comme agissant naturellement sur $\mathbb{Q}((x)) = \{\, x^{-r}f \mid r \in \mathbb{N},\ f \in \mathbb{Q}[[x]]\,\}$, qui n'est autre que l'algèbre des suites sur $\mathbb{Z}$ nulles pour les indices n strictement inférieurs à un entier $-r$ (dépendant de la suite considérée).

2.2. Séries binomiales.

Exercice 1. Une série binomiale est une série de la forme $\sum_{n\geq 0} u_n\binom{x}{n}$. Montrer que les solutions en série binomiale d'une équation fonctionnelle à différence

$$a_r(x)f(x+r) + \cdots + a_0(x)f(x) = 0$$

ont des coefficients u_n qui vérifient une récurrence et expliciter le morphisme entre algèbres de polynômes tordus correspondant.

2.3. Changements de variables. Lorsqu'une série D-finie $f(x)$ est solution d'une équation différentielle $L\cdot f = 0$ donnée par un polynôme tordu

$$L = L(x, \partial_x) = a_r(x)\partial_x^r + \cdots + a_0(x),$$

la série $g(x) := f(\lambda x)$ vérifie $g^{(i)}(x) = \lambda^i f^{(i)}(\lambda x)$ pour chaque $i \in \mathbb{N}$. En substituant λx pour x dans $L\cdot f = 0$, on obtient que g est solution de l'équation différentielle associée à

$$L(\lambda x, \lambda^{-1}\partial_x) = a_r(\lambda x)\lambda^{-r}\partial_x^r + \cdots + a_0(\lambda x).$$

C'est encore le résultat d'un morphisme d'algèbres, Λ, défini cette fois par $\Lambda(x) = \lambda x$ et $\Lambda(\partial_x) = \lambda^{-1}\partial_x$.

Lorsque f est une fonction D-finie, la fonction $z \mapsto f(1/z)$ est elle aussi D-finie, en z cette fois, pour autant que la fonction composée ait un sens. En effet, pour toute fonction g, notons $\tilde{g}(z) = g(1/z)$ (avec la même réserve de définition). Puisque $g(x) = \tilde{g}(1/x)$, par dérivation on a $g'(x) = -\tilde{g}'(1/x)/x^2$, ce qui est l'évaluation en $z = 1/x$ de $-z^2\partial_z\cdot\tilde{g}$. Autrement dit, on a $\widetilde{g'} = -z^2\partial_z\cdot\tilde{g}$, d'où par récurrence $\widetilde{g^{(\beta)}} = (-z^2\partial_z)^\beta\cdot\tilde{g}$. Ainsi, $\tilde{f}$ est D-finie, donnée comme vérifiant l'équation différentielle associée à l'image de L par le morphisme de $\mathbb{Q}[x]\langle\partial_x; \mathrm{id}, D_x\rangle$ dans $\mathbb{Q}[z, z^{-1}]\langle\partial_z; \mathrm{id}, D_z\rangle$ qui envoie x sur z^{-1} et ∂_x sur $-z^2\partial_z$.

Exercice 2. Plus généralement, la fonction obtenue par substitution rationnelle de la variable, donnée par $h(u) = f\big(r(u)\big)$, est encore D-finie. Nous laissons en exercice le soin de montrer ce résultat par la même approche dans le cas où la dérivée r' s'exprime comme une fraction rationnelle en r.

3. Division euclidienne

Dans cette section et les suivantes, nous nous appuyons sur des propriétés particulières des anneaux de polynômes tordus quand l'anneau A de la construction est un corps, que nous prendrons de la forme $\mathbb{Q}(x)$.

La commutation $\partial a = \sigma(a)\partial + \delta(a)$ dans $\mathbb{Q}(x)\langle\partial;\sigma,\delta\rangle$ permet d'écrire tout polynôme tordu sous la forme $a_0(x)+\cdots+a_r(x)\partial^r$, pour des fractions rationnelles a_i de $\mathbb{Q}(x)$ uniques. Une conséquence de l'injectivité de σ est l'existence d'un degré en ∂ bien défini : l'entier r de l'écriture précédente lorsque a_r est non nulle. En particulier, le degré d'un produit L_1L_2 de polynômes tordus est la somme des degrés des L_i. Il s'ensuit que la division euclidienne du cas commutatif, et toute la théorie qui en découle, se transpose avec peu d'altérations dans le cas tordu.

Procédé de division. La différence principale avec le cas commutatif est qu'on distingue division euclidienne à gauche et division euclidienne à droite. Vu notre interprétation en termes d'opérateurs linéaires, nous ne considérerons que la division à droite, qui se fait en retranchant des multiples à gauche. Soit à diviser $A = a_r(x)\partial^r + \cdots + a_0(x)$ de degré r par $B = b_s(x)\partial^s + \cdots + b_0(x)$ de degré s. On suppose $s \leq r$. Alors,

$$\partial^{r-s}B = \sigma^{r-s}\big(b_s(x)\big)\partial^r + \text{termes d'ordres inférieurs},$$

où la puissance de σ représente une itération (par composition), et ainsi

$$A - a_r(x)\sigma^{r-s}\big(b_s(x)\big)^{-1}\partial^{r-s}B$$

est de degré strictement inférieur à r. Cette étape de réduction est l'étape élémentaire de la division euclidienne. En itérant le procédé, on aboutit à un reste R de degré strictement inférieur à s. En regroupant les facteurs gauches, on obtient un quotient à gauche Q tel que $A = QB + R$.

Exemple 1. On considère l'anneau $\mathbb{Q}(n)\langle\partial_n; S_n\rangle$ des polynômes tordus représentant les opérateurs de décalage. La division de $A = (n^2-1)\partial_n^2 - (n^3+3n^2+n-2)\partial_n + (n^3+3n^2+2n)$, qui annule les combinaisons linéaires de $n!$ et n, par $B = n\partial_n^2 - (n^2+3n+1)\partial_n + (n^2+2n+1)$, qui annule les combinaisons linéaires de $n!$ et 1, s'écrit

$$A = n^{-1}(n^2-1)B - n^{-1}(n^2+n+1)\big(\partial_n - (n+1)\big).$$

Le reste est multiple de $\partial_n - (n+1)$, qui représente la récurrence $u_{n+1} = (n+1)u_n$, vérifiée par la factorielle.

Notons une propriété de cette division : si A est multiplié à gauche par un facteur $m(x)$ sans que B ne soit changé, alors Q et R sont multipliés à gauche par le même facteur $m(x)$. Ceci ne vaut plus (en général) pour un facteur faisant intervenir ∂.

Réduction d'une grande puissance de ∂. La division euclidienne nous donne une nouvelle interprétation du calcul du N^{e} terme d'une suite P-récurrente $u = (u_n)$ relativement à $\mathbb{Q}(n)\langle\partial_n; S_n\rangle$. Supposons que u soit solution de l'équation de récurrence

$$a_r(n)u_{n+r} + \cdots + a_0(n)u_n = 0.$$

En déroulant la récurrence, on voit que u_N peut, pour tout N sauf annulation malvenue de a_r, se mettre sous la forme $\alpha_{r-1,N}u_{r-1}+\cdots+\alpha_{0,N}u_0$. Plus généralement, on a une relation qui récrit u_{n+N} en terme de u_{n+r-1}, …, u_n. Pour l'obtenir, associons à la récurrence sur u le polynôme tordu $P = a_r(n)\partial_n^r + \cdots + a_0(n)$. Pour

un N donné, la division euclidienne de ∂_n^N par P s'écrit

$$\partial_n^N = Q_N(n)P + \alpha_{r-1,N}(n)\partial_n^{r-1} + \cdots + \alpha_{0,N}(n)$$

pour des fractions rationnelles $\alpha_{i,N}(n)$. Après application sur u et évaluation en n, nous obtenons

$$u_{n+N} = 0 + \alpha_{r-1,N}(n)u_{n+r-1} + \cdots + \alpha_{0,N}(n)u_n,$$

d'où le résultat annoncé pour $\alpha_{i,N} = \alpha_{i,N}(0)$.

Exercice 3. Nous laissons le lecteur se convaincre que la réécriture d'une dérivée $f^{(N)}$ d'une fonction D-finie f décrite par une équation différentielle d'ordre r en terme de ses dérivées d'ordre strictement inférieur à r s'interprète de façon analogue comme le calcul d'un reste de division euclidienne.

Clôture par addition. Le même ingrédient se retrouve dans l'algorithme donnant la clôture par addition de deux fonctions D-finies ou de deux suites P-récurrentes : pour deux objets f et g à additionner, décrits comme solutions des équations respectives $L_f \cdot f = 0$ et $L_g \cdot g = 0$ pour des polynômes tordus de degrés respectifs r et s dans un anneau adéquat $A\langle\partial;\sigma,\delta\rangle$, l'algorithme exprime, pour des i successifs, $\partial^i \cdot (f+g)$ sous la forme $(\partial^i \bmod L_f)\cdot f + (\partial^i \bmod L_g)\cdot g$, où la notation $A \bmod B$ note le reste de la division euclidienne à droite de A par B. Lorsque suffisamment de i ont été considérés, l'algorithme qui a été donné au Chapitre 14 calcule par de l'algèbre linéaire des cofacteurs $a_0, \ldots, a_{r+s}$ tels que, simultanément,

$$\sum_{i=0}^{r+s} a_i(\partial^i \bmod L_f) = 0 \qquad \text{et} \qquad \sum_{i=0}^{r+s} a_i(\partial^i \bmod L_g) = 0.$$

Notons que $P = \sum_{i=0}^{r+s} a_i\partial^i$ est un multiple commun à gauche de L_f et de L_g, puisque $P \bmod L_f = P \bmod L_g = 0$. Puisque l'algorithme fonctionne en recherchant un P d'ordre minimal, l'algorithme de clôture par addition est donc un algorithme de calcul de p. p. c. m. (à gauche).

4. Recherche de solutions et factorisation d'opérateurs

Comme pour les anneaux de polynômes commutatifs usuels, une notion de factorisation est présente pour les anneaux de polynômes tordus. Une nuance importante réside dans le lien entre les « zéros » des polynômes tordus et la position des facteurs. Nous allons voir que la factorisation de polynômes tordus se relie aux algorithmes vus en cours pour la recherche de solutions polynomiales, rationnelles, et hypergéométriques dans le cas de récurrences. Par ailleurs, il n'y a plus d'unicité des facteurs, même à ordre près et modulo multiplication par des unités. Un exemple est donné dans le cas différentiel par l'infinité de factorisations

$$\partial^2 = \left(\partial + \frac{1}{x+r}\right)\left(\partial - \frac{1}{x+r}\right)$$

quand r est une constante.

Dans le cas d'un polynôme commutatif w se factorisant sous la forme uv pour des facteurs polynomiaux de degré au moins 1, tout zéro de u et tout zéro de v est zéro de w ; à l'inverse, quitte à se placer dans une clôture algébrique, tout zéro α de w en fournit un facteur $x-\alpha$ et un quotient exact $u(x)$ tel que $w(x) = u(x)(x-\alpha)$. Le phénomène est différent dans le cas tordu, comme attesté par l'inégalité

$$\partial^2 \neq \left(\partial - \frac{1}{x+r}\right)\left(\partial + \frac{1}{x+r}\right)$$

quand r est une constante. Plus généralement, une factorisation $L = PQ$ dans $A\langle\partial;\sigma,\delta\rangle$ (où A est un corps) a des propriétés différentes selon le facteur : une solution f de l'équation $Q\cdot f = 0$ est encore solution de $L\cdot f = 0$, car $L\cdot f = P\cdot(Q\cdot f) = P\cdot 0 = 0$; mais une solution g de P ne donne lieu à des solutions f de L que par la relation $Q\cdot f = g$. Inversement, une solution f de L donne lieu à un facteur droit d'ordre 1 de L, à condition d'étendre A par f et tous ses itérés par σ et δ. Ce facteur est de la forme $\partial - (\partial\cdot f)/f$, c'est-à-dire $\partial - \delta(f)/f$ ou $\partial - \sigma(f)/f$ selon l'action de l'anneau de polynômes tordus sur les fonctions.

Les algorithmes de recherche de solutions dans des classes particulières fournissent donc implicitement, pour chaque solution trouvée, un facteur droit d'ordre 1. Ils interviennent donc naturellement comme base des algorithmes de factorisation [2, 3]. Plus précisément, dans le cas différentiel, une solution polynomiale ou rationnelle f d'une équation $L\cdot f = 0$ pour L dans l'anneau $\mathbb{Q}(x)\langle\partial_x;\mathrm{id},D_x\rangle$ fournit un facteur droit $D = \partial_x - (D_x\cdot f)/f$ où $(D_x\cdot f)/f$ est rationnel ; dans le cas à récurrence, une solution polynomiale, rationnelle ou hypergéométrique f d'une équation $L\cdot f = 0$ pour L dans l'anneau $\mathbb{Q}(x)\langle\partial_x;S_x\rangle$ fournit un facteur droit $D = \partial_x - S_x(f)/f$ où $S_x(f)/f$ est rationnel. Dans les deux cas, le quotient Q tel que $L = QD$ est aussi à coefficients rationnels.

Exemple 2. Un exemple simple, en réalité sans extension stricte de A, est donné dans $A = \mathbb{Q}[x]\langle\partial_x;\mathrm{id},D_x\rangle$ par l'opérateur

$$L = x^2(4x^2+5)\partial_x^3 - x(8x^4+14x^2-5)\partial_x^2 - (4x^4+17x^2+5)\partial_x - 2x^3(4x^2+9),$$

dont une solution particulière est $f = \exp x^2$. Cette solution se vérifie à partir de $D_x\cdot f = 2xf$, $D_x^2\cdot f = (4x^2+2)f$ et $D_x^3\cdot f = (8x^3+12x)f$, en établissant la nullité de

$$x^2(4x^2+5)(8x^3+12x) - x(8x^4+14x^2-5)(4x^2+2) - (4x^4+17x^2+5)(2x) - 2x^3(4x^2+9).$$

Par division euclidienne, on trouve

$$L = \big(x^2(4x^2+5)\partial_x^2 - x(4x^2-5)\partial_x + 4x^4 + 13x^2 - 5\big)\,(\partial_x - 2x).$$

On montre que L n'a pas d'autre facteur droit non-trivial unitaire ; l'exercice 4 fournit les facteurs gauches d'ordre 1 :

$$L = \big(x(4x^2+5)\partial_x - (2x^2+5)(4x^2+1)\big)(x\partial_x^2 + \partial_x + x).$$

Exemple 3. L'opérateur $L = 2(2x+3)\partial_x^2 + 3(5x+7)\partial_x + 9(x+1)$ fournit un autre exemple, pour $A = \mathbb{Q}[x]\langle\partial_x;S_x\rangle$. Comme une solution particulière est $(-3)^x$, ce qui s'obtient par l'algorithme de Petkovšek et peut s'établir simplement en observant la nullité de L en $\partial_x = -3$, un facteur droit de L est $\partial_x + 3$, et une division euclidienne montre

$$L = \big(2(2x+3)\partial_x + 3(x+1)\big)(\partial_x + 3).$$

On montre que L n'a pas d'autre facteur droit non-trivial unitaire.

Exercice 4. Un antimorphisme ϕ entre anneaux est une application $\mathbb{Q}$-linéaire qui renverse les produits : $\phi(PQ) = \phi(Q)\phi(P)$. Montrer l'existence d'antimorphismes $\mu : \mathbb{Q}(x)\langle\partial_x;\mathrm{id},D_x\rangle \to \mathbb{Q}(u)\langle\partial_u;\mathrm{id},D_u\rangle$ et $\nu : \mathbb{Q}(x)\langle\partial_x;S_x\rangle \to \mathbb{Q}(u)\langle\partial_u;S_u\rangle$, définis par les relations

$$\mu(x) = u,\quad \mu(\partial_x) = -\partial_u, \qquad \text{et} \qquad \nu(x) = -u,\quad \nu(\partial_x) = \partial_u.$$

Expliquer comment ces antimorphismes fournissent des facteurs gauches d'ordre 1 de polynômes tordus.

5. Algorithme d'Euclide

Rappelons qu'un idéal d'un anneau commutatif A est un sous-groupe additif de A clos par multiplication par les éléments de A. Il est classique que les anneaux commutatifs euclidiens — ceux dans lesquels l'existence d'un degré permet une division euclidienne — sont principaux : tout idéal peut être engendré par un unique générateur. C'est le cas des anneaux de polynômes commutatifs à coefficients dans un corps. Le p. g. c. d. p de deux polynômes f et g est alors l'unique polynôme unitaire engendrant l'idéal (f, g), somme des idéaux (f) et (g). Il se calcule comme dernier reste non nul par l'algorithme d'Euclide.

Pour un anneau de polynômes tordus $A = K\langle\partial; \sigma, \delta\rangle$ sur un corps K, la situation est la même si on prend soin de ne considérer que des idéaux à gauche, c'est-à-dire avec la clôture par multiplication à gauche par les éléments de A. Les notions qui en découlent sont celles de divisions euclidiennes à droite et de plus grands communs diviseurs à droite (p. g. c. d. d.). Soient P_0 et P_1 deux polynômes de A. Si P_1 n'est pas nul, on écrit la division euclidienne de P_0 par P_1, sous la forme $P_0 = Q_0P_1 + P_2$. Tant que P_{i+2} n'est pas nul, on itère en divisant P_{i+1} par P_{i+2}. Soit j la valeur finale de i, telle que $P_{j+1} \neq 0$ et $P_{j+2} = 0$. Alors :

$$\begin{bmatrix} P_0 \\ P_1 \end{bmatrix} = \begin{bmatrix} Q_0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = \cdots = \begin{bmatrix} Q_0 & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} Q_j & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} P_{j+1} \\ 0 \end{bmatrix},$$

d'où on déduit que P_{j+1} divise P_0 et P_1 à droite : $P_0 = FP_{j+1}$ et $P_1 = GP_{j+1}$ pour des polynômes tordus F et G adéquats. Puis en inversant les matrices

$$\begin{bmatrix} P_{j+1} \\ 0 \end{bmatrix} = \begin{bmatrix} U & V \\ R & S \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \end{bmatrix} \qquad \text{pour} \qquad \begin{bmatrix} U & V \\ R & S \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -Q_j \end{bmatrix} \cdots \begin{bmatrix} 0 & 1 \\ 1 & -Q_0 \end{bmatrix}.$$

En particulier, $P_{j+1} = UP_0 + VP_1$ est élément de l'idéal à gauche $AP_0 + AP_1$. Un élément quelconque $L = MP_0 + NP_1$ de cet idéal est aussi multiple de P_{j+1} : $L = (MF+NG)P_{j+1}$. En normalisant P_{j+1} pour le rendre unitaire, on obtient donc un p. g. c. d. d. distingué de P_0 et P_1. Par ailleurs, le polynôme tordu $RP_0 = -SP_1$ est un plus petit multiple commun à gauche (p. p. c. m. g.) de P_0 et P_1, par le même argument que dans le cas commutatif, en suivant de près les degrés tout au long de l'algorithme.

On a vu que les algorithmes de clôture par addition entre fonctions D-finies ou entre suites P-récurrentes renvoient un multiple commun à gauche des polynômes tordus L_f et L_g décrivant les deux objets f et g à additionner. Comme ces algorithmes opèrent par degrés croissants, le polynôme renvoyé est de degré minimal en ∂, parmi ceux qui annulent la somme $f + g$. Le polynôme annulateur de la somme $f + g$ renvoyé par ces algorithmes est donc le p. p. c. m. g. de L_f et de L_g.

Exemple 4. Nous repartons des polynômes A et B de l'exemple 1 pour en calculer un p. g. c. d. d. et un p. p. c. m. g. On pose $P_0 = A$, $P_1 = B$; on a déjà calculé $P_2 = -n^{-1}(n^2+n+1)\big(\partial_n - (n+1)\big)$ avec $P_0 = n^{-1}(n^2-1)P_1 + P_2$. Il vient ensuite

$$P_1 = \left(-\frac{n(n+1)}{n^2+3n+3}\partial_n + \frac{n(n+1)}{n^2+n+1}\right) P_2 + 0.$$

Ainsi, le p. g. c. d. d. unitaire est $\partial_n - (n+1)$. Remarquons qu'il annule les solutions communes de A et B, à savoir les multiples de $n!$. Le p. p. c. m. g. unitaire s'obtient par renormalisation de $Q_1P_0 = (Q_1Q_0 + 1)P_1$:

$$\partial_n^3 - \frac{n^3+6n^2+8n+5}{n^2+n+1}\partial_n^2 + \frac{2n^3+9n^2+13n+7}{n^2+n+1}\partial_n - \frac{(n^2+3n+3)(n+1)}{n^2+n+1}.$$

Notons que ses solutions sont toutes les solutions de A et B : les combinaisons linéaires de $n!$, n et 1.

6. Relations de contiguïté

Un grand nombre de fonctions spéciales sont en fait des fonctions $f_n(x)$ d'une variable continue x et d'une variable discrète n. Les familles de telles fonctions dont l'intérêt a été révélé, par exemple par la physique mathématique, sont telles que la dépendance en x est liée à la dépendance en n. Il apparaît que très fréquemment, la fonction $f_{n+\alpha}(x)$, pour $\alpha = \pm 1$, est reliée à la fonction $f_n(x)$ et à ses dérivées. C'est le cas pour la classe importante des *fonctions hypergéométriques*, c'est-à-dire, essentiellement, pour les séries génératrices de suites hypergéométriques, et pour des fonctions limites de fonctions hypergéométriques, dont un certain nombre de familles de polynômes orthogonaux classiques, et pour des généralisations.

Dans cette section, nous considérons des fonctions D-finies paramétrées et résolvons algorithmiquement des problèmes tels que la détermination d'une relation de la forme

$$f_{n+1}(x) = \sum_{i=0}^{r-1} a_i(x) f_n^{(i)}(x)$$

pour la suite de polynômes

$$f_n(x) = \sum_{k=0}^{n} (-1)^k \binom{n}{k}^2 \binom{n+k}{k}^2 x^k.$$

Ici, la relation explicite est

$$\begin{aligned} f_{n+1}(x) = {} & 12\frac{x^3(x+1)}{(n+1)^3} f_n'''(x) + 4\frac{x^2(2n+2xn+11+14x)}{(n+1)^3} f_n''(x) \\ & - 4\frac{x(5xn^2-n^2-4n-6+2xn-9x)}{(n+1)^3} f_n'(x) - \frac{16xn-n-1+4x}{n+1} f_n(x). \end{aligned}$$

L'opérateur différentiel linéaire implicitement au membre droit s'appelle un *opérateur de montée*; un opérateur qui donnerait $f_{n-1}(x)$ s'appelle un *opérateur de descente*. Une relation linéaire entre les decalées $f_{n+i}(x)$ et ne faisant intervenir aucune dérivation s'appelle une *relation de contiguïté*.

6.1. Fonction hypergéométrique de Gauss. La plus simple des fonctions hypergéométriques est la fonction hypergéométrique de Gauss, définie par

$$F(a,b;c;x) = {}_2F_1\left(\left.\begin{matrix} a, b \\ c \end{matrix}\right| x\right) = \sum_{k\geq 0} \frac{(a)_k (b)_k}{(c)_k} \frac{x^k}{k!}.$$

La notation ${}_2F_1$ a déjà été introduite au Chapitre 27, ainsi que le symbole de Pochhammer

$$(s)_n = s(s+1)\cdots(s+n-1) = \frac{\Gamma(s+n)}{\Gamma(s)},$$

lequel vérifie

$$\frac{(s)_{n+1}}{(s)_n} = s+n \qquad \text{et} \qquad \frac{(s+1)_n}{(s)_n} = \frac{s+n}{s}.$$

Oublions la dépendance en b et c du coefficient de x^k dans cette somme, coefficient que nous notons $u_{a,k}$. Nous avons

$$u_{a,k+1} = \frac{(a+k)(b+k)}{(c+k)(k+1)} u_{a,k} \qquad \text{et} \qquad u_{a+1,k} = \left(\frac{k}{a}+1\right) u_{a,k}.$$

La première de ces identités nous fournit le polynôme tordu

$$(c+k)(k+1)\partial_k - (a+k)(b+k) \in \mathbb{Q}(a,b,c)[k]\langle \partial_k; S_k\rangle$$

qui annule u. Pour $k=-1$, cette récurrence impose $u_{a,-1}=0$, ce qui permet d'étendre la suite u à toute valeur $k\in\mathbb{Z}$ tout en continuant de vérifier la récurrence. Par le morphisme qui effectue le passage à la série génératrice, nous obtenons

$$\begin{aligned}(c+x\partial_x)(x\partial_x+1)x^{-1}-(a+x\partial_x)(b+x\partial_x)&\\ =\big((x\partial_x)^2+(c+1)x\partial_x+c\big)x^{-1}-\big((x\partial_x)^2+(a+b)x\partial_x+ab\big)&\\ =x(1-x)\partial_x^2+\big(c-(a+b+1)x\big)\partial_x-ab&\\ \in\mathbb{Q}(a,b,c)[x,x^{-1}]\langle\partial_x;\mathrm{id},D_x\rangle.&\end{aligned}$$

Notons L ce polynôme tordu en ∂_x. La deuxième identité sur u donne, après sommation, $F(a+1,b;c;x)=(a^{-1}x\partial_x+1)\cdot F(a,b;c;x)$; c'est-à-dire qu'un opérateur de montée est donné par le polynôme tordu $L_\uparrow=a^{-1}x\partial_x+1$.

Définissons $G(a,b;c;x)$ comme étant $F(a+1,b;c;x)$. Supposons qu'il existe un inverse V de $L_\uparrow$ modulo L à droite. Alors $VL_\uparrow-1$ est un multiple à gauche de L, qui annule donc F. Ainsi, $F=VL_\uparrow\cdot F=V\cdot G$, autrement dit, V représente un opérateur de descente de G. On obtient un opérateur de descente pour F par un simple décalage arrière de a dans V. Pour un calcul effectif, la division euclidienne

$$L=Q(a^{-1}x\partial_x+1)-(c-a-1)ax^{-1}\qquad\text{où}\qquad Q=a(1-x)\partial_x+(c-a-1)ax^{-1}-ab$$

donne l'opérateur de descente après avoir décalé a dans $(c-a-1)^{-1}a^{-1}xQ$ par

$$L_\downarrow=\frac{x(1-x)}{a-c}\partial_x-\frac{bx}{a-c}-1.$$

Notre objectif est maintenant de calculer une relation de contiguïté pour F. Nous avons obtenu $L_\uparrow(a)\cdot F=\partial_a\cdot F$, où nous avons noté explicitement la dépendance en a du polynôme tordu $L_\uparrow$. Il s'ensuit la relation

$$\partial_a^i\cdot F=L_{\uparrow,i}(a)\cdot F\qquad\text{où}\qquad L_{\uparrow,i}(a)=L_\uparrow(a+i-1)\cdots L_\uparrow(a+1)L_\uparrow(a),$$

dans laquelle nous pouvons, comme toujours, remplacer un polynôme agissant sur la fonction F par le reste de la division euclidienne de ce polynôme par L, qui annule F. Ainsi, une relation de contiguïté s'obtient en recherchant une combinaison linéaire, à coefficients dans $\mathbb{Q}(a,b,c,x)$ des restes modulo L à droite des $L_{\uparrow,i}(a)$ pour $i=0,1,2$.

Exercice 5. Terminer ce calcul pour retrouver :

$$\begin{aligned}(a+1)(1-x)F(a+2,b;c;x)+\big(c-xb+(a+1)(x-2)\big)F(a+1,b;c;x)&\\ +(a-c+1)F(a,b;c;x)=0.&\end{aligned}$$

6.2. Extension aux séries partiellement hypergéométriques. Nous considérons maintenant des sommes

$$f_n(x)=\sum_{k\geq 0}u_{n,k}x^k$$

dans lesquelles u n'est hypergéométrique qu'en n, mais est seulement P-récurrente en k, et satisfait à la relation

$$a_p(n,k)u_{n,k+p}+\cdots+a_0(n,k)u_{n,k}=0.$$

Comme dans le cas doublement hypergéométrique, cette relation fournit une relation purement différentielle sur f. Comme précédemment, aussi, la relation de récurrence du premier ordre en n sur u donne une expression de $f_{n+1}(x)$ comme combinaison linéaire de dérivées. On procède donc comme dans la section précédente pour calculer opérateurs de montée, de descente, et relations de contiguïté.

Exercice 6 (Assez calculatoire). Calculer l'opérateur de montée annoncé dans l'introduction de cette section.

Notes

La structure d'anneau de polynômes tordus a été introduite et étudiée par Ore dans [8], avec le dessein explicite de traiter (et résoudre ?) des systèmes fonctionnels linéaires [7]. Dans ces travaux, les polynômes tordus sont simplement appelés « non commutatifs ». Dans la littérature moderne, on parle souvent de « polynômes » ou d'« opérateurs de Ore », et encore d'« opérateurs pseudo-linéaires » [4].

L'interprétation de la Section 2 du lien par morphisme entre série et suite de coefficients est implicite dans les travaux sur la théorie des D-modules (voir [5] pour une exposition simple). Elle a été utilisée de façon explicite (pour le cadre binomial) dans [1].

La question du calcul algorithmique de relations de contiguïté a été abordée d'abord dans le cas purement hypergéométrique dans [12], puis dans le cas partiellement hypergéométrique dans [6].

Les questions de factorisation d'opérateurs linéaires et de résolution mènent à une théorie de Galois : la théorie de Galois différentielle [10, 11] comme celle à différence [9] étudient les applications linéaires qui échangent les solutions d'une équations fonctionnelle linéaire. Un succès de la théorie différentielle est l'étude des solutions dites « liouviliennes » d'une équation linéaire différentielle, données explicitement en termes de compositions d'opérations élémentaires (exponentiations, prises de logarithmes, extensions algébriques). La théorie fournit aussi des informations sur les développements asymptotiques des solutions, notamment en présence de singularités faisant intervenir des exponentielles.

Bibliographie

[1] Bostan, A., F. Chyzak, T. Cluzeau et B. Salvy (2006). « Low complexity algorithms for linear recurrences ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 31–38.

[2] Bronstein, M. et M. Petkovšek (1994). « On Ore rings, linear operators and factorisation ». In : *Programmirovanie*, vol. 1. Also available as Research Report 200, Informatik, ETH Zürich, p. 27–44.

[3] Bronstein, Manuel (1992). « Linear Ordinary Differential Equations : breaking through the order 2 barrier ». In : *ISSAC'92 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 42–48.

[4] Bronstein, Manuel et Marko Petkovšek (1996). « An introduction to pseudo-linear algebra ». In : *Theoretical Computer Science*, vol. 157, p. 3–33.

[5] Cartier, Pierre (1992). « Démonstration "automatique" d'identités et fonctions hypergéométriques (d'après D. Zeilberger) ». In : *Astérisque*, vol. 1991/92, n°206. Séminaire Bourbaki, Exp. No. 746, 3, 41–91.

[6] Chyzak, Frédéric et Bruno Salvy (1998). « Non-commutative Elimination in Ore Algebras Proves Multivariate Holonomic Identities ». In : *Journal of Symbolic Computation*, vol. 26, n°2, p. 187–227. doi : 10.1006/jsco.1998.0207.

[7] Ore, Oystein (1931). « Linear equations in non-commutative fields ». In : *Annals of Mathematics*, vol. 32, p. 463–477.

[8] — (1933). « Theory of non-commutative polynomials ». In : *Annals of Mathematics*, vol. 34, n°3, p. 480–508.

[9] Put, Marius van der et Michael F. Singer (1997). *Galois theory of difference equations*. Vol. 1666. Lecture Notes in Mathematics. Springer-Verlag, viii+180 pages.

[10] — (2003). *Galois theory of linear differential equations*. Vol. 328. Grundlehren der Mathematischen Wissenschaften. Springer-Verlag, xviii+438 pages.

[11] Singer, M. F. (2007). « Introduction to the Galois Theory of Linear Differential Equations ». In :

[12] Takayama, Nobuki (1989). « Gröbner basis and the problem of contiguous relations ». In : *Japan Journal of Applied Mathematics*, vol. 6, n°1, p. 147–160.

CHAPITRE 29

Algorithmes pour les fonctions spéciales dans les algèbres de Ore

1. Algèbres de Ore rationnelles

Une généralisation à plusieurs dérivations et décalages de la notion d'anneau de polynômes tordus du chapitre précédent est donnée par la définition qui suit.

DÉFINITION (Algèbre de Ore). *Étant donnés*

- *un corps (commutatif) $k(x) = k(x_1, \ldots, x_r)$ de fractions rationnelles,*
- *r endomorphismes σ_i de k-algèbre de $k(x)$, injectifs et commutant deux à deux,*
- *pour chaque i une σ-dérivation δ_i relative à σ_i, c'est-à-dire pour chaque i un endomorphisme linéaire pour lequel $\delta_i(ab) = \sigma_i(a)\delta_i(b) + \delta_i(a)b$ dès que a et b sont dans $k(x)$, toutes ces σ-dérivations commutant deux à deux et δ_i commutant avec σ_j chaque fois que $i \neq j$,*
- *r indéterminées ∂_i,*

l'algèbre de Ore (rationnelle) notée $k(x_1, \ldots, x_r)\langle \partial_1, \ldots, \partial_r; \sigma_1, \ldots, \sigma_r, \delta_1, \ldots, \delta_r\rangle$ ou plus simplement $k(x)\langle \partial; \sigma, \delta\rangle$ est la $k(x)$-algèbre associative engendrée par les ∂_i modulo les relations

$$\partial_i a = \sigma_i(a)\partial_i + \delta_i(a), \qquad \partial_i\partial_j = \partial_j\partial_i,$$

quand a est dans $k(x)$. On note plus simplement $k(x_1, \ldots, x_r)\langle \partial_1, \ldots, \partial_r; \sigma_1, \ldots, \sigma_r\rangle$ ou encore $k(x)\langle \partial; \sigma\rangle$ le cas où tous les δ_i sont nuls.

Donnons un exemple : en notant x pour l'opérateur de multiplication par x et n pour celui de multiplication par n, on vérifie l'existence d'une algèbre de Ore $A = \mathbb{C}(n, x)\langle \partial_n, \partial_x; S_n, \mathrm{id}, 0, D_x\rangle$ avec $S_n(n) = n + 1$, $S_n(x) = x$, $D_x(n) = 0$, $D_x(x) = 1$, et plus généralement $S_n(a) = a(n + 1, x)$ et $D_x(a) = da/dx$ quand $a = a(n, x) \in \mathbb{C}(n, x)$.

2. Idéal annulateur et module quotient

Les éléments des algèbres de Ore représentent des opérateurs linéaires, différentiels, de récurrence, ou autres, et agissent donc sur des fonctions, suites, suites de fonctions, etc. Donnons un exemple qui montre que ces objets sont une bonne représentation polynomiale des opérateurs linéaires, l'exemple de la famille des polynômes orthogonaux de Laguerre qui va nous servir pour toute la suite du chapitre.

Pour chaque paramètre strictement positif α, l'intégrale

$$\langle f, g\rangle = \int_0^\infty f(x)g(x)x^\alpha e^{-x}\,dx$$

définit un produit scalaire sur les fonctions polynomiales réelles. Par la théorie des polynômes orthogonaux, on déduit l'existence de bases orthogonales échelonnées en

degré. On a par exemple la base des polynômes orthogonaux de Laguerre, donnée par

$$L_n^{(\alpha)}(x) = \frac{1}{n!}x^{-\alpha}e^x\left(\frac{d}{dx}\right)^n\left(e^{-x}x^{n+\alpha}\right) = \frac{1}{n!}\sum_{k=0}^{n}(-1)^k\binom{n}{k}(\alpha+k+1)\cdots(\alpha+n)x^k.$$

On vérifie que ces polynômes vérifient les relations (linéaires)

$$\begin{aligned}(n+2)L_{n+2}^{(\alpha)} - (2n+\alpha+3-x)L_{n+1}^{(\alpha)} + (n+\alpha+1)L_n^{(\alpha)} &= 0,\\ xL_n^{(\alpha)\prime} - (n+1)L_{n+1}^{(\alpha)} + (n+\alpha+1-x)L_n^{(\alpha)} &= 0,\\ xL_n^{(\alpha)\prime\prime} + (\alpha+1-x)L_n^{(\alpha)\prime} + nL_n^{(\alpha)} &= 0,\end{aligned}$$

avec les conditions initiales $L_0^{(\alpha)} = 1$ et $L_1^{(\alpha)} = \alpha + 1 - x$. Dans l'algèbre de Ore A ci-dessus, ces équations se recodent en les polynômes tordus suivant, qui annulent la suite de fonctions polynomiales $L^{(\alpha)}$:

$$\begin{aligned}p_1 &= (n+2)\partial_n^2 - (2n+\alpha+3-x)\partial_n + (n+\alpha+1),\\ p_2 &= x\partial_x - (n+1)\partial_n + (n+\alpha+1-x),\\ p_3 &= x\partial_x^2 + (\alpha+1-x)\partial_x + n.\end{aligned}$$

Ces trois polynômes engendrent un idéal à gauche dans A, l'idéal annulateur de $L^{(\alpha)}$ dans A. Pour mémoire, un idéal à gauche I d'un anneau R est un sous-ensemble non vide de R stable par addition et par multiplication à gauche par tout élément de R. Cette stabilité reflète le fait que l'addition terme à terme de deux relations linéaires vérifiées par $L^{(\alpha)}$ est une nouvelle relation linéaire vérifiée par $L^{(\alpha)}$, de même qu'en appliquant un opérateur linéaire sur une relation linéaire vérifiée par $L^{(\alpha)}$, on retrouve une relation linéaire vérifiée par $L^{(\alpha)}$.

Exemple 1. En partant de la troisième équation donnée pour caractériser les polynômes de Laguerre, un décalage avant de n et une dérivation par rapport à x donnent respectivement

$$xL_{n+1}^{(\alpha)\prime\prime} + (\alpha+1-x)L_{n+1}^{(\alpha)\prime} + (n+1)L_{n+1}^{(\alpha)} = 0$$

et

$$xL_n^{(\alpha)\prime\prime\prime} + (\alpha+2-x)L_n^{(\alpha)\prime\prime} + (n-1)L_n^{(\alpha)\prime} = 0.$$

L'addition de ces deux équations donne l'équation fonctionnelle linéaire

$$xL_n^{(\alpha)\prime\prime\prime} + (\alpha+2-x)L_n^{(\alpha)\prime\prime} + (n-1)L_n^{(\alpha)\prime} + xL_{n+1}^{(\alpha)\prime\prime} + (\alpha+1-x)L_{n+1}^{(\alpha)\prime} + (n+1)L_{n+1}^{(\alpha)} = 0,$$

laquelle correspond au polynôme tordu

$$(\partial_x+\partial_n)p_3 = x\partial^3 + (\alpha+2-x)\partial_x^2 + (n-1)\partial_x + x\partial_x^2\partial_n + (\alpha+1-x)\partial_x\partial_n + (n+1)\partial_n.$$

Dans le cas commutatif, le quotient d'une algèbre A de polynômes par l'un de ses idéaux (bilatères) I reste munie d'un produit canonique et est donc une algèbre. Cette propriété n'est plus réalisée dans le cas d'une algèbre de Ore $A = k(x)\langle\partial;\sigma,\delta\rangle$. Mais, en voyant A comme un idéal à gauche trivial de lui-même, le quotient A/I conserve une addition canonique, ainsi que la stabilité par multiplication à gauche par tout élément de A, ce qui fait de ce quotient un module à gauche sur A. Ce module est en particulier un espace vectoriel sur $k(x)$.

Dans le cas commutatif, un cadre particulier important est celui d'un quotient de dimension finie comme espace vectoriel, car il représente une famille finie de points solutions. Le cas d'un quotient d'une algèbre de Ore qui est un espace vectoriel sur $k(x)$ de dimension finie est lui-aussi important ; dans l'interprétation

en opérateurs linéaires, il correspond en règle générale à un espace vectoriel de solutions de dimension finie sur k.

Dans la Section 4, nous allons détailler ce lien dans le cas d'opérateurs différentiels ; d'autres cadres fournissent le même genre de résultats. Nous irons plus loin sur le sujet dans la Section 5 sur les fonctions ∂-finies.

Toute autre famille de polynômes orthogonaux classique se traiterait de la même manière et aurait pu servir de support à ce chapitre. La même nature de système linéaire avec une dérivation sur une variable et un décalage sur une autre permet de traiter de la même façon nombre de familles de fonctions spéciales paramétrées, telles les fonctions de Bessel, de Hankel, etc.

3. Bases de Gröbner pour les idéaux à gauche

La propriété essentielle qui fait fonctionner toute la théorie des bases de Gröbner et l'algorithme de Buchberger dans le cadre de polynômes commutatifs est que le monôme de tête d'un produit de polynômes est le produit des monômes de tête des termes du produit. Cette propriété reste vérifiée sur des polynômes non commutatifs sujets aux relations de définition des algèbres de Ore rationnelles, dès lors qu'on considère des ordres monomiaux sur les ∂_i. En refaisant la théorie en s'efforçant de faire toutes les combinaisons linéaires avec des facteurs à gauche, on obtient le résultat suivant (*cf.* le Chapitre 21 et la Section 3 du Chapitre 22 sur les bases de Gröbner classiques) :

THÉORÈME. *Soit A une algèbre de Ore rationnelle.*

(i) *Tout idéal à gauche I de A admet pour chaque ordre monomial (admissible) sur les ∂_i une unique base de Gröbner minimale réduite G, au sens où l'une quelconque des propriétés équivalentes suivantes est vérifiée :*

1. *la partie stable du monoïde des monômes en les ∂_i engendrée par les monômes de tête des éléments de G est égale celle engendrée par ceux de I ;*
2. *tout f non nul de I est réductible par G ;*
3. *pour tout f dans A, il existe un unique r dans A dont aucun monôme ne soit divisible par un monôme de tête d'un élément de G et tel que $f-r$ soit dans l'idéal I ;*
4. *pour tout f dans I, le reste de la division (à droite) de f par G est nul.*

(ii) *Soit $P = \{p_k\}_{1\leq k\leq r}$ un système de générateurs non nuls d'un idéal à gauche I de A. Tous les S-polynômes* $\mathrm{Spoly}(p_i, p_j)$ *(définis par des combinaisons linéaires à gauche) se réduisent à 0 par P si et seulement si P est une base de Gröbner de l'idéal.*

(iii) *Une variante de l'algorithme de Buchberger termine et renvoie une base de Gröbner de tout idéal à gauche I de A.*

Une spécificité du cas non commutatif réside dans le calcul des S-polynômes. Considérons le cas commutatif et faisons l'hypothèse que les calculs, pour des raisons d'efficacité, écrivent tous les polynômes en chassant les dénominateurs. (Ceci suppose donc que le corps des coefficients peut être vu comme l'anneau des fractions d'un certain anneau, en règle générale $\mathbb{Z}[\alpha, \dots]$ pour des paramètres α, ...) Le S-polynôme de deux polynômes non nuls f_1 et f_2 se définit alors par

$$\mathrm{Spoly}(f_1, f_2) = \frac{c_2}{c_1 \wedge c_2}\frac{m_2}{m_1 \wedge m_2} f_1 - \frac{c_1}{c_1 \wedge c_2}\frac{m_1}{m_1 \wedge m_2} f_2,$$

où $u \wedge v$ est une notation pour le p. g. c. d., c_i dénote le coefficient de tête de f_i et m_i son monôme de tête, pour $i = 1, 2$. Cette formule doit être adaptée dans le cas de polynômes tordus : chacun des c_i dénote maintenant le coefficient de tête de

$$\frac{m_{3-i}}{m_1 \wedge m_2} f_i,$$

qui peut dépendre des σ_j, mais où les notions divisions et p. g. c. d. sont vues comme si les m_i étaient dans un monde commutatif.

Plutôt que de poursuivre la théorie dans le détail, montrons le calcul sur un exemple.

En repartant des polynômes p_1 et p_2 qui annulent la suite des polynômes orthogonaux de Laguerre, montrons que le polynôme p_3 s'obtient par élimination de S par un calcul pour l'ordre $\mathsf{lex}(\partial_n, \partial_x)$. Pour cet ordre, le terme de tête de p_1 est $(n+2)\times\partial_n^2$, celui de p_2 est $-(n+1)\times\partial_n$. On calcule donc d'abord le S-polynôme de p_1 et de p_2, sous la forme

$$\mathrm{Spoly}(p_1, p_2) = p_1 + \partial_n p_2 = x\partial_x\partial_n - (n+1)\partial_n + (n+\alpha+1).$$

Remarquons que ce S-polynôme n'est pas $(n+1)p_1 + (n+2)\partial_n p_2$, lequel aurait $(n+2)\partial_n^2$ comme terme de tête, et non pas $\partial_x\partial_n$ pour monôme de tête. Il est réductible par p_2 ; après multiplication par $(n+1)$ et ajout de $x\partial_x p_2$, on obtient

$$x^2\partial_x^2 + (n+\alpha+2-x)x\partial_x - (n+1)^2\partial_n + (n+1)(n+\alpha+1) - x.$$

Ce polynôme a ∂_n pour monôme de tête et est réductible par p_2 ; après retranchement de $(n+1)p_2$, on aboutit à

$$x^2\partial_x^2 + (\alpha+1-x)x\partial_x + nx,$$

qui n'est autre que xp_3. En poursuivant, on montre que les S-polynômes de p_1 et p_2 avec p_3 se réduisent à 0 ; puisque le monôme de tête de p_2, ∂_n, divise celui de p_1, ∂_n^2, une base de Gröbner minimale pour l'ordre $\mathsf{lex}(\partial_n, \partial_x)$ est $\{p_2, p_3\}$.

De façon analogue, une base de Gröbner pour l'ordre $\mathsf{lex}(\partial_x, \partial_n)$ de l'idéal engendré par p_2 et p_3 est $\{p_1, p_2\}$. Les bases de Gröbner permettent de déterminer la redondance du système $\{p_1, p_2, p_3\}$.

Exercice 1. Calculer une base de Gröbner pour l'ordre $\mathsf{lex}(\partial_x, \partial_n)$ de l'idéal engendré par p_2 et p_3 et vérifier le point ci-dessus.

Les polynômes p_1, p_2, p_3 qui annulent la suite des polynômes orthogonaux de Laguerre sont encore plus contraints qu'il n'y paraît jusqu'à présent : p_2 se déduit en fait de p_1. En effet, ne connaissant que p_1, on peut rechercher le polynôme p_2 sous la forme indéterminée

$$p_2 = \partial_x - u(n,x)\partial_n - v(n,x),$$

pour des fractions rationnelles à déterminer u et v, et faire l'hypothèse heuristique que $\{p_1, p_2\}$ est une base de Gröbner pour l'ordre $\mathsf{lex}(\partial_x, \partial_n)$. (Cette hypothèse heuristique est en fait naturelle dès qu'on sait qu'on a affaire à une famille de polynômes orthogonaux.)

Exercice 2 (Presque un problème). Utiliser la théorie des bases de Gröbner pour donner un système de récurrences linéaires sur u et v qui, après résolution, redonne le polynôme p_2. (Pour la résolution, on se souviendra des conditions initiales $L_0^{(\alpha)} = 1$ et $L_1^{(\alpha)} = \alpha + 1 - x$.)

4. Module quotient et dimension de l'espace des solutions

4.1. Séries formelles solutions en un point régulier dans le cas différentiel. Considérons une algèbre de Ore

$$A = \mathbb{C}(x_1, \ldots, x_r)\langle\partial_1, \ldots, \partial_r; \mathrm{id}, \ldots, \mathrm{id}, D_{x_1}, \ldots, D_{x_r}\rangle,$$

un idéal à gauche I de cette algèbre, donné par un système différentiel linéaire. Nous voulons décrire les solutions séries annulées par tous les éléments de I, où une série est ici un élément de $\mathbb{C}[[x_1, \ldots, x_r]]$, c'est-à-dire une combinaison linéaire formelle éventuellement infinie de monômes à exposants entiers positifs. Dans cette

objectif, cette section ébauche un analogue en plusieurs variables de la conversion entre équation différentielle décrivant une série différentiellement finie d'une variable et équation de récurrence vérifiée par la suite polynomialement récurrente des coefficients.

Fixons un ordre monomial sur les monômes en les ∂_i, puis, pour cet ordre, une base de Gröbner B de I, donnée par des éléments de A sans fractions, c'est-à-dire avec des coefficients polynomiaux. Cette base de Gröbner B fournit un escalier ; notons S l'ensemble des multi-exposants $s = (s_1, \dots, s_r)$ des monômes $\partial^s = \partial_1^{s_1} \cdots \partial_r^{s_r}$ sous l'escalier, c'est-à-dire des monômes qui ne sont pas réductibles par B. Le module quotient A/I a alors une base d'espace vectoriel sur $\mathbb{C}(x)$ constituée des $\partial^s + I$, les classes des ∂_s modulo I, pour s décrivant S. Soit u le polynôme produit des coefficients de tête des éléments de B et faisons l'hypothèse que u ne s'annule pas pour $x_1 = \cdots = x_r = 0$. Nous affirmons qu'alors, l'idéal I admet un espace vectoriel de solutions séries dans $\mathbb{C}[[x_1, \dots, x_r]]$ de dimension le cardinal de S, c'est-à-dire la dimension sur $\mathbb{C}(x)$ de A/I vu comme espace vectoriel. On dit dans ce cas que le point $(0, \dots, 0)$ est régulier pour le système différentiel linéaire définissant l'idéal I.

En effet, pour tout multi-exposant $n = (n_1, \dots, n_r)$, la réduction du monôme ∂^n par B fournit une combinaison linéaire $\sum_{s\in S} v_{n,s}\partial^s$ congrue à ∂^n modulo I. Ceci généralise la réécriture d'un ∂^N faite en Section 3 du Chapitre 28. Notons que par construction, les coefficients $v_{n,s}$ sont éléments de $\mathbb{C}[x_1, \dots, x_r, u^{-1}]$ et ont ainsi une évaluation bien définie en $x_1 = \cdots = x_r = 0$. Maintenant, puisque chaque élément de I s'annule sur toute solution série

$$\phi = \sum_{n_1\in\mathbb{N},\dots,n_r\in\mathbb{N}} c_{n_1,\dots,n_r} x_1^{n_1} \cdots x_r^{n_r}$$

de I, le monôme ∂^n et la somme $\sum_{s\in S} v_{n,s}\partial^s$ ont la même action sur ϕ :

$$\partial^n \cdot \phi = \sum_{s\in S} v_{n,s}\partial^s \cdot \phi.$$

Une évaluation en $x_1 = \cdots = x_r = 0$ donne la relation

$$n_1! \dots n_r!\, c_{n_1,\dots,n_r} = \sum_{s\in S} v_{n,s}(0, \dots, 0) s_1! \dots s_r!\, c_{s_1,\dots,s_r}.$$

Autrement dit, la série ϕ est totalement déterminée par ses quelques premiers coefficients c_s pour $s \in S$, en nombre donné par la dimension de A/I.

Illustrons cette idée en reprenant l'exemple des polynômes orthogonaux de Laguerre, qui étendent déjà légèrement le cadre purement différentiel qui précède. Posons

$$L_n^{(\alpha)}(x) = \sum_{k=0}^{n} \ell_{n,k} x^k.$$

En multipliant chaque p_i pour $i = 1, 2, 3$ par ∂_x^k, il vient

$$\begin{aligned}
\partial_x^k p_1 &= (n+2)\partial_n^2\partial_x^k - (2n+\alpha+3-x)\partial_n\partial_x^k + k\partial_n\partial_x^{k-1} + (n+\alpha+1)\partial_x^k,\\
\partial_x^k p_2 &= x\partial_x^{k+1} - (n+1)\partial_n\partial_x^k + (n+\alpha+k+1-x)\partial_x^k - k\partial_x^{k-1},\\
\partial_x^k p_3 &= x\partial_x^{k+2} + (\alpha+k+1-x)\partial_x^{k+1} + (n-k)\partial_x^k.
\end{aligned}$$

Après application sur $L^{(\alpha)}$, évaluation en $x = 0$ et division par $k!$, on trouve les relations de récurrence sur la famille doublement indexée des $\ell_{n,k}$

$$\begin{aligned}
(n+2)\ell_{n+2,k} - (2n+\alpha+3)\ell_{n+1,k} + \ell_{n+1,k-1} + (n+\alpha+1)\ell_{n,k} &= 0,\\
-(n+1)\ell_{n+1,k} + (n+\alpha+k+1)\ell_{n,k} - \ell_{n,k-1} &= 0,\\
(k+1)(\alpha+k+1)\ell_{n,k+1} + (n-k)\ell_{n,k} &= 0.
\end{aligned}$$

En décalant la dernière vers l'arrière en k puis éliminant $\ell_{n,k-1}$ entre la relation obtenue et la deuxième récurrence ci-dessus, on obtient la récurrence

$$(n+1-k)\ell_{n+1,k} - (n+\alpha+1)\ell_{n,k} = 0.$$

Ce jeu de récurrences fournit tous les $\ell_{n,k}$.

4.2. Solutions en séries des systèmes hypergéométriques de Gel'fand, Kapranov et Zelevinsky. Prenons un exemple concret, celui des systèmes hypergéométriques dans la formulation de Gel'fand, Kapranov et Zelevinsky. L'algèbre de Ore qui intervient dans cet exemple est l'algèbre A engendrée par quatre indéterminées ∂_1, …, ∂_4 sur le corps $\mathbb{C}(x_1, \ldots, x_4)$, chaque ∂_i représentant l'opérateur de dérivation par rapport à x_i. Le système GKZ est le système

$$\begin{aligned} p_1 &= \partial_2\partial_3 - \partial_1\partial_4, \\ p_2 &= x_1\partial_1 - x_4\partial_4 + (1-c), \\ p_3 &= x_2\partial_2 + x_4\partial_4 + a, \\ p_4 &= x_3\partial_3 + x_4\partial_4 + b, \end{aligned}$$

pour des paramètres complexes a, b et c génériques. L'objectif de l'exemple est de montrer que ce système admet un espace vectoriel de solutions formelles de dimension exactement 2, où par solution formelle nous entendons maintenant une série plus générale de la forme

$$x_1^{a_1}\ldots x_4^{a_4} \sum_{n_1\in\mathbb{Z},\ldots,n_4\in\mathbb{Z}} c_{n_1,\ldots,n_4} x_1^{n_1}\cdots x_4^{n_4},$$

pour des a_i et des coefficients complexes, ou une combinaison linéaire de telles séries. (Il y a bien un espace vectoriel sur $\mathbb{C}$ où vivent ces séries, mais pas de produit sur ces séries. En revanche, toute série peut être multipliée par un polynôme en les x_i et leurs inverses x_i^{-1}, ainsi que dérivée formellement par rapport à chacune des indéterminées, tout en restant dans l'espace vectoriel.)

Soit I l'idéal engendré par le système $\{p_1, \ldots, p_4\}$ et calculons à partir de ce système une base de Gröbner de I pour l'ordre $\mathsf{lex}(\partial_1, \ldots, \partial_4)$. Les monômes de tête respectifs de p_1 et p_2 sont $\partial_1\partial_4$ et ∂_1. Le S-polynôme de p_1 et de p_2 est donc

$$\mathrm{Spoly}(p_1, p_2) = x_1p_1 + \partial_4 p_2 = x_1\partial_2\partial_3 - x_4\partial_4^2 - c\partial_4,$$

dont le monôme de tête est $\partial_2\partial_3$; il est donc réductible par p_3. Après multiplication par $-x_2$ et ajout de $x_1\partial_3 p_3$, on obtient

$$x_1x_4\partial_3\partial_4 + ax_1\partial_3 + x_2x_4\partial_4^2 + cx_2\partial_4.$$

Ce polynôme a $\partial_3\partial_4$ pour monôme de tête et est donc réductible par p_4. Après multiplication par x_3 et retranchement de $x_1x_4\partial_4 p_4$, on aboutit à

$$ax_1x_3\partial_3 + (x_2x_3 - x_1x_4)x_4\partial_4^2 + \big(cx_2x_3 - (b+1)x_1x_4\big)\partial_4,$$

qui est encore réductible par p_4. Après retranchement de ax_1p_4, on a finalement un polynôme qui n'est pas réductible par $\{p_1, \ldots, p_4\}$, à savoir

$$p_5 = (x_2x_3 - x_1x_4)x_4\partial_4^2 + \big(cx_2x_3 - (a+b+1)x_1x_4\big)\partial_4 - abx_1.$$

Par ailleurs, les S-polynômes entre les polynômes p_2, p_3 et p_4 pris deux à deux sont tous nuls, comme on le vérifie en observant que les $x_i\partial_i$ commutent deux à deux. En poursuivant les calculs sur les S-polynômes $\mathrm{Spoly}(p_i, p_5)$, on montre que tous ces derniers se réduisent à 0 par $\{p_1, \ldots, p_5\}$. On obtient ainsi qu'une base de Gröbner minimale est $\{p_2, p_3, p_4, p_5\}$, avec les monômes de tête respectifs ∂_1, ∂_2, ∂_3 et ∂_4^2.

Le module quotient A/I a donc une base d'espace vectoriel sur $\mathbb{C}(x_1, \ldots, x_4)$ constituée de $1+I$ et ∂_4+I, les classes respectives de 1 et ∂_4 modulo I. La structure

de module est donnée explicitement par l'action des ∂_i sur ces deux éléments de base.

EXERCICE 3. Donner l'expression explicite de cette action en récrivant chaque $\partial_i \cdot (1+I)$ et chaque $\partial_i \cdot (\partial_4 + I)$ sur la base $(1+I, \partial_4 + I)$.

Revenons sur les solutions séries du système GKZ. Le polynôme p_2 agit sur un monôme par

$$p_2 \cdot (x_1^{\lambda_1} \cdots x_4^{\lambda_4}) = (\lambda_1 - \lambda_4 + 1 - c) x_1^{\lambda_1} \cdots x_4^{\lambda_4}.$$

(Notons la distinction entre le produit dans A noté $p_2 x_1^{\lambda_1} \cdots x_4^{\lambda_4}$ et l'opération de p_2, ici sur une serie h en x, notée $p_2 \cdot h$; on comparera par exemple $\partial_1 x_1^5 = x_1^5 \partial_1 + 5x_1^4$ et $\partial_1 \cdot x_1^5 = 5x_1^4$.) Ainsi, un monôme $x_1^{\lambda_1} \cdots x_4^{\lambda_4}$ ne peut apparaître avec un coefficient non nul dans une série ϕ solution du système GKZ que si $\lambda_1 - \lambda_4 + 1 - c$ est nul. En poursuivant ce type de raisonnement avec p_3 et p_4, on obtient de même les contraintes $\lambda_2 + \lambda_4 + a = 0$ et $\lambda_3 + \lambda_4 + a = 0$ et on aboutit à ce que les seuls monômes pouvant apparaître avec un coefficient non nul sont de la forme

$$x_1^{\lambda_4 + c - 1} x_2^{-\lambda_4 - a} x_3^{-\lambda_4 - b} x_4^{\lambda_4} = \frac{x_1^{c-1}}{x_2^a x_3^b} \left(\frac{x_1 x_4}{x_2 x_3} \right)^{\lambda_4},$$

et une solution ϕ est nécessairement de la forme

$$\phi = \frac{x_1^{c-1}}{x_2^a x_3^b} f\left(\frac{x_1 x_4}{x_2 x_3} \right),$$

pour une série formelle f en y à exposants entiers relatifs. Reste à exploiter que ϕ est solution de p_1, ou de manière équivalente, puisque sous la forme ci-dessus ϕ est déjà solution de p_2, p_3 et p_4, de p_5. Après avoir évalué en $y = x_1 x_4 / x_2 x_3$, on a

$$0 = \frac{x_2^a x_3^b}{x_1^{c-2}} p_5 \cdot \phi = (1-y) y f''(y) + \big(c - (a+b+1)y\big) f'(y) - ab f(y).$$

On reconnaît là l'équation hypergéométrique de Gauss, annulée par la série

$$f_1 = {}_2\mathrm{F}_1(a, b; c; y) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n}{(c)_n} \frac{y^n}{n!}$$

déjà rencontré en Section 6.1 du Chapitre 28. On vérifie qu'une solution formelle linéairement indépendante avec f_1 est $f_2 = y^{1-c} {}_2\mathrm{F}_1(a-c+1, b-c+1; 2-c; y)$. On a ainsi obtenu deux solutions formelles linéairement indépendantes du système GKZ, $\phi_i = x_1^{c-1} / x_2^a x_3^b \times f_i(x_1 x_4 / x_2 x_3)$ pour $i = 1$ et $i = 2$.

Pour tout système différentiel représenté par un idéal I de A, un résultat d'analyse, le théorème de Cauchy–Kovalevskaya, affirme l'existence, au voisinage de tout point en dehors d'une certaine variété singulière, d'un $\mathbb{C}$-espace vectoriel de solutions analytiques de dimension celle sur $\mathbb{C}(x)$ de l'espace vectoriel A/I. Or, on montre que cette variété singulière est incluse dans le lieu des zéros du produit des coefficients polynomiaux de tête d'une base de Gröbner de I écrite sans fractions.

Dans le cas de notre exemple, la dimension de A/I est 2 et la variété singulière est incluse dans le lieu des zéros de $x_1 \cdots x_4 (x_2 x_3 - x_1 x_4)$. Hors de ce lieu, y n'est ni nul, ni infini, ni égal à 1, et les fonctions ϕ_1 et ϕ_2 sont donc analytiques puisque, moyennant quelques hypothèses sur les paramètres a, b et c, les deux séries solutions de l'équation de Gauss, f_1 et f_2, représentent des fonctions analytiques sur $\mathbb{C} \setminus \{0, 1\}$. On a donc trouvé un espace de solutions analytiques de dimension 2, et par le théorème de Cauchy–Kovalevskaya, toutes les solutions analytiques du système GKZ en dehors de sa variété singulière pour a, b et c génériques.

5. Les fonctions ∂-finies et leurs clôtures

Nous poursuivons maintenant avec le cas particulier important des systèmes fonctionnels linéaires correspondant à des modules A/I de dimension finie sur $\mathbb{C}(x)$. L'objectif est ici de montrer que pour une algèbre A donnée, leurs solutions, que nous appellerons « fonctions ∂-finies », forment une algèbre sur $\mathbb{C}$. Nous allons donner un algorithme pour calculer les clôtures correspondantes. Voici tout de suite la définition, déjà motivée par les sections et chapitres précédents sur les fonctions D-finies et les suites P-récursives (en particulier, le Chapitre 14).

Définition (Fonction ∂-finie). *Étant donnée une algèbre de Ore (rationnelle)*

$$A = \mathbb{C}(x_1, \dots, x_r)\langle \partial_1, \dots, \partial_r; \sigma_1, \dots, \sigma_r, \delta_1, \dots, \delta_r \rangle$$

agissant sur un $\mathbb{C}(x_1, \dots, x_r)$-espace vectoriel V, un élément f de V est dit ∂-fini lorsque les conditions équivalentes suivantes sont vérifiées :

1. *pour chaque i entre 1 et r, il existe un polynôme $P_i = P_i(x_1, \dots, x_r, \partial_i)$ non nul dont l'action annule f ;*
2. *la famille des $\partial^a \cdot f$, où ∂^a décrit les monômes de A, engendre un espace vectoriel de dimension finie sur $\mathbb{C}(x)$;*
3. *le module quotient A/I où I note l'idéal annulateur de f pour l'action de A est un espace vectoriel de dimension finie sur $\mathbb{C}(x)$.*

Par commodité, nous appellerons « fonctions » les éléments de l'espace vectoriel V, ce quand bien même il ne s'agirait pas de fonctions de l'analyse, mais afin d'éviter la terminologie plus lourde et moins imagée d'« éléments ∂-finis d'un module sur A ».

Vérifions l'équivalence entre les trois points de la définition ci-dessus. Faisons d'abord l'hypothèse de la première condition, en notant d_i le degré de P_i en ∂_i. Alors, tout ∂^a a un reste par division par les P_i qui s'exprime comme combinaison linéaire des $\partial_1^{j_1} \dots \partial_r^{j_r}$ pour $0 \le j_1 < d_1$, …, $0 \le j_r < d_r$. Ainsi, la famille des $\partial^a \cdot f$ reste dans l'espace vectoriel engendré par les $\partial_1^{j_1} \dots \partial_r^{j_r} \cdot f$ pour $0 \le j_1 < d_1$, …, $0 \le j_r < d_r$, et la deuxième condition est prouvée. De manière réciproque, pour chaque i, la famille des $\partial_i^j \cdot f$ est une sous-famille de celle des $\partial^a \cdot f$; quand cette dernière engendre un espace vectoriel de dimension finie, la première est une famille liée, ce qui fournit un polynôme non nul P_i. Enfin, l'application linéaire sur A qui à P de A associe $P \cdot f$ dans l'espace vectoriel V engendré par les $\partial^a \cdot f$ est surjective, de noyau I. Les A-modules V et A/I sont donc isomorphes, et ils le sont donc aussi en tant qu'espaces vectoriels sur $\mathbb{C}(x)$. Ceci prouve l'équivalence entre les deux dernières conditions.

5.1. Méthode du vecteur cyclique et généralisation à plusieurs variables. L'algorithme envisagé pour les clôtures des fonctions ∂-finies s'appuie le calcul de vecteur cyclique. Rappelons-en l'idée. Classiquement, étant donné un espace vectoriel V sur un corps k, sur lequel on suppose donnée une action de $k[X]$, un vecteur $v \in V$ est dit *cyclique* si la famille $\{X^i \cdot v\}$ engendre V comme k-espace vectoriel. Alors, v engendre V comme $k[X]$-module. Pour calculer, on suppose que V est de dimension finie d et que l'action de X est donnée sur une base $B = (b_1, \dots, b_d)$ de V par une matrice M telle que $X \cdot w = (a_1, \dots, a_d)M\,{}^tB$ pour tout vecteur $w = a_1 b_1 + \cdots + a_d b_d = (a_1, \dots, a_d)\,{}^tB$. Pour tester si v est cyclique et le cas échéant rendre explicite la structure de module de V, on range dans une matrice les lignes $(a_1, \dots, a_d)M^i$ pour $0 \le i \le m$ avec m à déterminer et on cherche, par exemple par l'algorithme de Gauss, une dépendance linéaire entre les lignes de la matrice obtenue. En procédant successivement avec $m = 0, 1, 2, \dots$, la

première dépendance linéaire fournit le polynôme minimal de v sous l'action de X sur V ; son degré m vérifie $m \leq d$.

Ce calcul s'étend au cadre commutatif de l'action d'une algèbre $k[X] = k[X_1, \ldots, X_r]$ de polynômes en plusieurs indéterminées. (Il peut être vu comme une réminiscence de l'algorithme de [2] pour le changement d'ordre dans les bases de Gröbner du cadre commutatif.) Chaque X_i correspond alors à une matrice M_i et la commutativité des X_i dans $k[X]$ induit la commutativité entre les matrices M_i. Au lieu d'itérer sur les monômes X^i par ordre croissant de i, on itère maintenant sur les monômes $X^a = X_1^{a_1} \cdots X_r^{a_r}$ selon tout ordre qui assure qu'un monôme n'est considéré qu'après tous ses diviseurs. Soit $a(0)$, $a(1)$, etc, l'ordre dans lequel les multi-exposants des monômes sont énumérés. À chaque étape, on recherche une dépendance linéaire entre des vecteurs

$$X^{a(0)} \cdot v, \ldots, X^{a(m)} \cdot v.$$

En cas d'échec, on conserve ces $m+1$ vecteurs et on reprend la recherche après avoir ajouté le nouveau vecteur $X^{a(m+1)} \cdot v$; en cas de succès, on retire le dernier vecteur introduit, $X^{a(m)} \cdot v$, on évite par la suite tous les multiples du monôme $X^{a(m)}$, et on introduit le nouveau vecteur $X^{a(m+1)} \cdot v$ pour reprendre la recherche sur la famille

$$X^{a(0)} \cdot v, \ldots, X^{a(m-1)} \cdot v, X^{a(m+1)} \cdot v.$$

Ce calcul termine si et seulement si le quotient $k[X]/I$, vu comme k-espace vectoriel, est de dimension finie. Chaque dépendance linéaire calculée fournit un polynôme P tel que $P(X_1, \ldots, X_r) \cdot v = 0$. Lorque l'itération sur les monômes X^a suit l'ordre croissant selon un ordre monomial (admissible), l'ensemble des polynômes annulateurs obtenus constitue une base de Gröbner de I pour l'ordre choisi, car l'itération maintient une famille de X^a qui sont tous strictement sous l'escalier, sauf au plus un élément qui est alors sur l'escalier. Par construction, les P obtenus ont cet élément sur un coin de l'escalier.

5.2. Algorithmes de clôture des fonctions ∂-finies. Le procédé du vecteur cyclique s'étend au cas de l'action d'une algèbre de Ore (rationnelle) en présence de fonctions ∂-finies. Pour une algèbre de Ore

$$A = \mathbb{C}(x)\langle \partial_1, \ldots, \partial_r; \sigma_1, \ldots, \sigma_r, \delta_1, \ldots, \delta_r \rangle,$$

l'espace vectoriel utilisé est un module du type A/I, vu comme espace vectoriel sur $\mathbb{C}(x)$, ou plutôt un module obtenu à partir de quelques constructions de base sur des modules de la forme A/I, comme on va le voir sur l'exemple plus bas. L'espace V étant d'une certaine dimension finie d et une base $B = (b_1, \ldots, b_d)$ de V étant fixée, l'action de chaque ∂_i sur un vecteur $v = a_1 b_1 + \cdots + a_d b_d$ est donnée par une matrice M_i indépendante de v sous la forme

$$\partial_i \cdot v = \big(\sigma_i(a_1, \ldots, a_d) M_i + \delta_i(a_1, \ldots, a_d)\big)\, {}^tB, \tag{1}$$

en adoptant une notation selon laquelle les σ_i et δ_i agissent distributivement sur les coefficients de vecteurs ou de matrices. Pour le choix particulier $v = b_\ell$, c'est-à-dire $a_i = 1$ quand $i = \ell$ et $a_i = 0$ sinon, on observe que la ℓ^{e} ligne de M_i n'est autre que le vecteur ligne des composantes de $\partial_i \cdot b_\ell$ sur la base B, de sorte que $\partial_i \cdot {}^tB = M_i {}^tB$ (avec encore une notation où ∂_i agit distributivement sur les coefficients du vecteur).

En faisant maintenant agir ∂_j sur (1) et en posant $a = (a_1, \ldots, a_d)$, on a

$$\begin{aligned}
\partial_j \partial_i \cdot v &= \partial_j \cdot \big(\sigma_i(a) M_i + \delta_i(a)\big)\, {}^tB \\
&= \sigma_j\big(\sigma_i(a) M_i + \delta_i(a)\big)\, \big(\partial_j \cdot {}^tB\big) + \delta_j\big(\sigma_i(a) M_i + \delta_i(a)\big)\, {}^tB \\
&= \big(\sigma_j\sigma_i(a)\sigma_j(M_i) + \sigma_j\delta_i(a)\big) M_j + \big(\sigma_j\sigma_i(a)\delta_j(M_i) + \delta_j\sigma_i(a) M_i + \delta_j\delta_i(a)\big)\, {}^tB \\
&= \big(\sigma_j\sigma_i(a)\sigma_j(M_i) M_j + \sigma_j\delta_i(a) M_j + \sigma_j\sigma_i(a)\delta_j(M_i) + \delta_j\sigma_i(a) M_i + \delta_j\delta_i(a)\big)\, {}^tB.
\end{aligned}$$

En tenant compte, pour $i \neq j$ de la commutation $\partial_i \partial_j = \partial_j \partial_i$ et des commutations entre endomorphismes et σ-dérivations données par la définition des algèbres de Ore, on déduit la relation suivante, qui remplace la commutation entre les matrices M_ℓ du cas commutatif,

$$\sigma_j(M_i)M_j + \delta_j(M_i) = \sigma_i(M_j)M_i + \delta_i(M_j).$$

(La simplification par $\sigma_i\sigma_j(a)$ se justifie quand les σ_ℓ sont surjectifs.) Lorsque de telles relations sont assurées, la même méthode de recherche de dépendances linéaires par la méthode de Gauss que dans le cas commutatif s'applique et fournit un calcul de l'addition et du produit de fonctions ∂-finies, ou même d'une expression polynomiale en des fonctions ∂-finies. Dans la pratique, les matrices M_i sont le plus souvent obtenues à partir de bases de Gröbner : les b_ℓ correspondent à des points sous un escalier. Plutôt que de faire une présentation formelle de ces algorithmes, nous en donnons l'idée sur un exemple.

Prenons celui du calcul du produit des deux fonctions f et g en deux variables x et y, données par $f(x, y) = \exp(xy)$ et $g(x, y) = J_\mu(x+y)$, où, pour un paramètre μ complexe, J_μ est la fonction de Bessel de première espèce, solution de l'équation différentielle

$$z^2 J''_\mu(z) + zJ'_\mu(z) + (z^2 - \mu^2)J_\mu(z) = 0$$

qui admet à l'origine le développement asymptotique

$$J_\mu(z) \sim \frac{1}{2^\mu} \sum_{n=0}^{\infty} \frac{(-1)^n (z/2)^{2n}}{n!\,\Gamma(n + \mu + 1)}.$$

Considérons l'algèbre de Ore $A = \mathbb{C}(\mu, x, y)\langle \partial_x, \partial_y; \mathrm{id}, \mathrm{id}, D_x, D_y\rangle$, où μ est maintenant un paramètre formel. Des bases de Gröbner des annulateurs I et J dans A de f et g pour l'ordre $\mathsf{lex}(\partial_y, \partial_x)$ sont respectivement

$$\{\partial_x - y, \partial_y - x\} \qquad \text{et} \qquad \{(x+y)^2\partial_x^2 + (x+y)\partial_x + (x+y)^2 - \mu^2, \partial_y - \partial_x\}.$$

On désigne encore maintenant par f et g les vecteurs cycliques générateurs des modules A/I et A/J, avec un petit abus de notation. Pour pouvoir parler d'un produit entre ces objets a priori seulement algébriques, on introduit l'espace vectoriel sur $\mathbb{C}(\mu)$ de base $B = \big(f \otimes g, f \otimes (\partial_x \cdot g)\big)$, où l'on voit le produit $h = f \otimes g$ comme donné par ses coordonnées $(1, 0)$. (Pour mémoire, on peut voir un produit tensoriel comme un produit bilinéaire sans commutativité : en général, $u \otimes v \neq v \otimes u$, mais, pour tout $\lambda \in \mathbb{C}(\mu)$, $(\lambda u)\otimes v = u\otimes(\lambda v) = \lambda(u\otimes v)$, que l'on note donc par simplicité $\lambda u \otimes v$.) Sur cet espace de base B, qui n'est autre que $A/I \otimes A/J$, la dérivée d'un produit tensoriel est donnée par

$$\partial_x \cdot (u \otimes v) = (\partial_x \cdot u) \otimes v + u \otimes (\partial_x \cdot v),$$

ce que l'on peut résumer en $\partial_x = \partial_x \otimes \mathrm{id} + \mathrm{id} \otimes \partial_x$, pour des domaines de définitions adéquats. Cette action fait du produit tensoriel $A/I \otimes A/J$ un nouveau module sur A. Puisque

$$\partial_x \cdot (f \otimes g) = (\partial_x \cdot f) \otimes g + f \otimes (\partial_x \cdot g) = yf \otimes g + f \otimes (\partial_x \cdot g)$$

et

$$\begin{aligned}\partial_x \cdot \big(f \otimes (\partial_x \cdot g)\big) &= yf \otimes (\partial_x \cdot g) + f \otimes (\partial_x^2 \cdot g) \\ &= yf \otimes (\partial_x \cdot g) + \big((x+y)^{-2}\mu^2 - 1\big) f \otimes g - (x+y)^{-1} f \otimes (\partial_x \cdot g),\end{aligned}$$

et des relations similaires pour l'action de ∂_y, on trouve les matrices

$$M_x = \begin{pmatrix} y & 1 \\ (x+y)^{-2}\mu^2 - 1 & y - (x+y)^{-1} \end{pmatrix}$$

et

$$M_y = \begin{pmatrix} x & 1 \\ (x+y)^{-2}\mu^2 - 1 & x - (x+y)^{-1} \end{pmatrix}.$$

Choisissons d'itérer selon un ordre raffinant le degré total en ∂_x et ∂_y. On fait d'abord agir ∂_y pour trouver $\partial_y \cdot h = \big((1,0)M_y + d(1,0)/dy\big)\,{}^tB = (x,1)\,{}^tB$, qui avec $(1,0)\,{}^tB$ ne constitue pas une famille liée. De même, on trouve $\partial_x \cdot h = (y,1)\,{}^tB$, qui fournit la liaison $p_1 \cdot h = 0$ pour $p_1 = \partial_x - \partial_y + (x-y)$. Pour la suite du calcul, on exclut alors tous les monômes divisibles par ∂_x ; le monôme considéré suivant est ∂_y^2. Son action sur h donne

$$\partial_y^2 \cdot h = \big((x,1)M_y + d(x,1)/dy\big)\,{}^tB = \big((x+y)^{-2}\mu^2 + x^2 - 1, 2x - (x+y)^{-1}\big)\,{}^tB,$$

et l'on obtient un second annulateur de h,

$$p_2 = (x+y)^2\partial_y^2 - (x+y)(2x^2+2xy-1)\partial_y + (x+y)(x^3+x^2y+y) - \mu^2.$$

Pour la suite du calcul, on exclut donc tous les monômes divisibles par ∂_y^2, si bien qu'il ne reste plus aucun monôme à considérer. L'idéal annulateur de h est l'idéal $Ap_1 + Ap_2$, dont $\{p_1, p_2\}$ est une base de Gröbner pour l'ordre $\mathsf{lex}(\partial_y, \partial_x)$, de monômes de tête respectifs ∂_x et ∂_y^2 ; le module A/I est donné comme $\mathbb{C}(x,y)$-espace vectoriel par sa base $(1+I, \partial_x + I)$.

Le calcul qui précède se revisite en abandonnant l'écriture matricielle et en faisant apparaître plus explicitement les calculs de restes modulo une base de Gröbner. On récrit d'abord $\partial_y \cdot h$ sous la forme

$$\partial_y \cdot h = (\partial_y \cdot f) \otimes g + f \otimes (\partial_y \cdot g) = xf \otimes g + f \otimes (\partial_x \cdot g),$$

après réductions par les bases de Gröbner pour I et J ; ce vecteur et h constituent donc une famille linéairement indépendante. On procède ensuite de même pour $\partial_x \cdot h$, de façon à avoir

$$\partial_x \cdot h = (\partial_x \cdot f) \otimes g + f \otimes (\partial_x \cdot g) = yf \otimes g + f \otimes (\partial_x \cdot g);$$

on retrouve ainsi l'annulateur p_1. Le monôme considéré suivant est ∂_y^2, d'action sur h

$$\partial_y^2 \cdot h = x^2 f \otimes g + 2xf \otimes (\partial_x \cdot g) - (x+y)^{-2} f \otimes \big((x+y)\partial_x + (x+y)^2 - \mu^2\big)g;$$

ce vecteur, h et $\partial_y \cdot h$ forment donc une famille linéairement liée, et l'on retrouve le second annulateur p_2. Le calcul se termine de la même manière.

Pour l'algorithme d'addition, les mêmes idées algorithmiques fonctionnent en calculant dans la somme directe $A/I \oplus A/J$.

Notes

Les clôtures dans le cas à plusieurs variables ont été largement exploitées dans le traitement d'identités en fonctions spéciales à partir de [5]. L'algorithmisation de ces méthodes s'est ensuite peu à peu fait jour, d'abord dans le cas différentiel dans [4], puis dans le cas de polynômes tordus généraux dans [1], où la définition des fonctions ∂-finies a été posée. La présentation matricielle de la Section 5 nous a paru plus synthétique que la présentation classique à base de réécriture par bases de Gröbner dans [1]. Les deux présentations trouvent leur inspiration dans l'algorithme de [2] pour le changement d'ordre dans les bases de Gröbner du cadre commutatif.

Un autre cadre de bases de Gröbner est disponible pour des algèbres de Ore polynomiales, de la forme $k[x_1, \ldots, x_r]\langle \partial_1, \ldots, \partial_r; \sigma_1, \ldots, \sigma_r, \delta_1, \ldots, \delta_r\rangle$ et non $k(x_1, \ldots, x_r)\langle \partial_1, \ldots, \partial_r; \sigma_1, \ldots, \sigma_r, \delta_1, \ldots, \delta_r\rangle$. Les ordres monomiaux garantissant la terminaison d'un algorithme de Buchberger sont alors sujet à plus de contraintes. Tout ceci est raconté de façon relativement accessible dans [3], d'où est aussi tirée la Section 4.

Bibliographie

[1] Chyzak, Frédéric et Bruno Salvy (1998). « Non-commutative Elimination in Ore Algebras Proves Multivariate Holonomic Identities ». In : *Journal of Symbolic Computation*, vol. 26, n°2, p. 187–227. doi : 10.1006/jsco.1998.0207.

[2] Faugère, J. C., P. Gianni, D. Lazard et T. Mora (1993). « Efficient computation of zero-dimensional Gröbner bases by change of ordering ». In : *Journal of Symbolic Computation*, vol. 16, n°4, p. 329–344.

[3] Saito, Mutsumi, Bernd Sturmfels et Nobuki Takayama (2000). *Gröbner deformations of hypergeometric differential equations*. Springer-Verlag, viii+254 pages.

[4] Takayama, Nobuki (1992). « An approach to the zero recognition problem by Buchberger algorithm ». In : *Journal of Symbolic Computation*, vol. 14, n°2-3, p. 265–282.

[5] Zeilberger, Doron (1990). « A holonomic systems approach to special functions identities ». In : *Journal of Computational and Applied Mathematics*, vol. 32, n°3, p. 321–368. doi : 10.1016/0377-0427(90)90042-X.

CHAPITRE 30

Sommation et intégration symboliques des fonctions spéciales

Résumé

Dans ce chapitre, nous décrivons un algorithme qui peut se voir comme une extension de l'algorithme de Zeilberger pour des sommants ∂-finis, et qui traite dans le même formalisme sommation et intégration. Les quelques sommes et intégrales suivantes, que nous envisageons de traiter avec cet algorithme, montrent une variété d'applications qui vont de la combinatoire à la physique mathématique en passant par la théorie des fonctions spéciales :

$$\sum_{k=0}^{n}\left(\sum_{j=0}^{k}\binom{n}{j}\right)^3 = n2^{3n-1}+2^{3n}-3n2^{n-2}\binom{2n}{n},$$

$$\sum_{n=0}^{\infty} H_n(x)H_n(y)\frac{u^n}{n!} = \frac{\exp\left(\frac{4u(xy-u(x^2+y^2))}{1-4u^2}\right)}{\sqrt{1-4u^2}},$$

$$\frac{1}{2}J_0(x)^2+J_1(x)^2+J_2(x)^2+\cdots=\frac{1}{2},$$

$$\int_{-1}^{+1}\frac{e^{-px}T_n(x)}{\sqrt{1-x^2}}\,dx=(-1)^n\pi I_n(p),$$

$$\int_0^{+\infty} xe^{-px^2}J_n(bx)I_n(cx)\,dx=\frac{1}{2p}\exp\left(\frac{c^2-b^2}{4p}\right)J_n\left(\frac{bc}{2p}\right),$$

$$\int_0^{+\infty} xJ_1(ax)I_1(ax)Y_0(x)K_0(x)\,dx=-\frac{\ln(1-a^4)}{2\pi a^2},$$

$$\sum_{k=0}^{n}\frac{q^{k^2}}{(q;q)_k(q;q)_{n-k}}=\sum_{k=-n}^{n}\frac{(-1)^kq^{(5k^2-k)/2}}{(q;q)_{n-k}(q;q)_{n+k}}.$$

Ici, J, Y, I et K sont des variantes de fonctions de Bessel, qui apparaissent fréquemment pour décrire des modèles physiques à symétrie cylindrique ou sphérique ; H et T sont des familles de polynômes orthogonaux de Hermite et Tchébichev ; $(q;q)_n$ représente le produit $(1-q)\cdots(1-q^n)$. La première identité intervient dans une discrétisation d'une question de probabilités sur la position du maximum de trois variables aléatoires gaussiennes ; la dernière est une variante finie d'une des identités de Rogers–Ramanujan, en théorie des partitions.

1. Expression du télescopage créatif en termes d'algèbres de Ore rationnelles

On a déjà exposé dans ce cours la méthode du télescopage créatif, en l'appliquant à la sommation hypergéométrique définie par une combinaison de l'algorithme de Gosper et d'une idée due à Zeilberger. Cette approche se généralise en des algorithmes de sommation et intégration pour les suites et fonctions ∂-finies.

Rappelons le principe de la méthode. Soit à évaluer une somme paramétrée

$$F_n = \sum_{k=a}^{b} f_{n,k}.$$

En toute généralité, le principe du télescopage créatif est de déterminer une suite auxiliaire $g = (g_{n,k})$ ainsi que des coefficients η_0, …, η_r, fonctions de la variable n, tels que se trouve vérifiée la relation

$$\eta_r(n) f_{n+r,k} + \cdots + \eta_0(n) f_{n,k} = g_{n,k+1} - g_{n,k}.$$

Ici, nous ne faisons pas plus d'hypothèses sur les η_i et g que celle de pouvoir évaluer la relation ci-dessus pour tout n quand k décrit les entiers de a à b. Dans ce cas, une sommation sur k fournit l'égalité

$$\eta_r(n) F_{n+r} + \cdots + \eta_0(n) F_n = g_{n,b+1} - g_{n,a}.$$

Si le membre de droite n'est pas déjà nul, on recherche un opérateur annulateur de ce second membre ; par composition, on obtient une récurrence homogène sur F. Dans bien des cas, on sait prédire à partir de conditions analytiques sur f la nullité du terme $g_{n,b+1} - g_{n,a}$.

Des algorithmes d'efficacités différentes ont été donnés selon le domaine de recherche des η_i et de g, et selon le compromis choisi entre efficacité et richesse de la classe de suites f en entrée. En particulier, l'algorithme de Zeilberger, optimisé pour une suite f hypergéométrique, revient à rechercher des η_i polynomiaux et une suite g similaire à f, c'est-à-dire un multiple ϕf pour une fraction rationnelle ϕ en n et k. La suite $g = \phi f$ devant être une somme indéfinie, la recherche de ϕ et des η_i se fait par une variante paramétrée de l'algorithme de Gosper. Notons que le domaine de recherche de g est l'espace vectoriel $\mathbb{C}(n,k)f$, qui n'est autre, dans le cas hypergéométrique, que le module engendré par f sur l'algèbre de Ore $A = \mathbb{C}(n,k)\langle \partial_n, \partial_k; S_n, S_k\rangle$. Nous considérons ici la généralisation au cas où f est une fonction ∂-finie et où le module $A \cdot f$ est un espace vectoriel de dimension finie sur $\mathbb{C}(n,k)$, mais pas forcément de dimension 1. Soit v_1, …, v_d les éléments d'une base vectorielle de $A \cdot f$; l'algorithme de Zeilberger étendu recherche g sous la forme indéterminée $\phi_1 v_1 + \cdots + \phi_d v_d$, pour des fractions rationnelles ϕ_i en n et k. Cette recherche se fait par une extension ∂-finie de la variante paramétrée de l'algorithme de Gosper.

Tout ce qui a été dit s'étend au monde différentiel pour l'évaluation d'une intégrale paramétrée

$$F(x) = \int_a^b f(x,y)\,dy.$$

On cherche alors une relation

$$\eta_r(x) \frac{\partial^r f}{dx^r}(x,y) + \cdots + \eta_0(n) f(x,y) = \frac{\partial g}{dy}(x,y),$$

qui après intégration fournit l'égalité

$$\eta_r(n) F^{(r)}(x) + \cdots + \eta_0(n) F(x) = \int_a^b g(x,y)\,dy.$$

La même méthode permet aussi de traiter des sommations paramétrées continûment,

$$F(x) = \sum_{k=a}^{b} f_k(x),$$

et des suites d'intégrales de la forme

$$F_n = \int_a^b f_n(y)\,dy.$$

EXERCICE 1. Formuler la relation entre f et g à rechercher dans ces deux derniers cas.

2. L'algorithme sur l'exemple $\frac{1}{2}J_0(x)^2 + J_1(x)^2 + J_2(x)^2 + \cdots = \frac{1}{2}$

Nous allons montrer que la famille paramétrée des fonctions de Bessel de première espèce, J_ν, où chaque J_ν est une solution que nous allons préciser de l'équation de Bessel

$$x^2y''(x) + xy'(x) + (x^2 - \nu^2)y(x) = 0,$$

a une somme $\frac{1}{2}J_0(x)^2 + J_1(x)^2 + J_2(x)^2 + \cdots$ qui s'évalue à $\frac{1}{2}$.

L'équation de Bessel et les fonctions de Bessel peuvent être considérées pour des valeurs complexes du paramètre ν, mais vu la nature de la somme à étudier, nous nous limiterons dorénavant à des valeurs entières $\nu \in \mathbb{N}$. En étudiant l'équation indicielle de l'équation de Bessel, on s'aperçoit qu'il existe pour chaque ν des solutions dans les séries formelles $\mathbb{C}[[x]]$ et que ces solutions constituent un espace vectoriel de dimension 1 sur $\mathbb{C}$ de séries. Une base de ces solutions formelles est donnée par la série de Bessel

$$J_\nu(x) = (x/2)^\nu \sum_{n=0}^{\infty} \frac{(-1)^n (x/2)^{2n}}{n!\,(n+\nu)!},$$

de valuation ν, qui vu la décroissance de ses coefficients est pour chaque entier ν une série entière.

EXERCICE 2. Vérifier ces résultats.

On vérifie par simple substitution et évaluation que ces fonctions J_ν satisfont aussi aux relations

$$xJ'_\nu(x) + xJ_{\nu+1}(x) - \nu J_\nu(x) = 0 \qquad \text{et} \qquad xJ_{\nu+2}(x) - 2(\nu+1)J_{\nu+1}(x) + xJ_\nu(x) = 0.$$

En introduisant l'algèbre de Ore $A = \mathbb{C}(\nu, x)\langle \partial_\nu, \partial_x; S_\nu, \mathrm{id}, 0, D_x\rangle$ où S_ν est le décalage avant sur ν et D_x est la dérivation par rapport à x, on a donc un système d'annulateurs pour J,

$$\begin{aligned} p_1 &= x^2\partial_x^2 + x\partial_x + x^2 - \nu^2, \\ p_2 &= x\partial_x + x\partial_\nu - \nu, \\ p_3 &= x\partial_\nu^2 - 2(\nu+1)\partial_\nu + x. \end{aligned}$$

Les deux premiers forment une base de Gröbner de l'idéal engendré pour l'ordre $\mathsf{lex}(\partial_\nu, \partial_x)$; les deux derniers pour l'ordre $\mathsf{lex}(\partial_x, \partial_\nu)$.

EXERCICE 3. Pour chacun des idéaux $Ap_1 + Ap_2$ et $Ap_2 + Ap_3$, calculer la base de Gröbner minimale réduite pour chacun des deux ordres $\mathsf{lex}(\partial_\nu, \partial_x)$ et $\mathsf{lex}(\partial_x, \partial_\nu)$.

Bien évidemment, J est une fonction ∂-finie. Le module $A \cdot J$ est donné, par exemple, comme l'espace vectoriel sur $\mathbb{C}(\nu, x)$ de base $(J, \partial_\nu \cdot J)$. Pour représenter le carré de J en vue d'une sommation, on peut observer que, en tant qu'espace vectoriel, le module $A \cdot J^2$ admet la base $\big(J^2, J \times (\partial_\nu \cdot J), (\partial_\nu \cdot J)^2\big)$ et utiliser l'algorithme de clôture par produit pour obtenir une base de Gröbner. En fait, le calcul qui suit n'a même pas besoin d'une représentation aussi explicite de $f = J^2$: pour calculer la somme $\frac{1}{2}J_0(x)^2 + J_1(x)^2 + J_2(x)^2 + \cdots$ comme fonction de x, on recherche une fonction η de x, indépendante de ν, telle que $f' + \eta f$ soit la différence

finie en ν d'un élément g de $A \cdot J^2$. Pour la suite du calcul, nous fixons cet élément sous la forme indéterminée donnée par

$$g(\nu) = \phi_0(\nu)J_\nu^2 + \phi_1(\nu)J_{\nu+1}^2 + \phi_2(\nu)J_\nu J_{\nu+1},$$

où nous avons omis de faire référence à la variable x dans les évaluations de g, des ϕ_i et de J, car cette variable ne va intervenir que comme paramètre dans le calcul des fractions rationnelles ϕ_i. (On peut penser qu'on travaille temporairement dans l'algèbre de Ore $A' = \mathbb{C}(\nu, x)\langle\partial_\nu; S_\nu\rangle$.)

En supposant le problème résolu, on a alors par construction la relation $f' + \eta f = (\partial_\nu - 1)\cdot g$, puis, après réduction de chaque occurence des dérivées et décalées de J par la base de Gröbner $\{p_2, p_3\}$,

$$\begin{aligned}
&2J_\nu J_\nu' + \eta J_\nu^2 = (\partial_\nu - 1)\cdot\big(\phi_0(\nu)J_\nu^2 + \phi_1(\nu)J_{\nu+1}^2 + \phi_2(\nu)J_\nu J_{\nu+1}\big)\\
&= \phi_0(\nu+1)J_{\nu+1}^2 - \phi_0(\nu)J_\nu^2 + \phi_1(\nu+1)x^{-2}\big(2(\nu+1)J_{\nu+1} - xJ_\nu\big)^2 - \phi_1(\nu)J_{\nu+1}^2\\
&\qquad + \phi_2(\nu+1)x^{-1}J_{\nu+1}\big(2(\nu+1)J_{\nu+1} - xJ_\nu\big) - \phi_2(\nu)J_\nu J_{\nu+1},
\end{aligned}$$

laquelle se récrit

$$\begin{aligned}
&\big(2\nu x^{-1} + \eta\big)J_\nu^2 - 2J_\nu J_{\nu+1}\\
&= \big(\phi_1(\nu+1) - \phi_0(\nu)\big)J_\nu^2 - \big(4(\nu+1)x^{-1}\phi_1(\nu+1) + \phi_2(\nu+1) + \phi_2(\nu)\big)J_\nu J_{\nu+1}\\
&\quad + \big(\phi_0(\nu+1) + 4(\nu+1)^2x^{-2}\phi_1(\nu+1) - \phi_1(\nu) + 2(\nu+1)x^{-1}\phi_2(\nu+1)\big)J_{\nu+1}^2.
\end{aligned}$$

De l'indépendance linéaire des fonctions J_ν^2, $J_{\nu+1}^2$ et $J_\nu J_{\nu+1}$ sur $\mathbb{C}(\nu, x)$, on déduit les relations nécessaires

$$\begin{aligned}
-\phi_0(\nu) + \phi_1(\nu+1) &= 2\nu x^{-1} + \eta,\\
4(\nu+1)x^{-1}\phi_1(\nu+1) + \phi_2(\nu) + \phi_2(\nu+1) &= 2,\\
\phi_0(\nu+1) - \phi_1(\nu) + 4(\nu+1)^2x^{-2}\phi_1(\nu+1) + 2(\nu+1)x^{-1}\phi_2(\nu+1) &= 0.
\end{aligned}$$

En résolvant les deux premières respectivement en ϕ_0 et en ϕ_1, puis en substituant dans la dernière, on trouve la récurrence

$$\begin{aligned}
&x^2(\nu+1)\phi_2(\nu+3) - (\nu+1)(4\nu^2 + 20\nu + 24 - x^2)\phi_2(\nu+2)\\
&+(\nu+3)(4\nu^2+12\nu+8-x^2)\phi_2(\nu+1) - x^2(\nu+3)\phi_2(\nu) = -4x(\eta\nu^2+4\eta\nu+3\eta+x).
\end{aligned}$$

Nous résolvons maintenant celle-ci en ses solutions rationnelles par l'algorithme d'Abramov, dans sa variante paramétrée qui résoud en $\phi_2 \in \mathbb{C}(n, \nu)$ et simultanément en $\eta \in \mathbb{C}$. Les coefficients extrêmes de la partie homogène indiquent que toute solution rationnelle doit être polynomiale, puisque le p. g. c. d. entre $(\nu - 3) + 1$ et $\nu + 3$ vaut 1. La mise sous forme de différences finies de la partie homogène indique que l'opérateur associé accroît de 2 le degré d'un polynôme. Le degré de la partie inhomogène étant 2, toute solution rationnelle ne peut être qu'une constante. On trouve $\phi_2 = 1$ et $\eta = 0$, d'où après report $\phi_1 = 0$ et $\phi_0 = -2\nu/x$. Autrement dit, on a

$$\partial_x \cdot J_\nu^2 = (\partial_\nu - 1)\cdot\big(J_\nu J_{\nu+1} - 2\nu x^{-1}J_\nu^2\big),$$

qui par sommation fournit

$$\partial_x \cdot \sum_{\nu=0}^{N} J_\nu^2 = J_{N+1}\big(J_{N+2} - 2(N+1)x^{-1}J_{N+1}\big) - J_0J_1.$$

Comme la série $J_N \in \mathbb{C}[[x]]$ a valuation N, le membre droit tend vers $-J_0J_1 = \frac{1}{2}\partial_x \cdot J_0^2$ quand N tend vers ∞ pour la topologie usuelle donnée par la métrique $|s| = 2^{-v}$ pour toute série non nulle s de valuation v. On a donc

$$\partial_x \cdot \big(\tfrac{1}{2}J_0(x)^2 + J_1(x)^2 + J_2(x)^2 + \cdots\big) = 0$$

qui caractérise la somme par une condition initiale. Une simple évaluation en 0 montre que la somme vaut $\frac{1}{2}$, ce qui achève la preuve de l'identité annoncée.

3. Bases de Gröbner de modules et découplage de systèmes

Dans l'exemple qui précède, on a pour le moment effectué le découplage « à la main », mais un procédé systématique et automatique est disponible, par le biais des bases de Gröbner de modules, qui généralisent la notion de base de Gröbner pour les idéaux. Cette notion existe tant dans le domaine des polynômes commutatifs que dans le cadre non commutatif des algèbres de Ore ; nous la présentons directement dans ce second cas.

Dans le cas d'un idéal I d'une algèbre de Ore A, les éléments de I s'interprètent comme autant d'équations vérifiées par une fonction inconnue ϕ. Dans une perspective algorithmique, chaque idéal est engendré par un nombre fini de générateurs. Une question naturelle est celle de systèmes linéaires sur un vecteur de fonctions inconnues $(\phi_1, \dots, \phi_d)$, à coefficients dans A. On considère des systèmes d'un nombre fini d'équations de la forme $g_i = g_{i,1} \cdot \phi_1 + \cdots + g_{i,d} \cdot \phi_d$ pour des polynômes tordus $g_{i,j}$ de A. Un tel système se représente de façon compacte par une matrice $(g_{i,j})$ à entrées dans A. Les questions qui sont alors naturelles sont celle de l'algèbre linéaire pour ces matrices, dont en particulier celle de donner un algorithme du pivot de Gauss pour des coefficients dans A, c'est-à-dire non plus dans un corps, mais dans un anneau, et non commutatif de surcroît.

Pour ce faire, au lieu de considérer simplement un ordre monomial sur les monômes ∂^a d'une algèbre de Ore $A = k(x)\langle\partial; \sigma, \delta\rangle$, pour lequel tout idéal à gauche admet une base de Gröbner, on s'intéresse plus généralement à un module libre de rang fini sur A, donné par une base sous la forme $A^d = Ae_1 + \cdots + Ae_d$ et muni d'un ordre sur les $\partial^a e_i$, dans lequel on va étendre la notion de base de Gröbner pour des sous-modules à gauche de A^d. La notion d'ordre monomial conserve formellement la même définition, si ce n'est que les e_i ne peuvent apparaître que linéairement dans les monômes $\partial^a e_i$ et qu'ils ne peuvent servir pour des multiplications à gauche. La notion de S-polynôme s'étend aussi mot pour mot, à ceci près que deux polynômes de monômes de tête $\partial^a e_i$ et $\partial^b e_j$ ont un S-polynôme nul dès lors que i et j sont différents. Les définitions et caractérisations équivalentes des bases de Gröbner d'idéaux restent alors valables pour les sous-modules du module libre A^d. L'algorithme de Buchberger, modifié pour suivre ces nouvelles définitions, termine sur tout sous-module en fournissant une base de Gröbner. Pour certains ordres, ce calcul correspond à l'algorithme de Gauss.

Un point de vue presque équivalent, mais qui donne une variante des calculs avec un peu plus de réductions, est que le calcul est celui d'une base de Gröbner dans l'anneau $A[e_1, \dots, e_d]$ des polynômes en les indéterminées commutatives e_i à coefficients dans l'anneau A pour l'idéal à gauche engendré par les g_i initiaux et tous les produits $e_i e_j = 0$.

Reprenons l'exemple du découplage des relations entre les coordonnées ϕ_i donnant g dans la section précédente sur la somme des carrés fonctions de Bessel. Ces relations se recodent par les éléments

$$\begin{aligned}
g_1 &:= -e_0 + \partial_\nu e_1 - (2\nu x^{-1} + \eta)e_3,\\
g_2 &:= 4(\nu+1)x^{-1}\partial_\nu e_1 + (\partial_\nu + 1)e_2 - 2e_3,\\
g_3 &:= \partial_\nu e_0 + \big(4(\nu+1)^2 x^{-2}\partial_\nu - 1\big)e_1 + 2(\nu+1)x^{-1}\partial_\nu e_2,\\
g_4 &:= (\partial_\nu - 1)e_3,
\end{aligned}$$

du module libre A^4 pour l'algèbre de Ore $A = \mathbb{C}(n,k)\langle\partial_n, \partial_k; S_n, S_k\rangle$. Ici, chaque e_i représente la fonction rationnelle inconnue ϕ_i, et l'on a astucieusement représenté les

second membres de équations inhomogènes d'origine comme multiple d'une nouvelle inconnue représentée par e_3 et contrainte par g_4 à être constante.

Le découplage effectué dans la section précédente revient au calcul d'une base de Gröbner pour l'ordre $\mathsf{lex}(e_0, e_1, e_2, e_3, \partial_\nu)$. Les monômes de tête respectifs des g_i sont e_0, $\partial_\nu e_1$, $\partial_\nu e_0$ et $\partial_\nu e_3$, si bien que le seul S-polynôme non nul est $\mathrm{Spoly}(g_1, g_3)$. Il est donné par

$$\begin{aligned}\mathrm{Spoly}(g_1, g_3) &= \partial_\nu g_1 + g_3\\ &= \big(\partial_\nu^2 + 4(\nu+1)^2 x^{-2}\partial_\nu - 1\big)e_1 + 2(\nu+1)x^{-1}\partial_\nu e_2 - \big(2(\nu+1)x^{-1} + \eta\big)\partial_\nu e_3.\end{aligned}$$

Après réductions par g_2 et g_3, ce polynôme devient

$$g_5 = -e_1 - \left(\frac{x}{4(\nu+2)}\partial_\nu^2 + \frac{x^2 - 4\nu^2 - 12\nu - 8}{4x(\nu+2)}\partial_\nu + \frac{\nu+1}{x}\right)e_2 + \left(\frac{x}{2(\nu+2)} - \eta\right)e_3,$$

qui est adjoint à la base de Gröbner en cours de calcul. L'unique nouvel S-polynôme à considérer est celui entre ce g_5 et g_2, qui est $\mathrm{Spoly}(g_2, g_4) = g_2 + 4(\nu+1)x^{-1}\partial_\nu g_5$ et a pour monôme de tête $\partial_\nu^3 e_2$. Après réduction par e_3 et renormalisation, le dernier polynôme introduit dans la base de Gröbner est

$$\begin{aligned}\big((\nu+1)x^2\partial_\nu^3 + (\nu+1)(x^2 - 4\nu^2 - 20\nu - 24)\partial_\nu^2 &\\ - (\nu+3)(x^2 - 4\nu^2 - 12\nu - 8)\partial_\nu - (\nu+3)x^2\big)e_2 &\\ + 4\big((\nu^2 + 4\nu + 3)\eta + x\big)xe_3.&\end{aligned}$$

Ce polynôme n'est autre qu'un recodage de l'équation inhomogène du troisième ordre qui a permis de déterminer ϕ_2 dans la section précédente.

Bibliographie

[1] Chyzak, Frédéric (2000). « An Extension of Zeilberger's Fast Algorithm to General Holonomic Functions ». In : *Discrete Mathematics*, vol. 217, n°1-3, p. 115–134.

Septième partie

Compléments

CHAPITRE 31

Réduction de réseaux et algorithme LLL

Résumé

La recherche de vecteurs courts dans des réseaux de vecteurs à coefficients entiers permet, en calcul formel, la factorisation en temps polynomial et la recherche de dépendances linéaires entre constantes réelles données par des approximations, et, dans des domaines plus proches de la cryptanalyse, la mise en évidence de faiblesses de cryptosystèmes et de générateurs pseudo-aléatoires. Nous présentons l'algorithme LLL maintenant célèbre pour la réduction de réseaux et analysons sa complexité.

Ce texte suit d'assez près mais dans un autre ordre les chapitres 16 et 17 du livre *Modern computer algebra* de von zur Gathen et Gerhard.

1. Réseaux, vecteurs courts et résultats principaux

On appelle *réseau* de $\mathbb{Z}^n$ un $\mathbb{Z}$-module $\sum_{i=1}^{n} \mathbb{Z}v_i$ engendré par des vecteurs v_i linéairement indépendants sur $\mathbb{Z}$. Le thème de ce cours est la recherche de vecteurs « courts » dans un réseau donné : par exemple, le réseau engendré par les vecteurs $(12, 2)$ et $(13, 4)$ contient le vecteur plus court $(1, 2)$. Ici, court s'entend pour la norme euclidienne : un vecteur $v = (v_1, \ldots, v_n)$ a pour norme $\|v\| = (v_1^2 + \cdots + v_n^2)^{1/2}$.

Cette question est motivée par un certain nombre d'applications :

- factorisation en temps polynomial ;
- cassage d'un cryptosystème basé sur le problème du sac-à-dos ;
- mise en évidence de la faiblesse de générateurs pseudo-aléatoires ;
- recherche de dépendances linéaires entre nombres donnés par des approximations numériques, dont la recherche du polynôme minimal d'un nombre algébrique.

La deuxième et la dernière de ces applications sont détaillées dans la section qui suit.

La recherche d'un vecteur de norme minimale dans un réseau donné est un problème difficile, en fait, même NP-difficile. Il est donc naturel de relâcher la contrainte de minimalité et de considérer le problème de la recherche approchée à un facteur constant près, mais le problème de la recherche à un facteur $\sqrt{2}$ près reste NP-difficile. En revanche, le problème redevient polynomial si on autorise le facteur d'approximation à croître avec la dimension n du réseau.

De façon précise, on décrit dans la suite l'algorithme LLL, qui pour une base $(v_1, \ldots, v_n)$ d'un réseau L renvoie en particulier un vecteur u approximant les plus courts vecteur du réseau à pas plus d'un facteur $2^{(n-1)/2}$ près :

$$\|u\| \leq 2^{(n-1)/2} \min\{\, \|f\| : f \in L,\ f \neq 0 \,\}.$$

Cet algorithme termine après $O(n^4 \log A)$ opérations arithmétiques correspondant à $O_{\log}(n^5 \log^2 A)$ opérations binaires, où la constante A borne chacun des v_i. (Seule la complexité arithmétique est démontrée dans ce qui suit.)

2. Applications

2.1. Cassage du cryptosystème de Merkle et Hellman. Merkle et Hellman ont proposé en 1978 un cryptosystème basé sur le problème du sac-à-dos. Ce cryptosystème devait nécessiter des calculs moins lourds que le célèbre système RSA. Néanmoins, une faiblesse du système reposait sur l'existence d'un vecteur court dans un réseau.

L'idée de Merkle et Hellman est la suivante. Bob se dote d'une clé secrète, une famille d'entiers positifs b_1, …, b_n, telle que chaque b_i soit supérieur à la somme des précédents. Bob se donne aussi un multiplicateur c et un module m, eux deux aussi secrets. Il publie la clé privée constituée des restes positifs a_i de cb_i modulo m. Quand Alice veut lui envoyer le message constitué de la suite de bits $(x_1, \ldots, x_n)$, elle construit la somme $s = x_1a_1 + \cdots + x_na_n$ qui constitue le message crypté. Bob n'a plus qu'à multiplier par l'inverse de c modulo m et à tester si les b_i sont dans la somme restante (en commençant par les plus grands) pour déterminer les x_i.

Le problème sur lequel s'appuie ce cryptosystème, celui du sac-à-dos, consiste précisément à décider l'existence des $x_i \in \{0, 1\}$ tels que $s = x_1a_1 + \cdots + x_na_n$. Ce problème est NP-complet en l'absence d'hypothèse sur la famille des a_i. Mais dans le cas présent, l'origine des a_i crée une faiblesse cryptographique. Considérons les vecteurs $(0, \ldots, 0, 1, 0, \ldots, 0, -a_i) \in \mathbb{Z}^{n+1}$, où 1 est en i^{e} position, ainsi que le vecteur $(0, \ldots, 0, s)$. Tous ces vecteurs sont « longs », puisque de l'ordre de m, mais le réseau qu'ils engendrent contient le vecteur $(x_1, \ldots, x_n, 0)$, lequel est manifestement très court. L'algorithme LLL qui va suivre permet de le retrouver en complexité polynomiale.

2.2. Relations de dépendance entre constantes numériques. Étant donnés n nombres réels non nuls, r_1, …, r_n, une relation de dépendance linéaire à coefficients entiers entre ces réels est une relation de la forme

$$c_1r_1 + \cdots + c_nr_n = 0$$

pour des coefficients c_i entiers non tous nuls. Lorsqu'on se donne une approximation rationnelle de chaque réel r_i, ou mieux, la possibilité d'obtenir autant de chiffres décimaux que souhaité, la recherche des vecteurs courts dans un réseau permet la détermination de relations de dépendance linéaire entre les r_i. Pour cela, on considère les combinaisons linéaires à coefficients entiers des vecteurs lignes de la matrice

$$F = \begin{bmatrix} 1 & 0 & \ldots & 0 & a_1 \\ 0 & 1 & \ldots & 0 & a_2 \\ & & \ddots & & \vdots \\ 0 & \ldots & 0 & 1 & a_{n-1} \\ 0 & \ldots & 0 & 0 & a_n \end{bmatrix},$$

où chaque a_i est une troncature entière de Nr_i pour un grand entier N. (Par exemple, N est une grande puissance de 10.)

Un vecteur court est alors de la forme

$$v = (c_1, \ldots, c_{n-1}, c_1a_1 + \cdots + c_na_n) \simeq \big(c_1, \ldots, c_{n-1}, N(c_1r_1 + \cdots + c_nr_n)\big).$$

En particulier,

$$|c_1r_1 + \cdots + c_nr_n| \simeq \big|N^{-1}(c_1a_1 + \cdots + c_na_n)\big| \leq N^{-1}|v|$$

se doit d'être petit, ce qui fait de

$$c_1r_1 + \cdots + c_nr_n = 0$$

un bon candidat pour une relation de dépendance entre les r_i. Bien qu'un tel argument ne constitue pas une preuve, la preuve formelle de relation entre constantes a

dans un bon nombre de cas été grandement facilitée une fois que la bonne relation a pu être découverte expérimentalement par la méthode proposée ci-dessus. Dans un certain nombre de cas, même, des identités n'ont pu être prouvées que parce qu'elles avaient été découvertes heuristiquement en utilisant l'algorithme LLL.

Donnons un exemple. Des arguments théoriques donnent la certitude que le nombre

$$V = \int_0^\infty \frac{\sqrt{x}\,\ln^5 x}{(1-x)^5}\,\mathrm{d}x$$

peut se représenter comme l'évaluation en π d'une fonction polynomiale à coefficients rationnels. Il s'agit donc de trouver une dépendance linéaire entre

$$V, 1, \pi, \ldots, \pi^d$$

pour un degré de polynôme d à déterminer. Tout système de calcul formel généraliste connait des approximations pour π et permet de calculer aisément une approximation numérique de V. On prend donc par exemple $a_1 = N = 10^{25}$, $a_2 = 31415926535897932384626434 \simeq N\pi$, …, $a_9 = 94885310160705740071285755038 \simeq N\pi^8$, $a_{10} = -16699473719229070496187243 3 \simeq NV$ pour construire la matrice F. Les normes des vecteurs lignes de cette matrice sont toutes supérieures à $N = 10^{25}$. Par l'algorithme LLL, on trouve une base du même réseau de vecteurs, dont le premier vecteur ligne,

$$v = (c_1, \ldots, c_{n-1}, c_1a_1 + \cdots + c_na_n) = (0, 0, 120, 0, 140, 0, -15, 0, 0, 33),$$

est de norme inférieure à 200, tous les autres vecteurs de base étant de norme supérieure à 400. Les a_i étant connus, on trouve

$$(c_1, \ldots, c_n) = (0, 0, 120, 0, 140, 0, -15, 0, 0, 24),$$

d'où la relation linéaire

$$V = \int_0^\infty \frac{\sqrt{x}\,\ln^5 x}{(1-x)^5}\,\mathrm{d}x = \frac{5\pi^2}{24}(3\pi^4 - 28\pi^2 - 24)$$

qui résout le problème posé.

2.3. Polynôme minimal de nombres algébriques. Un nombre complexe α est dit algébrique (sur les nombres rationnels) lorsqu'il est solution d'un polynôme P à coefficients rationnels

$$P(\alpha) = p_0 + \cdots + p_d\alpha^d = 0.$$

Pour pouvoir utiliser l'approche de la section précédente, on se limite au cas de nombres réels.

Étant donnée une approximation numérique (réelle) d'un nombre α qu'on a de bonnes raisons de penser être algébrique, la détermination heuristique de P peut se voir comme la recherche d'une relation de dépendance linéaire sur des approximations numériques rationnelles de

$$1, \alpha, \ldots, \alpha^d.$$

Dès lors qu'on a une borne supérieure sur le degré d de P, on peut donc employer la méthode de la section précédente.

Par exemple, soit à déterminer si un nombre

$$r \simeq 0{,}266252646290196114530247765575844548176501286103 95\ldots$$

est algébrique. Par la méthode esquissée, en testant jusqu'à $d = 6$ et pour $N = 10^{28}$, on trouve (à partir de vecteurs lignes de norme supérieure à 10^{20}), le vecteur court

$$v = (-1, 0, 0, 54, 0, 0, -10),$$

de norme inférieure à 55, tous les autres vecteurs calculés étant de norme supérieure à 5000. En poursuivant avec la méthode, on trouve

$$(c_1, \dots, c_7) = (-1, 0, 0, 54, 0, 0, -54),$$

soit

$$P = 54X^6 - 54X^3 + 1.$$

En effet, le nombre r s'avère être une approximation du nombre algébrique

$$\alpha = \sqrt[3]{\frac{1}{2} - \frac{5\sqrt{3}}{18}}.$$

3. Le procédé d'orthogonalisation de Gram–Schmidt

L'algorithme LLL aura fort à voir avec le procédé d'orthogonalisation de Gram–Schmidt qui, partant d'une base $(f_1, \dots, f_n)$ d'un espace vectoriel sur $\mathbb{Q}$, calcule une base orthogonale du même espace vectoriel.

Le contexte de cette méthode est celui de l'espace vectoriel $\mathbb{Q}^n$ muni du produit scalaire euclidien $(a, b) \mapsto (a|b) = a_1b_1 + \cdots + a_nb_n$ qui a servi à définir la norme euclidienne par $\|a\|^2 = (a|a)$. Rappelons que les vecteurs a et b sont dits orthogonaux lorsque leur produit scalaire est nul et que l'orthogonal d'un ensemble $S \subset \mathbb{Q}^n$ est l'ensemble noté $S^\perp$ des vecteurs orthogonaux à tous les éléments de S. On introduit les espaces vectoriels emboîtés $U_i = \mathbb{Q}f_1 \oplus \cdots \oplus \mathbb{Q}f_i$. La projection orthogonale sur U_i est le projecteur linéaire sur U_i parallèlement à $U_i^\perp$. On définit alors f_i^* comme la projection orthogonale de f_i sur $U_i^\perp$ (donc parallèlement à ${U_i^\perp}^\perp$, qui n'est autre que U_i).

Les considérations qui précèdent définissent uniquement les f_i^* à partir des f_i et des U_i ; pour le calcul, on détermine les f_i^* pour des i successifs par les formules

$$f_i^* = f_i - \sum_{j=1}^{i-1} \mu_{i,j} f_j^* \qquad \text{où} \qquad \mu_{i,j} = \frac{(f_i|f_j^*)}{(f_j^*|f_j^*)},$$

où $\mu_{i,j}$ est ajusté à l'unique valeur convenable pour avoir orthogonalité entre f_i^* et f_j^* quand $i > j$.

Après avoir posé $\mu_{i,i} = 1$ et $\mu_{i,j} = 0$ quand $i < j$, on obtient une matrice

$$M = (\mu_{i,j}) = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ \mu_{i,j} & & 1 \end{bmatrix}.$$

vérifiant la relation

$$M \begin{bmatrix} f_1^* \\ \vdots \\ f_n^* \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}.$$

En particulier, les espaces vectoriels sur $\mathbb{Q}$ engendrés par les f_i d'une part et par les f_i^* d'autre part sont les mêmes.

Soit maintenant un vecteur non nul f du réseau engendré par les f_i, qui s'écrit donc $f = \lambda_1 f_1 + \cdots + \lambda_n f_n$ pour des entiers λ_i non tous nuls. En passant au carré de la norme et en utilisant le théorème de Pythagore, on a

$$\|f\|^2 = \sum_{i=1}^{n} \lambda_i^2 \|f_i^*\|^2 \geq \|f_k^*\|^2 \geq \left(\min_{j=1}^{n} \|f_j^*\| \right)^2$$

dès lors que $\lambda_k \neq 0$, car alors $\lambda_k^2 \geq 1$.

À ce stade, nous aurions trouvé des vecteurs parmi les plus courts, si ce n'est que les f_i^* sont généralement éléments de l'espace vectoriel engendré par les f_i (avec

des coefficients rationnels), mais hors du réseau engendré par les même f_i (avec des coefficients entiers). Il n'en reste pas moins que l'algorithme LLL qui va suivre s'appuie sur l'orthogonalisation de Gram–Schmidt de façon à contrôler des transformations de la base du réseau, car, à un niveau intuitif, plus une base de réseau est « orthogonale », plus elle est à même de contenir un vecteur court. Cette heuristique se justifie par la notion de déterminant d'un réseau : étant donné une base $(f_1, \ldots, f_n)$ d'un réseau, le déterminant de cette famille, c'est-à-dire le déterminant de la matrice $F = (f_{i,j})$ obtenue en plaçant les vecteurs lignes $f_i = (f_{i,1}, \ldots, f_{i,n})$ les uns au dessus des autres, est, au signe près, invariant par changement de base. En effet, soit $(g_1, \ldots, g_n)$ une autre base du même réseau et $G = (g_{i,j})$ la matrice associée, avec des notations évidentes ; il existe alors une matrice U à coefficients entiers, admettant un inverse à coefficients entiers, telle que $F = UG$. En passant aux déterminants, on a que $\det U$ vaut 1 au signe près, donc que F et G ont même déterminant au signe près. Toujours au signe près, ce déterminant n'est autre que le volume du parallèloïde construit en s'appuyant sur les f_i. Les côtés de ce parallèloïde sont d'autant plus courts que ce parallèloïde est orthogonal.

Une base $(f_1, \ldots, f_n)$ d'un réseau est dite *réduite* lorsque les images f_i^* des f_i par le procédé d'orthogonalisation de Gram–Schmidt ont la propriété que $\|f_i^*\|^2 \leq 2\|f_{i+1}^*\|^2$ pour tout i entre 1 et $n-1$. En particulier, f_i^* n'est dans ce cas pas plus de $2^{(i-1)/2}$ fois plus petit que f_1^*, c'est-à-dire que f_1. Il s'ensuit que pour tout vecteur non nul f du réseau, les inégalités suivantes sont vérifiées :

$$\|f\| \geq \min_{j=1}^{n} \|f_j^*\| \geq \min_{j=1}^{n} 2^{-(j-1)/2}\|f_1\| \geq 2^{-(n-1)/2}\|f_1\|.$$

Autrement dit, le premier élément d'une base réduite est un vecteur court, au sens où pour tout f non nul du réseau

$$\|f_1\| \leq 2^{(n-1)/2}\|f\|.$$

4. L'algorithme LLL

L'algorithme qui suit a été introduit par Lenstra, Lenstra et Lovász en 1982 dans l'objectif de réaliser la factorisation de polynômes d'une variable à coefficients entiers en complexité arithmétique polynomiale. Pour un réel x, on note $\lceil x \rfloor$ l'entier de plus proche de x (par défaut, l'entier immédiatement inférieur si x est un demi-entier).

Algorithme : LLL
Entrée : f_1, …, f_n, des vecteurs de $\mathbb{Z}^n$ engendrant un réseau
Sortie : g_1, …, g_n, des vecteurs de $\mathbb{Z}^n$ constituant une base réduite du même réseau
Complexité : $O(n^4 \log A)$ opérations arithmétiques et $O(n^5 \log^2 A)$ opérations binaires, où $A = \max_{i=1}^{n} \|f_i\|$

1. initialiser g_i à f_i pour chaque i de 1 à n
2. calculer l'orthogonalisation de Gram–Schmidt (g_i^*) de (g_i), ainsi que la matrice M des $\mu_{i,j}$ telle que $M\,{}^t(g_1^*, \ldots, g_n^*) = {}^t(g_1, \ldots, g_n)$
3. pour i à partir de 2, tant que $i \leq n$, faire
   (a) pour j de $i-1$ à 1, remplacer g_i par $g_i - \lceil \mu_{i,j} \rfloor g_j$ et réaliser la même transformation sur les lignes de M
   (b) si $i \geq 2$ et si $\|g_{i-1}^*\|^2 > 2\|g_i^*\|^2$,
      (i) échanger g_{i-1} et g_i

(ii) recalculer g_{i-1}^* et g_i^* et les lignes correspondantes de M par le procédé de Gram–Schmidt

(iii) décrémenter i

sinon incrémenter i

4. renvoyer $(g_1, \dots, g_n)$

Traitons l'exemple donné par les deux vecteurs $f_1 = (12, 2)$ et $f_2 = (13, 4)$ de $\mathbb{Z}^2$. Après les étapes (1) et (2), on a la relation

$$\begin{bmatrix} 1 & 0 \\ \frac{41}{37} & 1 \end{bmatrix} \begin{bmatrix} 12 & 2 \\ -\frac{11}{37} & \frac{66}{37} \end{bmatrix} = \begin{bmatrix} 12 & 2 \\ 13 & 4 \end{bmatrix}.$$

Comme $\lceil \frac{41}{37} \rfloor = 1$, la transformation (de réduction) de l'étape (3)(a) revient à des multiplications à gauche par la matrice $\left[\begin{smallmatrix} 1 & 0 \\ -1 & 1 \end{smallmatrix}\right]$ et transforme la relation matricielle ci-dessus en

$$\begin{bmatrix} 1 & 0 \\ \frac{4}{37} & 1 \end{bmatrix} \begin{bmatrix} 12 & 2 \\ -\frac{11}{37} & \frac{66}{37} \end{bmatrix} = \begin{bmatrix} 12 & 2 \\ 1 & 2 \end{bmatrix}.$$

Les normes à considérer vérifient $\|g_1^*\|^2 = 4 \times 37 > 2\|g_2^*\|^2 = 2 \times 11^2/37$, ce qui provoque un échange en (3)(b)(i) avec recalcul en (3)(b)(ii) par le procédé de Gram–Schmidt, et fournit la nouvelle relation matricielle

$$\begin{bmatrix} 1 & 0 \\ \frac{16}{5} & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ \frac{44}{5} & -\frac{22}{5} \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 12 & 2 \end{bmatrix}.$$

Comme $\lceil \frac{16}{5} \rfloor = 3$, une nouvelle transformation (de réduction) à l'étape (3)(a) revient à des multiplications à gauche par la matrice $\left[\begin{smallmatrix} 1 & 0 \\ -3 & 1 \end{smallmatrix}\right]$ et transforme la relation matricielle ci-dessus en

$$\begin{bmatrix} 1 & 0 \\ \frac{1}{5} & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ \frac{44}{5} & -\frac{22}{5} \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 9 & -4 \end{bmatrix}.$$

Comme $\|g_1^*\|^2 = 5 \leq 2\|g_2^*\| = 2 \times 11^2 \times 4$, l'algorithme termine en renvoyant $g_1 = (1, 2)$ et $g_2 = (9, -4)$.

5. Preuve de l'algorithme LLL

Nous développons maintenant des invariants et des variants de l'algorithme qui permettent de montrer la correction et la terminaison de l'algorithme, puis une borne polynomiale sur sa complexité arithmétique.

5.1. Correction. À l'entrée de la boucle (3), la base de réseau (g_i) et son orthogonalisée (g_i^*) par la méthode de Gram–Schmidt vérifient la relation $M\,{}^t(g_1^*, \dots, g_n^*) = {}^t(g_1, \dots, g_n)$. Observons d'abord que cet invariant reste respecté à chaque sortie du corps de boucle. Il est immédiat que la relation matricielle reste vérifiée après la transformation de l'étape (3)(a) ; montrons que, pour λ quelconque et tout $j < i$, le remplacement de g_i par $g_i - \lambda g_j$ et de chaque $\mu_{i,k}$ par $\mu_{i,k} - \lambda\mu_{j,k}$ quand $1 \leq k \leq j$ fait de la nouvelle famille des g_i^* l'orthogonalisée de la nouvelle famille des g_i. (Dans l'algorithme, on choisit $\lambda = \lceil \mu_{i,j} \rfloor$.) Puisque la famille d'espaces emboîtés $\mathbb{Q}g_1 \oplus \cdots \oplus \mathbb{Q}g_k$, d'une part, et celle des $\mathbb{Q}g_1^* \oplus \cdots \oplus \mathbb{Q}g_k^*$, d'autre part, n'ont pas changé, il suffit pour cela de vérifier que le nouveau g_i^* vérifie bien sa définition. En termes des anciennes valeurs, ceci se résume, pour $j < i$, à vérifier la relation

$$g_i^* = g_i - \sum_{k=1}^{i-1} \mu_{i,k} g_k^* = \big(g_i - \lambda g_j\big) - \sum_{k=1}^{i-1} \big(\mu_{i,k} - \lambda\mu_{j,k}\big) g_k^* + \lambda\left(g_j - \sum_{k=1}^{j} \mu_{j,k} g_k^*\right).$$

La dernière parenthèse étant nulle, l'invariant recherché reste préservé par la transformation (3)(a). Quand à l'étape (3)(b), on observe de nouveau que la famille d'espaces emboîtés $\mathbb{Q}g_1 \oplus \cdots \oplus \mathbb{Q}g_k$, d'une part, et celle des $\mathbb{Q}g_1^* \oplus \cdots \oplus \mathbb{Q}g_k^*$, d'autre part, ne sont pas perturbées par l'échange (3)(b)(i), sauf peut-être pour $k = i$ ou $k = i - 1$. C'est pourquoi l'algorithme recalcule explicitement g_{i-1}^* puis g_i^* par les formule de Gram–Schmidt pour restaurer l'invariant.

Un second invariant respecté par la même boucle (3) est la suite d'inégalités $\|g_1^*\|^2 \leq 2\|g_2^*\|^2$, …, $\|g_{i-2}^*\|^2 \leq 2\|g_{i-1}^*\|^2$. Cet invariant se réduit en effet à une absence de contraintes à l'entrée de la boucle (3), pour $i = 2$. Puis, la stratégie de l'étape (3)(b) est de simplement incrémenter i si la suite d'inégalité peut se prolonger par $\|g_{i-1}^*\|^2 \leq 2\|g_i^*\|^2$, ou au contraire de décrémenter i lorsque g_{i-1}^* est modifié.

Ainsi, si une exécution de l'algorithme sort de la boucle (3), la base (g_i) est réduite, ce qui prouve la correction de l'algorithme.

5.2. Terminaison. Pour montrer que l'algorithme termine, on va montrer que l'échange et la mise à jour (3)(b)(i–ii), seule étape qui modifie les g_i^*, les réduit en norme, et réduit dans le même temps une grandeur entière positive associée aux carrés des normes $\|g_i\|^2$. Le nombre d'échange ne pourra donc être que fini. Un invariant entre nombre d'incrémentations et de décrémentations dans l'étape (3)(b) montre alors la terminaison de la boucle (3), et donc de l'algorithme.

Pour un passage dans l'étape (3)(b) avec $i \geq 2$ et $\|g_{i-1}^*\|^2 > 2\|g_i^*\|^2$, notons (h_k) la base du réseau obtenue après l'échange (3)(b)(i) et (h_k^*) l'orthogonalisée de Gram–Schmidt obtenue après le recalcul (3)(b)(ii). On a $h_i = g_{i-1}$, $h_{i-1} = g_i$, et $h_k = g_k$ sinon. De même, on a les égalités entre espaces vectoriels $\mathbb{Q}g_1 \oplus \cdots \oplus \mathbb{Q}g_k = \mathbb{Q}h_1 \oplus \cdots \oplus \mathbb{Q}h_k$, sauf pour $k = i - 1$. Ainsi, les vecteurs orthogonalisés h_k^* et g_k^* sont égaux, sauf peut-être pour $k = i$ et $k = i - 1$. Posons $U = \mathbb{Q}g_1 \oplus \cdots \oplus \mathbb{Q}g_{i-2}$. Par la définition du procédé d'orthogonalisation de Gram–Schmidt, on a d'abord $g_i = h_{i-1} = h_{i-1}^* + u_1$ pour $u_1 \in U$, et aussi $g_i = g_i^* + \mu_{i,i-1}g_{i-1}^* + u_2$ pour $u_2 \in U$. Comme u_1 et u_2 sont les projetés orthogonaux du même vecteur orthogonalement à U, ils sont égaux, de même que h_{i-1}^* et $g_i^* + \mu_{i,i-1}g_{i-1}^*$. En passant aux carrés des normes, on obtient $\|h_{i-1}^*\|^2 = \|g_i^*\|^2 + \mu_{i,i-1}^2\|g_{i-1}^*\|^2 < \|g_{i-1}^*\|^2/2 + (1/2)^2\|g_{i-1}^*\|^2$, où on a utilisé l'hypothèse que le test en (3)(b) a été positif et le fait que l'étape (3)(a) a forcé la relation $\mu_{i,i-1} \leq 1/2$. En résumé, $\|h_{i-1}^*\|^2 < 3\|g_{i-1}^*\|^2/4$. Par la même méthode, en considérant des projections orthogonales de g_{i-1} sur $U' = U \oplus \mathbb{Q}g_{i-1}$, on montre la relation $\|h_i^*\| \leq \|g_i^*\|$.

Notons G_i la matrice obtenue en superposant les i premiers vecteurs g_1, …, g_i, et G_i^* celle obtenue à partir des g_k^* correspondants. La matrice de Gram des vecteurs g_1, …, g_i est la matrice de taille $i \times i$ dont l'entrée (u, v) est le produit scalaire $(g_u|g_v)$. C'est donc une matrice de $\mathbb{Z}^{i\times i}$, qui s'exprime aussi $G_i\,{}^tG_i$. Puisque le bloc M_i de taille $i \times i$ en haut à gauche dans la matrice M est encore trigonal avec des 1 sur la diagonale, donc de déterminant 1, on a

$$d_i = \det(M_i G_i^{*\,t} G_i^{*t} M_i) = \det(M_i)\det(G_i^{*\,t}G_i^*)\det({}^tM_i) = \|g_1^*\|^2 \ldots \|g_i^*\|^2.$$

Ce nombre est donc un entier strictement positif.

Le variant qui va s'avérer adéquat est le produit $D = d_1 \ldots d_{n-1}$, de nouveau un entier strictement positif. Ce nombre est divisé par au moins 4/3 à chaque échange aux étapes (3)(b)(i–ii), tout en restant entier strictement positif. Il ne peut donc y avoir qu'un nombre fini d'échanges lors d'une exécution de l'algorithme LLL. Considérons les nombres e d'échanges en (3)(b)(i) et v de passages en (3)(b) avec un test négatif effectués depuis le début d'une exécution de l'algorithme. En observant ces valeurs à chaque entrée de la boucle (3), on a l'invariant $i = 2 - e + v$. Chaque passage dans la boucle incrémente l'un ou l'autre de ces deux nombres.

Mais comme le nombre e ne peut croître indéfiniment, le nombre v doit ultimement ne plus cesser de croître, et i avec, ce jusqu'à la valeur $i = n + 1$ qui provoque la fin de la boucle (3). L'algorithme termine donc.

5.3. Complexité arithmétique. Au début de l'algorithme, le nombre D qui vient d'être introduit pour montrer la terminaison vaut

$$D_0 = \|g_1^*\|^{2(n-1)} \|g_2^*\|^{2(n-2)} \dots \|g_{n-1}^*\|^2.$$

Mais chaque g_i^* étant alors une certaine projection de l'entrée f_i, on a

$$D_0 \leq \|f_1\|^{2(n-1)} \|f_2\|^{2(n-2)} \dots \|f_{n-1}\|^2 \leq A^{n(n-1)}.$$

Après e échanges en (3)(b)(i), on a l'encadrement $1 \leq D \leq (3/4)^e D_0$, d'où la borne supérieure $n(n-1)\log_{4/3} A$ sur e.

Notons e_{final} et v_{final} les valeurs finales de e et v. On vient de prouver la relation $e_{\text{final}} = O(n^2 \log A)$; par ailleurs l'invariant sur i donne $n+1 = 2-e_{\text{final}}+v_{\text{final}}$, d'où $v_{\text{final}} = O(n^2 \log A)$. Comme l'orthogonalisation initiale se fait en $O(n^3)$ opérations arithmétiques et que chacune des étapes (3)(a) et (3)(b)(i–ii) se fait en $O(n^2)$ opérations arithmétiques, la complexité arithmétique totale de l'algorithme LLL est

$$O(n^3) + (e_{\text{final}} + v_{\text{final}})O(n^2) = O(n^4 \log A).$$

La borne sur la complexité en bits est réellement plus technique et n'est pas présentée ici. L'idée de la preuve est que les entiers utilisés dans l'algorithme ne dépassent pas la taille $O(n \log A)$. La borne annoncée n'est ensuite que le produit de la borne sur la complexité arithmétique par cette borne sur les entiers utilisés.

CHAPITRE 32

Factorisation des polynômes

1. Introduction

Le but de la dernière partie du cours est de montrer comment factoriser des polynômes dans $\mathbb{Z}[X]$. Pour factoriser un polynôme F à coefficients entiers, la majorité des algorithmes actuels procèdent ainsi :

1.: Factorisation de $F \mod p$ dans $\mathbb{Z}/p\mathbb{Z}[X]$, p étant un nombre premier.

2.: Remontée de Hensel, pour passer à une factorisation modulo une puissance suffisante de p.

3.: Recombinaison (par recherche exhaustive ou réduction de réseau) pour retrouver les « vrais » facteurs, c'est-à-dire ceux sur $\mathbb{Z}$.

Le mieux est de donner un exemple : soit $F = x^4 - 2x^3 + 9x^2 - 8x + 20$. Modulo $p = 5$, on a

$$F \equiv x(x+1)(x+3)(x+4) \mod 5.$$

Ensuite, on obtient successivement

— $F \equiv (x+10)(x+11)(x+13)(x+14) \mod 5^2$

— $F \equiv (x+260)(x+261)(x+363)(x+364) \mod 5^4$

— $F \equiv (x+220885)(x+220886)(x+169738)(x+169739) \mod 5^8$.

On renormalise tous les coefficients entre -195312 et 195312 (ici, $195312 = \frac{5^8}{2}$) :

$$\begin{aligned} F &\equiv (x-169740)(x-169739)(x+169738)(x+169739) \mod 5^8 \\ &\equiv f_1 f_2 f_3 f_4 \mod 5^8. \end{aligned}$$

En effectuant toutes les combinaisons possibles, on constate que

$$f_2 f_4 \equiv x^2 + 4 \mod 5^8 \quad \text{et} \quad f_1 f_3 \equiv x^2 - 2x + 5 \mod 5^8.$$

Voilà la factorisation.

Dans le cours présent, on s'intéresse à la première partie, la factorisation dans un corps fini. Il existe quantité d'algorithmes pour cette question, probabilistes ou déterministes. Après quelques rappels / compléments sur les corps finis, on présente l'algorithme de Berlekamp, algorithme « classique » (et déterministe) pour cette question.

2. Corps finis, quelques rappels

Préliminaires : groupes commutatifs finis, arithmétique. Soit G un groupe commutatif fini, noté multiplicativement (dans la suite, G sera le groupe $K - \{0\}$, K étant un corps fini). On appelle *ordre* d'un élément a de G le plus petit entier positif (non nul) ω tel que $a^\omega = 1$. On utilisera dans la suite les résultats suivants :

LEMME 1. *L'ordre de tout élément divise le cardinal $|G|$ de G, et donc $a^{|G|} = 1$ pour tout a de G.*

DÉMONSTRATION. L'ensemble des puissances de a est un sous-groupe de G, de cardinal ω. On utilise alors le théorème de Lagrange, qui dit que le cardinal d'un sous-groupe de G divise $|G|$. □

Le résultat suivant n'est pas bien difficile non plus ; n'importe quel cours sur la théorie des groupes se doit d'en donner une preuve.

LEMME 2. *Soit Ω le ppcm des ordres des éléments de G. Alors, il existe un élément d'ordre Ω ; en particulier, Ω divise $|G|$.*

Enfin, on utilisera le résultat arithmétique suivant.

LEMME 3. *Soit p, m, n trois entiers positifs, avec $p > 1$. Si $p^m - 1$ divise $p^n - 1$ alors m divise n.*

DÉMONSTRATION. L'hypothèse dit que $p^n \equiv 1$ modulo $p^m - 1$. On pose $n = qm + r$, avec $0 \leq r < m$. Modulo $p^m - 1$, on a donc $p^r \equiv p^n$, donc $p^r \equiv 1$. Puisque $r < m$, on a donc $p^r = 1$ et $r = 0$. □

Corps finis. L'exemple le plus simple de corps fini est un quotient de la forme $\mathbb{Z}/p\mathbb{Z}$, où p est un nombre premier, et on le note à l'anglaise $\mathbb{F}_p$.

Soit ensuite P un polynôme irréductible de $\mathbb{F}_p[X]$, de degré d. Alors, $K = \mathbb{F}_p[X]/P$ est également un corps ; puisque $1, X, \ldots, X^{d-1}$ forment une base de K comme espace vectoriel sur $\mathbb{F}_p$, le cardinal de K est $q = p^d$.

PROPOSITION 1. *Soit K un corps fini à q éléments. Pout tout a dans $K - \{0\}$, on a $a^{q-1} = 1$; par suite, on a*

$$X^q - X = \prod_{a \in K} (X - a).$$

DÉMONSTRATION. Soit alors a non nul dans K. Le corps K ayant cardinal q, $K - \{0\}$ est un groupe fini de cardinal $q-1$, donc $a^{q-1} = 1$ (Lemme 1). On en déduit que $a^q = a$ pout tout a de K, nul ou pas. Ceci montre que le polynôme $X^q - X$ s'annule sur les q éléments de K, donc $\prod_{a \in K}(X - a)$ divise $X^q - X$. Comme les deux polynômes sont unitaires et de même degré, ils sont égaux. □

COROLLAIRE 1. *Pour tout $i = 1, \ldots, p-1$, le coefficient binomial $\binom{p}{i}$ vaut zéro modulo p.*

DÉMONSTRATION. Rappelons que ce coefficient est le coefficient de X^i dans $(X + 1)^p$. La proposition précédente entraîne les égalités

$$\begin{array}{rcl} X^p - X & = & \prod_{a \in \mathbb{F}_p}(X - a) \\ & = & \prod_{a \in \mathbb{F}_p}(X - a + 1) \\ & = & \prod_{a \in \mathbb{F}_p}((X + 1) - a) \\ & = & (X + 1)^p - (X + 1). \end{array}$$

On en déduit que $(X + 1)^p = X^p + 1$, qui est ce que l'on voulait montrer. □

Le corollaire suivant est, d'une manière ou d'une autre, à la base de la plupart des résultats « arithmétiques » sur les corps finis.

COROLLAIRE 2. *Soit P un polynôme quelconque dans $\mathbb{F}_p[X]$ et soit K le quotient $\mathbb{F}_p[X]/P$ (qui n'est pas nécessairement un corps). L'application*

$$\phi : \begin{array}{ccc} K & \to & K \\ a & \mapsto & a^p \end{array}$$

est $\mathbb{F}_p$-linéaire.

DÉMONSTRATION. Soient λ dans $\mathbb{F}_p$ et a, b dans K. L'égalité $\phi(\lambda a) = \lambda\phi(a)$ est une conséquence de la Proposition 1, appliquée à $\mathbb{F}_p$. L'égalité $\phi(a+b) = \phi(a)+\phi(b)$ est une conséquence du corollaire précédent (développer le produit $(a+b)^p$). □

Voyons quelques conséquences de ce corollaire.

PROPOSITION 2. *Pour tous $q_0, \ldots, q_n$ dans $\mathbb{F}_p$, on a*

$$\left(\sum_i q_i X^i\right)^p = \sum_i q_i X^{ip}.$$

DÉMONSTRATION. Conséquence facile de la Proposition 1 et du Corollaire 1. □

Le résultat suivant n'est pas utilisé dans l'algorithme de Berlekamp, mais on s'en servira ultérieurement (phase « Distinct Degree Factorization » d'algorithmes probabilistes).

PROPOSITION 3. *Pour $n > 0$, le polynôme $X^{p^n} - X$ est le produit des polynômes unitaires et irréductibles de $\mathbb{F}_p[X]$ dont le degré divise n.*

DÉMONSTRATION. Soit P un polynôme irréductible de degré m, avec $m \mid n$, et soit K le quotient $\mathbb{F}_p[X]/P$; on se rappelle que $|K| = p^m$. Soit ensuite x la classe de X dans K ; la Proposition 1 appliquée à x montre que $x^{p^m-1} = 1$ donc $x^{p^m} = x$, d'où on déduit $x^{p^n} = x$. Donc P divise $X^{p^n} - X$.

Réciproquement, soit P un polynôme irréductible de degré m tel que P divise $X^{p^n} - X$; on veut montrer que m divise n. Soit K le quotient $K = \mathbb{F}_p[X]/P$ et soit x l'image de X dans K. Puisque $P(x) = 0$, $x^{p^n} - x$ est nul. On en tire que pour tous λ_i dans $\mathbb{F}_p$, on a les égalités :

$$\begin{array}{rcl} (\sum_i \lambda_i x^i)^{p^n} & = & \sum_i \lambda_i (x^i)^{p^n} \\ & = & \sum_i \lambda_i (x^{p^n})^i \\ & = & \sum_i \lambda_i x^i, \end{array}$$

la première égalité s'obtenant en appliquant n fois le Corollaire 2. Donc tout élément de K est racine de $X^{p^n} - X$, de sorte que tout élément non nul de K est racine de $X^{p^n-1} - 1$.

Soit maintenant Ω l'exposant du groupe $K - \{0\}$; d'après le Lemme 2, Ω divise $|K - \{0\}| = p^m - 1$. Par ailleurs, par définition, tout élément non nul de K annule $X^\Omega - 1$, donc Ω est plus grand que $p^m - 1$: Il s'en suit que $\Omega = p^m - 1$. Le Lemme 2 montre alors qu'il existe un élément α d'ordre $p^m - 1$ dans $K - \{0\}$.

On a vu que $\alpha^{p^n-1} = \alpha$; on en déduit que $p^m - 1$ divise $p^n - 1$. Le Lemme 3 permet de conclure que m divise n.

À ce stade, on sait donc que les facteurs irréductible de $X^{p^n} - X$ sont exactement les polynômes irréductibles de degré divisant n. Il reste à montrer qu'il n'y a pas de facteur multiple. Pour cela, il suffit de constater que la dérivée de $X^{p^n} - X$ vaut -1, et en particulier que son pgcd avec $X^{p^n} - X$ vaut 1. On utilise ensuite la Proposition 4. □

3. Partie sans carré et décomposition sans carré

Une première étape des algorithmes de factorisation est souvent de se ramener à un polynôme sans facteurs multiples, que l'on appellera ici polynômes sans carré.

Pour cette question, on peut distinguer deux niveaux de raffinement, illustrés sur un exemple : soit par exemple

$$P = (X+1)(X+2)(X+3)^2(X+4)^2(X+5)^3(X+6)^3,$$

sur un corps dans lequel $1, 2, 3, 4, 5$ et 6 sont tous distincts. La *partie sans carré* de P sera le produit où on a « supprimé » toutes les multiplicités, c'est-à-dire

$$(X+1)(X+2)(X+3)(X+4)(X+5)(X+6).$$

La *décomposition sans carré* va plus loin, et sépare les facteurs selon leur multiplicité. Ici, elle donne donc :

$$[(X+1)(X+2), 1], \quad [(X+3)(X+4), 2], \quad \text{et} \quad [(X+5)(X+6), 3].$$

On voit bien que le calcul de décomposition sans carré est l'amorce de la factorisation ; en outre, c'est un calcul sensiblement plus facile que celui de la factorisation.

En effet, on va voir plus bas un algorithme de calcul de la partie sans carré, de complexité essentiellement linéaire en le degré, qui repose uniquement sur des calculs de pgcd bien choisis.

Il n'est pas difficile d'en déduire un algorithme pour la décomposition sans carré : on divise P par sa partie sans carré, on calcule la partie sans carré du résultat, etc ... Ce faisant, on n 'est plus en complexité linéaire, ce qui n'est pas très bon ; il est possible de faire mieux, et de rester essentiellement linéaire, en utilisant un algorithme dû à Yun, que je ne donnerai pas ici faute de temps.

Polynômes sans carrés. Soit P un polynôme de $\mathbb{F}_p[X]$. On dit que P est *sans carré* s'il n'existe pas de polynôme Q de $\mathbb{F}_p[X]$ tel que Q^2 divise P.

LEMME 4. *Soit P dans $\mathbb{F}_p[X]$. La dérivée P' vaut 0 si et seulement si P s'écrit sous la forme Q^p.*

DÉMONSTRATION. Si $P = Q^p$, on a bien $P' = pQ'Q^{p-1} = 0$. Réciproquement, on voit que si $P' = 0$, P est de la forme $P = \sum_i q_i X^{pi}$, qui est une puissance p-ième d'après la Proposition 2. □

PROPOSITION 4. *P est sans carré si et seulement si* $\operatorname{pgcd}(P, P') = 1$.

DÉMONSTRATION. Écrivons la factorisation en irréductibles de P :

$$P = P_1^{a_1} \cdots P_s^{a_s},$$

où tous les a_i sont ≥ 1. En dérivant, on obtient

$$P' = \sum_i a_i P_i' \frac{P}{P_i}.$$

Si P admet un facteur carré, il existe au moins un $a_i \geq 2$, pour lequel P_i divise $\frac{P}{P_i}$. Alors, P_i divisant tous les autres $\frac{P}{P_j}$, il divise P', de sorte que $\operatorname{pgcd}(P, P')$ est différent de 1.

Pour la réciproque, on note pour commencer que les P_i étant irréductibles, leurs dérivées ne sont pas nulles (proposition précédente). Supposons que alors $\operatorname{pgcd}(P, P')$ est différent de 1. Alors, il existe un P_i qui divise P'. Puisqu'il divise tous les autres $\frac{P}{P_j}$, il divise $a_i P_i' \frac{P}{P_i}$.

Puisque P_i' est non nul, P_i ne peut pas le diviser (raison de degré), donc P_i divise $a_i \frac{P}{P_i}$. Si a_i est non nul, P_i divise $\frac{P}{P_i}$, donc P_i^2 divise P, et on a fini. Si a_i est nul (dans $\mathbb{F}_p$), cela signifie que p divise a_i dans $\mathbb{N}$, donc a_i ne vaut pas 1 ; à nouveau, on a fini. □

Décomposition et partie sans carré. Soit toujours P dans $\mathbb{F}_p[X]$, et écrivons sa factorisation en irréductibles :

$$P = P_1^{a_1} \cdots P_s^{a_s}.$$

On regroupe alors les P_i apparaissant avec des exposants similaires. On note ainsi Q_1 le produit des P_i apparaissant à la puissance 1, Q_2 le produit des P_i apparaissant à la puissance 2, … Q_s le produit des P_i apparaissant à la puissance s. Il est alors possible de réécrire P sous la forme :

$$P = Q_1 Q_2^2 \cdots Q_s^s. \tag{1}$$

Attention il est possible que certains Q_i soient égaux à 1. Par ailleurs, tous les Q_i sont sans carré, et ils sont tous premiers entre eux deux à deux.

On appelle *décomposition sans carré* de P la donnée des polynômes $Q_1, \ldots, Q_s$, et *partie sans carré* de P le produit $Q_1 \cdots Q_s$; il s'agit manifestement d'un polynôme sans carré.

Proposition 5. *Si p est plus grand que s, alors la partie sans carré de P est $P/\mathrm{pgcd}(P, P')$.*

Démonstration. En dérivant $P = Q_1 Q_2^2 \cdots Q_s^s$, on obtient

$$P' = \left(Q_2 \cdots Q_s^{s-1}\right)\left(Q_1' Q_2 \cdots Q_s + \cdots + s Q_1 \cdots Q_{s-1} Q_s'\right).$$

Les Q_i' ne sont pas nuls (car les Q_i sont sans carré). Puisque $1, \ldots, s$ ne sont pas nuls modulo p, le membre de droite est donc premier avec tous les Q_i, donc avec P. Le pgcd de P et P' est donc $Q_2 \cdots Q_s^{s-1}$, ce qui donne le résultat. □

Si p est plus petit que (ou égal à) s, il est possible que ce résultat soit pris en défaut : dans $\mathbb{F}_2[X]$, on a $X^2 + 1 = (X+1)^2$ et $(X^2+1)' = 0$. Dans ce cas, $P/\mathrm{pgcd}(P, P')$ vaut 1, alors que la partie sans carré est $X + 1$.

La proposition suivante précise ce type de comportement.

Proposition 6. *Le quotient $P/\mathrm{pgcd}(P, P')$ vaut*

$$\prod_{i \text{ non multiple de } p} Q_i.$$

Démonstration. En reprenant la preuve précédente, on voit que le pgcd de P et P' vaut $(Q_2 \cdots Q_s^{s-1})$ multiplié par

$$\prod_{i \text{ multiple de } p} Q_i.$$

□

Il reste cependant possible d'obtenir les parties manquantes par calcul de pgcd.

Proposition 7. *Soient n le degré de P, $U = \mathrm{pgcd}(P, P')$ et $V = P/U$. Soit ensuite $W = \mathrm{pgcd}(U, V^n)$. Alors U/W vaut*

$$\prod_{i \text{ multiple de } p} Q_i^i.$$

Démonstration. D'après la proposition précédente, V^n vaut

$$\prod_{i \text{ non multiple de } p} Q_i^n,$$

et U vaut

$$(Q_2 \cdots Q_s^{s-1}) \prod_{i \text{ multiple de } p} Q_i.$$

Puisque n est une borne supérieure sur les multiplicités s, on en déduit que W vaut

$$\prod_{i \text{ non multiple de } p} Q_i^{i-1},$$

d'où le résultat. □

Remarque : le U/W de la proposition est une puissance p-ième, d'après la Proposition 2.

Il est enfin possible de déduire un algorithme de calcul de la partie sans carré de toutes ces considérations.

Partie sans carré d'un polynôme.:

Entrée :: P dans $\mathbb{F}_p[X]$.

Sortie :: La partie sans carré de P.

1. Calculer $U = \text{pgcd}(P, P')$, $V = P/U$ et $W = \text{pgcd}(U, V^n)$;
2. Calculer la racine p-ième W_0 de U/W ;
3. Calculer récursivement la partie sans carré S de W_0 ;
4. Renvoyer VS.

PROPOSITION 8. *Cet algorithme calcule la partie sans carré de P en $O(\mathsf{M}(n)\log(n))$ opérations, où M est une fonction de multiplication.*

DÉMONSTRATION. La correction de cet algorithme provient des propositions précédentes, seule la complexité est plus délicate à analyser. Les calculs de pgcd se font en $O(\mathsf{M}(n)\log(n))$ opérations ; remarquer qu'il suffit de calculer V^n modulo U, et que cela se fait en $O(\log(n))$ multiplications modulo U par exponentiation binaire.

Ensuite, W_0 s'obtient sans calcul (voir la Proposition 2) ; son degré est au plus égal à n/p. On en déduit que le temps de calcul $T(n)$ satisfait la récurrence

$$T(n) \leq T\left(\frac{n}{p}\right) + C\ \mathsf{M}(n)(\log(n)),$$

C étant une constante. On en déduit le résultat. □

4. Algorithme de Berlekamp

On va exposer ici un algorithme historique de factorisation, l'algorithme de Berlekamp. Son analyse de complexité mène au résultant suivant.

PROPOSITION 9. *Soit P un polynôme sans carré de $\mathbb{F}_p[X]$. On peut factoriser P par un algorithme déterministe dont la complexité est de*

$$O(n^\omega \ + \ n\,\mathsf{M}(n)\log(n)\,p)$$

opérations de $\mathbb{F}_p$, où ω est l'exposant de l'algèbre linéaire sur $\mathbb{F}_p$ et M une fonction de multiplication.

Quelques remarques :

— L'hypothèse que P est sans carré n'est pas une limitation ici : même en utilisant la version « naïve » de décomposition sans carré suggérée dans la section précédente, on reste dans une complexité inférieure au n^ω qui apparaît ici.

— La complexité de cet algorithme est linéaire en p : cela le rend rapidement peu pratique, dès lors que p atteint quelques milliers ou quelques millions. À l'heure actuelle, on ne connaît pas d'algorithme de factorisation déterministe de complexité logarithmique en p (borne qu'atteignent les algorithmes probabilistes).

Rappel : restes chinois. Soient $P_1, \dots, P_r$ des polynômes définis sur un corps k quelconque. Supposons que $P_1, \dots, P_r$ sont tous premiers entre eux deux-à-deux, et soit P leur produit. Soit ϕ l'application « restes modulo $P_1, \dots, P_r$ » :

$$\phi : \begin{array}{ccc} k[X] & \to & k[X]/P_1 \times \cdots \times k[X]/P_r \\ a & \mapsto & (a \mod P_1, \dots, a \mod P_r). \end{array}$$

Les P_i étant premiers entre eux deux à deux, on sait que :

— ϕ est surjective,

— son noyau est l'ensemble des multiples de P.

Autrement dit, ϕ induit un isomorphisme

$$k[X]/P \equiv k[X]/P_1 \times \cdots \times k[X]/P_r.$$

L'application de Berlekamp. Soit P un polynôme sans carré. En vertu du Corollaire 2, l'application

$$\phi : \begin{array}{ccc} \mathbb{F}_p[X]/P & \to & \mathbb{F}_p[X]/P \\ a & \mapsto & a^p - a \end{array}$$

est une application $\mathbb{F}_p$-linéaire. L'étude de son noyau va permettre de factoriser P, à partir de la proposition suivante. On note $P_1, \dots, P_r$ les facteurs (inconnus) de P :

$$P = P_1 \cdots P_r.$$

PROPOSITION 10. *Soit $a \in \mathbb{F}_p[X]$, non constant, de degré plus petit que P, et tel que $\phi(a \mod P) = 0$. Alors :*

— *pour $i = 1, \dots, r$, $a \mod P_i$ est une constante ;*

— *ces constantes ne sont pas toutes égales.*

DÉMONSTRATION. Puisque tous les P_i sont distincts et irréductibles, on a d'après le paragraphe précédent un isomorphisme

$$\mathbb{F}_p[X]/P \equiv \mathbb{F}_p[X]/P_1 \times \cdots \times \mathbb{F}_p[X]/P_r,$$

donné par

$$a \mod P \mapsto (a \mod P_1, \dots, a \mod P_r).$$

Les P_i étant irréductibles, chaque $\mathbb{F}_p[X]/P_i$ est un corps. Dans la représentation « produit de corps », l'application ϕ s'écrit bien sûr :

$$(a_1, \dots, a_r) \mapsto (a_1^p - a_1 \mod P_1, \dots, a_r^p - a_r \mod P_r).$$

Dans cette représentation, on a alors la caractérisation suivante du noyau de ϕ.

LEMME 5. *Le r-uplet $(a_1, \dots, a_r)$ est dans le noyau de ϕ si et seulement si $a_i \in \mathbb{F}_p \subset \mathbb{F}_p[X]/P_i$ pour tout i.*

DÉMONSTRATION. On sait (Proposition 1) que le polynôme $X^p - X$ se factorise dans $\mathbb{F}_p$, et donc dans $\mathbb{F}_p[X]/P_i$, sous la forme $\prod_{a \in \mathbb{F}_p}(X - a)$. Puisque $\mathbb{F}_p[X]/P_i$ est un corps, $a_i \in \mathbb{F}_p[X]/P_i$ est une racine de $X^p - X$ si et seulement s'il annule un des facteurs $X - a$, donc si et seulement si $a_i \in \mathbb{F}_p$. □

Ceci prouve le premier point de la proposition. Quant au second, remarquons que si tous les $a_i = a \mod P_i$ sont les mêmes, on a nécessairement $a = a_i$ (pout tout i), car on a supposé a de degré plus petit que P. En particulier, a devrait être constant, ce qui est contraire à nos hypothèses. □

La clé de l'algorithme de Berlekamp est la proposition suivante.

PROPOSITION 11. *Soit a comme dans la proposition précédente. Il existe b dans $\mathbb{F}_p$ tel que* $\mathrm{pgcd}(P, a-b)$ *est un diviseur non trivial de P.*

DÉMONSTRATION. Pour b quelconque dans $\mathbb{F}_p$, on a l'égalité

$$\mathrm{pgcd}(P, a-b) = \prod_i P_i,$$

où le produit est pris sur tous les P_i tels que P_i divise $a-b$, c'est-à-dire tels que $a \mod P_i = b$. On sait qu'il existe au moins deux valeurs distinctes pour les $a \mod P_i$. Chacune d'entre elles donnant un diviseur de P, ces diviseurs sont stricts. □

D'où l'algorithme.

Calcul d'un facteur.:

Entrée :: P dans $\mathbb{F}_p[X]$, sans carré.

Sortie :: Un facteur de P.

1. Calculer la matrice de l'application ϕ, et un vecteur a dans le noyau.
2. Calculer tous les $\mathrm{pgcd}(P, a-b)$, jusqu'à trouver un facteur non trivial.

PROPOSITION 12. *L'algorithme précédent calcule un facteur non trivial de P en*

$$O(n^\omega + \mathsf{M}(n)\log(n)\,p)$$

opérations de $\mathbb{F}_p$.

DÉMONSTRATION. Le premier pas est de calculer la matrice de ϕ. On commence par calculer $X^p \mod P$: cela demande $O(\log(p))$ opérations modulo P, soit $O(\mathsf{M}(n)\log(p))$ opérations dans $\mathbb{F}_p$. Ensuite, il faut calculer $(X^2)^p = (X^p)^2, (X^3)^p = (X^p)^3, \ldots$, ce qui se fait en n multplications mod P, soit $O(n\mathsf{M}(n))$ opérations en tout.

L'algèbre linéaire coûte n^ω. Une fois obtenu un vecteur du noyau, chaque essai de pgcd coûte $O(\mathsf{M}(n)\log(n))$ opérations, et il faut potentiellement en faire p. En mettant tout bout-à-bout, on obtient le résultat ci-dessus. □

Pour trouver la factorisation complète, il suffit de réitérer le procédé ce qui entraîne *a priori* un surcoût de n. En fait, il n'y a pas besoin de refaire de l'algèbre linéaire, de sorte que le coût total de la factorisation est

$$O(n^\omega + n\mathsf{M}(n)\log(n)\,p)$$

opérations de $\mathbb{F}_p$.

Exercices

EXERCICE 1 (Factorisation probabiliste en temps polynomial). Le but de cet exercice est de donner un algorithme (probabiliste) pour la factorisation dans $\mathbb{F}_p[X]$, dont la complexité soit polynomiale en $(n, \log(p))$, où n est le degré du polynôme à factoriser. Ici, p est un nombre premier différent de 2 ; les complexités des algorithmes seront énoncées en nombre d'opérations dans $\mathbb{F}_p$.

L'algorithme proposé fonctionne en plusieurs étapes. On admettra le fait suivant (qui généralise un résultat montré en cours) : pour tout $d \geq 0$, le polynôme

$$X^{p^d} - X$$

est le produit de tous les polynômes irréductibles et unitaires de $\mathbb{F}_p[X]$, dont le degré divise d.

Question 1: Donner un algorithme, qui, étant donné P dans $\mathbb{F}_p[X]$, calcule le produit de tous les polynômes unitaires de degré 1 qui divisent P. Quelle est sa complexité ?

Question 2: Donner un algorithme, qui, étant donné P dans $\mathbb{F}_p[X]$, calcule $P_1, \ldots, P_n$, où n est le degré de P, et P_i est le produit de tous les polynômes irréductibles et unitaires de degré i qui divisent P. Quelle est sa complexité ?

Question 3: Expliquer comment retrouver simplement les facteurs de P et leurs multiplicités à partir des facteurs des P_i.

Dans la seconde étape, on peut donc faire l'hypothèse que le polynôme à factoriser est de la forme

$$Q = Q_1 \cdots Q_r,$$

avec $\deg Q_1 = \cdots = \deg Q_r = s$ et $\deg Q = m = rs$, et où les Q_i sont irréductibles.

On admet dans ces conditions que si on tire un élément A aléatoirement dans $\mathbb{F}_p[X]/Q_i - \{0\}$, avec la distribution uniforme, alors

$$A^{\frac{p^s-1}{2}}$$

vaut 1 (resp. -1) avec probabilité $\frac{1}{2}$.

Question 4: Soit A dans $\mathbb{F}_p[X]/Q$, et $B = A^{\frac{p^s-1}{2}}$ dans $\mathbb{F}_p[X]/Q$. Donner une borne supérieure sur la complexité du calcul de B. Que peut valoir $(B \bmod Q_1, \ldots, B \bmod Q_r)$?

On admettra que si A est tiré aléatoirement dans $\mathbb{F}_p[X]/Q - \{0\}$, avec la distribution uniforme, les valeurs de $(B \bmod Q_1, \ldots, B \bmod Q_r)$ sont toutes équiprobables.

Question 5: Montrer que

$$\operatorname{pgcd}(B-1, Q) = \prod_{i \text{ tel que } (B \bmod Q_i)=1} Q_i.$$

En déduire que si $(B \bmod Q_1, \ldots, B \bmod Q_r) \neq (1, \ldots, 1)$ ainsi que $(B \bmod Q_1, \ldots, B \bmod Q_r) \neq (-1, \ldots, -1)$ alors $\operatorname{pgcd}(B-1, Q)$ est non-trivial. En déduire ensuite un algorithme qui calcule un facteur propre de Q, non constant, avec probabilité $\geq \frac{1}{2}$. Quelle est sa complexité ?

Conclure sur la complexité de la factorisation de Q (on ne demande pas une estimation fine de complexité).

CHAPITRE 33

Exercices récapitulatifs

Sauf mention contraire, dans toute la suite, $\mathbb{K}$ désigne un corps effectif de caractéristique nulle.

Exercice 1 (Équivalence des complexités de diverses opérations polynomiales). Soit $\mathbb{K}$ un corps de caractéristique différente de 2 et soient $\mathsf{M}(n), \mathsf{I}(n), \mathsf{D}(n)$ et $\mathsf{S}(n)$ les complexités (mesurées en nombre d'opérations arithmétiques dans $\mathbb{K}$) pour calculer respectivement : le produit de deux polynômes de $\mathbb{K}[X]$ de degré $< n$, l'inverse modulo X^n d'un polynôme de degré $< n$, le reste de la division d'un polynôme de degré $< 2n$ par un polynôme de degré n et le carré d'un polynôme de degré $< n$. Le but de l'exercice est de prouver que les fonctions $\mathsf{M}, \mathsf{I}, \mathsf{D}$ et S ont le même ordre de grandeur.

1. Par quelle technique peut-on montrer que $\mathsf{I} \in \mathcal{O}(\mathsf{M})$ et $\mathsf{D} \in \mathcal{O}(\mathsf{M})$?
2. Prouver l'identité $y^2 = \left(y^{-1} - (y+1)^{-1}\right)^{-1} - y$ et conclure que $\mathsf{S} \in \mathcal{O}(\mathsf{I})$.
3. Montrer que $\mathsf{M} \in \mathcal{O}(\mathsf{S})$, en utilisant l'égalité $2fg = (f+g)^2 - f^2 - g^2$.
4. Le polynôme réciproque d'un polynôme $b \in \mathbb{K}[X]$ de degré n est noté $\mathrm{réc}_n(b)$. Établir une relation entre $\mathrm{réc}_n(b)^{-1} \bmod X^n$ et le quotient de la division de X^{2n} par b. En déduire que $\mathsf{I} \in \mathcal{O}(\mathsf{D})$.
5. Conclure que $\mathcal{O}(\mathsf{M}) = \mathcal{O}(\mathsf{I}) = \mathcal{O}(\mathsf{S}) = \mathcal{O}(\mathsf{S})$.

Exercice 2 (Décalage rapide de polynômes).

1. Montrer que si P est un polynôme dans $\mathbb{Q}[X]$ de degré au plus n, alors, quel que soit $a \in \mathbb{Q}$, on peut calculer les coefficients du « polynôme décalé » $Q(X) = P(X+a)$ en $O(\mathsf{M}(n))$ opérations dans $\mathbb{Q}$.

   Indication. Commencer par prouver l'identité suivante dans $\mathbb{Q}[X]$:

$$\sum_{i=0}^{n} P^{(i)}(a) X^{n-i} = \left(\sum_{i=0}^{n} P^{(i)}(0) X^{n-i}\right) \times \left(\sum_{i=0}^{n} \frac{(aX)^i}{i!}\right) \mod X^{n+1}.$$

2. Montrer que les coefficients des numérateurs et des dénominateurs des fractions rationnelles

$$R_0 = \frac{1}{X-1} + \frac{1}{X-2} + \cdots + \frac{1}{X-n} \quad \text{et} \quad R_1 = \frac{1}{X-1} + \frac{2}{X-2} + \cdots + \frac{n}{X-n}$$

   peuvent être calculés en $O(\mathsf{M}(n))$ opérations dans $\mathbb{Q}$.

   Indication. Pour R_0, donner un algorithme de type « diviser pour régner » basé sur l'algorithme de la question (1). Pour R_1, en ramener le calcul à celui de R_0.

La formule de Lagrange montre qu'il existe un unique polynôme $P(X) \in \mathbb{Q}[X]$ de degré strictement inférieur à n tel que

$$P(i) = (-1)^i \binom{n}{i-1}^{-1}, \quad \text{pour tout} \quad i = 1, 2, \ldots, n.$$

3. Exprimer P en termes de R_0 et R_1 et donner un algorithme qui calcule les coefficients de P en $O(\mathsf{M}(n))$ opérations dans $\mathbb{Q}$. Ces coefficients sont supposés connus dans la suite de l'exercice.
4. Montrer que la suite d'éléments de $\mathbb{Q}$ de terme général $a_i = P(i)$ vérifie une récurrence linéaire d'ordre n à coefficients constants.
5. En déduire un algorithme pour calculer les valeurs $P(n+1), \ldots, P(2n)$ en $O(\mathsf{M}(n))$ opérations dans $\mathbb{Q}$.
6. Montrer qu'on peut calculer $P(n+1), \ldots, P(2n)$ en $O(n)$ opérations dans $\mathbb{Q}$.

Exercice 3 (Extrapolation sur une suite arithmétique). Soit P un polynôme inconnu de $\mathbb{K}[X]$. Le but de l'exercice est de montrer que, à partir de la donnée d'un élément $a \in \mathbb{K}$, d'une borne d sur le degré de P, et des valeurs $P(0), P(1), \ldots, P(d)$, il est possible de déterminer l'ensemble des valeurs $P(a), P(a+1), \ldots, P(a+d)$ en $O(\mathsf{M}(d))$ opérations dans $\mathbb{K}$.

1. Traiter d'abord le cas où a appartient à l'ensemble $\{-d, -d+1, \ldots, d-1, d\}$.

Supposons dans la suite que $a \in \mathbb{K}$ est tel que tous les éléments $a-d, \ldots, a+d$ sont non nuls. On rappelle la formule d'interpolation de Lagrange

$$P(X) = \sum_{i=0}^{d} p_i \prod_{j=0,\, j\neq i}^{d} (X-j), \quad \text{où} \quad p_i = \frac{P(i)}{\prod_{j=0,\, j\neq i}^{d}(i-j)}. \tag{L}$$

2. Montrer que les éléments $p_0, p_1, \ldots, p_d$ peuvent être tous calculés en $O(d)$ opérations dans $\mathbb{K}$.

En substituant $X = a + k$ dans l'équation (L), on obtient l'égalité

$$P(a+k) = \Delta_k \cdot \sum_{i=0}^{d} \frac{p_i}{a+k-i}, \quad \text{où} \quad \Delta_k = \prod_{j=0}^{d}(a+k-j). \tag{$\tilde{\text{L}}$}$$

3. Montrer que les éléments Δ_k pour $k = 0, \ldots, d$, peuvent être tous calculés en $O(d)$ opérations dans $\mathbb{K}$.
4. Utiliser l'égalité ($\tilde{\text{L}}$) pour conclure que l'ensemble des valeurs $P(a), P(a+1), \ldots, P(a+d)$ peut se calculer en $O(\mathsf{M}(d))$ opérations dans $\mathbb{K}$.

Exercice 4 (Décomposition en éléments simples). On considère une fraction rationnelle

$$F(X) = \frac{P(X)}{Q_1^{\ell_1}(X) \cdots Q_k^{\ell_k}(X)},$$

où $P, Q_1, \ldots, Q_k$ sont des polynômes de $\mathbb{Q}[X]$ et les ℓ_i sont des entiers strictement positifs tels que

(H1) $\deg(P) < \sum_{i=1}^{k} \ell_i \cdot \deg(Q_i) = n$, et

(H2) $Q_1, \ldots, Q_k$ sont deux à deux premiers entre eux.

La décomposition en éléments simples de F est une écriture

$$\frac{P(X)}{Q_1^{\ell_1}(X) \cdots Q_k^{\ell_k}(X)} = \sum_{i=1}^{k} \sum_{j=1}^{\ell_i} \frac{C_{i,j}(X)}{Q_i^j(X)},$$

où les $C_{i,j}$ sont des polynômes de $\mathbb{Q}[X]$ tels que $\deg(C_{i,j}) < \deg(Q_i)$ pour tous i, j. L'existence et l'unicité des polynômes $C_{i,j}$ est admise. On appelle DES_n le problème du calcul des polynômes $C_{i,j}$ à partir des polynômes $P, Q_1, \ldots, Q_k$ et des entiers $n, \ell_1, \ldots, \ell_k$ satisfaisant aux hypothèses (H1) et (H2).

L'objectif de cet exercice est de montrer qu'il est possible de résoudre le problème DES_n en $O(\mathsf{M}(n)\log(n))$ opérations. Ici, et dans toute la suite de l'exercice, par *opération* on entend *opération arithmétique dans le corps* $\mathbb{Q}$ et $\mathsf{M}(n)$ représente une borne sur le nombre d'opérations suffisant à multiplier deux polynômes de $\mathbb{Q}[X]$ de degré au plus n.

1. Donner un algorithme simple, reposant sur de l'algèbre linéaire, qui résout le problème DES_n et estimer le nombre d'opérations que cet algorithme utilise.
2. Dans le cas $k = 1$, le problème DES_n se spécialise au problème du calcul, à partir de deux polynômes $P, Q \in \mathbb{Q}[X]$ tels que $\deg(P) < \ell \cdot \deg(Q) = n$, des polynômes $C_1, \ldots, C_\ell$ vérifiant

$$\frac{P(X)}{Q^\ell(X)} = \sum_{j=1}^{\ell} \frac{C_j(X)}{Q^j(X)}, \quad \text{et } \deg(C_j) < \deg(Q) \text{ pour tout } j \geq 1.$$

   Montrer que dans ce cas, le problème peut se résoudre en $O(\mathsf{M}(n)\log(n))$ opérations.

Le but des questions 3–9 ci-dessous est de montrer que le problème DES_n avec $\ell_i = 1$ pour tout i peut se résoudre en $O(\mathsf{M}(n)\log(n))$ opérations. Autrement dit, étant donnés les polynômes $P, R_1, \ldots, R_k$ dans $\mathbb{Q}[X]$, avec $\deg(P) < \sum_{i=1}^{k} \deg(R_i) = n$ et $R_1, \ldots, R_k$ deux à deux premiers entre eux, on veut calculer en $O(\mathsf{M}(n)\log(n))$ opérations les polynômes $C_1, \ldots, C_k \in \mathbb{Q}[X]$ tels que

$$\frac{P(X)}{R_1(X)\cdots R_k(X)} = \sum_{i=1}^{k} \frac{C_i(X)}{R_i(X)}, \quad \text{et } \deg(C_i) < \deg(R_i) \text{ pour tout } i \geq 1.$$

Pour ce faire, on introduit

$$R(X) = \sum_{i=1}^{k} \left(\prod_{\substack{j=1 \\ j \neq i}}^{k} R_j(X) \right)$$

et pour $1 \leq i \leq k$, on considère le reste $D_i(X)$ de la division euclidienne de $R(X)$ par $R_i(X)$.

3. Donner un algorithme qui calcule le polynôme $R(X)$ en $O(\mathsf{M}(n)\log(k))$ opérations.
4. Montrer que les polynômes $D_1, \ldots, D_k$ peuvent être calculés eux aussi en $O(\mathsf{M}(n)\log(k))$ opérations.
5. Montrer que pour tout $1 \leq i \leq k$, les polynômes D_i et R_i sont premiers entre eux.
6. En notant E_i l'inverse de D_i modulo R_i, montrer que C_i est égal au reste de la division euclidienne de $P \cdot E_i$ par R_i.
7. Donner un algorithme pour le calcul des E_i en $O(\mathsf{M}(n)\log(n))$ opérations.
8. Donner un algorithme pour le calcul des C_i, en $O(\mathsf{M}(n)\log(k))$ opérations.
9. Conclure qu'on peut résoudre le problème DES_n avec $\ell_i = 1$ pour tout i en $O(\mathsf{M}(n)\log(n))$ opérations.
10. Conclure qu'il est possible de résoudre le problème DES_n en $O(\mathsf{M}(n)\log(n))$ opérations.

EXERCICE 5 (Polynômes de Fibonacci). Soit $F_n(X) \in \mathbb{K}[X]$ la suite des polynômes de Fibonacci, définie par les conditions initiales $F_0 = 1$, $F_1 = X$ et la récurrence $F_{n+1} = XF_n + F_{n-1}$ pour tout $n \geq 1$. Soit $N \geq 1$.

1. Estimer le coût d'un calcul direct de tous les coefficients de $F_N(X)$.
2. Montrer qu'il est possible de calculer tous les coefficients de $F_N(X)$ en utilisant $O\big(\mathsf{M}(N)\big)$ opérations dans $\mathbb{K}$.
3. Montrer qu'il est possible de calculer la somme des coefficients de F_N en seulement $O(\log N)$ opérations dans $\mathbb{K}$.
4. Donner un algorithme qui permet le calcul des coefficients de F_N en $O(N)$ opérations dans $\mathbb{K}$.

EXERCICE 6 (Fonctions algébriques et résultants). Soit $y(X)$ une fonction algébrique annulée par $P(X, Y) \in \mathbb{Q}[X, Y]$.

1. Montrer que la dérivée $y'(X)$ est aussi algébrique, en exhibant un polynôme résultant $R(X, Y)$ qui l'annule.
2. Déterminer un tel polynôme R dans le cas où $P(X, Y) = XY^2 - Y + 1$.
3. Supposant que P est de degré au plus d en X et en Y, estimer les degrés en X et en Y du polynôme R, puis proposer un algorithme pour son calcul.
4. Analyser la complexité de votre algorithme, mesurée en nombre d'opérations arithmétiques dans $\mathbb{Q}$. En utilisant les meilleurs algorithmes vus en cours pour les opérations fondamentales sur les polynômes et les matrices, à quel exposant aboutissez-vous ? Ici, par *exposant* on entend un réel α tel que votre algorithme a une complexité en $\tilde{O}(n^\alpha)$, c'est-à-dire linéaire en n^α à des facteurs logarithmiques près.

EXERCICE 7 (Produit diamant). On considère deux polynômes unitaires $f, g \in \mathbb{Q}[T]$, de degrés strictement plus petits que n. Étant donné $H \in \mathbb{Q}[X, Y]$, dont les degrés partiels par rapport à chacune des variables X et Y sont strictement inférieurs à n, le *produit diamant* $f \diamond_H g$ est le polynôme de degré $D = n^2$ défini par

$$f \diamond_H g = \prod_{f(\alpha)=0,\, g(\beta)=0} \big(T - H(\alpha, \beta)\big),$$

le produit étant pris sur les racines complexes de f et de g comptées avec multiplicités. On s'intéresse dans cet exercice au calcul efficace du produit diamant.

1. Montrer que le produit diamant peut s'exprimer à l'aide de résultants et que ses coefficients sont rationnels.

On note A l'algèbre quotient $\mathbb{Q}[X, Y]/(f(X), g(Y))$. Un élément $g \in A$ admet une écriture unique

$$g = \sum_{0 \leq i,j < n} g_{i,j} x^i y^j$$

avec $g_{ij} \in \mathbb{Q}$, où x et y sont les images canoniques de X et Y dans A. On admet que si P_1, P_2 sont deux polynômes de $\mathbb{Q}[X, Y]$ de degrés au plus d_X en X et au plus d_Y en Y, on peut calculer leur produit $P_1 P_2$ en $O\big(\mathsf{M}(d_X d_Y)\big)$ opérations dans $\mathbb{Q}$.

2. Montrer que $O\big(\mathsf{M}(D)\big)$ opérations suffisent pour effectuer le produit de deux éléments de A.

À tout polynôme G de $\mathbb{Q}[X,Y]$ on associe l'application $\mathbb{Q}$-linéaire de multiplication par $G(x,y)$ dans A, définie par $v \in A \mapsto G(x,y)v \in A$. On admet que la trace, notée $\mathrm{Tr}(G)$, de cette application linéaire est égale à $\sum_{\alpha,\beta} G(\alpha,\beta)$, la somme étant prise sur les racines complexes α de f et β de g comptées avec multiplicités.

3. Montrer que, pour tout $s \in \mathbb{N}$, la trace $\mathrm{Tr}(H^s)$ s'exprime comme la somme des puissances s^{es} des racines de $f \diamond_H g$.
4. Expliciter pour $0 \leq i < n$ ce que valent $\mathrm{Tr}(X^i)$ et $\mathrm{Tr}(Y^i)$.

On rappelle que tout polynôme unitaire $P \in \mathbb{Q}[T]$ de degré au plus D peut être représenté soit par la suite de ces coefficients, soit par la suite des sommes des puissances s^{es} de ses racines, pour $s = 0, \ldots, D$, et qu'il est possible de passer d'une représentation à l'autre en $O\big(\mathsf{M}(D)\big)$ opérations.

5. Donner un algorithme pour le calcul des valeurs $\mathrm{Tr}(X^i), 0 \leq i < n$, en $O\big(\mathsf{M}(n)\big) \subseteq O(D)$ opérations arithmétiques. Même question pour les valeurs $\mathrm{Tr}(Y^i)$, $0 \leq i < n$.
6. En déduire que l'ensemble des valeurs $\mathrm{Tr}(X^iY^j)$, pour $0 \leq i,j < n$, peut être calculé en $O(D)$ opérations dans $\mathbb{Q}$.
7. Donner un algorithme qui, pour tout $N \geq 1$, calcule la suite
$$\mathrm{Tr}(1), \mathrm{Tr}(H), \mathrm{Tr}(H^2), \ldots, \mathrm{Tr}(H^{N-1})$$
en $O\big(N\,\mathsf{M}(D)\big)$ opérations dans $\mathbb{Q}$.
8. En déduire un algorithme pour le calcul de $f \diamond_H g$ en $O\big(D\,\mathsf{M}(D)\big)$ opérations dans $\mathbb{Q}$.

Exercice 8 (Multiplication polynomiale de complexité quasi-linéaire). Le but de cet exercice est de montrer que la multiplication de polynômes de degré au plus N à coefficients dans un corps $\mathbb{K}$ contenant N éléments distincts en progression géométrique peut s'effectuer en complexité arithmétique quasi-linéaire par rapport à N.

Soit A et B deux polynômes de $\mathbb{K}[X]$, de degré strictement inférieurs à N. Soient n et d deux entiers tels que $N \leq nd$ et écrivons
$$A = A_0(X) + A_1(X)X^d + \cdots + A_{n-1}(X)(X^d)^{n-1},$$
$$B = B_0(X) + B_1(X)X^d + \cdots + B_{n-1}(X)(X^d)^{n-1},$$
où les polynômes $A_i(X), B_i(X) \in \mathbb{K}[X]$ ont degré au plus $d-1$. Soit $\mathcal{G}$ un sous-ensemble de $\mathbb{K}$ contenant $2d$ éléments distincts en progression géométrique.

1. Si T opérations de $\mathbb{K}$ suffisent pour multiplier dans $\mathbb{K}[X,Y]$ les polynômes
$$\mathsf{A}(X,Y) = A_0(X) + A_1(X)Y + \cdots + A_{n-1}(X)Y^{n-1} \quad \text{et}$$
$$\mathsf{B}(X,Y) = B_0(X) + B_1(X)Y + \cdots + B_{n-1}(X)Y^{n-1},$$
montrer qu'on peut multiplier $A(X)$ et $B(X)$ en $\mathsf{T} + O(nd)$ opérations de $\mathbb{K}$.

Pour calculer AB, il suffit donc d'effectuer le produit $\mathsf{C}(X,Y) = \mathsf{A}(X,Y)\mathsf{B}(X,Y)$, c'est-à-dire de calculer les coefficients $C_i(X) \in \mathbb{K}[X]$, avec $\deg(C_i) < 2d$, de
$$\mathsf{C}(X,Y) = C_0(X) + C_1(X)Y + \cdots + C_{2n-2}(X)Y^{2n-2}.$$

2. Montrer que le calcul du produit AB se ramène à $O(d)$ multiplications de $\mathbb{K}[Y]$ en degré $< n$ et $O(n)$ évaluations/interpolations sur les points de $\mathcal{G}$. Indication : utiliser l'égalité $\mathsf{A}(g,Y)\mathsf{B}(g,Y) = \mathsf{C}(g,Y)$, valable pour $g \in \mathcal{G}$.

3. En déduire qu'il existe trois constantes $\alpha, \beta, \gamma \geq 1$, universelles (c'est-à-dire indépendantes de N, n, d), telles que la complexité arithmétique M de l'algorithme de (2) vérifie $\mathsf{M}(nd) \leq \alpha n\,\mathsf{M}(d) + \beta d\,\mathsf{M}(n) + \gamma nd$.
4. Écrire un algorithme récursif de multiplication dans $\mathbb{K}[X]$ de complexité arithmétique $O\big(N(\log N)^{\log(\alpha\beta)}\big)$ en degré N.

Exercice 9 (Multiplication d'opérateurs différentiels). Dans l'algèbre de polynômes non-commutatifs $\mathbb{Q}[X]\langle\theta\rangle$ en les indéterminées X et $\theta = \frac{d}{dX}$ satisfaisant à la règle de commutation $\theta X = X\theta + X$, on considère deux éléments A et B donnés sous la forme

$$A = \sum_{i=0}^{n}\sum_{j=0}^{n} a_{i,j} X^i \theta^j \qquad \text{et} \qquad B = \sum_{i=0}^{n}\sum_{j=0}^{n} b_{i,j} X^i \theta^j$$

avec $a_{i,j} \in \mathbb{Q}$ et $b_{i,j} \in \mathbb{Q}$. Le but de cet exercice est d'étudier la complexité arithmétique du calcul du produit $C = BA$, c'est-à-dire du calcul, à partir des constantes $a_{i,j}, b_{i,j} \in \mathbb{Q}$, des constantes $c_{i,j} \in \mathbb{Q}$ telles que

$$C = \sum_{i=0}^{n}\sum_{j=0}^{n} c_{i,j} X^i \theta^j.$$

Les questions 1–3 qui suivent sont largement indépendantes.

1. (a) Montrer que $\theta^i X^\ell = X^\ell(\theta + \ell)^i$ pour $i \geq 0$ et $\ell \geq 0$, et en déduire que le calcul du produit $\theta^i X^\ell$ peut s'effectuer en $O(i)$ opérations arithmétiques dans $\mathbb{Q}$.
   (b) Donner des bornes, en fonction de n, sur les degrés de C en X et en θ.
   (c) Proposer un algorithme naïf pour le calcul de C et estimer son coût arithmétique.
2. Montrer qu'il est possible de calculer C en $O\big(n^2\,\mathsf{M}(n)\big)$ opérations dans $\mathbb{Q}$, à l'aide d'un schéma itératif. On pensera à écrire B sous la forme

$$B = \sum_{i=0}^{n} b_i(X)\theta^i, \qquad \text{avec } b_i(X) \in \mathbb{Q}[X].$$

Dans la suite, on montre comment améliorer les résultats des questions (1) et (2), grâce à un algorithme de type évaluation-interpolation.

3. (a) Calculer $\theta^j(X^k)$, pour $j \geq 0$ et $k \geq 0$. En déduire $P(\theta)(X^k)$, pour $k \geq 0$ et $P \in \mathbb{Q}[X]$.
   (b) Montrer que les coefficients de C sont uniquement déterminés par la suite des polynômes $C(X^0), C(X^1), \ldots, C(X^{2n})$.
   (c) Donner un algorithme de complexité arithmétique $O\big(n\,\mathsf{M}(n)\log n\big)$ pour le calcul de tous les polynômes $p_k := A(X^k)$, pour $0 \leq k \leq 2n$, et $q_i := B(X^i)$, pour $0 \leq i \leq 3n$.
   (d) Montrer que le calcul des polynômes $B(p_0), \ldots, B(p_{2n})$ se ramène à une multiplication de matrices de tailles $(4n+1) \times (3n+1)$ et $(3n+1) \times (2n+1)$.
   (e) Montrer que, à partir de $C(X^0), \ldots, C(X^{2n})$, il est possible de calculer les coefficients de C en $O\big(n\,\mathsf{M}(n)\log n\big)$ opérations arithmétiques.
   (f) Conclure en écrivant un algorithme complet pour le calcul de C basé sur (c), (d) et (e). Estimer sa complexité.

Exercice 10 (Une famille d'intégrales). L'intégrale

$$c_{n,k} = \int_0^{+\infty} t^k K_0(t)^n \, dt,$$

où K_0 est une fonction de Bessel (voir ci-dessous), est intervenue récemment dans des études de physique liées au modèle d'Ising. Empiriquement, il est apparu que ces intégrales vérifient une récurrence de la forme

$$(k+1)^{n+1} c_{n,k} + \sum_{\substack{2 \le j < n \\ j \text{ pair}}} P_{n,j}(k) c_{n,k+j} = 0, \tag{R}$$

où les $P_{n,j}$ sont des polynômes de degré au plus $n+1-j$.

Le but de cet exercice est de prouver l'existence d'une telle récurrence et de développer un algorithme permettant de la calculer. Le point de départ est l'équation différentielle

$$\theta^2(y) - t^2 y = 0$$

vérifiée par $y = K_0(t)$. Dans cette équation, θ représente l'opérateur $t\frac{d}{dt}$. Ainsi, $\theta^2(y) = t(ty')'$. Il sera plus commode de travailler avec θ qu'avec d/dt, et on rappelle $\theta(uv) = u\theta(v) + \theta(u)v$.

On admettra que les intégrales $c_{n,k}$ convergent pour tous n et k entiers positifs, et que pour tout i, j entiers positifs on a en outre

$$\lim_{t\to 0} t^j (K_0(t)^n)^{(i)} = \lim_{t\to\infty} t^j (K_0(t)^n)^{(i)} = 0.$$

1. Montrer que K_0^2 vérifie l'équation différentielle
$$\theta^3(y) - 4t^2\theta(y) - 4t^2 y = 0.$$
2. Plus généralement, montrer que pour tout $n \in \mathbb{N}$, K_0^n vérifie une équation différentielle linéaire. Donner une borne supérieure sur son ordre. Vérifier que cette borne donne la bonne valeur pour $n = 2$.
3. En déduire l'existence d'une récurrence linéaire vérifiée par les $c_{n,k}$, pour chaque n. Donner un algorithme calculant cette récurrence.
4. Montrer par exemple que la suite $c_{2,k}$ vérifie
$$k^3 c_{2,k-1} = 4(k+1)c_{2,k+1}.$$
5. Borner le degré des coefficients dans l'équation calculée en question (2).
6. En déduire une borne de complexité sur l'algorithme de (3) en nombre d'opérations dans $\mathbb{Q}$.

La suite de l'exercice montre comment accélérer ce calcul en tirant parti du fait que l'équation différentielle vérifiée par K_0 est d'ordre 2 seulement.

7. Soit $\phi_k := n(n-1)\cdots(n-k+1)K_0^{n-k}\theta(K_0)^k$. Montrer que
$$\phi_{k+1}(t) = \theta(\phi_k)(t) - k(n-k+1)t^2\phi_{k-1}(t).$$
8. Utiliser cette récurrence pour $n = 2$ pour retrouver l'équation de la question (1), en récrivant $\phi_1, \phi_2, \ldots$ au moyen de ϕ_0.
9. Plus généralement, décrire un algorithme de calcul d'une équation différentielle linéaire vérifiée par K_0^n à partir du résultat de la question (7).
10. Montrer que la complexité du calcul de l'équation différentielle par cette méthode est bornée par $\mathcal{O}(n^3)$.

EXERCICE 11 (Calcul rapide du polynôme caractéristique). On considère un réel $\theta > 2$ tel que la multiplication de deux matrices de $\mathcal{M}_n(\mathbb{K})$ peut se faire en $\mathsf{MM}(n) = n^\theta$ opérations dans $\mathbb{K}$. L'objectif de cet exercice est de prouver que le polynôme caractéristique d'une matrice « générique » de $\mathcal{M}_n(\mathbb{K})$ peut être calculé en $O(\mathsf{MM}(n))$ opérations de $\mathbb{K}$.

Une matrice $F = (f_{ij})_{1\leq i,j\leq n}$ de $\mathcal{M}_n(\mathbb{K})$ sera appelée « de type m-Frobenius » si ses $n-m$ premières colonnes ne contiennent que des éléments nuls, à l'exception des éléments $f_{m+1,1}, f_{m+2,2}, \ldots, f_{n,n-m}$ qui valent tous 1. Par exemple, les matrices de type 1-Frobenius sont les matrices compagnon.

Dans la suite de l'exercice, on suppose que la taille n est une puissance de 2 et on note $n = 2^r$. On associe à A une suite de matrices $A_r = A, A_{r-1}, \ldots, A_1, A_0$ qui lui sont toutes semblables (c'est-à-dire de la forme $P^{-1}AP$, avec P inversible). Ces matrices sont définies de manière inductive. Pour passer de A_{i+1} à A_i, on calcule $s_i = n/2^i$ matrices, notées $A_{i,1} = A_{i+1}, A_{i,2}, \ldots, A_{i,s_i} = A_i$ et définies comme suit :

- On considère la matrice U_{ij} de type 2^i-Frobenius dont les 2^i dernières colonnes coïncident avec les 2^i dernières colonnes de A_{ij}.
  On suppose que A est « suffisamment générique » pour que U_{ij} soit inversible.
- On pose $A_{i,j+1} := U_{ij}^{-1} A_{i,j} U_{ij}$.

On admettra sans preuve que la matrice A_i ainsi construite est de type 2^i-Frobenius.

1. Montrer que tous les éléments de l'inverse $D = (d_{ij})$ d'une matrice compagnon $C = (c_{ij})$ avec $c_{1n} \neq 0$ sont nuls, à l'exception des éléments suivants

$$d_{1,2} = \cdots = d_{n-1,n} = 1,\ d_{n,1} = {c_{1,n}}^{-1}, d_{n-j+1,1} = -c_{j,n}\, d_{n,1},\ \text{pour } 2 \leq j \leq n.$$

   En déduire que les éléments de D peuvent se calculer en $O(n)$ opérations dans $\mathbb{K}$.
2. Étendre le résultat de la question (1), en montrant que l'inverse d'une matrice de type m-Frobenius peut se calculer en $O(n\,\mathsf{MM}(m)/m)$ opérations de $\mathbb{K}$.
3. En déduire que le passage de A_{i+1} à A_i se fait en $O(n^2\mathsf{MM}(2^i)/4^i)$ opérations de $\mathbb{K}$.
4. Conclure que le polynôme caractéristique de A peut se calculer en $O(\mathsf{MM}(n))$ opérations de $\mathbb{K}$.

EXERCICE 12 (Résolution de systèmes de Vandermonde transposés). Soient $a_0, a_1, \ldots, a_n$ des éléments distincts de $\mathbb{K}$ et soit V la matrice de Vandermonde $V = (a_i^j)_{0\leq i,j\leq n}$. Le but de cet exercice est de montrer que, pour tout vecteur $\mathsf{b} \in \mathbb{K}^{n+1}$, on peut calculer l'unique solution $\mathsf{x} \in \mathbb{K}^{n+1}$ du système linéaire ${}^tV \cdot \mathsf{x} = \mathsf{b}$ en $\mathcal{O}(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$.

1. Donner un algorithme pour le calcul des coefficients du numérateur et du dénominateur de la fraction rationnelle $R = \sum_{i=0}^{n} \frac{1}{1-a_i X}$ en $\mathcal{O}(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$.
2. Montrer qu'à partir de R, on peut calculer en $\mathcal{O}(\mathsf{M}(n))$ opérations dans $\mathbb{K}$ les $2n$ premiers éléments de la suite $(z_i)_{i\geq 0}$ définie par $z_i = a_0^i + a_1^i + \cdots + a_n^i$.

On rappelle que, par définition, une matrice de type Hankel est constante le long de toutes ses anti-diagonales. On rappelle le théorème suivant vu au Chapitre 10 :

Si H est une matrice de Hankel inversible de taille $n+1$, alors on peut résoudre le système linéaire $H\mathsf{u} = \mathsf{b}$ en $\mathcal{O}(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$.

3. Montrer que la matrice $H := {}^tV \cdot V$ vérifie l'hypothèse de ce théorème.
4. Utiliser (1) et (2) pour estimer la complexité du calcul des éléments de H.
5. Conclure.
6. (Bonus) Écrire un algorithme résolvant ${}^tV \cdot \mathsf{x} = \mathsf{b}$, obtenu par transposition (au sens du Chapitre 12) de l'algorithme d'interpolation rapide vu au Chapitre 5.

EXERCICE 13 (Évaluation rapide de polynôme en une matrice). On rappelle que le polynôme caractéristique d'une matrice arbitraire de $\mathcal{M}_n(\mathbb{K})$ peut se calculer en $O(\mathsf{MM}(n)\log n)$ opérations de $\mathbb{K}$. Le but de cet exercice est d'étudier la complexité arithmétique du problème suivant :

> $\mathcal{E}(P, A)$: étant donnés un polynôme $P = p_0 + p_1X + \cdots + p_DX^D$ de $\mathbb{K}[X]$ et une matrice $A \in \mathcal{M}_n(\mathbb{K})$, calculer la matrice $P(A) = p_0I_n + p_1A + \cdots + p_DA^D$.

1. Montrer que $O(\mathsf{MM}(n)\sqrt{D})$ opérations de $\mathbb{K}$ suffisent pour résoudre $\mathcal{E}(P, A)$.
   Indication : Écrire $P(X)$ sous la forme $P_0(X) + P_1(X)X^d + P_2(X)(X^d)^2 + \cdots$, avec d bien choisi et $P_i(X)$ de degrés au plus $d - 1$.
2. Montrer qu'on peut améliorer ce résultat à $O(\mathsf{MM}(n)\sqrt{n} + \mathsf{M}(D))$ opérations.
   Indication : commencer par calculer le polynôme caractéristique de A.
3. Montrer qu'on peut résoudre $\mathcal{E}(X^D, A)$ en $O(\mathsf{MM}(n)\sqrt{n}+\mathsf{M}(n)\log D)$ opérations de $\mathbb{K}$.
4. Montrer que si C est une matrice compagnon, alors il est possible de résoudre $\mathcal{E}(P, C)$ en $O(n^2 + \mathsf{M}(D))$ opérations de $\mathbb{K}$.
5. Montrer que s'il existe un réel $\beta > 2$ tel que $\mathcal{E}(X^2, A)$ puisse se résoudre en n^β opérations de $\mathbb{K}$ pour toute matrice A, alors la multiplication de deux matrices quelconques de $\mathcal{M}_n(\mathbb{K})$ peut se faire en $O(n^\beta)$ opérations dans $\mathbb{K}$.
6. Montrer que si $A \in \mathcal{M}_n(\mathbb{K})$ et $\deg(P) = n$, la première colonne de $P(A)$ peut être calculée en $O(\mathsf{MM}(n)\log n)$ opérations de $\mathbb{K}$.

Dans la suite, on suppose que la matrice A est « suffisamment générique », en ce sens que la matrice $U \in \mathcal{M}_n(\mathbb{K})$, dont la ℓ^{e} colonne coïncide avec la première colonne de A^ℓ, est une matrice inversible.

7. Montrer que la matrice $C = UAU^{-1}$ est une matrice de type compagnon et qu'elle peut être calculée en $O(\mathsf{MM}(n)\log n)$ opérations de $\mathbb{K}$.
   Indication : Remarquer que l'espace vectoriel $\mathcal{V} = \mathbb{K}^n$ admet $\{A^\ell e\}_{1\le\ell\le n}$ comme base, où $e = (1, 0, \ldots, 0)^t \in \mathcal{V}$, et que U est la matrice de passage entre cette base et la base canonique de $\mathcal{V}$.
8. En déduire que, pour une matrice A « suffisamment générique », le problème $\mathcal{E}(P, A)$ peut se résoudre en $O(\mathsf{MM}(n)\log n + \mathsf{M}(D))$ opérations de $\mathbb{K}$.
9. Montrer que, pour une matrice A « suffisamment générique », on peut résoudre $\mathcal{E}(X^D, A)$ en $O(\mathsf{MM}(n)\log n + \mathsf{M}(n)\log D)$ opérations de $\mathbb{K}$.

Exercice 14 (Somme de solutions d'équations différentielles linéaires). Dans tout cet exercice, on considère deux équations différentielles

$$a_r(X)y^{(r)}(X) + \cdots + a_0(X)y(X) = 0 \tag{E1}$$

et

$$b_s(X)y^{(s)}(X) + \cdots + b_0(X)y(X) = 0 \tag{E2}$$

et on recherche une équation différentielle vérifiée par toutes les sommes $h = f + g$ où f est solution de (E1) et g une solution de (E2). On suppose que les a_i et les b_i sont des polynômes de $\mathbb{Q}[X]$ tels que $\deg a_i \leq m$ et $\deg b_i \leq n$, et qu'aucun de r, s, a_r et b_s n'est nul. On note $R = r + s$ et $D = \max\{m, n\}$.

Préambule

1. Montrer que h est solution d'une équation $\mathcal{R}(y) = 0$ d'ordre au plus R.

Les équations (E1) et (E2) permettent de récrire toute dérivée d'ordre arbitraire de f ou de g comme combinaison linéaire des $r + s$ fonctions f, …, $f^{(r-1)}$, g, …, et $g^{(s-1)}$. L'espace vectoriel sur $\mathbb{Q}(X)$ engendré par les dérivées d'ordre arbitraire f, f', g'', …, g, g', g'', …, a donc dimension au plus R.

Dans cet espace vectoriel, h, …, $h^{(R)}$ sont liées, et une relation de liaison non triviale fournit $\mathcal{R}$ après avoir chassé les dénominateurs.

Première méthode : recombinaison des équations

Pour $i \in \mathbb{N}$, $j \in \mathbb{N}$, $k \in \{1, 2\}$, on note $\mathcal{M}_{i,j}^{(k)}(y) = 0$ l'équation obtenue en dérivant j fois l'équation (k) puis en multipliant le résultat par X^i. On voit $\mathcal{M}_{i,j}^{(k)}$ comme une application de la variable y. Pour un entier $\delta \geq m + n$, on considère :

$$S^{(\mathrm{E1})} = \left\{ \mathcal{M}_{i,j}^{(\mathrm{E1})} : i \leq \delta - m,\ j \leq s \right\} \quad \text{et} \quad S^{(\mathrm{E2})} = \left\{ \mathcal{M}_{i,j}^{(\mathrm{E2})} : i \leq \delta - n,\ j \leq r \right\}.$$

2. Déterminer des bornes supérieures $d_{\max}$ et $r_{\max}$ telles que la famille des $X^i y^{(j)}$ pour $0 \leq i \leq d_{\max}$ et $0 \leq j \leq r_{\max}$ permette d'exprimer tous les $\mathcal{R}(y)$ quand $\mathcal{R}$ décrit $S^{(\mathrm{E1})} \cup S^{(\mathrm{E2})}$.
3. On veut résoudre

$$\sum_{\mathcal{R} \in S^{(\mathrm{E1})}} \lambda_{\mathcal{R}} \mathcal{R}(y) = \sum_{\mathcal{R} \in S^{(\mathrm{E2})}} \mu_{\mathcal{R}} \mathcal{R}(y) \tag{R}$$

en des coefficients λ et μ dans $\mathbb{Q}$. En égalant les coefficients sur la base des $X^i y^{(j)}$, compter le nombre d'équations et le nombre d'inconnues. Choisir δ pour assurer l'existence de solutions non triviales.

4. Donner un algorithme pour le calcul d'une équation pour les sommes $h = f + g$ et justifier sa correction.
5. Évaluer en termes de R, D et avec des $\mathcal{O}(\cdot)$ le coût arithmétique du calcul des $\mathcal{M}_{i,j}^{(k)}(y)$ de proche en proche et celui de la résolution du système. Conclure sur la complexité arithmétique de la méthode.

Deuxième méthode : réductions par les équations

6. Pour une solution f de (E1) et une combinaison linéaire $u = p_0 f + \cdots + p_{r-1} f^{(r-1)}$ à coefficients des polynômes de degré maximal δ, montrer que $a_r u'$ s'exprime comme une combinaison linéaire de f, …, $f^{(r-1)}$ pour des coefficients polynomiaux dont on bornera le degré en fonction de δ.
7. En déduire que pour tout entier $i \geq r - 1$, le produit $a_r^{i+1-r} f^{(i)}$ s'exprime comme une combinaison linéaire $p_{i,0} f + \cdots + p_{i,r-1} f^{(r-1)}$ pour des polynômes $p_{i,j}$ dont on bornera le degré.

8. Énoncer un résultat analogue pour $b_s^{i+1-s} g^{(j)}$ et pour des polynômes $q_{i,j}$.
9. On veut résoudre

$$\text{(P)} \qquad \sum_{i=0}^{R} c_i(X) f^{(i)} = 0 = \sum_{i=0}^{R} c_i(X) g^{(i)}$$

pour des polynômes c_i dans $\mathbb{Q}[X]$ communs aux deux équations. Montrer qu'en multipliant le membre de gauche de (P) par une puissance convenable de a_r et le membre de droite par une puissance convenable de b_s, et en réduisant par les équations (E1) et (E2), on obtient un système d'équations linéaires sur les c_i à coefficients polynomiaux. Indiquer les dimensions de ce système et borner le degré de ses coefficients en termes de R et D.

10. Donner un algorithme pour le calcul d'une équation pour les sommes $h = f + g$ et justifier sa correction.
11. Évaluer en termes de R, D et avec des $\tilde{O}(\cdot)$ (c'est-à-dire à des facteurs logarithmiques près) le coût arithmétique de cet algorithme.
12. Comparer avec la première méthode.

Exercice 15 (Quartique de Macaulay). On considère l'idéal homogène I engendré par les quatre polynômes

$$f_1 = X_3^3 - X_0^2 X_2, \quad f_2 = X_2 X_3 - X_0 X_1,$$
$$f_3 = X_2^3 - X_1^2 X_3, \quad f_4 = X_1 X_3^2 - X_0 X_2^2,$$

dans $\mathbb{Q}[X_0, X_1, X_2, X_3]$. On ordonne les monômes par l'ordre lexicographique induit par $X_3 < X_2 < X_1 < X_0$ et on convient de prendre comme monôme dominant d'un polynôme non nul le *plus petit* de ses monômes.

Pour un entier $n > 0$, on identifie librement un monôme $X_0^{e_0} \cdots X_n^{e_n}$ et le $(n+1)$-uplet d'entiers $(e_0, \ldots, e_n)$. Pour un idéal J de $\mathbb{Q}[X_0, \ldots, X_n]$, on note $E(J)$ l'ensemble des monômes dominants des éléments non nuls de J, ou de façon équivalente, la partie stable correspondante dans $\mathbb{N}^{n+1}$.

1. Calculer une base standard de I relativement à l'ordre indiqué. Donner un système générateur minimal de $E(I) \subset \mathbb{N}^4$.
2. Rappelons que la fonction de Hilbert HF_E associée à une partie stable E de $\mathbb{N}^{n+1}$ est définie par :

$$HF_E(s) = \mathrm{Card}\Big\{(e_0, \ldots, e_n) \in \mathbb{N}^{n+1} \;\Big|\; (e_0, \ldots, e_n) \notin E,\ e_0 + \cdots + e_n = s\Big\}.$$

Pour i dans $\mathbb{N}$, on considère la partie de $E(I)$ constituée des éléments dont la première coordonnée X_0 vaut i et on identifie cette partie à une partie E_i de $\mathbb{N}^3$.

   (a) Donner un système générateur minimal de E_0 et dessiner E_0. Calculer la fonction de Hilbert, le polynôme de Hilbert, la dimension et le degré de E_0.
   (b) Que vaut E_i pour un i général ?
   (c) L'application induite par la multiplication par X_0 dans $\mathbb{Q}[X_0, X_1, X_2, X_3]/I$ est-elle injective ?

3. Calculer la fonction de Hilbert, le polynôme de Hilbert, la dimension et le degré de $E(I)$.
4. Rappelons que l'espace projectif $\mathbb{P}^n_{\mathbb{C}}$ s'identifie à l'ensemble des $(n+1)$-uplets non nuls $(u_0 : \ldots : u_n)$ de $\mathbb{C}^{n+1}$ vus à multiplication près par un scalaire non nul. On souhaite maintenant étudier le lieu des zéros $Z(I)$

de l'idéal homogène I dans l'espace projectif $\mathbb{P}^3_{\mathbb{C}}$. Dans cette question, on considère X_1 comme indéterminée d'homogénéisation, et non comme d'habitude X_0.

(a) Quel est l'ensemble des zéros à l'infini, c'est-à-dire de la forme $(x_0 : 0 : x_2 : x_3)$?

(b) Déshomogénéiser le système polynomial initial en faisant $x_1 = 1$ et décrire l'ensemble algébrique affine de $\mathbb{C}^3$ ainsi obtenu.

(c) Conclure en décrivant $Z(I)$ et comparer avec les invariants calculés dans la question 3.

Exercice 16 (Diagonales de séries différentiellement finies). Étant donnée une série double différentiellement finie

$$f = \sum_{i,j\in\mathbb{N}} c_{i,j} x^i y^j, \tag{1}$$

cet exercice vise à donner un algorithme de calcul d'une équation différentielle pour la diagonale

$$\Delta(f) = \sum_{i\in\mathbb{N}} c_{i,i} t^i,$$

laquelle est aussi différentiellement finie. Par exemple, la série

$$\frac{1}{1-x-y} = \sum_{i,j\in\mathbb{N}} \binom{i+j}{i} x^i y^j, \tag{2}$$

qui vérifie deux équations différentielles linéaires d'ordre 1, l'une en x, l'autre en y, a pour diagonale

$$\Delta\left(\frac{1}{1-x-y}\right) = \sum_{i\in\mathbb{N}} \binom{2i}{i} t^i = \frac{1}{\sqrt{1-4t}}, \tag{3}$$

donnée par $(1-4t)y' - 2y = 0$.

En préambule, nous donnons quelques résultats à admettre et utiliser librement sur les deux natures de séries formelles un peu particulières utilisées dans cet énoncé. Ces résultats établissent que les séries considérées peuvent être manipulées formellement sans ambages (multiplications, dérivations, substitutions, identifications de termes).

Pour des coefficients $c_{i,j}$ dans un corps C, les séries du type (1) forment une C-algèbre notée $C[[x,y]]$, stable par les dérivations par x et y, dont le corps des fractions $F_{x,y}$ contient celui des fractions rationnelles, $C(x,y)$, et est lui-même stable par les deux dérivations. Une série f donnée par (1) étant aussi dans $F_{x,y}$, on dit qu'elle est différentiellement finie lorsque ses dérivées mixtes $\partial_x^i \partial_y^j(f)$ engendrent un sous-espace vectoriel de $F_{x,y}$ de dimension finie sur $C(x,y)$.

Les séries

$$g = \sum_{i\in\mathbb{Z},\ j\in\mathbb{N},\ i+j\geq k} c_{i,j} s^i t^j,$$

où $k \in \mathbb{Z}$ ne dépend que de g, forment une autre C-algèbre, notée $S_{s,t}$, stable par les dérivations par s et t, dont le corps des fractions $F_{s,t}$ contient celui des fractions rationnelles, $C(s,t)$, et est lui-même stable par les deux dérivations. Une série g est dite différentiellement finie lorsque ses dérivées mixtes $\partial_s^i \partial_t^j(f)$ engendrent un sous-espace vectoriel de $F_{s,t}$ de dimension finie sur $C(s,t)$.

On admet que la substitution $x = t/s$, $y = s$ donne une application bien définie de $C[[x,y]]$ dans $S_{s,t}$.

1. On considère l'application C-linéaire ϕ de $C[[x,y]]$ dans $S_{s,t}$ définie sur f donnée par (1) par
$$\phi(f) = \frac{1}{s} f\left(\frac{t}{s}, s\right) = \sum_{i,j\in\mathbb{N}} c_{i,j} s^{j-i-1} t^i.$$
On note
$$\delta(f) = \sum_{i\in\mathbb{N}} c_{i,i} x^i y^i = \Delta(f)(xy).$$
Vérifier que
$$\phi\big(\delta(f)\big) = \frac{1}{s}\Delta(f).$$
2. Pour une série $f \in C[[x,y]]$, des fractions rationnelles $r_{i,j}(s,t)$ en nombre fini et un opérateur différentiel linéaire $L(t,\partial_t)$, on suppose vérifiée l'identité
$$L(t,\partial_t)(g) = \partial_s\left(\sum_{i,j} r_{i,j}(s,t)\partial_s^i\partial_t^j(g)\right)$$
quand $g = \phi(f)$. Montrer que L annule la diagonale $\Delta(f)$.
3. Pour $g = \phi(f)$, exprimer $\partial_s(g)$ et $\partial_t(g)$ comme les images par ϕ d'expressions linéaires en f et ses dérivées premières, qu'on exprimera sous la forme
$$\partial_s(g) = \phi\big(L_s(f)\big) \quad \text{et} \quad \partial_t(g) = \phi\big(L_t(f)\big).$$
Pour valider son résultat, on pourra observer que L_s et L_t commutent.
4. En déduire que si f est différentiellement finie, il en est de même de g.
5. Proposer un algorithme qui, étant donné un idéal annulateur pour f, calcule un idéal annulateur pour g.
6. En déduire un algorithme à base de la version différentielle de l'algorithme de Zeilberger qui, étant donné un idéal annulateur pour f, calcule un polynôme tordu $L(t,\partial_t)$ annulateur de $\Delta(f)$.
7. Montrer que $g = \phi(f) = 1/(s - t - s^2)$.
8. Appliquer à g l'algorithme de Zeilberger différentiel et produire une équation différentielle d'ordre 1 en t sur la diagonale (3).

Exercice 17 (Intégration définie de fractions rationnelles par réduction de Hermite). À l'exclusion de la première question, d'ordre plus général et indépendante de la suite, cet exercice s'intéresse à l'intégration définie rapide de fractions rationnelles bivariées.

Pour la première question seulement, on considère l'intégration de fonctions f hyperexponentielles en deux variables x et y, c'est-à-dire dont les deux dérivées logarithmiques sont données par des fractions rationnelles :
$$\frac{\frac{\partial f}{\partial x}(x,y)}{f} \in \mathbb{Q}(x,y) \qquad \text{et} \qquad \frac{\frac{\partial f}{\partial y}(x,y)}{f} \in \mathbb{Q}(x,y).$$

1. Utiliser les algorithmes du cours pour décrire un analogue de l'algorithme de Zeilberger réalisant la création télescopique pour ce cadre, c'est-à-dire pour calculer un opérateur différentiel et une fraction rationnelle,
$$L = \sum_{i=0}^{\rho} \eta_i(x)\partial_x^i \in \mathbb{Q}(x)\langle\partial_x; \mathrm{id}, d/dx\rangle \qquad \text{et} \qquad \phi \in \mathbb{Q}(x,y),$$
vérifiant la relation $L(f) = \partial_y(\phi f)$.

Observons que cette approche ne permet pas de calculer L sans calculer ϕ, alors que ce dernier n'est pas toujours utilisé dans les applications, et par ailleurs que la taille de ce ϕ s'avère souvent être le facteur limitant dans les calculs.

On s'intéresse maintenant à l'intégration dans le cas de fonctions f qui soient des fractions rationnelles, afin d'obtenir un algorithme spécialisé de bonne complexité. Dorénavant, f est une fraction rationnelle de la forme p/q pour des polynômes non nuls p et q de $\mathbb{Q}[x,y]$ de degrés partiels en x et y bornés respectivement par d_x et d_y et tels que $\deg_y(p) < \deg_y(q)$. On suppose de plus que le polynôme $q \in \mathbb{Q}[x,y]$ est sans facteur carré (relativement à y), au sens où le pgcd de q et $\partial q/\partial y$, dans $\mathbb{Q}(x)[y]$, est 1. On rappelle qu'il existe alors une relation de Bézout de la forme

$$uq + v\frac{\partial q}{\partial_y} = 1$$

pour des polynômes bivariés u et v dans $\mathbb{Q}(x)[y]$, vérifiant les contraintes $\deg_y(u) < d_y - 1$ et $\deg_y(v) < d_y$. La notation $\mathbb{Q}_{<d}[x]$, resp. $\mathbb{Q}_{\leq d}[x]$, dénote l'ensemble des polynômes de degré strictement inférieur, resp. inférieur, à d. On admettra l'existence d'un algorithme qui, étant donnée une matrice $r \times r$ à coefficients polynomiaux de $\mathbb{Q}[x]$ de degré au plus n, calcule une base du noyau, constituée de vecteurs de polynômes, en $\tilde{O}(r^\omega n)$ opérations dans $\mathbb{Q}$.

2. (a) Pour $h \in \mathbb{Q}(x)[y]_{<2d_y}$, montrer l'existence de polynômes A et a dans $\mathbb{Q}(x)[y]_{<d_y}$ satisfaisant à

$$\frac{h}{q^2} = \partial_y\left(\frac{A}{q}\right) + \frac{a}{q}.$$

(b) Pour $\deg_y(h) < md_y$, généraliser à l'existence de polynômes $A \in \mathbb{Q}(x)[y]_{<(m-1)d_y}$ et $a \in \mathbb{Q}(x)[y]_{<d_y}$ satisfaisant à

$$\frac{h}{q^m} = \partial_y\left(\frac{A}{q^{m-1}}\right) + \frac{a}{q}.$$

On admettra l'unicité de a dans l'écriture précédente. Cette réduction de h/q^m à a/q est la réduction de Hermite.

3. On considère l'application ϕ qui à A et a dans $\mathbb{Q}(x)[y]_{<d_y}$ associe

$$\phi(A,a) = q\frac{\partial A}{\partial y} - \frac{\partial q}{\partial y}A + qa$$

dans $\mathbb{Q}(x)[y]_{<2d_y}$, ainsi que sa matrice $\mathcal{H}$ sur la base monomiale. Expliciter les dimensions de cette matrice $\mathcal{H}$ ainsi qu'une borne sur le degré en x de ses coefficients.

4. On admettra que l'équation $\phi(A,a) = h$ admet pour tout $h \in \mathbb{Q}[x,y]$ avec $\deg_y(h) < 2d_y$ une unique solution (A,a) pour des polynômes de $\mathbb{Q}(x)[y]$ vérifiant $\deg_y(A) < d_y$ et $\deg_y(a) < d_y$. Ces polynômes peuvent donc se récrire avec un dénominateur commun δ sous la forme $A = B/\delta$ et $a = b/\delta$. On pose $\mu = \deg_x(h)$. Utiliser les formules de Cramer pour expliciter un bon choix de δ et donner des bornes sur les degrés en x de B, b et δ.

5. (a) Donner un algorithme qui, étant donné un polynôme $h \in \mathbb{Q}[x,y]$ vérifiant $\deg_y(h) < 2d_y$, détermine B et b dans $\mathbb{Q}(x)[y]$ en fonction de h, u, v et q, de sorte que

$$\frac{h}{q^2} = \partial_y\left(\frac{B}{\delta q}\right) + \frac{b}{\delta q},$$

avec $\deg_y(b) < d_y$.

(b) Estimer la complexité de cet algorithme en supposant que B est renvoyé sans nécessairement développer les sommes et les produits, mais que b est renvoyé sous forme développée et réduite.

6. (a) Par réduction de Hermite, chaque $\partial_x^i(f)$ peut s'écrire sous la forme
$$\partial_x^i(f) = \partial_y(g_i) + \frac{r_i}{q}$$
pour $g_i \in \mathbb{Q}(x,y)$ et un unique $r_i \in \mathbb{Q}(x)[y]$ tels que $\deg_y(r_i) < \deg_y(q)$. Donner les étapes d'un algorithme incrémental donnant r_{i+1} à partir de r_i et expliciter le g_{i+1} correspondant en fonction de g_i.
(b) Donner un multiple explicite du dénominateur commun des coefficients de r_i et établir une borne sur le degré en x des numérateurs correspondants.
(c) En déduire la complexité du calcul de tous les r_i pour $0 \le i \le \rho$.

7. (a) Montrer que $\{r_0, \ldots, r_{d_y}\}$ est liée et en déduire un algorithme pour calculer un opérateur différentiel et une fraction rationnelle,
$$L = \sum_{i=0}^{\rho} \eta_i(x)\partial_x^i \in \mathbb{Q}(x)\langle \partial_x; \mathrm{id}, d/dx\rangle \qquad \text{et} \qquad g \in \mathbb{Q}(x,y),$$
vérifiant la relation $L(f) = \partial_y(g)$. Décrire l'algorithme complet.
(b) Évaluer sa complexité.

Exercice 18 (Rencontre entre une hyperbole et deux droites). Soit I l'idéal de $\mathbb{Q}[x,y]$ engendré par les deux polynômes $y^2 - x^2$ et $xy - 1$. On ordonne les monômes de $\mathbb{Q}[x,y]$ d'abord par degré croissant, puis en cas d'égalité de degré, par degré croissant en y :
$$1 < x < y < x^2 < xy < y^2 < x^3 < x^2y < xy^2 < y^3 < \ldots$$
Le monôme dominant d'un polynôme non nul sera le plus grand pour cet ordre.

1. Calculer une base standard de I pour cet ordre et donner une base monomiale du $\mathbb{Q}$-espace vectoriel $\mathbb{Q}[x,y]/I$.
2. La multiplication par x induit un endomorphisme linéaire de $\mathbb{Q}[x,y]/I$. Calculer sa matrice dans la base ci-dessus, puis son polynôme caractéristique et ses valeurs propres. Que représentent ces dernières ?
3. Soit J un idéal de $\mathbb{Q}[x_1, \ldots, x_n]$ définissant une sous-variété algébrique V de $\mathbb{C}^n$ de dimension zéro, de degré r, et composée de r points distincts. Soit f un polynôme de $\mathbb{C}[x_1, \ldots, x_n]$ séparant les points de V, c'est-à-dire que les images par f de deux points distincts de V restent distincts. On nomme $P(T)$ le polynôme caractéristique de l'endomorphisme linéaire F de $\mathbb{Q}[x_1, \ldots, x_n]/J$ induit par la multiplication par f, et $Q(T)$ le polynôme dont les racines sont les images par f des points de V.
   (a) Quels sont les degrés de P et Q ?
   (b) Soit x un point de V. Montrer que $f(x)$ est une valeur propre de F en construisant explicitement un vecteur propre par interpolation.

Exercice 19 (Problème des moments pour une suite hypergéométrique). Soit $(C_n)_{n\ge 0} = (1, 1, 2, 5, 14, 42, \ldots)$ la suite de terme général $C_n = \frac{1}{n+1}\binom{2n}{n}$. Le but de cet exercice est de résoudre le *problème des moments* pour cette suite, c'est-à-dire de trouver deux réels a et b, et une fonction continue $\mu : [a,b] \to \mathbb{R}_+$ tels que

$$C_n = \int_a^b \mu(x) x^n \, dx \quad \text{pour tout } n \ge 0. \tag{P}$$

Une série algébrique. Soit $C(z)$ la série génératrice $C(z) = \sum_{n\ge 0} C_n z^n$.

1. Expliciter une récurrence linéaire satisfaite par la suite (C_n).
2. Montrer que tout approximant de Padé-Hermite de type $(1,1,1)$ du vecteur de séries ${}^t(1, C, C^2)$ est proportionnel à $(1,-1,z)$. En déduire un polynôme $P(z,w) \in \mathbb{Q}[z,w]$ tel que $P(z,C(z)) = 0 \bmod z^5$.

   Pour cela, exprimer le problème d'approximation en termes d'algèbre linéaire. Expliciter la matrice du système linéaire associé. Est-elle structurée ? Quel est son rang de déplacement ?
3. Montrer qu'il existe une unique série $S(z) \in \mathbb{Q}[[z]]$ telle que $P(z,S(z)) = 0$. Combien vaut $S(z) \bmod z^5$?
4. Écrire un algorithme à base d'itérations de Newton, prenant en entrée un entier positif κ, et renvoyant en sortie les $N = 2^\kappa$ premiers coefficients de S.
5. Montrer que S vérifie une équation différentielle linéaire, que l'on explicitera. On exhibera d'abord une relation de Bézout dans $\mathbb{Q}(z)[w]$ entre P et $\frac{\partial P}{\partial w}$.
6. En déduire que les coefficients de $S(z) = \sum_{n\geq 0} s_n z^n$ vérifient la récurrence
$$(n+2)s_{n+1} - (4n+2)s_n = 0, \quad \text{pour tout } n \geq 0.$$
7. Conclure que la série $C(z)$ est algébrique, et est racine du polynôme $P(z,w)$.

Moments et résultants. Soit $G(z)$ la série $G(z) = C(1/z)/z$ dans $\mathbb{Q}[[1/z]]$.

Nous admettons que G définit une fonction sur $\mathbb{C} \setminus [0,4]$, et que la solution du problème (P) est donnée par la *formule d'inversion de Stieltjes* :
$$\mu(x) = -\frac{\psi(x)}{\pi}, \quad \text{où} \quad \psi(x) = \lim_{y\to 0^+} \Big(\operatorname{Im} G(x+iy)\Big).$$

8. Montrer que $G(z)$ est une fonction algébrique, en exhibant un polynôme annulateur $K(z,w) \in \mathbb{Q}[z,w]$.
9. Écrivons $G(x+iy)$ sous la forme $A(x,y)+iB(x,y)$, avec A et B des fonctions réelles, et notons $\varphi(x)$ et $\psi(x)$ les limites en $y = 0^+$ de A et de B. Justifier l'égalité $K(x, \varphi(x)+i\psi(x)) = 0$. En déduire l'algébricité de $\varphi(x)$ et de $\psi(x)$.
10. Déterminer, à l'aide d'un résultant, un polynôme annulateur de $\psi(x)$. En déduire une forme explicite de $\mu(x)$.

Preuve par télescopage créatif. Nous allons prouver l'identité

(M) $$C_n = \frac{1}{2\pi} \int_0^4 x^n \sqrt{\frac{4-x}{x}}\, dx.$$

11. On veut montrer que la fonction $U(n,x) = x^n \sqrt{\frac{4-x}{x}}$ est solution d'une équation fonctionnelle linéaire de la forme

(C) $$(a(n)S_n + b(n)) \cdot U = D_x \cdot (r(n,x)U),$$

où S_n est l'opérateur de décalage, D_x est l'opérateur de dérivation, et où $r \in \mathbb{Q}(n,x)$ et $a, b \in \mathbb{Q}(n)$ sont des fractions rationnelles inconnues. (Ici, la notation « $\cdot$ » désigne l'application d'un opérateur à une fonction.)

Afin de déterminer a, b et r, on parcourra les étapes suivantes :

(a) Expliciter deux opérateurs qui annulent U, de la forme $S_n - \alpha(n,x)$ et $D_x - \beta(n,x)$, avec $\alpha, \beta \in \mathbb{Q}(n,x)$.

(b) Utiliser ces deux opérateurs pour montrer que l'équation (C) est équivalente à l'équation différentielle linéaire inhomogène paramétrée d'ordre 1
$$\frac{\partial r}{\partial x}(n,x) + \frac{nx - 4n + 2}{x(x-4)}\, r(n,x) = a(n)\, x + b(n).$$

(c) Montrer que r n'a pas de pôle en $x = 0$ et $x = 4$. En déduire que r est un polynôme en x, à coefficients dans $\mathbb{Q}(n)$.

(d) Borner le degré (en x) de r, et montrer que g est divisible par $x(x-4)$ dans $\mathbb{Q}(n)[x]$.

(e) Expliciter g, puis a et b, et conclure.

12. En déduire une récurrence linéaire vérifiée par la suite $f_n = \int_0^4 U(n,x)\,dx$.

13. En admettant que $\displaystyle\int_0^4 \sqrt{\frac{4-x}{x}} = 2\pi$, conclure la preuve de l'égalité (M).

Compléments.

14. Quelle est la complexité arithmétique et binaire du calcul des N premiers termes de la série $S(z)$ en utilisant l'itération de Newton donnée en (4) ?

15. Quelle est la taille binaire de C_n, et quel est le coût binaire de son calcul, en utilisant la récurrence obtenue en (1) et un algorithme : (i) *naïf* ; (ii) par *scindage binaire* ; (iii) par *« pas de bébés, pas de géants »* ?

Exercice 20 (Relation de contiguïté entre fonctions de Bessel). Les fonctions de Bessel de première espèce, $J_n(x)$, sont solutions de l'équation différentielle et de l'équation mixte différentielle et de récurrence

$$x^2y_n''(x) + xy_n'(x) + (x^2 - n^2)y_n(x) = 0 \quad \text{et} \quad xy_n'(x) + xy_{n+1}(x) - ny_n(x) = 0.$$

1. Écrire dans l'algèbre de Ore $\mathbb{Q}(n,x)\langle\partial_n, \partial_x; S_n, I, 0, d/dx\rangle$ les polynômes tordus associés à ces équations, et montrer qu'ils constituent une base de Gröbner pour l'ordre lexicographique induit par $\partial_n > \partial_x$.

2. Soit I l'idéal engendré par les polynômes tordus de la question précédente. Justifier, avec le moins de calculs possible, l'existence d'un polynôme de I non nul et indépendant de ∂_x. Quel est le degré minimal en ∂_n d'un tel polynôme ?

3. Indiquer une méthode de calcul de ce polynôme.

Exercice 21 (Équations différentielles pour les fonctions algébriques). On s'intéresse aux fonctions α solutions d'une équation polynomiale $P\big(x, \alpha(x)\big) = 0$ pour un polynôme $P(X,Y) \in \mathbb{Q}[X,Y]$ de degrés partiels D_X en X et D_Y en Y, avec $D_X \geq 1, D_Y > 1$. Nous voulons trouver des équations différentielles linéaires vérifiées par ces fonctions algébriques α.

On note P_X et P_Y les dérivées de P relativement à X et Y, respectivement. Pour éviter que P n'ait des facteurs multiples, on fait l'hypothèse que P et P_Y sont premiers entre eux.

Résolvante de Cockle. Un premier algorithme, attribué à Cockle (1861), représente les dérivées $\alpha^{(k)}$ comme des polynômes de degrés au plus $D_Y - 1$ en α et se ramène à de l'algèbre linéaire sur $\mathbb{Q}(X)$. Il passe par une première famille de polynômes W_k, introduits ci-dessous.

1. Montrer la relation $\alpha' = -P_X(x,\alpha)/P_Y(x,\alpha)$.

2. Montrer l'existence de polynômes $W_k(X,Y)$ tels que $\alpha^{(k)} = W_k(x,\alpha)/P_Y(x,\alpha)^{2k-1}$ en établissant une récurrence sur les polynômes W_k.

3. Que vaut W_1 ?

4. Montrer les inégalités $\deg_X W_k \leq (2D_X - 1)k - D_X$ et $\deg_Y W_k \leq 2(D_Y - 1)k - D_Y + 2$.

Cockle définit une deuxième suite $(V_k)_{k\geq 0}$ de polynômes de $\mathbb{Q}(X)[Y]$ de degrés au plus $D_Y - 1$ en Y, de sorte que, pour $k \geq 1$, $\alpha^{(k)} = W_k(x,\alpha)/P_Y(x,\alpha)^{2k-1} = V_k(x,\alpha)$, et par ailleurs, $V_0 = 1$.

5. Justifier que P_Y est inversible modulo P et indiquer comment se calcule V_1.
6. Donner un procédé de calcul itératif des V_k.
7. En déduire un algorithme par algèbre linéaire sur $\mathbb{Q}(X)$ pour calculer une équation différentielle linéaire d'ordre minimal annulant toute solution $\alpha(x)$ de $P\big(x, \alpha(x)\big) = 0$.
8. Soit r l'ordre de la sortie de cet algorithme. Donner une borne simple sur cet ordre, valable pour tout P.

On s'intéresse à la complexité de cette approche en nombre d'opérations dans $\mathbb{Q}$ et on commence par donner des bornes sur les degrés des résultats intermédiaires. Tout polynôme Q de $\mathbb{Q}(X)[Y]$ peut s'écrire sous la forme $q(X)^{-1}\,\bar{Q}(X,Y)$ pour $\bar{Q}$ dans $\mathbb{Q}[X,Y]$. On appellera degré de Q en X le maximum des degrés en X de q et $\bar{Q}$. On note $p(X)$ le coefficient dominant de P.

On admettra l'existence de polynômes A et B de $\mathbb{Q}[X,Y]$ tels que $\mathrm{Res}_Y(P, P_Y) = AP + BP_Y$, et tels que les degrés en Y de A et B sont dans $O(D_Y)$ et les degrés en X sont dans $O(D_X D_Y)$.

9. Pour un polynôme $Q(X,Y)$ de $\mathbb{Q}[X,Y]$, de degrés respectifs δ_X et δ_Y en X et Y, on considère sa division euclidienne par P sous la forme $p^{\delta_Y - D_Y} Q = UP + V$, pour U et V dans $\mathbb{Q}(X)[Y]$, avec V de degré en Y inférieur à δ_Y. Expliquer que U et V sont en réalité des polynômes de $\mathbb{Q}[X,Y]$. Montrer que le reste V vérifie
$$\deg_X V \leq \delta_X + (\delta_Y - D_Y) D_X.$$
10. Pour un polynôme $Q(X,Y)$ de $\mathbb{Q}(X)[Y]$, de degrés respectifs δ_X et δ_Y en X et Y, expliciter une borne sur le degré en X du reste de Q modulo P.
11. Déduire du calcul des V_k que le degré en X de V_k est dans $O(kD_X D_Y)$ pour tout $k \geq 1$.

On passe maintenant à l'étude de complexité proprement dite. On pourra se contenter de donner des bornes sous la forme de $\tilde{O}(\,)$ de quantités polynomiales en D_X, D_Y et k.

12. Donner en la justifiant la complexité du calcul de V_{k+1} à partir de V_k.
13. Décrire les systèmes linéaires considérés à la question 7 : quelles sont leurs tailles et les degrés de leurs coefficients ? Donner et justifier la complexité de leur résolution.
14. Conclure en donnant une borne de complexité pour l'algorithme de Cockle. Quelle est la taille arithmétique totale de sa sortie ?

Par conception, l'algorithme de Cockle recherche une équation d'ordre minimal. En relâchant cette contrainte sur l'ordre, on peut grandement abaisser la taille arithmétique de la sortie.

Approche par télescopage créatif. On note $F = YP_Y/P$ et on admet la formule suivante, qui voit α comme un résidu :
$$\alpha(x) = \frac{1}{2i\pi} \oint F(x,y)\, dy$$
pour un contour d'intégration qui tourne autour d'$\alpha(x)$ sans contenir aucun autre zéro de $P(x,Y)$.

On rappelle que le résidu d'une fraction rationnelle $S(Y)$ en un de ses pôles y d'ordre ρ est le coefficient c_{-1} de $1/(Y-y)$ dans le développement
$$S(Y) = \frac{c_{-\rho}}{(Y-y)^\rho} + \cdots + \frac{c_{-1}}{Y-y} + \sum_{n \geq 0} c_n \cdot (Y-y)^n.$$

On considère l'algèbre de Ore $\mathbb{Q}(X,Y)\langle\partial_X,\partial_Y;\mathrm{id},\mathrm{id},d/dX,d/dY\rangle$ que l'on fait agir sur les fractions rationnelles. Soit $\Lambda(X,Y,\partial_X,\partial_Y) = A(X,\partial_X) + \partial_Y\, B(X,Y,\partial_X,\partial_Y)$ un polynôme tordu.

1. Que vaut le résidu de $\partial_Y(S)$ en l'un quelconque de pôles y de S ?
2. Si Λ annule une fraction rationnelle S, montrer que A annule tout résidu de S en l'un quelconque de ses pôles y.

On veut maintenant construire un polynôme tordu Λ annulant F par recombinaison linéaire.

3. Montrer que les fractions rationnelles
$$F_{i,j,k} = X^i\partial_X^j\partial_Y^k(F), \qquad 0 \le i \le N,\ 0 \le j+k \le M$$
s'écrivent comme combinaisons linéaires sur $\mathbb{Q}$ des éléments
$$E_{i,j} = \frac{X^iY^j}{P^{M+1}}, \qquad 0 \le i \le N + D_X\,(M+1),\ 0 \le j \le D_Y\,(M+1).$$
4. Pour des entiers N et M fixés, combien y a-t-il de $F_{i,j,k}$ et combien y a-t-il de $E_{i,j}$?
5. Fixer $N = 3D_XD_Y - 1$ et $M = 6D_Y - 1$ et montrer l'existence de Λ non nul qui annule F.

On s'intéresse maintenant à la complexité du calcul de Λ.

6. Décrire un calcul de l'ensemble des $F_{i,j,k}$ exprimés en termes des $E_{i,j}$ et en donner la complexité.
7. Indiquer comment en déduire Λ et donner la complexité de cette étape.
8. Quelle est la complexité globale du calcul de Λ ? Quelle est sa taille arithmétique totale ?

Dans les bons cas, on trouve $A \neq 0$ qui annule α ; quand A est nul, on peut montrer l'existence d'un autre annulateur de α dont les degrés ne sont pas plus que le double de ceux de A.

Exercice 22 (Transporteur de deux idéaux). Soit $\mathbb{K}$ un corps. Soient I et J deux idéaux de $R := \mathbb{K}[x_1,\ldots,x_n]$. On définit le *transporteur* de J dans I, noté $(I:J)$, comme $(I:J) := \{f \in R \mid fJ \subseteq I\}$.

1. Montrer que $(I:J)$ est un idéal.
2. Montrer que pour tout idéal K de R l'inclusion $KJ \subseteq I$ est équivalente à $K \subseteq (I\ :\ J)$.
3. Montrer que si $J \subseteq I$ alors $(I:J) = R$.
4. Montrer que $(I:R) = I$.
5. Soient J_1 et J_2 deux idéaux de R. Montrer que $(I\ :\ (J_1+J_2)) = (I\ :\ J_1) \cap (I:J_2)$.
6. Montrer que $(I:(r)) = \dfrac{1}{r}(I\cap(r))$, où $\dfrac{1}{r}(I\cap(r))$ représente $\{\dfrac{f}{r} \mid f \in I\cap(r)\}$.
7. Proposer un algorithme, utilisant un calcul de base standard, pour calculer $I\cap J$ lorsque I et J sont donnés par un ensemble de polynômes générateurs $I = (f_1,\ldots,f_s)$ et $J = (g_1,\ldots,g_r)$. On pourra, par exemple, considérer l'idéal $K := (tf_1,\ldots,tf_s,(1-t)g_1,\ldots,(1-t)g_r)$ de $\mathbb{K}[x_1,\ldots,x_s,t]$, où t est une nouvelle variable.
8. Lorsque I est donné par un ensemble de polynômes générateurs $I = (f_1,\ldots,f_s)$, en déduire un algorithme pour calculer un ensemble de polynômes générateurs de $(I:(r))$.

9. Supposons que I et J sont explicitement donnés par des polynômes générateurs $I = (f_1, \dots, f_s)$ et $J = (g_1, \dots, g_r)$. À partir des résultats précédents, proposer un algorithme pour calculer $(I : J)$.
10. Si $\mathbb{K}$ est algébriquement clos, et si V et W sont deux variétés algébriques de $\mathbb{A}^n_{\mathbb{K}}$, alors montrer que $(\mathbf{I}(V) : \mathbf{I}(W)) = \mathbf{I}(V \setminus W)$ (on rappelle que $\mathbf{I}(V)$ représente l'ensemble des polynômes de R qui s'annulent sur V).

EXERCICE 23. On s'intéresse à la somme $F_n := \sum_{k=0}^{n} k\binom{n}{k}$, pour laquelle on veut trouver une forme explicite. On pose $f_{n,k} := k\binom{n}{k}$.

1. Calculer $f_{n+1,k}$, $f_{n,k+1}$ et $f_{n+1,k+1}$.
2. Montrer que la combinaison linéaire
$$Af_{n,k} + Bf_{n+1,k} + Cf_{n,k+1} + Df_{n+1,k+1}$$
s'annule si et seulement si un certain polynôme s'annule.
3. Déterminer des constantes A, B, C et D qui annulent ce polynôme.
4. En déduire, par télescopage créatif qu'il conviendra de justifier, que la somme vérifie la récurrence
$$nF_{n+1} = 2(n+1)F_n.$$
5. En déduire une forme explicite pour F_n.
6. Proposer une méthode de sommation pour des suites hypergéométriques qui généralise cet exemple. On tâchera de fournir une formule explicite pour le télescopage créatif.

EXERCICE 24. On s'intéresse à la somme $F_n := \sum_{k=0}^{n} \binom{n}{2k}\binom{2k}{k}\frac{1}{4^k}$, pour laquelle on souhaite vérifier par l'algorithme de Zeilberger qu'elle vaut $\frac{1}{2^{n-1}}\binom{2n-1}{n-1}$. On pourra suivre les indications données.

1. Représenter $f_{n,k} := \binom{n}{2k}\binom{2k}{k}\frac{1}{4^k}$ à l'aide de factorielles et calculer $f_{n+1,k}$ et $f_{n,k+1}$.
2. Passer directement à l'étape de l'algorithme de Zeilberger qui recherche une récurrence en n d'ordre 1. Écrire explicitement la récurrence qui relie une combinaison linéaire des décalées en n de f et une différence finie en k du terme $R(n,k)f_{n,k}$. Quelles sont les inconnues de ce problème ? Dans quels ensembles cherche-t-on les solutions ?

Dans ce qui suit, on ne notera plus la dépendance de R en n et on écrira simplement $R(k)$, $R(k+1)$, etc.

3. Justifier que le seul dénominateur possible de R est $n - 2k + 1$. Écrire R sous la forme $S(k)/(n-2k+1)$ pour un polynôme S en k (à coefficient des fractions rationnelles en n) et remplacer dans l'équation.
4. On se restreint momentanément à l'équation homogène associée, c'est-à-dire au membre qui fait apparaître S. Remplacer S par un polynôme unitaire de degré α en k et développer juste suffisamment pour justifier que l'équation homogène ne peut avoir de solution polynomiale.
5. En déduire que le degré des solutions polynomiales de l'équation complète est 2.

6. Poser $S(k) = Ak^2 + Bk + C$. Relier d'abord η et A.
7. Spécialiser en $k = -1$ pour obtenir une expression très simple pour C.
8. Spécialiser en $k = 0$ pour relier A et B.
9. Spécialiser en $k = (n+1)/2$ pour obtenir une autre relation entre A et B.
10. Conclure sur les valeurs de A, B, C et η.
11. En déduire que la somme F_n vérifie $(n+1)F_{n+1} = (2n+1)F_n$.
12. Vérifier que la forme explicite annoncée est solution et prouver qu'elle est bien forme explicite de la somme des $f_{n,k}$.

Exercice 25 (Composition itérée de séries formelles). Soit F une série de la forme $F(X) = f_1X + f_2X^2 + \cdots$ à coefficients dans un corps $\mathbb{K}$. En notant $F^{[0]}(X) = X$, on définit la q-ième itérée de F par

$$F^{[q]}(X) = F^{[q-1]}(F(X)). \tag{E}$$

L'objectif de cet exercice est d'étudier la complexité $C_q(N)$ du calcul des N premiers termes de cette série pour N grand en nombre d'opérations dans $\mathbb{K}$. On note $\mathsf{M}(N)$ une borne sur le nombre d'opérations dans $\mathbb{K}$ nécessaires pour calculer le produit de deux polynômes de degré N dans $\mathbb{K}[X]$. On fait sur M les hypothèses habituelles de régularité. On note également $C(N)$ une borne sur le nombre d'opérations dans $\mathbb{K}$ nécessaires pour calculer la composition $f(g(X))$ de deux séries tronquées à l'ordre N telles que $g(0) = 0$. On rappelle qu'aucun algorithme quasi-optimal n'est connu pour la composition et que la meilleure complexité connue est $C(N) = O(\sqrt{N \log N}\mathsf{M}(N))$.

La notation $g(q, N) = O(f(q, N))$ pour des fonctions de q et N voudra dire dans cet exercice qu'il existe un entier N_0 tel que pour tout $N > N_0$,

$$|g(q, N)| \leq K|f(q, N)|,$$

la constante K ne dépendant ni de N, ni de q.

1. En n'utilisant que de l'algorithmique naïve, montrer que le calcul des N premiers termes de $F^{[q]}$ peut être effectué en $C_q(N) = O(qN^3)$ opérations dans $\mathbb{K}$.
2. En exploitant la composition rapide de séries, montrer que cette complexité peut être abaissée à $C_q(N) = O(q\sqrt{N \log N}\mathsf{M}(N))$.
3. Décrire une idée simple permettant de réduire la dépendance en q pour aboutir à $C_q(N) = O(\log q\sqrt{N \log N}\mathsf{M}(N))$.

Le reste de l'exercice étudie un algorithme permettant d'abaisser encore cette complexité pour finir à la même complexité que la composition avec F, indépendamment de q. L'idée de départ est que dans le cas de la puissance, le calcul de $F(X)^q$ peut être réalisé efficacement sous la forme $\exp(q \log F(X))$. L'objectif est d'adapter cette technique à la composition.

On suppose maintenant que $f_1 \neq 0$ et que f_1 n'est pas une racine de l'unité.

On définit la série de Schroeder $S(X) = s_1X + s_2X^2 + \cdots$ par

$$S(F(X)) = f_1S(X), \qquad s_1 = 1. \tag{S}$$

4. Montrer que cette équation définit bien une série $S(X)$, et que celle-ci est unique.
5. Montrer que pour tout entier q,
$$F^{[q]}(X) = S^{[-1]}(f_1^q S(X)),$$
où $S^{[-1]}$ est l'inverse de S pour la composition. En déduire une borne sur la complexité du calcul de $F^{[q]}$ une fois la série S connue à précision N.

6. Pour calculer S solution de (S), on va s'intéresser à la résolution d'une équation plus générale :

$$A(X)Y(F(X)) - B(X)Y(X) - C(X) = 0 \mod X^k, \tag{E}$$

où A, B, C, F sont des séries données et Y est l'inconnue. L'approche est une technique de « diviser pour régner ». En posant

$$Y(X) = U(X) + X^n V(X),$$

où U est un polynôme de degré inférieur à n, montrer que U et V sont solutions d'équations du même type que (E). Expliciter les coefficients de ces équations.

7. Décrire un algorithme de calcul des N premiers coefficients de Y en nombre d'opérations $O(\sqrt{N \log N}\mathsf{M}(N))$. En notant a_0, b_0, c_0 les termes constants des séries A, B, C, on supposera que $a_0 f_1^m \neq b_0$, pour $m = 1, 2, 3, \ldots$, et que si $a_0 = b_0$, alors $c_0 = 0$.
8. Décrire enfin l'algorithme de calcul de $F^{[q]}$ dans la complexité annoncée.
9. À l'inverse, décrire un algorithme calculant une série G telle que $F = G^{[q]}$, dans la même complexité.

Exercice 26 (Calcul du radical). On rappelle que le radical d'un idéal I, noté $\sqrt{I}$, est défini par $\{f \mid \exists r \in \mathbb{N}, f^r \in I\}$. On rappelle aussi qu'un polynôme est dit sans carré si les multiplicités de ses facteurs irréductibles sont égales à un. Dans cet exercice $\mathbb{K}$ représente un corps de caractéristique zéro, et R est l'anneau des polynômes $\mathbb{K}[X_1, \ldots, X_n]$.

1. Si $n = 1$, et si I est un idéal de R engendré par $f_1, \ldots, f_s$, décrire un algorithme pour calculer un ensemble de générateurs du radical de I.
2. Dans cette question nous supposons que, pour chaque $i \in \{1, \ldots, n\}$, il existe $g_i \in I \cap \mathbb{K}[X_i]$ sans carré, et souhaitons montrer que I est radical, en suivant les étapes suivantes :
   (a) Si $n = 1$ montrer que I est radical.
   (b) Si $h_1, \ldots, h_m$ sont les facteurs irréductibles de g_1 alors montrer l'égalité $I = \cap_{j=1}^m (I + (h_j))$, en utilisant le fait qu'il existe des polynômes $v_1, \ldots, v_m$ de $\mathbb{K}[X_1]$ tels que $v_1 \frac{g_1}{h_1} + \cdots + v_m \frac{g_1}{h_m} = 1$.
   (c) Montrer que si $I + (h_j)$ est radical pour tout $j \in \{1, \ldots, m\}$, alors I est radical.
   (d) Soit $\mathbb{L}$ l'extension du corps $\mathbb{K}$ définie par $\mathbb{K}[X_1]/(h_1(X_1))$ et soit ϕ la projection $\mathbb{K}[X_1, \ldots, X_n] \to \mathbb{L}[X_2, \ldots, X_n]$. Montrer que si $J := \phi(I + (h_1))$ est radical alors $I + (h_1)$ est radical.
   (e) Finalement, en utilisant les questions précédentes, montrer par récurrence sur n que I est radical.
3. On rappelle que I est dit de dimension affine zéro si la dimension de la $\mathbb{K}$-algèbre R/I est finie. Nous supposons dans cette question que I est un idéal de dimension affine zéro engendré par les polynômes $f_1, \ldots, f_s$, et nous nous intéressons au calcul d'un ensemble de générateurs de son radical.
   (a) Montrer que pour tout $i \in \{1, \ldots, n\}$, il existe un polynôme sans carré g_i dans $\sqrt{I} \cap \mathbb{K}[X_i]$ et décrire un algorithme pour calculer de tels polynômes g_i.
   (b) Montrer que $\sqrt{I}$ est engendré par $f_1, \ldots, f_s, g_1, \ldots, g_n$.

EXERCICE 27. Cet exercice montre la terminaison de l'algorithme de Zeilberger sur la classe des *termes hypergéométriques propres*, qui sont des termes hypergéométriques de la forme

$$h_{n,k} = P(n,k)\,\zeta^n \xi^k \prod_{\ell=1}^{L} \Gamma(a_\ell n + b_\ell k + c_\ell)^{\epsilon_\ell}, \tag{4}$$

où, pour un corps $\mathbb{K}$ inclus dans $\mathbb{C}$, les a_ℓ et les b_ℓ sont dans $\mathbb{Z}$, les ϵ_ℓ valent ± 1, les c_ℓ, ζ et ξ sont dans $\mathbb{K}$, et P est un polynôme dans $\mathbb{K}[n,k]$. On rappelle que la fonction gamma vérifie pour $s \in \mathbb{C} \setminus \mathbb{Z}^-$, $\Gamma(s+1) = s\Gamma(s)$, et pour tout $n \in \mathbb{N}$, $\Gamma(n+1) = n!$.

1. Rappeler comment l'existence d'une relation

$$\sum_{i=0}^{r} \sum_{j=0}^{s} f_{i,j}(n)\, h_{n+i,k+j} = 0 \tag{5}$$

pour $r, s \geq 0$ et des fractions rationnelles $f_{i,j} \in \mathbb{K}(n)$ non toutes nulles entraîne l'existence de fractions rationnelles $\theta_0, \ldots, \theta_r \in \mathbb{K}(n)$ et $q \in \mathbb{K}(n,k)$, non toutes nulles, vérifiant la relation

$$\theta_r(n)\, h_{n+r,k} + \cdots + \theta_0(n)\, h_{n,k} = q(n,k+1)\, h_{n,k+1} - q(n,k)\, h_{n,k}. \tag{6}$$

2. Donner un exemple de terme hypergéométrique qui ne peut pas s'exprimer comme terme hypergéométrique propre.
3. Pour chaque ℓ, borner en fonction de r et s la valeur absolue de l'entier u intervenant dans les termes $\Gamma(a_\ell n + b_\ell k + c_\ell + u)$ apparaissant dans l'égalité (5) après prise en compte de la définition (4) et des décalages sur n et k.
4. Montrer que, pour les entiers u définis comme à la question précédente, les deux quotients

$$\frac{\Gamma(a_\ell n + b_\ell k + c_\ell + u)}{\Gamma(a_\ell n + b_\ell k + c_\ell - \sigma_\ell)} \quad \text{et} \quad \frac{\Gamma(a_\ell n + b_\ell k + c_\ell + \sigma_\ell)}{\Gamma(a_\ell n + b_\ell k + c_\ell + u)}$$

sont des polynômes dont on majorera le degré en fonction de la borne σ_ℓ obtenue à la question précédente.
5. Construire un terme $H_{n,k}$ tel que chaque produit $H_{n,k} h_{n+i,k+j}$ pour $0 \leq i \leq r$ et $0 \leq j \leq s$ soit un polynôme de degré en k au plus linéaire en r et s.
6. En déduire que pour i et j comme à la question précédente

$$\deg_k\big(H_{n,k} h_{n+i,k+j}\big) \leq \deg_k(P) + Ar + Bs$$

pour des entiers A et B que l'on précisera.
7. En déduire une contrainte polynomiale sur r et s pour que l'équation (5) ait une solution non triviale $\{f_{i,j}\}_{0\leq i\leq r,\ 0\leq j\leq s}$ pour des $f_{i,j}$ dans $\mathbb{K}(n)$.
8. En déduire que le choix

$$r = B \quad \text{et} \quad s = (A-1)\,B + \deg_k(P) + 1$$

fournit toujours une solution à l'équation (5).
9. Décrire à partir de l'analyse précédente un algorithme prenant en entrée un terme hypergéométrique propre de la forme (4) et renvoyant une relation (6).
10. Expliciter (4) sur l'exemple $h_{n,k} = \binom{n+k}{k}^2 \binom{n}{k}^2$, puis calculer les grandeurs principales intervenant dans l'algorithme : A, B, r, s, $H_{n,k}$. Décrire le système linéaire à résoudre : taille, degré des coefficients.

11. De nouveau sur le cas général, par quel algorithme spécialisé du cours suggérez-vous d'effectuer l'algèbre linéaire ? En admettant que cette étape est l'étape dominante du calcul, estimer la complexité de l'algorithme, en fonction de A et B.

12. Ce qui précède ne justifie pas totalement la terminaison de l'algorithme de Zeilberger, à cause de la possible annulation du membre gauche de (6). Pour achever la justification, on introduit l'algèbre de Ore $S = \mathbb{K}(n,k)\langle \partial_n, \partial_k; S_n, S_k, 0, 0\rangle$, et on remplace (5) par le polynôme tordu
$$Z = \sum_{i=0}^{r} \sum_{j=0}^{s} f_{i,j}(n)\, \partial_n^i \partial_k^j.$$
Montrer qu'il existe un entier v, un polynôme tordu P non nul indépendant de k et ∂_k, et un polynôme tordu général Q tels que
$$Z = (\partial_k - 1)^v \big(P(n, \partial_n) - (\partial_k - 1)Q(n, \partial_n, \partial_k)\big).$$

13. On introduit la factorielle descendante $k^{\underline{s}} = k(k-1)\dots(k-s+1)$, où $s \in \mathbb{N}$. Montrer la commutation, pour $a \in \mathbb{K}$,
$$(k-a)^{\underline{s}}\,(\partial_k - 1) = (\partial_k - 1)\,(k-a-1)^{\underline{s}} - s\,(k-s-1)^{\underline{s-1}}.$$

14. Calculer $k^{\underline{s}}\,(\partial_k - 1)^s$ modulo $\partial_k - 1$ à gauche.

15. En déduire que l'existence d'une relation (6) non triviale implique l'existence d'un $P = \eta_r(n)\,\partial_n^r + \cdots + \eta_0(n)$ non nul de S et d'une fraction rationnelle $Q \in \mathbb{K}(n,k)$ vérifiant
$$\big(P - (\partial_k - 1)Q\big) \cdot h = 0.$$

16. Conclure sur l'algorithme de Zeilberger : donner une classe de termes hypergéométriques sur laquelle il termine et indiquer des bornes sur l'ordre et le degré du télescopeur obtenu.

17. L'exemple $\binom{n+k}{k}^2 \binom{n}{k}^2$ a suggéré que l'algorithme sous-jacent à ce problème n'est (tel quel) pas efficace en pratique. Suggérez ce qui permet à l'algorithme de Zeilberger d'être plus efficace.

Notes

L'exercice 9 vient de [10]. Il montre que le produit d'opérateurs différentiels de bidegrés (n,n) en (X,θ) se ramène à un nombre constant de multiplications de matrices carées de taille n. L'article plus récent [5] montre que ces deux problèmes sont équivalents du point de vue de la complexité. L'exercice 1 est inspiré par l'article [2], qui montre que la prise de racine carrée $\mathsf{R}(n)$ d'une série tronquée à précision n vérifie aussi $O(\mathsf{R}) = O(\mathsf{M})$. L'exercice 10 est tiré de [3]. L'exercice 4 provient de [12]. L'exercice 25 est une adaptation de l'article [9]. L'exercice 14 est extrait de [4], l'exercice 7 de [6]. La première question de l'exercice 2 est tirée de [1], la seconde est implicite dans [8]. L'exerice 11 est inspiré par [11], l'exercice 3 est tiré de [7].

Bibliographie

[1] Aho, A. V., K. Steiglitz et J. D. Ullman (1975). « Evaluating polynomials at fixed sets of points ». In : *SIAM Journal on Computing*, vol. 4, n°4, p. 533–539.

[2] Alt, H. (1978/79). « Square rooting is as difficult as multiplication ». In : *Computing*, vol. 21, n°3, p. 221–232.

[3] Borwein, Jonathan M. et Bruno Salvy (2008). « A proof of a recurrence for Bessel moments ». In : *Experimental Mathematics*, vol. 17, n°2, p. 223–230.

[4] Bostan, A., F. Chyzak, Z. Li et B. Salvy. « Common multiples of linear ordinary differential and difference operators ». In preparation.

[5] Bostan, Alin, Frédéric Chyzak et Nicolas Le Roux (2008). « Products of ordinary differential operators by evaluation and interpolation ». In : *ISSAC'08 : International Symposium on Symbolic and Algebraic Computation*. Éd. par David J. Jeffrey. ACM Press, p. 23–30.

[6] Bostan, Alin, Philippe Flajolet, Bruno Salvy et Éric Schost (2006). « Fast Computation of Special Resultants ». In : *Journal of Symbolic Computation*, vol. 41, n°1, p. 1–29. doi : 10.1016/j.jsc.2005.07.001.

[7] Bostan, Alin, Pierrick Gaudry et Éric Schost (2007). « Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator ». In : *SIAM Journal on Computing*, vol. 36, n°6, p. 1777–1806.

[8] Bostan, Alin et Éric Schost (2005). « Polynomial evaluation and interpolation on special sets of points ». In : *Journal of Complexity*, vol. 21, n°4, p. 420–446.

[9] Brent, R. P. et J. F. Traub (1980). « On the complexity of composition and generalized composition of power series ». In : *SIAM Journal on Computing*, vol. 9, n°1, p. 54–66.

[10] Hoeven, Joris van der (2002). « FFT-like multiplication of linear differential operators ». In : *Journal of Symbolic Computation*, vol. 33, n°1, p. 123–127.

[11] Keller-Gehrig, Walter (1985). « Fast algorithms for the characteristic polynomial ». In : *Theoretical Computer Science*, vol. 36, n°2-3, p. 309–317.

[12] Kung, H. T. et D. M. Tong (1977). « Fast algorithms for partial fraction decomposition ». In : *SIAM Journal on Computing*, vol. 6, n°3, p. 582–593.

Bibliographie générale

Abdeljaoued, J. et H. Lombardi (2004). *Méthodes matricielles : introduction à la complexité algébrique*. Vol. 42. Mathématiques & Applications. Springer-Verlag, xvi+376 pages.

Abel, Niels Henrik (1992). *Œuvres complètes. Tome II*. Réimpression de la seconde édition (1881). Éditions Jacques Gabay, vi+716 pages. URL : http://www.gallica.fr.

Abramov, S. A. (1989). « Rational solutions of linear differential and difference equations with polynomial coefficients ». In : *USSR Computational Mathematics and Mathematical Physics*, vol. 29, n°11, p. 1611–1620.

— (1995). « Rational solutions of linear difference and q-difference equations with polynomial coefficients ». In : *ISSAC'95 : International Symposium on Symbolic and Algebraic Computation*. Éd. par A. H. M. Levelt. New York : ACM Press, p. 285–289.

— (2003). « When does Zeilberger's algorithm succeed ? » In : *Advances in Applied Mathematics*, vol. 30, n°3, p. 424–441.

Abramov, Sergei A., Manuel Bronstein et Marko Petkovšek (1995). « On Polynomial Solutions of Linear Operator Equations ». In : *ISSAC'95 : International Symposium on Symbolic and Algebraic Computation*. Éd. par A. H. M. Levelt. New York : ACM Press, p. 290–296.

Abramov, Sergei A. et Marko Petkovšek (1994). « D'Alembertian solutions of linear differential and difference equations ». In : *ISSAC'94 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 169–174.

Agrawal, Manindra, Neeraj Kayal et Nitin Saxena (2004). « PRIMES is in P ». In : *Annals of Mathematics. Second Series*, vol. 160, n°2, p. 781–793.

Aho, A. V., K. Steiglitz et J. D. Ullman (1975). « Evaluating polynomials at fixed sets of points ». In : *SIAM Journal on Computing*, vol. 4, n°4, p. 533–539.

Aho, Alfred V., John E. Hopcroft et Jeffrey D. Ullman (1974). *The design and analysis of computer algorithms*. Addison-Wesley Publishing Co., x+470 pages.

Alt, H. (1978/79). « Square rooting is as difficult as multiplication ». In : *Computing*, vol. 21, n°3, p. 221–232.

Antoniou, A. (1979). *Digital Filters : Analysis and Design*. McGraw-Hill Book Co.

Arrondo, Enrique (2006). « Another elementary proof of the Nullstellensatz ». In : *The American Mathematical Monthly*, vol. 113, n°2, p. 169–171.

Baker Jr., George A. et Peter Graves-Morris (1996). *Padé approximants*. 2e éd. Vol. 59. Encyclopedia of Mathematics and its Applications. Cambridge University Press, xiv+746 pages.

Baur, W. et V. Strassen (1983). « The complexity of partial derivatives ». In : *Theoretical Computer Science*, vol. 22, p. 317–330.

Beck, Matthias, Bruce C. Berndt, O-Yeat Chan et Alexandru Zaharescu (2005). « Determinations of Analogues of Gauss Sums and Other Trigonometric Sums ». In : *International Journal of Number Theory*, vol. 1, n°3, p. 333–356.

Beckermann, B. et G. Labahn (1994). « A uniform approach for the fast computation of matrix-type Padé approximants ». In : *SIAM Journal on Matrix Analysis and Applications*, vol. 15, n°3, p. 804–823.

Beckermann, Bernhard (1992). « A reliable method for computing M-Padé approximants on arbitrary staircases ». In : *Journal of Computational and Applied Mathematics*, vol. 40, n°1, p. 19–42.

Beckermann, Bernhard et George Labahn (2000). « Fraction-free computation of matrix rational interpolants and matrix GCDs ». In : *SIAM Journal on Matrix Analysis and Applications*, vol. 22, n°1, p. 114–144.

Beeler, Michael, R. William Gosper et Richard Schroeppel (1972). *HAKMEM*. Artificial Intelligence Memo No. 239. Massachusetts Institute of Technology. URL : http://www.inwap.com/pdp10/hbaker/hakmem/hakmem.html.

Ben-Or, M. et P. Tiwari (1988). « A deterministic algorithm for sparse multivariate polynomial interpolation ». In : *STOC'88 : ACM Symposium on Theory of Computing*. ACM Press, p. 301–309.

Bentley, Jon Louis, Dorothea Haken et James B. Saxe (1980). « A general method for solving divide-and-conquer recurrences ». In : *SIGACT News*, vol. 12, n°3, p. 36–44. DOI : 10.1145/1008861.1008865.

Berkovich, L. M. et V. G. Tsirulik (1986). « Differential resultants and some of their applications ». In : *Differentsial'nye Uravneniya*, vol. 22, n°5. English translation in : Differential Equations, Plenum Publ. Corp., Vol. 22, no. 5, pp. 530–536. NY Plenum 1986, p. 750–757, 914.

Berkowitz, Stuart J. (1984). « On Computing the determinant in small parallel time using a small number of processors ». In : *Information Processing Letters*, vol. 18, p. 147–150.

Berlekamp, E. R. (1967). « Factoring polynomials over finite fields ». In : *Bell System Technical Journal*, vol. 46, p. 1853–1859.

— (1970). « Factoring polynomials over large finite fields ». In : *Math. Comp.* vol. 24, p. 713–735.

Bernstein, Daniel J. (2001a). *Multidigit multiplication for mathematicians*. URL : http://cr.yp.to/papers.html (visible en 2001).

— (2001b). *Removing redundancy in high-precision Newton iteration*. URL : http://cr.yp.to/fastnewton.html (visible en 2001).

— *The transposition principle*. Preprint. URL : http : / / cr . yp . to / transposition.html.

— (1998). « Composing power series over a finite ring in essentially linear time ». In : *Journal of Symbolic Computation*, vol. 26, n°3, p. 339–341.

— (2008). « Fast multiplication and its applications ». In : *Algorithmic number theory : lattices, number fields, curves and cryptography*. Vol. 44. Publications of the Research Institute for Mathematical Sciences. Cambridge University Press, p. 325–384.

Bézout, É. (1764). « Recherches sur le degré des équations résultantes de l'évanouissement des inconnues ». In : *Histoire de l'académie royale des sciences*, p. 288–338.

Bini, D. (1980). « Relations between exact and approximate bilinear algorithms. Applications ». In : *Calcolo*, vol. 17, n°1, p. 87–97. DOI : 10.1007/BF02575865.

Bini, D., M. Capovani, F. Romani et G. Lotti (1979). « $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication ». In : *Information Processing Letters*, vol. 8, n°5, p. 234–235.

Bitmead, Robert R. et Brian D. O. Anderson (1980). « Asymptotically fast solution of Toeplitz and related systems of linear equations ». In : *Linear Algebra and its Applications*, vol. 34, p. 103–116.

Bläser, Markus (2001). « A $\frac{5}{2}n^2$-lower bound for the multiplicative complexity of $n \times n$-matrix multiplication ». In : *STACS 2001 (Dresden)*. Vol. 2010. Lecture Notes in Computer Science. Springer-Verlag, p. 99–109. DOI : 10.1007/3-540-44693-1_9.

— (2003). « On the complexity of the multiplication of matrices of small formats ». In : *Journal of Complexity*, vol. 19, n°1, p. 43–60.

Blondel, Vincent D. et Natacha Portier (2002). « The presence of a zero in an integer linear recurrent sequence is NP-hard to decide ». In : *Linear Algebra and its Applications*, vol. 351/352. Fourth special issue on linear systems and control, p. 91–98.

Bluestein, L. I. (1970). « A linear filtering approach to the computation of the discrete Fourier transform ». In : *IEEE Trans. Electroacoustics*, vol. AU-18, p. 451–455.

Bluestein, Leo I. (1968). « A linear filtering approach to the computation of the discrete Fourier transform ». In : *IEEE Northeast Electronics Research and Engineering Meeting*, vol. 10, p. 218–219.

Bodrato, Marco et Alberto Zanoni (2007). « Integer and polynomial multiplication : towards optimal Toom-Cook matrices ». In : *ISSAC'07 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Dongming Wang. ACM Press, p. 17–24. DOI : 10.1145/1277548.1277552.

Bordewijk, J. L. (1956). « Inter-reciprocity applied to electrical networks ». In : *Appl. Sci. Res. B.* vol. 6, p. 1–74.

Borodin, A. et R. T. Moenck (1974). « Fast modular transforms ». In : *Comput. Sys. Sci.* vol. 8, n°3, p. 366–386.

Borodin, A. et I. Munro (1971). « Evaluating polynomials at many points ». In : *Information Processing Letters*, vol. 1, n°2, p. 66–68. DOI : 10.1016/0020-0190(71)90009-3.

Borwein, Jonathan M. et Bruno Salvy (2008). « A proof of a recurrence for Bessel moments ». In : *Experimental Mathematics*, vol. 17, n°2, p. 223–230.

Borwein, Peter B. (1985). « On the complexity of calculating factorials ». In : *Journal of Algorithms*, vol. 6, n°3, p. 376–380.

Bostan, A., F. Chyzak, T. Cluzeau et B. Salvy (2006). « Low complexity algorithms for linear recurrences ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 31–38.

Bostan, A., F. Chyzak, Z. Li et B. Salvy. « Common multiples of linear ordinary differential and difference operators ». In preparation.

Bostan, A., F. Chyzak, F. Ollivier, B. Salvy, É. Schost et A. Sedoglavic (2007). « Fast computation of power series solutions of systems of differential equations ». In : *SODA'07 : ACM-SIAM Symposium on Discrete Algorithms*. SIAM, p. 1012–1021.

Bostan, A., G. Lecerf, B. Salvy, É. Schost et B. Wiebelt (2004c). « Complexity issues in bivariate polynomial factorization ». In : *ISSAC'04 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 42–49.

Bostan, A. et É. Schost (2004). « On the complexities of multipoint evaluation and interpolation ». In : *Theoretical Computer Science*, vol. 329, n°1–3, p. 223–235.

Bostan, Alin (2003). « Algorithmique efficace pour des opérations de base en Calcul formel ». Thèse de Doctorat. École polytechnique.

Bostan, Alin, Frédéric Chyzak et Nicolas Le Roux (2008). « Products of ordinary differential operators by evaluation and interpolation ». In : *ISSAC'08 :*

International Symposium on Symbolic and Algebraic Computation. Éd. par David J. Jeffrey. ACM Press, p. 23–30.

Bostan, Alin, Frédéric Chyzak, Grégoire Lecerf, Bruno Salvy et Éric Schost (2007). « Differential equations for algebraic functions ». In : *ISSAC'07 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Dongming Wang. ACM Press, p. 25–32. doi : 10.1145/1277548.1277553.

Bostan, Alin, Thomas Cluzeau et Bruno Salvy (2005). « Fast Algorithms for Polynomial Solutions of Linear Differential Equations ». In : *ISSAC'05 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 45–52. doi : 10.1145/1073884.1073893.

Bostan, Alin, Philippe Flajolet, Bruno Salvy et Éric Schost (2006). « Fast Computation of Special Resultants ». In : *Journal of Symbolic Computation*, vol. 41, n°1, p. 1–29. doi : 10.1016/j.jsc.2005.07.001.

Bostan, Alin, Pierrick Gaudry et Éric Schost (2007). « Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator ». In : *SIAM Journal on Computing*, vol. 36, n°6, p. 1777–1806.

Bostan, Alin, Claude-Pierre Jeannerod et Éric Schost (2008). « Solving structured linear systems with large displacement rank ». In : *Theoretical Computer Science*, vol. 407, n°1-3, p. 155–181.

Bostan, Alin, Grégoire Lecerf et Éric Schost (2003). « Tellegen's principle into practice ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par J. R. Sendra. ACM Press, p. 37–44.

Bostan, Alin, Bruno Salvy et Éric Schost (2008). « Power Series Composition and Change of Basis ». In : *ISSAC'08 : International Symposium on Symbolic and Algebraic Computation*. Éd. par David J. Jeffrey. ACM Press, p. 269–276. doi : 10.1145/1390768.1390806.

Bostan, Alin et Éric Schost (2005). « Polynomial evaluation and interpolation on special sets of points ». In : *Journal of Complexity*, vol. 21, n°4, p. 420–446.

Brent, R. P. et H. T. Kung (1978). « Fast algorithms for manipulating formal power series ». In : *Journal of the ACM*, vol. 25, n°4, p. 581–595.

Brent, R. P. et J. F. Traub (1980). « On the complexity of composition and generalized composition of power series ». In : *SIAM Journal on Computing*, vol. 9, n°1, p. 54–66.

Brent, Richard P. (1976). « Multiple-precision zero-finding methods and the complexity of elementary function evaluation ». In : *Analytic computational complexity*. Proceedings of a Symposium held at Carnegie-Mellon University, Pittsburgh, PA, 1975. Academic Press, p. 151–176.

Brent, Richard P., Fred G. Gustavson et David Y. Y. Yun (1980). « Fast solution of Toeplitz systems of equations and computation of Padé approximants ». In : *Journal of Algorithms*, vol. 1, n°3, p. 259–295.

Brigham, E. Oran (1974). *The fast Fourier transform*. Prentice-Hall, xiii+252 pages.

Bronstein, M. et M. Petkovšek (1994). « On Ore rings, linear operators and factorisation ». In : *Programmirovanie*, vol. 1. Also available as Research Report 200, Informatik, ETH Zürich, p. 27–44.

Bronstein, Manuel (1992). « Linear Ordinary Differential Equations : breaking through the order 2 barrier ». In : *ISSAC'92 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 42–48.

Bronstein, Manuel et Marko Petkovšek (1996). « An introduction to pseudo-linear algebra ». In : *Theoretical Computer Science*, vol. 157, p. 3–33.

Brown, W. S. (1971). « On Euclid's algorithm and the computation of polynomial greatest common divisors ». In : *SYMSAC'71 : ACM Symposium on Symbolic and Algebraic Manipulation*. Los Angeles, California, United States : ACM, p. 195–211. DOI : 10.1145/800204.806288.

Brown, W. S. et J. F. Traub (1971). « On Euclid's algorithm and the theory of subresultants ». In : *Journal of the Association for Computing Machinery*, vol. 18, p. 505–514.

Buchberger, B. (1965). « Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal) ». Thèse de doct. Mathematical Institute, University of Innsbruck, Austria.

Bunch, James R. et John E. Hopcroft (1974). « Triangular factorization and inversion by fast matrix multiplication ». In : *Mathematics of Computation*, vol. 28, p. 231–236.

Bürgisser, P., M. Karpinski et T. Lickteig (1991). « Some computational problems in linear algebra as hard as matrix multiplication ». In : *Computational Complexity*, vol. 1, n°2, p. 131–155. DOI : 10.1007/BF01272518.

Bürgisser, Peter, Michael Clausen et M. Amin Shokrollahi (1997c). *Algebraic complexity theory*. Vol. 315. Grundlehren der Mathematischen Wissenschaften. Springer-Verlag, xxiv+618 pages.

Bürgisser, Peter et Martin Lotz (2004). « Lower bounds on the bounded coefficient complexity of bilinear maps ». In : *Journal of the Association for Computing Machinery*, vol. 51, n°3, p. 464–482. DOI : 10.1145/990308.990311.

Cabay, S. et G. Labahn (1989). « A fast, reliable algorithm for calculating Padé-Hermite forms ». In : *ISSAC'89 : International Symposium on Symbolic and Algebraic Computation*. Portland, Oregon, United States : ACM Press, p. 95–100. DOI : 10.1145/74540.74553.

Cabay, S., G. Labahn et B. Beckermann (1992). « On the theory and computation of nonperfect Padé-Hermite approximants ». In : *Journal of Computational and Applied Mathematics*, vol. 39, n°3, p. 295–313.

Cabay, Stanley et Dong Koo Choi (1986). « Algebraic computations of scaled Padé fractions ». In : *SIAM Journal on Computing*, vol. 15, n°1, p. 243–270.

Camion, P. (1983). « A deterministic algorithm for factorizing polynomials of $\mathbf{F}_q[X]$ ». In : *Combinatorial mathematics (Marseille-Luminy, 1981)*. Vol. 75. North-Holland Math. Stud. North-Holland, Amsterdam, p. 149–157.

Canny, J., E. Kaltofen et L. Yagati (1989). « Solving systems of non-linear polynomial equations faster ». In : *ISSAC'89 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 121–128.

Cantor, D. G. et E. Kaltofen (1991a). « On Fast Multiplication of Polynomials over Arbitrary Algebras ». In : *Acta Informatica*, vol. 28, n°7, p. 693–701.

Cantor, D. G. et H. Zassenhaus (1981). « A new algorithm for factoring polynomials over finite fields ». In : *Mathematics of Computation*, vol. 36, n°154, p. 587–592.

Cardinal, Jean-Paul (1999). « On a property of Cauchy-like matrices ». In : *Comptes Rendus de l'Académie des Sciences. Série I. Mathématique*, vol. 328, n°11, p. 1089–1093.

Cartier, Pierre (1992). « Démonstration "automatique" d'identités et fonctions hypergéométriques (d'après D. Zeilberger) ». In : *Astérisque*, vol. 1991/92, n°206. Séminaire Bourbaki, Exp. No. 746, 3, 41–91.

Cenk, M., C. K. Koc et F. Ozbudak (2009). « Polynomial Multiplication over Finite Fields Using Field Extensions and Interpolation ». In : *ARITH'09 :*

IEEE Symposium on Computer Arithmetic. IEEE Computer Society, p. 84–91. DOI : 10.1109/ARITH.2009.11.

Cerlienco, L., M. Mignotte et F. Piras (1987). « Suites récurrentes linéaires. Propriétés algébriques et arithmétiques ». In : *L'Enseignement Mathématique*. II, vol. 33, p. 67–108.

Chardin, Marc (1991). « Differential resultants and subresultants ». In : *Fundamentals of computation theory*. Vol. 529. Lecture Notes in Computer Science. Gosen : Springer-Verlag, p. 180–189.

Chen, William Y.C., Peter Paule et Husam L. Saad (2008). « Converging to Gosper's algorithm ». In : *Advances in Applied Mathematics*, vol. 41, n°3, p. 351–364. DOI : 10.1016/j.aam.2007.11.004.

Cheng, Unjeng (1984). « On the continued fraction and Berlekamp's algorithm ». In : *IEEE Transactions on Information Theory*, vol. 30, n°3, p. 541–544.

Chu, Eleanor et Alan George (2000). *Inside the FFT black box*. Serial and parallel fast Fourier transform algorithms. CRC Press, xxii+312 pages.

Chudnovsky, D. V. et G. V. Chudnovsky (1988). « Approximations and complex multiplication according to Ramanujan ». In : *Ramanujan revisited*. MA : Academic Press, p. 375–472.

Chung, Jaewook et M. Anwar Hasan (2007). « Asymmetric Squaring Formulae ». In : *ARITH'07 : IEEE Symposium on Computer Arithmetic*. IEEE, p. 113–122. DOI : 10.1109/ARITH.2007.11.

Chyzak, Frédéric et Bruno Salvy (1998a). « Non-commutative Elimination in Ore Algebras Proves Multivariate Holonomic Identities ». In : *Journal of Symbolic Computation*, vol. 26, n°2, p. 187–227. DOI : 10.1006/jsco.1998.0207.

Cohn, H., R. Kleinberg, B. Szegedy et C. Umans (2005). « Group-theoretic Algorithms for Matrix Multiplication ». In : *FOCS'05 : IEEE Conference on Foundations of Computer Science*. IEEE Computer Society, p. 379–388. DOI : 10.1109/SFCS.2005.39.

Cohn, Henry et Christopher Umans (2003). « A Group-Theoretic Approach to Fast Matrix Multiplication ». In : *FOCS'03 : IEEE Conference on Foundations of Computer Science*. Washington, DC, USA : IEEE Computer Society, p. 438.

Collins, G. E. (1966). « Polynomial remainder sequences and determinants ». In : *The American Mathematical Monthly*, vol. 73, p. 708–712.

Collins, George E. (1967). « Subresultants and reduced polynomial remainder sequences ». In : *Journal of the Association for Computing Machinery*, vol. 14, p. 128–142.

— (1971). « The calculation of multivariate polynomial resultants ». In : *Journal of the Association for Computing Machinery*, vol. 18, p. 515–532.

Comtet, L. (1970). *Analyse Combinatoire*. 2 volumes. PUF.

Cook, S. (1966). « On the minimum computation time of functions ». Thèse de doct. Harvard University.

Cook, S. A. et S. O. Aanderaa (1969). « On the minimum computation time of functions ». In : *Transactions of the American Mathematical Society*, vol. 142, p. 291–314.

Cooley, James W. (1990). « How the FFT gained acceptance ». In : *A history of scientific computing*. ACM Press Hist. Ser. P : ACM, p. 133–140.

Cooley, James W. et John W. Tukey (1965). « An algorithm for the machine calculation of complex Fourier series ». In : *Mathematics of Computation*, vol. 19, p. 297–301.

— (1993). « On the Origin and Publication of the FFT Paper ». In : *Current Contents*, vol. 33, n°51–52, p. 8–9. URL : www.garfield.library.upenn.edu/classics1993/%20A1993MJ84500001.pdf.

COPPERSMITH, Don (1994). « Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm ». In : *Mathematics of Computation*, vol. 62, n°205, p. 333–350.

COPPERSMITH, Don et Shmuel WINOGRAD (1990). « Matrix multiplication via arithmetic progressions ». In : *Journal of Symbolic Computation*, vol. 9, n°3, p. 251–280.

COX, David, John LITTLE et Donal O'SHEA (1996). *Ideals, varieties, and algorithms*. 2e éd. Springer-Verlag, xiv+536 pages.

DANILEVSKII, A. M (1937). « The numerical solution of the secular equation ». In : *Matematicheskiĭ Sbornik*, vol. 44, n°2. (in Russian), p. 169–171.

DAVIS, Martin, Hilary PUTNAM et Julia ROBINSON (1961). « The decision problem for exponential diophantine equations ». In : *Annals of Mathematics. Second Series*, vol. 74, p. 425–436.

DE, A., P. P. KURUR, C. SAHA et R. SAPTHARISHI (2008). « Fast integer multiplication using modular arithmetic ». In : *STOC'08 : ACM Symposium on Theory of Computing*. Victoria, British Columbia, Canada : ACM, p. 499–506. DOI : 10.1145/1374376.1374447.

DECKER, W., G.-M. GREUEL, G. PFISTER et H. SCHÖNEMANN (2012). SINGULAR *3-1-6 — A computer algebra system for polynomial computations*. http://www.singular.uni-kl.de.

DELLA DORA, J. et C. DICRESCENZO (1984a). « Approximants de Padé-Hermite. I. Théorie ». In : *Numerische Mathematik*, vol. 43, n°1, p. 23–39.

— (1984b). « Approximants de Padé-Hermite. II. Programmation ». In : *Numerische Mathematik*, vol. 43, n°1, p. 41–57.

DEMAZURE, M. (1987). « Le théorème de complexité de Mayr et Meyer ». In : *Géométrie algébrique et applications, I (La Rábida, 1984)*. Vol. 22. Travaux en Cours. Paris : Hermann, p. 35–58.

DEMAZURE, Michel (1997). *Cours d'algèbre*. Nouvelle Bibliothèque Mathématique n°1. Cassini, Paris, xviii+302 pages.

DEMILLO, Richard A. et Richard J. LIPTON (1978). « A probabilistic remark on algebraic program testing ». In : *Information Processing Letters*, vol. 7, n°4, p. 193–195.

DERKSEN, Harm (1994). *An algorithm to compute generalized Padé-Hermite forms*. Report n°9403. Dept. of Math., Catholic University Nijmegen.

DIXON, John D. (1982). « Exact solution of linear equations using p-adic expansions ». In : *Numerische Mathematik*, vol. 40, n°1, p. 137–141.

DONGARRA, J. et F. SULLIVAN (2000). « Top ten algorithms ». In : *Computing in Science & Engineering*, vol. 2, n°1, p. 22–23.

DORNSTETTER, Jean-Louis (1987). « On the equivalence between Berlekamp's and Euclid's algorithms ». In : *IEEE Transactions on Information Theory*, vol. 33, n°3, p. 428–431.

DRMOTA, Michael et Wojciech SZPANKOWSKI (2011). « A master theorem for discrete divide and conquer recurrences ». In : *SODA'11 : ACM-SIAM Symposium on Discrete Algorithms*. San Francisco, CA : SIAM, p. 342–361.

DUHAMEL, P. et M. VETTERLI (1990). « Fast Fourier transforms : a tutorial review and a state of the art ». In : *Signal Processing. An Interdisciplinary Journal*, vol. 19, n°4, p. 259–299.

DURVYE, Clémence (2008). « Algorithmes pour la décomposition primaire des idéaux polynomiaux de dimension nulle donnés en évaluation ». Thèse de doct. Université de Versailles Saint-Quentin-en-Yvelines.

Durvye, Clémence et Grégoire Lecerf (2007). « A Concise Proof of the Kronecker Polynomial System Solver from Scratch ». In : *Expositiones Mathematicae*, vol. 26, n°2, p. 101–139.

Evdokimov, S. (1994). « Factorization of polynomials over finite fields in subexponential time under GRH ». In : *Algorithmic number theory (Ithaca, NY, 1994)*. Vol. 877. Lecture Notes in Computer Science. Springer-Verlag, p. 209–219.

Everest, Graham, Alf van der Poorten, Igor Shparlinski et Thomas Ward (2003). *Recurrence sequences*. Vol. 104. Mathematical Surveys and Monographs. American Mathematical Society, xiv+318 pages.

Faugère, J. C., P. Gianni, D. Lazard et T. Mora (1993). « Efficient computation of zero-dimensional Gröbner bases by change of ordering ». In : *Journal of Symbolic Computation*, vol. 16, n°4, p. 329–344.

Fettweis, A. (1971). « A general theorem for signal-flow networks, with applications ». In : *Archiv für Elektronik und Übertragungstechnik*, vol. 25, n°12, p. 557–561.

Fiduccia, C. M. (1972a). « On obtaining upper bounds on the complexity of matrix multiplication ». In : *Complexity of computer computations*. IBM Thomas J. Watson Research Center, Yorktown Heights, New York : Plenum, p. 31–40.

— (1972b). « Polynomial evaluation via the division algorithm : The fast Fourier transform revisited. » In : *STOC'72 : ACM Symposium on Theory of Computing*, p. 88–93.

— (1973a). « On the algebraic complexity of matrix multiplication ». Thèse de doct. Brown Univ., Providence, RI, Center Comput. Inform. Sci., Div. Engin.

— (1985). « An efficient formula for linear recurrences ». In : *SIAM Journal on Computing*, vol. 14, n°1, p. 106–112.

Fiduccia, Charles M. (1972c). « On obtaining upper bounds on the complexity of matrix multiplication ». In : *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*. New York : Plenum, p. 31–40.

Fischer, P. C. et R. L. Probert (1974). « Efficient procedures for using matrix algorithms ». In : *Automata, languages and programming*. Vol. 14. Lecture Notes in Computer Science. Springer-Verlag, p. 413–427.

Fischer, Patrick C. (1974). « Further schemes for combining matrix algorithms ». In : *Automata, languages and programming*. Vol. 14. Lecture Notes in Computer Science. Springer-Verlag, p. 428–436.

Fitchas, N., M. Giusti et F. Smietanski (1995). « Sur la complexité du théorème des zéros ». In : *Approximation and optimization in the Caribbean, II (Havana, 1993)*. Vol. 8. Approx. Optim. Frankfurt am Main : Lang, p. 274–329.

Flajolet, Ph. et J.-M. Steyaert (1982). « A branching process arising in dynamic hashing, trie searching and polynomial factorization ». In : *Automata, Languages and Programming*. Éd. par M. Nielsen et E. Schmidt. Vol. 140. Lecture Notes in Computer Science. Springer-Verlag, p. 239–251.

Flajolet, Philippe et Bruno Salvy (1997). « The SIGSAM Challenges : Symbolic Asymptotics in Practice ». In : *ACM SIGSAM Bulletin*, vol. 31, n°4, p. 36–47. doi : 10.1145/274888.274890.

Frigo, M. et S. G Johnson (2005). « The Design and Implementation of FFTW3 ». In : *Proceedings of the IEEE*, vol. 93, n°2, p. 216–231.

Fröhlich, A. et J. C. Shepherdson (1956). « Effective procedures in field theory ». In : *Philosophical Transactions of the Royal Society A*, vol. 248, p. 407–432.

Fürer, Martin (2007). « Faster integer multiplication ». In : *STOC'07 : ACM Symposium on Theory of Computing*. New York : ACM, p. 57–66.

— (2009). « Faster integer multiplication ». In : *SIAM Journal on Computing*, vol. 39, n°3, p. 979–1005. DOI : 10.1137/070711761.

Garner, Harvey L. (1959). « The residue number system ». In : *IRE-AIEE-ACM'59 Western joint computer conference*. San Francisco, California : ACM, p. 146–153. DOI : 10.1145/1457838.1457864.

Gathen, J. von zur (1984). « Hensel and Newton methods in valuation rings ». In : *Mathematics of Computation*, vol. 42, n°166, p. 637–661.

— (1987). « Factoring polynomials and primitive elements for special primes ». In : *Theoretical Computer Science*, vol. 52, n°1-2, p. 77–89.

— (2006). « Who was who in polynomial factorization ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computationt*. ACM Press, p. 1–2.

Gathen, J. von zur et J. Gerhard (2003a). *Modern computer algebra*. 2e éd. Cambridge University Press.

Gathen, Joachim von zur et Thomas Lücking (2003). « Subresultants revisited ». In : *Theoretical Computer Science*, vol. 297, n°1-3, p. 199–239.

Gathen, Joachim von zur et Jürgen Gerhard (1999). *Modern computer algebra*. Cambridge University Press, xiv+753 pages.

— (2003). *Modern computer algebra*. 2e éd. Cambridge University Press, xiv+785 pages.

Gauss, Carl Friedrich (1966). *Disquisitiones arithmeticae*. Translated into English by Arthur A. Clarke, S. J. Yale University Press, xx+472 pages.

Geddes, Keith O., Stephen R. Czapor et George Labahn (1992). *Algorithms for Computer Algebra*. Kluwer Academic Publishers.

Gentleman, W. M. et G. Sande (1966). « Fast Fourier Transforms : for fun and profit ». In : *AFIPS'66*. San Francisco, California : ACM, p. 563–578. DOI : 10.1145/1464291.1464352.

Gianni, P. et B. Trager (1996a). « Square-free algorithms in positive characteristic ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 7, n°1, p. 1–14.

Giesbrecht, M., A. Lobo et B. D. Saunders (1998). « Certifying inconsistency of sparse linear systems ». In : *ISSAC'98 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 113–119.

Giesbrecht, Mark (1995). « Nearly optimal algorithms for canonical matrix forms ». In : *SIAM Journal on Computing*, vol. 24, n°5, p. 948–969.

— (2001). « Fast computation of the Smith form of a sparse integer matrix ». In : *Computational Complexity*, vol. 10, n°1, p. 41–69.

Gilbert, J.-C., G. Le Vey et J. Masse (1991). *La différentiation automatique de fonctions représentées par des programmes*. Rapp. tech. RR INRIA 1557.

Giorgi, Pascal, Claude-Pierre Jeannerod et Gilles Villard (2003). « On the complexity of polynomial matrix computations ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par J. R. Sendra. New York : ACM Press, p. 135–142.

Giusti, M., K. Hägele, J. Heintz, J. L. Montaña, J. E. Morais et L. M. Pardo (1997). « Lower bounds for Diophantine approximations ». In : *J. Pure Appl. Algebra*, vol. 117/118, p. 277–317.

Giusti, M., J. Heintz, J. E. Morais, J. Morgenstern et L. M. Pardo (1998). « Straight-line programs in geometric elimination theory ». In : *J. Pure Appl. Algebra*, vol. 124, n°1-3, p. 101–146.

Giusti, M., J. Heintz, J. E. Morais et L. M. Pardo (1995). « When polynomial equation systems can be "solved" fast ? » In : *Applied algebra, algebraic*

algorithms and error-correcting codes (Paris, 1995). Vol. 948. Lecture Notes in Computer Science. Springer-Verlag, p. 205–231.

Giusti, M., J. Heintz et J. Sabia (1993). « On the efficiency of effective Nullstellensätze ». In : *Computational Complexity*, vol. 3, n°1, p. 56–95.

Giusti, Marc (1988). « Combinatorial dimension theory of algebraic varieties ». In : *Journal of Symbolic Computation*, vol. 6, n°2-3. Special issue on Computational aspects of commutative algebra, p. 249–265.

Giusti, Marc, Klemens Hägele, Grégoire Lecerf, Joël Marchand et Bruno Salvy (2000). « The projective Noether Maple package : computing the dimension of a projective variety ». In : *J. Symbolic Comput.* vol. 30, n°3, p. 291–307.

Giusti, Marc et Joos Heintz (1993). « La détermination des points isolés et de la dimension d'une variété algébrique peut se faire en temps polynomial ». In : *Computational algebraic geometry and commutative algebra (Cortona, 1991)*. Vol. XXXIV. Sympos. Math. Cambridge : Cambridge Univ. Press, p. 216–256.

Giusti, Marc, Grégoire Lecerf et Bruno Salvy (2001). « A Gröbner free alternative for polynomial system solving ». In : *Journal of Complexity*, vol. 17, n°1, p. 154–211.

Gohberg, I. C. et A. A. Semencul (1972). « On the inversion of finite Toeplitz matrices and their continuous analogues (In Russian) ». In : *Matematicheskie Issledovaniya*, vol. 7, n°2, p. 201–223.

Gohberg, I. et V. Olshevsky (1994). « Complexity of multiplication with vectors for structured matrices ». In : *Linear Algebra and its Applications*, vol. 202, p. 163–192.

Golub, Gene H. et Charles F. Van Loan (1996). *Matrix computations*. 3e éd. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, xxx+698 pages.

Gosper, R. William (1978). « Decision procedure for indefinite hypergeometric summation ». In : *Proceedings of the National Academy of Sciences USA*, vol. 75, n°1, p. 40–42.

Gragg, William B., Fred G. Gustavson, Daniel D. Warner et David Y. Y. Yun (1982). « On fast computation of superdiagonal Padé fractions ». In : *Mathematical Programming Study*, n°18. Algorithms and theory in filtering and control (Lexington, Ky., 1980), p. 39–42.

Grigoriev, D. Yu. (1990). « Complexity of factoring and calculating the GCD of linear ordinary differential operators ». In : *Journal of Symbolic Computation*, vol. 10, n°1, p. 7–37.

Gustavson, Fred G. et David Y. Y. Yun (1979). « Fast algorithms for rational Hermite approximation and solution of Toeplitz systems ». In : *IEEE Transactions on Circuits and Systems*, vol. 26, n°9, p. 750–755.

Hanrot, Guillaume, Michel Quercia et Paul Zimmermann (2004). « The Middle Product Algorithm I ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 14, n°6, p. 415–438.

Hanrot, Guillaume et Paul Zimmermann (2004). *Newton iteration revisited*. url : http://www.loria.fr/~zimmerma/papers (visible en 2004).

Hardy, G. H. et E. M. Wright (2008). *An introduction to the theory of numbers*. 6e éd. Oxford University Press, xxii+621 pages.

Harley, Rev. Robert (1862). « On the theory of the transcendental solution of algebraic equations ». In : *Quarterly Journal of Pure and Applied Mathematics*, vol. 5, p. 337–360.

HARRIS, William A. et Yasutaka SIBUYA (1985). « The reciprocals of solutions of linear ordinary differential equations ». In : *Advances in Mathematics*, vol. 58, n°2, p. 119–132.

HARTER, Richard (1972). « The optimality of Winograd's formula ». In : *Communications of the ACM*, vol. 15, n°5, p. 352. DOI : 10.1145/355602.361315.

HARVEY, David (2010). *Faster exponentials of power series*. Rapp. tech. arXiv. eprint : abs/0911.3110.

— (2011). « Faster algorithms for the square root and reciprocal of power series ». In : *Mathematics of Computation*, vol. 80, p. 387–394.

HARVEY, David, Joris van der HOEVEN et Grégoire LECERF (2014a). « Even faster integer multiplication ». http://arxiv.org/abs/1407.3360.

— (2014b). « Faster polynomial multiplication over finite fields ». http://arxiv.org/abs/1407.3361.

HEIDEMAN, Michael T., Don H. JOHNSON et C. Sidney BURRUS (1985). « Gauss and the history of the fast Fourier transform ». In : *Archive for History of Exact Sciences*, vol. 34, n°3, p. 265–277.

HEINDEL, L. E. et E. HOROWITZ (1971). « On decreasing the computing time for modular arithmetic ». In : *12th symposium on switching and automata theory*. IEEE, p. 126–128.

HEINTZ, J. et C.-P. SCHNORR (1982). « Testing polynomials which are easy to compute ». In : *Logic and algorithmic (Zurich, 1980)*. Vol. 30. Monograph. Enseign. Math. Geneva : Univ. Genève, p. 237–254.

HERMITE, C. (1873a). « Extrait d'une lettre de Monsieur Ch. Hermite à Monsieur Paul Gordan ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 78, p. 303–311.

— (1873b). « Sur la fonction exponentielle ». In : *Comptes-rendus de l'Académie des sciences de Paris*, vol. 77, p. 18–24.

— (1893). « Sur la généralisation des fractions continues algébriques ». In : *Annali di Matematica pura ed applicata*, vol. 21, n°2, p. 289–308.

HIRONAKA, Heisuke (1964). « Resolution of singularities of an algebraic variety over a field of characteristic zero. I, II ». In : *Annals of Mathematics. Second Series*, vol. 79, p. 205–326.

HOEIJ, Mark van (1998). « Rational solutions of linear difference equations ». In : *ISSAC'98 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 120–123.

HOEVEN, J. van der, G. LECERF, B. MOURRAIN et al. (2002). *Mathemagix*. http://www.mathemagix.org.

HOEVEN, Joris van der (2002a). « FFT-like multiplication of linear differential operators ». In : *Journal of Symbolic Computation*, vol. 33, n°1, p. 123–127.

— (2002b). « Relax, but don't be too lazy ». In : *Journal of Symbolic Computation*, vol. 34, n°6, p. 479–542.

— (2004). « The truncated Fourier transform and applications ». In : *ISSAC'04 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 290–296.

— (2010). « Newton's method and FFT trading ». In : *Journal of Symbolic Computation*, vol. 45, n°8, p. 857–878.

HONG, Hoon (2001a). « Ore principal subresultant coefficients in solutions ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, n°3, p. 227–237.

— (2001b). « Ore subresultant coefficients in solutions ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 12, n°5, p. 421–428.

Hopcroft, J. E. et L. R. Kerr (1971). « On minimizing the number of multiplications necessary for matrix multiplication ». In : *SIAM Journal on Applied Mathematics*, vol. 20, p. 30–36.

Hopcroft, J. et J. Musinski (1973). « Duality applied to the complexity of matrix multiplication and other bilinear forms ». In : *SIAM Journal on Computing*, vol. 2, p. 159–173.

Horowitz, E. (1972). « A fast Method for Interpolation Using Preconditioning ». In : *Information Processing Letters*, vol. 1, n°4, p. 157–163.

Huang, Ming-Deh A. (1991). « Generalized Riemann hypothesis and factoring polynomials over finite fields ». In : *Journal of Algorithms*, vol. 12, n°3, p. 464–481.

Huss-Lederman, S., E. M. Jacobson, A. Tsao, T. Turnbull et J. R. Johnson (1996). « Implementation of Strassen's algorithm for matrix multiplication ». In : *Supercomputing'96*. 32 pp. Pittsburgh, PA : IEEE Computer Society. doi : 10.1145/369028.369096.

Ja'Ja', Joseph (1979). « On the complexity of bilinear forms with commutativity ». In : *STOC'79 : ACM Symposium on Theory of Computing*. Atlanta, Georgia, United States : ACM, p. 197–208. doi : 10.1145/800135.804413.

Janet, M. (1920). « Sur les systèmes d'équations aux dérivées partielles ». In : *J. Math. Pure et Appl.* vol. 3, p. 65–151.

Jeannerod, Claude-Pierre et Gilles Villard (2005). « Essentially optimal computation of the inverse of generic polynomial matrices ». In : *Journal of Complexity*, vol. 21, n°1, p. 72–86.

Jeronimo, G., T. Krick, J. Sabia et M. Sombra (2004). « The computational complexity of the Chow form ». In : *Foundations of Computational Mathematics. The Journal of the Society for the Foundations of Computational Mathematics*, vol. 4, n°1, p. 41–117.

Jeronimo, G., S. Puddu et J. Sabia (2001). « Computing Chow forms and some applications ». In : *J. Algorithms*, vol. 41, n°1, p. 52–68.

Jeronimo, G. et J. Sabia (2000). « Probabilistic equidimensional decomposition ». In : *Comptes Rendus des Séances de l'Académie des Sciences. Série I. Mathématique*, vol. 331, n°6, p. 485–490.

— (2002). « Effective equidimensional decomposition of affine varieties ». In : *Journal of Pure and Applied Algebra*, vol. 169, n°2-3, p. 229–248.

Kailath, T., S. Y. Kung et M. Morf (1979a). « Displacement ranks of a matrix ». In : *American Mathematical Society. Bulletin. New Series*, vol. 1, n°5, p. 769–773.

Kailath, Thomas, Sun Yuan Kung et Martin Morf (1979b). « Displacement ranks of matrices and linear equations ». In : *Journal of Mathematical Analysis and Applications*, vol. 68, n°2, p. 395–407.

Kalman, R. E. (1959). « On the General Theory of Control Systems ». In : *IRE Transactions on Automatic Control*, vol. 4, n°3, p. 481–491.

Kaltofen, E., R. M. Corless et D. J. Jeffrey (2000). « Challenges of symbolic computation : my favorite open problems ». In : *Journal of Symbolic Computation*, vol. 29, n°6, p. 891–919.

Kaltofen, E. et V. Shoup (1998). « Subquadratic-time factoring of polynomials over finite fields ». In : *Mathematics of Computation*, vol. 67, n°223, p. 1179–1197.

Kaltofen, Erich (1993a). « Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems ». In : *Applied algebra, algebraic algorithms and error-correcting codes*. Vol. 673. Lecture Notes in Computer Science. Springer-Verlag, p. 195–212.

— (1993b). « Computational differentiation and algebraic complexity theory ». In : *Workshop Report on First Theory Institute on Computational Differentiation*, p. 28–30.
— (1994). « Asymptotically fast solution of Toeplitz-like singular linear systems ». In : *ISSAC'94 : International Symposium on Symbolic and Algebraic Computation*. Oxford, United Kingdom : ACM Press, p. 297–304. DOI : 10.1145/190347.190431.
— (1995c). « Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems ». In : *Mathematics of Computation*, vol. 64, n°210, p. 777–806.
KALTOFEN, Erich et B. David SAUNDERS (1991). « On Wiedemann's method of solving sparse linear systems ». In : *Applied algebra, algebraic algorithms and error-correcting codes (New Orleans, LA, 1991)*. Vol. 539. Lecture Notes in Computer Science. Springer-Verlag, p. 29–38.
KAMINSKI, M., D. G. KIRKPATRICK et N. H. BSHOUTY (1988). « Addition requirements for matrix and transposed matrix products ». In : *Journal of Algorithms*, vol. 9, n°3, p. 354–364.
KAMINSKI, Michael (1985). « A lower bound for polynomial multiplication ». In : *Theoretical Computer Science*, vol. 40, n°2-3, p. 319–322. DOI : 10.1016/0304-3975(85)90174-4.
KAPORIN, I (2004). « The aggregation and cancellation techniques as a practical tool for faster matrix multiplication ». In : *Theoretical Computer Science*, vol. 315, n°2-3, p. 469–510.
KAPORIN, Igor (1999). « A practical algorithm for faster matrix multiplication ». In : *Numerical Linear Algebra with Applications*, vol. 6, n°8, p. 687–700.
KARATSUBA, A. et Y. OFMAN (1963). « Multiplication of multidigit numbers on automata ». In : *Soviet Physics Doklady*, vol. 7, p. 595–596.
KARP, A. H. et P. MARKSTEIN (1997b). « High-precision division and square root ». In : *ACM Transactions on Mathematical Software*, vol. 23, n°4, p. 561–589.
KEDLAYA, Kiran S. et Christopher UMANS (2008). « Fast Modular Composition in any Characteristic ». In : *FOCS'08 : IEEE Conference on Foundations of Computer Science*. IEEE Computer Society, p. 146–155. DOI : 10.1109/FOCS.2008.13.
KELLER-GEHRIG, Walter (1985). « Fast algorithms for the characteristic polynomial ». In : *Theoretical Computer Science*, vol. 36, n°2-3, p. 309–317.
KHAN, M. A. H. (2002). « High-order differential approximants ». In : *Journal of Computational and Applied Mathematics*, vol. 149, n°2, p. 457–468.
KLYUYEV, V. V. et N. I. KOKOVKIN-SHCHERBAK (1965). « On the minimization of the number of arithmetic operations for the solution of linear algebraic systems of equations ». In : *USSR Computational Mathematics and Mathematical Physics*, vol. 5, p. 25–43.
KNUTH, D. et P. BENDIX (1970). « Computational problems in abstract algebra ». In : éd. par John LEECH. Pergamon Press. Chap. Simple word problems in universal algebras, p. 263–297.
KNUTH, Donald E. (1971). « The analysis of algorithms ». In : *Actes du Congrès International des Mathématiciens*. Vol. 3. Nice : Gauthier-Villars, p. 269–274.
— (1997). *The Art of Computer Programming*. 3e éd. Vol. 2 : Seminumerical Algorithms. Computer Science and Information Processing. Addison-Wesley Publishing Co., xiv+762 pages.
KNUTH, Donald E. et Christos H. PAPADIMITRIOU (1981). « Duality in addition chains ». In : *Bulletin of the European Association for Theoretical Computer Science*, vol. 13, p. 2–4.

Kogge, P. et H. Stone (1973). « A parallel algorithm for the efficient solution of a general class of recurrence equations ». In : *IEEE Transactions on Computers*, vol. C-22, p. 786–793.

Krick, T. et L. M. Pardo (1996). « A computational method for Diophantine approximation ». In : *Algorithms in algebraic geometry and applications (Santander, 1994)*. Vol. 143. Progr. Math. Birkhäuser, p. 193–253.

Kronecker, Leopold (1882b). « Grundzüge einer arithmetischen Theorie der algebraischen Grössen ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 92, p. 1–122.

Krylov, A. N. (1931). « On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined ». In : *Izvestiya Akademii Nauk SSSR*, vol. 7, n°4. (in Russian), p. 491–539.

Kung, H. T. (1974). « On computing reciprocals of power series ». In : *Numerische Mathematik*, vol. 22, p. 341–348.

Kung, H. T. et D. M. Tong (1977). « Fast algorithms for partial fraction decomposition ». In : *SIAM Journal on Computing*, vol. 6, n°3, p. 582–593.

Laderman, Julian D. (1976). « A noncommutative algorithm for multiplying 3×3 matrices using 23 muliplications ». In : *Bulletin of the American Mathematical Society*, vol. 82, n°1, p. 126–128.

Laderman, Julian, Victor Pan et Xuan He Sha (1992). « On practical algorithms for accelerated matrix multiplication ». In : *Linear Algebra and its Applications*, vol. 162/164, p. 557–588.

Lagrange, Joseph-Louis (1768). « Nouvelle méthode pour résoudre les équations littérales par le moyen des séries ». In : *Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Berlin*, vol. 24, p. 251–326.

Landsberg, J. M. (2006). « The border rank of the multiplication of 2×2 matrices is seven ». In : *Journal of the American Mathematical Society*, vol. 19, n°2, p. 447–459.

— (2008). « Geometry and the complexity of matrix multiplication ». In : *Bulletin of the American Mathematical Society*, vol. 45, n°2, p. 247–284.

Lang, Serge (2002). *Algebra*. 3e éd. Vol. 211. Graduate Texts in Mathematics. Springer-Verlag, xvi+914 pages.

Lascoux, Alain (2003). *Symmetric functions and combinatorial operators on polynomials*. Vol. 99. CBMS Regional Conference Series in Mathematics. Published for the Conference Board of the Mathematical Sciences, Washington, DC, xii+268 pages.

Le Gall, François (2014). « Powers of Tensors and Fast Matrix Multiplication ». In : *ISSAC'14 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Katsusuke Nabeshima. New York, NY, USA : ACM, p. 296–303.

Lecerf, G. (2000). « Computing an equidimensional decomposition of an algebraic variety by means of geometric resolutions ». In : *ISSAC'00 : International Symposium on Symbolic and Algebraic Computation*. ACM, p. 209–216.

— (2002). « Quadratic Newton Iteration for Systems with Multiplicity ». In : *Found. Comput. Math.* vol. 2, n°3, p. 247–293.

— (2003). « Computing the equidimensional decomposition of an algebraic closed set by means of lifting fibers ». In : *J. Complexity*, vol. 19, n°4, p. 564–596.

— (2008a). « Fast Separable Factorization and Applications ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 19, n°2, p. 135–160.

— (2013). « Factorisation des polynômes à plusieurs variables (Cours no. II) ». In : *Journées nationales de calcul formel*. Éd. par G. Chèze, P. Boito, C. Pernet et M. Safey el Din. Vol. 3. Les cours du CIRM n°1. http://ccirm.cedram.org/ccirm-bin/fitem?id=CCIRM_2013__3_1_A2_0. Cedram, p. 1–85.

Lecerf, G. et É. Schost (2003b). « Fast Multivariate Power Series Multiplication in Characteristic Zero ». In : *SADIO Electronic Journal on Informatics and Operations Research*, vol. 5, n°1, p. 1–10.

Lehmer, D. H. (1938). « Euclid's Algorithm for Large Numbers ». In : *The American Mathematical Monthly*, vol. 45, n°4, p. 227–233.

Levinson, Norman (1947). « The Wiener RMS (root mean square) error criterion in filter design and prediction ». In : *Journal of Mathematics and Physics*, vol. 25, p. 261–278.

Li, Z. et I. Nemes (1997). « A modular algorithm for computing greatest common right divisors of Ore polynomials ». In : *ISSAC'97 : International Symposium on Symbolic and Algebraic Computation*. Maui, Hawai : ACM Press, p. 282–289.

Li, Ziming (1998). « A subresultant theory for Ore polynomials with applications ». In : *ISSAC'98 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 132–139.

— (2000). « Greatest common right divisors, least common left multiples, and subresultants of Ore polynomials ». In : *Mathematics mechanization and applications*. San Diego, CA : Academic Press, p. 297–324.

Liouville, Joseph (1833). « Second mémoire sur la détermination des intégrales dont la valeur est algébrique ». In : *Journal de l'École polytechnique*, vol. 14, p. 149–193.

Lipshitz, L. (1989). « *D*-Finite Power Series ». In : *Journal of Algebra*, vol. 122, n°2, p. 353–373.

Lipson, J. D. (1976a). « Newton's Method : A Great Algebraic Algorithm ». In : *SYMSAC'76 : ACM Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 260–270.

Macintyre, Angus et A. J. Wilkie (1996). « On the decidability of the real exponential field ». In : *Kreiseliana*. A. K. Peters, p. 441–467.

Mahler, K. (1968). « Perfect systems ». In : *Compositio Mathematica*, vol. 19, 95–166 (1968).

Makarov, O. M. (1986). « An algorithm for multiplying 3×3 matrices ». In : *USSR Computational Mathematics and Mathematical Physics*, vol. 26, n°1, p. 179–180. doi : 10.1016/0041-5553(86)90203-X.

Marotte, F. M. (1898). « Les équations différentielles linéaires et la théorie des groupes ». In : *Annales de la Faculté des Sciences de Toulouse*, vol. 12, n°4, p. 1–92. url : http://www.numdam.org/item?id=AFST_1898_1_12_4_H1_0.

Massey, James L. (1969). « Shift-register synthesis and BCH decoding ». In : *IEEE Transactions on Information Theory*, vol. IT-15, p. 122–127.

Matiyasevich, Yuri V. (1993). *Hilbert's tenth problem*. Foundations of Computing Series. Translated from the 1993 Russian original by the author, With a foreword by Martin Davis. MIT Press, xxiv+264 pages.

Mayr, Ernst W. et Albert R. Meyer (1982). « The complexity of the word problems for commutative semigroups and polynomial ideals ». In : *Advances in Mathematics*, vol. 46, n°3, p. 305–329.

McEliece, Robert J. et James B. Shearer (1978). « A property of Euclid's algorithm and an application to Padé approximation ». In : *SIAM Journal on Applied Mathematics*, vol. 34, n°4, p. 611–615.

Mersereau, Russell M. (1974). « An algorithm for performing an inverse chirp z-transform ». In : *IEEE Transactions on Audio and Electroacoustics*, vol. ASSP-22, n°5, p. 387–388.

Mignotte, M. et C. Schnorr (1988). « Calcul déterministe des racines d'un polynôme dans un corps fini ». In : *Comptes Rendus des Séances de l'Académie des Sciences. Série I. Mathématique*, vol. 306, n°12, p. 467–472.

Miller, J. C. P. et D. J. Spencer Brown (1966). « An algorithm for evaluation of remote terms in a linear recurrence sequence ». In : *Computer Journal*, vol. 9, p. 188–190.

Mills, W. H. (1975). « Continued fractions and linear recurrences ». In : *Mathematics of Computation*, vol. 29, p. 173–180.

Mines, R., F. Richman et W. Ruitenburg (1988b). *A course in constructive algebra*. Universitext. Springer-Verlag.

Moenck, R. T. (1973). « Fast computation of GCDs ». In : *STOC'73 : ACM Symposium on Theory of Computing*. Austin : ACM, p. 142–151.

— (1977). « On the efficiency of algorithms for polynomial factoring ». In : *Mathematics of Computation*, vol. 31, p. 235–250.

Moenck, R. T. et A. Borodin (1972). « Fast modular transforms via division ». In : *Thirteenth Annual IEEE Symposium on Switching and Automata Theory*, p. 90–96.

Moenck, Robert T. (1976). « Another polynomial homomorphism ». In : *Acta Informatica*, vol. 6, n°2, p. 153–169.

Moenck, Robert T. et John H. Carter (1979). « Approximate algorithms to derive exact solutions to systems of linear equations ». In : *EUROSAM'79 : International Symposium on Symbolic and Algebraic Computation*. Vol. 72. Lecture Notes in Computer Science. London, UK : Springer-Verlag, p. 65–73.

Möller, H. Michael et Ferdinando Mora (1986). « New constructive methods in classical Ideal theory ». In : *J. Algebra*, vol. 100, n°1, p. 138–178.

Möller, Niels (2008). « On Schönhage's algorithm and subquadratic integer GCD computation ». In : *Mathematics of Computation*, vol. 77, n°261, p. 589–607.

Montgomery, P. L. (1992). « An FFT extension of the elliptic curve method of factorization ». Thèse de doct. University of California, Los Angeles CA.

Montgomery, Peter L. (1985). « Modular multiplication without trial division ». In : *Mathematics of Computation*, vol. 44, n°170, p. 519–521.

— (2005). « Five, Six, and Seven-Term Karatsuba-Like Formulae ». In : *IEEE Transactions on Computers*, vol. 54, n°3, p. 362–369. doi : 10.1109/TC.2005.49.

Mora, Teo (2005). *Solving Polynomial Equation Systems II : Macaulay's Paradigm and Gröbner Technology*. Vol. 99. Encyclopedia of Mathematics and its Applications. Cambridge Univ. Press.

Morf, M. (1980). « Doubling algorithms for Toeplitz and related equations ». In : *IEEE Conference on Acoustics, Speech, and Signal Processing*, p. 954–959.

Morgenstern, Jacques (1973). « Note on a lower bound of the linear complexity of the fast Fourier transform ». In : *Journal of the Association for Computing Machinery*, vol. 20, p. 305–306.

Mulders, Thom (2000). « On Short Multiplications and Divisions ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, n°1, p. 69–88.

— (2004). « Certified sparse linear system solving ». In : *Journal of Symbolic Computation*, vol. 38, n°5, p. 1343–1373.

Mullen, G. L. et D. Panario (2013). *Handbook of Finite Fields*. Discrete Mathematics and Its Applications. Chapman et Hall/CRC.

Munro, I. (1973). « Problems related to matrix multiplication ». In : *Proceedings Courant Institute Symposium on Computational Complexity, October 1971*. Éd. par R. Rustin. New York : Algorithmics Press, p. 137–151.

Musser, D. R. (1971). « Algorithms for Polynomial Factorization ». Thèse de doct. Univ. of Wisconsin : C.S. Department.

Newton, Isaac (1740). *La méthode des fluxions, et les suites infinies*. Traduction française par G. Buffon du texte de 1671 de Newton. de Bure. URL : http://www.gallica.fr.

Nussbaumer, Henri J. (1980). « Fast polynomial transform algorithms for digital convolution ». In : *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, n°2, p. 205–215.

— (1981). *Fast Fourier transform and convolution algorithms*. Vol. 2. Springer Series in Information Sciences. Springer-Verlag, x+248 pages.

Ore, Oystein (1931). « Linear equations in non-commutative fields ». In : *Annals of Mathematics*, vol. 32, p. 463–477.

— (1933). « Theory of non-commutative polynomials ». In : *Annals of Mathematics*, vol. 34, n°3, p. 480–508.

Ostrowski, A. (1954). « On two problems in abstract algebra connected with Horner's rule ». In : *Studies in mathematics and mechanics presented to Richard von Mises*. NY : Academic Press Inc., p. 40–48.

Padé, H. (1892). « Sur la Représentation Approchée d'une Fonction par des Fractions Rationelles ». In : *Annales scientifiques de l'É.N.S.* vol. 9, p. 3–93.

— (1894). « Sur la généralisation des fractions continues algébriques ». In : *Journal de mathématiques pures et appliquées*, vol. 10, n°4, p. 291–330.

Pan, V. (1972). « Computation schemes for a product of matrices and for the inverse matrix ». In : *Akademiya Nauk SSSR i Moskovskoe Matematicheskoe Obshchestvo. Uspekhi Matematicheskikh Nauk*, vol. 27, n°5(167), p. 249–250.

Pan, V. Y. et X. Wang (2002). « Acceleration of Euclidean algorithm and extensions ». In : *ISSAC'02 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Teo Mora. Lille : ACM Press, p. 207–213.

Pan, V. Ya. (1966b). « Methods of computing values of polynomials ». In : *Russian Mathematical Surveys*, vol. 21, n°1, p. 105–136.

— (1978). « Strassen's algorithm is not optimal. Trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations ». In : *SFCS '78*. Washington, DC, USA : IEEE Computer Society, p. 166–176. DOI : 10.1109/SFCS.1978.34.

— (1980). « New fast algorithms for matrix operations ». In : *SIAM Journal on Computing*, vol. 9, n°2, p. 321–342.

— (1981). « New combinations of methods for the acceleration of matrix multiplication ». In : *Computers & Mathematics with Applications. An International Journal*, vol. 7, n°1, p. 73–125.

Pan, Victor (1984a). « How can we speed up matrix multiplication ? » In : *SIAM Review*, vol. 26, n°3, p. 393–415.

— (1984b). *How to multiply matrices faster*. Vol. 179. Lecture Notes in Computer Science. Springer-Verlag, xi+212 pages.

— (1990). « On computations with dense structured matrices ». In : *Mathematics of Computation*, vol. 55, n°191, p. 179–190.

Pan, Victor Y. (2001). *Structured matrices and polynomials*. Unified superfast algorithms. Birkhäuser Boston Inc., xxvi+278 pages.

Pan, Victor Y. et Xinmao Wang (2004). « On rational number reconstruction and approximation ». In : *SIAM Journal on Computing*, vol. 33, n°2, p. 502–503.

Pan, Victor Y. et Ailong Zheng (2000). « Superfast algorithms for Cauchy-like matrix computations and extensions ». In : *Linear Algebra and its Applications*, vol. 310, n°1-3, p. 83–108.

PARI/GP (2012). Software available from http://pari.math.u-bordeaux.fr. The PARI Group. Bordeaux.

Paszkowski, Stefan (1987). « Recurrence relations in Padé-Hermite approximation ». In : *Journal of Computational and Applied Mathematics*, vol. 19, n°1, p. 99–107.

Paterson, M. S., M. J. Fischer et A. R. Meyer (1974). « An improved overlap argument for on-line multiplication ». In : *Complexity of computation*. AMS, p. 97–111.

Paterson, M. S. et L. J. Stockmeyer (1973). « On the Number of Nonscalar Multiplications Necessary to Evaluate Polynomials ». In : *SIAM Journal on Computing*, vol. 2, n°1, p. 60–66.

Paterson, Michael S. (1975). « Complexity of monotone networks for Boolean matrix product ». In : *Theoretical Computer Science*, vol. 1, n°1, p. 13–20.

Penfield Jr., P., R. Spence et S. Duinker (1970). *Tellegen's theorem and electrical networks*. The M.I.T. Press, Cambridge, Mass.-London, xv+143 pages.

Pernet, Clément et Arne Storjohann (2007). « Faster algorithms for the characteristic polynomial ». In : *ISSAC'07 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Dongming Wang. ACM Press, p. 307–314.

Petkovšek, Marko, Herbert S. Wilf et Doron Zeilberger (1996). $A = B$. A. K. Peters, xii+212 pages.

Probert, Robert L. (1976). « On the additive complexity of matrix multiplication ». In : *SIAM Journal on Computing*, vol. 5, n°2, p. 187–203.

Put, Marius van der et Michael F. Singer (1997). *Galois theory of difference equations*. Vol. 1666. Lecture Notes in Mathematics. Springer-Verlag, viii+180 pages.

— (2003). *Galois theory of linear differential equations*. Vol. 328. Grundlehren der Mathematischen Wissenschaften. Springer-Verlag, xviii+438 pages.

Rabiner, L. R., R. W. Schafer et C. M. Rader (1969). « The chirp z-transform algorithm and its application ». In : *Bell System Technical Journal*, vol. 48, p. 1249–1292.

Raz, Ran (2003). « On the complexity of matrix product ». In : *SIAM Journal on Computing*, vol. 32, n°5, p. 1356–1369. doi : 10.1137/S0097539702402147.

Richardson, Daniel (1968). « Some Undecidable Problems Involving Elementary Functions of a Real Variable ». In : *Journal of Symbolic Logic*, vol. 33, n°4, p. 514–520.

— (1969). « Solution of the identity problem for integral exponential functions ». In : *Zeitschrift fur mathematischen Logik und Grundlagen der Mathematik*, vol. 15, p. 333–340.

— (1997a). « How to recognize zero ». In : *Journal of Symbolic Computation*, vol. 24, n°6, p. 627–645.

Risler, Jean-Jacques et Felice Ronga (1990). « Testing polynomials ». In : *Journal of Symbolic Computation*, vol. 10, n°1, p. 1–5.

Ritzmann, P. (1986). « A fast numerical algorithm for the composition of power series with complex coefficients ». In : *Theoretical Computer Science*, vol. 44, p. 1–16.

Roman, Steven (1984). *The umbral calculus*. Vol. 111. Pure and Applied Mathematics. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], x+193 pages.

RÓNYAI, L. (1989b). « Factoring polynomials modulo special primes ». In : *Combinatorica. An International Journal on Combinatorics and the Theory of Computing*, vol. 9, n°2, p. 199–206.
— (1992). « Galois Groups and Factoring Polynomials over Finite Fields ». In : *SIAM Journal on Discrete Mathematics*, vol. 5, n°3, p. 345–365.
ROURA, Salvador (2001). « Improved master theorems for divide-and-conquer recurrences ». In : *Journal of the Association for Computing Machinery*, vol. 48, n°2, p. 170–205. DOI : 10.1145/375827.375837.
STEIN, W. A. et al. (2004). *Sage Mathematics Software*. http://www.sagemath.org. The Sage Development Team.
SAITO, Mutsumi, Bernd STURMFELS et Nobuki TAKAYAMA (2000). *Gröbner deformations of hypergeometric differential equations*. Springer-Verlag, viii+254 pages.
SALVY, Bruno et Paul ZIMMERMANN (1994). « Gfun : a Maple package for the manipulation of generating and holonomic functions in one variable ». In : *ACM Transactions on Mathematical Software*, vol. 20, n°2, p. 163–177. DOI : 10.1145/178365.178368.
SCHÖNHAGE, A. (1971b). « Schnelle Berechnung von Kettenbruchentwicklugen ». In : *Acta Informatica*, vol. 1, p. 139–144.
— (1972/73). « Unitäre Transformationen grosser Matrizen ». In : *Numerische Mathematik*, vol. 20, p. 409–417.
— (1976/77). « Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2 ». In : *Acta Informatica*, vol. 7, n°4, p. 395–398.
— (1981). « Partial and total matrix multiplication ». In : *SIAM Journal on Computing*, vol. 10, n°3, p. 434–455.
— (1982). *The fundamental theorem of algebra in terms of computational complexity*. Preliminary Report. 73 pages. Tübingen, Germany : Mathematisches Institut der Universität.
— (1988). « Probabilistic computation of integer polynomial GCDs ». In : *Journal of Algorithms*, vol. 9, n°3, p. 365–371.
SCHÖNHAGE, A. et V. STRASSEN (1971). « Schnelle Multiplikation großer Zahlen ». In : *Computing*, vol. 7, p. 281–292.
SCHOOF, R. (1985). « Elliptic curves over finite fields and the computation of square roots mod p ». In : *Mathematics of Computation*, vol. 44, n°170, p. 483–494.
SCHREYER, F. (1980). « Die Berechnung Syzygien mit dem verallgemeinerten Weierstrasschen Divisionsatz ». Diplomarbeit. Hamburg.
SCHULZ, G. (1933). « Iterative Berechnung der reziproken Matrix ». In : *Zeitschrift für angewandte Mathematik und Physik*, vol. 13, p. 57–59.
SCHWARTZ, J. T. (1980b). « Fast probabilistic algorithms for verification of polynomial identities ». In : *Journal of the Association for Computing Machinery*, vol. 27, n°4, p. 701–717.
SERGEYEV, A. V. (1986). « A recursive algorithm for Padé-Hermite approximations ». In : *USSR Comput. Maths Math. Phys*, vol. 26, n°2, p. 17–22.
SHAFER, R. E. (1974). « On quadratic approximation ». In : *SIAM Journal on Numerical Analysis*, vol. 11, p. 447–460.
SHOUP, V. (1990). « On the deterministic complexity of factoring polynomials over finite fields ». In : *Information Processing Letters*, vol. 33, n°5, p. 261–267.
— (1991a). « A Fast Deterministic Algorithm for Factoring Polynomials over Finite Fields of Small Characteristic ». In : *ISSAC'91 : International Symposium on Symbolic and Algebraic Computation*. Éd. par S. M. WATT. ACM Press, p. 14–21.

SHOUP, V. (1991b). « A Fast Deterministic Algorithm for Factoring Polynomials over Finite Fields of Small Characteristic ». In : *ISSAC'91 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 14–21.
— (1995). « A new polynomial factorization algorithm and its implementation ». In : *Journal of Symbolic Computation*, vol. 20, n°4, p. 363–397.
— (1999). « Efficient computation of minimal polynomials in algebraic extensions of finite fields ». In : *ISSAC'99 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 53–58.
— (2009). *A Computational Introduction to Number Theory and Algebra*. 2e éd. Cambridge University Press.
— (from 1996). *NTL : A Library for doing Number Theory*. http://www.shoup.net.
SHOUP, Victor (1994). « Fast construction of irreducible polynomials over finite fields ». In : *Journal of Symbolic Computation*, vol. 17, n°5, p. 371–391.
SHOUP, Victor et Roman SMOLENSKY (1996/97). « Lower bounds for polynomial evaluation and interpolation problems ». In : *Computational Complexity*, vol. 6, n°4, p. 301–311.
SIEVEKING, M. (1972). « An algorithm for division of powerseries ». In : *Computing*, vol. 10, p. 153–156.
SINGER, M. F. (2007). « Introduction to the Galois Theory of Linear Differential Equations ». In :
SINGER, Michael F. (1981). « Liouvillian solutions of n-th order homogeneous linear differential equations ». In : *American Journal of Mathematics*, vol. 103, n°4, p. 661–682.
— (1986). « Algebraic relations among solutions of linear differential equations ». In : *Transactions of the American Mathematical Society*, vol. 295, n°2, p. 753–763.
SINGER, Michael F. et Felix ULMER (1997). « Linear differential equations and products of linear forms ». In : *Journal of Pure and Applied Algebra*, vol. 117/118, p. 549–563.
SMITH, Warren D. (2002). « Fast matrix multiplication formulae – report of the prospectors ». Preprint. URL : http://www.math.temple.edu/~wds/prospector.pdf.
SPEAR, D. (1978). « A constructive approach to commutative ring theory ». In : *1977 MACSYMA User's Conference*, p. 369–376.
STANLEY, R. P. (1980). « Differentiably Finite Power Series ». In : *European Journal of Combinatorics*, vol. 1, n°2, p. 175–188.
STANLEY, Richard P. (1999). *Enumerative combinatorics*. Vol. 2. Cambridge University Press, xii+581 pages.
STEEL, A. (2005b). « Conquering inseparability : primary decomposition and multivariate factorization over algebraic function fields of positive characteristic ». In : *Journal of Symbolic Computation*, vol. 40, n°3, p. 1053–1075.
STEHLÉ, D. et P. ZIMMERMANN (2004). « A binary recursive Gcd algorithm ». In : *ANTS-VI*. Vol. 3076. Lecture Notes in Computer Science. Springer-Verlag, p. 411–425.
STORJOHANN, Arne (2001). « Deterministic computation of the Frobenius form (extended abstract) ». In : *FOCS'01 : IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, p. 368–377.
— (2002). « High-Order Lifting ». In : *ISSAC'02 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Teo MORA. ACM Press, p. 246–254.

— (2003). « High-order lifting and integrality certification ». In : *Journal of Symbolic Computation*, vol. 36, n°3-4, p. 613–648.
— (2005). « The shifted number system for fast linear algebra on integer matrices ». In : *Journal of Complexity*, vol. 21, n°4, p. 609–650.
Storjohann, Arne et Gilles Villard (2005). « Computing the rank and a small nullspace basis of a polynomial matrix ». In : *ISSAC'05 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 309–316.
Strassen, V. (1969). « Gaussian Elimination is Not Optimal ». In : *Numerische Mathematik*, vol. 13, p. 354–356.
— (1972/73). « Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten ». In : *Numerische Mathematik*, vol. 20, p. 238–251.
— (1976/77). « Einige Resultate über Berechnungskomplexität ». In : *Jahresbericht der Deutschen Mathematiker-Vereinigung*, vol. 78, n°1, p. 1–8.
— (1981). « The computational complexity of continued fractions ». In : *SYMSAC'81*. Snowbird, Utah, United States : ACM, p. 51–67. doi : 10.1145/800206.806371.
— (1983). « The computational complexity of continued fractions ». In : *SIAM Journal on Computing*, vol. 12, n°1, p. 1–27.
— (1987). « Relative bilinear complexity and matrix multiplication ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 375/376, p. 406–443.
Strassen, Volker (1990). « Algebraic complexity theory ». In : *Handbook of theoretical computer science, Vol. A*. Amsterdam : Elsevier, p. 633–672.
Svoboda, A. et M. Valach (1955). « Operátorové obvody (Operational circuits) ». In : *Stroje na Zpracování Informací (Information Processing Machines)*, vol. 3, p. 247–295.
Sýkora, Ondrej (1977). « A fast non-commutative algorithm for matrix multiplication ». In : *Mathematical foundations of computer science (Proc. Sixth Sympos., Tatranská Lomnica, 1977)*. Vol. 53. Lecture Notes in Computer Science. Springer-Verlag, p. 504–512.
Sylvester, James Joseph (1839). « On rational derivation from equations of coexistence, that is to say, a new and extended theory of elimination, *1839–40* ». In : *The Collected mathematical papers of James Joseph Sylvester*. Baker, H. F., p. 40–53.
— (1840). « A method of determining by mere inspection the derivatives from two equations of any degree ». In : *Philosophical Magazine Series 3*, vol. 16, n°101, p. 132–135.
Takahasi, H. et Y. Ishibashi (1961). « A new method for exact calculation by a digital computer ». In : *Information Processing in Japan*, p. 28–42.
Takayama, Nobuki (1989). « Gröbner basis and the problem of contiguous relations ». In : *Japan Journal of Applied Mathematics*, vol. 6, n°1, p. 147–160.
— (1992). « An approach to the zero recognition problem by Buchberger algorithm ». In : *Journal of Symbolic Computation*, vol. 14, n°2-3, p. 265–282.
Tannery, Jules (1874). « Propriétés des intégrales des équations différentielles linéaires à coefficients variables ». Thèse de doctorat ès sciences mathématiques. Faculté des Sciences de Paris. url : http://gallica.bnf.fr.
Tellegen, B. (1952). « A general network theorem, with applications ». In : *Philips Research Reports*, vol. 7, p. 259–269.
Thomé, E. (2002). « Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm ». In : *Journal of Symbolic Computation*, vol. 33, n°5, p. 757–775.

Toom, A. L. (1963). « The complexity of a scheme of functional elements simulating the multiplication of integers ». In : *Doklady Akademii Nauk SSSR*, vol. 150, p. 496–498.

Tourigny, Y. et Ph. G. Drazin (2000). « The asymptotic behaviour of algebraic approximants ». In : *The Royal Society of London. Proceedings. Series A. Mathematical, Physical and Engineering Sciences*, vol. 456, n°1997, p. 1117–1137.

Trench, William F. (1964). « An algorithm for the inversion of finite Toeplitz matrices ». In : *J. Soc. Indust. Appl. Math.* vol. 12, p. 515–522.

Turner, William J. (2006). « A block Wiedemann rank algorithm ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 332–339.

Valiant, L. G. (1979). « The complexity of computing the permanent ». In : *Theoretical Computer Science*, vol. 8, n°2, p. 189–201.

Van Barel, M. et A. Bultheel (1992). « A general module-theoretic framework for vector M-Padé and matrix rational interpolation ». In : *Numerical Algorithms*, vol. 3, n°1-4, p. 451–461.

Van Barel, Marc et Adhemar Bultheel (1991). « The computation of nonperfect Padé-Hermite approximants ». In : *Numerical Algorithms*, vol. 1, n°3, p. 285–304.

Van Loan, Charles (1992). *Computational frameworks for the fast Fourier transform*. Vol. 10. Frontiers in Applied Mathematics. SIAM, xiv+273 pages.

Villard, G. (1997a). « Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems ». In : *ISSAC'97 : International Symposium on Symbolic and Algebraic Computation*. Kihei, Maui, Hawaii, United States : ACM Press, p. 32–39. doi : 10.1145/258726.258742.

Waksman, Abraham (1970). « On Winograd's algorithm for inner products ». In : *IEEE Transactions on Computers*, vol. C-19, n°4, p. 360–361.

Wang, Xinmao et Victor Y. Pan (2003). « Acceleration of Euclidean algorithm and rational number reconstruction ». In : *SIAM Journal on Computing*, vol. 32, n°2, p. 548–556.

Waring, E. (1779). « Problems concerning interpolations ». In : *Philosophical Transactions of the Royal Society of London*, vol. 59, p. 59–67.

Weimerskirch, André et Christof Paar (2006). *Generalizations of the Karatsuba Algorithm for Efficient Implementations*. Cryptology ePrint Archive : 2006/224. (Visible en 2006).

Welch, L. R. et R. A. Scholtz (1979). « Continued fractions and Berlekamp's algorithm ». In : *IEEE Transactions on Information Theory*, vol. 25, n°1, p. 19–27.

Wiedemann, D. (1986). « Solving sparse linear equations over finite fields ». In : *IEEE Transactions on Information Theory*, vol. IT-32, p. 54–62.

Williams, Virginia Vassilevska (2012). « Multiplying matrices faster than Coppersmith-Winograd ». In : *STOC'12 : ACM Symposium on Theory of Computing*. New York, NY : ACM, p. 887–898. doi : 10.1145/2213977.2214056.

— (2014). « Multiplying matrices in $O(n^{2.373})$ time ». http://theory.stanford.edu/~virgi/matrixmult-f.pdf.

Winograd, S. (1967). « On the number of multiplications required to compute certain functions ». In : *Proceedings of the National Academy of Sciences of the United States of America*, vol. 58, p. 1840–1842.

— (1968). « A new algorithm for inner-product ». In : *IEEE Transactions on Computers*, vol. 17, p. 693–694.

— (1971). « On multiplication of 2×2 matrices ». In : *Linear Algebra and Applications*, vol. 4, p. 381–388.

YAMAMOTO, Tetsuro (2000). « Historical developments in convergence analysis for Newton's and Newton-like methods ». In : *Journal of Computational and Applied Mathematics*, vol. 124, n°1-2, p. 1–23.

YAP, Chee (2000). *Fundamental Problems in Algorithmic Algebra*. Oxford University Press.

— (2011). « A real elementary approach to the master recurrence and generalizations ». In : *Theory and applications of models of computation*. Vol. 6648. Lecture Notes in Computer Science. Tokyo, Japan : Springer-Verlag, p. 14–26. DOI : 10.1007/978-3-642-20877-5_3.

YPMA, Tjalling J. (1995). « Historical development of the Newton-Raphson method ». In : *SIAM Review*, vol. 37, n°4, p. 531–551. DOI : 10.1137/1037125.

YUN, David Y.Y. (1976). « On square-free decomposition algorithms ». In : *SYMSAC'76*. Yorktown Heights, New York, United States : ACM, p. 26–35. DOI : 10.1145/800205.806320.

— (1977). « On the equivalence of polynomial GCD and squarefree factorization problems ». In : *MACSYMA Users' Conference*. NASA, Langley Res. Center, Washington D.C., United States, p. 65–70.

ZASSENHAUS, H. (1969b). « On Hensel Factorization I ». In : *J. Number Theory*, vol. 1, n°1, p. 291–311.

ZEILBERGER, Doron (1990). « A holonomic systems approach to special functions identities ». In : *Journal of Computational and Applied Mathematics*, vol. 32, n°3, p. 321–368. DOI : 10.1016/0377-0427(90)90042-X.

ZIERLER, N. (1968). « Linear recurring sequences and error-correcting codes ». In : *Error Correcting Codes (Proc. Sympos. Math. Res. Center, Madison, Wis., 1968)*. Wiley, p. 47–59.

ZIPPEL, R. (1990). « Interpolating polynomials from their values ». In : *Journal of Symbolic Computation*, vol. 9, n°3, p. 375–403.

ZIPPEL, Richard (1979c). « Probabilistic algorithms for sparse polynomials ». In : *EUROSAM'79*. Vol. 72. Lecture Notes in Computer Science. Springer-Verlag, p. 216–226.

ŹRAŁEK, B. (2010). « Using partial smoothness of $p-1$ for factoring polynomials modulo p ». In : *Mathematics of Computation*, vol. 79, p. 2353–2359.