Master 1 - Calcul Formel

Année Universitaire 2016-2017

Notes de Cours

Thomas Cluzeau
Université de Limoges - CNRS ; XLIM UMR 7252
123 avenue Albert Thomas, 87060 Limoges CEDEX
thomas.cluzeau@unilim.fr
http://www.unilim.fr/pages_perso/thomas.cluzeau/

Table des matières

Chapitre 1

Introduction

1.1 Qu'est-ce que le calcul formel ?

En mathématiques, lorsqu'un problème consiste à étudier l'existence d'un objet répondant à certains critères, il existe deux façons différentes d'y apporter une réponse. La première consiste à prouver l'existence de l'objet en question sans se préoccuper de sa construction alors que la seconde produit un procédé de construction de l'objet (ce qui prouve donc son existence). C'est ce qu'on appelle une preuve *constructive*. On distingue alors deux types de procédés constructifs : les procédés *numériques* qui fournissent une approximation du résultat et les procédés *formels* qui fournissent un résultat exact. Ceux sont ces derniers qui seront étudiés dans ce cours.

Le terme *calcul formel* (en anglais *computer algebra* ou *symbolic computation*) désigne l'étude de la représentation et de la manipulation effective d'objets mathématiques dans un ordinateur. En d'autres termes, le calcul formel consiste à étudier dans quelle mesure et à quelle vitesse un ordinateur peut faire des mathématiques. Pour un problème donné, nous serons donc amené à produire un algorithme (*i.e.*, une suite de règles opératoires permettant, en un nombre finie d'étapes, d'arriver avec certitude au résultat) et en évaluer les performances.

Les systèmes de calcul formel manipulent principalement trois types d'objets à savoir des nombres, *e.g.*, des entiers relatifs (arithmétique), des polynômes et des matrices (algèbre linéaire). Un premier aspect du calcul formel consiste donc à proposer des algorithmes de calcul efficaces pour la manipulation de ces structures de bases. Un deuxième aspect se propose de ramener d'autres problèmes à ces derniers comme par exemple montrer que la plupart des opérations d'algèbre linéaire peuvent se réduire à calculer des produits de matrices (voir le théorème 2.3.1 dans la section 2.3). Les procédés que nous construirons reposeront toujours sur de l'algèbre donc avant de construire un algorithme il faudra toujours se placer dans un certain modèle/cadre algébrique.

Le calcul formel est une discipline récente qui a été surtout développée, à la fois par des mathématiciens, des informaticiens et des physiciens, à partir de la seconde moitié du XX^{e} siècle et l'apparition des ordinateurs. Aujourd'hui les grands systèmes de calcul formel

comme MAPLE ou MATHEMATICA sont utilisés dans l'enseignement et la recherche mais aussi dans l'industrie.

1.2 Structures algébriques effectives

Étant donnée une structure algébrique (*e.g.*, groupe, anneau, corps, espace vectoriel) la première question qui se pose avant de construire des algorithmes manipulant des objects de cette structure est le problème de la *calculabilité* : suis-je capable de faire des calculs dans cette structure et d'apprendre à un ordinateur à les faire ? Le principal problème est le *test d'égalité* c'est-à-dire le problème de décider si une expression donnée dans cette structure vaut 0 ou non. En effet pour effectuer certains calculs, comme par exemple une division, il est souvent crucial de déterminer si une expression représente 0 ou non. Notons que ce test à zéro n'est pas décidable dans l'ensemble des nombres réels $\mathbb{R}$. En calcul formel, on se ramène donc toujours à faire des calculs dans une structure *effective* où le test à zéro est décidable.

Définition 1.2.1. *Une structure algébrique est dite* effective *si l'on dispose :*

1. *d'une structure de données pour en représenter les éléments ;*
2. *d'algorithmes pour en effectuer les opérations et pour y tester l'égalité à zéro.*

Le lemme suivant regroupe les résultats d'effectivité sur lesquels nous nous appuierons dans ce cours.

Lemme 1.2.1. *On a les résultats suivants :*

1. *L'anneau $\mathbb{Z}$ des entiers relatifs est effectif ;*
2. *Si $\mathbb{A}$ est un anneau effectif, alors son corps des fractions est effectif, $\mathbb{A}[X]$ est effectif et pour $N \in \mathbb{N}$, $\mathbb{A}[X]/(X^{N+1})$ est effectif ;*
3. *Si $\mathbb{K}$ est un corps effectif, alors sa cloture algébrique $\overline{\mathbb{K}}$, l'espace vectoriel $\mathbb{K}^n$ et l'anneau $\mathbb{M}_n(\mathbb{K})$ sont effectifs.*

1.3 Analyse d'un algorithme

Le calcul formel a pour but de produire des algorithmes qui calculent des objects mathématiques. Rappelons tout d'abord que la correction d'un algorithme est définie comme suit :

Définition 1.3.1. *Un algorithme est dit* correct *si :*

1. *chaque étape est bien définie ;*
2. *l'algorithme se termine en un nombre fini d'étapes ;*
3. *le résultat renvoyé est toujours le bon, i.e., celui que l'on attend.*

Le problème de la calculabilité abordé ci-dessus indique uniquement la faisabilité de certaines opérations. Pour un problème mathématique donné, il peut exister plusieurs algorithmes menant au même résultat. Le calcul formel s'intéresse alors à la comparaison entre ces différents algorithmes pour pouvoir exprimer le fait que pour un problème donné certains algorithmes sont « meilleurs » que d'autres. Pour ce faire, on évalue l'efficacité d'un algorithme en estimant sa *complexité*. Estimer la complexité d'un algorithme permet d'évaluer sa qualité indépendamment de la manière dont il est programmé.

En pratique, une fois l'algorithme programmé sur un ordinateur, ses performances sont essentiellement mesurées par le temps d'exécution du programme (et la quantité de mémoire nécessaire à son exécution). Le temps d'exécution va dépendre à la fois du nombre d'opérations effectuées et du coût de chaque opération. Le programmeur contrôle essentiellement le nombre d'opérations effectuées donc c'est avant tout sur ce point qu'il doit agir pour optimiser les performances de son programme. Une estimation *a priori* du temps d'exécution peut donc être obtenue en analysant le programme c'est-à-dire en comptant le nombre d'opérations effectuées, le nombre d'appels récursifs, ...

Dans le cadre de ce cours, c'est justement le nombre d'opérations arithmétiques effectuées par un algorithme qui nous servira de *mesure de complexité*. Cette mesure est évaluée en fonction d'une (ou plusieurs) grandeurs apparaissant dans l'entrée de l'algorithme (*e.g.*, le degré d'un polynôme, la taille d'une matrice, ...). De plus nous n'avons pas toujours une mesure exacte du nombre d'opérations mais souvent un ordre de grandeur (ou une borne supérieure). Nous utilisons alors la notation $O(.)$ pour exprimer la complexité de nos algorithmes.

Définition 1.3.2. *Soient f et g deux fonctions. On note $f(n) = O(g(n))$ s'il existe $K > 0$ et $N > 0$ tels que pour tout $n > N$, $|f(n)| \leq K\,|g(n)|$.*

Un algorithme sera dit *linéaire* (resp. *quadratique, cubique, logarithmique, polynomial*) s'il utilise $O(n)$ (resp. $O(n^2)$, $O(n^3)$, $O(\log(n))$, $O(n^k)$ avec $k \in \mathbb{N}$) opérations arithmétiques. Notons que ceci donne une estimation *asymptotique* de la complexité (l'inégalité est vérifiée pour $n > N$) et que la valeur de la constante K peut jouer un rôle important dans les performances de l'algorithme.

Exemple 1.3.1. *Soit $\mathbb{K}$ un corps effectif et considérons le problème de l'évaluation d'un polynôme $P = \sum_{i=0}^{n} p_i\,X^i \in \mathbb{K}[X]$ en un point $a \in \mathbb{K}$.*
En faisant les calculs naïvement, nous devons calculer a^2, $a^3, \ldots, a^n$ puis les n produits $p_i\,a^i$ et enfin faire la somme. Cela nous coutera donc $(n-1)+n = 2\,n-1$ multiplications dans $\mathbb{K}$ et n additions dans $\mathbb{K}$ soit $3\,n-1$ opérations dans $\mathbb{K}$.
Le schéma de Horner *produira le même résultat en utilisant la formule suivante :*

$$P(a) = p_0 + a\,(p_1 + a\,(p_2 + a\,(\cdots + a\,(p_{n-1} + a\,p_n)\cdots))).$$

En nous basant sur cette formule, nous pouvons donc calculer $P(a)$ en posant $u = p_n$ puis en calculant, pour $i = 1, \ldots, n$, $u = a\,u + p_{n-i}$. Cela nous coutera donc n additions et n multiplications soit $2\,n$ opérations dans $\mathbb{K}$.
Ainsi, nous voyons que l'algorithme basé sur le schéma de Horner semble meilleur que celui

basé sur le calcul naïf. Dans cet exemple particulier, nous obtenons une complexité linéaire en $O(n)$ pour les deux algorithmes mais parfois le nombre d'opérations peut changer significativement en passant d'une méthode à l'autre (Cf le calcul du déterminant d'une matrice carrée à coefficients dans un corps effectif). Notons aussi qu'ici l'algorithme naïf nécessite l'utilisation de n cases mémoire alors que celui de Horner n'en a besoin que d'une seule.

Dans la suite du cours, nous nous restreindrons à évaluer la *complexité arithmétique* de nos algorithmes c'est-à-dire le nombre total d'opérations arithmétiques dans la structure algébrique effective sous-jacente $\mathbb{A}$. En pratique, cette mesure reflète le coût réel de l'algorithme lorsque le coup des opérations dans $\mathbb{A}$ est prépondérant et que chaque opération dans $\mathbb{A}$ a un coût constant. Ceci est le cas par exemple lorsque $\mathbb{A} = \mathbb{F}_p$ est un corps fini mais pas lorsque $\mathbb{A} = \mathbb{Z}$ ou $\mathbb{Q}$ où le coût d'une opération dépend de la taille des entiers qui entrent en jeu. Dans le cas d'algorithmes manipulant des objects sur $\mathbb{Z}$, la complexité arithmétique n'est pas toujours pertinente car il faut prendre en compte la taille des entiers que l'on manipule qui peut grossir considérablement au cours des calculs. Un meilleur moyen d'évaluer la performance *a priori* d'un algorithme est alors d'évaluer sa *complexité binaire* à savoir le nombre d'opérations binaires effectuées par l'algorithme.

1.4 Principes algorithmiques généraux

Dans cette section, nous allons donner brièvement trois principes algorithmiques qui seront utilisés par la suite pour développer des algorithmes efficaces/rapides en calcul formel. Cette liste n'est évidemment pas exhaustive et il existe bien d'autres principes très utiles pour certains problèmes comme par exemple la récursivité, la technique pas de bébés/pas de géants, ...

1.4.1 Évaluation-Interpolation

Le principe d'évaluation-interpolation est utilisé pour développer des algorithmes manipulant des polynômes. Soit une opération **op** à effectuer sur deux polynômes P et Q à coefficients dans un anneau effectif $\mathbb{A}$. Le principe de l'évaluation-interpolation pour calculer $R = P$ **op** Q est alors le suivant :

1. On choisit un certain nombre de points d'évaluation $a_1, \ldots, a_n \in \mathbb{A}$;
2. On évalue les polynômes P et Q en $a_1, \ldots, a_n$ (évaluation) ;
3. On en déduit l'évaluation de R en $a_1, \ldots, a_n$;
4. À partir des valeurs $R(a_1), \ldots, R(a_n)$, on reconstruit R (interpolation).

L'efficacité de la stratégie repose alors sur le choix des points $a_1, \ldots, a_n$ dans l'anneau $\mathbb{A}$. Le nombre n de points d'évaluation doit être suffisamment grand pour garantir la faisabilité de l'étape 4. De plus les points eux mêmes doivent être choisis pour optimiser le coût des étapes 2 et 4. Par exemple, choisir comme points d'évaluation des racines de l'unité, où des points en progression arithmétique peut s'avérer particulièrement intéressant.

1.4.2 Méthodes modulaires

En calcul formel, beaucoup d'algorithmes manipulant des polynômes, des fractions rationnelles ou des matrices à coefficients entiers souffrent de la croissance des expressions intermédiaires. En d'autres termes, les entiers apparaissants au cours des calculs sont de très grande taille par rapport à ceux qui figurent à la fois dans l'entrée et dans la sortie de l'algorithme. Pour illustrer ce propos, considérons le problème du calcul du PGCD par l'algorithme d'Euclide (voir l'algorithme 7 au chapitre 3).

Exemple 1.4.1. *Considérons les polynômes* $P = 7\,X^5 - 22\,X^4 + 55\,X^3 + 94\,X^2 - 87\,X + 56$ *et* $Q = 62\,X^4 - 97\,X^3 + 73\,X^2 + 4\,X + 83$. *Partant de* $R_0 = P$, $R_1 = Q$, *l'algorithme d'Euclide produit alors la suite de restes suivante :*

$$R_2 = \mathrm{rem}(R_0, R_1) = \tfrac{113293}{3844}\,X^3 + \tfrac{409605}{3844}X^2 - \tfrac{183855}{1922}\,X + \tfrac{272119}{3844},$$

$$R_3 = \mathrm{rem}(R_1, R_2) = \tfrac{18423282923092}{12835303849}\,X^2 - \tfrac{15239170790368}{12835303849}\,X + \tfrac{10966361258256}{12835303849},$$

$$R_4 = \mathrm{rem}(R_2, R_3) = -\tfrac{216132274653792395448637}{44148979404824831944178}\,X - \tfrac{631179956389122192280133}{882979588096496638888356},$$

$$R_5 = \mathrm{rem}(R_3, R_4) = \tfrac{20556791167692068695002336923491296504125}{3639427682941980248860941972667354081}.$$

On voit donc que les coefficients dans les calculs intermédiaires font intervenir des entiers qui croissent de manière exponentielle alors que le résultat est $\mathrm{PGCD}(P, Q) = 1$.

Les méthodes modulaires permettent de remédier à ce problème. Au lieu d'effectuer les calculs sur $\mathbb{Z}$, nous effectuons les calculs dans $\mathbb{F}_p = \mathbb{Z}/(p\,\mathbb{Z})$ pour un (ou plusieurs) nombre(s) premier(s) p. Deux cas sont alors à différencier :

1. Si notre algorithme consiste à décider si une propriété est vraie ou fausse, ou à calculer un degré ou une dimension, alors l'algorithme appliqué aux entrées réduites modulo p donne un algorithme *probabiliste* répondant à la question. De plus, si on peut maitriser les « mauvais » nombres premiers pour lesquels la réponse est fausse, alors on obtient un algorithme *déterministe* ;

2. Lorsqu'une borne sur la taille des entiers du résultat est connue, on peut alors utiliser une stratégie basée sur le théorème des restes chinois (voir Chapitre 3) qui implique qu'un entier inférieur à un produit de nombres premiers $p_1 \cdots p_k$ peut être reconstruit à partir de ses réductions modulo $p_1, \ldots, p_k$. On effectue alors le calcul modulo suffisamment de nombres premiers et on reconstruit le résultat. Cette stratégie est alors du même type que l'évaluation-interpolation.

1.4.3 Diviser pour régner

Un autre principe important utilisé (peut-être même le plus important) pour la construction d'algorithmes efficaces en calcul formel est le paradigme « diviser pour régner ». Il consiste

à résoudre un problème en le réduisant à m instances du même problème sur des entrées de taille divisée par p (en général $p = 2$) puis à reconstruire le résultat. Pour estimer le coût total $C(n)$ d'un tel algorithme, on doit connaitre le coût $T(n)$ de la recombinaison et du découpage initial ainsi que le coût κ de l'algorithme utilisé lorsque la taille $s \geq p$ de l'entrée est suffisamment petite. Le coût total obéit alors souvent à une relation de la forme suivante :

$$C(n) \leq \begin{cases} T(n) + m\,C(\lceil n/p \rceil), & \text{si } n \geq s, \\ \kappa & \text{sinon.} \end{cases} \tag{1.1}$$

L'efficacité de la technique dépend alors fortement de la fonction T et on a le résultat (admis) suivant qui sera suffisant dans le cadre ce cours.

Théorème 1.4.1. *Soit $C(n)$ satisfaisant l'inégalité (1.1) avec $m > 0$, $\kappa > 0$ et où T est une fonction telle qu'il existe q et r avec $1 < q \leq r$ vérifiant $q\,T(n) \leq T(p\,n) \leq r\,T(n)$ pour n assez grand*[1]*. Alors lorsque $n \to \infty$, on a :*

$$C(n) = \begin{cases} O(T(n)), & \text{si } q > m, \\ O(T(n)\log(n)), & \text{si } q = m, \\ O\left(n^{\log_p(m)}\,\frac{T(n)}{n^{\log_p(q)}}\right), & \text{si } q < m. \end{cases} \tag{1.2}$$

Exemple 1.4.2. *Au chapitre 2, nous verrons que le coût $K(n)$ de l'algorithme de Karatsuba pour le calcul du produit de deux polynômes de degré au plus $n-1$ vérifie :*

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

Avec les notations précédentes, on a donc $m = 3$, $p = s = 2$, $\kappa = 1$ et $T(n) = 4\,n$ vérifie $q\,T(n) \leq T(p\,n) \leq r\,T(n)$ avec $q = r = p = 2$. On a donc $q < m$ de sorte que (1.2) implique

$$K(n) = O\left(n^{\log_2(3)}\,\frac{4\,n}{n^{\log_2(2)}}\right) = O(n^{\log_2(3)}).$$

[1] Notons que cette condition est vérifiée lorsque $T(n) = n^\alpha\,\log^\beta(n)$ avec $\alpha > 0$.

Chapitre 2

Algorithmique rapide pour les opérations de base en calcul formel

Dans ce chapitre, nous allons nous intéresser à trois opérations de base en calcul formel à savoir la multiplication de polynômes, la division des séries formelles (et la division euclidienne des polynômes) et la multiplication de matrices. Pour chaque problème, nous allons donner des algorithmes rapides permettant d'effectuer l'opération en question. Notons qu'il est d'autant plus important d'avoir des algorithmes rapides pour ces opérations que la complexité de beaucoup d'autres algorithmes manipulant les mêmes objects peut s'exprimer en fonction de la complexité de l'algorithme utilisé pour effectuer ces trois opérations.

2.1 Multiplication de polynômes

Soient P et Q deux polynômes univariés de degré au plus n et à coefficients dans un anneau effectif $\mathbb{A}$. Le but de cette section est de donner des algorithmes rapides pour calculer le produit $P \times Q = P\,Q$. En d'autres termes, nous allons étudier, en fonction n, en combien d'opérations dans $\mathbb{A}$ nous pouvons effectuer le produit $P\,Q$. Notons que l'on obtient des résultats très semblables pour la complexité binaire du produit d'entiers. En effet, intuitivement nous pouvons décomposer un entier sur une base B (e.g., $B = 10,\, 2, \ldots$) et obtenir « en gros » les mêmes algorithmes modulo la gestion des retenues.

2.1.1 Algorithme naïf

Soient à multiplier les polynômes

$$P = \sum_{i=0}^{n} p_i\, X^i = p_0 + p_1\, X + \cdots + p_n\, X^n, \quad Q = \sum_{i=0}^{n} q_i\, X^i = q_0 + q_1\, X + \cdots + q_n\, X^n,$$

avec $p_0, \ldots, p_n, q_0, \ldots, q_n \in \mathbb{A}$. Le produit $P\,Q$ s'écrit alors

$$P\,Q = \sum_{i=0}^{2\,n} h_i\, X^i, \quad \forall\, i = 0, \ldots, 2\,n,\ h_i = \sum_{j+k=i} p_j\, q_k. \tag{2.1}$$

Proposition 2.1.1. *En utilisant la formule (2.1), le produit $P\,Q$ de deux polynômes P et Q de degré au plus n peut s'effectuer en utilisant n^2 additions et $(n+1)^2$ multiplications dans $\mathbb{A}$. La complexité arithmétique de l'algorithme naïf est donc $O(n^2)$ opérations dans $\mathbb{A}$: c'est un algorithme quadratique.*

Démonstration. Pour $i = 0, \ldots, n$, calculer h_i nécessite i additions et $(i+1)$ multiplications. De même, pour $i = 0, \ldots, n-1$, calculer $h_{2\,n-i}$ nécessite i additions et $(i+1)$ multiplications. En conséquence le nombre total d'additions est égal à $2\sum_{i=0}^{n-1} i + n = n^2$ et le nombre total de multiplications est égal à $2\sum_{i=0}^{n-1}(i+1) + (n+1) = (n+1)^2$. □

2.1.2 Algorithme de Karatsuba

L'algorithme pour la multiplication de polynômes dû à Karatsuba repose sur la remarque suivante dans le cas de polynômes $P = p_0 + p_1\,X$ et $Q = q_0 + q_1\,X$ de degré 1. Le produit $P\,Q$ s'écrit alors

$$P\,Q = p_0\,q_0 + (p_0\,q_1 + p_1\,q_0)\,X + p_1\,q_1\,X^2,$$

de sorte que l'algorithme naïf effectuera les 4 produits $p_0\,q_0$, $p_0\,q_1$, $p_1\,q_0$ et $p_1\,q_1$ et 1 addition. En revanche, en remarquant que

$$(p_0\,q_1 + p_1\,q_0) = (p_0 + p_1)\,(q_0 + q_1) - p_0\,q_0 - p_1\,q_1,$$

on voit que l'on peut calculer $P\,Q$ en effectuant seulement des 3 produits $p_0\,q_0$, $p_1\,q_1$ et $(p_0+p_1)\,(q_0+q_1)$ et 4 additions. Nous perdons certes 3 additions par rapport à l'algorithme naïf mais le gain d'une multiplication va au final entrainer une amélioration de la complexité.

Pour deux polynômes quelconques, le principe de l'algorithme de Karatsuba consiste à utiliser l'observation précédente de manière récursive en utilisant une stratégie diviser pour régner (voir la sous-section 1.4.3). Soient à multiplier deux polynômes P et Q de degré au plus $n-1$ avec $n = 2^k$. On écrit alors

$$P = P_0 + P_1\,X^{\frac{n}{2}}, \quad Q = Q_0 + Q_1\,X^{\frac{n}{2}}, \tag{2.2}$$

où P_0, P_1, Q_0 et Q_1 sont des polynômes de degré au plus $n/2-1 = 2^{k-1}-1$. On applique alors la stratégie expliquée ci-dessus pour les polynômes de degré 1 et on aura 3 appels récursifs sur des polynômes de degré au plus $n/2-1$ pour calculer $P_0\,Q_0$, $P_1\,Q_1$ et $(P_0+P_1)\,(Q_0+Q_1)$. On obtient l'algorithme 1.

Proposition 2.1.2. *L'algorithme 1 de Karatsuba calcule le produit $P\,Q$ de deux polynômes de degré au plus $n-1$ avec $n = 2^k$ en $O(n^{1.59})$ opérations dans $\mathbb{A}$.*

Démonstration. Soit $K(n)$ le nombre d'opérations arithmétiques effectuées par l'algorithme de Karatsuba. Un appel sur des polynômes de degré au plus $n-1$ fait 3 appels récursifs sur des polynômes de degré au plus $n/2-1$ ainsi que 2 additions de polynômes de degré au plus $n/2-1$ et 2 additions de polynômes de degré au plus $n-1$. Finalement, la dernière étape peut

Algorithme 1 *Algorithme de Karatsuba*

Entrées : $P, Q \in \mathbb{A}[X]$ de degré au plus $n-1$ avec $n = 2^k$.
Sorties : Le produit $P\,Q$.
  Si $n = 1$ **Alors**
    - Retourner $P\,Q$;
  Sinon
    - Décomposer P et Q sous la forme (2.2) ;
    - Calculer $H_0 = P_0\,Q_0$ et $H_2 = P_1\,Q_1$ en utilisant deux appels récursifs ;
    - Calculer $P_0 + P_1$ et $Q_0 + Q_1$;
    - Calculer $G = (P_0 + P_1)\,(Q_0 + Q_1)$ en utilisant un appel récursif ;
    - Calculer $H_1 = G - H_0 - H_2$;
    - Retourner $H_0 + H_1\,X^{\frac{n}{2}} + H_2\,X^n$.
  Finsi

se faire à l'aide d'une seule addition de polynômes de degré au plus $n-1$ (car H_0 et $H_2\,X^n$ sont à supports monomiaux disjoints et peuvent donc être additionnés gratuitement). On a donc $K(n) \leq 3\,K(n/2) + 4\,n$ avec $K(1) = 1$ (coût de la multiplication de deux polynômes de degré 0) d'où le résultat d'après l'exemple 1.4.2 ($\log_2(3) \approx 1.584962501$). □

Remarque 2.1.1. *Lorsque $n-1$ n'est pas de la forme 2^k, on complète les polynômes avec des coefficients nuls pour se ramener au cas $n-1 = 2^k$. Puisque le premier entier plus grand que $n-1$ de la forme 2^k est au plus égal à $2\,(n-2)$, la complexité reste en $O(n^{1.59})$ opérations dans $\mathbb{A}$.*

2.1.3 Transformée de Fourier rapide (FFT)

L'algorithme basé sur la transformée de Fourier rapide est à ce jour le meilleur algorithme connu pour la multiplication de polynômes univariés. Cet algorithme est basé sur une stratégie évaluation-interpolation (voir la sous-section 1.4.1).

Étant donné un polynôme $P = p_0 + p_1\,X + \cdots + p_{n-1}\,X^{n-1}$ de degré au plus $n-1$ avec $n = 2^k$, on pose

$$P_{\text{pair}} = p_0 + p_2\,X + p_4\,X^2 + \cdots + p_{n-2}\,X^{\frac{n}{2}-1}, \quad P_{\text{impair}} = p_1 + p_3\,X + p_5\,X^2 + \cdots + p_{n-1}\,X^{\frac{n}{2}-1},$$

de sorte que

$$P(X) = P_{\text{pair}}(X^2) + X\,P_{\text{impair}}(X^2). \tag{2.3}$$

Évaluer P en n points $a_0, \ldots, a_{n-1}$ revient donc à évaluer P_{pair} et P_{impair} en $a_0^2, \ldots, a_{n-1}^2$.

L'idée consiste maintenant à utiliser comme points d'évaluation des racines de l'unité.

Définition 2.1.1. *Un élément $\omega \in \mathbb{A}$ est une* racine n^{e} de l'unité *si $\omega^n = 1$. C'est une racine* primitive n^{e} de l'unité *si de plus $\omega^m \neq 1$ pour tout $m < n$.*

Plaçons nous tout d'abord sur $\mathbb{C}$ où nous disposons de la racine primitive n^{e} de l'unité donnée par $\omega = \exp\left(\frac{2\,i\,\pi}{n}\right)$. La suite des carrés $1, \omega^2, (\omega^2)^2, \ldots, (\omega^2)^{n-1}$ correspond alors à $1, \omega^2, \omega^4, \ldots, \omega^{n-2}, 1, \omega^2, \omega^4, \ldots, \omega^{n-2}$. Évaluer P_{pair} (respectivement P_{impair}) en les n points $1, \omega^2, \ldots, \omega^{2\,n-2}$ peut donc se faire en évaluant P_{pair} (resp. P_{impair}) en les $n/2$ points $1, \omega^2, \ldots, \omega^{n-2}$.

On obtient alors l'algorithme 2 ci-dessous utilisant une stratégie diviser pour régner pour évaluer P en $1, \omega, \ldots, \omega^{n-1}$ ce qu'on appelle aussi calculer la *transformée de Fourier discrete (DFT) de P*.

Algorithme 2 *Transformée de Fourier Rapide (FFT)*

Entrées : $P = \sum_{i=0}^{n-1} p_i\, X^i \in \mathbb{C}[X]$ avec $n = 2^k$ et $\omega = \exp\left(\frac{2\,i\,\pi}{n}\right)$.
Sorties : $P(1), \ldots, P(\omega^{n-1})$.
Si $n = 1$ **Alors**
- Retourner p_0 ;

Sinon
- Décomposer P sous la forme (2.3) ;
- Évaluer P_{pair} et P_{impair} en $1, \omega^2, \ldots, \omega^{n-2}$ à l'aide de deux appels récursifs ;
- Retourner les valeurs de $P(1), \ldots, P(\omega^{n-1})$ obtenus à partir de (2.3).

Finsi

Proposition 2.1.3. *Soit $\omega = \exp\left(\frac{2\,i\,\pi}{n}\right)$ une racine primitive n^{e} de l'unité. L'algorithme 2 ci-dessus, appelé, transformée de Fourier rapide (FFT) calcule la transformée de Fourier discrete (DFT) de $P \in \mathbb{C}[X]$ de degré au plus $n-1$ avec $n = 2^k$ en $O(n \log(n))$ opérations dans $\mathbb{C}$.*

Démonstration. Soit $F(n)$ le nombre d'opérations arithmétiques effectuées par l'algorithme FFT. L'algorithme utilise deux appels récursifs sur des polynômes de degré au plus $n/2 - 1$ à évaluer en $n/2$ points plus un nombre d'opérations proportionnel à n à la dernière étape. On a donc $F(n) \leq 2\,F(n/2) + C\,n$, où C est une constante et $F(1) = 0$. Le résultat est donc une application directe du théorème 1.4.1 avec $p = m = q = r = 2$ et $\kappa = 0$. □

On peut montrer (voir TD) que l'opération inverse, i.e., l'interpolation qui consiste, étant données les valeurs de P en $1, \omega, \ldots, \omega^{n-1}$ à retrouver les coefficients du polynôme P se ramène à calculer une transformée de Fourier discrete sur les points $1, \omega^{-1}, \ldots, (\omega^{-1})^{n-1}$.

On peut donc donner l'algorithme de multiplication de polynômes basé sur la FFT.

Proposition 2.1.4. *L'algorithme 3 de multiplication par FFT ci-dessus calcule le produit de deux polynômes de degré au plus $n-1$ avec $n = 2^k$ en $O(n \log(n))$ opérations dans $\mathbb{C}$.*

Démonstration. Les étapes d'évaluation et d'interpolation se font à l'aide de 3 FFT. Les autres étapes ont un coût en $O(n)$ opérations dans $\mathbb{C}$. Le résultat découle donc de la proposition 2.1.3. □

Algorithme 3 *Multiplication de deux polynômes par FFT*

Entrées : $P, Q \in \mathbb{C}[X]$ de degré au plus $\frac{n}{2} - 1$ avec $n = 2^k$ et $\omega = \exp\left(\frac{2\,i\,\pi}{n}\right)$.
Sorties : Le produit $P\,Q$.
- Pré-calculer les puissances $\omega^2, \ldots, \omega^{n-1}$;
- `Évaluation` : utiliser la FFT pour évaluer P et Q en 1, $\omega, \ldots,$ ω^{n-1} ;
- Calculer les produits $P(1)\,Q(1)$, $P(\omega)\,Q(\omega), \ldots, P(\omega^{n-1})\,Q(\omega^{n-1})$;
- `Interpolation` : utiliser la FFT pour calculer $P\,Q$.

Les deux algorithmes donnés dans cette sous-section reposent sur la notion de racine de l'unité. Lorsque nous sommes dans $\mathbb{C}$, nous disposons toujours de $\omega = \exp\left(\frac{2\,i\,\pi}{n}\right)$ comme racine primitive n^{e} de l'unité. Si l'anneau $\mathbb{A}$ des coefficients de nos polynômes n'est pas $\mathbb{C}$, alors deux cas sont à distinguer : si $\mathbb{A}$ vérifie qu'il existe des racines primitives n^{e} de l'unité pour tout $n = 2^k$, alors on peut encore multiplier deux polynômes de degré au plus $n - 1$ en $O(n \log(n))$. Par contre, si ce n'est pas le cas, nous devons construire ces racines ce qui entraine un surcoût et la multiplication utilise alors $O(n \log(n) \log(\log(n)))$ opérations.

Résumé : nous avons donc vu dans cette section que le produit de deux polynômes de degré au plus n et à coefficients dans $\mathbb{A}$ peut se calculer en :

1. $O(n^2)$ opérations dans $\mathbb{A}$ par l'algorithme naïf ;
2. $O(n^{1.59})$ opérations dans $\mathbb{A}$ par l'algorithme de Karatsuba ;
3. $O(n \log(n) \log(\log(n)))$ opérations dans $\mathbb{A}$ en utilisant la FFT.

En pratique, il faut faire attention aux constantes qui se cachent dans la notation $O(.)$ car elles peuvent être déterminantes pour l'efficacité pratique de l'algorithme. Dans les meilleures implémentations actuelles, l'algorithme de Karatsuba l'emporte sur l'algorithme naïf pour des degrés de l'ordre de 20 et l'algorithme basé sur la FFT gagne pour des degrés de l'ordre de 100. Notons que pour certains problèmes (e.g., en cryptographie) nous devons manipuler des polynômes de degrés de l'ordre de 100000.

2.2 Inversion de séries formelles et division euclidienne

2.2.1 L'anneau des séries formelles

Définition 2.2.1. *Soit $\mathbb{A}$ un anneau. On note $\mathbb{A}[[X]]$ l'ensemble des séries formelles de la forme*

$$F = f_0 + f_1\,X + f_2\,X^2 + \cdots = \sum_{i \geq 0} f_i\,X^i, \quad f_i \in \mathbb{A}.$$

Le terme f_0 s'appelle le terme constant *de la série F.*

Une série formelle peut s'identifier avec la suite $(f_i)_{i\in\mathbb{N}}$ de ses coefficients qui est donc une suite d'éléments de $\mathbb{A}$. L'ensemble $\mathbb{A}[[X]]$ est muni d'une addition et d'une multiplication données par :

$$\sum_{i\geq 0} f_i\, X^i + \sum_{i\geq 0} g_i\, X^i = \sum_{i\geq 0} (f_i + g_i)\, X^i,$$

$$\left(\sum_{i\geq 0} f_i\, X^i\right)\, \left(\sum_{i\geq 0} g_i\, X^i\right) = \sum_{i\geq 0} h_i\, X^i, \quad h_i = \sum_{j+k=i} f_j\, g_k.$$

L'élément neutre pour le produit est alors la série formelle $1 + \sum_{i>0} 0\, X^i$ où $1 = 1_{\mathbb{A}}$ est l'élément neutre de $\mathbb{A}$ pour la multiplication. Si on calcule le produit de $F = 1+X+\sum_{i>1} 0\, X^i$ par $G = 1 - X + X^2 - X^3 + \sum_{i>3} (-1)^i\, X^i$, alors on trouve $F\, G = 1 + \sum_{i>0} 0\, X^i$. La série G est alors l'*inverse* de F pour la multiplication et on note $G = F^{-1}$. D'une manière générale, nous avons le résultat suivant :

Proposition 2.2.1. *L'ensemble $\mathbb{A}[[X]]$ des séries formelles est un anneau qui est commutatif si $\mathbb{A}$ est commutatif. Les éléments inversibles de $\mathbb{A}[[X]]$ sont les séries formelles de terme constant inversible dans $\mathbb{A}$.*

Démonstration. La première partie de la proposition est claire. Si F une série formelle de la forme $F = 1 + X\, G$, où G est une série formelle, alors on a $F^{-1} = 1 - X\, G + X^2\, G^2 - \cdots$. Maintenant si le terme constant de F est un élément inversible $a \in \mathbb{A}$, on écrit $F = a\,(1+X\, G)$ avec $G = a^{-1}\,(F - a)/X$ et on a donc $F^{-1} = (1 + X\, G)^{-1}\, a^{-1}$. On a donc montré que toutes les séries formelles dont le terme constant est inversible dans $\mathbb{A}$ sont inversibles dans $\mathbb{A}[[X]]$. Inversement si F est inversible dans $\mathbb{A}[[X]]$, si G désigne son inverse et si f_0 (resp. g_0) désigne le terme constant de F (resp. G), alors on a $f_0\, g_0 = 1$ dans $\mathbb{A}$ de sorte que f_0 est inversible dans $\mathbb{A}$. □

2.2.2 Calcul formel dans l'anneau des séries formelles

Du point de vue du calcul formel se pose alors la question suivante : comment faire des calculs dans $\mathbb{A}[[X]]$ lorsque $\mathbb{A}$ est un anneau effectif ? En pratique, on ne peut bien évidemment pas manipuler des objets « infini » mais on doit considérer des *troncatures* de ces objets c'est-à-dire, dans le cas des séries formelles, un certain nombre des premiers termes des séries. Dans la suite de cette section, *se donner une série formelle* (resp. *calculer une série formelle*) F signifiera donc se donner (resp. calculer) les N premiers termes de la série. On notera $F = P + O(X^N)$ où P est donc un polynôme de degré au plus $N - 1$. Les séries formelles deviennent alors des polynômes de sorte que les algorithmes vus dans la section précédente peuvent être utilisés pour calculer efficacement le produit de deux séries formelles.

On va maintenant s'intéresser au problème suivant : étant donnée $F \in \mathbb{A}[[X]]$ de terme constant inversible dans un anneau effectif $\mathbb{A}$, comment calculer efficacement son inverse ? La méthode standard pour faire cette opération consiste à utiliser la *division selon les puissances croissantes.*

Définition 2.2.2. *Soit $P, Q \in \mathbb{A}[X]$ tels que $Q(0) \neq 0$. Alors pour tout entier $n \in \mathbb{N}$, il existe un unique couple de polynômes $(Q_n, R_n) \in \mathbb{A}[X]^2$ avec $\deg(Q_n) \leq n$ tel que*

$$P = Q_n\, Q + X^{n+1}\, R_n.$$

Le polynôme Q_n (resp. R_n) s'appelle quotient *(resp.* le reste*) à l'ordre n de la division selon les puissances croissantes de P par Q.*

Notons que comme pour une division (euclidienne) classique de polynômes, les quotients et restes successifs dans la division selon les puissances croissantes peuvent s'obtenir en « posant » la division mais, contrairement à la division classique, on ordonne les polynômes selon leurs puissances croissantes et non décroissantes.

Exemple 2.2.1. *Soient $P = X + 1$ et $Q = X^2 + 1$. On a alors*

$$P = (1 + X - X^2)\, Q + X^3\, (-1 + X),$$

de sorte que les quotients et restes respectifs d'ordre 2 dans la division selon les puissances croissantes de P par Q sont donnés par $1 + X - X^2$ et $-1 + X$.

Étant donnée une série formelle $F = P + O(X^N) \in \mathbb{A}[[X]]$ de terme constant égal à 1, on peut effectuer la division selon les puissances croissantes de 1 par P (en ne conservant à chaque étape n que les termes de degré au plus $N - (n + 1)$ dans R_n). L'inverse de F est alors donné par $Q_{N-1} + O(X^N)$ où Q_{N-1} est le quotient à l'ordre $N - 1$ de cette division.

Exemple 2.2.2. *Considérons la série formelle $F = 1+2\,X-3\,X^2+X^3+O(X^4)$. En posant la division selon les puissances croissantes de 1 par $P = 1+2\,X-3\,X^2+X^3$, on trouve que le quotient à l'ordre $4-1=3$ est $Q_3 = 1-2\,X+7\,X^2-21\,X^3$. On peut alors vérifier que $Q_3+O(X^4)$ est bien l'inverse cherché car $(1+2\,X-3\,X^2+X^3)\,(1-2\,X+7\,X^2-21\,X^3) = 1+O(X^4)$.*

Proposition 2.2.2. *Soit $F = P+O(X^N) \in \mathbb{A}[[X]]$ de terme constant égal à 1. Calculer l'inverse $G = Q+O(X^N) \in \mathbb{A}[[X]]$ de F en utilisant la division selon les puissances croissantes nécessite $O(N^2)$ opérations dans $\mathbb{A}$.*

Démonstration. Exercice. □

Nous allons maintenant voir un algorithme plus efficace pour effectuer cette opération en utilisant *l'itération de Newton*. La méthode de Newton (très utilisée notamment en analyse numérique) permet de calculer les zéros de fonctions réelles $\phi : \mathbb{R} \to \mathbb{R}$ de classe $\mathcal{C}^1$ en choisissant un point $x_0 \in \mathbb{R}$ puis en utilisant l'itération de Newton

$$x_{k+1} = \mathcal{N}(x_k) = x_k - \frac{\phi(x_k)}{\phi'(x_k)}, \tag{2.4}$$

pour calculer les points suivants. Sous de bonnes hypothèses, cette suite va converger vers un zéro de ϕ et la convergence sera quadratique, i.e., le nombre de décimales exactes double à chaque itération.

Soit $F = P + O(X^N) \in \mathbb{A}[[X]]$ de terme constant égal à 1. L'idée de l'algorithme consiste à utiliser la méthode de Newton pour résoudre l'équation $\phi(G) = 0$ où $\phi : \mathbb{A}[[X]] \to \mathbb{A}[[X]]$ est donnée par $\phi(G) = 1/G - F$. En effet l'unique zéro de ϕ est alors l'inverse de la série formelle F que l'on cherche. L'itération (2.4) devient alors :

$$G_{k+1} = \mathcal{N}(G_k) = G_k - \frac{\frac{1}{G_k} - P}{-\frac{1}{G_k^2}} = 2\,G_k - G_k^2\,P = G_k\,(2 - P\,G_k). \tag{2.5}$$

Proposition 2.2.3. *Soit $F = P + O(X^N) \in \mathbb{A}[[X]]$ de terme constant égal à 1. Alors la suite définie par $G_0 = 1$ et (2.5) vérifie qu'à chaque itération k, il existe un polynôme $R_k \in \mathbb{A}[X]$ tel que $P\,G_k = 1 + X^{2^k}\,R_k$. Autrement dit, $1/F = G_k + O(X^{2^k})$.*

Démonstration. Raisonnons par récurrence sur k. La propriété est vraie pour $k = 0$ car $F(0) = P(0) = 1$ et $G_0 = 1$. Supposons maintenant $P\,G_k = 1 + X^{2^k}\,R_k$. On a

$$P\,G_{k+1} = 2\,P\,G_k - G_k^2\,P^2 = 2\,(1 + X^{2^k}\,R_k) - (1 + X^{2^k}\,R_k)^2 = 1 + X^{2^{k+1}}\,(-R_k)^2,$$

d'où le résultat. □

Exemple 2.2.3. *Reprenons l'exemple 2.2.2. L'itération de Newton appliquée au polynôme $P = 1 + 2\,X - 3\,X^2 + X^3$ donne*

$$G_0 = 1, \quad G_1 = -X^3 + 3\,X^2 - 2\,X + 1, \quad G_2 = -21\,X^3 + 7\,X^2 - 2\,X + 1 + O(X^4),$$

de sorte que l'on retrouve le résultat obtenu à l'exemple 2.2.2 à l'aide de division selon les puissances croissantes.

On obtient alors l'algorithme suivant pour calculer l'inverse d'une série formelle :

Algorithme 4 *Inversion d'une série formelle par l'itération de Newton*

Entrées : $N = 2^k \in \mathbb{N}^*$ et $F = P + O(X^N) \in \mathbb{A}[[X]]$ de terme constant égal à 1.
Sorties : $G = Q + O(X^N) \in \mathbb{A}[[X]]$ tel que $F\,G = 1 + O(X^N)$.

Si $N = 1$ **Alors**
- Retourner 1 ;

Sinon
- Poser $G_0 = 1$;
- Calculer récursivement $G_1, \ldots, G_k$ (les N premiers termes) à l'aide de (2.5) ;
- Retourner G_k.

Finsi

Proposition 2.2.4. *Soit $\mathbf{M}$ une fonction telle que le produit de deux polynômes de degré au plus n puisse se calculer en $O(\mathbf{M}(n))$ avec $\mathbf{M}(n) \geq 2\,\mathbf{M}(n/2)$[1]. L'algorithme 4 calcule l'inverse d'une série formelle à l'aide de l'opérateur de Newton en $O(\mathbf{M}(N))$ opérations dans $\mathbb{A}$.*

[1]Notons que cette propriété est vérifiée par les algorithmes rapides donnés dans la section 2.1.

Démonstration. Soit $I(N)$ le nombre d'opérations arithmétiques effectuées par l'algorithme. Connaissant G_k, le calcul de G_{k+1} nécessite $2\,\mathbf{M}(N)$ opérations dans $\mathbb{A}$ de sorte que nous avons $I(N) \leq I(N/2) + 2\,\mathbf{M}(N)$. En itérant et en utilisant $\mathbf{M}(n) \geq 2\,\mathbf{M}(n/2)$, on obtient alors

$$I(N) \leq I\left(\frac{N}{2}\right) + 2\,\mathbf{M}(N) \leq 2\sum_{i=0}^{k-1} \mathbf{M}\left(\frac{N}{2^i}\right) \leq 2\,\mathbf{M}(N)\sum_{i=0}^{k-1}\frac{1}{2^i} \leq 4\,\mathbf{M}(N),$$

d'où le résultat. □

Lorsque N n'est pas une puissance de 2 on peut encore compléter avec des 0 (voir la remarque 2.1.1) et obtenir la même estimation de complexité.

2.2.3 Application à la division euclidienne

Intéressons nous maintenant au problème de la division euclidienne de deux polynômes : étant donnés deux polynômes A, $B \in \mathbb{A}[X]$ avec $\deg(A) = n$, $\deg(B) = m$ et B unitaire, nous voulons calculer $(Q, R) \in \mathbb{A}[X]^2$ tel que

$$A = Q\,B + R, \quad \deg(R) < \deg(B).$$

La méthode classique consiste à poser la division. Notons

$$A = a\,X^n + A_0, \quad B = X^m + B_0, \quad \deg(A_0) < n,\ \deg(B_0) < m.$$

Si $n < m$, nous n'avons rien à faire sinon nous effectuons la soustraction

$$A - a\,X^{n-m}\,B = a\,X^n + A_0 - a\,X^{n-m}\,(X^m + B_0) = A_0 - a\,X^{n-m}\,B_0 \tag{2.6}$$

pour éliminer le terme de plus haut degré et obtenir un polynôme de degré strictement inférieur à n. On itère ensuite ce procédé jusqu'à ce que le polynôme obtenu soit de degré strictement inférieur à m.

Proposition 2.2.5. *Soient A, $B \in \mathbb{A}[X]$ de degrés respectifs n et m. Effectuer la division euclidienne de A par B en posant la division nécessite $O(m\,(n-m))$ opérations dans $\mathbb{A}$.*

Démonstration. À chaque itération (2.6), l'algorithme effectue $O(m)$ opérations. Le nombre maximal d'itérations étant égal à $n - m + 1$, on obtient le résultat annoncé. □

Notons que le pire cas est obtenu pour $n = 2\,m$ où l'on a une complexité quadratique en $O(m^2)$ opérations dans $\mathbb{A}$.

Nous allons maintenant donner un algorithme rapide pour la division euclidienne basé sur l'inversion de séries formelles (voir la sous-section précédente 2.2.2). L'idée consiste à ré-écrire l'égalité $A = Q\,B + R$ sous la forme

$$\frac{A}{B} = Q + \frac{R}{B},$$

de sorte que Q peut s'obtenir en calculant le développement asymptotique de A/B à l'infini. Notons que la contrainte $\deg(R) < \deg(B)$ implique que R/B tend vers 0 à l'infini. Pour réduire le problème à un calcul de division dans l'anneau des séries formelles, on ramène l'infini en 0 en faisant le changement de variable $X = 1/T$. Si l'on suppose que A est de degré n et B unitaire de degré $m \leq n$, on obtient alors

$$\frac{T^n A(1/T)}{T^m B(1/T)} = T^{n-m} Q(1/T) + \frac{T^n R(1/T)}{T^m B(1/T)},$$

où on a multiplié par des puissances de T pour avoir des polynômes. Notons que le terme constant de $T^m B(1/T)$ est alors 1 (car B est unitaire) de sorte que nous allons pouvoir l'inverser dans l'anneau $\mathbb{A}[[X]]$. On obtient l'algorithme suivant :

Algorithme 5 *Division euclidienne rapide*

Entrées : A, $B \in \mathbb{A}[X]$ avec $\deg(A) = n$, $\deg(B) = m \leq n$ et B unitaire.
Sorties : Q, $R \in \mathbb{A}[X]$ avec $A = Q\,B + R$ et $\deg(R) < \deg(B)$.

- Calculer $T^n A(1/T)$ et $T^m B(1/T)$;
- Calculer l'inverse de $T^m B(1/T)$ (les $n - m + 1$ premiers termes) ;
- Calculer le produit de $T^n A(1/T)$ par $1/(T^m B(1/T))$ (les $n - m + 1$ premiers termes) ;
- En déduire Q ;
- Calculer $R = A - Q\,B$ (les m premiers termes) ;
- Retourner (Q, R).

Proposition 2.2.6. *Soit A, $B \in \mathbb{A}[X]$ avec $\deg(A) = n$, $\deg(B) = m \leq n$ et B unitaire. Soit $\mathbf{M}$ une fonction telle que le produit de deux polynômes de degré au plus n puisse se calculer en $O(\mathbf{M}(n))$ opérations arithmétiques avec $\mathbf{M}(n) \geq 2\,\mathbf{M}(n/2)$[2]. L'algorithme 5 ci-dessus effectue la division euclidienne de A par B en $O(\mathbf{M}(n))$ opérations dans $\mathbb{A}$.*

Démonstration. Regardons le nombre d'opérations nécessaire à chaque étape de l'algorithme. Les étapes 1 et 4 ne nécessitent aucune opération car elles consistent simplement à inverser l'ordre des coefficients des polynômes. D'après la proposition 2.2.4 la deuxième étape peut s'effectuer en $O(\mathbf{M}(n-m))$ opérations. L'étape 3 nécessite $O(\mathbf{M}(n-m))$ opérations et l'étape 5 requiert $O(\mathbf{M}(m))$ opérations d'où le résultat. □

2.3 Multiplication de matrices

Les algorithmes traitant de problèmes d'*algèbre linéaire* sont omniprésents en calcul formel. La complexité arithmétique de ces algorithmes reposent très souvent sur celle de la multiplication de matrices. Voir le théorème 2.3.1 à la fin de cette sous-section.

[2]Notons que cette propriété est vérifiée par les algorithmes rapides donnés dans la section 2.1.

2.3.1 Algorithme naïf

Soient $A = (a_{i,j})_{1 \leq i,j \leq n}$ et $B = (b_{i,j})_{1 \leq i,j \leq n}$ deux matrices carrées de taille n à coefficients dans un corps effectif $\mathbb{K}$. On note A, $B \in \mathbb{M}_n(\mathbb{K})$. Le produit $C = (c_{i,j})_{1 \leq i,j \leq n}$ des matrices A et B est alors défini par

$$\forall (i,j) \in \{1, \ldots, n\}, \quad c_{i,j} = \sum_{k=1}^{n} a_{i,k}\, b_{k,j}.$$

La multiplication naïve des matrices consiste à utiliser les formules ci-dessus pour calculer chaque $c_{i,j}$.

Proposition 2.3.1. *Soient A, $B \in \mathbb{M}_n(\mathbb{K})$. L'algorithme de multiplication naïve calcule $C = A\,B$ en $O(n^3)$ opérations dans $\mathbb{K}$.*

Démonstration. Le calcul d'un coefficient $c_{i,j}$ nécessite n multiplications et $n-1$ additions dans $\mathbb{K}$. Comme nous avons n^2 coefficients $c_{i,j}$ à calculer, nous obtenons le résultat annoncé. □

2.3.2 Algorithme de Strassen

Nous allons maintenant donner un algorithme plus efficace dû à Strassen qui utilise une technique analogue à celle de l'algorithme de Karatsuba pour la multiplication de polynômes. L'idée consiste donc à voir comment gagner un produit dans $\mathbb{K}$ sur le produit de matrices de taille 2 (quitte à effectuer plus d'additions) pour pouvoir tirer partie de ce gain dans une stratégie diviser pour régner.

Considérons les deux matrices carrées de taille 2 suivantes :

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad B = \begin{pmatrix} x & y \\ z & t \end{pmatrix}, \quad a, b, c, d, x, y, z, t \in \mathbb{K}.$$

Alors les entrées de la matrice $C = A\,B = (c_{i,j})_{1 \leq i,j \leq 2}$ peuvent s'obtenir comme suit

$$\begin{cases} c_{1,1} = a\,x + b\,z = a\,(x+z) + (b-a)\,z, \\ c_{1,2} = a\,y + b\,t = d\,(y+t) + (d-a)\,(z-y) + (b-d)\,(z+t) - (b-a)\,z, \\ c_{2,1} = c\,x + d\,z = a\,(x+z) + (d-a)\,(z-y) + (c-a)\,(x+y) - (c-d)\,y, \\ c_{2,2} = c\,y + d\,t = d\,(y+t) + (c-d)\,y, \end{cases}$$

ce qui ne requiert que 7 multiplications dans $\mathbb{K}$. En utilisant cette observation, on obtient l'algorithme suivant :

Proposition 2.3.2. *L'algorithme 6 de Strassen calcule le produit de deux matrices carrées de taille n en $O(n^{2.81})$ opérations arithmétiques.*

Algorithme 6 *Algorithme de Strassen pour la multiplication de matrices*

Entrées : A, $B \in \mathbb{M}_n(\mathbb{K})$ avec $n = 2^k$.
Sorties : $C \in \mathbb{M}_n(\mathbb{K})$ telle que $C = A\,B$.

Si $n = 1$ **Alors**

- Retourner $A\,B$;

Sinon

- Décomposer A et B par blocs de taille $\frac{n}{2}$:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad B = \begin{pmatrix} x & y \\ z & t \end{pmatrix}, \quad a, b, c, d, x, y, z, t \in \mathbb{M}_{\frac{n}{2}}(\mathbb{K});$$

- À l'aide de 7 appels récursifs, calculer

$$\begin{cases} p_1 = a\,(x + z), \\ p_2 = d\,(y + t), \\ p_3 = (d - a)\,(z - y), \\ p_4 = (b - d)\,(z + t), \\ p_5 = (b - a)\,z, \\ p_6 = (c - a)\,(x + y), \\ p_7 = (c - d)\,y; \end{cases}$$

- Calculer

$$\begin{cases} c_{1,1} = p_1 + p_5, \\ c_{1,2} = p_2 + p_3 + p_4 - p_5, \\ c_{2,1} = p_1 + p_3 + p_6 - p_7, \\ c_{2,2} = p_2 + p_7; \end{cases}$$

- Retourner $C = (c_{i,j})_{1 \leq i,j \leq 2}$.

Finsi

Démonstration. Soit $S(n)$ le nombre d'opérations arithmétiques effectuées par l'algorithme de Strassen. Un appel sur des matrices de taille n fait 7 appels récursifs sur des matrices de taille $n/2$ ainsi qu'un nombre constant $c = 18$ d'additions de matrices de taille $n/2$. On a donc $S(n) \leq 7\,S(n/2) + c\,(n/2)^2$ avec $S(1) = 1$. On applique alors le théorème 1.4.1 avec $m = 7$, $p = 2$, $q = r = 4$ et $\kappa = 1$ ce qui implique que

$$S(n) = O\left(n^{\log_2(7)}\,\frac{c\,(\frac{n}{2})^2}{n^{\log_2(4)}}\right) = O(n^{\log_2(7)}),$$

d'où le résultat ($\log_2(7) \approx 2.807354922$). □

Remarque 2.3.1. *Lorsque la taille n des matrices à multiplier n'est pas une puissance de 2, alors on rajoute artificiellement des lignes et des colonnes aux matrices pour s'y ramener ce qui n'entraine pas de surcoût.*

Nous concluons ce chapitre par le théorème suivant (admis dans le cadre de ce cours) qui montre que la complexité du produit matriciel contrôle celui de toutes les opérations d'algèbre linéaire :

Théorème 2.3.1. *Soit $\mathbb{K}$ un corps et supposons que deux matrices carrées de taille n à coefficients dans $\mathbb{K}$ peuvent être multipliées en $O(\mathbf{MM}(n))$ opérations dans $\mathbb{K}$. Alors, pour une matrice $A \in \mathbb{M}_n(\mathbb{K})$ donnée, il est possible de calculer*

- *le déterminant* $\det(A)$ *de A,*
- *l'inverse A^{-1} de A (si A est inversible),*
- *la solution $x \in \mathbb{K}^n$ du système linéaire $A\,x = b$, pour tout $b \in \mathbb{K}^n$ (si A est inversible),*
- *le polynôme caractéristique de A,*
- *une décomposition LU de A (avec la matrice de permutation P associée si nécessaire),*
- *le rang de A et une forme échelonnée réduite de A,*
- *une base du noyau de A,*

en $\tilde{O}(\mathbf{MM}(n))$ opérations dans $\mathbb{K}$, où la notation $\tilde{O}(.)$ cache des facteurs logarithmiques.

Chapitre 3

Pgcd et résultant

3.1 Rappels : anneaux factoriels et anneaux euclidiens

Nous allons rappeler ici quelques résultats autour de l'existence et l'unicité (à unité près) du PGCD (resp. PPCM) et d'une division euclidienne dans un anneau $\mathbb{A}$ avec en point de mire les cas particuliers $\mathbb{A} = \mathbb{Z}$ et $\mathbb{A} = \mathbb{K}[X]$ qui nous intéresserons ensuite. Certains résultats sont rappelés sans preuve. Il suffit de consulter un bon ouvrage d'algèbre pour retrouver les démonstrations.

Soit $\mathbb{A}$ un anneau commutatif (i.e., $\forall a, b \in \mathbb{A}$, $a\,b = b\,a$), unitaire (i.e., qui admet un élément neutre 1 pour la multiplication) et intègre (i.e., A ne possède pas de diviseur de zéro c'est-à-dire $\forall a, b \in \mathbb{A}$, $a\,b = 0 \Rightarrow (a = 0$ ou $b = 0)$). On note $A^\times$ l'ensemble des *éléments inversibles* (aussi appelés *unités*) de $\mathbb{A}$:

$$\mathbb{A}^\times = \{a \in \mathbb{A} \mid \exists b \in \mathbb{A},\, a\,b = 1\}.$$

Les exemples classiques pour ce cours seront $\mathbb{A} = \mathbb{Z}$, l'anneau des entiers relatifs qui vérifie $\mathbb{Z}^\times = \{-1, 1\}$ et $\mathbb{A} = \mathbb{K}[X]$, l'anneau des polynômes univariés à coefficients dans un corps $\mathbb{K}$ avec $\mathbb{K}[X]^\times = \mathbb{K} \setminus \{0\}$.

3.1.1 Divisibilité, PGCD et PPCM

Définition 3.1.1. *Soient a, $b \in \mathbb{A}$. On dit que b* divise *a ou que a* est divisible par *b et on note $b \mid a$ s'il existe $c \in \mathbb{A}$ avec $a = b\,c$. On dit que a* et *b* sont associés *et on note $a \sim b$ si on a à la fois $b \mid a$ et $a \mid b$.*

On peut montrer que $\sim$ est une relation d'équivalence et que $a \sim b \Leftrightarrow \exists u \in \mathbb{A}^\times,\ a = u\,b$. Si on note $(a) = \{a\,x \mid x \in \mathbb{A}\}$ l'idéal principal engendré par $a \in \mathbb{A}$, alors on a aussi $a \sim b \Leftrightarrow (a) = (b)$.

Définition 3.1.2. *Soit $a \in \mathbb{A}$. On dit que a* est irréductible *si $a \notin \mathbb{A}^\times$ et si $a = b\,c$ avec b, $c \in \mathbb{A}$ implique $b \in \mathbb{A}^\times$ ou $c \in \mathbb{A}^\times$.*

Si l'idéal (a) est un idéal premier (i.e., si $\mathbb{A}/(a)$ est intègre), alors a est irréductible mais la réciproque est fausse (Exercice : dans $\mathbb{A} = \mathbb{Z}[i\sqrt{5}]$, l'idéal principal $(1 + i\sqrt{5})$ n'est pas premier mais $1 + i\sqrt{5}$ est irréductible). Dans $\mathbb{A} = \mathbb{Z}$, les éléments irréductibles sont les entiers de la forme $\pm p$, où p est un nombre premier, dans $\mathbb{A} = \mathbb{K}[X]$ l'ensemble des éléments irréductibles dépend du corps $\mathbb{K}$. Si $\mathbb{K} = \mathbb{C}$, alors les éléments irréductibles sont les polynômes de degré 1 mais si $\mathbb{K} = \mathbb{R}$ alors il faut ajouter les polynômes de degré 2 de discriminant strictement négatif.

Définition 3.1.3. *Soient a, b, $d \in \mathbb{A}$. On dit que d* est un **p**lus **g**rand **c**ommun **d**iviseur (PGCD) de a et b *et on note $d = \text{PGCD}(a, b)$ si*

- *$d \mid a$ et $d \mid b$,*
- *si $c \in \mathbb{A}$ vérifie aussi $c \mid a$ et $c \mid b$, alors $c \mid d$.*

Un PGCD n'existe pas toujours dans un anneau commutatif unitaire et intègre (Exercice : dans $\mathbb{A} = \mathbb{Z}[i\sqrt{5}]$, les éléments $a = 6$ et $b = 2(1 + i\sqrt{5})$ n'ont pas de PGCD) mais s'il existe alors il est unique modulo la relation d'équivalence $\sim$, i.e., si d_1 et d_2 sont deux PGCD de a et b, alors il existe $u \in \mathbb{A}^\times$ tel que $d_1 = u\, d_2$. Deux éléments a et b d'un anneau $\mathbb{A}$ sont dits *premiers entre eux* (ou *étrangers*) si $\text{PGCD}(a, b) = 1$.

Définition 3.1.4. *Soient a, b, $m \in \mathbb{A}$. On dit que m* est un **p**lus **p**etit **c**ommun **m**ultiple (PPCM) de a et b *et on note $m = \text{PPCM}(a, b)$ si*

- *$a \mid m$ et $b \mid m$,*
- *si $c \in \mathbb{A}$ vérifie aussi $a \mid c$ et $b \mid c$, alors $m \mid c$.*

Tout comme le PGCD, si le PPCM existe, alors il est unique.

Pour avoir l'existence du PGCD, nous devons nous placer dans le cas d'un anneau *factoriel*.

Définition 3.1.5. *Soit $\mathbb{A}$ un anneau commutatif unitaire et intègre. L'anneau $\mathbb{A}$ est dit* factoriel *si :*

- *tout élément non-nul est soit inversible soit produit d'éléments irréductibles,*
- *Si $p_1 \cdots p_n = q_1 \cdots q_m$ avec les p_i et les q_i irréductibles dans $\mathbb{A}$, alors $n = m$ et il existe une permutation σ de $\{1, \ldots, n\}$ telle que, pour tout $i \in \{1, \ldots, n\}$, $p_i \sim q_{\sigma(i)}$, i.e., il existe $u_i \in \mathbb{A}^\times$ avec $p_i = u_i\, q_{\sigma(i)}$.*

Les anneaux $\mathbb{Z}$ et $\mathbb{K}[X]$ sont factoriels mais pas $\mathbb{Z}[i\sqrt{5}]$.

Soit $\mathbb{A}$ un anneau factoriel, un *système représentatif d'éléments irréductibles de* $\mathbb{A}$ est une partie $\mathcal{P}$ de $\mathbb{A}$ formée d'éléments irréductibles tels que :

- pour tout $q \in \mathbb{A}$ irréductible, il existe $p \in \mathcal{P}$ tel que $q \sim p$,

- les éléments de $\mathcal{P}$ sont deux à deux non associés.

Par exemple, pour $\mathbb{A} = \mathbb{Z}$, on peut peut prendre pour $\mathcal{P}$ l'ensemble des nombres premiers et pour $\mathbb{A} = \mathbb{K}[X]$, on peut prendre l'ensemble des polynômes irréductibles unitaires. Une fois un système représentatif d'éléments irréductibles $\mathcal{P}$ de $\mathbb{A}$ fixé, tout élément de $a \in \mathbb{A}$ s'écrit de manière unique

$$a = u \prod_{i=1}^{n} p_i^{e_i}, \quad p_1, \ldots, p_n \in \mathcal{P} \text{ avec } \forall i \neq j,\ p_i \neq p_j, \quad e_1, \ldots, e_n \in \mathbb{N}, \quad u \in \mathbb{A}^\times. \tag{3.1}$$

Par conséquent, dans un anneau factoriel, si un élément irréductible p divise un produit $a\,b$, alors $p \,|\, a$ ou $p \,|\, b$ de sorte que p irréductible implique que l'idéal (p) est premier. Dans un anneau factoriel, nous avons donc l'équivalence : a irréductible $\Leftrightarrow$ (a) premier.

Théorème 3.1.1. *Soit $\mathbb{A}$ un anneau factoriel. Alors deux éléments non nuls a et b de $\mathbb{A}$ admettent un* PGCD *unique à multiplication par unité près.*

Démonstration. Avec les notations de (3.1), écrivons $a = u \prod_{i=1}^{n} p_i^{e_i}$ et $b = v \prod_{i=1}^{n} p_i^{f_i}$. Montrons que $d = \prod_{i=1}^{n} p_i^{g_i}$ avec, pour $i = 1, \ldots, n$, $g_i = \min(e_i, f_i)$ est un PGCD de a et b. En effet, nous avons clairement $d \,|\, a$ et $d \,|\, b$. De plus si $c \,|\, a$ et $c \,|\, b$, alors nécessairement $c = w \prod_{i=1}^{n} p_i^{h_i}$ avec $w \in \mathbb{A}^\times$ et pour $i = 1, \ldots, n$, $h_i \in \mathbb{N}$ ($h_i \leq e_i$ et $h_i \leq f_i$) de sorte que $c \,|\, d$ car pour $i = 1, \ldots, n$, $g_i = \min(e_i, f_i)$. □

On obtient un résultat analogue pour l'existence du PPCM dans un anneau factoriel (prendre $g_i = \max(e_i, f_i)$ au lieu de $\min(e_i, f_i)$ dans la preuve).

Le théorème suivant sera admis pour ce cours :

Théorème 3.1.2. *Si un anneau $\mathbb{A}$ est factoriel, alors $\mathbb{A}[X]$ est factoriel.*

Par récurrence, on voit donc que si $\mathbb{A}$ est factoriel, alors l'anneau des polynômes en n variables $\mathbb{A}[X_1, \ldots, X_n]$ est factoriel.

3.1.2 Division euclidienne

Définition 3.1.6. *Soit $\mathbb{A}$ un anneau commutatif unitaire et intègre. Alors l'anneau $\mathbb{A}$ est dit* euclidien *s'il existe une application $\phi : \mathbb{A} \setminus \{0\} \to \mathbb{N}$ appelée* stathme euclidien *telle que :*

- *pour tout a, $b \in \mathbb{A} \setminus \{0\}$, $\phi(a\,b) \geq \phi(a)$;*
- *pour tout a, $b \in \mathbb{A} \setminus \{0\}$, il existe q, $r \in \mathbb{A}$ tels que $a = q\,b + r$ avec $r = 0$ ou $\phi(r) < \phi(b)$.*

Les éléments q et r de $\mathbb{A}$ sont alors appelés respectivement quotient *et* reste *de la* division euclidienne de a par b.

L'anneau $\mathbb{A} = \mathbb{Z}$ est euclidien en prenant pour stathme la *valeur absolue*, i.e., $\phi : \mathbb{Z}\backslash\{0\} \to \mathbb{N},\ r \mapsto |r|$. Notons que dans ce cas, le couple (q, r) n'est pas unique ($8 = 2 \times 3 + 2 = 3 \times 3 - 1$). L'anneau $\mathbb{A} = \mathbb{K}[X]$ est euclidien en prenant pour stathme la fonction *degré*, i.e., $\phi : \mathbb{K}[X] \setminus \{0\} \to \mathbb{N},\ P \mapsto \deg_X(P)$. La fonction degré vérifie de plus l'inégalité $\deg_X(P_1 - P_2) \leq \max(\deg_X(P_1), \deg_X(P_2))$, pour tous polynômes P_1, $P_2 \in \mathbb{K}[X]$ ce qui implique alors l'unicité du quotient et du reste de la division euclidienne dans $\mathbb{K}[X]$.

Théorème 3.1.3. *Soient $\mathbb{A}$ un anneau et A, $B \in \mathbb{A}[X]$ avec $B \neq 0$ et de coefficient de tête inversible dans $\mathbb{A}$. Alors il existe un unique couple $(Q, R) \in \mathbb{A}[X]^2$ tel que $A = Q\,B + R$ et $\deg(R) < \deg(B)$.*

Si $\mathbb{A}$ est un corps $\mathbb{K}$, alors nous avons le même résultat sans l'hypothèse sur le coefficient de tête de B.

Théorème 3.1.4. *Tout anneau euclidien est principal, i.e., tout idéal d'un anneau euclidien est engendré par un seul élément.*

Démonstration. Soit $\mathbb{A}$ un anneau euclidien muni du stathme ϕ et soit I un idéal non-nul de $\mathbb{A}$. Soit $a \in I$ tel que $\phi(a)$ soit minimal, i.e., pour tout $x \in I \setminus \{0\}$, $\phi(x) \geq \phi(a)$. On va montrer que $I = (a)$. Soit $b \in I \setminus \{0\}$, alors il existe $(q, r) \in \mathbb{A}^2$ tel que $b = q\,a + r$ et $r = 0$ ou $\phi(r) < \phi(a)$. Supposons $r \neq 0$. On a alors $r = b - q\,a \in I$ et $\phi(r) < \phi(a)$ ce qui contredit le fait que $\phi(a)$ soit minimal. Par conséquent $r = 0$ et $b = q\,a \in (a)$ ce qui termine la preuve. □

On peut aussi montrer que tout anneau principal est factoriel (admis pour ce cours) de sorte que l'on a les implications suivantes :

$$\mathbb{A} \text{ euclidien} \Rightarrow \mathbb{A} \text{ principal} \Rightarrow \mathbb{A} \text{ factoriel.}$$

Certaines propriétés des anneaux factoriels restent vraies dans un anneau principal.

Proposition 3.1.1. *Soit $\mathbb{A}$ un anneau principal. Alors toute famille d'éléments de $\mathbb{A}$ admet un* PGCD *et un* PPCM.

Démonstration. Soient a, $b \in \mathbb{A}$ et considérons l'idéal $I = (a) + (b) = \{x\,a + y\,b,\ x, y \in \mathbb{A}\}$. Comme $\mathbb{A}$ est principal, il existe $d \in \mathbb{A}$ tel que $I = (d)$. On vérifie alors que d est un PGCD de a et b. De la même façon, on considère l'idéal $J = (a) \cap (b)$ et on montre que si $J = (m)$, alors m est un PPCM de a et b. □

Une conséquence directe de cette proposition est l'existence de l'*égalité de Bézout* dans un anneau principal.

Théorème 3.1.5. *Soient $\mathbb{A}$ un anneau principal, a, $b \in \mathbb{A}$ et $d = \text{PGCD}(a, b)$ un* PGCD *de a et b. Alors il existe $(u, v) \in \mathbb{A}^2$ tel que $u\,a + v\,b = d$.*

En particulier, si a et b sont premiers entres eux, alors il existe $(u, v) \in \mathbb{A}^2$ tel que $u\,a + v\,b = 1$.

Théorème 3.1.6 (Gauss)**.** *Soient $\mathbb{A}$ un anneau principal et a, b, $c \in \mathbb{A}$. Si c divise le produit $a\,b$ et si c est premier avec b, alors c divise a.*

Démonstration. Comme c et b sont premiers entres eux, alors il existe $(u, v) \in \mathbb{A}^2$ tel que $u\,c + v\,b = 1$. En multipliant cette égalité par a, il vient $u\,a\,c + v\,a\,b = a$. Maintenant comme c divise $a\,b$, il existe $q \in \mathbb{A}$ tel que $a\,b = q\,c$ de sorte que l'on obtient $(u\,a + v\,q)\,c = a$ ce qui montre que c divise a. □

Corollaire 3.1.1. *Soient $\mathbb{A}$ un anneau principal et a, b, $p \in \mathbb{A}$ avec p irréductible. Si p divise $a\,b$, alors p divise a ou p divise b.*

Finalement, on remarque que l'équivalence (a irréductible $\Leftrightarrow$ (a) premier) reste vérifiée dans un anneau principal.

3.2 Calcul du PGCD et des coefficients de Bézout

3.2.1 Dans un anneau euclidien : le cas des entiers

Soit $\mathbb{A}$ un anneau effectif euclidien muni d'un stathme ϕ dans lequel on suppose que l'on sait effectuer la division euclidienne c'est-à-dire qu'étants donnés a et b dans $\mathbb{A}$, on sait déterminer $(q, r) \in \mathbb{A}^2$ tel que $a = q\,b + r$ avec $r = 0$ ou $\phi(r) < \phi(b)$. On note alors $q = \text{quo}(a, b)$ et $r = a \mod b$ ou $r = \text{rem}(a, b)$ le quotient et le reste de la division euclidienne de a par b.

Le problème du calcul de PGCD dans $\mathbb{A}$ est alors le suivant : étant donnés $a\,, b \in \mathbb{A}$, calculer $\text{PGCD}(a, b)$, i.e., un PGCD de a et b.

Pour ce faire, on commence par effectuer la division euclidienne de a par b pour obtenir q et r tels que $a = q\,b + r$ avec $r = 0$ ou $\phi(r) < \phi(b)$. Si $r = 0$, alors b divise a et $\text{PGCD}(a, b) = b$. Sinon on a $\text{PGCD}(a, b) \sim \text{PGCD}(b, r)$. On effectue alors la division euclidienne de b par r et ainsi de suite. En notant $r_0 = a$, $r_1 = b$, on définit ainsi la suite des restes successifs :

$$r_{i-2} = q_i\,r_{i-1} + r_i,\ i \geq 2, \quad r_i = 0 \text{ ou } \phi(r_i) < \phi(r_{i-1}). \tag{3.2}$$

Comme le stathme ϕ est à valeur dans $\mathbb{N}$, il existe $k \in \mathbb{N}$ tel que $r_{k+1} = 0$ et $r_k \neq 0$. On a alors $r_k = \text{PGCD}(a, b)$. En effet $r_{k+1} = 0$ implique r_k divise r_{k-1} et par conséquent $r_k = \text{PGCD}(r_{k-1}, r_k) \sim \text{PGCD}(r_{k-2}, r_{k-1}) \sim \cdots \sim \text{PGCD}(r_1, r_2) \sim \text{PGCD}(r_0, r_1) = \text{PGCD}(a, b)$. On obtient alors l'algorithme d'Euclide 7 pour le calcul du PGCD dans un anneau euclidien : Cet algorithme peut aussi s'écrire à l'aide d'appels récursifs (Algorithme 8).
Comme nous l'avons vu précédemment, en général, il n'y a pas unicité du couple (q, r) dans la division euclidienne donc ces algorithmes peuvent produire plusieurs suites différentes de restes successifs. De plus le nombre d'étapes avant de tomber sur un reste nul peut varier :

Exemple 3.2.1. *Soit $\mathbb{A} = \mathbb{Z}$, $a = 30$ et $b = 18$. La suite des restes peut alors s'écrire*

$$\begin{cases} 30 &= 1 \times 18 + 12, \\ 18 &= 1 \times 12 + 6, \\ 12 &= 2 \times 6 + 0, \end{cases}$$

Algorithme 7 *Algorithme d'Euclide*

Entrées : $\mathbb{A}$ anneau euclidien, a, $b \in \mathbb{A}$.
Sorties : $\text{PGCD}(a, b)$ (**un** PGCD de a et b).

$r_0 = a$, $r_1 = b$, $i = 1$;
Tant que $r_i \neq 0$ **Faire**
  $r_{i+1} = r_{i-1} \mod r_i$;
  $i = i + 1$;
Fin tant que
Retourner r_{i-1}.

Algorithme 8 *Algorithme d'Euclide récursif*

Entrées : $\mathbb{A}$ anneau euclidien, a, $b \in \mathbb{A}$.
Sorties : $\text{PGCD}(a, b)$.

$r = a \mod b$;
Si $r = 0$ **Alors**
  Retourner b ;
Sinon
  Calculer $\text{PGCD}(b, r)$ à l'aide d'un appel récursif.
Finsi

d'où $\text{PGCD}(30, 18) = 6$. *Mais on a aussi*

$$\begin{cases} 30 &= 2 \times 18 + (-6), \\ 18 &= (-3) \times (-6) + 0, \end{cases}$$

qui nous donne $\text{PGCD}(30, 18) = -6$.

Nous nous intéressons maintenant au calcul des coefficients de Bézout : étant donnés $a\,, b \in \mathbb{A}$, calculer $(u, v) \in \mathbb{A}^2$ tel que $u\,a + v\,b = \text{PGCD}(a, b)$. Pour ceci on va devoir garder trace des quotients q_i calculés à chaque étape de l'algorithme d'Euclide. Définissons la suite des triplets successifs $T_i = (r_i, u_i, v_i) \in \mathbb{A}^3$ définis par $T_0 = (a, 1, 0)$, $T_1 = (b, 0, 1)$ et

$$T_i = T_{i-2} - q_i\,T_{i-1},\ i \geq 2, \quad q_i = \text{quo}(r_{i-2}, r_{i-1}).$$

Proposition 3.2.1. *Avec les notations ci-dessus, pour tout i, on a $r_i = u_i\,a + v_i\,b$.*

Démonstration. Récurrence sur i. □

En notant toujours k l'entier tel que $r_{k+1} = 0$ et $r_k \neq 0$, on a alors $r_k = \text{PGCD}(a, b)$ et $u_k\,a + v_k\,b = r_k$. On obtient alors l'algorithme d'Euclide étendu pour le calcul du PGCD et des coefficients de Bézout.

Exemple 3.2.2. *Reprenons l'exemple (3.2.1) où $a = 30$ et $b = 18$. En utilisant la suite de divisions euclidiennes*

$$\begin{cases} 30 &= 1 \times 18 + 12, \\ 18 &= 1 \times 12 + 6, \\ 12 &= 2 \times 6 + 0, \end{cases}$$

Algorithme 9 *Algorithme d'Euclide étendu*

Entrées : $\mathbb{A}$ anneau euclidien, a, $b \in \mathbb{A}$.
Sorties : $\text{PGCD}(a,b)$ (**un** PGCD de a et b) et $(u,v) \in \mathbb{A}^2$ tel que $u\,a + v\,b = \text{PGCD}(a,b)$.

$r_0 = a$, $u_0 = 1$, $v_0 = 0$, $r_1 = b$, $u_1 = 0$, $v_1 = 1$, $i = 1$;
Tant que $r_i \neq 0$ **Faire**
$q_{i+1} = \text{quo}(r_{i-1}, r_i)$;
$r_{i+1} = r_{i-1} \mod r_i$;
$u_{i+1} = u_{i-1} - q_{i+1}\,u_i$;
$v_{i+1} = v_{i-1} - q_{i+1}\,v_i$;
$i = i + 1$;
Fin tant que
Retourner r_{i-1} et (u_{i-1}, v_{i-1}).

on obtient le tableau de valeur suivant :

Itération i	r_i	u_i	v_i	r_{i+1}	u_{i+1}	v_{i+1}	q_{i+1}
0	30	1	0	18	0	1	
1	18	0	1	12	1	-1	1
2	12	1	-1	6	-1	2	1
3	6	-1	2	0	3	-5	2

On a donc $(-1) \times 30 + 2 \times 18 = 6$.

3.2.2 Le théorème des restes chinois

Le théorème des restes chinois permet de reconstruire un nombre entier à partir de ses valeurs modulo certains autres entiers. Il est très utile pour développer des méthodes modulaires en calcul formel. Le problème s'énonce de la manière suivante : étant donnés quatre nombres entiers a, b, m et n avec $m\,n \geq 1$, déterminer tous les entiers $x \in \mathbb{Z}$ tels que :

$$(S) : \begin{cases} x \equiv a \mod m, \\ x \equiv b \mod n. \end{cases}$$

Théorème 3.2.1. *Avec les notations précédentes, si* $\text{PGCD}(m,n) = 1$, *alors* (S) *admet une et une seule solution modulo le produit* $m\,n$.

Démonstration. On considère l'application φ définie par :

$$\begin{array}{rccc} \varphi : & \mathbb{Z} & \to & (\mathbb{Z}/m\,\mathbb{Z}) \times (\mathbb{Z}/n\,\mathbb{Z}), \\ & x & \mapsto & (x \mod m, x \mod n). \end{array}$$

On vérifie alors que φ est un morphisme d'anneau. Son noyau $\ker(\varphi)$ est donné par

$$\ker(\varphi) = \{x \in \mathbb{Z} \mid x \equiv 0 \mod m \text{ et } x \equiv 0 \mod n\} = m\,n\,\mathbb{Z},$$

puisque $\text{PGCD}(m,n)=1$. φ induit donc une injection $\tilde{\varphi}:\mathbb{Z}/(m\,n\,\mathbb{Z})\to(\mathbb{Z}/m\,\mathbb{Z})\times(\mathbb{Z}/n\,\mathbb{Z})$ qui est une bijection car $\text{Card}(\mathbb{Z}/(m\,n\,\mathbb{Z}))=\text{Card}((\mathbb{Z}/m\,\mathbb{Z})\times(\mathbb{Z}/n\,\mathbb{Z}))=m\,n$ ce qui prouve le résultat. □

D'un point de vue effectif, la solution de (S) peut être calculée à l'aide de l'algorithme d'Euclide étendu. En effet, en appliquant l'algorithme d'Euclide étendu à m et n premiers entre eux, on trouve deux entiers u et v tels que $u\,m+v\,n=1$. On a alors

$$\begin{cases} u\,m \equiv 0 \mod m, \\ u\,m \equiv 1 \mod n, \end{cases} \qquad \begin{cases} v\,n \equiv 1 \mod m, \\ v\,n \equiv 0 \mod n, \end{cases}$$

de sorte que $x_0=a\,v\,n+b\,u\,m$ est une solution particulière de (S). La solution générale de (S) s'écrit $x_0+k\,m\,n,\ k\in\mathbb{Z}$.

Exemple 3.2.3. *Prenons $a=2$, $b=1$, $m=3$ et $n=5$. En appliquant l'algorithme d'Euclide étendu à 3 et 5 premiers entre eux, on trouve $2\times 3+(-1)\times 5=1$. Une solution de (S) est alors donné par $x_0=2\times(-1)\times 5+1\times 2\times 3=-4$ qui vérifie bien*

$$\begin{cases} x_0 \equiv 2 \mod 3, \\ x_0 \equiv 1 \mod 5. \end{cases}$$

Le théorème 3.2.1 se généralise au cas où le système (S) possède plus de deux congruences.

Corollaire 3.2.1. *Soient $m_1,\ldots,m_s$ des entiers positifs deux à deux premiers entre eux, i.e., $\text{PGCD}(m_i,m_j)=1$, pour tout $i,\ j=1,\ldots,s,\ i\neq j$. Soit $M=m_1\times\cdots\times m_s=\prod_{i=1}^{s}m_i$ le produit des m_i. Alors pour tout s-uplet d'entiers $a_1,\ldots,a_s$ le système de congruences*

$$(S):\begin{cases} x \equiv a_1 \mod m_1, \\ \vdots \\ x \equiv a_s \mod m_s, \end{cases}$$

admet une et une seule solution modulo M.

Comme précédemment, le système de congruences (S) peut se résoudre en utilisant l'algorithme d'Euclide étendu. Pour $i=1,\ldots,s$, posons $M_i=M/m_i$ qui vérifie $\text{PGCD}(m_i,M_i)=1$. En appliquant l'algorithme d'Euclide étendu, on peut calculer un entier w_i tel que $w_i\,M_i\equiv 1 \mod m_i$. Alors $x_0=a_1\,w_1\,M_1+\cdots+a_s\,w_s\,M_s=\sum_{i=1}^{s}a_i\,w_i\,M_i$ est une solution particulière de (S). La solution générale de (S) s'écrit $x_0+k\,M,\ k\in\mathbb{Z}$. Cette méthode est appelée *méthode de Lagrange* pour la résolution du système de congruences (S).

Exemple 3.2.4. *Considérons le système de congruences*

$$(S):\begin{cases} x \equiv 3 \mod 11, \\ x \equiv 2 \mod 4, \\ x \equiv 7 \mod 15. \end{cases}$$

Les entiers 11, 4 et 15 sont bien premiers entre eux de sorte que la solution est unique modulo $M=660$. On a $M_1=M/11=60$, $M_2=M/4=165$ et $M_3=M/15=44$.

1. *On cherche* w_1 *tel que* $60\,w_1 \equiv 1 \mod 11$, *i.e.,* $5\,w_1 \equiv 1 \mod 11$. *L'algorithme d'Euclide étendu donne* $(-2)\times 5 + 1 \times 11 = 1$ *de sorte que* $w_1 = -2$ *convient ;*

2. *On cherche* w_2 *tel que* $165\,w_2 \equiv 1 \mod 4$, *i.e.,* $w_2 \equiv 1 \mod 4$. *L'algorithme d'Euclide étendu donne* $1 \times 1 + 0 \times 4 = 1$ *de sorte que* $w_2 = 1$ *convient ;*

3. *On cherche* w_3 *tel que* $44\,w_3 \equiv 1 \mod 15$, *i.e.,* $(-1)\,w_3 \equiv 1 \mod 15$. *L'algorithme d'Euclide étendu donne* $(-1)\times(-1) + 0 \times 15 = 1$ *de sorte que* $w_3 = -1$ *convient.*

On a donc trouvé la solution particulière $x_0 = 3\times(-2)\times 60 + 2\times 1\times 165 + 7\times(-1)\times 44 = -338$. *La solution générale est* $x = -338 + k \times 660,\ k \in \mathbb{Z}$. *La solution positive modulo* $M = 660$ *est alors* $-338 + 660 = 322$.

Il existe un moyen d'obtenir la solution positive la plus petite en utilisant *la méthode de Garner* décrite ci-dessous et qui est basée sur le théorème suivant :

Théorème 3.2.2. *Soient* $m_1, \ldots, m_s$ *des entiers positifs (pas nécessairement distincts). Alors l'application*

$$\begin{array}{rccl} \psi : & [\![0, m_1[\![\times \cdots \times [\![0, m_s[\![& \to & [\![0, m_1 \cdots m_s[\![, \\ & (r_1, \ldots, r_s) & \mapsto & r_1 + r_2\, m_1 + r_3\, m_1\, m_2 + \cdots + r_s\, m_1 \cdots m_{s-1}, \end{array}$$

est bijective.

Démonstration. Exercice. □

Considérons le système de congruences suivant :

$$(S) : \begin{cases} x & \equiv & a_1 & \mod & m_1, \\ \vdots & & & & \\ x & \equiv & a_s & \mod & m_s, \end{cases}$$

où les m_i sont deux à deux premiers entre eux et cherchons la solution de (S) appartenant à $[\![0, m_1 \cdots m_s[\![$. Cette solution s'écrit

$$x = r_1 + r_2\, m_1 + r_3\, m_1\, m_2 + \cdots + r_s\, m_1 \cdots m_{s-1}, \quad r_i \in [\![0, m_i[\![,\ i = 1, \ldots, s.$$

Les r_i se calculent alors par récurrence comme suit :

- $r_1 \equiv a_1 \mod m_1$ est le reste de la division euclidienne de a_1 par m_1 ;
- Supposons que l'on connaisse $r_1, \ldots, r_{k-1}$. On cherche r_k tel que $x \equiv a_k \mod m_k$ donc $r_1 + r_2\, m_1 + \cdots + r_{k-1}\, m_1 \cdots m_{k-2} + r_k\, m_1 \cdots m_{k-1} \equiv a_k \mod m_k$. Puisque $\text{PGCD}(m_k, m_1 \cdots m_{k-1}) = 1$, l'algorithme d'Euclide étendu nous donne (u_k, v_k) tel que $u_k\,(m_1 \cdots m_{k-1}) + v_k\, m_k = 1$. On pose alors

$$r_k = u_k\,(a_k - (r_1 + r_2\, m_1 + \cdots + r_{k-1}\, m_1 \cdots m_{k-2})) \mod m_k.$$

Exemple 3.2.5. *On considère le système de congruence*

$$(S) : \left\{ \begin{array}{rcrcr} x & \equiv & 1000 & \mod & 1891, \\ x & \equiv & 501 & \mod & 2499. \end{array} \right.$$

Cherchons x sous la forme $x = r_1 + r_2 \times 1891$ avec $r_1 \in [\![0, 1891[\![$ et $r_2 \in [\![0, 2499[\![$.

- *L'entier positif r_1 s'obtient comme reste de la division euclidienne de 1000 par 1891 de sorte que $r_1 = 1000$;*

- *Calculons r_2. L'algorithme d'Euclide étendu appliqué à 1891 et 2499 nous fournit u et v tels que $u \times 1891 + v \times 2499 = 1$. On trouve $u = -1007$ et $v = 762$. D'où $r_2 = -1007\,(501 - 1000) \mod 2499 = 194$.*

Finalement, on a $x = 1000 + 194 \times 1891 = 367854$.
Remarquons que sur cet exemple, la solution particulière fournit par la méthode de Lagrange est $x_0 = 1000 \times 762 \times 2499 + 501 \times (-1007) \times 1891 = 950215263$.

Le théorème des restes chinois peut se généraliser à tout anneau commutatif unitaire.

Théorème 3.2.3. *Soient $\mathbb{A}$ un anneau commutatif unitaire, $I_1, \ldots, I_s$ des idéaux de $\mathbb{A}$ tels que $I_j + I_k = \mathbb{A}$ pour tout j, $k = 1, \ldots, s$, $j \neq k$. Alors on a l'isomorphisme*

$$\mathbb{A}/\left(\cap_{j=1}^{s} I_j\right) \cong (\mathbb{A}/I_1) \times \cdots \times (\mathbb{A}/I_s)\,.$$

Dans le cas particulier où $\mathbb{A}$ est principal, alors pour tout $j = 1, \ldots, s$, $I_j = n_j\,\mathbb{A}$, $n_j \in \mathbb{A}$ avec les n_j premiers entre eux deux à deux et si on note $n = \prod_{i=1}^{s} n_i$, on a :

$$\mathbb{A}/(n\,\mathbb{A}) \cong (\mathbb{A}/n_1\,\mathbb{A}) \times \cdots \times (\mathbb{A}/n_s\,\mathbb{A})\,.$$

En particulier, lorsque $\mathbb{A} = \mathbb{K}[X]$ est l'anneau des polynômes univariés à coefficients dans un corps $\mathbb{K}$. Si $P_1, \ldots, P_s$ sont des polynômes de $\mathbb{K}[X]$ premiers entre eux deux à deux, alors pour tout s-uplet de polynômes $(Q_1, \ldots, Q_s) \in \mathbb{K}[X]^s$, il existe un polynôme P, unique modulo le produit $P_1 \cdots P_s$, tel que

$$\left\{ \begin{array}{rcrcr} P & \equiv & Q_1 & \mod & P_1, \\ \vdots & & & & \\ P & \equiv & Q_s & \mod & P_s. \end{array} \right.$$

Le problème d'interpolation de Lagrange peut alors se voir comme un cas particulier où les polynômes P_i sont de la forme $X - x_i$ avec les $x_i \in \mathbb{K}$ deux à deux distincts et les Q_i sont des constantes $y_i \in \mathbb{K}$.

3.2.3 Dans un anneau de polynômes

Nous nous plaçons maintenant dans le cas d'un anneau de polynômes $\mathbb{A}[X]$ à coefficients dans un anneau effectif $\mathbb{A}$. Étant donnés A, $B \in \mathbb{A}[X]$, on cherche à déterminer $\text{PGCD}(A, B)$. Lorsque $\mathbb{A}$ est un corps $\mathbb{K}$, l'anneau $\mathbb{K}[X]$ étant euclidien pour le stathme deg, nous pouvons appliquer l'algorithme d'Euclide 7 pour calculer $\text{PGCD}(A, B)$.

Proposition 3.2.2. *L'algorithme d'Euclide 7 calcule un* PGCD *de deux polynômes A et B à coefficients dans un corps $\mathbb{K}$ en $O(\deg(A)\deg(B))$ opérations dans $\mathbb{K}$.*

Démonstration. Supposons $\deg(A) \geq \deg(B)$. D'après la sous-section 2.2.3, le calcul de P mod Q peut se faire par la méthode naïve en $2\deg(Q)(\deg(P)-\deg(Q)+1)$ opérations dans $\mathbb{K}$. En notant R_i les restes successifs, le coût de l'algorithme d'Euclide est donc majoré par $\sum_{i\geq 1} 2\deg(R_i)(\deg(R_{i-1})-\deg(R_i)+1)$. En majorant tous les $\deg(R_i)$, $i \geq 1$ par $\deg(B)$ on obtient $2\deg(B)\sum_{i\geq 1}(\deg(R_{i-1})-\deg(R_i)+1) = 2\deg(B)(\deg(R_0)+\deg(B))$ puisque dans le pire cas le nombre d'étapes est égal à $\deg(B)$. D'où le résultat puisque $R_0 = A$. □

Exemple 3.2.6. *Considérons les polynômes $A = X^8 + X^6 - 3X^4 - 3X^3 + 8X^2 + 2X - 5$ et $B = 3X^6 + 5X^4 - 4X^2 - 9X + 21$ de $\mathbb{Z}[X] \subset \mathbb{Q}[X]$. La suite des restes obtenue en appliquant l'algorithme d'Euclide est alors donnée par :*

$$\begin{cases} R_2 = -\frac{5}{9}X^4 + \frac{1}{9}X^2 - \frac{1}{3}, \\ R_3 = -\frac{117}{25}X^2 - 9X + \frac{441}{25}, \\ R_4 = \frac{233150}{19773}X - \frac{102500}{6591}, \\ R_5 = -\frac{1288744821}{543589225}, \\ R_6 = 0, \end{cases}$$

de sorte qu'un PGCD *de A et B est donné par $R_5 = -\frac{1288744821}{543589225}$. Notons que $R_5 \sim 1$ donc A et B sont premiers entre eux.*

L'exemple précédent illustre deux problèmes apparaissant lorsque l'on applique l'algorithme d'Euclide à deux polynômes :

1. même si les polynômes de départ sont à coefficients entiers, des dénominateurs apparaissent dans les calculs ;
2. la taille des entiers apparaissant dans les restes successifs augmente significativement.

L'apparition des dénominateurs provient du fait que pour chaque division euclidienne nous devons inverser le coefficient de tête du polynôme par lequel on divise (voir Théorème 3.1.3). Dans notre exemple, dès la première étape le coefficient de tête de B est 3 donc nous devons l'inverser pour trouver R_2 qui aura donc des dénominateurs dans ses coefficients. Notons que même si nous partons de deux polynômes unitaires, des dénominateurs peuvent quand même apparaitre au cours de l'algorithme comme l'illustre l'exemple suivant :

Exemple 3.2.7. *Considérons les polynômes* $A = X^4 - 13\,X^3 + 2\,X^2 - X - 1$ *et* $B = X^2 - X - 1$. *La suite des restes est alors donnée par :*

$$\begin{cases} R_2 = -22\,X - 10, \\ R_3 = -\frac{41}{121} \\ R_4 = 0. \end{cases}$$

Afin d'éviter l'apparition des dénominateurs[1], une idée consiste à utiliser la *pseudo-division euclidienne.*

Définition 3.2.1. *Soient* A *et* B *deux polynômes à coefficients dans un anneau* $\mathbb{A}$ *et* b *le coefficient de tête de* B. *Le* pseudo-reste $\tilde{R}$ *de* A *par* B *noté* $\mathrm{prem}(A, B)$ *est défini par :*

$$b^{\deg(A)-\deg(B)+1}\,A = \tilde{Q}\,B + \tilde{R}, \quad \tilde{Q}, \tilde{R} \in \mathbb{A}[X], \quad \deg(\tilde{R}) < \deg(B).$$

On peut donc obtenir un algorithme d'Euclide « sans dénominateurs » en remplaçant les restes par des pseudo-restes dans l'algorithme d'Euclide.

Algorithme 10 *Algorithme d'Euclide par pseudo-restes pour le calcul du* PGCD

Entrées : A, $B \in \mathbb{A}$.
Sorties : $\mathrm{PGCD}(A, B)$ (**un** PGCD de A et B).

```
R_0 = A, R_1 = B, i = 1 ;
Tant que R_i ≠ 0 Faire
    R_{i+1} = prem(R_{i-1}, R_i) ;
    i = i + 1 ;
Fin tant que
Retourner R_{i-1}.
```

Exemple 3.2.8. *Reprenons l'exemple 3.2.6 avec la pseudo-division euclidienne. La suite des pseudo-restes est alors :*

$$\begin{cases} R_2 = -15\,X^4 + 3\,X^2 - 9, \\ R_3 = 15795\,X^2 + 30375\,X - 59535, \\ R_4 = 1254542875143750\,X - 1654608338437500, \\ R_5 = 125933387955007431009311419921875 00. \\ R_6 = 0. \end{cases}$$

On remarque qu'en appliquant la pseudo-division euclidienne, on fait certes disparaitre les dénominateurs mais la taille des entiers apparaissant dans les coefficients des pseudo-restes

[1] Notons qu'en calcul formel, on évite autant que possible les calculs sur les nombres rationnels pour lesquels l'addition elle-même requiert des calculs de multiplications et éventuellement des calculs de PGCD.

croît très rapidement donc que cela n'est pas très satisfaisant en pratique. Pour diminuer ce phénomène de croissance des coefficients, une possibilité consiste à diviser à chaque étape le pseudo-reste par le PGCD de ses coefficients (aussi appelé *contenu*) pour obtenir une suite de polynômes *primitifs*, i.e., dont le contenu vaut 1. On parle alors de *suite des pseudo-restes primitifs*.

Exemple 3.2.9. *Dans l'exemple 3.2.6, la suite des pseudo-restes primitifs est :*

$$\begin{cases} R_2 = -5\,X^4 + X^2 - 3, \\ R_3 = 13\,X^2 + 25\,X - 49, \\ R_4 = 4663\,X - 6150, \\ R_5 = 1. \\ R_6 = 0. \end{cases}$$

Cependant en pratique, cela rajoute un calcul de PGCD des coefficients à chaque étape ce qui est trop coûteux. Notons finalement qu'il existe un moyen de modifier légèrement cette approche en *prévoyant* les facteurs communs qui apparaissent dans les coefficients de sorte à limiter leur croissance (sans calculer de PGCD dans $\mathbb{A}$) : c'est l'*algorithme des sous-résultants* que nous ne détaillerons pas dans ce cours.

Une autre façon d'éviter la croissance des coefficients est d'utiliser une méthode modulaire c'est-à-dire de réduire modulo un ou plusieurs nombres premiers p et d'appliquer l'algorithme d'Euclide dans $\mathbb{F}_p = \mathbb{Z}/(p\,\mathbb{Z})$. Pour construire un algorithme modulaire pour le calcul du PGCD, plusieurs choses doivent être prises en considération. Premièrement, si on note $D = \text{PGCD}(P, Q)$, $\overline{P}$, $\overline{Q}$ et $\overline{D}$ les réductions de P, Q et D modulo un nombre premier p, alors nous n'avons pas toujours $\overline{D} = \text{PGCD}(\overline{P}, \overline{Q})$. En effet si on note P_1 et Q_1 les cofacteurs de P et Q, i.e., $P = P_1\,D$ et $Q = Q_1\,D$, alors on a nécessairement $\overline{P} = \overline{P_1}\,\overline{D}$ et $\overline{Q} = \overline{Q_1}\,\overline{D}$ de sorte que $\overline{D} \,|\, \text{PGCD}(\overline{P}, \overline{Q})$ mais pour avoir l'égalité il faut (et il suffit) que $\overline{P_1}$ et $\overline{Q_1}$ soient premiers entre eux dans $\mathbb{Z}/(p\,\mathbb{Z})$ ce qui n'est pas toujours vrai comme l'illustre l'exemple suivant.

Exemple 3.2.10. *Soient* $P = (X^3 + X + 3)\,(X + 1)$ *et* $Q = X\,(X + 1)$ *de sorte que* $D = \text{PGCD}(P, Q) = X + 1$. *Pour* $p = 3$, *on a* $\overline{P} = (X^3 + X)\,(X + 1)$ *et* $\overline{Q} = X\,(X + 1)$ *de sorte que* $\text{PGCD}(\overline{P}, \overline{Q}) = X\,(X + 1) \neq \overline{D} = X + 1$. *Cela provient du fait que les cofacteurs* $X^3 + X + 3$ *et* X *ne sont pas premiers entre eux dans* $\mathbb{Z}/(3\,\mathbb{Z})$.

Ceci implique que tout nombre premier ne sera pas « bon » mais qu'il n'y aura qu'un nombre fini de « mauvais » nombres premiers (avec les notations précédentes, ceux qui divisent le résultant de P_1 et Q_1 - cf Sous-section 3.3 suivante). De plus, pour choisir judicieusement ces nombres premiers de sorte à pouvoir reconstruire D à partir de $\overline{D}$, il nous faut avoir une idée de la taille des coefficients de D que l'on cherche. Pour $P = \sum_{i=1}^{d} p_i\,X^i \in \mathbb{Z}[X]$, on note $||P||_\infty = \max_{0 \leq i \leq d} |p_i|$ la *norme infinie* de P (aussi appelée *hauteur* de P) et $||P||_1 = \sum_{i=0}^{d} |p_i|$ la *norme 1* de P. Nous avons alors la borne suivante :

Lemme 3.2.1 (Borne de Mignotte). *Soit* $P = \sum_{i=0}^{d} p_i X^i \in \mathbb{Z}[X]$ *un* polynôme primitif, *i.e., le* PGCD *de ses coefficients* p_i *(aussi appelé* contenu de P*) vaut* 1. *Alors si un polynôme* $Q \in \mathbb{Z}[X]$ *de degré* n *divise* P, *on a :*

$$||Q||_\infty \leq (\sqrt{d+1})\, 2^n\, ||P||_\infty.$$

Démonstration. Admis pour ce cours. □

À partir de ce résultat, deux stratégies s'offrent à nous : soit on choisit un seul « grand » nombre premier pour calculer directement le PGCD dans $\mathbb{Z}[X]$ soit on choisit plusieurs « petits » nombres premiers, on déroule l'algorithme d'Euclide modulo chacun de ces nombres premiers p et on reconstruit le PGCD dans $\mathbb{Z}[X]$ à l'aide du théorème des restes chinois. Cette deuxième stratégie nécessite d'appliquer l'algorithme d'Euclide plusieurs fois mais les coefficients qui interviennent sont plus petits ce qui est un avantage non négligeable pour des polynômes de départ de degré et de norme infinie élevés. Les algorithmes 11 et 12 montrent comment chacune de ces stratégies peut être mise en oeuvre pour obtenir un algorithme modulaire calculant le PGCD de deux polynômes.

Notons que dans les deux algorithmes la condition utilisée pour vérifier que l'on a bien obtenu le PGCD, i.e., $\left(||\tilde{P}||_1\, ||S||_1 \leq B \text{ et } ||\tilde{Q}||_1\, ||S||_1 \leq B\right)$ est en fait une condition nécessaire et suffisante pour que $\mathrm{pp}(S)$ soit bien le PGCD de P et Q.

Algorithme 11 *Algorithme modulaire pour le* PGCD *utilisant un grand premier*

Entrées : P, $Q \in \mathbb{Z}[X]$ primitifs tels que $\deg(P) = n \geq \deg(Q)$.
Sorties : $\mathrm{PGCD}(P, Q)$ (**un** PGCD de P et Q).
- Soit A un majorant de $||P||_\infty$ et $||Q||_\infty$;
- Calculer le PGCD b des coefficients dominants de P et Q ;
- Soit $B = (\sqrt{n+1})\, 2^n\, A\, b$;
- Choisir un nombre premier p entre $2\,B$ et $4\,B$;
- Appliquer l'algorithme d'Euclide pour calculer le polynôme unitaire $R \in \mathbb{Z}[X]$ tel que $||R||_\infty \leq p/2$ et tel que la réduction modulo p de R est un PGCD des réductions $\overline{P}$ et $\overline{Q}$ de P et Q modulo p ;
- Calculer S, $\tilde{P}$ et $\tilde{Q}$ tels que $S \equiv b\,R \mod p$, $\tilde{P}\,S \equiv b\,P \mod p$, $\tilde{Q}\,S \equiv b\,Q \mod p$;

Si $\left(||\tilde{P}||_1\, ||S||_1 \leq B \text{ et } ||\tilde{Q}||_1\, ||S||_1 \leq B\right)$ **Alors**
- Retourner $\mathrm{pp}(S)$ la partie primitive de S, i.e., S divisé par son contenu ;

Sinon
- Choisir un autre nombre premier à l'étape 4 et recommencer.

Finsi

Étant donnés deux polynômes A, $B \in \mathbb{K}[X]$, l'algorithme d'Euclide étendu 9 peut être appliqué pour calculer $\mathrm{PGCD}(A, B)$ ainsi que deux polynômes U, $V \in \mathbb{K}[X]$ tels que

$$U\,A + V\,B = \mathrm{PGCD}(A, B), \quad \deg(U\,\mathrm{PGCD}(A, B)) < \deg(B),\ \deg(V\,\mathrm{PGCD}(A, B)) < \deg(A).$$

Algorithme 12 *Algorithme modulaire pour le* PGCD *utilisant plusieurs petits premiers*

Entrées : P, $Q \in \mathbb{Z}[X]$ primitif tels que $\deg(P) = n \geq \deg(Q)$.
Sorties : PGCD(P, Q) (**un** PGCD de P et Q).

- Soit A un majorant de $||P||_\infty$ et $||Q||_\infty$;
- Calculer le PGCD b des coefficients dominants de P et Q ;
- Soit $k = \lceil 2 \log_2 ((n+1)^n \, b \, A^{2n}) \rceil$, $B = (\sqrt{n+1}) \, 2^n \, A \, b$ et $l = \lceil \log_2(2\,B + 1) \rceil$
- Choisir un ensemble $\mathcal{S}$ de $2\,l$ nombres premiers inférieurs à $2\,k\,\log(k)$;
- Retirer de $\mathcal{S}$ les nombres premiers qui divisent b ;

Pour tout $p \in \mathcal{S}$ **Faire**

- Appliquer l'algorithme d'Euclide pour calculer le polynôme unitaire $R_p \in \mathbb{Z}[X]$ à coefficients dans $\{0, \ldots, p-1\}$ et tel que la réduction modulo p de R_p est un PGCD des réductions $\overline{P}$ et $\overline{Q}$ de P et Q modulo p ;

Fin pour

- Soit $e = \min_{p \in \mathcal{S}}(\deg(R_p))$ et $\mathcal{R} = \{p \in \mathcal{S} \mid \deg(R_p) = e\}$;

Si $\text{Card}(\mathcal{R}) \geq l$ **Alors**

- Enlever $\text{Card}(\mathcal{R}) - l$ éléments de $\mathcal{R}$ de sorte à n'en garder que l ;
- Appliquer le théorème des restes chinois pour calculer S, $\tilde{P}$ et $\tilde{Q}$ dans $\mathbb{Z}[X]$ de norme infinie inférieure à $(\prod_{p \in \mathcal{R}} p)/2$ tels que pour tout $p \in \mathcal{R}$, $S \equiv b\,R_p \mod p$, $\tilde{P}\,S \equiv b\,P \mod p$, $\tilde{Q}\,S \equiv b\,Q \mod p$;

Si $\Big(||\tilde{P}||_1 \, ||S||_1 \leq B \text{ et } ||\tilde{Q}||_1 \, ||S||_1 \leq B \Big)$ **Alors**

- Retourner pp(S) la partie primitive de S, i.e., S divisé par son contenu ;

Sinon

- Choisir un autre ensemble $\mathcal{S}$ à l'étape 4 et recommencer ;

Finsi

Sinon

- Choisir un autre ensemble $\mathcal{S}$ à l'étape 4 et recommencer.

Finsi

La condition sur les degrés se montre facilement par récurrence à partir de la proposition 3.2.1 et assure l'unicité des cofacteurs U et V une fois $\text{PGCD}(A, B)$ fixé. La complexité de l'algorithme d'Euclide étendu appliqué à deux polynômes A et B avec $\deg(A) = n \geq \deg(B)$ reste quadratique en n. Cet algorithme est très important en calcul formel. Il permet entre autres d'effectuer des *inversions modulaires* (et donc des divisions modulaires). En effet, si A et B sont deux polynômes premiers entre eux et U et V les cofacteurs tels que

$$U\,A + V\,B = 1, \quad \deg(U) < \deg(B),\ \deg(V) < \deg(A),$$

alors V est un inverse de B modulo A : $V = 1/B \mod A$.

Exemple 3.2.11. *Soient $A = X^2 - X - 1$ et $B = X^3 - 1$. Les polynômes A et B sont premiers entre eux et on a :*

$$-\frac{X^2-1}{2}\,A + \frac{X-1}{2}\,B = 1 \iff \frac{X-1}{2} = \frac{1}{X^3-1} \mod (X^2 - X - 1).$$

Si on note ϕ le « nombre d'or » défini par $A(\phi) = 0$, on peut par exemple en déduire l'égalité

$$\frac{\phi - 1}{\phi^3 - 1} = (\phi - 1)\,\frac{\phi - 1}{2} = \frac{(\phi-1)^2}{2}.$$

On peut ainsi utiliser l'algorithme d'Euclide étendu pour « simplifier » des expressions symboliques ce qui est un enjeu crucial en calcul formel.

Remarque 3.2.1. *Il existe un algorithme rapide pour le calcul du* PGCD *(d'entiers ou de polynômes). Cet algorithme se base sur des multiplications et des divisions euclidiennes rapides et utilise la notion de* demi-PGCD. *Cet algorithme ne sera pas présenté dans ce cours (voir TD) mais permet de calculer le* PGCD *de deux polynômes de degré majoré par n en $O(\mathbf{M}(n)\log(n))$ opérations arithmétiques (où $\mathbf{M}$ une fonction telle que le produit de deux polynômes de degré au plus n puisse se calculer en $O(\mathbf{M}(n))$ avec $\mathbf{M}(n) \geq 2\,\mathbf{M}(n/2)$) ce qui est le meilleur résultat connu à ce jour.*

3.3 Résultant de deux polynômes

Dans cette section $\mathbb{A}$ désigne un anneau commutatif, unitaire et intègre et $\mathbb{K}$ un corps.

3.3.1 Matrice de Sylvester et forme échelonnée

Définition 3.3.1. *Soient $A = \sum_{i=0}^{n} a_i\,X^i$, $B = \sum_{i=0}^{m} b_i\,X^i \in \mathbb{A}[X]$ de degrés respectifs n et m. La* matrice de Sylvester de A et B *notée* $\text{Syl}(A, B)$ *est la matrice carrée de taille $n + m$*

définie par

$$\mathrm{Syl}(A,B) = \begin{pmatrix} a_n & a_{n-1} & \dots & \dots & a_0 & & & \\ & a_n & a_{n-1} & \dots & \dots & a_0 & & \\ & & \ddots & \ddots & & & \ddots & \\ & & & a_n & a_{n-1} & \dots & \dots & a_0 \\ b_m & b_{m-1} & \dots & \dots & b_0 & & & \\ & b_m & b_{m-1} & \dots & \dots & b_0 & & \\ & & \ddots & \ddots & & & \ddots & \\ & & & b_m & b_{m-1} & \dots & \dots & b_0 \end{pmatrix},$$

où les m premières lignes contiennent les coefficients de A et les n suivantes contiennent les coefficients de B.

Cette matrice est étroitement liée à l'algorithme d'Euclide étendu car sa transposée $\mathrm{Syl}(A,B)^T$ représente la matrice de l'application linéaire

$$\begin{array}{ccc} \mathbb{A}_{m-1}[X] \times \mathbb{A}_{n-1}[X] & \to & \mathbb{A}_{n+m-1}[X], \\ (\quad U \quad , \quad V \quad) & \mapsto & U\,A + V\,B, \end{array}$$

où pour $k \in \mathbb{N}$, $\mathbb{A}_k[X]$ désigne l'ensemble des polynômes de degré inférieur ou égal à k. Les combinaisons linéaires des lignes de $\mathrm{Syl}(A,B)$ donnent alors les coefficients des polynômes qui peuvent s'écrire sous la forme $U\,A+V\,B$. Si $\mathbb{A}$ est un corps $\mathbb{K}$, le PGCD de A et B est alors caractérisé comme l'élément de plus petit degré qui peut être obtenu ainsi et peut être calculé à partir d'une *forme échelonnée en ligne* de $\mathrm{Syl}(A,B)$. Cette dernière forme échelonnée en ligne est donnée par une matrice $P\,\mathrm{Syl}(A,B)$ avec P inversible telle que :

1. Toutes les lignes nulles de $P\,\mathrm{Syl}(A,B)$ sont en-dessous de ses lignes non-nulles ;

2. Le premier coefficient non nul de chaque ligne non nulle de $P\,\mathrm{Syl}(A,B)$ est strictement à droite du premier coefficient non nul de sa ligne précédente.

Lemme 3.3.1. *Soient A et B deux polynômes à coefficients dans un corps $\mathbb{K}$. La dernière ligne non nulle d'une forme échelonnée en ligne de* $\mathrm{Syl}(A,B)$ *contient les coefficients d'un* PGCD *de A et B.*

Notons qu'une forme échelonnée en ligne d'une matrice à coefficients dans un corps $\mathbb{K}$ peut s'obtenir en appliquant la réduction de Gauss.

Exemple 3.3.1. *Considérons les polynômes $A = X^4 - X^3 - 7\,X^2 + 2\,X + 3$ et $B = X^3 -$*

$4X^2 + 2X + 3$. *On a*

$$\mathrm{Syl}(A,B) = \begin{pmatrix} 1 & -1 & -7 & 2 & 3 & 0 & 0 \\ 0 & 1 & -1 & -7 & 2 & 3 & 0 \\ 0 & 0 & 1 & -1 & -7 & 2 & 3 \\ 1 & -4 & 2 & 3 & 0 & 0 & 0 \\ 0 & 1 & -4 & 2 & 3 & 0 & 0 \\ 0 & 0 & 1 & -4 & 2 & 3 & 0 \\ 0 & 0 & 0 & 1 & -4 & 2 & 3 \end{pmatrix},$$

qui admet comme forme échelonnée en ligne

$$\begin{pmatrix} 1 & -1 & -7 & 2 & 3 & 0 & 0 \\ 0 & 1 & -1 & -7 & 2 & 3 & 0 \\ 0 & 0 & 1 & -1 & -7 & 2 & 3 \\ 0 & 0 & 0 & -14 & 45 & -3 & -18 \\ 0 & 0 & 0 & 0 & -\frac{5}{7} & \frac{12}{7} & \frac{9}{7} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{10} & -\frac{3}{10} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

On obtient donc $\mathrm{PGCD}(A,B) = X - 3$. *Remarquons aussi que la dimension du noyau de la matrice donne le degré du* PGCD.

Définition 3.3.2. *Soient* A, $B \in \mathbb{A}[X]$. *Le* résultant de A et B *noté* $\mathrm{Res}(A,B)$ *ou encore* $\mathrm{Res}_X(A,B)$ *est le déterminant de la matrice de Sylvester* $\mathrm{Syl}(A,B)$ *de* A *et* B.

Le résultant de A et B appartenant à $\mathbb{A}[X]$ est donc un élément de l'anneau $\mathbb{A}$.

Proposition 3.3.1. *Soient* A *et* B *deux polynômes à coefficients dans un corps* $\mathbb{K}$. *Alors* A *et* B *sont premiers entre eux si et seulement si* $\mathrm{Res}(A,B) \neq 0$.

Démonstration. Le déterminant de la matrice de Sylvester s'obtient comme produit des éléments diagonaux de sa forme échelonnée en ligne. Par conséquent, $\mathrm{Res}(A,B) \neq 0$ implique qu'aucun élément diagonal n'est nul donc le PGCD obtenu est sur la dernière ligne donc de degré 0, i.e., A et B sont premiers entre eux. Inversement si le PGCD de A et B est une constante non nulle $d \in \mathbb{K}$ alors le dernier élément diagonal de la forme échelonnée en ligne de $\mathrm{Syl}(A,B)$ est non nul. De plus en multipliant par X^k, $k = 1, \ldots, n+m-1$ l'égalité $U\,A + V\,B = d$ on voit que tout élément diagonal de la forme échelonnée en ligne est non nul et donc $\mathrm{Res}(A,B) \neq 0$. □

La proposition 3.3.1 nous fournit une méthode pour tester si deux polynômes sont premiers entre eux : il suffit de calculer $\mathrm{Res}(A,B)$ et de regarder s'il est nul ou non. Notons que l'on peut montrer que la proposition 3.3.1 se généralise au cas de polynômes à coefficients dans un anneau factoriel.

3.3.2 Propriétés du résultant

Le premier résultat que nous donnons est une formule exprimant le résultant de deux polynômes A et B en fonction des racines de A et B.

Théorème 3.3.1 (Formule de Poisson)**.** *Soient A, $B \in \mathbb{A}[X]$. Si A et B s'écrivent*

$$A = a\,(X-\alpha_1)\cdots(X-\alpha_n), \quad B = b\,(X-\beta_1)\cdots(X-\beta_m),$$

alors on a

$$\mathrm{Res}(A,B) = a^m\,b^n\prod_{i,j}(\alpha_i-\beta_j), \tag{3.3}$$

ou encore

$$\mathrm{Res}(A,B) = (-1)^{m\,n}\,b^n\prod_{1\le j\le m}A(\beta_j) = a^m\prod_{1\le j\le n}B(\alpha_j) = (-1)^{m\,n}\,\mathrm{Res}(B,A).$$

Démonstration. Le facteur $a^m\,b^n$ provient de la multilinéarité du déterminant donc on peut supposer que les polynômes sont unitaires. Pour montrer l'égalité (3.3) nous allons supposer que les α_i et les β_i sont des indéterminées. La proposition 3.3.1 appliquée sur le corps $\mathbb{K} = \mathbb{Q}(\alpha_1,\ldots,\alpha_n,\beta_1,\ldots,\beta_n)$ montre que $\prod_{i,j}(\alpha_i-\beta_j)$ divise $\mathrm{Res}(A,B)$. De plus le degré en α_i des m premières lignes de $\mathrm{Syl}(A,B)$ est 1 et le degré en α_i des n lignes suivantes est 0 de sorte que le degré du résultant en α_i est majoré par m. De la même façon le degré en β_j de $\mathrm{Res}(A,B)$ est borné par n. Par conséquent $\mathrm{Res}(A,B) = c\prod_{i,j}(\alpha_i-\beta_j)$ pour une certaine constante c. Pour déterminer c, il suffit de considérer le cas où $\alpha_i = 0$ pour tout $i = 1,\ldots,n$. Dans ce cas la matrice de Sylvester est triangulaire et son déterminant vaut $B(0)^n$ ce qui implique $c = 1$. Nous avons donc prouvé (3.3). Les égalités suivantes sont des conséquences directes de (3.3). □

Corollaire 3.3.1. *Soient A, B, $C \in \mathbb{A}[X]$. Alors on a* $\mathrm{Res}(A\,B,C) = \mathrm{Res}(A,C)\,\mathrm{Res}(B,C)$.

Démonstration. Le résultat est une conséquence de la formule de Poisson que l'on peut appliquer dans une clôture algébrique du corps des fractions de l'anneau intègre $\mathbb{A}$ et du fait que le résultant est un élément de $\mathbb{A}$. □

Proposition 3.3.2. *Soient A, $B \in \mathbb{A}[X]$. Il existe $(U,V) \in \mathbb{A}[X]^2$ tel que*

$$U\,A + V\,B = \mathrm{Res}(A,B), \quad \deg(U) < \deg(B),\ \deg(V) < \deg(A).$$

Démonstration. Notons C_j, $j = 1,\ldots,n+m$, les colonnes de la matrice de Sylvester $\mathrm{Syl}(A,B)$, $n = \deg(A)$ et $m = \deg(B)$. En ajoutant à la dernière colonne C_{n+m} de $\mathrm{Syl}(A,B)$, l'élément $X^{n+m}\sum_{i=1}^{n+m}C_j\,X^{-j}$, la dernière colonne de la matrice devient

$$\begin{pmatrix} X^{m-1}\,A(X) & X^{m-2}\,A(X) & \ldots & A(X) & X^{n-1}\,B(X) & X^{n-2}\,B(X) & \ldots & B(X)\end{pmatrix}^T.$$

En développant le déterminant par rapport à cette colonne, on obtient donc

$$\begin{aligned}\mathrm{Res}(A,B) = {} & A(X)\,\left(X^{m-1}\,M_{1,n+m} + X^{m-2}\,M_{2,n+m} + \cdots + M_{m,n+m}\right)\\ & + B(X)\,\left(X^{n-1}\,M_{m+1,n+m} + X^{m-2}\,M_{m+2,n+m} + \cdots + M_{m+n,n+m}\right),\end{aligned}$$

où $M_{i,n+m} \in \mathbb{A}$ désigne le déterminant de la co-matrice de $\mathrm{Syl}(A,B)$ obtenue en supprimant la $i$ème ligne et la dernière colonne. D'où le résultat. □

Corollaire 3.3.2. *Soient A, $B \in \mathbb{A}[X]$. Alors $\mathrm{Res}(A,B) = 0$ si et seulement si il existe $(U,V) \neq (0,0) \in \mathbb{A}[X]^2$ tel que*

$$U\,A + V\,B = 0, \quad \deg(U) < \deg(B),\ \deg(V) < \deg(A).$$

Démonstration. Supposons $\mathrm{Res}(A,B) = 0$, alors d'après la proposition 3.3.2, il existe un couple $(U,V) \in \mathbb{A}[X]^2$ tel que $U\,A + V\,B = 0$ avec $\deg(U) < \deg(B) = m$ et $\deg(V) < \deg(A) = n$. Vérifions que $U \neq 0$ ou $V \neq 0$. En notant $U = \sum_{i=0}^{m-1} u_i\,X^i$ et $V = \sum_{i=0}^{n-1} v_i\,X^i$, l'égalité $U\,A + V\,B = 0$ s'écrit $\mathrm{Syl}(A,B)^T\,w = 0$ où $w = (u_{m-1} \ldots u_0\ v_{n-1} \ldots v_0)^T \in \mathbb{A}^{n+m}$. Or $\det(\mathrm{Syl}(A,B)^T) = 0$ donc le système linéaire homogène $\mathrm{sSyl}(A,B)^T\,w = 0$ admet une solution non nulle. Inversement s'il existe $(U,V) \neq (0,0) \in \mathbb{A}[X]^2$ tel que $U\,A + V\,B = 0$ avec $\deg(U) < \deg(B) = m$ et $\deg(V) < \deg(A) = n$, alors le système linéaire homogène de matrice $\mathrm{Syl}(A,B)^T$ admet une solution non nulle ce qui implique que son déterminant est nul. □

3.3.3 Calcul du résultant

Étant donnés deux polynômes A, $B \in \mathbb{K}[X]$ à coefficients dans un corps effectif $\mathbb{K}$, posons nous le problème du calcul du résultant de A et B. En utilisant la définition, calculer le résultant revient à calculer le déterminant de la matrice de Sylvester. Le calcul du déterminant d'une matrice pouvant se faire en la même complexité que celui du produit matriciel (Théorème 2.3.1), les résultats de la sous-section 2.3 montrent que le résultant de A et B peut se calculer en $\tilde{O}(n^{2.81})$ opérations dans $\mathbb{K}$ en notant $n = \max(\deg(A), \deg(B))$. Notons qu'il existe un algorithme plus efficace (de complexité quadratique) qui calcule le résultant en $O(\deg(A)\,\deg(B))$ opérations dans $\mathbb{K}$. Nous ne détaillerons pas cet algorithme dans ce cours mais il est basé sur la division euclidienne et le résultat suivant qui découle de la formule de Poisson (Théorème 3.3.1) : soit $A = Q\,B + R$ la division euclidienne de A de degré n par B de degré m. Alors en notant b_m le coefficient dominant de B et $r = \deg(R)$, on a $\mathrm{Res}(A,B) = (-1)^{n\,m}\,b_m^{n-r}\,\mathrm{Res}(B,R)$.

3.3.4 Résultant en plusieurs variables et élimination

L'une des principales applications du résultant en calcul formel est le fait que cela peut permettre d'éliminer des variables dans l'étude d'un système polynomial. En effet, par définition le résultant $\mathrm{Res}_X(A,B)$ de deux polynômes de $\mathbb{A}[X]$ est un élément de $\mathbb{A}$. Donc si on prend le cas particulier où l'anneau $\mathbb{A}$ lui même est un anneau de polynômes, par exemple $\mathbb{A} = \mathbb{K}[Y]$ pour un certain corps $\mathbb{K}$, alors le résultant $\mathrm{Res}_X(A,B)$ de deux polynômes de $\mathbb{K}[X,Y] = \mathbb{K}[Y][X] = \mathbb{A}[X]$ est un élément de $\mathbb{A} = \mathbb{K}[Y]$ et nous avons donc *éliminé* la variable X. Plus généralement, étant donné un système polynomial défini par des polynômes de plusieurs variables, on peut alors calculer des résultants pour éliminer successivement des variables et ramener l'étude de ce système à l'étude de polynômes à une seule variable sur

lesquels on sait faire beaucoup plus de choses. Ce point ne sera pas détaillé ici mais nous illustrerons simplement cet aspect avec le résultat suivant dans le cas de deux polynômes de deux variables.

Théorème 3.3.2. *Soient $A,\ B \in \mathbb{K}[X,Y] = \mathbb{K}[X][Y]$ deux polynômes bivariés à coefficients dans un corps $\mathbb{K}$ algébriquement clos. Si on note $A = \sum_{i=0}^{n} a_i(X)\,Y^i$ et $B = \sum_{i=0}^{m} b_i(X)\,Y^i$, alors les racines du résultant $\mathrm{Res}_Y(A,B) \in \mathbb{K}[X]$ sont les abscisses des solutions du système polynomial $\{A(X,Y) = 0,\ B(X,Y) = 0\}$ et les racines des termes de têtes $a_n(X)$ et $b_m(X)$.*

Démonstration. Exercice à partir du théorème 3.3.1 et de la proposition 3.3.2. □

Terminons ce chapitre en illustrant ce résultat sur un exemple très simple.

Exemple 3.3.2. *Soient $A = X^2\,Y + X + 1$ et $B = X\,Y - 1$. On trouve alors le résultant $\mathrm{Res}_Y(A,B) = -X\,(2\,X + 1)$. On retrouve donc la racine $X = 0$ des termes de tête et la racine $X = -1/2$ qui correspond à l'abscisse de la solution $(-1/2, -2)$ du système polynomial $\{X^2\,Y + X + 1 = 0,\ X\,Y - 1 = 0\}$.*

Chapitre 4

Factorisation de polynômes à une variable

Dans ce chapitre, on note $\mathbb{K}$ un corps effectif. On considère un polynôme $P \in \mathbb{K}[X]$ unitaire d'une seule variable X. On sait que P s'écrit de manière unique (à permutation près) sous la forme

$$P = F_1^{n_1} \times \cdots \times F_k^{n_k}, \tag{4.1}$$

avec pour $i = 1, \ldots, k$, F_i irréductible, $n_i \in \mathbb{N}^*$ et les F_i deux à deux distincts. Le problème de la factorisation s'énonce alors de la manière suivante : étant donné P, calculer les F_i et les n_i pour $i = 1, \ldots, k$.

Deux cas particuliers importants sont celui où le corps $\mathbb{K}$ est un corps fini de caractéristique $p > 0$ et celui où $\mathbb{K} = \mathbb{Q}$ est le corps des nombres rationnels de caractéristique zéro (ou, de façon analogue, le cas d'un polynôme à coefficients dans $\mathbb{Z}$). Le cas de la factorisation sur un corps fini a déjà été traité dans le cours *Corps finis* donc nous nous concentrerons ici sur le cas de la factorisation sur un corps de caractéristique nulle. Les algorithmes procèdent en général en plusieurs étapes la première étape consistant à calculer une factorisation dite *sans carré* du polynôme.

4.1 Factorisation sans carré

4.1.1 Définition et existence

Définition 4.1.1. *Un polynôme $P \in \mathbb{K}[X]$ unitaire est dit* sans carré *(ou* sans facteur multiple*) s'il n'existe pas de polynôme $F \in \mathbb{K}[X] \setminus \mathbb{K}$ tel que F^2 divise P.*

Définition 4.1.2. *Soit $P \in \mathbb{K}[X]$ unitaire. La* factorisation sans carré *de P est donnée par*

$$P = P_1 \, P_2^2 \cdots P_s^s, \tag{4.2}$$

où pour tout $i = 1, \ldots, s$, P_i est sans carré et les P_i sont premiers entre eux deux à deux. La partie sans carré *de P est alors définie par le produit $P_1 \cdots P_s$.*

Notons que dans la factorisation sans carré (4.2), les P_i ne sont pas nécessairement irréductibles et certains P_i peuvent être égaux à 1.

Exemple 4.1.1. *Soit* $P = X^8 - 11\,X^6 + 36\,X^4 - 16\,X^2 - 64 \in \mathbb{Q}[X]$. *La factorisation sans carré de* P *s'écrit alors*

$$P = P_1\,P_2^2\,P_3^3, \quad P_1 = X^2 + 1,\ P_2 = 1,\ P_3 = (X+2)\,(X-2).$$

Proposition 4.1.1. *Tout polynôme* $P \in \mathbb{K}[X]$ *unitaire admet une et une seule factorisation sans carré.*

Démonstration. On sait que P s'écrit de manière unique sous la forme $P = F_1^{n_1} \cdots F_k^{n_k}$, avec pour $i = 1, \ldots, k$, F_i irréductible, $n_i \in \mathbb{N}^*$ et les F_i deux à deux distincts. On peut alors écrire $P = P_1\,P_2^2 \cdots P_s^s$ avec $s = \max_{1 \le i \le k} n_i$ et pour $j = 1, \ldots, s$,

$$P_j = \begin{cases} 1 & \text{si} \quad j \notin \{n_1, \ldots, n_k\}, \\ \prod_{n_i = j} F_i & \text{si} \quad j \in \{n_1, \ldots, n_k\}. \end{cases}$$

L'unicité découle de l'unicité des F_i (à permutation près). □

4.1.2 Calcul de la factorisation sans carré

Intéressons nous maintenant au calcul de factorisation sans carré dans le cas où le corps $\mathbb{K}$ est de caractéristique zéro, e.g., $\mathbb{K} = \mathbb{Q}$. Un avantage de la caractéristique nulle est que pour $P \in \mathbb{K}[X]$, on a équivalence entre le fait que le polynôme dérivé[1] P' soit égal à zéro est le fait que le polynôme P soit constant, i.e., $P \in \mathbb{K} : P' = 0 \Leftrightarrow P \in \mathbb{K}$.

Proposition 4.1.2. *Soient* $\mathbb{K}$ *un corps de caractéristique zéro et* $P \in \mathbb{K}[X]$ *un polynôme unitaire. Alors* P *est sans carré si et seulement si* $\text{PGCD}(P, P') = 1$, *i.e., le polynôme* P *et son polynôme dérivée* P' *sont premiers entre eux.*

Démonstration. Supposons tout d'abord que P est sans carré et que $G = \text{PGCD}(P, P') \neq 1$. Comme P est supposé sans carré on a $P = P_1 \cdots P_s$ avec les P_i irréductibles, deux à deux distincts et $P_i \notin \mathbb{K}$ car P unitaire. En dérivant, on obtient alors

$$P' = P_1'\,P_2 \cdots P_s + \cdots + P_1 \cdots P_{s-1}\,P_s'.$$

Comme $G \neq 1$, il existe $j \in \{1, \ldots, s\}$ tel que P_j divise P' et donc P_j divise $(\prod_{i \neq j} P_i)\,P_j'$. Or $\text{PGCD}(P_j, P_j') = 1$ et $\text{PGCD}(P_j, P_i) = 1$ pour $i \neq j$ car les P_j sont irréductibles et deux à deux distincts d'où la contradiction. Inversement supposons qu'il existe $F \in \mathbb{K}[X] \setminus \mathbb{K}$ tel que F^2 divise P. On alors $P = F^2\,H$ pour un certain polynôme $H \in \mathbb{K}[X]$. En dérivant on obtient alors $P' = 2\,F'\,F\,H + F^2\,H' = F\,(2\,F'\,H + F\,H')$. On a donc F divise P et F divise P' d'où F divise $\text{PGCD}(P, P')$ et donc $\text{PGCD}(P, P') \neq 1$ puisque $F \notin \mathbb{K}$ ce qui conclut la preuve. □

[1] Si $P = \sum_{i=0}^{n} p_i\,X^i$, avec $p_i \in \mathbb{K}$, alors $P' = \sum_{i=1}^{n} i\,p_i X^{i-1}$.

Proposition 4.1.3. *Soit $\mathbb{K}$ un corps de caractéristique nulle. Soient $P_1, \ldots, P_s \in \mathbb{K}[X] \setminus \mathbb{K}$ des polynômes sans carré, $P = \prod_{i=1}^{s} P_i$ et $(c_1, \ldots, c_s) \in \mathbb{K}^s$. Alors si les P_i sont premiers entre eux deux à deux, on a*

$$\text{PGCD}\left(P, \sum_{i=1}^{s} c_i \frac{P}{P_i} P_i'\right) = \prod_{i \text{ t.q. } c_i = 0} P_i.$$

Démonstration. On a

$$\text{PGCD}\left(P, \sum_{i=1}^{s} c_i \frac{P}{P_i} P_i'\right) = \prod_{j=1}^{s} \text{PGCD}\left(P_j, \sum_{i=1}^{s} c_i \frac{P}{P_i} P_i'\right) = \prod_{j=1}^{s} \text{PGCD}\left(P_j, c_j \frac{P}{P_j} P_j'\right).$$

D'où le résultat puisque le dernier PGCD vaut 1 si $c_j \neq 0$ et P_j sinon. □

À partir de ces résultats, nous pouvons procéder comme suit pour calculer la factorisation sans carré (4.2) d'un polynôme $P \in \mathbb{K}[X]$ unitaire. En dérivant la relation (4.2), on obtient

$$P' = P_1' P_2^2 \cdots P_s^s + P_1 (2 P_2 P_2') P_3^3 \cdots P_s^s + \cdots + P_1 P_2^2 \cdots P_{s-1}^{s-1} (s P_s' P_s^{s-1}).$$

On a alors $G = \text{PGCD}(P, P') = P_2 P_3^2 \cdots P_s^{s-1}$ et le polynôme $H = P/G = P_1 \cdots P_s$ est un polynôme sans carré. On en déduit que le polynôme P_1 cherché est $P_1 = H/\text{PGCD}(H, G)$ puisque $\text{PGCD}(H, G) = P_2 \cdots P_s$. On applique ensuite la même méthode à G pour déterminer P_2 et ainsi de suite. On obtient alors l'algorithme 13 parfois appelé *algorithme de Musser* pour le calcul de la factorisation sans carré d'un polynôme sur un corps de caractéristique nulle.

Algorithme 13 *Factorisation sans carré d'un polynôme sur un corps de caractéristique* 0

Entrées : $P \in \mathbb{K}[X]$ unitaire où $\mathbb{K}$ est un corps de caractéristique zéro.
Sorties : la liste $(P_1, \ldots, P_s)$ telle que (4.2) soit la factorisation sans carré de P.

- Poser $i = 1$;
- Calculer $G_1 = \text{PGCD}(P, P')$ et $H_1 = \frac{P}{G_1}$;
Tant que $\deg(G_i) \geq 1$ **Faire**
  - Calculer $G_{i+1} = \text{PGCD}(G_i, G_i')$ et $H_{i+1} = \frac{G_i}{G_{i+1}}$;
  - Calculer $P_i = \frac{H_i}{H_{i+1}}$;
  - $i = i + 1$;
Fin tant que
- Poser $P_i = G_{i-1}$;
- Retourner $(P_1, \ldots, P_i)$.

Proposition 4.1.4. *Soit* **M** *une fonction telle que le produit de deux polynômes de degré au plus n puisse se calculer en $O(\mathbf{M}(n))$ opérations arithmétiques. Alors l'algorithme 13 est correct et permet de calculer la factorisation sans carré d'un polynôme de degré n à coefficients dans un corps de caractéristique zéro en $O(n\,\mathbf{M}(n)\log(n))$ opérations dans $\mathbb{K}$.*

Démonstration. La correction de l'algorithme est laissée à titre d'exercice. Elle se déduit des explications données ci-dessus. Le nombre d'étape de l'algorithme est majoré par n. Sa complexité est donc majorée par n calculs de PGCD de polynômes de degré majoré par n d'où le résultat d'après la remarque 3.2.1. □

Illustrons maintenant le déroulement de l'algorithme 13 sur un exemple.

Exemple 4.1.2. *Reprenons le polynôme P considéré à l'exemple 4.1.1. L'algorithme 13 se déroule alors de la manière suivante :*

1. $i = 1$, $G_1 = \text{PGCD}(P, P') = X^4 - 8\,X^2 + 16$, $H_1 = P/G_1 = X^4 - 3\,X^2 - 4$;
2. $G_2 = \text{PGCD}(G_1, G_1') = X^2 - 4$, $H_2 = G_1/G_2 = X^2 - 4$, $P_1 = H_1/H_2 = X^2 + 1$, $i = 2$;
3. $G_3 = \text{PGCD}(G_2, G_2') = 1$, $H_3 = G_2/G_3 = X^2 - 4$, $P_2 = H_2/H_3 = 1$, $i = 3$;
4. $P_3 = G_2 = X^2 - 4$.

On retrouve bien la factorisation sans carré

$$P = P_1\,P_2^2\,P_3^3 = (X^2+1)\,(X^2-4)^3 = (X^2+1)\,((X-2)\,(X+2))^3.$$

Nous donnons maintenant un autre algorithme 14 appelé *algorithme de Yun* qui permet de supprimer le facteur n dans la complexité, i.e., d'obtenir une complexité en $O(\mathbf{M}(n)\log(n))$ opérations arithmétiques dans $\mathbb{K}$.

Algorithme 14 *Algorithme de Yun pour la factorisation sans carré en caractéristique* 0

ENTRÉES : $P \in \mathbb{K}[X]$ unitaire où $\mathbb{K}$ est un corps de caractéristique zéro.
SORTIES : la liste $(P_1, \ldots, P_s)$ telle que (4.2) est la factorisation sans carré de P.

- Poser $i = 1$;
- Calculer $G = \text{PGCD}(P, P')$, $H = \frac{P}{G}$ et $K = \frac{P'}{G}$;

Tant que $\deg(H) \geq 1$ **Faire**

- Calculer $R = \text{PGCD}(H, K - H')$, $K = \frac{(K-H')}{R}$ et $H = \frac{H}{R}$;
- Poser $P_i = R$;
- $i = i + 1$;

Fin tant que

- Retourner $(P_1, \ldots, P_i)$.

Proposition 4.1.5. *Soit $\mathbf{M}$ une fonction telle que le produit de deux polynômes de degré au plus n puisse se calculer en $O(\mathbf{M}(n))$ opérations arithmétiques avec $\mathbf{M}(n+n') \geq \mathbf{M}(n) + \mathbf{M}(n')$. Alors l'algorithme 14 est correct et permet de calculer la factorisation sans carré d'un polynôme de degré n à coefficients dans un corps de caractéristique zéro en $O(\mathbf{M}(n)\log(n))$ opérations dans $\mathbb{K}$.*

Démonstration. Montrons tout d'abord que l'algorithme est correct. Notons H_i et K_i les polynômes H et K au départ du $i$ème passage dans la boucle **Tant que** de l'algorithme et R_i le polynôme R calculé à l'étape i. Soit m la valeur finale de i lorsqu'on sort de la boucle **Tant que** et soient H_m et K_m les polynômes H et K obtenus à la sortie de la boucle. On montre par récurrence que pour tout $i = 1, \dots, m$, on a alors :

$$H_i = \prod_{j=i}^{s} P_j, \quad K_i = \sum_{j=i}^{s} (j-i+1) \frac{H_i}{P_j} P_j',$$

où les P_i sont les polynômes cherchés vérifiant (4.2). Pour $i = 1$, nous avons $H_1 = P/G$ et $K_1 = P'/G$. Comme $G = \text{PGCD}(P, P') = \prod_{i=1}^{s} P_i^{i-1}$, la propriété est vérifiée pour $i = 1$. Supposons la propriété vraie pour $i \in \{1, \dots, m-1\}$. On a alors, d'après la proposition 4.1.3

$$R_i = \text{PGCD}(H_i, K_i - H_i') = \text{PGCD}\left(\prod_{j=i}^{s} P_j, \sum_{j=i}^{s} (j-i) \frac{H_i}{P_j} P_j'\right) = \prod_{j=i} P_j = P_i,$$

d'où les formules pour K_{i+1} et H_{i+1}. L'entier m est le premier entier pour lequel $\deg(H_m) = 0$ donc l'entier s de la factorisation sans carré (4.2) vaut $m-1$ et on retourne donc bien tous les facteurs sans carrés $P_1, \dots, P_s$. Estimons maintenant la complexité de l'algorithme. L'étape 2 nécessite $O(\mathbf{M}(n)\log(n))$ opérations dans $\mathbb{K}$. Ensuite, l'étape i de la boucle **Tant que** nécessite $O(\mathbf{M}(\deg(H_i))\log(\deg(H_i)))$ opérations dans $\mathbb{K}$ car $\deg(K_i) \leq \deg(H_i) - 1$. Le coût total de la boucle **Tant que** est alors majoré par $O(\sum_{i=1}^{m-1} \mathbf{M}(\deg(H_i))\log(\deg(H_i))) \leq O(\mathbf{M}(\sum_{i=1}^{m-1} \deg(H_i))\log(n))$ en utilisant l'inégalité $\mathbf{M}(n) + \mathbf{M}(n') \leq \mathbf{M}(n+n')$ d'où le résultat puisque $\prod_{i=1}^{m-1} H_i = P$. □

Exemple 4.1.3. *Reprenons le polynôme P considéré à l'exemple 4.1.1. L'algorithme 14 se déroule alors de la manière suivante :*

1. $i = 1$, $G = \text{PGCD}(P, P') = X^4 - 8X^2 + 16$, $H = P/G = X^4 - 3X^2 - 4$, $K = P'/G = 2X(4X^2 - 1)$;
2. $R = \text{PGCD}(H, K - H') = X^2 + 1$, $K = (K - H')/R = 4X$, $H = H/R = X^2 - 4$, $P_1 = R = X^2 + 1$, $i = 2$;
3. $R = \text{PGCD}(H, K - H') = 1$, $K = (K - H')/R = 2X$, $H = H/R = X^2 - 4$, $P_2 = R = 1$, $i = 3$;
4. $R = \text{PGCD}(H, K - H') = X^2 - 4$, $K = (K - H')/R = 0$, $H = H/R = 1$, $P_3 = R = X^2 - 4$, $i = 4$.

On retrouve bien la factorisation sans carré

$$P = P_1 P_2^2 P_3^3 = (X^2 + 1)(X^2 - 4)^3 = (X^2 + 1)((X - 2)(X + 2))^3.$$

Lorsque le corps $\mathbb{K}$ est un corps fini de caractéristique $p > 0$, alors une difficulté supplémentaire est que si la caractéristique p est plus petite que le degré d'un polynôme $P \in \mathbb{K}[X]$, alors P' peut être nul sans que P soit nécessairement constant. Dans ce cas, la bonne notion n'est alors pas la factorisation sans carré mais la factorisation dite *séparable* que nous n'aborderons pas dans ce cours. Notons que la factorisation séparable d'un polynôme de degré n peut aussi se calculer en $O(\mathbf{M}(n)\log(n))$ opérations arithmétiques.

4.2 Factorisation sur $\mathbb{Z}$

4.2.1 Rappels : factorisation sur un corps fini

La factorisation d'un polynôme univarié à coefficients dans un corps fini $\mathbb{F}_q$ a été étudiée dans le cours *Corps finis*. Les deux principaux algorithmes de factorisation dans ce cas sont celui de Berlekamp et celui de Cantor et Zassenhaus. Nous résumons les résultats obtenus dans le théorème suivant :

Théorème 4.2.1. *Soit $P \in \mathbb{F}_q[X]$ unitaire à coefficients dans le corps fini à q éléments $\mathbb{F}_q$. Alors, on a les résultats suivants :*

1. *L'algorithme de Berlekamp déterministe (resp. probabiliste) calcule la factorisation (4.1) de P en $O(n^3 + n^2 q)$ (resp. $O(n^3 + n^2 \log(n)\log(q))$) opérations dans $\mathbb{F}_q$;*

2. *L'algorithme (probabiliste) de Cantor et Zassenhaus calcule la factorisation (4.1) de P en $O(n^3 \log(q))$ opérations dans $\mathbb{F}_q$ en utilisant de l'arithmétique classique et en $O(n^2 \log^2(n)\log(\log(n))\log(q))$ opérations dans $\mathbb{F}_q$ en utilisant de l'arithmétique rapide.*

On supposera donc dans la section suivante que l'on sait factoriser un polynôme à coefficient dans un corps fini.

4.2.2 Algorithme utilisant un grand nombre premier

Soit $P \in \mathbb{Z}[X]$ un polynôme primitif. On suppose qu'une factorisation sans carré de P a déjà été calculée (voir la section précédente) de sorte que le problème de factoriser P est ramené au problème de factoriser un polynôme sans carré. On supposera donc ici que le polynôme à factoriser, que l'on note encore P, est primitif et sans carré.

L'existence de la borne de Mignotte (Lemme 3.2.1) sur la taille des coefficients des facteurs d'un polynôme donné nous permet d'utiliser une stratégie modulaire pour calculer la factorisation (4.1) avec $n_i = 1$, $i = 1, \ldots, k$, d'un polynôme primitif et sans carré à coefficients dans $\mathbb{Z}$. En notant B la borne de Mignotte, on peut alors choisir un nombre premier p tel que $p > 2\,B$ factoriser le polynôme P dans $\mathbb{Z}/(p\,\mathbb{Z})$ et essayer, par une recherche exhaustive, de retrouver un facteur de P sur $\mathbb{Z}$. Cette stratégie mène à l'algorithme 15.

Algorithme 15 *Factorisation d'un polynôme sur $\mathbb{Z}$ en utilisant un grand premier*

Entrées : $P \in \mathbb{Z}[X]$ un polynôme sans-carré et primitif de degré n.
Sorties : P (si P est irréductible sur $\mathbb{Z}$) ou un facteur non-trivial de P sur $\mathbb{Z}$.
- Soit $B = \sqrt{n+1}\, 2^n ||P||_\infty$ la borne de Mignotte ;
- Choisir un premier p tel que $p > 2\,B$;
- Factoriser P dans $\mathbb{Z}/(p\,\mathbb{Z})[X]$: $\overline{P} = \overline{F_1} \cdots \overline{F_s}$ avec les $\overline{F_i}$ irréductibles dans $\mathbb{Z}/(p\,\mathbb{Z})[X]$ et à coefficients dans $\{-\lfloor \frac{p}{2} \rfloor, \ldots, \lfloor \frac{p}{2} \rfloor\}$;

Si $s = 1$ **Alors**
- $\overline{P}$ est irréductible dans $\mathbb{Z}/(p\,\mathbb{Z})[X]$ et donc Retourner P qui est irréductible sur $\mathbb{Z}$;

Sinon
Pour tout $\mathcal{I} \subset \{1, \ldots, s\}$ tel que $\mathcal{I} \neq \emptyset$ **Faire**
Si $\prod_{i \in \mathcal{I}} \overline{F_i}$ divise P dans $\mathbb{Z}[X]$ **Alors**
- Retourner $\prod_{i \in \mathcal{I}} \overline{F_i}$ qui est un facteur non trivial de P sur $\mathbb{Z}$;

Finsi
Fin pour
- Retourner P qui est irréductible sur $\mathbb{Z}$.

Finsi

Lorsque l'algorithme 15 renvoie un facteur F de P alors nous pouvons appliquer à nouveau l'algorithme à F et P/F et ainsi de suite jusqu'à ce que tous les facteurs obtenus soient irréductibles auquel cas nous avons calculé la factorisation cherchée de P. Notons que dans le pire cas, nous aurons 2^n ensembles $\mathcal{I}$ à tester ce qui mène à une complexité exponentielle. Cependant, si le nombre s de facteurs dans $\mathbb{Z}/(p\,\mathbb{Z})[X]$ est petit, alors cet algorithme a un coût raisonnable et est utilisable en pratique. Finalement, on remarque qu'un autre inconvénient de cette méthode est que la borne de Mignotte B peut être grande (si n et $||P||_\infty$ sont grands) et dans ce cas, le nombre premier p sera lui aussi grand ce qui impliquera que l'étape de factorisation dans $\mathbb{Z}/(p\,\mathbb{Z})[X]$ sera très couteuse. L'algorithme proposé dans la sous-section suivante évite cet inconvénient en utilisant seulement un petit nombre premier p.

4.2.3 Algorithme utilisant la remontée de Hensel

L'algorithme que nous allons présenter maintenant consiste à choisir un petit nombre premier p convenable, à factoriser P modulo p puis à utiliser la *remontée de Hensel* pour en déduire des factorisations de P modulo p^k. En choisissant alors k de sorte que $p^k > 2\,B$ nous pourrons ainsi calculer des facteurs de P sur $\mathbb{Z}$.

Lemme 4.2.1 (Lemme de Hensel)**.** *Soient $P \in \mathbb{Z}[X]$ et p un nombre premier. On suppose que $P \equiv F_1\, G_1 \mod p$ avec $F_1, G_1 \in \mathbb{Z}/(p\,\mathbb{Z})[X]$ et F_1 et G_1 premiers entre eux dans $\mathbb{Z}/(p\,\mathbb{Z})[X]$. Alors, pour tout $k \in \mathbb{N}^*$, il existe $(F_k, G_k) \in (\mathbb{Z}/(p^k\,\mathbb{Z})[X])^2$ tel que*

$$\begin{cases} P \equiv F_k\, G_k \mod p^k, \\ F_k \equiv F_1 \mod p, \quad G_k \equiv G_1 \mod p. \end{cases}$$

Démonstration. Raisonnons par récurrence sur k. Par hypothèse, l'énoncé est vrai pour $k = 1$. Supposons l'énoncé vrai pour k et cherchons $(F_{k+1}, G_{k+1}) \in (\mathbb{Z}/(p^{k+1}\,\mathbb{Z})[X])^2$ tel que

$$P \equiv F_{k+1}\,G_{k+1} \mod p^{k+1}, \quad F_{k+1} \equiv F_1 \mod p, \quad G_{k+1} \equiv G_1 \mod p.$$

On pose $F_{k+1} = F_k + p^k\,\hat{F}_k$ et $G_{k+1} = G_k + p^k\,\hat{G}_k$, où $\hat{F}_k$ et $\hat{G}_k$ sont deux polynômes à déterminer. On obtient alors

$$F_{k+1}\,G_{k+1} = F_k\,G_k + p^k\,(\hat{F}_k\,G_k + F_k\,\hat{G}_k) + p^{2\,k}\,\hat{F}_k\,\hat{G}_k.$$

Donc $P \equiv F_{k+1}\,G_{k+1} \mod p^{k+1}$ équivaut à $P \equiv F_k\,G_k + p^k\,(\hat{F}_k\,G_k + F_k\,\hat{G}_k) \mod p^{k+1}$. Or, par hypothèse de récurrence, on a $P = F_k\,G_k + p^k\,\hat{P}_k$ pour un certain polynôme $\hat{P}_k$, d'où

$$p^k\,\hat{P}_k \equiv p^k\,(\hat{F}_k\,G_k + F_k\,\hat{G}_k) \mod p^{k+1},$$

et donc

$$\hat{P}_k \equiv \hat{F}_k\,G_k + F_k\,\hat{G}_k \mod p.$$

Par hypothèse de récurrence, nous avons aussi $F_k \equiv F_1 \mod p$ et $G_k \equiv G_1 \mod p$ ce qui implique alors

$$\hat{P}_k \equiv \hat{F}_k\,G_1 + F_1\,\hat{G}_k \mod p. \tag{4.3}$$

Comme F_1 et G_1 sont supposés premiers entre eux dans $\mathbb{Z}/(p\,\mathbb{Z})[X]$, on peut toujours calculer $(\hat{F}_k, \hat{G}_k)$ vérifiant (4.3) (Cf Chapitre 3) ce qui conclut la preuve. □

Notons que la preuve ci-dessus donne une méthode effective pour effectuer la remontée de Hensel en utilisant l'algorithme d'Euclide étendu (Cf Chapitre 3). Illustrons ceci sur un exemple :

Exemple 4.2.1. *Considérons le polynôme $P = X^3 + 8\,X - 7$, le nombre premier $p = 7$ et utilisons les notations de la preuve précédente. On a $P \equiv F_1\,G_1 \mod 7$ avec $F_1 = X$ et $G_1 = X^2 + 1$ premiers entre eux dans $\mathbb{Z}/(7\,\mathbb{Z})$ et $P = F_1\,G_1 + 7\,(X - 1)$ de sorte que $\hat{P}_1 = X - 1$. Calculons alors F_2 et G_2 tels que $P \equiv F_2\,G_2 \mod 7^2$, $F_2 \equiv F_1 \mod 7$, $G_2 \equiv G_1 \mod 7$. D'après la preuve précédente, on peut prendre $F_2 = F_1 + 7\,\hat{F}_1$ et $G_2 = G_1 + 7\,\hat{G}_1$ où $\hat{F}_1$ et $\hat{G}_1$ sont tels que $\hat{P}_1 \equiv \hat{F}_1\,G_1 + F_1\,\hat{G}_1 \mod 7$. En utilisant l'algorithme d'Euclide étendu, on calcule U et V dans $\mathbb{Z}/(7\,\mathbb{Z})$ tels que $1 \equiv U\,F_1 + V\,G_1 \mod 7$. On trouve ici $U = 6\,X$ et $V = 1$. On pose ensuite $\hat{F}_1 = \hat{P}_1\,V = X - 1$ et $\hat{F}_1 = \hat{P}_1\,U = 6\,X\,(X - 1)$ et sorte que $F_2 = F_1 + 7\,\hat{F}_1 = 8\,X - 7$ et $G_2 = G_1 + 7\,\hat{G}_1 = 43\,X^2 - 42\,X + 1$. On peut alors vérifier que l'on a $P \equiv F_2\,G_2 \mod 7^2$, $F_2 \equiv F_1 \mod 7$, $G_2 \equiv G_1 \mod 7$.*

Dans cette stratégie nous allons devoir choisir un petit nombre premier p. Pour s'assurer que notre polynôme sans carré à coefficients dans $\mathbb{Z}$ reste sans carré dans $\mathbb{Z}/(p\,\mathbb{Z})$, il nous suffit de supposer que p ne divise pas le *discrimant* de P, i.e., que P ne divise pas le résultant $\mathrm{Res}(P, P')$. On obtient l'algorithme 16.

Lorsque l'algorithme 16 renvoie un facteur G_k de P alors nous pouvons appliquer à nouveau l'algorithme à G_k et P/G_k et ainsi de suite jusqu'à ce que tous les facteurs obtenus

Algorithme 16 *Factorisation d'un polynôme sur $\mathbb{Z}$ en utilisant la remontée de Hensel*

Entrées : $P \in \mathbb{Z}[X]$ un polynôme sans-carré et primitif de degré n.
Sorties : P (si P est irréductible sur $\mathbb{Z}$) ou un facteur non-trivial de P sur $\mathbb{Z}$.

- Soit p un « petit » nombre premier ne divisant pas $\mathrm{Res}(P, P')$;
- Factoriser P dans $\mathbb{Z}/(p\,\mathbb{Z})[X]$: $\overline{P} \equiv \overline{F_1} \cdots \overline{F_s} \mod p$;

Si $s = 1$ **Alors**

- $\overline{P}$ est irréductible dans $\mathbb{Z}/(p\,\mathbb{Z})[X]$ et donc Retourner P qui est irréductible sur $\mathbb{Z}$;

Sinon

Pour tout $\mathcal{I} \subset \{1, \ldots, s\}$ tel que $\mathcal{I} \neq \emptyset$ **Faire**

- Poser $G_1 = \prod_{i \in \mathcal{I}} \overline{F_i}$ et $H_1 = \prod_{i \notin \mathcal{I}} \overline{F_i}$;
- Utiliser le lemme de Hensel pour calculer G_k et H_k à coefficients dans $\mathbb{Z}$ tels que $P \equiv G_k\, H_k \mod p^k$ avec $p^k > 2\,B$;

Si G_k divise P dans $\mathbb{Z}[X]$ **Alors**

- Retourner G_k qui est un facteur non trivial de P sur $\mathbb{Z}$;

Finsi

Fin pour

- Retourner P qui est irréductible sur $\mathbb{Z}$.

Finsi

soient irréductibles auquel cas nous avons calculé la factorisation cherchée de P. Comme celle de l'algorithme précédent, la complexité de cet algorithme est exponentielle à cause de la combinatoire, i.e., du nombre de sous-ensembles de $\{1, \ldots, s\}$ et du fait que dans le pire des cas on doit tester de manière exhaustive tous les sous-ensembles. Notons qu'il existe des algorithmes de complexité polynomiale pour factoriser des polynômes sur $\mathbb{Z}$ où cette recherche exhaustive est remplacée par la résolution d'un problème de type *sac à dos* en utilisant l'*algorithme LLL* pour la réduction des réseaux.

Chapitre 5

Intégration symbolique

Dans ce chapitre nous allons aborder le problème de l'intégration symbolique « sous forme close », i.e., étant donnée une expression rationnelle formée de fonctions obtenues par composition d'exponentielles et de logarithmes, décider si elle admet une primitive de la même nature et si oui la calculer. Par exemple, les algorithmes que nous allons voir permettent de montrer que la fonction $x \mapsto \exp(x^2)$ n'admet pas de primitive s'écrivant comme expression rationnelle formée de fonctions obtenues par composition d'exponentielles et de logarithmes. Par contre la fonction

$$\frac{x\,\left((x^2\,\exp(2\,x^2)-\ln^2(x+1))^2+2\,x\,\exp(3\,x^2)\,(x-(2\,x^2+1)\,(x+1)\,\ln(x+1))\right)}{(x+1)\,(\ln^2(x+1)-x^2\,\exp(2\,x^2))^2}$$

admet pour primitive la fonction

$$x-\ln(x+1)+\frac{x\,\exp(x^2)\,\ln(x+1)}{x^2\,\exp(2\,x^2)-\ln^2(x+1)}+\frac{1}{2}\,\ln\left(\ln(x+1)+x\,\exp(x^2)\right)-\frac{1}{2}\,\ln\left(\ln(x+1)-x\,\exp(x^2)\right).$$

Le chapitre est organisé comme suit. Dans un premier temps nous poserons brièvement un cadre algébrique pour l'étude de ce problème. Ensuite nous aborderons en détails le cas particulier où la fonction dont on cherche une primitive est une fraction rationnelle. Enfin, nous donnerons l'idée de l'algorithme traitant le cas général en se ramenant au cas des fractions rationnelles.

5.1 Algèbre différentielle et fonctions élémentaires

Définition 5.1.1. *Soit* $\mathbb{A}$ *un anneau. Une* dérivation de $\mathbb{A}$ *est une application* $\delta : \mathbb{A} \to \mathbb{A}$ *vérifiant :*

$$\forall f,\, g \in \mathbb{A}, \quad \delta(f+g) = \delta(f) + \delta(g), \quad \delta(f\,g) = \delta(f)\,g + f\,\delta(g).$$

Un anneau $\mathbb{A}$ *muni d'une dérivation* δ *est appelé* anneau différentiel *et noté* $(\mathbb{A}, \delta)$. *Si* $\mathbb{A}$ *est un corps, alors on parle de* corps différentiel.

L'exemple classique est le corps $\mathbb{C}(x)$ des fractions rationnelles à coefficients dans $\mathbb{C}$ muni de la dérivation usuelle $\delta = d/dx$. La fonction exponentielle $x \mapsto \exp(x)$ est souvent définie comme *une* fonction f vérifiant $f' = f$. Cette dernière relation permet alors d'obtenir récursivement toutes les dérivées de f. Pour modéliser et généraliser cette situation, nous introduisons la notion d'extension différentielle :

Définition 5.1.2. *Soit $(\mathbb{K}, \delta)$ un corps différentiel. On dit que $(\mathbb{K}_1, \delta_1)$ est une* extension différentielle de $(\mathbb{K}, \delta)$ *si $\mathbb{K} \subset \mathbb{K}_1$ et si la restriction de δ_1 à $\mathbb{K}$ coïncide avec δ.*

Par exemple si on prend $(\mathbb{K}, \delta) = (\mathbb{C}(x), d/dx)$, alors l'extension différentielle $(\mathbb{K}_1, \delta_1)$ avec $\mathbb{K}_1 = \mathbb{K}(X)$ et δ_1 définie par $\delta_1(X) = X$ modélise le corps différentiel $\mathbb{C}(x, \exp(x))$ muni de la dérivation usuelle. Dans la suite, par abus de notation, nous noterons souvent de la même façon, e.g., nous utiliserons le symbole $'$, la dérivation δ sur $\mathbb{K}$ et δ_1 sur $\mathbb{K}_1$.

Pour modéliser une expression rationnelle formée de fonctions obtenues par composition d'exponentielles et de logarithmes, nous utilisons le formalisme suivant :

Définition 5.1.3. *Soit $(\mathbb{K},')$ un corps différentiel. On dit que $(\mathbb{K}_1,')$ est une* extension élémentaire de $(\mathbb{K},')$ *si $\mathbb{K}_1 = \mathbb{K}(t_1, \ldots, t_n)$ où, pour $i = 1, \ldots, n$, t_i vérifie l'une des trois conditions suivantes :*

1. *t_i algébrique sur $\mathbb{K}(t_1, \ldots, t_{i-1})$, i.e., il existe $P \in \mathbb{K}(t_1, \ldots, t_{i-1})[X]$ tel que $P(t_i) = 0$;*
2. *il existe $a_i \in \mathbb{K}(t_1, \ldots, t_{i-1})$ tel que $t_i' = a_i'/a_i$. Le symbole t_i modélise alors $\ln(a_i)$, le logarithme de a_i ;*
3. *il existe $a_i \in \mathbb{K}(t_1, \ldots, t_{i-1})$ tel que $t_i' = a_i' t_i$. Le symbole t_i modélise alors $\exp(a_i)$, l'exponentielle de a_i.*

Dans la suite nous considérerons uniquement le cas où $(\mathbb{K},') = (\mathbb{C}(x), d/dx)$ et nous appellerons *fonction élémentaire* toute fonction appartenant à une extension élémentaire de $(\mathbb{C}(x), d/dx)$.

Exemple 5.1.1. *Avec le formalisme de la définition 5.1.3, pour construire la fonction élémentaire $\exp(x^2)$ nous introduisons t_1 tel que $t_1' = 2\,x\,t_1$ et $\exp(x^2)$ est alors représenté par t_1. Pour construire $\ln(\ln(x))$, nous introduisons t_1 tel que $t_1' = 1/x$ puis t_2 tel que $t_2' = t_1'/t_1 = 1/(x\,t_1)$ et le représentant de $\ln(\ln(x))$ est alors t_2. De la même manière, pour introduire la fonction élémentaire $\ln(1+\exp(\cos(x)))$, nous procédons comme suit. On introduit tout d'abord t_1 tel que $t_1' = i\,t_1$ (t_1 modélise alors $\exp(i\,x)$), puis on introduit t_2 tel que $t_2' = ((t_1^2+1)/(2\,t_1))'\,t_2$ (t_2 modélise alors $\exp(\cos(x))$) et finalement t_3 tel que $t_3' = t_2'/(1+t_2)$ qui représente $\ln(1+\exp(\cos(x)))$.*

Calculer la dérivée d'une fonction élémentaire peut se faire simplement en utilisant la définition successives des éléments t_i dans la définition 5.1.3. En outre, la dérivée d'une fonction élémentaire est une fonction élémentaire. En ce qui concerne l'opération inverse consistant à calculer une primitive d'une fonction élémentaire, le problème s'avère plus compliqué d'autant plus qu'une fonction élémentaire n'admet pas nécessairement de primitive

élémentaire. Le problème de l'intégration symbolique en calcul formel se pose alors de la manière suivante : étant donnée une fonction élémentaire f, décider si elle admet une primitive F qui est elle-même une fonction élémentaire et si oui déterminer F. Avant de considérer le cas général, nous allons tout d'abord nous concentrer sur le cas particulier où la fonction élémentaire f dont on cherche à calculer une primitive est une fraction rationnelle de $\mathbb{C}(x)$.

5.2 Le cas particulier des fractions rationnelles

Étant donnée une fraction rationnelle $f/g \in \mathbb{C}(x)$, l'objectif de cette section est de calculer une primitive $F = \int f/g$ de f/g, i.e., calculer F telle que $F' = f/g$. Une telle primitive se décompose alors sous la forme $F = \int f/g = q + c/d + \int a/b$ où $q \in \mathbb{C}[x]$ est la *partie polynomiale*, $c/d \in \mathbb{C}(x)$ est la *partie rationnelle* et $\int a/b$ est la *partie logarithmique* de $\int f/g$. Le calcul de la primitive se décompose donc en trois étapes :

1. Calcul de la partie polynomiale ;
2. Calcul de la partie rationnelle ;
3. Calcul de la partie logarithmique.

Les trois sous-sections suivantes donnent des algorithmes pour effectuer chacune de ces étapes.

5.2.1 Calcul de la partie polynomiale

Le calcul de la partie polynomiale ne pose pas de problème particulier et peut être effectué simplement par l'algorithme 17 :

Algorithme 17 *Partie polynomiale de la primitive d'une fraction rationnelle*

Entrées : f, $g \in \mathbb{C}[x]$ deux polynômes de degré au plus n.
Sorties : q, $h \in \mathbb{C}[x]$ tels que $\deg(q) \leq n$, $\deg(h) < \deg(g)$ et $\int f/g = q + \int h/g$.
- Appliquer l'algorithme d'Euclide pour calculer le quotient $\tilde{q} = \sum_{i=0}^{d} q_i\, x^i$ et le reste r de la division euclidienne de f par g ;
- Calculer $q = \sum_{i=0}^{d} \frac{q_i}{i+1}\, x^{i+1}$;
- Retourner q et $h = r$.

Proposition 5.2.1. *Soit* $\mathbf{M}$ *une fonction telle que le produit de deux polynômes de degré au plus* n *puisse se calculer en* $O(\mathbf{M}(n))$ *opérations arithmétiques. L'algorithme 17 calcule la partie polynomiale de* $\int f/g$ *où* f *et* g *sont deux polynômes de degré au plus* n *en* $O(\mathbf{M}(n))$ *opérations arithmétiques.*

Démonstration. Le coût est dominé par celui de la division euclidienne d'où le résultat d'après la proposition 2.2.6. □

Nous sommes donc ramené à calculer $\int h/g$ où h et g sont deux polynômes tels que $\deg(h) < \deg(g)$.

5.2.2 Calcul de la partie rationnelle

Pour calculer la partie rationnelle de $\int h/g$ nous allons utiliser la *décomposition en fractions partielles* qui est une alternative à la décomposition en éléments simples utilisant une factorisation sans carré du dénominateur g au lieu de sa factorisation complète en produits de facteurs irréductibles. On rappelle que d'après les résultats de la sous-section 4.1.2, la factorisation sans carré d'un polynôme g de degré n s'écrit $g = g_1\, g_2^2 \cdots g_s^s$ où chaque g_i est un polynôme sans carré (pas nécessairement irréductible) et les g_i sont premiers entre eux deux à deux et peut se calculer en $O(\mathbf{M}(n)\log(n))$ opérations arithmétiques. Étant donnée la factorisation sans carré $g = g_1\, g_2^2 \cdots g_s^s$, le calcul de la décomposition en fractions partielles de h/g se déroule en deux étapes. On commence par écrire h/g sous la forme

$$\frac{h}{g} = \frac{p_1}{g_1} + \frac{p_2}{g_2^2} + \cdots + \frac{p_s}{g_s^s}, \quad \forall i \in \{1,\ldots,s\},\ \deg(p_i) < \deg(g_i^i), \tag{5.1}$$

puis on décompose à nouveau chaque p_i/g_i^i sous la forme

$$\frac{p_i}{g_i^i} = \frac{p_{i,1}}{g_i} + \frac{p_{i,2}}{g_i^2} + \cdots + \frac{p_{i,i}}{g_i^i}, \quad \forall i \in \{1,\ldots,s\},\ \forall j \in \{1,\ldots,i\},\ \deg(p_{i,j}) < \deg(g_i). \tag{5.2}$$

En multipliant l'équation (5.1) par $g = g_1\, g_2^2 \cdots g_s^s$, on obtient

$$h = p_1\, g_2^2 \cdots g_s^s + p_2\, g_1\, g_3^3 \cdots g_s^s + \cdots + p_s\, g_1 \cdots g_{s-1}^{s-1},$$

de sorte que

$$\forall i \in \{1,\ldots,s\}, \quad h \equiv p_i\, g_1 \cdots g_{i-1}^{i-1}\, g_{i+1}^{i+1} \cdots g_s^s \mod g_i^i.$$

Les polynômes g_j^j et g_i^i étant premiers entre eux pour $i \neq j$, g_j^j est inversible modulo g_i^i et son inverse modulaire g_j^{-j} modulo g_i^i peut être calculé à l'aide de l'algorithme d'Euclide étendu (voir la sous-section 3.2.3). On obtient donc les congruences

$$\forall i \in \{1,\ldots,s\}, \quad p_i \equiv h\, g_1^{-1} \cdots g_{i-1}^{-(i-1)}\, g_{i+1}^{-(i+1)} \cdots g_s^{-s} \mod g_i^i.$$

L'hypothèse $\deg(p_i) < \deg(g_i^i)$ nous assure alors que les polynômes p_i sont déterminés de manière unique. Une fois les p_i déterminés, pour obtenir l'écriture (5.2), on pose $r_i = p_i$ puis on effectue successivement les divisions euclidiennes de r_j par g_i pour $j = i,\ldots,2$. On écrit alors $r_j = r_{j-1}\, g_i + p_{i,j}$, on définit $p_{i,1} = r_1$ et on a $p_i = p_{i,1}\, g_i^{i-1} + p_{i,2}\, g_i^{i-2} \cdots + p_{i,i}$ qui implique (5.2). Finalement on obtient l'algorithme 17 :

Proposition 5.2.2. *Soit* $\mathbf{M}$ *une fonction telle que le produit de deux polynômes de degré au plus* n *puisse se calculer en* $O(\mathbf{M}(n))$ *opérations arithmétiques. Soient* h *et* g *deux polynômes tels que* $\deg(h) < \deg(g) = n$ *et soit* $g = g_1\, g_2^2 \cdots g_s^s$ *la factorisation sans carré de* g*. Alors l'algorithme 18 est correct et calcule la décomposition en fractions partielles de* h/g *en* $O(s\,\mathbf{M}(n)\log(n))$ *opérations arithmétiques.*

Algorithme 18 *Décomposition en fractions partielles*

Entrées : h, $g \in \mathbb{C}[x]$ deux polynômes tels que $\deg(h) < \deg(g) = n$ et la factorisation sans carré $g = g_1\, g_2^2 \cdots g_s^s$ de g.

Sorties : des polynômes $p_{i,j} \in \mathbb{C}[x]$ tels que l'on a (5.1) et (5.2).

Pour tout $i = 1, \ldots, s$ **Faire**

- Appliquer l'algorithme d'Euclide étendu pour calculer deux polynômes u_i et v_i tels que $u_i \left(\frac{g}{g_i^i}\right) + v_i\, g_i^i = 1$;
- Calculer le reste p_i dans la division euclidienne de $h\, u_i$ par g_i^i ;
- Poser $r_i = p_i$, puis effectuer successivement les divisions euclidiennes $r_j = r_{j-1}\, g_i + p_{i,j}$ de r_j par g_i pour $j = i, \ldots, 2$ et poser $p_{i,1} = r_1$;

Fin pour

- Retourner les $p_{i,j}$.

Démonstration. La correction de l'algorithme 18 découle directement des explications données ci-dessus. Étudions sa complexité arithmétique. Le calcul de chaque p_i requiert une division exacte $\frac{g}{g_i^i}$ de deux polynômes de degré au plus n, une inversion modulaire en degré au plus n (qui peut se faire à l'aide de l'algorithme d'Euclide étendu - voir la sous-section 3.2.3 - en complexité $O(\mathbf{M}(n)\log(n))$ d'après la remarque 3.2.1), une multiplication $h\, u_i$ de polynômes de degré au plus n et une division euclidienne d'un polynôme de degré au plus $2\,n$ par un polynôme de degré n. La complexité du calcul de tous les p_i est donc en $O(s\,\mathbf{M}(n)\log(n))$ opérations arithmétiques. Pour un i fixé, les polynômes r_j sont de degrés au plus $j\,\deg(g_i)$ donc le coût du calcul de $p_{i,i}, \ldots, p_{i,1}$ est majoré par $O(i^2\,\mathbf{M}(\deg(g_i)))$ opérations arithmétiques. En faisant la somme pour i allant de 1 à s et en utilisant $\mathbf{M}(n+n') \geq \mathbf{M}(n)+\mathbf{M}(n')$ on obtient que l'extraction de tous les $p_{i,j}$ peut se faire en $O(s\,\mathbf{M}(n))$ opérations arithmétiques d'où le résultat. □

Si $g = g_1\, g_2^2 \cdots g_s^s$ désigne la factorisation sans carré de g, nous avons donc réduit le problème du calcul de primitive de h/g avec $\deg(h) < \deg(g)$ à celui du calcul de primitive de $\sum_{i=1}^{s}\sum_{j=1}^{i} p_{i,j}/g_i^j$ avec pour tout i, j, $\deg(p_{i,j}) < \deg(g_i)$. Les polynômes g_i sont sans carrés de sorte que $\mathrm{PGCD}(g_i, g_i') = 1$ (voir Proposition 4.1.2) ce qui entraine qu'ils existent deux polynômes u_i et v_i tels que $u_i\, g_i + v_i\, g_i' = 1$. On obtient donc

$$\int \frac{p_{i,j}}{g_i^j} = \int \frac{p_{i,j}\, u_i\, g_i + p_{i,j}\, v_i\, g_i'}{g_i^j} = \int \frac{p_{i,j}\, u_i}{g_i^{j-1}} + \int \left(\frac{-p_{i,j}\, v_i}{j-1}\right)\left(\frac{-(j-1)\, g_i'}{g_i^j}\right),$$

puis, en intégrant par parties

$$\int \frac{p_{i,j}}{g_i^j} = \frac{-p_{i,j}\, v_i}{(j-1)\, g_i^{j-1}} + \int \frac{p_{i,j}\, u_i + (p_{i,j}\, v_i)'/(j-1)}{g_i^{j-1}}.$$

On voit donc que l'on peut ainsi ramener l'intégration d'un pôle d'ordre j à celle d'un pôle d'ordre $j-1$. En appliquant ce procédé de manière itérative, on obtient le *procédé de réduction de Hermite* qui permet de calculer la partie rationnelle de la primitive de h/g en suivant l'algorithme 19.

Algorithme 19 *Partie rationnelle de la primitive d'une fraction rationnelle : réduction de Hermite*

Entrées : h, $g \in \mathbb{C}[x]$ deux polynômes tels que $\deg(h) < \deg(g) = n$ et la décomposition en fractions partielles $h/g = \sum_{i=1}^{s} \sum_{j=1}^{i} p_{i,j}/g_i^j$ où $g = g_1\, g_2^2 \cdots g_s^s$ est la factorisation sans carré de g.

Sorties : des polynômes a_i, b_i, c_i et d_i tels que $\int h/g = \sum_i c_i/d_i + \int \sum_i a_i/b_i$ et où les b_i divisent $g_1\, g_2 \cdots g_s$ et les d_i divisent $g_2\, g_3^2 \cdots g_s^{s-1}$.

- Poser $l = 0$;

Pour tout $i = 1, \ldots, s$ **Faire**

- Appliquer l'algorithme d'Euclide étendu pour calculer u_i et v_i de degré $< \deg(g_i)$ tels que $u_i\, g_i + v_i\, g_i' = 1$;

Pour tout $j = i, \ldots, 2$ **Faire**

- Effectuer la division euclidienne de $v_i\, p_{i,j}$ par g_i : on écrit $v_i\, p_{i,j} = q\, g_i + r$ avec $\deg(r) < \deg(g_i)$;
- Poser $t = u_i\, p_{i,j} + (q\, g_i')/(j-1)$;
- Poser $p_{i,j-1} = p_{i,j-1} + t + r'/(j-1)$ (normaliser l'expression obtenue pour $p_{i,j-1}$) ;
- Poser $l = l + 1$;
- Poser $c_l = -r/(j-1)$ et $d_l = g_i^{j-1}$;

Fin pour

Fin pour

- Pour $i = 1, \ldots, s$, poser $a_i = p_{i,1}$ et $b_i = g_i$;
- Retourner les polynômes a_i, b_i, c_i et d_i.

Proposition 5.2.3. *Soit* $\mathbf{M}$ *une fonction telle que le produit de deux polynômes de degré au plus* n *puisse se calculer en* $O(\mathbf{M}(n))$ *opérations arithmétiques. Soient* h *et* g *deux polynômes tels que* $\deg(h) < \deg(g) = n$ *et soit* $h/g = \sum_{i=1}^{s} \sum_{j=1}^{i} p_{i,j}/g_i^j$ *la décomposition en fraction partielles de* h/g *où* $g = g_1\, g_2^2 \cdots g_s^s$ *est la factorisation sans carré de* g*. Alors l'algorithme 19 est correct et calcule la partie rationnelle de* $\int h/g$ *en* $O(\mathbf{M}(n)\log(n))$ *opérations arithmétiques.*

Démonstration. La correction de l'algorithme 19 découle directement des explications données ci-dessus. L'étude de la complexité de l'algorithme est laissée en exercice. Elle utilise les mêmes arguments que ceux de la preuve de la proposition 5.2.2. □

Exemple 5.2.1. *Appliquons la réduction de Hermite pour le calcul de la primitive de la faction rationnelle*

$$\frac{1}{(2\,x^2+1)^2}.$$

Posons $h = 1/4$ *et* $g = (x^2+1/2)^2 = g_2^2$ *avec* $g_2 = x^2+1/2$ *sans carré et appliquons la réduction de Hermite pour calculer la partie rationnelle de* h/g*. Avec les notations de l'algorithme 19, nous avons donc* $h/g = p_{2,2}/g_2^2$ *avec* $p_{2,2} = h = 1/4$*. L'algorithme d'Euclide étendu nous fournit donc* $u_2 = 2$ *et* $v_2 = -x$ *tels que* $u_2\, g_2 + v_2\, g_2' = 1$*. On a alors* $v_2\, p_{2,2} = q\, g_2 + r$ *avec* $q = 0$ *et* $r = v_2\, p_{2,2} = -x/4$*. On obtient alors* $p_{2,1} = u_2\, p_{2,2} - 1/4 = 1/4$ *puis* $c_1 = x/4$ *et* $d_1 = g_2$*. Finalement on pose* $a_2 = p_{2,1} = 1/4$ *et* $b_2 = g_2$ *de sorte que*

$$\int \frac{1}{(2\,x^2+1)^2} = \frac{x/4}{x^2+1/2} + \int \frac{1/4}{x^2+1/2} = \frac{x}{2\,(2x^2+1)} + \int \frac{1}{2\,(2\,x^2+1)}.$$

La forme des polynômes d_i et b_i retournés par l'algorithme 19 montre que la sortie de l'algorithme peut s'écrire sous la forme normalisée $c/(g_2\, g_3^2 \cdots g_s^{s-1}) + \int a/(g_1\, g_2 \cdots g_s)$. De plus, en notant δ_i, $i = 1, \ldots, s$, le degré de g_i et en étudiant le degré des polynômes c_i et a_i dans l'algorithme, on voit que le degré de c sera strictement inférieur à $\delta_2 + 2\,\delta_3 + \cdots + (s-1)\,\delta_s$ et celui de a sera strictement inférieur à $\delta_1 + \delta_2 + \cdots + \delta_s$. On peut alors utiliser une méthode par coefficients indéterminés pour calculer directement a et b en résolvant un système linéaire. Cette méthode est dû à Horowitz.

Proposition 5.2.4. *Soit* $\mathbf{MM}$ *une fonction telle que le produit de deux matrices carrées de taille* n *puisse se calculer en* $O(\mathbf{MM}(n))$ *opérations arithmétiques. Soient* h *et* g *deux polynômes tels que* $\deg(h) < \deg(g) = n$ *et* $g = g_1\, g_2^2 \cdots g_s^s$ *la factorisation sans carré de* g*. Alors l'algorithme 20 est correct et calcule la partie rationnelle de* $\int h/g$ *en* $O(\mathbf{MM}(n))$ *opérations arithmétiques.*

Démonstration. La correction de l'algorithme découle des explications ci-dessus. Le coût est clairement dominé par la résolution du système linéaire qui a n équations et n inconnues d'où le résultat d'après le théorème 2.3.1. □

Exemple 5.2.2. *Reprenons l'exemple 5.2.1 et appliquons l'algorithme 20 de Horowitz. Nous avons toujours* $h = 1/4$ *et* $g_2 = x^2+1/2$*. Avec les notations de l'algorithme 20, on obtient*

Algorithme 20 *Partie rationnelle de la primitive d'une fraction rationnelle : algorithme de Horowitz*

Entrées : h, $g \in \mathbb{C}[x]$ deux polynômes tels que $\deg(h) < \deg(g) = n$ et $g = g_1\, g_2^2 \cdots g_s^s$ la factorisation sans carré de g.

Sorties : des polynômes a et c tels que $\int h/g = c/(g_2\, g_3^2 \cdots g_s^{s-1}) + \int a/(g_1\, g_2 \cdots g_s)$.

- Poser $\delta = \delta_1 + \delta_2 + \cdots + \delta_s$ où pour $i = 1, \ldots, s$, $\delta_i = \deg(g_i)$;
- Écrire $a = a_0 + a_1\, x + \cdots a_{\delta-1}\, x^{\delta-1}$ et $c = c_0 + c_1\, x + \cdots + c_{n-\delta-1}\, x^{n-\delta-1}$, où les a_i et les c_i sont des coefficients à déterminer ;
- Écrire le système linéaire obtenu en égalant à zéro tous les coefficients du polynôme $h - c'\,(g_1 \cdots g_s) + c \sum_{i=2}^{s} (i-1)\, g_1\, g_2 \cdots g_{i-1}\, g_i'\, g_{i+1} \cdots g_s - a\, g_2\, g_3^2 \cdots g_s^{s-1}$;
- Résoudre le système linéaire obtenu ;
- Substituer la solution du système linéaire dans a et c et retourner a et c.

$\delta = 2$ de sorte que l'on cherche a et c sous la forme $a = a_0 + a_1\, x$ et $c = c_0 + c_1\, x$ où a_0, a_1, c_0, c_1 sont des constantes à déterminer. On a alors

$$h - c'\, g_2 + c\, g_2' - a\, g_2 = -a_1\, x^3 + (c_1 - a_0)\, x^2 + \left(2\, c_0 - \frac{a_1}{2}\right) x + \left(\frac{1}{4} - \frac{c_1}{2} - \frac{a_0}{2}\right),$$

d'où le système linéaire $\{-a_1 = 0,\ c_1 - a_0 = 0,\ 2\, c_0 - a_1/2 = 0,\ 1/4 - c_1/2 - a_0/2 = 0\}$ que l'on résout pour trouver $a_0 = 1/4$, $a_1 = 0$, $c_0 = 0$, $c_1 = 1/4$. On a donc trouvé $a = 1/4$ et $c = (1/4)\, x$ et on retrouve donc

$$\int \frac{1}{(2\, x^2 + 1)^2} = \frac{c}{g_2} + \int \frac{a}{g_2} = \frac{x}{2\,(2x^2 + 1)} + \int \frac{1}{2\,(2\, x^2 + 1)}.$$

Notons que l'algorithme 20 de Horowitz est significativement moins bon du point de vue de la complexité que l'algorithme 19 utilisant la réduction de Hermite (voir le chapitre 2). Son avantage est qu'il ne nécessite pas le calcul préalable de la décomposition en fractions partielles et qu'il est très simple à programmer.

5.2.3 Calcul de la partie logarithmique

Nous sommes maintenant ramené au problème du calcul de la partie logarithmique de la primitive d'une fraction rationnelle. Autrement dit, nous voulons calculer $\int a/b$ où $a/b \in \mathbb{K}(x)$, $\deg(a) < \deg(b)$ et b est un polynôme sans carré. Dans ce cas là, nous allons naturellement voir apparaître des logarithmes (penser à $\int 1/x$) mais nous allons devoir faire face à une autre difficulté à savoir le fait que les coefficients de la primitive vivent en général dans une extension algébrique de $\mathbb{K}$. Nous allons utiliser le théorème suivant qui sera admis pour ce cours.

Théorème 5.2.1. *Soit $a/b \in \mathbb{K}(x)$ une fraction rationnelle telle que a et b sont premiers entre eux, $\deg(a) < \deg(b) = n$ et b est un polynôme unitaire et sans carré. Soit $\mathbb{L}$ une*

extension algébrique de $\mathbb{K}$. *Alors on a*

$$\int \frac{a}{b} = \sum_{i=1}^{m} c_i \ln(v_i),$$

pour des constantes $c_i \in \mathbb{L}$ *et des polynômes* $v_i \in \mathbb{L}[x]$ *premiers entre eux deux à deux si et seulement si le résultant* $R(y) = \mathrm{Res}_x(b, a - b'\,y)$ *(parfois appelé* résultant de Rothstein-Trager*) se factorise en produit de facteurs linéaires en* y *dont les* c_i *sont les zéros et les* v_i *sont les* $\mathrm{PGCD}(b, a - b'\,c_i)$.

Ce théorème mène à l'algorithme 21 de Rothstein-Trager pour le calcul de la partie logarithmique de la primitive d'une fraction rationnelle.

Algorithme 21 *Partie logarithmique de la primitive d'une fraction rationnelle : algorithme de Rothstein-Trager*

Entrées : $a, b \in \mathbb{K}[x]$ deux polynômes tels que $\deg(a) < \deg(b) = n$ avec b unitaire et sans carré.

Sorties : une extension algébrique $\mathbb{L}$ de $\mathbb{K}$, des constantes $c_1, \ldots, c_m \in \mathbb{L}$ et des polynômes $v_1, \ldots, v_m \in \mathbb{L}[x]$ tels que $\int a/b = \sum_{i=1}^{m} c_i \ln(v_i)$.

- Calculer le résultant $R(y) = \mathrm{Res}_x(b, a - b'\,y)$;
- Factoriser $R(y)$ en produit de facteurs linéaires en introduisant l'extension algébrique nécessaire $\mathbb{L}$ de $\mathbb{K}$;
- Soient $c_1, \ldots, c_m$ les zéros deux à deux distincts de $R(y)$ dans $\mathbb{L}$;
- Pour $i = 1, \ldots, m$, calculer $v_i = \mathrm{PGCD}(b, a - b'\,c_i)$;
- Retourner $\mathbb{L}$, les c_i et les v_i.

Proposition 5.2.5. *Soit* $\mathbf{M}$ *une fonction telle que le produit de deux polynômes de degré au plus* n *puisse se calculer en* $O(\mathbf{M}(n))$ *opérations arithmétiques. Soient* $a, b \in \mathbb{K}[x]$ *deux polynômes tels que* $\deg(a) < \deg(b) = n$ *avec* b *unitaire et sans carré. Alors l'algorithme 21 est correct et calcule la partie logarithmique de* $\int a/b$ *en* $O(n\,\mathbf{M}(n)^2 \log(n))$ *opérations arithmétiques dans* $\mathbb{K}$ *plus le coût de la factorisation du résultant (qui est de degré au plus* n*) à l'étape 2.*

Démonstration. La correction de l'algorithme est une conséquence directe du théorème 5.2.1. Le calcul du résultant de deux polynômes de degré au plus n peut se calculer en $O(n^2)$ (voir la sous-section 3.3.3). Le calcul de $R(y)$ nécessite donc $O(n^2)$ opérations dans $\mathbb{K}[y]$ et comme le degré en y des polynômes qui apparaissent ne dépasse pas n, le résultant $R(y)$ peut se calculer en $O(n^2\,\mathbf{M}(n))$ opérations dans $\mathbb{K}$. De même le calcul des PGCD à l'étape 4 coûte $O(\mathbf{M}(n)\log(n))$ opérations dans $\mathbb{L}$ soit au plus $O(\mathbf{M}(n)^2\log(n))$ opérations dans $\mathbb{K}$ puisque le degré de $\mathbb{L}$ sur $\mathbb{K}$ ne dépasse pas le degré de $R(y)$ qui est majoré par n. D'où l'estimation annoncée. □

Exemple 5.2.3. *Reprenons l'exemple 5.2.1 et appliquons l'algorithme de Rothstein-Trager pour calculer*

$$\int \frac{1}{2\,(2\,x^2+1)}.$$

On pose donc $a = 1/4$ *et* $b = x^2 + 1/2$. *On a*

$$R(y) = \mathrm{Res}_x(b, a - b'\,y) = \begin{vmatrix} 1 & 0 & \frac{1}{2} \\ -2\,y & \frac{1}{4} & 0 \\ 0 & -2\,y & \frac{1}{4} \end{vmatrix} = 2\,y^2 + \frac{1}{16}.$$

On introduit maintenant l'extension algébrique $\mathbb{Q}(i\,\sqrt{2})$ *de* $\mathbb{Q}$ *pour factoriser* $R(y)$ *en facteurs linéaires sous la forme*

$$R(y) = 2\left(y + \frac{i\,\sqrt{2}}{8}\right)\left(y - \frac{i\,\sqrt{2}}{8}\right).$$

On pose alors $c_1 = \frac{i\sqrt{2}}{8}$, $c_2 = -\frac{i\sqrt{2}}{8}$ *et on calcule* $v_1 = \mathrm{PGCD}(b, a - b'\,c_1) = x + \frac{i\sqrt{2}}{2}$ *et* $v_2 = \mathrm{PGCD}(b, a - b'\,c_2) = x - \frac{i\sqrt{2}}{2}$. *On obtient alors*

$$\int \frac{1}{2\,(2\,x^2+1)} = c_1\,\ln(v_1) + c_2\,\ln(v_2) = \frac{i\,\sqrt{2}}{8}\,\ln\left(x + \frac{i\,\sqrt{2}}{2}\right) - \frac{i\,\sqrt{2}}{8}\,\ln\left(x - \frac{i\,\sqrt{2}}{2}\right)$$

d'où

$$\int \frac{1}{(2\,x^2+1)^2} = \frac{x}{2\,(2x^2+1)} + \frac{i\,\sqrt{2}}{8}\left(\ln\left(x + \frac{i\,\sqrt{2}}{2}\right) - \ln\left(x - \frac{i\,\sqrt{2}}{2}\right)\right).$$

Notons finalement qu'en utilisant un algorithme « rapide » pour calculer le résultant $R(y)$, nous pouvons gagner un facteur n et obtenir une complexité en $O(\mathbf{M}(n)^2\log(n))$ pour le calcul de la partie logarithmique par l'algorithme de Rothstein-Trager.

5.3 Le cas général des fonctions élémentaires

Étant donnée une fonction élémentaire, nous allons maintenant donner l'idée d'un algorithme qui décide si elle admet une primitive qui est elle même une fonction élémentaire et si oui la calcule. L'idée de cet algorithme est basée sur une idée attribuée à Liouville qui consiste à généraliser le Théorème 5.2.1 aux fonctions élémentaires qui ne sont pas nécessairement des fractions rationnelles. On obtient alors le résultat suivant qui est admis dans le cadre de ce cours.

Théorème 5.3.1 (Principe de Risch-Liouville)**.** *Soient* $(\mathbb{K},')$ *une extension élémentaire de* $(\mathbb{C}(x),')$, $\mathcal{C}$ *son corps des constantes, i.e.,* $\mathcal{C} = \{f \in \mathbb{K} \mid f' = 0\}$, *et* $f \in \mathbb{K}$ *une fonction élémentaire. S'il existe une extension élémentaire* $(\mathbb{L},')$ *de* $(\mathbb{K},')$ *et* $g \in \mathbb{L}$ *telle que* $g' = f$,

alors il existe $v \in \mathbb{K}$*, des constantes* c_i*,* $i = 1, \ldots, n$*, algébriques sur* $\mathcal{C}$ *et des éléments* $v_i \in \mathbb{K}(c_1, \ldots, c_n)$*,* $i = 1, \ldots, n$*, tels que*

$$f = v' + \sum_{i=1}^{n} c_i \frac{v_i'}{v_i}.$$

Ce théorème de structure montre que s'il existe une primitive, alors elle sera d'une forme particulière et on sait à l'avance le type de fonctions qu'il faudra introduire pour l'exprimer. Ce théorème mène à un algorithme appelé *algorithme de Risch* qui consiste à reconnaitre cette forme particulière. Soit $f \in \mathbb{K} = \mathbb{C}(t_1, \ldots, t_n)$. Le principe général de l'algorithme de Risch consiste à procéder de manière récursive pour « éliminer » les t_i et se ramèner à l'intégration d'une fraction rationnelle. Cet algorithme est assez technique et ne sera pas détaillé dans ce cours.

Chapitre 6

Équations différentielles linéaires et séries formelles

Dans ce chapitre, l'objet de notre étude est une *équation différentielle linéaire à coefficients polynomiaux* de la forme

$$a_n(x)\, y^{(n)}(x) + a_{n-1}(x)\, y^{(n-1)}(x) + \cdots + a_1(x)\, y'(x) + a_0(x)\, y(x) = 0, \tag{6.1}$$

où, pour $i = 0, \ldots, n$, les $a_i(x) \in \mathbb{Q}[x]$ (ou $\mathbb{C}[x]$) sont des polynômes donnés, y est une fonction inconnue de la variable x et $y^{(i)}$ désigne la $i$ème dérivée de la fonction y par rapport à x. En notant

$$L = a_n(x) \left(\frac{d}{dx}\right)^n + a_{n-1}(x) \left(\frac{d}{dx}\right)^{n-1} + \cdots + a_1(x) \left(\frac{d}{dx}\right) + a_0(x)$$

l'*opérateur différentiel linéaire* associé, l'équation (6.1) s'écrit alors :

$$L(y) = 0.$$

L'objectif consiste à donner des méthodes de calcul de certains types de séries solutions de $L(y) = 0$ au voisinage d'un point $x_0 \in \mathbb{C}$.

6.1 Résolution de $L(y) = 0$ par des séries entières

6.1.1 Structure des solutions

Les solutions de $L(y) = 0$ forment un espace vectoriel sur $\mathbb{Q}$ (ou $\mathbb{R}$ ou $\mathbb{C}$). En effet, si c_1, c_2 sont des constantes et y_1, y_2 des solutions de $L(y) = 0$, alors on vérifie que $L(c_1\, y_1 + c_2\, y_2) = 0$. Pour étudier la dimension de cet espace vectoriel, on introduit la notion de *wronskien*.

Définition 6.1.1. *Soient $y_1, \dots, y_r$ des fonctions d'une variable x. On appelle* wronskien *de $y_1, \dots, y_r$ et on note $\mathrm{Wr}(y_1, \dots, y_r)$ le déterminant*

$$\mathrm{Wr}(y_1, \dots, y_r) = \begin{vmatrix} y_1 & y_2 & \dots & y_r \\ y_1' & y_2' & \dots & y_r' \\ \vdots & \vdots & & \vdots \\ y_1^{(r-1)} & y_2^{(r-1)} & \dots & y_r^{(r-1)} \end{vmatrix}.$$

Lemme 6.1.1. *Des fonctions $y_1, \dots, y_r$ d'une variable x sont linéairement indépendantes sur $\mathbb{Q}$ (ou $\mathbb{R}$ ou $\mathbb{C}$) si et seulement si $\mathrm{Wr}(y_1, \dots, y_r) \neq 0$.*

Démonstration. Des fonctions $y_1, \dots, y_r$ sont linéairement dépendantes sur $\mathbb{Q}$ si et seulement si il existe des constantes $c_1, \dots, c_r \in \mathbb{Q}$ non toutes nulles telles que $c_1\, y_1 + \dots + c_r\, y_r = 0$. En dérivant cette relation, on obtient $c_1\, y_1^{(i)} + \dots + c_r\, y_r^{(i)} = 0$ pour tout $i \geq 1$. Le système linéaire de r équations $\{c_1\, y_1^{(i)} + \dots + c_r\, y_r^{(i)} = 0,\ i = 0, \dots, r-1\}$ à r inconnues $c_1, \dots, c_r$ admet une solution non nulle si et seulement si son déterminant est égal à zéro d'où le résultat. □

Corollaire 6.1.1. *Soit $L(y) = y^{(n)} + \sum_{i=0}^{n-1} a_i(x)\, y^{(i)}(x)$. Les solutions de $L(y) = 0$ forment un espace vectoriel de dimension au plus n.*

Démonstration. Supposons que $y_1, \dots, y_{n+1}$ sont des solutions de $L(y) = 0$. On a

$$\mathrm{Wr}(y_1, \dots, y_{n+1}) = \begin{vmatrix} y_1 & y_2 & \dots & y_{n+1} \\ y_1' & y_2' & \dots & y_{n+1}' \\ \vdots & \vdots & & \vdots \\ y_1^{(n)} & y_2^{(n)} & \dots & y_{n+1}^{(n)} \end{vmatrix},$$

et donc, en utilisant les relations $y_j^{(n)} = -\sum_{i=0}^{n-1} a_i\, y_j^{(i)}$, $j = 1, \dots, n+1$, on obtient

$$\mathrm{Wr}(y_1, \dots, y_{n+1}) = \begin{vmatrix} y_1 & y_2 & \dots & y_{n+1} \\ y_1' & y_2' & \dots & y_{n+1}' \\ \vdots & \vdots & & \vdots \\ -\sum_{i=0}^{n-1} a_i\, y_1^{(i)} & -\sum_{i=0}^{n-1} a_i\, y_2^{(i)} & \dots & -\sum_{i=0}^{n-1} a_i\, y_{n+1}^{(i)}(x) \end{vmatrix}.$$

On en déduit donc $\mathrm{Wr}(y_1, \dots, y_{n+1}) = 0$ (la dernière ligne du déterminant ci-dessus est une combinaison linéaire des lignes précédentes) et le lemme 6.1.1 implique alors que $y_1, \dots, y_{n+1}$ sont linéairement dépendantes d'où le résultat. □

Définition 6.1.2. *Soient $y_1, \dots, y_n$ des solutions de $L(y) = 0$. Si elles sont linéairement indépendantes sur $\mathbb{C}$, alors on dit qu'elles forment un* système fondamental de solutions *de $L(y) = 0$.*

Dans la suite « résoudre $L(y) = 0$ » signifiera trouver un système fondamental de solutions. Le terme « solution » peut avoir plusieurs interprétations. Prenons l'exemple de l'équation $L(y) = y'' + y = 0$. On distingue alors :

- les *solutions sous forme close* (explicite finie) c'est-à-dire dans une classe de fonctions donnée (sin, cos, exp, ...). Ici $y(x) = c_1\,\cos(x) + c_2\,\sin(x)$ est solution pour tout couple de constantes $(c_1, c_2) \in \mathbb{C}^2$;
- les *solutions sous forme de séries formelles* (on ne se pré-occupe pas de la convergence des séries). Ici, si on note $y(x) = \sum_{i=0}^{+\infty} c_i\,x^i$ une telle série formelle, alors on a $y''(x) = \sum_{i=0}^{+\infty}(i+1)\,(i+2)\,c_{i+2}\,x^i$ d'où $y(x) + y''(x) = \sum_{i=0}^{+\infty}(c_i + (i+1)\,(i+2)\,c_{i+2})\,x^i$ donc les coefficients c_i d'une telle série solution doivent vérifier $c_i + (i+1)\,(i+2)\,c_{i+2} = 0$.
- les *solutions sous forme de séries analytiques* (en 0), i.e., les séries $y(x) = \sum_{i=0}^{\infty} y_i\,x^i$ solutions qui convergent dans un disque $D(0;R)$ centré en 0 et de rayon R.

6.1.2 Existence de séries solutions au voisinage d'un point ordinaire

Définition 6.1.3. *Soit* $L(y) = a_n(x)\,y^{(n)}(x) + \cdots + a_1(x)\,y'(x) + a_0(x)\,y(x) = 0$ *où pour tout* $i = 0, \ldots, n$, $a_i(x) \in \mathbb{C}[x]$. *On dit que* $x_0 \in \mathbb{C}$ *est un* point ordinaire de $L(y) = 0$ *si* $a_n(x_0) \neq 0$. *Sinon on dit que* x_0 *est un* point singulier *(ou une* singularité*) de* $L(y) = 0$.

On dira que l'infini est un point ordinaire (resp. singulier) de $L(y) = 0$ si 0 est un point ordinaire (resp. singulier) de l'équation obtenue après le changement de variable $X = 1/x$.

Théorème 6.1.1 (Théorème de Cauchy)**.** *Au voisinage d'un point ordinaire, une équation différentielle linéaire* $L(y) = 0$ *admet un système fondamental de solutions analytiques.*

Démonstration. Quitte à faire un changement de variable, on suppose que $x_0 = 0$ est un point ordinaire de $L(y) = 0$. Comme $a_n(0) \neq 0$, on sait que si y est solution de $L(y) = 0$, alors

$$y^{(n)}(0) = -\sum_{i=0}^{n-1} \frac{a_i(0)}{a_n(0)}\,y^{(i)}(0).$$

De plus, on a

$$\begin{aligned} y^{(n+1)}(x) &= -\left(\textstyle\sum_{i=0}^{n-1} \frac{a_i(x)}{a_n(x)}\,y^{(i)}(x)\right)' \\ &= -\left(\textstyle\sum_{i=0}^{n-1} \frac{a_i(x)'\,a_n(x) - a_i(x)\,a_n(x)'}{a_n(x)^2}\,y^{(i)}(x) + \sum_{i=0}^{n-2} \frac{a_i(x)}{a_n(x)}\,y^{(i+1)}(x) + \frac{a_{n-1}(x)}{a_n(x)}\,y^{(n)}(x)\right), \end{aligned}$$

puis en utilisant à nouveau la relation $y^{(n)}(x) = -\sum_{i=0}^{n-1} \frac{a_i(x)}{a_n(x)}\,y^{(i)}(x)$, on obtient

$$y^{(n+1)}(x) = \sum_{i=0}^{n-1} \frac{a_{i,1}(x)}{a_n(x)^2}\,y^{(i)}(x), \quad a_{i,1}(x) \in \mathbb{C}[x].$$

De la même façon, pour tout entier $k \in \mathbb{N}$, on obtient des relations de la forme

$$y^{(n+k)}(x) = \sum_{i=0}^{n-1} \frac{a_{i,k}(x)}{a_n(x)^{2^k}}\,y^{(i)}(x), \quad a_{i,k}(x) \in \mathbb{C}[x].$$

Par conséquent les $y^{(n+k)}(0)$ dépendent tous linéairement de $y(0), y'(0), \ldots, y^{(n-1)}(0)$. On construit alors un système fondamental $y_1, \ldots, y_n$ de solutions sous forme de séries de la manière suivante. Pour $j = 1, \ldots, n$, $y_j(x) = \sum_{i=0}^{+\infty} \frac{y_j^{(i)}(0)}{i!}\, x^i$ est la série définie par :

- Pour $k = 0, \ldots, n-1$, $y_j^{(k)}(0) = 1$ si $k = j-1$ et 0 sinon.
- Pour $k \in \mathbb{N}$, $y_j^{(n+k)}(0) = \sum_{i=0}^{n-1} \frac{a_{i,k}(0)}{a_n(0)^{2^k}}\, y_j^{(i)}(0) = \frac{a_{j-1,k}(0)}{a_n^{2^k}(0)}$.

Les séries $y_1, \ldots, y_n$ ainsi construites sont solutions de $L(y) = 0$ (par construction) et elles sont linéairement indépendantes. En effet, si $y = \sum_{i=1}^{n} c_i\, y_i = 0$, alors, pour $k = 0, \ldots, n-1$, la relation $y^{(k)}(0) = 0$ implique $c_{k+1} = 0$. Elles forment donc un système fondamental de solutions de $L(y) = 0$. On admettra que ces séries sont analytiques ce qui termine la preuve. □

Remarque 6.1.1. *Les séries $y_1, \ldots, y_n$ construites dans la preuve précédente sont convergentes (admis) sur un disque $D(0, R)$, où R est le module de la singularité la plus proche.*

Le procédé de construction d'un système fondamental de séries solutions d'une équation différentielle linéaire au voisinage d'un point ordinaire décrit dans la preuve précédente est souvent appelé *méthode de Cauchy*. Cette méthode a l'avantage de se généraliser à toutes les équations différentielles (non nécessairement linéaires) de la forme

$$y^{(n)}(x) = \frac{P(x, y(x), y'(x), \ldots, y^{(n-1)}(x))}{Q(x, y(x), y'(x), \ldots, y^{(n-1)}(x))},$$

où P et Q sont des polynômes (multivariés). À toute condition initiale $\big(x_0, y(x_0), \ldots, y^{(n-1)}(x_0)\big)$ telle que $Q(x_0, y(x_0), \ldots, y^{(n-1)}(x_0)) \neq 0$, on peut associer une unique série formelle solution de l'équation.

6.1.3 Calcul des séries solutions au voisinage d'un point ordinaire

Il existe des méthodes autres que celle de Cauchy abordée précédemment pour construire un système fondamental de séries formelles solutions d'une équation différentielle linéaire $L(y) = 0$ au voisinage d'un point ordinaire.

Méthode d'Euler-Newton : calcul par *approximations successives*. L'idée consiste à chercher chaque solution y_i (pour $i = 1, \ldots, n$) sous la forme

$$y_i(x) = \frac{x^{i-1}}{(i-1)!} + x^n\, \Sigma_n(x), \qquad \Sigma_n \in \mathbb{C}[[x]],$$

où Σ_n est une série formelle à déterminer. L'équation $L(y_i) = 0$ donne alors une nouvelle équation

$$\tilde{L}(\Sigma_n(x)) = -L\left(\frac{x^{i-1}}{(i-1)!}\right),$$

pour la série formelle Σ_n. On pose alors $\Sigma_n(x) = c_n + x\,\Sigma_{n+1}(x)$ et le terme de plus bas degré de la relation

$$\tilde{L}(c_n + x\,\Sigma_{n+1}(x)) = -L\left(\frac{x^{i-1}}{(i-1)!}\right)$$

donne une équation pour c_n que l'on résout. On réitère ensuite ce procédé pour trouver les coefficients suivants de Σ_n.

Méthode par relation de récurrence. En remarquant que

$$(x^k)^{(i)} = k\,(k-1)\cdots(k-i+1)\,x^{k-i},$$

et en notant $a_i(x) = \sum_{j=0}^{m_i} a_{i,j}\,x^j$ avec $a_{i,j} \in \mathbb{C}$ les coefficients (polynomiaux) de L, on obtient

$$L(x^k) = \sum_{i=0}^{n} a_i(x)\,(x^k)^{(i)} = \sum_{i=0}^{n}\sum_{j=0}^{m_i} k\,(k-1)\cdots(k-i+1)\,a_{i,j}\,x^{k-i+j},$$

de sorte que

$$L\left(\sum_{k=0}^{+\infty} c_k\,x^k\right) = \sum_{k=0}^{+\infty}\sum_{i=0}^{n}\sum_{j=0}^{m_i} k\,(k-1)\cdots(k-i+1)\,c_k\,a_{i,j}\,x^{k-i+j}.$$

Il en découle que les coefficients c_k d'une série formelle $y(x) = \sum_{k=0}^{+\infty} c_k\,x^k$ solution de $L(y) = 0$ satisfont une relation de récurrence de la forme

$$\sum_{\ell=0}^{M} \alpha_\ell(k)\,c_{k-\ell} = 0.$$

Pour calculer explicitement cette relation de récurrence, on peut procéder comme suit :

1. On décompose l'opérateur différentiel linéaire L sous la forme $L = L_{-A} + \cdots + L_B$ où les opérateurs différentiels linéaires L_i $(i = -A, \ldots, B)$ satisfont

   $$\forall k \in \mathbb{N}, \quad L_i(x^k) = \alpha_i(k)\,x^{k+i},$$

   et les $\alpha_i(k)$ sont des constantes (i.e., ne dépendent pas de la variable x).

2. Si $y(x) = \sum_{k=0}^{+\infty} c_k\,x^k$ est une série formelle solution de $L(y) = 0$, alors on a

   $$\begin{aligned} 0 = L(y) &= L\left(\textstyle\sum_{k=0}^{+\infty} c_k\,x^k\right) \\ &= L_{-A}\left(\textstyle\sum_{k=0}^{+\infty} c_k\,x^k\right) + \cdots + L_B\left(\textstyle\sum_{k=0}^{+\infty} c_k\,x^k\right) \\ &= \textstyle\sum_{k=0}^{+\infty} \alpha_{-A}(k)\,c_k\,x^{k-A} + \cdots + \sum_{k=0}^{+\infty} \alpha_B(k)\,c_k\,x^{k+B} \\ &= \textstyle\sum_{k=0}^{+\infty} \underbrace{\left(\alpha_{-A}(k+A)\,c_{k+A} + \cdots + \alpha_B(k-B)\,c_{k-B}\right)}_{(\mathcal{R})}\,x^k \end{aligned}$$

   et donc les coefficients c_k satisfont la relation de récurrence

   $$(\mathcal{R}) : \alpha_{-A}(k+A)\,c_{k+A} + \cdots + \alpha_B(k-B)\,c_{k-B} = 0.$$

Nous allons maintenant illustrer les trois différentes méthodes abordées ci-dessus sur un exemple.

Exemple 6.1.1. *On considère l'équation différentielle linéaire*

$$L(y) = (1-x^2)\,y''(x) - x\,y(x) = 0,$$

pour laquelle $x_0 = 0$ est un point ordinaire et on cherche à calculer un système fondamental de séries formelles solutions au voisinage de 0.

1. *Appliquons tout d'abord la méthode de Cauchy. On commence par chercher une première solution $y_1(x) = \sum_{i=0}^{+\infty} \frac{y_1^{(i)}(0)}{i!}\,x^i$, avec $y_1(0) = 1$ et $y_1'(0) = 0$. On a alors les relations*

$$y_1''(x) = \frac{x}{1-x^2}\,y_1(x), \quad y_1'''(x) = \frac{1+x^2}{1-x^2}\,y_1(x) + \frac{x}{1-x^2}\,y_1'(x), \quad \ldots$$

qui impliquent $y_1''(0) = 0$, $y_1'''(0) = 1$, ... de sorte que $y_1(x) = 1 + \frac{1}{6}\,x^3 + O(x^4)$. On peut ainsi calculer autant de termes que l'on veut de la série $y_1(x)$. De même, on cherche ensuite une seconde série formelle $y_2(x) = \sum_{i=0}^{+\infty} \frac{y_2^{(i)}(0)}{i!}\,x^i$, solution avec cette fois $y_2(0) = 0$ et $y_2'(0) = 1$. En utilisant les mêmes relations que précédemment on obtient alors $y_2''(0) = 0$, $y_2'''(0) = 0$, ... de sorte que $y_2(x) = x + O(x^4)$.

2. *Appliquons maintenant la méthode d'Euler-Newton. On commence par chercher une première série $y_1(x)$ solution sous la forme $y_1(x) = 1 + x^2\,\Sigma_2(x)$ avec $\Sigma_2 \in \mathbb{C}[[x]]$ à déterminer. En développant l'égalité $L(1 + x^2\,\Sigma_2(x)) = 0$, on obtient alors la nouvelle équation différentielle linéaire*

$$x^2\,(1-x^2)\,\Sigma_2''(x) + 4\,x\,(1-x^2)\,\Sigma_2'(x) + \left(2\,(1-x^2) - x^3\right)\,\Sigma_2(x) = x,$$

pour $\Sigma_2 \in \mathbb{C}[[x]]$. En injectant $\Sigma_2(x) = c_2 + x\,\Sigma_3(x)$ dans cette dernière équation et en regardant le terme de plus bas degré on trouve $2\,c_2 = 0$ de sorte que $c_2 = 0$. Puis, en injectant $\Sigma_3(x) = c_3 + x\,\Sigma_4(x)$ (i.e., $\Sigma_2(x) = c_3\,x + x^2\,\Sigma_4(x)$) et en regardant le terme de plus bas degré on trouve $6\,c_3 = 1$ de sorte que $c_3 = 1/6$. On retrouve finalement la série formelle solution $y_1(x) = 1 + \frac{1}{6}\,x^3 + O(x^4)$. De la même façon, on cherche ensuite une seconde solution sous la forme $y_2(x) = x + x^2\,\overline{\Sigma_2}(x)$ et on retrouve la solution $y_2(x) = x + O(x^4)$.

3. *Appliquons finalement la méthode basée sur la relation de récurrence satisfaite par les coefficients c_k d'une série formelle $y(x) = \sum_{k=0}^{+\infty} c_k\,x^k$ solution de $L(y) = 0$. On décompose tout d'abord L sous la forme $L = L_{-2} + L_0 + L_1$ où*

$$L_{-2}(y) = y'', \quad L_{-2}(x^k) = k\,(k-1)\,x^{k-2},$$

$$L_0(y) = -x^2\,y'', \quad L_0(x^k) = -k\,(k-1)\,x^k,$$

$$L_1(y) = -x\,y, \quad L_1(x^k) = -x^{k+1}.$$

La relation de récurrence satisfaite par les coefficients c_k est alors

$$(\mathcal{R}) : (k+2)\,(k+1)\,c_{k+2} - k\,(k-1)c_k - c_{k-1} = 0.$$

Pour $k = 0$, on obtient $c_2 = 0$, pour $k = 1$, on obtient $6\,c_3 - c_0 = 0$, pour $k = 2$, on obtient $12\,c_4 - 2\,c_2 - c_1 = 0$, ... En prenant comme conditions initiales $c_0 = y_1(0) = 1$ et $c_1 = y_1'(0) = 0$, on obtient $c_3 = 1/6$, $c_4 = 0$ et on retrouve la première solution $y_1(x) = 1 + \frac{1}{6}\,x^3 + O(x^4)$. Alternativement, en prenant comme conditions initiales $c_0 = y_2(0) = 0$ et $c_1 = y_2'(0) = 1$, on obtient $c_3 = 0$, $c_4 = 1/12$ et on retrouve la seconde solution $y_2(x) = x + \frac{1}{12}\,x^4 + O(x^5) = x + O(x^4)$.

6.2 Résolution de $L(y) = 0$ par des séries quasi-entières

6.2.1 Séries quasi-entières

Soit $L(y) = a_n(x)\,y^{(n)}(x) + \cdots + a_1(x)\,y'(x) + a_0(x)\,y(x) = 0$ où pour tout $i = 0, \ldots, n$, $a_i(x) \in \mathbb{C}[x]$. Nous avons vu dans la section précédente que si 0 est un point ordinaire de $L(y) = 0$ (i.e., $a_n(0) \neq 0$), alors nous pouvons construire un système fondamental de séries (formelles) entières solutions dans $\mathbb{C}[[x]]$. En revanche, lorsque 0 est un point singulier de $L(y) = 0$ (i.e., $a_n(0) = 0$), les séries entières forment une classe trop petite c'est-à-dire qu'au voisinage d'un point singulier, il n'est en général pas possible de trouver un système fondamental de solutions sous forme de séries entières. Par exemple, l'équation différentielle linéaire $2\,x\,y'(x) - y(x) = 0$ qui admet pour solution $y(x) = c\,\sqrt{x}$, où c est une constante arbitraire n'admet pas de série entière solution au voisinage de zéro. En effet, au voisinage de zéro nous avons $y(x) = x^{1/2}\,\Sigma(x)$ avec $\Sigma \in \mathbb{C}[[x]]$.

Dans la suite, pour $\lambda \in \mathbb{C}$ nous noterons x^λ la solution de $y'(x) = \frac{\lambda}{x}\,y(x)$ telle que $y(1) = 1$ et $\log(x)$ la solution de $y'(x) = \frac{1}{x}$ telle que $y(1) = 0$. Nous avons alors $x^\lambda = \exp(\lambda\,\log(x))$. Si $k \in \mathbb{N}$, alors nous identifierons $x^k\,x^\lambda = x^{\lambda+k}$. Enfin, nous avons les dérivées par rapport à x (resp. λ) suivantes :

$$\frac{dx^\lambda}{dx} = \lambda\,x^{\lambda-1}, \quad \frac{dx^\lambda}{d\lambda} = \log(x)\,x^\lambda.$$

Remarque 6.2.1. *On peut vérifier que si $\lambda \notin \mathbb{N}$, alors $y'(x) = \frac{\lambda}{x}\,y(x)$ n'admet pas de solution sous forme de série entière (au voisinage de 0).*

Définition 6.2.1. *On appelle* série quasi-entière *un objet de la forme $x^\lambda\,\Sigma(x)$ où $\lambda \in \mathbb{C}$ et $\Sigma \in \mathbb{C}[[x]]$ est une série entière. Si le terme constant de Σ est non nul (i.e., $\Sigma(0) \neq 0$), alors λ est appelé* l'ordre *de la série quasi-entière $x^\lambda\,\Sigma(x)$.*

Par exemple les *séries de Laurent* $\sum_{k=-k_0}^{+\infty} c_k\,x^k = x^{-k_0}\,\sum_{k=0}^{+\infty} c_{-k_0+k}\,x^k$ sont des séries quasi-entières. On appelle *série de Puiseux* une série de Laurent en $x^{1/r}$, i.e., $\sum_{k=-k_0}^{+\infty} c_k\,x^{k/r}$. Une série quasi-entière $x^\lambda\,\Sigma(x)$ avec $\lambda \in \mathbb{Q}$ est donc une série de Puiseux.

6.2.2 Équation indicielle et calcul des séries quasi-entières solutions

Soit $L = L_{-A} + \cdots + L_B$ la décomposition de L telle que les opérateurs différentiels linéaires L_i $(i = -A, \ldots, B)$ satisfont

$$\forall k \in \mathbb{N}, \quad L_i(x^k) = \alpha_i(k)\, x^{k+i}.$$

Définition 6.2.2. *Avec les notations précédentes, le polynôme $E_0(X) = \alpha_{-A}(X)$ est appelé* polynôme indiciel de L en 0 *et l'équation $E_0(X) = 0$ est appelée* équation indicielle.

En injectant une série quasi-entière $y(x) = x^\lambda \sum_{k=0}^{+\infty} c_k\, x^k$ dans l'équation $L(y) = 0$, on montre que :

$$L\left(x^\lambda \sum_{k=0}^{+\infty} c_k\, x^k\right) = 0 \Leftrightarrow \left\{ \begin{array}{l} E_0(\lambda) = 0, \\ \forall k \in \mathbb{N},\ E_0(\lambda + k + A)\, c_{k+A} + \cdots + \alpha_B(\lambda + k - B)\, c_{k-B} = 0. \end{array} \right. \tag{6.2}$$

Théorème 6.2.1. *Soit $L(y) = a_n(x)\, y^{(n)}(x) + \cdots + a_1(x)\, y'(x) + a_0(x)\, y(x) = 0$ où pour tout $i = 0, \ldots, n$, $a_i(x) \in \mathbb{C}[x]$.*

1. *L'équation $L(y) = 0$ admet une série quasi-entière solution d'ordre λ seulement si $E_0(\lambda) = 0$.*

2. *Si λ est une racine de $E_0(X) = 0$ et si, pour tout $k \in \mathbb{N}^*$, $E_0(\lambda + k) \neq 0$, alors l'équation $L(y) = 0$ admet une série quasi-entière solution d'ordre λ.*

Démonstration. Ceci découle de l'équivalence (6.2). Pour 2, si $E_0(\lambda + k) \neq 0$ pour tout $k \in \mathbb{N}^*$, alors on peut utiliser la dernière égalité de (6.2), pour calculer successivement chaque coefficient c_{k+A}. □

Les racines de l'équation indicielle $E_0(X) = 0$ sont appelées *exposants de L en* 0. Lorsque 0 est un point ordinaire de $L(y) = 0$, alors avec les notations précédentes, on a $A = n$ et $L_{-A}(x^k) = a\, k\, (k-1) \cdots (k-n+1)\, x^{k-n}$ pour une certaine constante non nulle $a \in \mathbb{C}$. L'équation indicielle est alors donnée par $E_0(X) = a\, X\, (X-1) \cdots (X-n+1)$ et nous avons les n exposants entiers $0, 1, \ldots, n-1$ de sorte que l'on retrouve un système fondamental de séries entières solutions (voir section précédente). Lorsque 0 est un point singulier, alors le degré du polynôme indiciel $E_0(X)$ est au plus égal à n. Dans le cas où $\deg(E_0(X)) = n$, on dit que 0 est un *point singulier régulier* de $L(y) = 0$.

Nous terminons ce chapitre avec deux exemples qui illustrent le calcul d'un système fondamental de séries solutions au voisinage d'un point singulier (ici au voisinage de 0).

Exemple 6.2.1. *Considérons* l'équation d'Euler d'ordre 2 *donnée par*

$$L(y) = x^2\, y''(x) + a\, x\, y'(x) + b\, y(x) = 0, \quad a, b \in \mathbb{C}.$$

La décomposition de L s'écrit alors simplement $L = L_0$ car nous avons

$$L(x^k) = (k\,(k-1) + a\,k + b)\; x^k.$$

L'équation indicielle est donc $E_0(X) = X\,(X-1) + a\,X + b = 0$ et nous devons distinguer deux cas :

1. *Cas 1 : $E_0(X)$ a deux racines distinctes λ_1 et λ_2. Dans ce cas nous avons deux séries quasi-entières solutions x^{λ_1} et x^{λ_2} (qui sont linéairement indépendantes).*

2. *Cas 2 : $E_0(X)$ a une racine double λ. Dans ce cas nous avons une série quasi-entière solution x^λ. Pour construire la deuxième solution nous considérons l'équation $L(x^k) = E_0(k)\,x^k$. Comme les dérivations d/dx et d/dk commutent, nous avons*

$$L\left(\frac{dx^k}{dk}\right) = \frac{d}{dk}\left(E_0(k)\,x^k\right) = E_0(k)'\,x^k + E_0(k)\,\log(x)\,x^k,$$

ou encore

$$L\left(\log(x)\,x^k\right) = (E_0(k)' + E_0(k)\,\log(x))\;x^k.$$

Comme λ est une racine double de $E_0(X)$, nous avons $E_0(\lambda) = E_0'(\lambda) = 0$ ce qui prouve que $\log(x)\,x^\lambda$ est solution de $L(y) = 0$. Clairement x^λ et $\log(x)\,x^\lambda$ sont linéairement indépendantes donc nous avons trouvé un système fondamental de solutions.

Exemple 6.2.2. *Considérons l'équation*

$$L(y) = x\,y''(x) + \frac{3}{2}\,y'(x) + y(x) = 0.$$

Nous avons $L = L_{-1} + L_0$ avec

$$L_{-1}(y) = x\,y'' + \frac{3}{2}\,y', \quad L_{-1}(x^k) = \left(k\,(k-1) + \frac{3}{2}\,k\right)\;x^{k-1},$$

$$L_0(y) = y, \quad L_0(x^k) = x^k.$$

L'équation indicielle est alors

$$E_0(X) = X\,(X-1) + \frac{3}{2}\,X = X\,\left(X + \frac{1}{2}\right).$$

Cette équation indicielle admet donc deux racines distinctes $\lambda_1 = 0$ et $\lambda_2 = -1/2$ qui ne diffèrent pas d'un entier. Le théorème 6.2.1 implique donc que nous avons une série entière $\Sigma_1 \in \mathbb{C}[[x]]$ solution et l'autre solution est une série quasi-entière $x^{-1/2}\,\Sigma_2(x)$ avec $\Sigma_2 \in \mathbb{C}[[x]]$. La relation de récurrence pour les coefficients c_k des séries Σ_1 et Σ_2 est donnée par

$$E_0(\lambda + k + 1)\,c_{k+1} + c_k = 0.$$

Pour chacune des séries Σ_1 et Σ_2, en se donnant la condition initiale c_0, on peut alors calculer c_1, $c_2, \ldots$ par la formule

$$c_{k+1} = \frac{-c_k}{E_0(\lambda + k + 1)},$$

puisqu'à la fois pour $\lambda = \lambda_1 = 0$ et $\lambda = \lambda_2 = -1/2$, nous avons $E_0(\lambda + k + 1) \neq 0$, pour tout $k \in \mathbb{N}$.

Remerciements

Ce cours a été essentiellement préparé à partir :

1. des notes de cours de calcul formel dispensés les années précédentes à l'Université de Limoges par M. A. Barkatou, J.-A. Weil et d'autres enseignants-chercheurs ;

2. des références [1, 3].

Bibliographie

[1] A. Bostan, F. Chyzak, M. Giusti, R. Lebreton, G. Lecerf, B. Salvy et É. Schost *Algorithmes Efficaces en Calcul Formel.* Notes du cours 2-22 du Master Parisien de Recherche en Informatique (MPRI), Version du 26/01/2016.

[2] J. von zur Gathen et J. Gerhard. *Modern computer algebra.* Cambridge University Press, New York, 1999.

[3] J.-P. Marco, P. Thieullen et J.-A. Weil. *Mathématiques L2 (Chapitre 5)* - Cours complet avec 700 tests et exercices corrigés, 838 pages. Pearson Education France, 2007.

[4] M. Mignotte. *Mathématiques pour le calcul formel.* Presses Universitaires de France, 1989.