N° Ordre : 2158
de la thèse

THÈSE

présentée

DEVANT L'UNIVERSITÉ DE RENNES 1

pour obtenir

le grade de : **DOCTEUR DE L'UNIVERSITÉ DE RENNES 1**

Mention : Mathématiques et applications

PAR

Frédéric LEHOBEY

Équipe d'accueil : **IRMAR – UMR CNRS 6625 – Rennes**

École Doctorale : **Mathématiques de l'Ouest**

Composante universitaire : **Institut de Mathématiques de Rennes**

TITRE DE LA THÈSE :

Calcul et factorisation interactive de résolvantes de Lagrange en théorie de Galois effective

Soutenue le 17 septembre 1999 devant la Commission d'Examen

COMPOSITION DU JURY :

Michel OLIVIER	**Rapporteur, Président**
James H. DAVENPORT	**Rapporteur**
Paul ZIMMERMANN	**Rapporteur**
Michel PETITOT	**Examinateur**
Félix ULMER	**Examinateur**
Marie-Françoise ROY	**Directrice**
Annick VALIBOUZE	**Directrice**

Remerciements

Je remercie James H. Davenport, Michel Olivier, Paul Zimmermann de m'avoir fait l'honneur d'accepter d'évaluer ce travail. J'ai tiré grand profit de leurs observations.

Je remercie Félix Ulmer et Michel Petitot d'avoir accepté de faire partie de ce jury. J'ai apprécié leur sympathique compagnie et leur compétence au sein des laboratoires où je les ai fréquentés.

Je remercie tout particulièrement Marie-Françoise Roy et Annick Valibouze pour le privilège que j'ai eu de les avoir comme directrices dans ce travail. Elles savent trop ce que je leur dois. Merci pour leur disponibilité, leur expérience et leur enthousiasme communicatif.

Je dois aussi beaucoup à tous les collègues qui ont fréquenté le projet Galois, Isabelle Gil Delesalle, Jean-Marie Arnaudiès, Antoine Colin, Lionel Ducos, Inès Abdeljaoued et tout particulièrement à Nicolas Rennert pour notre collaboration amicale et fructueuse.

Je remercie tous les autres collègues rencontrés dans les divers laboratoires que j'ai fréquentés au cours de cette thèse : ceux de l'IRMAR (Rennes), du LIP6 (Paris) et du LIFL (Lille) et plus spécifiquement ceux des équipes de calcul formel de ces laboratoires. Merci aussi à tous les personnels de ces laboratoires, secrétaires, administrateurs système, bibliothécaires, dont la présence et la compétence sont de grande valeur dans l'exercice quotidien de la recherche.

La réalisation de cette thèse a été possible grâce aux soutiens financiers et matériels du CNRS, de la région Bretagne, du Ministère de l'Éducation Nationale de la Recherche et de la Technologie et de l'UMS MEDICIS CNRS 658.

Je remercie tous les amis et ma famille qui ont fait preuve d'une grande patience et indulgence à mon égard au cours de ces années industrieuses. Merci aussi à tous ceux qui m'ont offert leur toit, ma porte leur sera toujours ouverte et mon amitié acquise.

Cette thèse est dédiée à tous ceux qui n'auraient pu assister à sa soutenance, même s'ils l'avaient voulu, empêchés qu'ils en étaient par notre loi.

Table des matières

Introduction

Pour déterminer le groupe de Galois d'un polynôme f, la théorie de Galois effective utilise la factorisation de polynômes déduits du polynôme f, les résolvantes de Lagrange, à même corps des coefficients que le polynôme f.

Nous améliorons les méthodes de calcul symbolique des résolvantes basées sur le résultant.

Nous montrons aussi comment adapter les algorithmes de la factorisation des polynômes pour utiliser les informations connues *a priori* sur les résolvantes (elles viennent de la théorie des groupes).

La factorisation étant pour nous un moyen et non un but, nous introduisons le concept de factorisation interactive pour rendre immédiatement accessibles les nouvelles informations sur la factorisation des résolvantes trouvées au cours du processus de factorisation.

Ce concept n'est pas spécifique à la théorie de Galois effective ni à la factorisation de résolvantes. Il peut être utilisé pour toute factorisation de polynômes pour lesquels l'information cherchée ne demande pas forcément une factorisation complète.

Historique schématique des travaux en théorie de Galois effective

Soit f un polynôme de degré n qui nous est connu par ses coefficients rationnels.

Le « théorème fondamental de l'algèbre » (ou théorème de d'Alembert) nous dit qu'il s'annule en exactement n points du plan complexe, ses racines.

Un problème ouvert de l'antiquité jusqu'au XIX$^{\text{e}}$ siècle était d'exprimer les racines du polynôme f en fonction de ses coefficients grâce à des fonctions « simples », en l'occurrence les opérations algébriques classiques (addition, soustraction, multiplication, division), auxquelles est adjointe l'extraction de racine.

Les solutions à ce problème sont appelées *résolutions par radicaux de l'équation* $f = 0$ et des formules de cette nature étaient connues depuis l'antiquité pour $n = 1$ et $n = 2$. Cardan et Tartaglia ont donné des solutions pour $n = 3$ et Ferrari pour $n = 4$.

Au XIX$^{\text{e}}$ siècle, Abel et Ruffini ont répondu négativement à la question pour $n = 5$. É. Galois a donné, en forgeant la notion de groupe, un critère valable **pour tout** n qui indique si un polynôme a ou non ses racines exprimables par radicaux.

L'idée de É. Galois, inspiré par J. L. Lagrange, consiste à étudier les relations entre les racines du polynôme f. Cela revient à étudier les expressions laissées invariantes par échange de racines.

Les échanges de racines correspondent à des permutations du groupe symétrique sur n éléments $\mathfrak{S}_n$. L'ensemble des permutations laissant invariantes **toutes** les relations algébriques

entre les racines forme un groupe. Ce groupe est appelé maintenant groupe de Galois du polynôme f. Si, et seulement si, ce groupe jouit de certaines propriétés (s'il est résoluble) les racines du polynôme f pourront être exprimées par radicaux en fonction de ses coefficients.

Au-delà de la résolubilité par radicaux, des polynômes de même groupe de Galois partagent les mêmes propriétés à l'égard de leurs racines et les mêmes méthodes de résolution pourront leur être appliquées.

La détermination du groupe de Galois permet donc de choisir la méthode la plus adaptée à la forme de résolution cherchée ou de répondre directement à certaines questions sur les propriétés du polynôme.

Cependant, si la question de détermination du groupe de Galois est théoriquement résolue, les calculs symboliques qui la sous-tendent sont impraticables et Galois lui-même refusait de s'y soumettre (en introduction de son mémoire à l'Académie des Sciences) :

> Si maintenant vous me donnez une équation que vous aurez choisie à votre gré, et que vous désiriez connaître si elle est ou non résoluble par radicaux, je n'aurai rien à y faire que de vous indiquer le moyen de répondre à votre question, sans vouloir charger ni moi ni personne de le faire. En un mot les calculs sont impraticables.

Sont apparus depuis É. Galois ces esclaves serviables du mathématicien que sont les ordinateurs et les logiciels de calcul formel (ou symbolique).

La théorie de Galois **effective** cherche, entre autre buts, à faire déterminer le groupe de Galois par ces outils modernes en introduisant une notion d'efficacité : nous souhaitons obtenir les résultats de l'algorithme de notre vivant et si possible dans un délai raisonnable (qui dépend de la patience de l'interrogateur et de la puissance des moyens de calcul mis à sa disposition).

L'outil utilisé par É. Galois pour calculer (inefficacement) le groupe qui porte son nom a été introduit par J. L. Lagrange [59]. Il s'agit d'un polynôme à coefficients dans le même corps que f : la **résolvante** de Galois. Sa factorisation permet de déterminer le corps de décomposition du polynôme f (comme extension algébrique de $\mathbb{Q}$) et donc le groupe de Galois de f. Il s'agit malheureusement d'un polynôme de degré $n!$ dont la factorisation, ou même la manipulation, devient très vite impossible pour les systèmes de calcul formel.

En fait la résolvante, introduite par J. L. Lagrange et reprise dans [14], [114] et [60], peut être calculée pour un sous-groupe H de $\mathfrak{S}_n$. Dans ce cas, le degré de la résolvante est alors l'indice de H dans $\mathfrak{S}_n$. Pour des sous-groupes de faible indice, sa factorisation redevient accessible au calcul formel.

R. P. Stauduhar a introduit dans [91] la notion de résolvante relative. Il y a montré comment calculer les résolvantes (relatives et absolues) par calcul numérique à partir de valeurs approchées dans $\mathbb{C}$ des racines. Malheureusement, l'information obtenue sur ces autres résolvantes ne porte que sur l'inclusion ou non du groupe de Galois dans le sous-groupe H. Il devient alors indispensable de connaître auparavant la classification des sous-groupes de $\mathfrak{S}_n$. Cette classification a été effectuée jusqu'au degré 31 pour les sous-groupes transitifs, qui correspondent aux polynômes irréductibles, grâce aux avancées algorithmiques de la théorie des groupes des années 1960 à nos jours [22], [23] et [34].

Y. Eichenlaub et M. Olivier ont prolongé la méthode de Stauduhar jusqu'au degré 11 (voir [42] et [41]).

J. MCKAY et L. SOICHER [70] et [89] ont proposé une méthode symbolique de calcul des résolvantes pour certains sous-groupes de $\mathfrak{S}_n$, les groupes de Young, auxquels sont associés des invariants linéaires. Ils appliquent leur méthode jusqu'au degré 7. C'est cet algorithme qui est implanté dans le logiciel de calcul formel `Maple`. C. WILLIAMSON a réalisé un paquetage `AXIOM` pour traiter les degrés $1, 2, 3, 4, 5, 7$. Nous avons ajouté à ce paquetage le cas du degré 6 [61].

A. COLIN [29] et [30] a donné une méthode permettant de calculer formellement les résolvantes relatives de Stauduhar.

J.-M. ARNAUDIÈS et A. VALIBOUZE [95] ont transformé en méthode déterministe ce qui n'était jusque là qu'une heuristique (utilisée aussi par S. J. BERWICK dans [14]). Ils produisent des matrices des degrés de toutes les factorisations de toutes les résolvantes possibles. La connaissance des seuls degrés des facteurs de résolvantes suffit à distinguer **tous** les groupes de Galois possibles.

L'utilisation du groupe de Galois des facteurs des résolvantes et le calcul de l'idéal des relations ont été ensuite ajoutés par A. VALIBOUZE [96] et [100].

De nombreux travaux récents[1] en théorie de Galois effective ont été réalisés. Citons, sans prétendre à l'exhaustivité, les travaux menés à Bordeaux autour de M. OLIVIER, à Heidelberg autour de B. MATZAT, au Japon autour de K. YOKOYAMA, à Montréal autour de J. MCKAY, à Poitiers autour de C. QUITTÉ et à Paris autour d'A. VALIBOUZE.

Cette thèse s'inscrit dans la démarche de recherche du groupe de Galois en calculant des résolvantes symboliquement et en les factorisant.

Présentation des chapitres

Chapitre 1. Ce chapitre rappelle des résultats classiques en théorie de Galois et introduit des notations de J.-M. ARNAUDIÈS et A. VALIBOUZE.

Chapitre 2. Ce chapitre décrit les outils utilisés pour manipuler des polynômes avec en particulier une revue de l'« état de l'art » sur le calcul de racine r-ième de polynôme.

Chapitre 3. Nous donnons dans ce chapitre 3 (théorème 3.9) une description algorithmique de la méthode de Lagrange-Soicher pour calculer les résolvantes. Nous y avons ajouté l'utilisation de calculs sur les polynômes réciproques et l'échange des calculs de résultants et de quotients présents dans cette méthode. Ces deux points contribuent à contenir les coûteuses manipulations symboliques inhérentes à cette méthode. Nous avons caractérisé une classe d'invariants (les invariants simples) pour lesquels les calculs symboliques sont réduits.

Chapitre 4. Le chapitre 4 dont le contenu a fait l'objet de la communication [62] applique les idées du chapitre 3 à la méthode de Valibouze-Rennert de calcul des résolvantes par les

1. Des pointeurs sur les travaux de ces groupes peuvent être trouvés sur la page du Projet Galois de l'UMS MEDICIS CNRS 658 : `http://marie.medicis.polytechnique.fr/medicis/projetGalois/`

modules de Cauchy (théorème 4.4). Nous améliorons cette méthode dans le cas des invariants présentant des symétries[2] (cas le plus fréquent).

Chapitre 5. Ce chapitre présente un algorithme général de factorisation en détaillant les multiples occasions d'apporter une information discriminante pour la recherche du groupe de Galois ainsi que les possibilités d'utilisation des informations *a priori* sur les factorisations possibles. Il introduit notre concept de factorisation interactive (paragraphe 5.6) destinée à dialoguer avec un algorithme de recherche du groupe de Galois et à éviter de mener plus de calculs que nécessaire à la recherche de l'information discriminante.

Chapitre 6. Ce chapitre décrit les nouveaux outils utilisés pour l'implantation d'une factorisation partielle interactive :

- un ordre sur les mots de n lettres dans l'alphabet $\{0, 1\}$
- une reformulation du problème de remontée dans la méthode de Collins et Encarnación en utilisant l'ordre défini ci-dessus pour éviter des calculs redondants.

Cette reformulation est nécessaire lors de l'utilisation de bornes ou d'heuristiques qui ne garantissent pas l'irréductibilité des facteurs trouvés. Dans la méthode de coût exponentiel dans le pire des cas (énumération des facteurs associés aux partitions possibles) notre ordre permet d'éviter certains calculs redondants sur les facteurs trouvés.

Chapitre 7. Ce chapitre propose une implantation en `AXIOM` de la factorisation interactive. Pour cela nous avons dû adapter tous les sous-algorithmes de la factorisation (chapitre 5) à la factorisation interactive.

Annexes A et B. Exemples de calculs de résolvantes et d'utilisations du factorisateur interactif.

Bibliographie. Elle ne contient que les travaux cités dans le texte.

Extensions possibles

Nous avons pris beaucoup de plaisir à travailler dans le cadre de la théorie de Galois effective et sur la factorisation en raison du carrefour que ces deux thèmes représentent entre théorie des groupes, théorie des nombres et théorie de l'élimination. Il est toujours stimulant de constater, comme fréquemment en sciences, comment une meilleure compréhension de l'une de ces théories fournit des outils fertiles dans les autres (pensons aux critères d'irréductibilité de polynômes, au théorème de Čebotarev effectif, aux algorithmes de décomposition polynomiale fonctionnelle).

2. En utilisant la suggestion de A. Valibouze et N. Rennert nous avons calculé pour le problème de Galois inverse trois polynômes de degré 12 qui apparaissent comme facteurs de résolvantes dont les calculs n'aboutissaient pas jusqu'alors (juin 1997).

Finir une thèse c'est aussi accepter de rester sur sa faim pour toutes les pistes qu'il ne nous a pas encore été possible d'explorer. Par exemple : la généralisation possible de notre factorisation interactive à la factorisation sur des extensions algébriques des rationnels, sur d'autres corps, parfaits ou de caractéristique nulle sur lesquels tiennent les méthodes de la théorie de Galois effective, ou sur des anneaux de polynômes en plusieurs variables pour lesquels le schéma de factorisation basé sur un lemme de Hensel effectif est analogue ; la complexité minimale de la composition polynomiale modulaire (calculer $g \circ h \bmod f \bmod p$) qui est au cœur de la présentation de V. SHOUP et n'est pas précisément connue.

Notre concept de factorisation interactive est adaptable par nature à tout progrès dans la factorisation ou à toute modification de ses algorithmes internes[3] (transparente pour l'utilisateur). L'algorithme de [63] pourrait y être introduit. Il est possible de l'adapter à la recherche de vrais facteurs (non nécessairement irréductibles). Le modulo à atteindre pour cette méthode en sera d'autant réduit que la borne pour trouver des vrais facteurs sera petite.

Les algorithmes de décomposition polynomiale nous incitent à leur trouver des pendants non-commutatifs (décomposition d'opérateurs différentiels).

La compatibilité de la méthode du chapitre 3 avec le calcul modulaire (paragraphe 3.7) pourrait peut-être être plus exploitée. La généralité du lemme 2.26 laisse espérer son utilisation dans d'autres contextes.

Les résultats du chapitre 4 semblent pouvoir être généralisés aux idéaux triangulaires apparaissant dans [7] pour le calcul de résolvantes relatives.

Les chapitres 3 et 4 font appel à des calculs de résultants sur des anneaux non-intègres très particuliers, de la forme $A[t]/(t^{s+1})$, pour lesquels une adaptation d'un algorithme de la nature de celui des sous-résultants serait intéressante.

La reformulation du problème de remontée n'est pas spécifique à la technique de réconciliation de coût exponentiel dans le pire des cas. Elle pourrait tout autant être utilisée dans une méthode du type [63].

Le concept de factorisation interactive n'est pas spécifique à la théorie de Galois effective, même si celle-ci s'y prête particulièrement. Il peut être utilisé dans toutes les situations où sont connues des informations *a priori* sur la factorisation attendue. C'est le cas de la factorisation des normes de polynômes sur une extension algébrique qui sont utilisées dans le méthode de B. M. TRAGER [93] pour factoriser sur des extensions algébriques. C'est aussi, d'après J. H. DAVENPORT, le cas de systèmes issus de calculs de base de Gröbner.

3. Afin de favoriser l'utilisation future de nos paquetages, nous souhaitons les voir traduits en `Aldor` (ce pour quoi ils ont été pensés) ou tout autre langage ou logiciel de calcul formel.

Chapitre 1

Outils de la théorie de Galois effective

Ce chapitre est consacré à la définition des notions et notations utilisées dans les chapitres 2, 3 et 4. Il ne contient que des résultats classiques ou exposés dans les travaux de J.-M. ARNAUDIÈS et A. VALIBOUZE [6].

Il les présente sous une forme inspirée par [30], [97], et [5].

1.1 Groupes

Les lettres G, H, L désignent des groupes finis dont la loi interne est notée multiplicativement par le symbole « . » ou par l'absence de symbole entre les éléments.

Notation 1.1. Soit $\mathcal{E}$ un ensemble fini ou G un groupe fini. Le *cardinal* de $\mathcal{E}$ ou l'*ordre* de G sont notés $\mathrm{card}\,(\mathcal{E})$ et $\mathrm{card}\,(G)$.

Notation 1.2. Soit $\mathcal{E}$ un ensemble fini. Le groupe des bijections sur $\mathcal{E}$ muni de la composition pour loi interne est noté $(\mathfrak{S}_{\mathcal{E}}, \circ)$.

Définition 1.3. Soit n un entier strictement positif, le *groupe* $\mathfrak{S}_n$ *des permutations d'ordre* n est défini par :

$$\mathfrak{S}_n = \mathfrak{S}_{\{1,\ldots,n\}}$$

Définition 1.4. Soient G un groupe fini muni de la loi interne « . » et n un entier strictement positif. Une *représentation symétrique de degré* n du groupe G est un morphisme de groupe $\rho : (G,.) \longrightarrow (\mathfrak{S}_n, \circ)$.

Définition 1.5. Soient G un groupe fini, n un entier strictement positif et ρ une représentation symétrique de degré n de G. La représentation est dite *fidèle* si le morphisme ρ est injectif.

Notation 1.6. Soit H un sous-groupe de G. La *classe de conjugaison* de H dans G est notée $[H]_G$, ou simplement $[H]$ lorsqu'il n'y a pas d'ambiguïté :

$$[H]_G = \left\{\sigma H \sigma^{-1} \mid \sigma \in G\right\} \quad .$$

Notation 1.7. Soit H un sous-groupe de G. L'ensemble des classes à gauche de H dans G est noté G/H.

L'*indice* de H dans G, noté $[G : H]$, est le cardinal de l'ensemble G/H.

1.2 Partitions et invariants

Notation 1.8. Soit n un entier strictement positif, $\mathbb{N}^n$ est muni de l'ordre produit, noté $\preceq$, induit par l'ordre de $\mathbb{N}$.

$$m = (m_1, \dots m_n) \preceq (m'_1, \dots, m'_n) = m'$$

si et seulement si $m_i \leq m'_i$ pour $1 \leq i \leq n$.

Définition 1.9. Soit n un entier strictement positif. Une *partition de* n (au sens combinatoire) est un n-uplet $\varpi = (m_1, \dots, m_n) \in \mathbb{N}^n$ vérifiant

$$\sum_{i=1}^{n} m_i \; i = n \quad .$$

Nous notons cette partition ϖ sous la forme multiplicative :

$$\varpi = \prod_{i=1}^{n} i^{m_i} \quad .$$

Hypothèse 1.10. Soient G et H des sous-groupes de $L \subset \mathfrak{S}_n$.

Nous supposons l'hypothèse 1.10 vérifiée pour tous les groupes G, H et L rencontrés dans le paragraphe 1.2.

Définition 1.11. En faisant agir G sur L/H par translation à gauche, soit o_i le nombre d'orbites de cardinal i pour cette action et $\varpi^L(H, G)$ la partition de $[L : H]$ qui leur correspond :

$$\varpi^L(H, G) = (o_1, \dots, o_{[L:H]}) \quad .$$

Proposition 1.12 (Arnaudiès-Valibouze). *La partition $\varpi^L(H, G)$ ne dépend que des classes de conjugaison $[H]_L$ et $[G]_L$.*

Preuve : Voir [6, proposition 10]. □

Notation 1.13. En conséquence pour les classes de conjugaison $[H]_L$ et $[G]_L$ nous pouvons noter :

$$\varpi^L([H]_L, [G]_L) = \varpi^L(H, G) \quad .$$

Notation 1.14. En choisissant une numérotation $[H_i]_L$ des s classes de conjugaison distinctes des sous-groupes de L où les H_i, avec $1 \leq i \leq s$ sont des représentants de chacune des classes de conjugaison (typiquement, en les ordonnant par indice dans L décroissant) nous notons $\mathcal{P}_L$ la *matrice des partitions de* L :

$$\mathcal{P}_L = \left(\varpi^L([H_i]_L, [H_j]_L)\right)_{1 \leq i,j \leq s} \quad .$$

Théorème 1.15 (Arnaudiès-Valibouze). *Les lignes de la matrice $\mathcal{P}_L$ des partitions de L sont distinctes.*

Preuve : Voir [6, théorème 14]. □

1.3 Polynômes et anneaux de polynômes

Hypothèse 1.16. A désigne un anneau intègre et K son corps des fractions est supposé parfait.

f est un polynôme en une variable, unitaire, **séparable** (c'est à dire sans racine multiple), de degré n à coefficients dans A :

$$f(x) \in A[x] \quad .$$

$\hat{K}$ est une extension algébrique du corps K contenant les racines distinctes de f notées $\alpha_1, \ldots, \alpha_n$.

Notation 1.17. Une *numérotation fixée* des racines distinctes de f dans une extension algébrique $\hat{K}$ fixée de K contenant ces racines est notée Ω_f, ou Ω quand il n'y a pas d'ambiguïté :

$$\Omega_f = (\alpha_1, \ldots, \alpha_n) \in \hat{K}^n \quad .$$

Notation 1.18. Pour un polynôme $\Theta \in A[x_1, \ldots, x_n]$, l'*évaluation* du polynôme Θ en les racines du polynôme f est notée

$$\Theta(\Omega_f) = \Theta(\alpha_1, \ldots, \alpha_n) \in \hat{K} \quad .$$

Définition 1.19. L'*action naturelle* notée « . » de $\mathfrak{S}_n$ sur $A[x_1, \ldots, x_n]$ est définie par :
soient $\Theta \in A[x_1, \ldots, x_n]$ et $\sigma \in \mathfrak{S}_n$,

$$(\sigma.\Theta)(x_1, \ldots, x_n) = \Theta\left(x_{\sigma(1)}, \ldots, x_{\sigma(n)}\right) \quad .$$

Définition 1.20. Soient $\Theta \in A[x_1, \ldots, x_n]$ et L un sous-groupe de $\mathfrak{S}_n$. L'*orbite* $L.\Theta$ *de* Θ *sous l'action de* L, est définie par :

$$L.\Theta = \{\sigma.\Theta \mid \sigma \in L\} \quad .$$

Définition 1.21. Soit H un sous-groupe de $\mathfrak{S}_n$. Un polynôme $\Theta \in A[x_1, \ldots, x_n]$ est un *invariant* de H (ou un *H-invariant*) si

$$H.\Theta = \{\Theta\} \quad .$$

Notation 1.22. Le A-module des H-invariants est noté $A[x_1, \ldots, x_n]^H$

Notation 1.23. Soit L un sous-groupe de $\mathfrak{S}_n$. Le *stabilisateur* sur L du polynôme Θ de $A[x_1, \ldots, x_n]$ est défini par :

$$\mathrm{Stab}_L(\Theta) = \left\{\sigma \in L \mid \Theta\left(x_1, \ldots, x_n\right) = \Theta\left(x_{\sigma(1)}, \ldots, x_{\sigma(n)}\right)\right\} \quad .$$

Définition 1.24. Soient L un sous-groupe de $\mathfrak{S}_n$ et H un sous-groupe de L. Un polynôme Θ de $A[x_1, \ldots, x_n]$ est un *H-invariant primitif L-relatif* (ou *H-invariant L-primitif*) si :

$$H = \mathrm{Stab}_L\Theta \quad .$$

Notation 1.25. Un H-invariant primitif $\mathfrak{S}_n$-relatif est appelé *H-invariant primitif (absolu).*

Remarque : La justification de la terminologie *invariant primitif* peut être trouvée dans [6, définitions 2 et 18]. Les H-invariants primitifs peuvent être calculés par les algorithmes dans [44] ou [2].

Définition 1.26. Soit n un entier et Θ un polynôme de $A[x_1, \ldots, x_n]$. L'*arité* de Θ est le nombre de variables effectivement présentes dans l'expression de Θ.

Les variables d'un invariant $\Theta \in A[x_1, \ldots, x_n]$ d'arité k peuvent être renumérotées de façon à ce que Θ s'écrive comme un élément de $A[x_1, \ldots, x_k]$. Lorsque nous parlons d'un invariant Θ d'arité k nous supposons donc que $\Theta \in A[x_1, \ldots, x_k]$.

1.4 Groupes de Galois

Nous nous plaçons pour ce paragraphe dans le cadre de l'hypothèse 1.16.

Remarque : Le corps de décomposition de f est $K[\Omega_f] \subset \hat{K}$.

Définition 1.27. Le *groupe de Galois* de f sur le corps K, noté $\mathrm{Gal}_K(f)$, est classiquement défini comme le groupe des K-automorphismes de $K[\Omega_f]$.

Définition 1.28. La représentation symétrique du groupe de Galois associé à la numérotation des racines Ω_f :

$$\begin{array}{rccc} \rho & : & \mathrm{Gal}_K(f) & \longrightarrow & \mathfrak{S}_n \\ & & g & \longmapsto & \sigma_g \end{array}$$

avec $g(\alpha_i) = \alpha_{\sigma_g(i)}$ pour $1 \leq i \leq n$ est notée $\Gamma = \rho(\mathrm{Gal}_K(f)) \subset \mathfrak{S}_n$.

Remarque : Cette représentation est fidèle.

Propriété 1.29. *Modifier la numérotation des racines Ω_f revient à prendre pour représentation symétrique de $\mathrm{Gal}_K(f)$ un conjugué de Γ dans $\mathfrak{S}_n$.*

Corollaire 1.30. *La classe de conjugaison $[\Gamma]_{\mathfrak{S}_n}$ ne dépend que du groupe de Galois $\mathrm{Gal}_K(f)$ et non de la numérotation fixée Ω_f des racines de f.*

Que signifie donc « déterminer le groupe de Galois » de f ?

Pour une numérotation **fixée** des racines Ω_f, le groupe de Galois de f est connu comme la représentation décrite par la définition 1.28.

Posséder une numérotation des racines de f signifie pour un ordinateur qu'il connaît une représentation des ces racines qui permet de les distinguer deux à deux.

Deux représentations classiques sont les représentations des racines par des nombres en virgule flottante de précision arbitraire ou comme des combinaisons linéaires dans la base formée par les puissances d'une racine du polynôme minimal d'un élément primitif du corps de décomposition du polynôme f.

Représentation numérique. Cette représentation consiste à représenter les racines par des valeurs numériques complexes approchées avec une précision fixée et arbitrairement grande.

L'intérêt de cette représentation est de souvent permettre des calculs très rapides.

Cependant elle impose de se limiter à la recherche du groupe de Galois de polynômes à coefficients dans des corps de nombres.

Un autre problème peut être rencontré : connaître les racines avec une précision numérique qui permette de les distinguer ne signifie pas que cette précision permette de déterminer les relations algébriques entre les racines. Voir par exemple, dans le cas des résolvantes, [41, paragraphe 2.1.3.].

Représentation par un élément primitif. Être capable de représenter chacune des racines comme des polynômes en un élément primitif γ du corps de décomposition de f signifie qu'un problème équivalent à la recherche du groupe de Galois de f a été résolu : les K-automorphismes du corps de décomposition de f correspondent aux conjugués du polynôme minimal de γ sur K. Ce polynôme minimal est donc de degré l'ordre du groupe de Galois. Cet ordre peut atteindre $n!$. Dans la pratique, dès que le degré de l'extension algébrique dépasse la dizaine, la factorisation symbolique dans cette extension se révèle impraticable avec les techniques actuelles (par exemple, la méthode de B. M. TRAGER [93] se ramène, pour factoriser un polynôme de degré n sur une extension de degré d de son corps des coefficients K, à la factorisation d'un polynôme de degré nd à coefficients dans le corps K). L'attaque directe par extensions successives par les racines de f semble donc sans espoir pratique sauf dans des cas très particuliers.

En raison de ce problème de représentation et de manipulation des racines, une autre approche consiste à se dispenser de la détermination des racines. Pour cela, en n'utilisant que des calculs sur des polynômes dont les coefficients sont symétriques en les racines de f, il est possible d'exprimer ces coefficients en fonction des seuls coefficients de f. Dans cette approche, la meilleure connaissance du groupe de Galois de f qu'il est possible d'obtenir, en l'absence d'une numérotation des racines, est la classe de conjugaison des représentations symétriques de ce groupe.

En fait, les méthodes connues faisant appel à la factorisation des polynômes, nous chercherons à manipuler des polynômes à coefficients dans l'anneau A plutôt que dans K. Et nous factoriserons les polynômes sur l'anneau A par des techniques efficaces du type Berlekamp-Zassenhaus exposées au chapitre 5.

1.5 Résolvantes

Nous supposons toujours dans ce paragraphe que l'hypothèse 1.16 est vérifiée.

J. L. LAGRANGE [59] a défini des polynômes particuliers dont la généralisation est utilisée dans les méthodes de théorie de Galois effective développées par J.-M. ARNAUDIÈS et A. VALIBOUZE [6].

Définition 1.31. Soient L et H des sous-groupes de $\mathfrak{S}_n$, avec $H \subset L$, et $\Theta \in A[x_1, \ldots, x_n]$ un H-invariant primitif L-relatif.

La *résolvante de Lagrange générique L-relative de* Θ, notée $\mathcal{L}^L_\Theta$, est définie par :

$$\mathcal{L}^L_\Theta(t) = \prod_{\Psi \in L.\Theta} (t - \Psi(x_1, \ldots, x_n)) \in A[x_1, \ldots, x_n]^L[t] \quad .$$

Notation 1.32. Lorsque $L = \mathfrak{S}_n$, la résolvante $\mathcal{L}^{\mathfrak{S}_n}_\Theta$ est appelée *résolvante de Lagrange générique (absolue) de* Θ et notée $\mathcal{L}_\Theta$.

Définition 1.33. Soient L et H des sous-groupes de $\mathfrak{S}_n$, avec $H \subset L$, $\Theta \in A[x_1, \ldots, x_n]$ un H-invariant primitif L-relatif et $f \in A[x]$ un polynôme de degré n et de racines $\Omega_f = (\alpha_1, \ldots, \alpha_n)$ dans une extension algébrique $\hat{K}$ du corps des fractions de A.

La *résolvante de Lagrange L-relative de* Ω_f *par* Θ, notée $\mathcal{L}^L_{\Theta,\Omega_f}$, est définie par :

$$\mathcal{L}^L_{\Theta,\Omega_f}(t) = \prod_{\Psi \in L.\Theta} (t - \Psi(\Omega_f)) \quad .$$

Propriété 1.34. *D'après la théorie de Galois, si le groupe de Galois* Γ *de* f *est un sous-groupe de* L*, alors :*

$$\mathcal{L}^L_{\Theta,\Omega_f} \in A[t] \quad .$$

Remarque : Si le polynôme f n'est pas unitaire, alors $\mathcal{L}^L_{\Theta,\Omega_f} \in K[t]$.

Notation 1.35. Lorsque $L = \mathfrak{S}_n$, la résolvante $\mathcal{L}^{\mathfrak{S}_n}_{\Theta,\Omega_f}$ est appelée *résolvante de Lagrange (absolue) de* Ω_f *par* Θ et notée $\mathcal{L}_{\Theta,\Omega_f}$. Comme dans ce cas elle ne dépend pas de la numérotation Ω_f des racines de f, elle est aussi notée $\mathcal{L}_{\Theta,f}$.

Hypothèse 1.36. Soient L et H des sous-groupes de $\mathfrak{S}_n$, avec $H \subset L$, $\Theta \in A[x_1, \ldots, x_n]$ un H-invariant primitif L-relatif. Soit $f \in A[x]$ un polynôme unitaire de degré n et de racines $\Omega_f = (\alpha_1, \ldots, \alpha_n)$ dans une extension algébrique $\hat{K}$ du corps des fractions de A dont la représentation symétrique Γ du groupe de Galois associé à Ω_f est un sous-groupe de L.

Définition 1.37. Sous l'hypothèse 1.36, l'invariant Θ est dit *Ω_f-séparable* si la résolvante $\mathcal{L}^L_{\Theta,\Omega_f}$ l'est, c'est à dire si cette résolvante n'a que des racines simples.

J.-M. Arnaudiès et A. Valibouze [6] ont établi le rapport entre résolvantes de Lagrange et matrices de partition définies en 1.14 :

Théorème 1.38 (Arnaudiès-Valibouze). *Sous l'hypothèse 1.36, en notant* m_i *le nombre de facteurs irréductibles de degré* i *sur* $K[t]$ *de* $\mathcal{L}^L_{\Theta,\Omega_f}$*,*

$$(m_1, \ldots, m_{[L:H]}) \preceq \varpi([\Gamma]_L, [H]_L)$$

et si, de plus, l'invariant Θ *est* Ω_f*-séparable alors*

$$(m_1, \ldots, m_{[L:H]}) = \varpi([\Gamma]_L, [H]_L) \quad .$$

Preuve : Voir [6, théorème 22]. □

1.6 La « chasse aux résolvantes »

Comme les lignes des matrices de partitions sont toutes distinctes (voir le théorème 1.15), la seule connaissance des degrés des facteurs irréductibles des résolvantes de Lagrange séparables permet de toujours distinguer les classes de conjugaison possibles du groupe de Galois (l'existence d'invariants Ω_f-séparables est garantie par [6, théorème 6] lorsque le corps K est infini).

Dans la pratique, plusieurs pré-requis sont encore nécessaires.

Tout d'abord la classification des classes de conjugaison de $\mathfrak{S}_n$. Elle peut être effectuée grâce au logiciel `GAP` [87] spécialisé dans les calculs sur les groupes. Ces résultats sont rappelés dans [94] jusqu'au degré 7. Au-delà la classification des sous-groupes transitifs de $\mathfrak{S}_n$ (qui correspondent aux cas où f est irréductible) a été réalisée jusqu'au degré 31 par les contributions, notamment de [24], [86], [76], [23] et [34] qui fournit une nomenclature exhaustive.

Par ailleurs, le calcul de résolvantes pouvant être très difficile, la « chasse aux résolvantes » consiste, en tenant compte de ces difficultés de calcul, à extraire des sous-matrices de la matrice de partition, qui permettront, en n'utilisant que des résolvantes raisonnablement calculables associées à certaines classes (elle sont appelées *classes tests*), de toujours distinguer deux classes de conjugaison possibles (elle sont appelées *classes candidates*).

Nous touchons là deux points clefs de la méthode formelle de détermination (de la classe de conjugaison) du groupe de Galois d'un polynôme :

1. être capable de calculer efficacement des résolvantes ;

2. factoriser efficacement ces résolvantes, ou plus exactement distinguer des factorisations de résolvantes parmi la liste des degrés des facteurs des factorisations possibles donnés par la matrice de partition.

1.7 Groupes de Galois des facteurs irréductibles des résolvantes

Théorème 1.39. *Sous l'hypothèse 1.36, si Θ est Ω_f-séparable, alors la classe de conjugaison des représentations symétriques de degré le degré de chaque facteur irréductible* ***simple*** *de $\mathcal{L}_{\Theta,\Omega_f}$ du groupe de Galois de chacun de ces facteurs ne dépend que de L, $[H]_L$ et $[\Gamma]_L$.*

Preuve : Voir [30, corollaire 6.11, page 53]. □

La conséquence de ce théorème sur la « chasse aux résolvantes », est que la matrice des partitions des degrés des facteurs des résolvantes peut être remplacée par la matrice des classes de conjugaison des groupes de Galois des facteurs des résolvantes.

Si les degrés des facteurs irréductibles sont plus élevés que le degré de f, le problème à résoudre est en général plus difficile que le problème original (le groupe de Galois de f). Cependant, parfois, la parité du groupe de Galois du facteur peut faire la distinction entre deux classes candidates alors qu'elle est très simple à tester (si le discriminant est un carré).

La « chasse aux résolvantes » peut aussi bénéficier des remarques suivantes :

1. lorsque la résolvante n'est pas séparable, et est alors plus facile à factoriser, utiliser la première inégalité du théorème 1.38 ;
2. pour distinguer les factorisations possibles dans la table des groupes de Galois, l'information discriminante peut ne pas nécessiter une factorisation complète de la résolvante : l'existence ou non de facteurs d'un certain degré peut suffire, ainsi que la détermination du groupe de Galois de ces facteurs spécifiques ou simplement de leur parité.

Chapitre 2

Les outils du calcul de résolvantes

Le chapitre 1 a montré l'utilisation des résolvantes de Lagrange en théorie de Galois effective. Ce chapitre introduit les techniques et notions qui seront utilisées dans les chapitres 3 et 4 pour calculer efficacement des résolvantes.

Le paragraphe 2.1 définit le polynôme caractéristique associé à un invariant ainsi que les relations qu'il entretient avec les résolvantes de Lagrange.

Le paragraphe 2.2 introduit les calculs sur les polynômes réciproques.

Le paragraphe 2.3 définit le résultant et rappelle certaines de ses propriétés que nous utiliserons.

Le paragraphe 2.4 décrit notre terminologie sur les motifs de partition qui seront utilisés aux chapitres 3 et 6.

Le paragraphe 2.5 passe en revue les différentes méthodes de calcul de la racine d'un polynôme p qui est une puissance exacte du polynôme recherché q. Ce calcul ne nécessite pas de connaître tous les coefficients de p.

Le paragraphe 2.6 détaille les problèmes du calcul du résultant sur un anneau non intègre et fournit le moyen de les surmonter.

2.1 Autour du polynôme caractéristique

Définition 2.1. Soient $\Theta \in A[x_1, \ldots, x_n]$, L un sous-groupe de $\mathfrak{S}_n$ et $f \in A[x]$ un polynôme de degré n et de racines $\Omega_f = (\alpha_1, \ldots, \alpha_n)$ dans une extension algébrique $\hat{K}$ du corps K des fractions de A ; le *polynôme Ω_f-caractéristique L-relatif de* Θ, noté $\mathcal{X}^L_{\Theta,\Omega_f}$, est défini par :

$$\mathcal{X}^L_{\Theta,\Omega_f}(t) = \prod_{\sigma \in L} \left(t - \Theta\left(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)}\right)\right) \quad .$$

Notation 2.2. $\mathcal{X}^{\mathfrak{S}_n}_{\Theta,\Omega_f}$ est aussi appelé *polynôme f-caractéristique de* Θ et simplement noté $\mathcal{X}_{\Theta,f}$ car il ne dépend pas de la numérotation des racines.

Remarque :

$$\deg_t\left(\mathcal{X}^L_{\Theta,\Omega_f}(t)\right) = \mathrm{card}\,(L) \quad .$$

2.1.1 Justification de la terminologie

Soient n un entier positif, un polynôme $\Theta \in A[x_1, \ldots, x_n]$, $f \in A[x]$ un polynôme séparable de degré n et de racines $\Omega_f = (\alpha_1, \ldots, \alpha_n)$ dans une extension algébrique $\hat{K}$ du corps K, supposé parfait, des fractions de A et L un sous-groupe de $\mathfrak{S}_n$ contenant le groupe de Galois de associé à Ω_f.

Définition 2.3. L'idéal

$$I^L_{\Omega_f} = \left\{ R \in K[x_1, \ldots x_n] \mid R(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)}) = 0 \ \forall \sigma \in L \right\}$$

est appelé *idéal des Ω_f-relations invariantes par L.*

Propriété 2.4. *L'idéal $I^L_{\Omega_f}$ est radical de dimension* 0.

Preuve : Voir [100, lemme 3.2]. □

Proposition 2.5. *La variété de l'idéal $I^L_{\Omega_f}$ dans $\hat{K}^n$, notée $V(I^L_{\Omega_f})$ est égale à :*

$$V(I^L_{\Omega_f}) = \left\{ \left(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)}\right) \mid \sigma \in L \right\} \quad .$$

Preuve : Voir [100, proposition 3.17] □

Notation 2.6. Pour un polynôme $\Theta \in K[x_1, \ldots, x_n]$, $\widehat{\Theta}$ désigne l'endomorphisme du K-espace vectoriel $K[x_1, \ldots, x_n]/I^L_{\Omega_f}$ de multiplication par la classe de Θ, notée $\overline{\Theta}$, dans $K[x_1, \ldots, x_n]/I^L_{\Omega_f}$:

$$\begin{array}{rccc} \widehat{\Theta} \ : & K[x_1, \ldots, x_n]/I^L_{\Omega_f} & \longrightarrow & K[x_1, \ldots, x_n]/I^L_{\Omega_f} \\ & P & \longmapsto & \overline{\Theta}P \end{array} \quad .$$

Le théorème de Stickelberger (voir [115]) implique que le polynôme caractéristique de l'endomorphisme $\widehat{\Theta}$ soit égal à

$$\mathcal{X}^L_{\Theta,\Omega_f}(t) = \prod_{\beta \in V(I^L_{\Omega_f})} (t - \Theta(\beta))$$

ce qui, d'après la proposition 2.5, coïncide sur K avec la définition 2.1.

2.1.2 Autre polynôme lié au polynôme caractéristique

Nous définissons ici un polynôme qui sera utilisé dans le chapitre 3 pour calculer récursivement les résolvantes de Lagrange.

Définition 2.7. Soit $f \in A[x]$ un polynôme séparable de degré n et de racines $\alpha_1, \ldots, \alpha_n$ dans une extension algébrique $\hat{K}$ du corps K des fractions de A. Soient k un entier, $0 \leq k \leq n$ et le polynôme Θ de $A[x_1, \ldots, x_k]$ d'arité k.

Le *polynôme f-évalué de* Θ, noté $\mathcal{E}_{\Theta,f}$, est défini par :

$$\mathcal{E}_{\Theta,f}(t) = \prod_{\beta_1 \in \{\alpha_1,\ldots,\alpha_n\}} \cdots \prod_{\beta_i \in \{\alpha_1,\ldots,\alpha_n\} \atop \beta_i \notin \{\beta_1,\ldots,\beta_{i-1}\}} \cdots \prod_{\beta_k \in \{\alpha_1,\ldots,\alpha_n\} \atop \beta_k \notin \{\beta_1,\ldots,\beta_{k-1}\}} (t - \Theta(\beta_1, \ldots, \beta_k)) \quad .$$

Remarque : Si le polynôme f est de degré n et l'invariant d'arité k alors

$$\deg_t\left(\mathcal{E}_{\Theta,f}\left(t\right)\right) = \frac{n!}{(n-k)!} \quad .$$

2.1.3 Polynôme caractéristique et résolvante de Lagrange

Proposition 2.8 (voir [85, égalité (4)]). *Soient A un anneau intègre, $f \in A[x]$ un polynôme séparable à coefficients dans A de degré n et de racines $\Omega_f = (\alpha_1, \ldots, \alpha_n)$ dans une extension algébrique $\hat{K}$ du corps K des fractions de A, L un sous-groupe de $\mathfrak{S}_n$ contenant de groupe de Galois associé à Ω_f et $\Theta \in A[x_1, \ldots, x_n]$ un H-invariant primitif L-relatif (c'est à dire $H = \mathrm{Stab}_L\Theta$) alors :*

$$\mathcal{X}^L_{\Theta,\Omega_f} = \left(\mathcal{L}^L_{\Theta,\Omega_f}\right)^{\mathrm{card}(H)} \quad .$$

Preuve :

$$\mathcal{L}^L_{\Theta,\Omega_f}(t) = \prod_{\Psi \in L.\Theta} \left(t - \Psi(\alpha_1, \ldots, \alpha_n)\right)$$

et $\mathrm{card}\left(L\right) = \mathrm{card}\left(L.\Theta\right)\mathrm{card}\left(H\right)$ d'où le résultat. □

2.1.4 Polynôme f-évalué et polynôme caractéristique

Proposition 2.9. *Soient A un anneau intègre, $f \in A[x]$ un polynôme séparable à coefficients dans A de degré n et de racines $\Omega_f = (\alpha_1, \ldots, \alpha_n)$ dans une extension algébrique $\hat{K}$ du corps K des fractions de A. Soient k un entier, $0 \leq k \leq n$ et Θ un $(H_k \times \mathfrak{S}_{\{k+1,\ldots,n\}})$-invariant primitif d'arité k appartenant à $A[x_1, \ldots, x_k]$ alors :*

$$\left(\mathcal{E}_{\Theta,f}\right)^{(n-k)!} = \mathcal{X}_{\Theta,f} \quad .$$

Preuve : D'après la définition 2.1 du polynôme caractéristique, en considérant Θ comme un élément de $A[x_1, \ldots, x_n]$,

$$\begin{aligned}
\mathcal{X}_{\Theta,f}(t) &= \prod_{\sigma \in \mathfrak{S}_n} \left(t - \Theta\left(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)}\right)\right) \\
&= \prod_{\beta_1 \in \{\alpha_1,\ldots,\alpha_n\}} \cdots \prod_{\beta_n \notin \{\beta_1,\ldots,\beta_{n-1}\}}^{\beta_n \in \{\alpha_1,\ldots,\alpha_n\}} \left(t - \Theta(\beta_1, \ldots, \beta_n)\right) \\
&= \prod_{\beta_1 \in \{\alpha_1,\ldots,\alpha_n\}} \cdots \prod_{\beta_k \notin \{\beta_1,\ldots,\beta_{k-1}\}}^{\beta_k \in \{\alpha_1,\ldots,\alpha_n\}} \cdots \prod_{\beta_n \notin \{\beta_1,\ldots,\beta_{n-1}\}}^{\beta_n \in \{\alpha_1,\ldots,\alpha_n\}} \left(t - \Theta(\beta_1, \ldots, \beta_k)\right) \\
&= \prod_{\beta_1 \in \{\alpha_1,\ldots,\alpha_n\}} \cdots \prod_{\beta_k \notin \{\beta_1,\ldots,\beta_{k-1}\}}^{\beta_k \in \{\alpha_1,\ldots,\alpha_n\}} \left(t - \Theta(\beta_1, \ldots, \beta_k)\right)^{(n-k)!} \\
&= \left(\mathcal{E}_{\Theta,f_n}\left(t\right)\right)^{(n-k)!} \quad .
\end{aligned}$$

□

2.1.5 Polynôme f-évalué et résolvante de Lagrange

Les propositions 2.8 et 2.9 établissent le rapport entre polynôme caractéristique et polynôme f-évalué :

Proposition 2.10. *Soient A un anneau intègre, $f \in A[x]$ un polynôme séparable à coefficients dans A de degré n et de racines $\Omega_f = (\alpha_1, \ldots, \alpha_n)$ dans une extension algébrique $\hat{K}$ du corps K des fractions de A. Soient k un entier, $0 \leq k \leq n$ et $\Theta \in A[x_1, \ldots, x_k]$ un $(H_k \times \mathfrak{S}_{\{k+1,\ldots,n\}})$-invariant primitif (avec $H_k = \mathrm{Stab}_{\mathfrak{S}_k}(\Theta) \subset \mathfrak{S}_k$). Alors :*

$$\mathcal{E}_{\Theta,f} = (\mathcal{L}_{\Theta,f})^{\mathrm{card}(H_k)} \quad .$$

Preuve : D'après la proposition 2.8,

$$\begin{aligned} \mathcal{X}_{\Theta,f} &= (\mathcal{L}_{\Theta,f})^{\mathrm{card}\left(H_k \times \mathfrak{S}_{\{k+1,\ldots,n\}}\right)} \\ &= (\mathcal{L}_{\Theta,f})^{\mathrm{card}(H_k)(n-k)!} \quad . \end{aligned}$$

D'où le résultat en utilisant la proposition 2.9. □

2.2 Polynômes réciproques et troncature

Ce paragraphe rappelle quelques règles de calcul sur les polynômes réciproques. Nous les utiliserons au paragraphe 2.5 pour ne conserver que les termes strictement nécessaires au calcul de racine r-ième de polynômes et dans les chapitres 3 et 4 pour accélérer les calculs de résolvantes dans le cas général.

Soient A un anneau intègre, n un entier positif, f un polynôme de degré n en la variable t à coefficients dans A.

Notation 2.11. Soit $f \in A[t]$, $\deg_t f$ désigne le *degré (partiel) par rapport à t* de f.

Définition 2.12. La transformation du polynôme $f \in A[t]$ en son polynôme réciproque, est notée Récip ou $\widetilde{\ }$:

$$\mathrm{Récip}\,(f)\,(t) = \widetilde{f}(t) = t^{\deg_t(f)} f\left(\frac{1}{t}\right) \quad .$$

Notation 2.13. Soit s un entier positif, le *modulo* du polynôme f *par rapport à* t^{s+1}, est noté $f(t) \bmod t^{s+1}$. Il signifie que ne sont conservés de f que ses termes de degré en t inférieur ou égal à s.

Notation 2.14. Soient n un entier positif et f un polynôme de degré n à coefficients dans l'anneau A :

$$f(t) = a_n t^n + \cdots + a_1 t + a_0$$

avec $a_n \neq 0$. Le coefficient dominant de f est noté $\mathrm{cd}\,(f) = a_n$

Définition 2.15. Soient f et g des polynômes de $A[t]$ tels que le coefficient dominant de g, $\mathrm{cd}\,(g)$, soit inversible dans A.

Les quotient et reste de la *division euclidienne* de f par g sont respectivement les uniques polynômes q et r de $A[t]$ vérifiant :

$$f = qg + r \qquad \text{avec} \qquad \deg_t(r) < \deg_t(g) \quad .$$

Le quotient q de cette division par rapport à la variable t est noté :

$$\left\lfloor \frac{f}{g} \right\rfloor_t = q \quad .$$

Définition 2.16. Soient f et g des polynômes de $A[t]$ tels que $g(0)$ soit inversible dans A et j un entier positif.

Les j-ièmes quotient et reste de la *division suivant les puissances croissantes* de f par g sont respectivement les uniques polynômes q_j et r_j de $A[t]$ vérifiant :

$$f = q_j g + t^{j+1} r_j \qquad \text{avec} \qquad \deg_t(q_j) \leq j \quad . \tag{2.1}$$

Le j-ième quotient q_j de cette division par rapport à la variable t est noté :

$$\left\lceil \frac{f}{g} \right\rceil_j^t = q \quad ,$$

et simplement

$$\left\lceil \frac{f}{g} \right\rceil^t = q \quad ,$$

lorsque $j = \deg_t(f) - \deg_t(g)$.

Propriété 2.17. *Sous les hypothèses de la définition 2.16, le j-ième quotient suivant les puissances croissantes de f par g peut être calculé à partir de leurs seuls $j+1$ premiers coefficients (pour les degrés variant de 0 à j), c'est-à-dire à partir des valeurs de f et g modulo t^{j+1}.*

Preuve : Il suffit de considérer l'équation (2.1) modulo t^{j+1}. □

Propriété 2.18. *Soient f et g des polynômes de $A[t]$ tels que le coefficient dominant de g, $\mathrm{cd}\,(g)$, soit inversible dans A et dont la différence des degrés $j = \deg_t(f) - \deg_t(g)$ est positive.*

Le polynôme réciproque du quotient de la division euclidienne *de f par g est égal au j-ième quotient suivant les puissances croissantes des polynômes réciproques respectifs de f et g.*

Soit avec les notations et définitions 2.12, 2.15 et 2.16 :

$$\mathrm{Récip}\left(\left\lfloor \frac{f}{g} \right\rfloor_t\right) = \left\lceil \frac{\mathrm{Récip}\,(f)}{\mathrm{Récip}\,(g)} \right\rceil^t \quad .$$

Preuve : Soit

$$f = qg + r \tag{2.2}$$

la division euclidienne de f par g, avec $\deg_t(r) < \deg_t(g)$.

En multipliant (2.2) par $t^{\deg_t(f)}$ après y avoir substitué $\frac{1}{t}$ à t, elle devient :

$$t^{\deg_t(f)} f\left(\frac{1}{t}\right) = t^j q\left(\frac{1}{t}\right) t^{\deg_t(g)} g\left(\frac{1}{t}\right) + t^{j+1} t^{\deg_t(g)-1} r\left(\frac{1}{t}\right)$$

qui montre bien le résultat cherché. □

2.3 Le résultant

Ce paragraphe rappelle les propriétés classiques du résultant ainsi que notre lemme déjà exposé dans [61] que nous allons réutiliser dans le chapitre 3.

Notation 2.19. Soient f et g des polynômes de degrés respectifs n et m en la variable y à coefficients dans un anneau intègre A. Ils sont notés :

$$f(y) = a_n y^n + \cdots + a_0 \qquad \text{et} \qquad g(y) = b_m y^m + \cdots + b_0$$

et $\alpha_1, \ldots, \alpha_n$ et $\beta_1, \ldots, \beta_m$ sont leurs racines respectives dans une extension algébrique $\hat{K}$ du corps K des fractions de A.

Définition 2.20. Le *résultant* $\text{Rés}_y(f,g)$ des polynômes f et g par rapport à la variable y est le déterminant de la matrice de Sylvester de f et g composée de m lignes de coefficients de f et de n lignes de coefficients de g:

$$\text{Rés}_y(f,g) = \begin{vmatrix} a_n & a_{n-1} & \cdots & \cdots & \cdots & a_0 & 0 & \cdots & 0 \\ 0 & a_n & a_{n-1} & \cdots & \cdots & \cdots & a_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & & & & & \ddots & 0 \\ 0 & \cdots & 0 & a_n & a_{n-1} & \cdots & \cdots & \cdots & a_0 \\ b_m & b_{m-1} & \cdots & \cdots & b_0 & 0 & \cdots & \cdots & 0 \\ 0 & b_m & b_{m-1} & \cdots & \cdots & b_0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & & & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & & & & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & b_m & b_{m-1} & \cdots & \cdots & b_0 \end{vmatrix}.$$

Une méthode efficace de calcul du résultant est donnée dans [40].

2.3.1 Propriétés classiques du résultant

Avec les notations ci-dessus :

Propriété 2.21.

$$\text{Rés}_y(f,g) = a_n^m \prod_{i=1}^{n} g(\alpha_i) \quad .$$

Et de cette propriété se déduit immédiatement, pour un polynôme g se factorisant en $g = g_1 g_2$:

Propriété 2.22.

$$\text{Rés}_y(f,g) = \text{Rés}_y(f,g_1)\,\text{Rés}_y(f,g_2) \quad .$$

Ainsi que :

Corollaire 2.23. *Pour tout entier k positif,*

$$\text{Rés}_y\left(f,g^k\right) = \text{Rés}_y(f,g)^k \quad .$$

2.3.2 Compatibilité du résultant avec le modulo

Soit t une variable transcendante sur l'anneau intègre A.

Lemme 2.24. *Soient A un anneau intègre, $f \in A[y]$ un polynôme **unitaire**, $r \in A[t][y]$ un polynôme en y à coefficients dans l'anneau $A[t]$ et s un entier positif. Nous avons :*

$$\text{Rés}_y\left(f, r \bmod t^{s+1}\right) = \text{Rés}_y(f,r) \bmod t^{s+1}$$

Preuve : Soit n le degré de f et $\alpha_1, \ldots, \alpha_n$ ses racines dans une extension algébrique $\hat{K}$ du corps K des fractions de A. Le polynôme f étant unitaire :

$$\text{Rés}_y(f,r) = \prod_{i=1}^{n} r(t,\alpha_i) \quad ,$$

égalité qui est aussi vraie modulo t^{s+1}. □

2.3.3 Compatibilité du résultant avec la transformation réciproque par rapport à la variable t

Lemme 2.25. *Soient A un anneau intègre, et $A[t][y]$ l'ensemble des polynômes de variable y à coefficients dans $A[t]$. Soient f et g appartenant à $A[t][y]$:*

$$f(t,y) = a_n(t)y^n + a_{n-1}(t)y^{n-1} + \cdots + a_0(t) \quad ;$$
$$g(t,y) = b_m(t)y^m + b_{m-1}(t)y^{m-1} + \cdots + b_0(t) \quad .$$

La transformation réciproque par rapport à la variable t est compatible avec le calcul du résultant de f et de g par rapport à la variable y :

Si

$$r(t) = \text{Rés}_y\left(f(t,y), g(t,y)\right)$$

alors

$$\text{Rés}_y\left(\text{Récip}\left(f\right)(t,y), \text{Récip}\left(g\right)(t,y)\right) = t^{\deg_t(f)m+\deg_t(g)n} r\left(\frac{1}{t}\right) \quad .$$

Preuve :

$$\text{Récip}\left(f\right)(t,y) = t^{\deg_t(f)} a_n\left(\frac{1}{t}\right) y^n + \cdots + a_0\left(\frac{1}{t}\right) \quad ;$$
$$\text{Récip}\left(g\right)(t,y) = t^{\deg_t(g)} b_m\left(\frac{1}{t}\right) y^m + \cdots + b_0\left(\frac{1}{t}\right) \quad .$$

Le résultant $r(t) = \text{Rés}_y\left(f(t,y), g(t,y)\right)$ de f et g par rapport à y est le déterminant de la matrice de Sylvester de f et g (voir la définition 2.20).

Identiquement, le résultant de Récip (f) et de Récip (g) par rapport à y est le déterminant de la matrice de Sylvester de Récip (f) et Récip (g), d'où en sortant du déterminant

$$\begin{vmatrix}
t^{\deg_t(f)} a_n\left(\frac{1}{t}\right) & \cdots & & \cdots & t^{\deg_t(f)} a_0\left(\frac{1}{t}\right) & & \\
& \ddots & & & & \ddots & \\
& & t^{\deg_t(f)} a_n\left(\frac{1}{t}\right) & \cdots & \cdots & & t^{\deg_t(f)} a_0\left(\frac{1}{t}\right) \\
t^{\deg_t(g)} b_m\left(\frac{1}{t}\right) & \cdots & & \cdots & t^{\deg_t(g)} b_0\left(\frac{1}{t}\right) & & \\
& \ddots & & & & \ddots & \\
& & t^{\deg_t(g)} b_m\left(\frac{1}{t}\right) & \cdots & \cdots & & t^{\deg_t(g)} b_0\left(\frac{1}{t}\right)
\end{vmatrix}$$

les puissances de t y apparaissant, le résultat cherché. □

2.3.4 Échange des opérations quotient et résultant

Nous rappelons ici l'énoncé et la preuve du lemme donné dans [61]. Il généralise une remarque de C. WILLIAMSON dans [113] pour le cas particulier des résolvantes linéaires d'arité 2 (voir la proposition 3.3).

Lemme 2.26. *Soient n un entier positif, A un anneau intègre et $f \in A[x]$ un polynôme* ***unitaire*** *séparable de degré n dont les racines dans une extension algébrique $\hat{K}$ du corps K des fractions de A sont $\alpha_1, \ldots, \alpha_n$. Soient $g(t,y)$, $h(t,y)$ des polynômes* ***unitaires*** *en t.*

Si $h(t,\alpha_i)$ divise $g(t,\alpha_i)$ pour $i \in \{1,\ldots,n\}$ alors

$$g(t,y) = \left\lfloor \frac{g(t,y)}{h(t,y)} \right\rfloor_t h(t,y) + f(y)s(t,y)$$

avec la notation de 2.15, le reste de cette division euclidienne s'écrivant sous la forme $f(y)s(t,y)$.

Le résultant $\text{Rés}_y(f(y),h(t,y))$ *divise alors* $\text{Rés}_y(f(y),g(t,y))$ *et*

$$\frac{\text{Rés}_y(f(y),g(t,y))}{\text{Rés}_y(f(y),h(t,y))} = \text{Rés}_y\left(f(y), \left\lfloor \frac{g(t,y)}{h(t,y)} \right\rfloor_t\right) \quad . \tag{2.3}$$

Preuve :

Soit $g(t,y) = \left\lfloor \frac{g(t,y)}{h(t,y)} \right\rfloor_t h(t,y) + r(t,y)$ la division euclidienne de g par h par rapport à t. Alors, par hypothèse, $(y-\alpha_i)$ divise $r(t,y)$ pour $i = 1, \ldots, n$. Comme les α_i sont distincts deux à deux, $f(y)$ divise $r(t,y)$ qui peut donc s'écrire $r(t,y) = f(y)s(t,y)$.

Les propriétés 2.21 et 2.22 utilisées pour des polynômes à coefficients dans l'anneau $A[t]$ s'appliquent :

$$\begin{aligned}
\text{Rés}_y(f(y),g(t,y)) &= \text{Res}_y\left(f(y), \left\lfloor \frac{g(t,y)}{h(t,y)} \right\rfloor_t h(t,y) + f(y)S(t,y)\right) \\
&= \text{Rés}_y\left(f(y), \left\lfloor \frac{g(t,y)}{h(t,y)} \right\rfloor_t h(t,y)\right) \\
&= \text{Rés}_y\left(f(y), \left\lfloor \frac{g(t,y)}{h(t,y)} \right\rfloor_t\right) \text{Rés}_y(f(y),h(t,y)) \quad .
\end{aligned}$$

□

Il est possible, et utile pour minimiser les calculs nécessaires, de formuler ce lemme sur les polynômes réciproques.

Corollaire 2.27. *Sous les hypothèses du lemme 2.26, le polynôme réciproque du résultant (2.3) peut être calculé, en notant* $j = \deg_t(g) - \deg_t(h)$, *par :*

$$\text{Récip}\left(\frac{\text{Rés}_y(f(y),g(t,y))}{\text{Rés}_y(f(y),h(t,y))}\right) = \text{Rés}_y\left(f(y), \left\lceil \frac{\text{Récip}(g)(t,y) \bmod t^{j+1}}{\text{Récip}(h)(t,y) \bmod t^{j+1}} \right\rceil^t\right) \quad .$$

Preuve : Il suffit d'appliquer les propriétés 2.17 et 2.18. □

2.3.5 Définition du discriminant d'un polynôme

Définition 2.28. Soit f un polynôme de degré n à coefficients dans un anneau intègre A, de racines $\alpha_1, \ldots, \alpha_n$ dans une extension algébrique $\hat{K}$ du corps K des fractions de A.

Le discriminant de f, noté $\Delta(f)$ est défini par :

$$\Delta(f) = \text{cd}(f)^{2n-2} \prod_{1 \le i < j \le m} (\alpha_i - \alpha_j)^2 \quad .$$

Propriété 2.29. *Le discriminant peut être calculé à partir du résultant :*

$$\text{cd}(f)\,\Delta(f) = (-1)^{\frac{n(n-1)}{2}} \text{Rés}(f,f') \quad .$$

2.4 Motifs de partition

Soit $\mathcal{E}$ un ensemble constitué de r éléments distincts indicés de 1 à r.

Dans la description de certains algorithmes (voir chapitres 3 et 6) nous avons besoin de parcourir l'ensemble de toutes les parties de ℓ éléments de $\mathcal{E}$ pour ℓ croissant de 0 à r.

Nous introduisons pour cela ici la notion de *motif de partition.*

Notation 2.30. Le corps à deux éléments $\{0,1\}$ est noté $\mathbb{F}_2$.

Soit r un entier positif, le produit cartésien de r copies de $\mathbb{F}_2$ muni de la structure canonique de $\mathbb{F}_2$-espace vectoriel est noté $M(r) = (\mathbb{F}_2)^r$. Les éléments de $M(r)$ sont notés en représentation positionnelle plutôt que sous forme de r-uplets :

$$m_1 m_2 \cdots m_{r-1} m_r = (m_1, m_2, \ldots, m_{r-1}, m_r) \in (\mathbb{F}_2)^r \quad .$$

Définition 2.31. Soit r un entier positif. Un *motif de partition* de longueur r est un élément m de l'espace vectoriel $M(r)$:

$$m = m_1 m_2 \cdots m_{r-1} m_r \in M(r) \quad .$$

Définition 2.32. Le *niveau* d'un motif de partition m de longueur r est le nombre de coordonnées de m égales à 1 :

$$\text{niveau}\,(m) = \text{card}\,(\{k \mid m_k = 1\}) \quad .$$

La partie de $\mathcal{E}$ associée à un motif de partition m de niveau ℓ et de longueur r est l'ensemble des ℓ éléments de $\mathcal{E}$ dont les indices correspondent aux coordonnées de m égales à 1.

Exemple : Pour l'ensemble à cinq éléments $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$, la partie de cet ensemble associée au motif de partition 11010 de niveau 3 est $\{\alpha_1, \alpha_2, \alpha_4\}$.

Notation 2.33. Le sous-ensemble de $M(r)$ composé des motifs de partition de niveau ℓ et de longueur r est noté $M(\ell, r)$.

Exemple : $M(2,3) = \{110, 101, 011\}$.

Remarques :

1. L'espace vectoriel $M(r)$ est l'union disjointe de tous les ensembles $M(\ell, r)$:

$$M(r) = \bigcup_{\ell=0}^{r} M(\ell, r) \quad .$$

2. Nous avons aussi $\text{card}\,(M(\ell, r)) = \binom{r}{\ell}$ et $\text{card}\,(M(r)) = 2^r$.

Un ordre sur les motifs de partitions sera défini dans le paragraphe 6.1.

Remarque : Il semble que les outils introduits dans ce paragraphe et le paragraphe 6.1 puissent être aussi décrits, sous une autre terminologie, par les algèbres de mélange (voir [84]) qui nous étaient inconnues au moment de la rédaction.

2.5 Calcul de racines r-ièmes de polynômes

Ce paragraphe passe en revue les différentes méthodes permettant de calculer exactement la racine r-ième d'un polynôme **unitaire** à coefficients dans un anneau A. Il met l'accent sur les méthodes de moindre complexité algorithmique, en nombre d'opérations et en espace mémoire. Il précise aussi les conditions d'utilisation de chacune de ces méthodes, notamment vis-à-vis de la caractéristique de l'anneau A.

Le paragraphe 2.5.2 expose les méthodes dérivées des algorithmes de factorisation sans facteur carré, avec notre adaptation de l'algorithme de Yun au calcul de racine r-ième.

Le paragraphe 2.5.3 rappelle l'algorithme de décomposition polynomiale proposé par D. KOZEN et S. LANDAU dans leur article [56] (voir aussi le paragraphe 5.3.3). La première étape de cet algorithme est un calcul de racine r-ième. Nous montrons comment le fait de travailler sur des polynômes réciproques améliore sa complexité tout en conservant la généralité de son utilisation.

Le paragraphe 2.5.5 rappelle des méthodes de calcul sur les séries formelles asymptotiquement les plus efficaces.

Le paragraphe 2.5.6 termine ce tour d'horizon sur une méthode simple de moins bonne complexité asymptotique mais dont l'efficacité est très grande dans la gamme de nos besoins.

2.5.1 Cadre du calcul de la racine r-ième

Soient n, r et s des entiers strictement positifs tels que

$$n = rs$$

et p un polynôme **unitaire** à coefficients dans un anneau intègre A ($a_i \in A$ pour $0 \leq i \leq n$; $a_n = 1$) :

$$p = a_n t^n + a_{n-1} t^{n-1} + \cdots + a_1 t + a_0 \tag{2.4}$$

qui est la puissance r-ième d'un polynôme q unitaire de degré s à coefficients dans A :

$$p = q^r \quad , \tag{2.5}$$

le polynôme q n'étant pas nécessairement irréductible.

Problème : Nous souhaitons calculer les coefficients du polynôme q à partir de ceux de p de la façon la plus efficace possible et avec les plus faibles hypothèses sur l'anneau A, qui pourra être typiquement $\mathbb{Z}$, $\mathbb{F}_p$, $\mathbb{Z}[x_1, \ldots, x_k]$ ou $\mathbb{F}_p[x_1, \ldots, x_k]$ pour un entier positif k fixé.

Remarque : Il est toujours possible de se ramener à un polynôme unitaire par la transformation :

$$a_n^{n-1} p\left(\frac{t}{a_n}\right) \quad .$$

2.5.2 Factorisation sans facteur carré

Hypothèse 2.34. La factorisation sans facteur carré est définie sur un anneau factoriel. Les algorithmes exposés dans ce paragraphe utilisent le pgcd. Nous supposons disposer d'un algorithme pour le calculer.

Définition 2.35. Soient n un entier positif et f un polynôme de degré n. La *factorisation sans facteur carré* du polynôme f est la donnée des facteurs f_i tels que

$$f = f_1^1 \cdots f_n^n$$

avec $\text{pgcd}\,(f_i, f_j) = 1$ et $\text{pgcd}\,(f_i, f_i') = 1$, pour $1 \leq i < j \leq n$.

Le polynôme f est dit *sans facteur carré* si la factorisation ci-dessus est *triviale* : $f_1 = f$ et $f_i = 1$ pour $i \geq 2$.

Exemple :

f	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
$t^3(t-1)^2(t-2)^3$	1	$t-1$	$t(t-2)$	1	1	1	1	1

Plusieurs algorithmes exposés par D. Y. Y. YUN dans [116] permettent de déterminer cette factorisation sans facteur carré dans le cas général.

Application au calcul de racine r-ième

La factorisation sans facteur carré du polynôme q racine cherchée de p s'écrit donc :

$$q = q_1 q_2^2 \cdots q_s^s \quad .$$

où les q_i, pour $1 \leq i \leq s$, sont premiers entre eux et sans facteur carré.

Le polynôme q_i est le produit des facteurs de q de multiplicité i.

La conséquence de la relation (2.5) est que la factorisation sans facteur carré de p est très particulière :

$$\begin{aligned} p &= q_1^r q_2^{2r} \cdots q_s^{sr} \\ &= p_1 p_2^2 \cdots p_s^s \end{aligned}$$

avec $p_i^r = q_i$ pour $1 \leq i \leq s$ et $p_j = 1$ pour $j \not\equiv 0 \bmod r$.

La factorisation sans facteur carré du polynôme q peut donc être obtenue à partir de celle de p :

$$q = p_r p_{2r}^2 \cdots p_{sr}^s \quad .$$

Algorithme de Soicher

L. SOICHER propose dans sa thèse [89] la spécialisation d'un algorithme de factorisation sans facteur carré au cas des racines r-ièmes.

La caractéristique de A est supposée nulle ou strictement supérieure au degré de p.

La dérivée formelle du polynôme p est notée p' et la fonction $\mathrm{pgcd}\,(p,q)$ calcule le plus grand dénominateur commun de p et q.

Algorithme POLYROOT de L. Soicher

Entrée : $r \geq 1$, $p \in A[t]$ tel que $p = q^r$; $\{p = q_1^r q_2^{2r} \cdots q_s^{sr}\}$
Sortie : q. $\{q = q_1 q_2^2 \cdots q_s^s\}$

(1) Si $r = 1$ alors retourne p
(2) $g \leftarrow p/\mathrm{pgcd}(p, p')$ $\{g = q_1 q_2 \cdots q_s\}$
(3) $q \leftarrow g$ $\{q = q_1 q_2 \cdots q_s\}$
(4) $s \leftarrow \mathrm{degré}\,(p)/r$ $\{\mathrm{degré}\,(p) = sr\}$
(5) Tant que $\mathrm{degré}\,(q) < s$ répète : $\{$A la i-ième itération : $\}$
  $p \leftarrow p/g^r$ $\{p = q_{i+1}^r q_{i+2}^{2r} \cdots q_s^{(s-i)r}\}$
  $g \leftarrow \mathrm{pgcd}(g, p)$ $\{g = q_{i+1} q_{i+2} \cdots q_s\}$
  $q \leftarrow qg$ $\{q = q_1^1 q_2^2 \cdots q_{i+1}^{i+1} q_{i+2}^{i+1} \cdots q_s^{i+1}\}$
(6) Retourne q

La validité de cet algorithme découle directement des commentaires en fin de ligne.

Inconvénients : L'opération la plus coûteuse de cet algorithme est le calcul du pgcd de p et sa dérivée première p'. Sa complexité précise dépend de la méthode choisie pour calculer le pgcd.

Son comportement dépendra très fortement de la nature de la décomposition sans facteur carré de q (de peu de facteurs de grande multiplicité jusqu'à un polynôme irréductible).

Dans le cadre de polynômes à coefficients dans l'anneau $\mathbb{Z}[x_1, \ldots, x_k]$, l'utilisation de la décomposition sans facteur carré est particulièrement coûteuse en raison de la difficulté d'effectuer des pgcd sur de tels anneaux.

Avantage : Cependant cette décomposition sans facteur carré peut nous donner, certes au prix de calculs coûteux, plus d'informations que la racine r-ième de p : elle aboutit parfois à une factorisation partielle de q.

Spécialisation d'un autre algorithme

D. Y. Y. YUN a proposé un algorithme de décomposition sans facteur carré, noté (c) dans [116], dont il montre les bonnes propriétés de complexité. Cet algorithme peut facilement être adapté au cas particulier du calcul de racine r-ième :

Algorithme de Yun adapté à la racine r-ième

Entrée : $r \geq 1$, $p \in A[t]$ tel que $p = q^r$; $\{p = q_1^r q_2^{2r} \cdots q_s^{sr}\}$
Sortie : q. $\{q = q_1 q_2^2 \cdots q_s^s\}$

(1) Si $r = 1$ alors retourne p
(2) $g \leftarrow \text{pgcd}(p, p')$
(3) $c \leftarrow p/g$
(4) $d \leftarrow p'/g - rc'$
(5) $q \leftarrow 1$
(6) Tant que $c \neq 1$ répète :
$q_i \leftarrow \text{pgcd}(c, d)$
$c \leftarrow c/q_i$
$d \leftarrow d/q_i - rc'$
$q \leftarrow qq_i$
(7) Retourne q

La preuve de l'algorithme découle directement de celle de [116]. L'algorithme de Yun calcule un nouveau facteur p_i de la factorisation sans facteur carré à chaque itération. Notre adaptation à la racine r-ième consiste simplement à regrouper ces itérations r par r et à ne pas effectuer les $r - 1$ pgcd dont nous savons par avance qu'ils seront égaux à 1.

Avantages et inconvénients : Il souffre des mêmes défauts de comportement que l'algorithme POLYROOT à l'égard des anneaux sur lesquels le pgcd est une opération difficile, puisqu'il impose également le calcul de pgcd (p, p'). Sa complexité est néanmoins meilleure (voir [116]).

2.5.3 Algorithmes issus de la décomposition polynomiale fonctionnelle

La décomposition polynomiale fonctionnelle qui est présentée au paragraphe 5.3.3 a pour but, connaissant le polynôme p, de trouver des polynômes f et q dans $A[t]$ tels que p soit égal à la composition du polynôme q par f :

$$p = f \circ q \quad .$$

Dans notre cas particulier de calcul de racine r-ième, nous cherchons à déterminer q, sachant que :

$$f(t) = t^r \quad .$$

L'algorithme de Kozen et Landau

D. KOZEN et S. LANDAU proposent dans [56] un algorithme permettant de calculer q en $O(n^2 r)$ opérations.

Nous rappelons leurs résultats en commençant par deux lemmes techniques :

Lemme 2.36 (Kozen et Landau). *Soient A un anneau commutatif et u, v, w des polynômes de $A[t]$. Si u et v coïncident sur leurs k premiers coefficients (par ordre de degrés décroissants) alors il en est de même de uw et vw.*

Lemme 2.37 (Kozen et Landau). *Soit q_k le polynôme dont les $k+1$ premiers coefficients sont ceux de q défini par :*

$$q_k = t^s + c_{s-1}t^{s-1} + \cdots + c_{s-k}t^{s-k} \in A[t], \quad 0 \leq k \leq s$$

avec $q_s = q$.

Alors les puissances q^r et q_k^r coïncident sur leurs $k+1$ premiers coefficients, pour $0 \leq k \leq s$.

Proposition 2.38 (Kozen et Landau). *Si les k premiers coefficients de q, correspondant à $1, c_{s-1}, \ldots, c_{s-(k-1)}$, sont connus, alors c_{s-k} peut être calculé par*

$$c_{s-k} = \frac{a_{rs-k} - d_k}{r}, \quad 1 \leq k \leq s \quad ,$$

où a_{rs-k} est le $(k+1)$-ième coefficient de p, d_k est le coefficient de t^{rs-k} dans q_{k-1}^r et à condition qu'il soit possible de diviser par r dans A.

Preuve : Le $(k+1)$-ième coefficient de q_k^r est le coefficient de t^{rs-k}, mais comme

$$\begin{aligned} q_k^r &= (q_{k-1} + c_{s-k}t^{s-k})^r \\ &= q_{k-1}^r + rc_{s-k}t^{s-k}q_{k-1}^{r-1} + \sum_{i=2}^{r} \binom{r}{i} c_{s-k}^i t^{(s-k)i} q_{k-1}^{r-i} \end{aligned}$$

où la somme $\sum_{i=2}^{r} \binom{r}{i} c_{s-k}^i t^{(s-k)i} q_{k-1}^{r-i}$ est de degré inférieur ou égal à $rs-2k$, ce coefficient est égal à $d_k + rc_{s-k}$ où d_k est le coefficient de t^{rs-k} dans q_{k-1}^r et rc_{s-k} est le coefficient de t^{rs-k} dans $rc_{s-k}t^{s-k}q_{k-1}^{r-1}$. D'après les lemmes 2.36 et 2.37, le $(k+1)$-ième coefficient de q_k^r est égal au coefficient a_{rs-k} de $p = q^r$ pour $1 \leq k \leq s$. □

La proposition 2.38 fournit un algorithme (en supposant qu'il est possible de diviser par r dans l'anneau A) :

Algorithme décomposition polynomiale [56]

Entrée : $r \geq 1$, $p \in A[t]$ tel que $p = q^r$; $\{p = t^{rs} + a_{rs-1}t^{rs-1} + \cdots + a_1 t + a_0\}$
Sortie : q. $\{q = t^s + c_{s-1}t^{s-1} + \cdots + c_1 t + c_0\}$

(1) Si $r = 1$ alors retourne p
(2) $c_s \leftarrow 1$; $q_{-1,0} \leftarrow 1$; $q_{-1,i} \leftarrow 0$, pour $1 \leq i \leq r$ {Initialisations}
(3) Pour k passant de 1 à s répète :
$q_{k-1,j} \leftarrow \sum_{i=0}^{j} \binom{j}{i} c_{s-k+1}^{i} t^{(s-k+1)i} q_{k-2,j-i}$, pour $0 \leq j \leq r$
{Calcul des q_{k-1}^{j} par la formule du binôme de Newton}
$d_k \leftarrow$ coefficient $(q_{k-1,r}, rs-k)$ {Coefficient de t^{rs-k} dans q_{k-1}^{r}}
$c_{s-k} \leftarrow (a_{rs-k} - d_k)/r$
(4) Retourne $q_{s-1,1} + c_0$

Complexité : La complexité de cet algorithme est de $O(n^2 r)$ opérations arithmétiques et nécessite le stockage de $O(nr)$ éléments de l'anneau A, l'opération la plus coûteuse étant le calcul des q_{k-1}^{r} pour $0 \leq k \leq s$.

Cet algorithme est totalement insensible aux multiplicités des facteurs de q et il n'utilise que les $s+1$ premiers coefficients de p (a_{rs-1} à a_{rs-s} dans la boucle). Nous tirons parti de cette propriété au paragraphe 2.5.4 pour éviter des calculs inutiles dans l'algorithme de Kozen et Landau.

2.5.4 Suppression des termes superflus

Nous avons observé dans l'algorithme de Kozen et Landau exposé au paragraphe 2.5.3 que nous n'avions besoin que des $s+1$ premiers termes de p pour en calculer la racine r-ième.

En utilisant la transformation des polynômes en leur polynôme réciproque, définie au paragraphe 2.2, nous remarquons que

$$\widetilde{p} = \widetilde{q}^{\,r} \tag{2.6}$$

car :

$$\widetilde{p} = \widetilde{q^r} = t^{rs} \left(q\left(\frac{1}{t}\right)\right)^r = \left(t^s q\left(\frac{1}{t}\right)\right)^r = \widetilde{q}^{\,r} \quad .$$

Cherchons maintenant à calculer la racine r-ième de $\widetilde{p}$. Il est immédiat de transposer le raisonnement du paragraphe 2.5.3 aux polynômes réciproques et de travailler donc sur les $s+1$ premiers termes de $\widetilde{p}$ et $\widetilde{q}$ **de plus bas degré** (du degré 0 au degré s). Ceci revient à dire que les calculs peuvent être effectués modulo t^{s+1}, en particulier, les calculs si coûteux des q_k^j.

Nous proposons donc une nouvelle version de l'algorithme du paragraphe 2.5.3 qui évite le calcul de termes inutiles en manipulant les polynômes réciproques tronqués modulo t^s :

Algorithme décomposition polynomiale modulo t^{s+1}

Entrée : $r \geq 1$, $p \in A[t]$ tel que $p = q^r$;
$\{\widetilde{p} = 1 + \widetilde{a}_1 t + \cdots + \widetilde{a}_{rs-1} t^{rs-1} + \widetilde{a}_{rs} t^{rs}\,;\ \widetilde{a}_k = a_{rs-k}, 0 \leq k \leq rs\}$
Sortie : q. $\{\widetilde{q} = 1 + \widetilde{c}_1 t + \cdots + \widetilde{c}_{s-1} t^{s-1} + \widetilde{c}_s t^s\,;\ \widetilde{c}_k = c_{s-k}, 0 \leq k \leq s\}$

(1) Si $r = 1$ alors retourne p
(2) $\widetilde{p} \leftarrow$ Récip$(p) \bmod t^{s+1}$ $\{\text{Récip}(p) = t^n p\left(\frac{1}{t}\right)\}$
(3) $\widetilde{c}_0 \leftarrow 1\,;\ \widetilde{q}_{-1,0} \leftarrow 1\,;\ \widetilde{q}_{-1,i} \leftarrow 0, \quad 1 \leq i \leq r$ {Initialisations}
(4) Pour k passant de 1 à s répète :
$\widetilde{q}_{k-1,j} \leftarrow \sum_{i=0}^{j} \binom{j}{i} \widetilde{c}_{k-1}^{\,i} t^{(k-1)i}\ \widetilde{q}_{k-2,j-i} \bmod t^{s+1}, \quad 0 \leq j \leq r$
{Calcul des $\widetilde{q}_{k-1}^{\,j} \bmod t^{s+1}$ par la formule du binôme de Newton}
$\widetilde{d}_k \leftarrow$ coefficient $(\widetilde{q}_{k-1,r}, k)$ {Coefficient de t^k dans $\widetilde{q}_{k-1}^{\,r} \bmod t^{s+1}$}
$\widetilde{c}_k \leftarrow (\widetilde{a}_k - \widetilde{d}_k)/r$
(5) Retourne Récip$(\widetilde{q}_{s-1,1} + \widetilde{c}_s t^s)$

Complexité : Cet algorithme ne nécessite plus que $O(n^2)$ opérations arithmétiques et le stockage de $O(n)$ éléments de l'anneau A.

Nous résumons le résultat de ce paragraphe dans un lemme.

Lemme 2.39. *Soit A un anneau intègre de caractéristique ℓ. Soit $\widetilde{p} \in A[t]$ un polynôme de degré n ayant 1 pour terme constant :*

$$\widetilde{p}(t) = 1 + \widetilde{a}_1 t + \cdots + \widetilde{a}_{n-1} t^{n-1} + \widetilde{a}_n t^n$$

tel que

$$\widetilde{p}(t) = \widetilde{q}(t)^r \qquad et \qquad n = rs$$

alors, si ℓ ne divise pas r, $\widetilde{q}(t)$, de degré s, peut être calculé à partir de :

$$1 + \widetilde{a}_1 t + \cdots + \widetilde{a}_s t^s = \widetilde{p}(t) \bmod t^{s+1} \quad .$$

La transformation des polynômes en leur polynômes réciproques cache une autre démarche : nous avons en fait considéré les polynômes comme des séries formelles (avec un nombre fini de coefficients non nuls). Ceci nous oriente vers une nouvelle famille d'algorithmes.

2.5.5 Méthodes d'inspiration analytique

Notation 2.40. L'anneau des séries formelles en la variable t à coefficients dans l'anneau A est noté $A[[t]]$.

Il s'agit d'évaluer la fonction racine r-ième définie sur l'anneau des séries formelles à coefficients dans A :

$$\begin{array}{rccc} \rho : & A[[t]] & \longrightarrow & A[[t]] \\ & u & \longmapsto & u^{\frac{1}{r}} \end{array}$$

Nous allons rappeler dans ce paragraphe comment la fonction $\rho(\widetilde{p})$ modulo t^{s+1} peut être calculée en $O(\mathcal{M}(s))$ opérations où $\mathcal{M}(s)$ est le nombre d'opérations arithmétiques nécessaires pour calculer modulo t^{s+1} le produit de deux séries formelles modulo t^{s+1} et $\widetilde{p}$ désigne toujours le polynôme de degré n à coefficients dans l'anneau A et de terme constant égal à 1 dont on cherche la racine r-ième $\widetilde{q}$.

Typiquement :

$\mathcal{M}(s)$	Conditions
$O(s^2)$	Implantation classique.
$O(s\ln(s))$	Si A admet une transformée de Fourier rapide ;
$O(s\ln(s)\ln(\ln(s)))$	sinon (voir par exemple [25]).

Nous supposons, pour les calculs de complexité asymptotique, que $\mathcal{M}(s)$ vérifie :

$$\mathcal{M}(\alpha(s)) \sim \alpha\mathcal{M}(s) \quad \text{pour} \quad \alpha > 0 \quad \text{quand} \quad s \to \infty$$

et que

$$s = o(\mathcal{M}(s)) \quad \text{quand} \quad s \to \infty \quad ,$$

c'est-à-dire que les additions modulo t^{s+1} sont négligeables devant le coût de la multiplication.

Remarque : Ces hypothèses sont vérifiées pour les formules $\mathcal{M}(s) = O(s\ln(s))$ ou $\mathcal{M}(s) = O(s\ln(s)\ln(\ln(s)))$.

La méthode qui donne les meilleures complexités asymptotiques (r et s grands) consiste à évaluer la fonction ρ par :

$$\rho(u) = \exp(\frac{1}{r}\ln(u)) \tag{2.7}$$

pour une série formelle dont le terme constant est égal à 1.

Nous utilisons pour calculer efficacement ces fonctions élémentaires (ln, exp, etc.) sur les séries formelles de coefficient constant égal à 1 une adaptation de la méthode de Newton.

Notre présentation de la méthode de Newton pour les séries formelles est inspirée par R. P. Brent, [17] et [18], ainsi que par P. Henrici [47].

Convergence quadratique de la méthode de Newton

Nous montrons la convergence quadratique (en degrés) de la méthode de Newton pour les séries formelles.

Soit f la fonction dont nous cherchons le zéro $\widetilde{q}$ dans $A[[t]]$ dont le terme constant est fixé égal à $\widetilde{c}_0$.

La fonction d'itération de la méthode de Newton est donnée par :

$$\varphi(u) = u - \frac{f(u)}{f'(u)} \quad . \tag{2.8}$$

En développant **formellement** φ en $\widetilde{q}$, nous obtenons :

$$\varphi(u) = \varphi(\widetilde{q}) + \varphi'(\widetilde{q})(u - \widetilde{q}) + \Phi(u, \widetilde{q})(u - \widetilde{q})^2 \quad , \tag{2.9}$$

où $\Phi(u, \widetilde{q})$ est une fonction de u et de $\widetilde{q}$. L'existence de ce développement est à prouver pour chacune des fonctions f pour laquelle nous souhaitons appliquer la méthode de Newton.

Par hypothèse $\varphi(\widetilde{q}) = \widetilde{q}$ et par construction de φ, $\varphi'(\widetilde{q}) = 0$.

D'où :

$$\varphi(u) = \widetilde{q} + \Phi(u, \widetilde{q})(u - \widetilde{q})^2 \quad . \tag{2.10}$$

Donc si

$$\widetilde{q}_k = \widetilde{q} \bmod t^k$$

alors l'équation (2.10) donne :

$$\varphi(\widetilde{q}_k) = \widetilde{q} \bmod t^{2k} \quad .$$

Ceci montre la convergence quadratique de la méthode de Newton, sous l'hypothèse de pouvoir écrire le développement (2.9) : le nombre de monômes calculés double à chaque itération. Nous traduisons cette méthode en un algorithme :

Algorithme méthode de Newton

Entrée : $f : A[[t]] \longmapsto A[[t]]$; s un entier positif
$\widetilde{c}_0$ valeur approchée du zéro $\widetilde{q}$ (son premier terme) ;
Sortie : $\widetilde{q}$ zéro de f modulo t^{s+1}.

(1) $\widetilde{q} \leftarrow \widetilde{c}_0$; $k \leftarrow 1$ $\{\widetilde{q} = \widetilde{c}_0 \bmod t\}$
(2) Tant que $k < s + 1$ répète :
$\widetilde{q} \leftarrow \varphi(\widetilde{q}) \bmod t^{min(s+1,2k)}$ $\{\varphi$ est la fonction d'itération définie par l'équation (2.8)$\}$
$k \leftarrow 2k$ {Le nombre de termes calculés a doublé}
(3) Retourne $\widetilde{q}$

Calcul des fonctions élémentaires

Application au calcul de l'inverse : Chercher l'inverse de la série formelle $\widetilde{p}^{-1}$ dont le coefficient $\widetilde{a}_0$ est égal à 1 revient à trouver le zéro $\widetilde{q}$, dont le terme constant est $\widetilde{c}_0 = 1$, de la fonction :

$$f(u) = \frac{1}{u} - \widetilde{p} \quad .$$

La fonction d'itération est alors :

$$\varphi(u) = u(2 - \widetilde{p}u)$$

pour laquelle existe bien un développement de la forme (2.9).

Complexité : En calculant $\varphi \bmod t^{s+1}$ selon la méthode de [106, algorithme 9.3 et théorème 9.4], le coût global du calcul de l'inverse d'une série formelle $I(s)$ est donné quand $s \to \infty$ par :

$$I(s) \sim 3\mathcal{M}(s) \quad .$$

Application au calcul du quotient : Calculer le quotient $\widetilde{q}/\widetilde{p}$ des séries formelles $\widetilde{p}$ et $\widetilde{q}$ dont les termes constants sont égaux à 1 revient à multiplier $\widetilde{q}$ par l'inverse de $\widetilde{p}$.

Complexité : La division de séries formelles modulo t^{s+1} a alors pour coût global $D(s) = \mathcal{M}(s) + I(s)$ quand $s \to \infty$:

$$D(s) \sim 4\mathcal{M}(s) \quad .$$

Application au calcul du logarithme : Calculer le logarithme $\widetilde{q} = \ln(\widetilde{p})$ de la série formelle $\widetilde{p}$ de terme constant égal $\widetilde{a}_0$ égal à 1 revient à effectuer un calcul de primitive.

La dérivée dans l'anneau $A[[t]]$ de l'expression $\widetilde{q} = \ln(\widetilde{p})$ est égale à :

$$\widetilde{q'} = \frac{\widetilde{p'}}{\widetilde{p}} \quad .$$

Le calcul du logarithme de $\widetilde{p}$ peut donc être effectué en dérivant modulo t^{s+1} la série formelle $\widetilde{p}$ puis en calculant la division modulo t^{s+1} de $\widetilde{p'}$ par $\widetilde{p}$ qui est égale à $\widetilde{q'}$. La valeur de $\widetilde{q}$ est celle de la primitive de la série formelle $\widetilde{q'}$ ayant son terme constant nul ($\widetilde{c}_0 = 0$).

Complexité : Le coût de la dérivation de $\widetilde{p}$ est de s multiplications. Le coût de calcul de la primitive de $\widetilde{q'}$ dont le terme constant est nul est de s divisions dans l'anneau A. Ces deux calculs ont, par hypothèse $s = o(\mathcal{M}(s))$, un coût négligeable devant celui de la division de $\widetilde{p'}$ par $\widetilde{p}$ modulo t^{s+1}. Le coût total $L(s)$ du calcul de $\ln(\widetilde{p})$ sera donc, pour $s \to \infty$:

$$L(s) \sim 4\mathcal{M}(s) \quad .$$

Remarque : Le calcul de la primitive modulo t^{s+1} nécessite *a priori* de pouvoir diviser dans l'anneau A par les entiers compris entre 1 et s.

Application au calcul de l'exponentielle : Calculer l'exponentielle $\widetilde{p} = \exp(\widetilde{q})$ de la série formelle $\widetilde{q}$ dont le terme constant est nul revient à trouver le zéro $\widetilde{p}$ de terme constant $\widetilde{a}_0$ égal à 1 de la fonction :

$$f(u) = \ln(u) - \widetilde{q} \quad .$$

La fonction d'itération est alors :

$$\varphi(u) = u(1 - \ln(u) + \widetilde{q})$$

pour laquelle existe bien un développement de la forme (2.9).

Complexité : En remarquant, sur une suggestion de Paul ZIMMERMANN, qu'à la dernière étape

$$\ln(u) - \widetilde{q} = 0 \bmod t^{\frac{s+1}{2}} \quad ,$$

l'évaluation de $\varphi \bmod t^{s+1}$ sous la forme

$$\varphi(u) = u + u(\widetilde{q} - \ln(u))$$

ne nécessite que le calcul de $\ln(u)$ et une multiplication modulo $t^{\frac{s+1}{2}}$ de u avec $\widetilde{q} - \ln(u)$ d'où un coût de $\mathcal{M}(s/2) + L(s) \sim \frac{9}{2}\mathcal{M}(s)$ opérations arithmétiques. L'étape précédente a demandé $\frac{9}{2}\mathcal{M}(s/2)$ opérations arithmétiques. Et ainsi de suite. D'où le coût global du calcul de l'exponentielle $E(s)$ quand $s \to \infty$:

$$E(s) \sim \frac{9}{2}\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots\right)\mathcal{M}(s) \sim 9\mathcal{M}(s)$$

Application au calcul de la racine r-ième : Le calcul de $\widetilde{q} = \rho(\widetilde{p}) = \widetilde{p}^{\frac{1}{r}}$ s'effectue en utilisant la formule (2.7), chacune des fonctions y apparaissant étant calculée efficacement en utilisant la méthode de Newton.

Complexité : La division (exacte) de $\ln(\widetilde{p})$ par r ne coûte que s divisions dans l'anneau A. Le coût global du calcul de racine r-ième de $\widetilde{p}$ par la formule (2.7) est donc $R(s) = E(s)+L(s)$, soit, quand $s \to \infty$:

$$R(s) \sim 13\mathcal{M}(s) \quad .$$

Remarques :

1. La formule (2.7) nécessite de pouvoir diviser par r dans l'anneau A.

2. Sur une suggestion de Paul ZIMMERMANN, dans ce dernier cas particulier de la racine r-ième, il est plus intéressant pour les plus petites valeurs de r, $r = 2, 3, 4, 5$, les plus fréquentes en pratique, de remplacer l'utilisation de la formule (2.7) par la détermination directe par la méthode de Newton de la racine de terme constant égal à 1 de $f(u) = u^r - \widetilde{p}$. La fonction d'itération est alors $\varphi(u) = (r-1)u - \widetilde{p}/u^{r-1}$ dont la complexité asymptotique est de $(8 + 2\sigma(r-1))\mathcal{M}(s)$ où $\sigma(k)$ est le nombre minimum de produits nécessaires pour calculer x^k. Ce qui conduit aux complexités respectivement pour $r = 2, 3, 4$ et 5 de $8\mathcal{M}(s)$, $10\mathcal{M}(2)$, $12\mathcal{M}(s)$ et $12\mathcal{M}(s)$ quand $s \to \infty$.

Commentaire : Les algorithmes exposés ci-dessus, dont les complexités sont prometteuses nécessitent des outils lourds (transformée de Fourier rapide, itération de la méthode de Newton, etc.) pour une complexité asymptotique en $\mathcal{M}(s)$ mais qui risque de ne manifester sa supériorité sur les méthodes plus classiques (en $O(s^2)$) que pour des degrés éloignés de ceux considérés en calcul formel. Par exemple, en calcul numérique, la transformée rapide de Fourier pour la multiplication de polynômes ne l'emporte sur les méthodes classiques que pour $s \geq 64$ (d'après [47, page 72]).

2.5.6 La méthode de Miller

Au prix d'hypothèses de même nature que pour les méthodes asymptotiquement rapides du paragraphe 2.5.5 (caractéristique de A supérieure à s, possibilité de diviser par r dans A), P. HENRICI propose dans [46] une méthode récursive de complexité $O(s^2)$ qu'il attribue à J. C. P. MILLER.

Lemme 2.41 (J. C. P. Miller). *Soient $\widetilde{p}$ et $\widetilde{q}$ des séries formelles dont le terme constant est égal à 1, telles que*

$$\widetilde{q} = \widetilde{p}^{\frac{1}{r}} \quad . \tag{2.11}$$

Le k-ième coefficient $\widetilde{c}_k$ de $\widetilde{q}$ peut être calculé récursivement à partir des k premiers coefficients de $\widetilde{q}$ et des $k+1$ premiers coefficients $\widetilde{a}_0, \ldots, \widetilde{a}_k$ de $\widetilde{p}$ par la formule :

$$\widetilde{c}_k = \frac{1}{k} \sum_{i=0}^{k-1} \frac{(k - (r+1)i)\widetilde{c}_i \widetilde{a}_{k-i}}{r} \quad . \tag{2.12}$$

Preuve : En dérivant l'équation (2.11), nous obtenons :

$$\widetilde{q}' = \frac{1}{r} \widetilde{p}^{\frac{1}{r}-1} \widetilde{p}' \tag{2.13}$$

Puis en multipliant (2.13) par $\widetilde{p}$, il vient d'après (2.11) :

$$\widetilde{q}'\widetilde{p} = \frac{1}{r}\widetilde{q}\widetilde{p}'$$

Soit :

$$\begin{aligned}(\widetilde{c}_1 + 2\widetilde{c}_2 t + 3\widetilde{c}_3 t^2 + \cdots)(1 + \widetilde{a}_1 t + \widetilde{a}_2 t^2 + \cdots) = \\ \frac{1}{r}(1 + \widetilde{c}_1 t + \widetilde{c}_2 t^2 + \cdots)(\widetilde{a}_1 + 2\widetilde{a}_2 t + 3\widetilde{a}_3 t^2 + \cdots) \quad .\end{aligned} \tag{2.14}$$

En identifiant les coefficients de t^{k-1} de chaque coté de l'égalité (2.14), nous obtenons :

$$\sum_{i=1}^{k} i\widetilde{c}_i\widetilde{a}_{k-i} = \frac{1}{r}\sum_{i=0}^{k-1}(k-i)\widetilde{a}_{k-i}\widetilde{c}_k$$

D'où la formule récursive. □

Remarque : La formule (2.12) nécessite elle-aussi de pouvoir *a priori* diviser dans l'anneau A par les entiers compris entre 1 et s ainsi que par l'entier r.

Il est facile de transformer le lemme 2.41 en un algorithme qui calcule la racine r-ième de polynômes dont le terme constant est égal à 1. Nous supposons que A est de caractéristique nulle ou supérieure à s et qu'elle ne divise pas r :

Algorithme de J. C. P. Miller

Entrée : $r \geq 1$, s un entier positif, $\widetilde{p} \bmod t^{s+1} \in A[t]$
tel que $\widetilde{p} = \widetilde{q}^{\,r} \bmod t^{s+1}$ et $\widetilde{p}(0) = 1$; $\{\widetilde{p} = 1 + \widetilde{a}_1 t + \cdots + \widetilde{a}_s t^s\}$
Sortie : $\widetilde{q}$. $\{\widetilde{q} = 1 + \widetilde{c}_1 t + \cdots + \widetilde{c}_{s-1} t^{s-1} + \widetilde{c}_s t^s\ \}$

(1) Si $r = 1$ alors retourne $\widetilde{p}$
(2) $\widetilde{c}_0 \leftarrow 1$; $\widetilde{q} = 1$
(3) Pour k passant de 1 à s répète :
$\quad \widetilde{c}_k \leftarrow (\sum_{i=0}^{k-1}((k-(r+1)i)\widetilde{c}_i\widetilde{a}_{k-i})/r)/k$
$\quad \widetilde{q} \leftarrow \widetilde{q} + \widetilde{c}_k t^k$
(4) Retourne $\widetilde{q}$

Complexité: Le calcul de $\widetilde{q}$ nécessite seulement $s(s+1)$ multiplications dans A, $s(s-1)/2$ additions dans A et s divisions par un entier. Ce qui assure la supériorité de cet algorithme sur les méthodes asymptotiquement plus performantes décrites au paragraphe 2.5.5, tant que s reste petit (typiquement inférieur ou égal à 70).

Nous résumons le résultat de ce paragraphe dans un lemme.

Lemme 2.42. *Soit A un anneau intègre de caractéristique nulle. Soit $\widetilde{p} \in A[t]$ un polynôme de degré n ayant 1 pour terme constant :*

$$\widetilde{p}(t) = 1 + \widetilde{a}_1 t + \cdots + \widetilde{a}_{n-1} t^{n-1} + \widetilde{a}_n t^n$$

tel que

$$\widetilde{p}(t) = \widetilde{q}(t)^r \qquad et \qquad n = rs$$

alors $\widetilde{q}(t)$, de degré s, peut être calculé à partir de :

$$1 + \widetilde{a}_1 t + \cdots + \widetilde{a}_s t^s = \widetilde{p}(t) \bmod t^{s+1} \quad .$$

Preuve : Utiliser un des algorithmes précédents. □

En fait, à l'examen des algorithmes, l'hypothèse *de caractéristique nulle* peut être remplacée par :

Lemme 2.43. *Soit A un anneau intègre de caractéristique ℓ. Soit $\widetilde{p} \in A[t]$ un polynôme de degré n ayant 1 pour terme constant :*

$$\widetilde{p}(t) = 1 + \widetilde{a}_1 t + \cdots + \widetilde{a}_{n-1} t^{n-1} + \widetilde{a}_n t^n$$

tel que

$$\widetilde{p}(t) = \widetilde{q}(t)^r \qquad et \qquad n = rs$$

alors, si $\ell > s$ et ℓ ne divise pas r, $\widetilde{q}(t)$, de degré s, peut être calculé à partir de :

$$1 + \widetilde{a}_1 t + \cdots + \widetilde{a}_s t^s = \widetilde{p}(t) \bmod t^{s+1} \quad .$$

Preuve : En utilisant la formule de J. C. P. Miller ou l'itération de Newton avec la transformée de Fourier rapide. □

2.6 Le calcul de résultants sur un anneau non intègre

Le lemme 2.24 montre que le calcul du résultant de deux polynômes à coefficients dans $A[t]$ est compatible avec la réduction de leurs coefficients modulo t^{s+1}. La méthode la plus efficace couramment utilisée pour calculer le résultant est l'algorithme des sous-résultants qui effectue des divisions exactes dans l'anneau. Mais, même si A est un anneau intègre, l'anneau

$A[t]/(t^{s+1})$ n'est pas intègre. Ceci nous empêche d'utiliser l'algorithme des sous-résultants. Ainsi, nous pouvons perdre en complexité l'économie que nous avons faite sur la taille des polynômes à manipuler pour achever le calcul.

Un moyen de contourner ce problème est de faire appel aux méthodes de calcul du résultant sans faire appel aux divisions, qui peuvent donc être utilisées sur des anneaux non-intègres, par exemple [11]. L'état de l'art sur le sujet peut être trouvé dans [3].

L'intérêt de faire appel à une telle classe d'algorithmes est aussi de pouvoir effectuer les transformations du paragraphe 3.6 modulo $f(y)$ ce qui permettra là aussi d'éviter le stockage de termes inutiles (cependant effectuer un modulo $f(y)$ est un peu plus coûteux que de tronquer les polynômes modulo t^{s+1}).

2.7 Conclusion

Nous avons présenté dans ce chapitre plusieurs algorithmes élémentaires de manipulation de polynômes. En particulier, nous avons vu comment calculer la racine r-ième d'un polynôme défini sur un anneau intègre, utilisables sous quelques hypothèses supplémentaires en caractéristique non-nulle. Les méthodes basées sur le calcul du pgcd de polynômes (paragraphes 2.5.2 et 2.5.3) sont très coûteuses, même si elles fournissent parfois une information supplémentaire : une factorisation partielle de la racine.

Les méthodes d'origine analytiques sont beaucoup plus efficaces, et ont en plus l'avantage de ne nécessiter que la connaissance des $s+1$ premiers termes du polynôme dont est recherchée la racine r-ième. Si le polynôme p est obtenu par un calcul, n'avoir à en calculer que ses $s+1$ premiers termes pour ensuite en déduire la racine r-ième allégera forcément le calcul de p. Nous tirerons pleinement parti de cette propriété pour le calcul des résolvantes dans les chapitres 3 et 4. Notre intérêt pour ces méthodes a été suscité par [107] et [105].

Nous introduisons une notation pour les algorithmes de racine r-ième.

Notation 2.44. Soient r et s des entiers. Soit $p \in A[t]$ un polynôme à coefficients dans l'anneau intègre A dont le terme constant est égal à 1 et dont les termes de degré inférieur ou égal à s correspondent à ceux d'un polynôme de degré s à coefficients dans A élevé à la puissance r. Sous les hypothèses adéquates de l'un des lemmes 2.42, 2.43, ou 2.39, $\sqrt[r]{p(t) \bmod t^{s+1}}$ désigne la *racine r-ième exacte* de p modulo t^{s+1}, calculée seulement à partir des termes de p de degré inférieur ou égal à s :

$$\left(\sqrt[r]{p(t) \bmod t^{s+1}}\right)^r = p(t) \bmod t^{s+1} \quad .$$

Chapitre 3

Calcul récursif de résolvantes de Lagrange absolues

Ce chapitre décrit un algorithme pour le calcul formel des résolvantes de Lagrange. Il était pratiqué par J. L. LAGRANGE sur des exemples [59]. Malheureusement, apparaissent au cours des calculs des facteurs et des multiplicités parasites. L. E. SOICHER ([89] ou [90]) semble l'avoir redécouvert indépendamment dans un cas particulier à partir de certaines propriétés des résultants. Sa méthode qui supprime les facteurs parasites a été améliorée dans [61]. Nous en donnons une nouvelle description dans le cas général. Cette technique souffre du recours à des calculs symboliques très lourds en espace mémoire. Nous y avons apporté trois contributions : l'« échange » du calcul du quotient avec le calcul de résultant (le résultant est calculé sur des degrés beaucoup plus petits), l'utilisation de calculs sur les polynômes réciproques tronqués (pour ne conserver que les termes strictement nécessaires) ainsi que la caractérisation d'une classe d'invariants, les invariants « simples », (qui contient ceux que calcule L. E. SOICHER) qui correspondent aux cas dans lesquels cette méthode est la plus efficace.

3.1 Définitions

Nous considérons dans ce chapitre, A un anneau intègre, K son corps des fractions et f un polynôme **unitaire** séparable de degré n à coefficients dans A dont les racines distinctes dans une extension algébrique $\hat{K}$ de K sont $\alpha_1, \ldots, \alpha_n$.

L. E. SOICHER a proposé dans [89] (voir aussi [90]) d'utiliser les résultants pour calculer les résolvantes associées à une certaine classe d'invariants : les *invariants linéaires*.

Nous présentons ici une généralisation de cet algorithme qui tire pleinement partie de la remarque de C. WILLIAMSON [113] déjà exploitée dans [61].

Définition 3.1. Soit k un entier positif. Notons $[\,]$ l'application « d'évaluation » de l'anneau $A[x_1, \ldots, x_{k+1}]$ dans $A[y][x_1, \ldots, x_k]$ définie par :

$$\begin{array}{rccc} [\,] : & A[x_1, \ldots, x_{k+1}] & \longrightarrow & A[y][x_1, \ldots, x_k] \\ & \Theta & \longmapsto & [\Theta] = \Theta(y, x_1, \ldots, x_k) \end{array} .$$

Exemples : $[x_1+x_2+x_3]=y+x_1+x_2$, $[x_1^5+x_2(x_3-x_1^2)]=y^5+x_1(x_2-y^2)$, $[x_2x_3x_4]=x_1x_2x_3$.

Définition 3.2. L'application, notée « . », associe à un motif de partition m (défini au paragraphe 2.4) de longueur k, de niveau ℓ vérifiant $0\leq\ell\leq k$ et à un polynôme Θ de $A[x_1,\ldots,x_{k+1}]$ un polynôme $m.\Theta$ de $A[x_1,\ldots,x_{k+1-\ell}]$:

$$\begin{array}{rccc} . : & M(\ell,k)\times A[x_1,\ldots,x_{k+1}] & \longrightarrow & A[x_1,\ldots,x_{k+1-\ell}] \\ & (m,\Theta) & \longmapsto & m.\Theta=\Theta(m.x_1,\ldots,m.x_{k+1}) \end{array} \quad ,$$

où en posant $m=m_1m_2\cdots m_k$ avec $m_i\in\{0,1\}$ pour $1\leq i\leq k$,

$$\text{et pour}\quad 2\leq j\leq k+1, \qquad \begin{array}{l} m.x_1=x_1 \\ m.x_j=\left\{\begin{array}{l} x_1 \text{ si } m_{j-1}=1 \\ x_{i+1} \text{ si } m_{j-1} \text{ est le } i\text{-ième zéro de } m \end{array}\right. \end{array} .$$

L'application « . » consiste donc à renommer en x_1 les ℓ variables x_{j-1} où j correspond aux coordonnées égales à 1 du motif de partition m et à renommer en x_{i+1} les variables x_{j-1} où j correspond à l'indice du i-ième zéro de m.

Exemples : Nous plaçons au-dessus de chaque variable la valeur de la coordonnée du motif de partition qui la conditionne.

Pour $\Theta\in A[x_1,\ldots,x_8]$, et $m_1=0000100\in M(7,1)$, $m_2=0000110\in M(7,2)$, $m_3=1101000\in M(7,3)$:

$$\begin{aligned} m_1.\Theta &= \Theta(x_1,\overset{0}{x_2},\overset{0}{x_3},\overset{0}{x_4},\overset{0}{x_5},\overset{1}{x_1},\overset{0}{x_6},\overset{0}{x_7})\in A[x_1,x_2,x_3,x_4,x_5,x_6,x_7] \\ m_2.\Theta &= \Theta(x_1,\overset{0}{x_2},\overset{0}{x_3},\overset{0}{x_4},\overset{0}{x_5},\overset{1}{x_1},\overset{1}{x_1},\overset{0}{x_6})\in A[x_1,x_2,x_3,x_4,x_5,x_6] \\ m_3.\Theta &= \Theta(x_1,\overset{1}{x_1},\overset{1}{x_1},\overset{0}{x_2},\overset{1}{x_1},\overset{0}{x_3},\overset{0}{x_4},\overset{0}{x_5})\in A[x_1,x_2,x_3,x_4,x_5] \end{aligned}$$

3.2 La méthode de Soicher

Soit $\Theta\in A[x_1,\ldots,x_{k+1}]$ un invariant d'arité au plus $k+1$. La méthode qui consiste à utiliser des résultants pour éliminer successivement les variables $x_1,\ldots,x_{k+1}$ du polynôme $t-\Theta$ est décrite sur des exemples par J. L. Lagrange dans [59] (il ne disposait pas alors de l'outil résultant qui date de la fin du 19$^{\text{e}}$ siècle). Elle a été utilisée par L. E. Soicher [89]. C'est aussi la méthode proposée par M. Giusti, D. Lazard et A. Valibouze dans [45]. Cependant ces derniers indiquent seulement que des polynômes supplémentaires apparaissent au cours du calcul : ils correspondent aux cas d'égalité entre les valeurs des variables lorsque les racines de f leur sont successivement substituées. L. E. Soicher a décrit exactement la nature de ces polynômes superflus dans le cas d'invariants particuliers, les *invariants linéaires*, qui s'écrivent sous la forme

$$\Theta(x_1,\ldots,x_{k+1})=e_1x_1+\cdots+e_{k+1}x_{k+1} \quad , \tag{3.1}$$

avec $(e_1,\ldots,e_{k+1})\in A^{k+1}$.

Ce chapitre contient la description d'un algorithme qui calcule récursivement ces polynômes supplémentaires dans le cas d'un invariant quelconque.

Plutôt que de diviser comme L. E. SOICHER le résultat final par les polynômes supplémentaires (considérés comme *superflus*), nous allons chercher à effectuer la division avant même de calculer le résultant, ceci dans le but de contenir l'explosion du nombre de termes symboliques à manipuler en machine.

La méthode de Soicher consiste, dans le cas des invariants linéaires de la forme (3.1) à éliminer une variable de Θ par la propriété 2.21 (du paragraphe 2.3) et à diviser le résultat par les polynômes superflus.

Cette méthode est illustrée ci-après par deux exemples.

Exemple 1, l'invariant linéaire $\Theta = x_1 + 2x_2 + 4x_3$

L. E. SOICHER remarque que la résolvante du polynôme f pour l'invariant Θ d'arité 3 vérifie :

$$\mathcal{L}_{x_1+2x_2+4x_3,f}(t) = \frac{\text{Rés}_y\left(f(y), \mathcal{L}_{2x_1+4x_2,f}(t-y)\right)}{\mathcal{L}_{3x_1+4x_2,f}(t) \times \mathcal{L}_{5x_1+2x_2,f}(t)} \quad . \tag{3.2}$$

Les résolvantes de Lagrange associées à des invariants d'arité 2 apparaissant dans cette formule peuvent être calculées (récursivement) de la même façon :

$$\mathcal{L}_{2x_1+4x_2,f}(t) = \frac{\text{Rés}_y\left(f(y), \mathcal{L}_{4x_1,f}(t-2y)\right)}{\mathcal{L}_{6x_1,f}(t)} \tag{3.3}$$

$$\mathcal{L}_{3x_1+4x_2,f}(t) = \frac{\text{Rés}_y\left(f(y), \mathcal{L}_{4x_1,f}(t-3y)\right)}{\mathcal{L}_{7x_1,f}(t)} \tag{3.4}$$

$$\mathcal{L}_{5x_1+2x_2,f}(t) = \frac{\text{Rés}_y\left(f(y), \mathcal{L}_{5x_1,f}(t-2y)\right)}{\mathcal{L}_{7x_1,f}(t)} \tag{3.5}$$

Le procédé qui fait décroître strictement l'arité des divers invariants considérés s'arrête sur les résolvantes associées aux invariants d'arité 1 restants, elles peuvent être aussi bien calculées par la formule

$$\mathcal{L}_{ex_1,f}(t) = \text{Rés}_y\left(f(y), t - ey\right) \tag{3.6}$$

que par

$$\mathcal{L}_{ex_1,f}(t) = e^{\deg_y(f(y))} f\left(\frac{t}{e}\right)$$

pour e prenant les valeurs $4, 5, 6$ et 7.

En réinjectant les expressions (3.4), (3.5) et (3.6) dans (3.2) et grâce à la propriété 2.22 du résultant, le calcul de $\mathcal{L}_{x_1+2x_2+4x_3,f}$ devient :

$$\mathcal{L}_{x_1+2x_2+4x_3,f}(t) = \frac{\text{Rés}_y\left(f(y), \mathcal{L}_{2x_1+4x_2,f}(t-y) \times (t-7y)^2\right)}{\text{Rés}_y\left(f(y), \mathcal{L}_{4x_1,f}(t-3y) \times \mathcal{L}_{5x_1,f}(t-2y)\right)} \quad . \tag{3.7}$$

Les résolvantes apparaissant dans (3.7) sont elles-mêmes calculées récursivement par la méthode de Soicher grâce aux formules (3.3) et (3.6).

C. J. WILLIAMSON a observé dans [113] que pour les invariants linéaires d'arité 2, la division peut être effectuée avant le calcul de résultant.

Proposition 3.3 (Williamson). *Pour $e_1 \neq e_2$,*

$$\begin{aligned}
\mathcal{L}_{e_1x_1+e_2x_2,f}(t) &= \frac{\text{Rés}_y\left(f(y), \mathcal{L}_{e_2x_1,f}(t-e_1y)\right)}{\text{Rés}_y\left(f(y), t-(e_1+e_2)y\right)} \\
&= \frac{\text{Rés}_y\left(f(y), \mathcal{L}_{e_2x_1,f}(t-e_1y) - \mathcal{L}_{e_2x_1,f}(e_2y)\right)}{\text{Rés}_y\left(f(y), t-(e_1+e_2)y\right)} \\
&= \text{Rés}_y\left(f(y), \frac{\mathcal{L}_{e_2x_1,f}(t-e_1y) - \mathcal{L}_{e_2x_1,f}(e_2y)}{t-(e_1+e_2)y}\right) \quad .
\end{aligned}$$

Preuve : Il suffit d'appliquer la propriété 2.21. □

Nous avons généralisé dans [61] ce résultat, à des invariants de plus grande arité : il suffit de prendre le quotient de la division par rapport à t (noté $\lfloor - \rfloor_t$, voir le paragraphe 2.3.4).

La formule (3.7) devient alors :

$$\mathcal{L}_{x_1+2x_2+4x_3,f}(t) = \text{Rés}_y\left(f(y), \left\lfloor \frac{N(t,y)}{D(t,y)} \right\rfloor_t\right) \tag{3.8}$$

avec

$$\begin{aligned}
& N(t,y) = \mathcal{L}_{2x_1+4x_2,f}(t-y) \times (t-7y)^2 \\
\text{et} \quad & D(t,y) = \mathcal{L}_{4x_1,f}(t-3y) \times \mathcal{L}_{5x_1,f}(t-2y) \quad .
\end{aligned}$$

Exemple 2, l'invariant linéaire $\Theta = x_1 + x_2 + x_3$

Pour l'invariant Θ, L. E. SOICHER remarque que la résolvante associée vérifie :

$$\mathcal{L}_{x_1+x_2+x_3,f}(t) = \sqrt[3]{\frac{\text{Rés}_y\left(f(y), \mathcal{L}_{x_1+x_2,f}(t-y)\right)}{\mathcal{L}_{x_1+2x_2,f}(t)}} \quad . \tag{3.9}$$

Les résolvantes de Lagrange associées à des invariants d'arité 2 apparaissant dans cette formule peuvent être calculées (récursivement) de la même façon :

$$\mathcal{L}_{x_1+2x_2,f}(t) = \frac{\text{Rés}_y\left(f(y), \mathcal{L}_{2x_1,f}(t-y)\right)}{\mathcal{L}_{3x_1,f}(t)} \tag{3.10}$$

$$\mathcal{L}_{x_1+x_2,f}(t) = \sqrt[2]{\frac{\text{Rés}_y\left(f(y), \mathcal{L}_{x_1,f}(t-y)\right)}{\mathcal{L}_{2x_1,f}(t)}} \tag{3.11}$$

Quant aux résolvantes d'arité 1 restantes, elles sont calculées par la formule (3.6) pour ϵ prenant les valeurs $1, 2$ et 3.

Comme dans l'exemple précédent, en réinjectant les expressions (3.10), et (3.6) dans (3.9), le calcul de $\mathcal{L}_{x_1+x_2+x_3,f}$ devient :

$$\mathcal{L}_{x_1+x_2+x_3,f}(t) = \sqrt[3]{\text{Rés}_y\left(f(y), \left\lfloor \frac{N(t,y)}{D(t,y)} \right\rfloor_t\right)} \tag{3.12}$$

avec

$$N(t,y) = \mathcal{L}_{x_1+x_2,f}(t-y) \times (t-3y)$$
$$\text{et} \qquad D(t,y) = \mathcal{L}_{2x_1,f}(t-y) \quad .$$

Nous utiliserons au paragraphe 3.5 les propriétés de certaines méthodes de calcul de racines r-ièmes exposées au paragraphe 2.5 pour contribuer à diminuer le nombre de calculs symboliques à stocker dans les cas où apparaissent de telles puissances.

3.3 Cas d'un invariant quelconque

Pour généraliser la méthode de Soicher à des invariants quelconques d'arité $k+1$, ce paragraphe décrit le calcul récursif des polynômes f-évalués définis au paragraphe 2.7 en faisant décroître strictement l'arité des invariants considérés à chaque appel récursif.

Propriété 3.4. *Soit $\Theta \in A[x_1, \ldots, x_{k+1}]$ un invariant d'arité $k+1$. L'élimination de la variable y de $[\Theta] \in A[y][x_1, \ldots, x_k]$ par le résultant donne:*

$$\text{Rés}_y\left(f(y), \mathcal{E}_{[\Theta],f}\right) = \mathcal{E}_{\Theta,f} \prod_{m \in M(1,k)} \mathcal{E}_{m.\Theta,f}$$

où $\mathcal{E}_{[\Theta],f} \in A[y]$ est calculé en considérant la définition 2.7 du polynôme f-évalué sur l'anneau $A[y]$ pour l'invariant $[\Theta] \in A[y][x_1, \ldots, x_k]$ d'arité k.

Soit encore :

$$\mathcal{E}_{\Theta,f} = \frac{\text{Rés}_y\left(f(y), \mathcal{E}_{[\Theta],f}\right)}{\prod_{m \in M(1,k)} \mathcal{E}_{m.\Theta,f}} \quad . \tag{3.13}$$

Preuve : Le résultant de f avec le polynôme $\mathcal{E}_{[\Theta],f}$ par rapport à la variable y est, d'après la propriété 2.21, le produit des polynômes $\mathcal{E}_{[\Theta],f}$ pour y évalué en chacune des racines de f. Or la définition de $\mathcal{E}_{\Theta,f}$ exclut les cas d'égalités des racines deux à deux. Ce sont ces k cas d'égalités introduits par le résultant qui apparaissent au dénominateur de la formule (3.13). □

Les polynômes au dénominateur de l'expression (3.13) peuvent eux-mêmes se calculer par élimination d'une variable en appliquant la propriété 3.4 à chacun des $m.\Theta \in A[x_1, \ldots, x_k]$

pour m parcourant $M(1,k)$:

$$\mathcal{E}_{m.\Theta,f} = \frac{\mathrm{Rés}_y\left(f(y), \mathcal{E}_{[m.\Theta],f}\right)}{\prod_{m' \in M(1,k-1)} \mathcal{E}_{m'.(m.\Theta),f}} \quad . \tag{3.14}$$

L'équation (3.13) devient donc :

$$\mathcal{E}_{\Theta,f} = \frac{\mathrm{Rés}_y\left(f(y), \mathcal{E}_{[\Theta],f}\right)}{\prod_{m \in M(1,k)} \mathrm{Rés}_y\left(f(y), \mathcal{E}_{[m.\Theta],f}\right)} \prod_{m \in M(1,k), m' \in M(1,k-1)} \mathcal{E}_{m'.(m.\Theta),f} \quad . \tag{3.15}$$

Pour calculer $\mathcal{E}_{[m.\Theta],f}$, l'ordre dans lequel les variables x_i de $m'.(m.\Theta)$ sont identifiées deux à deux par « . » importe peu, car, par définition, $\mathcal{E}$ évalue les variables x_i des $m'.(m.\Theta)$, pour $1 \leq i \leq k-1$, en toutes les racines de f.

Nous utiliserons donc la propriété suivante :

Propriété 3.5. *Soit $\Theta \in A[x_1, \ldots, x_{k+1}]$ un invariant d'arité $k+1$. Pour $0 \leq j < k$,*

$$\prod_{m \in M(j,k), m' \in M(1,k-j)} \mathcal{E}_{m'.(m.\Theta),f} = \prod_{m \in M(j+1,k)} \left(\mathcal{E}_{m.\Theta,f}\right)^{j+1} \quad .$$

Preuve : La liste de longueur $\mathrm{card}\left(M(1,k-j)\right) \times \mathrm{card}\left(M(j,k)\right) = (k-j) \times k!/((k-j)!j!)$ des invariants

$$\{m'.(m.\Theta) \mid m \in M(j,k), m' \in M(1,k-j)\}$$

ne contient que $k!/((k-(j+1))!(j+1)!)$ éléments distincts. Chacun est obtenu avec la multiplicité $j+1$. □

En utilisant la propriété 3.5, l'équation (3.15) s'écrit aussi :

$$\mathcal{E}_{\Theta,f} = \frac{\mathrm{Rés}_y\left(f(y), \mathcal{E}_{[\Theta],f}\right)}{\prod_{m \in M(1,k)} \mathrm{Rés}_y\left(f(y), \mathcal{E}_{[m.\Theta],f}\right)} \prod_{m \in M(2,k)} \left(\mathcal{E}_{m.\Theta,f}\right)^2 \quad . \tag{3.16}$$

L'induction du processus, qui fait strictement décroître l'arité des invariants successifs, sur les polynômes $\mathcal{E}_{m.\Theta,f}$ jusqu'à éliminer toutes les variables donne la formule générale :

$$\begin{aligned}\mathcal{E}_{\Theta,f} = \mathrm{Rés}_y\left(f(y), \mathcal{E}_{[\Theta],f}\right) \prod_{m \in M(1,k)} \mathrm{Rés}_y\left(f(y), \mathcal{E}_{[m.\Theta],f}\right)^{-1!} \prod_{m \in M(2,k)} \mathrm{Rés}_y\left(f(y), \mathcal{E}_{[m.\Theta],f}\right)^{2!} \\ \prod_{m \in M(3,k)} \mathrm{Rés}_y\left(f(y), \mathcal{E}_{[m.\Theta],f}\right)^{-3!} \cdots \prod_{m \in M(k,k)} \mathrm{Rés}_y\left(f(y), \mathcal{E}_{[m.\Theta],f}\right)^{(-1)^k k!} \quad .\end{aligned} \tag{3.17}$$

La formule (3.17) fournit une méthode permettant de calculer le polynôme $\mathcal{E}_{\Theta,f}$: les invariants $[m.\Theta] \in A[y][x_1, \ldots, x_{k-i-1}]$, pour $m \in M(i,k)$, apparaissant dans le membre

droit de l'égalité (3.17) sont tous d'arité strictement inférieure à celle de Θ, et la formule (3.17) peut leur être appliquée récursivement (sur l'anneau $A[y]$).

Cependant, nous n'allons pas utiliser la formule (3.17) directement. Il est possible de lui appliquer le lemme 2.26.

Notation 3.6. Soit Θ un invariant d'arité $k+1$. Le polynôme $P(i,k) \in A[y]$ désigne le produit :

$$P(i,k) = \prod_{m \in M(i,k)} \left(\mathcal{E}_{[m.\Theta],f}\right)^{i!} \quad .$$

L'utilisation du lemme 2.26 et de la notation 3.6 pour la formule (3.17) donne :

$$\mathcal{E}_{\Theta,f} = \text{Rés}_y \left(f(y), \left\lfloor \frac{N(t,y)}{D(t,y)} \right\rfloor_t \right) \tag{3.18}$$

où, si k est impair,

$$\begin{aligned} & N(t,y) = P(0,k)P(2,k)\cdots P(k-1,k) \\ \text{et} \quad & D(t,y) = P(1,k)P(3,k)\cdots P(k,k) \end{aligned}$$

sinon, si k est pair,

$$\begin{aligned} & N(t,y) = P(0,k)P(2,k)\cdots P(k,k) \\ \text{et} \quad & D(t,y) = P(1,k)P(3,k)\cdots P(k-1,k) \quad . \end{aligned}$$

Dans les deux cas les polynômes $N(t,y)$ et $D(t,y)$ vérifient les hypothèses du lemme 2.26.

Les polynômes $\mathcal{E}_{[m.\Theta],f} \in A[y]$ qui apparaissent dans $N(t,y)$ et $D(t,y)$, avec $m.\Theta \in A[y][x_1, \ldots, x_{k-i-1}]$, pour $m \in M(i,k)$, peuvent être eux-mêmes calculés récursivement par la formule (3.18) sur l'anneau $A[y]$.

La formule (3.18) fournit donc une méthode permettant de calculer récursivement le polynôme $\mathcal{E}_{\Theta,f}$.

Dans les paragraphes 3.4 et 3.5 cette méthode, qui nécessite *a priori* de très gros calculs symboliques sera améliorée. Puis sera traité au paragraphe 3.6 un cas particulier où l'explosion des calculs symboliques est contenue.

3.4 Élimination de puissances, acte 1

Au paragraphe 3.3, dans le calcul à partir d'élimination des variables d'un invariant Θ d'arité $k+1$, peuvent apparaître aussi des puissances superflues. Elles sont dues aux symétries de l'invariant (voir [45] et la proposition 2.10) :

$$\mathcal{E}_{\Theta,f} = \left(\mathcal{L}_{\Theta,f}\right)^{\text{card}\left(\text{Stab}_{\mathfrak{S}_{k+1}}(\Theta)\right)} \quad . \tag{3.19}$$

L'utilisation de la relation (3.19) dans l'équation (3.18) donne :

$$\left(\mathcal{L}_{\Theta,f}\right)^{\text{card}\left(\text{Stab}_{\mathfrak{S}_{k+1}}(\Theta)\right)} = \text{Rés}_y \left(f(y), \left\lfloor \frac{N(t,y)}{D(t,y)} \right\rfloor_t \right) \tag{3.20}$$

où $N(t,y)$ et $D(t,y)$ sont donnés dans la formule (3.18) en remarquant que

$$P(i,k) = \prod_{m \in M(i,k)} \left(\mathcal{L}_{[m.\Theta],f}\right)^{i!\,\mathrm{card}\left(\mathrm{Stab}_{\mathfrak{S}_{k-i}}([m.\Theta])\right)} \quad ,$$

car $[m.\Theta] \in A[x_1, \ldots, x_{k-i}]$.

Pour minimiser les degrés des polynômes apparaissant dans les calculs, il est préférable de remplacer le calcul récursif des polynômes $\mathcal{E}_{[m.\Theta],f}$ par celui des résolvantes $\mathcal{L}_{[m.\Theta],f}$.

Le calcul de résultant étant compatible avec l'élévation à la puissance (voir propriété 2.23), il est possible d'enlever les puissances communes connues *a priori*, les évaluations $[m.\Theta]$ n'étant pas forcément toujours toutes distinctes.

Soit $E(i,k)$ la liste d'invariants dont les éléments peuvent se répéter :

$$E(i,k) = \{[m.\Theta] \mid m \in M(i,k)\} \quad .$$

La présence d'éléments égaux dépend des symétries de Θ.

Notation 3.7. Soit $\equiv$ la relation d'équivalence définie, pour m et m' dans $M(i,k)$ par :

$$m \equiv m' \qquad \text{si} \qquad [m.\Theta] = [m'.\Theta] \quad .$$

Pour $m \in M(i,k)$, $\overline{m}$ désigne sa classe d'équivalence et $\overline{M}(i,k)$ désigne l'ensemble des classes d'équivalence de $M(i,k)$.

Le cardinal de la classe d'équivalence $\overline{m}$ est noté $d_{\overline{m}}$.

La valeur de $[\overline{m}.\Theta]$ est celle de $[m.\Theta]$ pour n'importe quel représentant m de la classe de conjugaison $\overline{m}$.

Donc, $E(i,k)$ s'écrit aussi sous la forme :

$$E(i,k) = \left\{[\overline{m}.\Theta]^{d_{\overline{m}}} \mid \overline{m} \in \overline{M}(i,k)\right\} \quad ,$$

en notant les multiplicités sous forme de puissances.

Nous pouvons donc écrire pour $0 \leq i \leq k$,

$$P(i,k) = \prod_{\overline{m} \in \overline{M}(i,k)} \left(\mathcal{L}_{[\overline{m}.\Theta],f}\right)^{i!\,\mathrm{card}\left(\mathrm{Stab}_{\mathfrak{S}_{k-i}}([\overline{m}.\Theta])\right) d_{\overline{m}}} \quad . \tag{3.21}$$

Exemples : La notation sous forme de puissances correspond aux multiplicités.

1. Pour $\Theta = x_1 + 2x_2 + 4x_3$,
   $M(1,2) = \{10, 01\}$, $\overline{M}(1,2) = \{\overline{10}, \overline{01}\}$,
   $E(1,2) = \{3y + 4x_1, 5y + 2x_1\}$,
   $M(2,2) = \{11\}$, $\overline{M}(2,2) = \{\overline{11}\}$,
   $E(2,2) = \{7y\}$.

2. Pour $\Theta = x_1 + x_2 + x_3$,
   $M(1,2) = \{10, 01\}$, $\overline{M}(1,2) = \{\overline{10}\}$ et $d_{\overline{10}} = 2$,
   $E(1,2) = \{(y + 2x_1)^2\}$,
   $M(2,2) = \{11\}$, $\overline{M}(2,2) = \{\overline{11}\}$,
   $E(2,2) = \{3y\}$.

3. Pour $\Theta = x_1 + 2x_2 + 4x_3 + 8x_4$,
   $M(1,3) = \{100, 010, 001\}$, $\overline{M}(1,3) = \{\overline{100}, \overline{010}, \overline{001}\}$,
   $E(1,3) = \{3y + 4x_1 + 8x_2, 5y + 2x_1 + 8x_2, 9y + 2x_1 + 4x_2\}$,
   $M(2,3) = \{110, 101, 011\}$, $\overline{M}(2,3) = \{\overline{110}, \overline{101}, \overline{011}\}$,
   $E(2,3) = \{7y + 8x_1, 11y + 4x_1, 13y + 2x_1\}$,
   $M(3,3) = \{111\}$, $\overline{M}(3,3) = \{\overline{111}\}$,
   $E(3,3) = \{15y\}$.

4. Pour $\Theta = x_1 + x_2 + x_3 + x_4$,
   $M(1,3) = \{100, 010, 001\}$, $\overline{M}(1,3) = \{\overline{100}\}$ et $d_{\overline{100}} = 3$,
   $E(1,3) = \{(2y + x_1 + x_2)^3\}$,
   $M(2,3) = \{110, 101, 011\}$, $\overline{M}(2,3) = \{\overline{110}\}$ et $d_{\overline{110}} = 3$,
   $E(2,3) = \{(3y + x_1)^3\}$,
   $M(3,3) = \{111\}$, $\overline{M}(3,3) = \{\overline{111}\}$,
   $E(3,3) = \{4y\}$.

5. Pour $\Theta = x_1 + x_2 + x_3 + 4x_4$,
   $M(1,3) = \{100, 010, 001\}$, $\overline{M}(1,3) = \{\overline{100}, \overline{001}\}$ et $d_{\overline{100}} = 2$, $d_{\overline{001}} = 1$,
   $E(1,3) = \{(2y + x_1 + 4x_2)^2, 5y + x_1 + x_2\}$,
   $M(2,3) = \{110, 101, 011\}$, $\overline{M}(2,3) = \{\overline{110}, \overline{101}\}$ et $d_{\overline{110}} = 1$, $d_{\overline{101}} = 2$,
   $E(2,3) = \{3y + 4x_1, (6y + x_1)^2\}$,
   $M(3,3) = \{111\}$, $\overline{M}(3,3) = \{\overline{111}\}$,
   $E(3,3) = \{7y\}$.

6. Pour $\Theta = x_1x_2 + x_3x_4$,
   $M(1,3) = \{100, 010, 001\}$, $\overline{M}(1,3) = \{\overline{100}, \overline{010}\}$ et $d_{\overline{100}} = 1$, $d_{\overline{010}} = 2$,
   $E(1,3) = \{y^2 + x_1x_2, (y(x_1 + x_2))^2\}$,
   $M(2,3) = \{110, 101, 011\}$, $\overline{M}(2,3) = \{\overline{110}\}$ et $d_{\overline{110}} = 3$,
   $E(2,3) = \{(y^2 + yx_1)^3\}$,
   $M(3,3) = \{111\}$, $\overline{M}(3,3) = \{\overline{111}\}$,
   $E(3,3) = \{2y^2\}$.

Pour éviter des calculs de puissances superflues nous calculons le pgcd des puissances apparaissant dans (3.21) :

$$d = \text{pgcd}\left\{ i!\,\text{card}\left(\text{Stab}_{\mathfrak{S}_{k-i}}([\overline{m}.\Theta])\right)\, d_{\overline{m}} \;\middle|\; \overline{m} \in \overline{M}(i,k), 0 \leq i \leq k \right\} \quad .$$

et ne retenons que la racine d-ième nécessaire au calcul.

Notation 3.8. Soit

$$L(i,k) = P(i,k)^{1/d} \quad . \tag{3.22}$$

En prenant la racine d-ième, le calcul (3.20) devient :

$$\mathcal{L}_{\Theta}^{\text{card}\left(\text{Stab}_{\mathfrak{S}_{k+1}}(\Theta)\right)/d} = \text{Rés}_y\left(f(y), \left\lfloor \frac{N(t,y)}{D(t,y)} \right\rfloor_t\right) \tag{3.23}$$

où, si k est impair,

$$\begin{aligned} & N(t,y) = L(0,k)L(2,k)\cdots L(k-1,k) \\ \text{et} \quad & D(t,y) = L(1,k)L(3,k)\cdots L(k,k) \end{aligned}$$

sinon, si k est pair,

$$\begin{aligned} & N(t,y) = L(0,k)L(2,k)\cdots L(k,k) \\ \text{et} \quad & D(t,y) = L(1,k)L(3,k)\cdots L(k-1,k) \quad . \end{aligned}$$

Nous avons vu dans ce paragraphe comment éliminer certaines puissances *a priori* dans le calcul récursif de résolvantes, cependant, dans (3.23) la résolvante cherchée peut encore apparaître avec une certaine puissance, car $\text{card}\left(\text{Stab}_{\mathfrak{S}_{k+1}}(\Theta)\right)/d$ peut ne pas être égal à 1 : nous calculons donc parfois toujours trop de termes, qui sont très consommateurs d'espace mémoire et alourdissent inutilement les calculs symboliques.

3.5 Élimination de puissances, acte 2

Le paragraphe 2.5 a rappelé des méthodes de calcul de racines r-ièmes montrant qu'il n'était pas nécessaire de connaître tous les termes d'un polynôme pour pouvoir en calculer sa racine exacte. Nous allons utiliser ici cette technique pour limiter le nombre de termes effectivement calculés.

Soit k un entier positif, et Θ un invariant d'arité au plus $k+1$, le degré de la résolvante de Lagrange absolue $\mathcal{L}_{\Theta,f}$, noté s, est

$$s = \frac{n!}{(n-(k+1))!\text{card}\left(\text{Stab}_{\mathfrak{S}_{k+1}}(\Theta)\right)} \quad .$$

Pour utiliser les algorithmes de racine r-ième sur des polynômes tronqués au degré s, nous allons donc effectuer les calculs de (3.23) modulo t^{s+1} sur les polynômes réciproques des résolvantes apparaissant dans (3.23) qui sont des polynômes à coefficients dans $A[y]$.

La puissance à laquelle la résolvante $\mathcal{L}_{\Theta,f}$ est obtenue dans la formule (3.23) est $r = \operatorname{card}\left(\operatorname{Stab}_{\mathfrak{S}_{k+1}}(\Theta)\right)/d$.

Théorème 3.9. *Avec la notation 2.44, la résolvante de Lagrange absolue de l'invariant Θ se calcule récursivement par la formule :*

$$\operatorname{Récip}\left(\mathcal{L}_{\Theta,f}\right) = \sqrt[r]{\operatorname{Rés}_y\left(f(y), \left\lceil \frac{\widetilde{N}(t,y)}{\widetilde{D}(t,y)} \right\rceil^t\right) \bmod t^{s+1}} \tag{3.24}$$

où $\left\lceil \frac{\widetilde{N}(t,y)}{\widetilde{D}(t,y)} \right\rceil^t$ désigne le j-ième quotient (de degré j) de la division des polynômes par rapport à la variable t selon les puissances croissantes où $j = \deg_t\left(N(t,y)\right) - \deg_t\left(D(t,y)\right)$; et, si k est impair,

$$\widetilde{N}(t,y) = \widetilde{L}(0,k)\widetilde{L}(2,k)\cdots\widetilde{L}(k-1,k) \bmod t^{j+1}$$
$$\text{et} \qquad \widetilde{D}(t,y) = \widetilde{L}(1,k)\widetilde{L}(3,k)\cdots\widetilde{L}(k,k) \bmod t^{j+1}$$

sinon, si k est pair,

$$\widetilde{N}(t,y) = \widetilde{L}(0,k)\widetilde{L}(2,k)\cdots\widetilde{L}(k,k) \bmod t^{j+1}$$
$$\text{et} \qquad \widetilde{D}(t,y) = \widetilde{L}(1,k)\widetilde{L}(3,k)\cdots\widetilde{L}(k-1,k) \bmod t^{j+1}$$

avec

$$\widetilde{L}(i,k) = \prod_{\overline{m}\in\overline{M}(i,k)} \operatorname{Récip}\left(\left(\mathcal{L}_{[\overline{m}.\Theta],f}\right)^{i!\operatorname{card}\left(\operatorname{Stab}_{\mathfrak{S}_{k-i}}([\overline{m}.\Theta])\right)d_{\overline{m}}/d}\right) \quad ,$$

$\overline{M}(i,k)$ pour $i \le k$, $d_{\overline{m}}$ pour $\overline{m} \in \overline{M}(i,k)$ et d ayant été définis au paragraphe 3.4 et r et s au début du paragraphe 3.5.

Preuve : La possibilité d'effectuer des calculs de résultants sur les polynômes réciproques vient du lemme 2.25.

La possibilité de calculer la racine r-ième du polynôme modulo t^{s+1} vient du lemme 2.42.

D'après la propriété 2.18, le j-ième quotient dans la division des polynômes réciproques $\widetilde{N}(t,y)$ par $\widetilde{D}(t,y)$ selon les puissances croissantes est bien égal au polynôme réciproque du quotient des polynômes $N(t,y)$ par $D(t,y)$ dans la division euclidienne et, d'après la propriété 2.17, ne sont nécessaires pour le calculer que les $j+1$ premiers termes (de plus bas degré) des polynômes $\widetilde{N}(t,y)$ et $\widetilde{D}(t,y)$.

La validité des formules sur les polynômes a été établie au cours des paragraphes 3.3 et 3.4. □

Ce théorème fournit bien un algorithme récursif de calcul de résolvantes de Lagrange dans le cas général : les résolvantes $\operatorname{Récip}\left(\mathcal{L}_{[\overline{m}.\Theta],f}\right)$ pour des invariants, d'arité strictement inférieure à celle de Θ, qui apparaissent dans la formule (3.24) sont elles-mêmes calculées par induction du théorème sur l'anneau intègre $A[y]$.

Des exemples d'application de ce théorème sont donnés au paragraphe 3.6.

3.6 Les invariants simples

L'utilisation du théorème 3.9 pour un invariant quelconque Θ d'arité $k+1$ nécessite, lors des appels récursifs du théorème 3.9 de calculer des résolvantes sur l'anneau $A[y]$. Les appels récursifs suivants, pour les calculs sur $A[y]$ nécessiteront l'adjonction d'une nouvelle variable transcendante, notons-la z, les calculs étant menés pour des résolvantes à coefficients dans $A[y,z]$ et ainsi de suite. Le nombre des termes des polynômes à manipuler peut se révéler très conséquent en raison du nombre de variables transcendantes apparaissant, et ôter tout intérêt à cette méthode de calcul des résolvantes par rapport à une méthode basée, par exemple, sur les fonctions symétriques, qui peut nécessiter elle aussi beaucoup de manipulations algébriques coûteuses.

Cependant, l'utilisation du théorème sur un anneau intègre plus « compliqué » (du point de vue de sa représentation en machine) en raison de l'apparition de nouvelles variables transcendantes n'est pas toujours nécessaire. Il est parfois possible de calculer les résolvantes sur $A[y]$ à partir d'une transformation simple de résolvantes à coefficients dans A :

1. si $[\overline{m}.\Theta] \in A[y, x_1, \ldots, x_k]$ correspond à la somme d'un polynôme en y, noté $S(y)$, et d'un invariant $\Psi \in A[x_1, \ldots, x_k]$,

$$[\overline{m}.\Theta] = \Psi + S(y) \tag{3.25}$$

la résolvante

$$\mathcal{L}_{[\overline{m}.\Theta],f}(t) = \mathcal{L}_{\Psi+S(y),f}(t)$$

s'obtient par

$$\mathcal{L}_{[\overline{m}.\Theta],f}(t) = \mathcal{L}_{\Psi,f}(t - S(y)) \tag{3.26}$$

qui devient pour le polynôme réciproque :

$$\text{Récip}\left(\mathcal{L}_{[\overline{m}.\Theta],f}\right)(t) = (1 - S(y)t)^{\deg_t\left(\mathcal{L}_{\Psi,f}\right)} \times \text{Récip}\left(\mathcal{L}_{\Psi,f}\right)\left(\frac{t}{1 - S(y)t}\right) \quad ; \tag{3.27}$$

2. si $[\overline{m}.\Theta] \in A[y, x_1, \ldots, x_k]$ correspond au produit d'un polynôme en y, noté $P(y)$, et d'un invariant $\Psi \in A[x_1, \ldots, x_k]$,

$$[\overline{m}.\Theta] = \Psi \times P(y) \tag{3.28}$$

la résolvante

$$\mathcal{L}_{[\overline{m}.\Theta],f}(t) = \mathcal{L}_{\Psi P(y),f}(t)$$

s'obtient par

$$\mathcal{L}_{[\overline{m}.\Theta],f}(t) = P(y)^{\deg_t\left(\mathcal{L}_{\Psi,f}\right)} \times \mathcal{L}_{\Psi,f}\left(\frac{t}{P(y)}\right) \tag{3.29}$$

qui devient pour le polynôme réciproque :

$$\text{Récip}\left(\mathcal{L}_{[\overline{m}.\Theta],f}\right)(t) = \text{Récip}\left(\mathcal{L}_{\Psi,f}\right)(tP(y)) \quad . \tag{3.30}$$

Exemples : Nous supposons que le polynôme f est de degré n.

La résolvante associée à l'invariant $\Theta = y + x_1$, d'arité 1, se calcule par :

$$\mathcal{L}_{y+x_1,f}(t) = \mathcal{L}_{x_1,f}(t-y) \quad .$$

Ce calcul devient en passant aux polynômes réciproques :

$$\text{Récip}\left(\mathcal{L}_{y+x_1,f}\right)(t) = (1-yt)^n \times \text{Récip}\left(\mathcal{L}_{x_1,f}\right)\left(\frac{t}{1-yt}\right) \quad .$$

La résolvante associée à l'invariant $\Theta = (y+1)x_1x_2$, d'arité 2, se calcule par :

$$\mathcal{L}_{(y+1)x_1x_2,f}(t) = (y+1)^{n(n-1)/2} \times \mathcal{L}_{x_1x_2,f}\left(\frac{t}{y+1}\right) \quad .$$

Ce calcul devient en passant aux polynômes réciproques :

$$\text{Récip}\left(\mathcal{L}_{(y+1)x_1x_2,f}\right)(t) = \text{Récip}\left(\mathcal{L}_{x_1x_2,f}\right)(t(y+1)) \quad .$$

La résolvante associée à l'invariant $\Theta = yx_1$, d'arité 1, se calcule par :

$$\mathcal{L}_{yx_1,f}(t) = y^n \times \mathcal{L}_{x_1,f}\left(\frac{t}{y}\right) \quad .$$

Ce calcul devient en passant aux polynômes réciproques :

$$\text{Récip}\left(\mathcal{L}_{yx_1,f}\right)(t) = \text{Récip}\left(\mathcal{L}_{x_1,f}\right)(ty) \quad .$$

La résolvante associée à l'invariant $\Theta = y + yx_1$, d'arité 1, se calcule par :

$$\begin{aligned}\mathcal{L}_{y+yx_1,f}(t) &= \mathcal{L}_{yx_1,f}(t-y) \\ &= y^n \times \mathcal{L}_{x_1,f}\left(\frac{t-y}{y}\right) \quad .\end{aligned}$$

Ce calcul devient en passant aux polynômes réciproques :

$$\begin{aligned}\text{Récip}\left(\mathcal{L}_{y+yx_1,f}\right)(t) &= (1-ty)^n \times \text{Récip}\left(\mathcal{L}_{yx_1,f}\right)\left(\frac{t}{1-ty}\right) \\ &= (1-ty)^n \times \text{Récip}\left(\mathcal{L}_{x_1,f}\right)\left(\frac{ty}{1-ty}\right) \quad .\end{aligned}$$

Définition 3.10. Un *invariant simple* est un invariant Θ d'arité $k+1$, $\Theta \in A[x_1, \ldots, x_{k+1}]$, dont tous les évalués distincts $[\overline{m}.\Theta]$ pour $\overline{m} \in \overline{M}(i,k)$, avec $0 \leq i \leq k$, ($\overline{M}(i,k)$ ayant été défini au paragraphe 3.4) s'obtiennent par l'une ou l'autre des deux opérations (3.25) et (3.28) (somme d'un polynôme en y, produit par un polynôme en y) ou par combinaisons de ces deux opérations.

Exemples : Les invariants $x_1 + x_2$, $(x_1 + 1)x_2x_3 = x_1x_2x_3 + x_2x_3$, x_1x_2, $x_1 + x_2x_3$ sont simples.

Pour les invariants simples, les appels récursifs au théorème 3.9 sont effectués pour des invariants qui sont, par définition, de la forme $[\overline{m}.\Theta] = C(\Psi(x_1, \ldots, x_{k-i}), y)$ où $C(\Psi, y)$ désigne le polynôme en y construit à partir de sommes et de multiplications successives de Ψ avec des polynômes en y. Il suffit pour calculer la résolvante de Lagrange associée à un tel $[\overline{m}.\Theta]$ de calculer d'abord la résolvante de Lagrange, $\mathcal{L}_{\Psi,f}$, **à coefficients dans** A de Ψ puis d'en déduire la résolvante de Lagrange de $[\overline{m}.\Theta]$, $\mathcal{L}_{[\overline{m}.\Theta],f}$, en appliquant la combinaison de transformations adéquates correspondant à celles utilisées pour le calcul de C, c'est-à-dire (3.26) pour chaque utilisation de (3.25) et (3.29) pour chaque utilisation de (3.28).

Remarque : Les invariants linéaires étudiés par L. E. SOICHER sont des invariants simples.

Nous allons illustrer cette utilisation particulière du théorème 3.9 par quelques exemples.

Invariant $\Theta = x_1$

$$\text{Récip}\left(\mathcal{L}_{x_1,f}\right)(t) = \text{Récip}\left(f\right)(t) \quad ;$$

Invariant $\Theta = x_1 + x_2$

$$\text{Récip}\left(\mathcal{L}_{x_1+x_2,f}\right)(t) = \sqrt[2]{\text{Rés}_y\left(f(y), \left\lceil \frac{\widetilde{N}(t,y)}{\widetilde{D}(t,y)} \right\rceil^t\right) \bmod t^{s+1}}$$

avec le degré s de cette résolvante qui est égal à $s = n(n-1)/2$ et j la différence des degrés en t des numérateur $N(t,y)$ et dénominateur $D(t,y)$ qui est égale à $j = n - 1$.

$$\widetilde{N}(t,y) = (1-yt)^n \times \text{Récip}\left(\mathcal{L}_{x_1,f}\right)\left(\frac{t}{1-yt}\right) \bmod t^{j+1}$$

$$\text{et} \qquad \widetilde{D}(t,y) = 1 - 2yt \quad ;$$

Invariant $\Theta = x_1 + x_2 + x_3$

$$\text{Récip}\left(\mathcal{L}_{x_1+x_2+x_3,f}\right)(t) = \sqrt[3]{\text{Rés}_y\left(f(y), \left\lceil \frac{\widetilde{N}(t,y)}{\widetilde{D}(t,y)} \right\rceil^t\right) \bmod t^{s+1}}$$

avec le degré s de cette résolvante qui est égal à $s = n(n-1)(n-2)/3!$ et j la différence des degrés en t des numérateur $N(t,y)$ et dénominateur $D(t,y)$ qui est égale à $j = (n-1)(n-2)/2$.

$$\widetilde{N}(t,y) = (1-yt)^{n(n-1)/2} \times \text{Récip}\left(\mathcal{L}_{x_1+x_2,f}\right)\left(\frac{t}{1-yt}\right) \times (1-3yt) \bmod t^{j+1}$$

$$\text{et} \qquad \widetilde{D}(t,y) = (1-2yt)^{n} \times \text{Récip}\left(\mathcal{L}_{x_1,f}\right)\left(\frac{t}{1-2yt}\right) \quad ;$$

Invariant $\Theta = x_1 + x_2 + x_3 + x_4$

$$\text{Récip}\left(\mathcal{L}_{x_1+x_2+x_3+x_4,f}\right)(t) = \sqrt[4]{\text{Rés}_y\left(f(y), \left\lceil \frac{\widetilde{N}(t,y)}{\widetilde{D}(t,y)} \right\rceil^t\right)} \bmod t^{s+1}$$

avec le degré s de cette résolvante qui est égal à

$$s = n(n-1)(n-2)(n-3)/4!$$

et j la différence des degrés en t des numérateur $N(t,y)$ et dénominateur $D(t,y)$ qui est égale à $j = (n-1)(n-2)(n-3)/6$.

$$\widetilde{N}(t,y) = (1-yt)^{n(n-1)(n-2)/2} \times \text{Récip}\left(\mathcal{L}_{x_1+x_2+x_3,f}\right)\left(\frac{t}{1-yt}\right)$$
$$\times (1-3yt)^{n} \times \text{Récip}\left(\mathcal{L}_{x_1,f}\right)\left(\frac{t}{1-3yt}\right) \bmod t^{j+1}$$

$$\text{et} \quad \widetilde{D}(t,y) = (1-2yt)^{n(n-1)/2} \times \text{Récip}\left(\mathcal{L}_{x_1+x_2,f}\right)\left(\frac{t}{1-2yt}\right) \times (1-4yt) \quad ;$$

Invariant $\Theta = x_1 x_2$

$$\text{Récip}\left(\mathcal{L}_{x_1x_2,f}\right)(t) = \sqrt[2]{\text{Rés}_y\left(f(y), \left\lceil \frac{\widetilde{N}(t,y)}{\widetilde{D}(t,y)} \right\rceil^t\right)} \bmod t^{s+1}$$

avec le degré s de cette résolvante qui est égal à $s = n(n-1)/2$ et j la différence des degrés en t des numérateur $N(t,y)$ et dénominateur $D(t,y)$ qui est égale à $j = n-1$.

$$\widetilde{N}(t,y) = \text{Récip}\left(\mathcal{L}_{x_1,f}\right)(yt) \bmod t^{j+1}$$

$$\text{et} \qquad \widetilde{D}(t,y) = 1 - y^2 t \quad ;$$

Invariant $\Theta = x_1x_2x_3$

$$\text{Récip}\left(\mathcal{L}_{x_1x_2x_3,f}\right)(t) = \sqrt[3]{\text{Rés}_y\left(f(y), \left\lceil \frac{\widetilde{N}(t,y)}{\widetilde{D}(t,y)} \right\rceil^t\right) \bmod t^{s+1}}$$

avec le degré s de cette résolvante qui est égal à $s = n(n-1)(n-2)/3!$ et j la différence des degrés en t des numérateur $N(t,y)$ et dénominateur $D(t,y)$ qui est égale à $j = (n-1)(n-2)/2$.

$$\begin{aligned} & \widetilde{N}(t,y) = \text{Récip}\left(\mathcal{L}_{x_1x_2,f}\right)(yt) \times (1-y^3t) \bmod t^{j+1} \\ \text{et} \quad & \widetilde{D}(t,y) = \text{Récip}\left(\mathcal{L}_{x_1,f}\right)(y^2t) \quad ; \end{aligned}$$

Invariant $\Theta = x_1 + x_2x_3$

$$\text{Récip}\left(\mathcal{L}_{x_1+x_2x_3,f}\right)(t) = \sqrt[2]{\text{Rés}_y\left(f(y), \left\lceil \frac{\widetilde{N}(t,y)}{\widetilde{D}(t,y)} \right\rceil^t\right) \bmod t^{s+1}}$$

avec le degré s de cette résolvante qui est égal à $s = n(n-1)(n-2)/2$ et j la différence des degrés en t des numérateur $N(t,y)$ et dénominateur $D(t,y)$ qui est égale à $j = (n-1)(n-2)/2$.

$$\begin{aligned} & \widetilde{N}(t,y) = (1-yt)^{n(n-1)/2} \times \text{Récip}\left(\mathcal{L}_{x_1x_2,f}\right)\left(\frac{t}{1-yt}\right) \times (1-(y+y^2)t) \bmod t^{j+1} \\ \text{et} \quad & \widetilde{D}(t,y) = (1-yt)^n \times \text{Récip}\left(\mathcal{L}_{x_1,f}\right)\left(\frac{yt}{1-yt}\right) \quad . \end{aligned}$$

Remarque : L'invariant $x_1x_2+x_3$ n'est pas simple à cause de l'évalué $[x_1x_2+x_3] = yx_1+x_2$ qui n'est ni le produit ni la somme d'un invariant avec le polynôme y.

L'invariant $\Theta = x_1x_2 + x_3x_4$ n'est pas non plus simple, à cause du terme $yx_1 + x_2x_3$ qui ne peut s'obtenir à partir d'un autre invariant par la composition de multiplications et d'additions de polynômes en y. Nous l'utilisons comme exemple d'un calcul dans la version générale de l'algorithme récursif. Il faut pour cet invariant faire des calculs sur des polynômes à coefficients dans l'anneau $A[y]$.

Invariant $yx_1 + x_2x_3$

Nous calculons d'abord $\mathcal{L}_{yx_1+x_2x_3,f}$ sur l'anneau $A[y]$. La variable z joue ici le rôle de y précédemment :

$$\text{Récip}\left(\mathcal{L}_{yx_1+x_2x_3,f}\right)(t) = \sqrt[2]{\text{Rés}_z\left(f(z), \left\lceil \frac{\widetilde{N}(t,z)}{\widetilde{D}(t,z)} \right\rceil^t\right) \bmod t^{s+1}} \quad .$$

Le degré s de cette résolvante est égal à $s = n(n-1)(n-2)/2$ et j, la différence des degrés en t des numérateur $N(t,y)$ et dénominateur $D(t,y)$, est égale à $j = (n-1)(n-2)/2$.

$$\widetilde{N}(t,z) = (1-yzt)^{n(n-1)/2} \times \text{Récip}\left(\mathcal{L}_{x_1x_2,f}\right)(1-yzt) \times (1-(yz+z^2)t) \bmod t^{j+1}$$

$$\text{et} \qquad \widetilde{D}(t,z) = (1-yzt)^n \times \text{Récip}\left(\mathcal{L}_{x_1,f}\right)\left(\frac{zt}{1-yzt}\right) \quad .$$

Invariant $x_1x_2 + x_3x_4$

Nous calculons ensuite $\mathcal{L}_{x_1x_2+x_3x_4,f}$ sur l'anneau A par :

$$\text{Récip}\left(\mathcal{L}_{x_1x_2+x_3x_4,f}\right)(t) = \sqrt[2]{\text{Rés}_y\left(f(y), \left\lceil \frac{\widetilde{N}(t,y)}{\widetilde{D}(t,y)} \right\rceil^t\right) \bmod t^{s+1}} \quad .$$

Le degré s de cette résolvante est égal à

$$s = n(n-1)(n-2)(n-3)/4$$

et j, la différence des degrés en t des numérateur $N(t,y)$ et dénominateur $D(t,y)$, est égale à $j = (n-1)(n-2)(n-3)/2$.

$$\widetilde{N}(t,y) = \text{Récip}\left(\mathcal{L}_{yx_1+x_2x_3,f}\right)(t) \times \left((1-y^2t)^n \times \text{Récip}\left(\mathcal{L}_{x_1,f}\right)\left(\frac{t}{1-y^2t}\right)\right)^3 \bmod t^{j+1}$$

$$\text{et} \qquad \widetilde{D}(t,z) = (1-y^2t)^{n(n-1)/2} \times \text{Récip}\left(\mathcal{L}_{x_1x_2,f}\right)\left(\frac{t}{1-y^2t}\right) \times \left(\text{Récip}\left(\mathcal{L}_{x_1+x_2,f}\right)(yt)\right)^2$$
$$\times (1-2y^2t)^3 \quad .$$

3.7 Conclusion

Nous avons proposé dans ce chapitre une généralisation de la méthode de Lagrange-Soicher à des invariants **quelconques**. Nous avons fourni pour cela une description combinatoire des polynômes superflus apparaissant lors de l'élimination des variables grâce au résultant. Nous avons apporté deux raffinements à cette méthode : une généralisation de la remarque (voir la proposition 3.3) de C. J. WILLIAMSON déjà proposée dans [61] ainsi que l'utilisation de techniques de calculs de racines r-ièmes permettant de ne calculer (sur les polynômes réciproques) que les termes strictement nécessaires au calcul de cette racine. Cela ne représente bien sûr une amélioration que dans le cas où les puissances superflues sont strictement plus grandes que 1, mais c'est fréquemment le cas des invariants primitifs utilisés lors de la recherche du groupe de Galois : ils correspondent à des sous-groupes de $\mathfrak{S}_n$.

Cependant, en raison du grand nombre d'appels récursifs, dans le cas d'invariants quelconques, sur des polynômes à coefficients dans des anneaux de polynômes sur A avec un nombre croissant de variables transcendantes, l'efficacité de cette méthode semble limitée aux invariants de très petite arité (disons jusqu'à 5). Nous avons toutefois caractérisé une

sous-classe d'invariants, les invariants simples, pour lesquels les calculs récursifs demeurent dans l'anneau A (sans introduire de nouvelles variables transcendantes) et sont alors beaucoup plus efficaces. Mais même dans ce cas, le nombre de polynômes superflus à calculer reste conséquent.

Nous décrivons dans le chapitre 4 une adaptation des idées de ce chapitre (les calculs tronqués au seul degré nécessaire pour le calcul de racine r-ième) à la méthode de calcul de A. VALIBOUZE et N. RENNERT dans laquelle les polynômes superflus n'apparaissent plus. Les méthodes peuvent être en quelque sorte considérées comme duales : informellement parlé, dans ce chapitre nous éliminons les polynômes « en trop » (en les calculant) alors que dans le chapitre 4, la méthode de A. VALIBOUZE et N. RENNERT enlève *a priori* les « racines en trop », ce qui évite l'apparition des polynômes superflus.

Il existe pourtant une autre différence, entre les deux méthodes qui pourtant calculent les mêmes objets. La méthode décrite dans ce chapitre peut être facilement et efficacement adaptée aux calculs modulaires dans le but de calculer la résolvante par le théorème chinois.

Plus précisément, si le polynôme f, irréductible dans la plupart des utilisations de cette méthode, peut être factorisé en *un grand nombre de facteurs* modulo un premier p,

$$f \equiv f_1 \cdots f_\ell \bmod p$$

alors les calculs de résultants de la forme

$$\text{Rés}_y\left(f(y), Q(t,y)\right)$$

peuvent être obtenus modulo p en utilisant la propriété 2.22 du résultant :

$$\text{Rés}_y\left(f(y), Q(t,y)\right) \equiv \prod_{i=1}^{\ell} \text{Rés}_y\left(f_i(y), Q(t,y)\right) \bmod p \quad . \tag{3.31}$$

Or les calculs de résultants modulaires, modulo des facteurs de plus petit degré que f, pour i variant de 1 à ℓ sont beaucoup plus faciles. Nous avons vu par ailleurs au paragraphe 2.5 à quelles conditions les algorithmes de calcul de racine r-ième pouvaient être utilisés en caractéristique non nulle. Il est donc possible de calculer les images d'une résolvante cherchée modulo un nombre suffisant d'entiers premiers (qui dépend d'une borne sur les coefficients de la résolvante cherchée). Ces premiers doivent convenir pour les calculs modulaires du théorème 3.9 utilisé en caractéristique non nulle (voir notation 2.44) et pour l'efficacité (avec grand nombre de facteurs de petit degré pour utiliser l'équation (3.31)). Les images de la résolvantes sont ensuite remontées sur $\mathbb{Z}$ par le théorème chinois. Des idées de cette nature ont été proposées par M. J. ENCARNACIÓN dans [43] pour des normes, qui ne sont en fait que des cas particuliers de résolvantes (linéaires d'arité 2, voir aussi [93] et [113]).

Une adaptation de la méthode de Valibouze-Rennert aux calculs modulaires a été proposée par N. RENNERT.

Chapitre 4

Calcul de résolvantes par les modules de Cauchy

Nous avons vu au chapitre 3 une généralisation de la méthode de Lagrange-Soicher pour calculer les résolvantes. Parallèlement, A. VALIBOUZE, dans [98] puis avec N. RENNERT dans [85], a proposé une autre méthode permettant d'éviter l'apparition des facteurs parasites. Il n'y reste plus que les puissances parasites. Nous appliquons à cette méthode les techniques du chapitre 3. Elles permettent de supprimer les multiplicités parasites *en cours de calcul* et d'améliorer ainsi l'efficacité de la méthode de Valibouze-Rennert.

Ce chapitre reprend les résultats exposés dans [62].

4.1 Modules de Cauchy

Soient $f \in A[x]$ un polynôme de degré n et x_i et y_i pour $1 \leq i \leq n$ des variables transcendantes sur $A[x]$.

Définition 4.1. Les *n fonctions interpolaires d'Ampère* de f (voir [98]) sont définies par :

$$f_n(x) = f(x) \quad ;$$
$$f_i(x) = f_i(x, y_{i+1}, \ldots, y_n) = \frac{f_{i+1}(x) - f_{i+1}(y_{i+1})}{x - y_{i+1}} \quad \text{pour } i \text{ passant de } n-1 \text{ à } 1.$$

Propriété 4.2. *Ces fonctions vérifient, pour $1 \leq i \leq n$, :*

$$f_i \in A[y_{i+1}, \ldots, y_n][x] \qquad et \qquad \deg_x(f_i) = i \quad .$$

Définition 4.3. Les *modules de Cauchy de f évalués en $(x_1, \ldots, x_n)$* sont les fonctions interpolaires d'Ampère f_i, pour $1 \leq i \leq n$, évaluées en $(x_i, \ldots, x_n)$:

$$f_i = f_i(x_i, \ldots, x_n) \in A[x_i, \ldots, x_n] \quad .$$

4.2 Résultat principal

Nous utilisons les notations du chapitre 2.

Théorème 4.4. *Soient A un anneau intègre de caractéristique nulle, $f \in A[x]$ un polynôme unitaire, de degré n, H_n un sous-groupe du groupe symétrique $\mathfrak{S}_n$, $\Theta \in A[x_1, \ldots, x_n]$ un H_n-invariant primitif; f_i, pour $1 \leq i \leq n$, les n modules de Cauchy de f évalués en $(x_1, \ldots, x_n)$ et $H_i = \mathrm{Stab}_{\mathfrak{S}_i}\Theta$, pour $1 \leq i \leq n-1$, les $n-1$ stabilisateurs sur $\mathfrak{S}_i$ de l'invariant Θ considéré comme un élément de $A[x_{i+1}, \ldots, x_n][x_1, \ldots, x_i]$.*

Soient les entiers d_i et m_i définis par:

$$d_i = \frac{i!}{\mathrm{card}\,(H_i)} \qquad et \qquad m_i = \frac{\mathrm{card}\,(H_i)}{\mathrm{card}\,(H_{i-1})} \quad pour\ 1 \leq i \leq n \quad ,$$

en posant $\mathrm{card}\,(H_0) = 1$.

Alors la suite finie $\{\mathcal{R}_0, \ldots, \mathcal{R}_n\}$, avec $\mathcal{R}_i \in A[x_{i+1}, \ldots, x_n][t]/(t^{d_i+1})$, définie récursivement par:

$$\mathcal{R}_0(t) = \mathrm{Récip}\,(t - \Theta) = 1 - t\Theta \quad ;$$
$$\mathcal{R}_i(t) = \sqrt[m_i]{\mathrm{Rés}_{x_i}\,(f_i, \mathcal{R}_{i-1}(t)) \bmod t^{d_i+1}} \quad pour \quad 1 \leq i \leq n \quad ;$$

calcule le polynôme réciproque de la résolvante (absolue) de f par Θ:

$$\mathcal{R}_n(t) = \mathrm{Récip}\,(\mathcal{L}_{\Theta,f})\,(t) \quad .$$

La preuve de ce théorème est donnée dans le paragraphe 4.4. Le paragraphe 4.3 rappelle un résultat clef pour la preuve du théorème 4.4. Le paragraphe 4.5 consiste en des remarques diverses sur le théorème 4.4 (polynômes non unitaires, invariants de petite arité, caractéristique non nulle) et le paragraphe 4.6 traite de la généralisation du théorème 4.4 aux multi-résolvantes.

4.3 Calcul du polynôme f-caractéristique

Le théorème suivant est une forme plus générale du théorème de [98, théorème 5.1]. Il peut aussi être vu comme un cas particulier de ce théorème (en prenant $T = 0$ dans [98, théorème 5.1]).

Théorème 4.5. *Soient A un anneau intègre; $f \in A[x]$ un polynôme unitaire, de degré n et de racines $\alpha_1, \ldots, \alpha_n$ dans une extension algébrique K du corps des fractions de A; f_i, pour $1 \leq i \leq n$, les n modules de Cauchy de f évalués en $(x_1, \ldots, x_n)$ et $\Phi \in A[x_1, \ldots, x_n]$.*

La suite finie $\{r_0, \ldots, r_n\}$ définie récursivement par:

$$r_0 = \Phi \in A[x_1, \ldots, x_n] \quad ;$$
$$r_i = \mathrm{Rés}_{x_i}\,(f_i, r_{i-1}) \in A[x_{i+1}, \ldots, x_n] \quad pour \quad 1 \leq i \leq n \quad ;$$

calcule

$$\prod_{\sigma \in \mathfrak{S}_n} \Phi\left(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)}\right) = r_n \in A \quad .$$

Preuve : Par récurrence sur $n = \deg f$.

Pour $n = 1$:

$$f_1(x) = x - \alpha_1 \quad ,$$

$$\text{Rés}_{x_1}(f_1, \Phi(x_1)) = \Phi(\alpha_1) \quad .$$

Le théorème étant supposé vrai jusqu'au degré $n-1$ pour tout anneau intègre ; soient f de degré n et $f_n = f, f_{n-1}, \ldots, f_1$ ses n modules de Cauchy évalués en $(x_1, \ldots, x_n)$.

L'hypothèse de récurrence peut être appliquée à f_{n-1}, polynôme unitaire à coefficients dans l'anneau intègre $A[x_n]$, dont les racines dans une extension algébrique du corps des fractions de $A[x_n]$ sont notées $\beta_1^{[x_n]}, \ldots, \beta_{n-1}^{[x_n]}$, et dont les $n-1$ modules de Cauchy évalués en $(x_1, \ldots, x_{n-1})$ sont justement $f_{n-1}, \ldots, f_1$:

$$r_{n-1} = \prod_{\sigma \in \mathfrak{S}_{n-1}} \Phi\left(\beta_{\sigma(1)}^{[x_n]}, \ldots, \beta_{\sigma(n-1)}^{[x_n]}, x_n\right) \quad ,$$

d'où

$$\begin{aligned} r_n &= \text{Rés}_{x_n}\left(f_n(x_n), \prod_{\sigma \in \mathfrak{S}_{n-1}} \Phi\left(\beta_{\sigma(1)}^{[x_n]}, \ldots, \beta_{\sigma(n-1)}^{[x_n]}, x_n\right)\right) \\ &= \prod_{i=1}^{n} \prod_{\sigma \in \mathfrak{S}_{n-1}} \Phi\left(\beta_{\sigma(1)}^{[\alpha_i]}, \ldots, \beta_{\sigma(n-1)}^{[\alpha_i]}, \alpha_i\right) \quad . \end{aligned}$$

Or, pour $1 \leq i \leq n$, $\beta_{\sigma(j)}^{[\alpha_i]}$, pour $1 \leq j \leq n-1$, sont les racines de $f_{n-1}^{[\alpha_i]}(x) = \frac{f_n(x) - f_n(\alpha_i)}{x - \alpha_i}$. Elles sont donc les racines de $f(x) = f_n(x)$ différentes de α_i. Les $n!$ n-uplets $\left(\beta_{\sigma(1)}^{[\alpha_i]}, \ldots, \beta_{\sigma(n-1)}^{[\alpha_i]}, \alpha_i\right)$ sont donc en fait les $n!$ n-uplets $\left(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)}\right)$, où σ parcourt $\mathfrak{S}_n$. □

En particulier (voir [98] ou [85]) le théorème 4.5 appliqué à l'invariant $\Theta \in A[x_1, \ldots, x_n]$ et à l'anneau $A[t]$, avec $r_0 = t - \Theta \in A[t][x_1, \ldots, x_n]$, calcule le polynôme f-caractéristique de Θ défini au paragraphe 2.1 :

Corollaire 4.6. *Soient A un anneau intègre, $f \in A[x]$ un polynôme unitaire de degré n ; f_i, pour $1 \leq i \leq n$, ses n modules de Cauchy évalués en $(x_1, \ldots, x_n)$ et un invariant $\Theta \in A[x_1, \ldots, x_n]$.*

La suite finie $\{r_0, \ldots, r_n\}$ définie récursivement par:

$$\begin{aligned} r_0(t) &= t - \Theta \quad ; \\ r_i(t) &= \text{Rés}_{x_i}(f_i, r_{i-1}(t)) \quad \text{pour} \quad 1 \leq i \leq n \quad ; \end{aligned}$$

calcule le polynôme f-caractéristique de Θ :

$$\mathcal{X}_{\Theta,f}^{\mathfrak{S}_n}(t) = r_n(t) \quad .$$

4.4 Preuve du théorème 4.4

Le corollaire 4.6 appliqué à l'anneau intègre $A[x_{i+1},\dots,x_n]$ et à $f_i \in A[x_{i+1},\dots,x_n][x]$, montre que :

$$r_i(t) = \mathcal{X}^{\mathfrak{S}_i}_{\Theta,f_i}(t) \quad .$$

La relation de récurrence sur les r_i peut donc s'écrire :

$$\mathcal{X}^{\mathfrak{S}_i}_{\Theta,f_i}(t) = \text{Rés}_{x_i}\left(f_i, \mathcal{X}^{\mathfrak{S}_{i-1}}_{\Theta,f_{i-1}}(t)\right) \quad . \tag{4.1}$$

Or d'après la proposition 2.8 (pour $L = \mathfrak{S}_i$)

$$\mathcal{X}^{\mathfrak{S}_i}_{\Theta,f_i} = \left(\mathcal{L}^{\mathfrak{S}_i}_{\Theta,f_i}\right)^{\text{card}(H_i)} \quad .$$

L'équation (4.1) devient donc :

$$\left(\mathcal{L}^{\mathfrak{S}_i}_{\Theta,f_i}(t)\right)^{\text{card}(H_i)} = \text{Rés}_{x_i}\left(f_i, \left(\mathcal{L}^{\mathfrak{S}_{i-1}}_{\Theta,f_{i-1}}(t)\right)^{\text{card}(H_{i-1})}\right) \quad . \tag{4.2}$$

En utilisant la propriété 2.23 du résultant, l'équation (4.2) devient :

$$\mathcal{L}^{\mathfrak{S}_i}_{\Theta,f_i}(t) = \sqrt[m_i]{\text{Rés}_{x_i}\left(f_i, \mathcal{L}^{\mathfrak{S}_{i-1}}_{\Theta,f_{i-1}}(t)\right)} \quad ,$$

avec $m_i = \frac{\text{card}(H_i)}{\text{card}(H_{i-1})}$.

Le degré de $\mathcal{L}^{\mathfrak{S}_i}_{\Theta,f_i}$ est :

$$d_i = \text{card}\left(\mathfrak{S}_i.\Theta\right) = \frac{\text{card}\left(\mathfrak{S}_i\right)}{\text{card}\left(H_i\right)} = \frac{i!}{\text{card}\left(H_i\right)} \quad ,$$

d'où, en posant

$$\mathcal{R}_i(x) = \text{Récip}\left(\mathcal{L}^{\mathfrak{S}_i}_{\Theta,f_i}(t)\right) \quad ,$$

la formule de récurrence sur les $\mathcal{R}_i$ du théorème 4.4.

La possibilité d'effectuer les calculs modulo t^{d_i+1} sur les polynômes réciproques par rapport à t découle directement des lemmes 2.24, 2.25 et 2.42, ce qui achève la preuve du théorème 4.4.

4.5 Remarques diverses

Polynômes non unitaires

Soient A un anneau intègre de caractéristique nulle et $\Theta \in A[x_1,\dots,x_n]$.

Si le polynôme $f \in A[x]$ n'est pas unitaire :

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \quad ,$$

il peut toujours être transformé, comme au paragraphe 2.5.1, en un polynôme unitaire $\hat{f}$ par :

$$\hat{f} = a_n^{n-1} f\left(\frac{x}{a_n}\right) \quad . \tag{4.3}$$

Le théorème 4.4 peut être appliqué à $\hat{f}$ fournissant $\mathcal{L}_{\Theta,\hat{f}}$.

Lemme 4.7. *Si le polynôme Θ est homogène de degré $\deg\Theta$, la résolvante $\mathcal{L}_{\Theta,f}$ peut alors être retrouvée par :*

$$\mathcal{L}_{\Theta,f}(t) = \frac{\mathcal{L}_{\Theta,\hat{f}}\left(a_n^{\deg\Theta} t\right)}{a_n^{\deg\Theta \deg \mathcal{L}_{\Theta,\hat{f}}}} \quad .$$

Preuve : Les racines de $\hat{f}$ sont celles de f multipliées par a_n. Si Θ est homogène de degré $\deg\Theta$, les racines de $\mathcal{L}_{\Theta,\hat{f}}$ sont celles de $\mathcal{L}_{\Theta,f}$ multipliées par $a_n^{\deg\Theta}$. D'où le lemme. □

Si Θ n'est pas homogène, ou pour éviter l'utilisation de la formule 4.3 qui entraîne un accroissement des coefficients, il suffit d'introduire dans les calculs les coefficients dominants des modules de Cauchy f_i (pour la variable x_i, tous égaux au coefficient dominant a_n de f) dans les calculs de résultants et de s'en débarrasser par la propriété 2.21 avant de calculer la racine m_i-ième (m_i étant défini dans le théorème 4.4) :

$$\mathcal{L}^{\mathfrak{S}_i}_{\Theta,f_i}(t) = \sqrt[m_i]{\frac{\mathrm{Rés}_{x_i}\left(f_i, \mathcal{L}^{\mathfrak{S}_{i-1}}_{\Theta,f_{i-1}}(t)\right)}{a_n^{\deg_{x_i} \mathcal{L}^{\mathfrak{S}_{i-1}}_{\Theta,f_{i-1}}}}} \quad ,$$

d'où la nouvelle formule de récurrence sur les $\mathcal{R}_i$ (d_i étant défini dans le théorème 4.4) :

$$\mathcal{R}_i(t) = \sqrt[m_i]{\frac{\mathrm{Rés}_{x_i}\left(f_i, \mathcal{R}_{i-1}(t)\right) \bmod t^{d_i+1}}{a_n^{\deg_{x_i}\left(\mathcal{R}_{i-1}(t) \bmod t^{d_i+1}\right)}}} \quad .$$

Prise en compte de l'arité

Soient A un anneau intègre de caractéristique nulle et $\Theta \in A[x_1, \ldots, x_n]$.

Si l'invariant Θ est d'arité k ses variables peuvent être renumérotées de façon à ce que Θ s'écrive comme un élément de $A[x_{n-k+1}, \ldots, x_n]$.

Dans ce cas, en appliquant le théorème 4.4 à Θ, il apparaît que, pour $1 \leq i \leq n-k$:

$$H_i = \mathfrak{S}_i, \qquad d_i = 1, \qquad m_i = i \quad .$$

D'où, pour $0 \leq i \leq n-k$:

$$\mathcal{R}_i(t) = \mathrm{Récip}\,(t - \Theta) = 1 - t\Theta \quad .$$

La résolvante $\mathcal{L}_{\Theta,f}$ peut donc se calculer en n'appliquant les formules récursives du théorème 4.4 que pour i passant de $n-k+1$ à n, $\mathcal{R}_{n-k}(t)$ étant connu et égal à $\mathrm{Récip}\,(t-\Theta)$.

Caractéristique non nulle

L'hypothèse *A de caractéristique nulle* dans le théorème 4.4 n'est nécessaire que pour pouvoir calculer des racines r-ièmes de polynômes alors que les lemmes 2.24 et 2.25 sont vrais sur tout anneau intègre.

En fait le lemme 2.43 peut être utilisé si la caractéristique ℓ de A est strictement supérieure au degré s de la racine et si ℓ ne divise pas r, la puissance à laquelle cette racine est élevée.

Le théorème 4.4 est donc vrai dans un *anneau intègre A de caractéristique ℓ tel que $d_i < \ell$ et ℓ ne divise pas m_i pour $1 \leq i \leq n$.*

Ou bien, au prix d'une moins bonne complexité, avec le lemme 2.39, il est possible de s'affranchir de la contrainte $\ell > s$. Le théorème 4.4 est alors vrai dans un *anneau intègre A de caractéristique ℓ tel que ℓ ne divise pas m_i pour $1 \leq i \leq n$.*

La possibilité d'effectuer des calculs sur des anneaux de caractéristique non nulle permet d'effectuer les calculs modulo plusieurs premiers pour obtenir ainsi diverses images modulaires de la résolvante. Si une borne sur les coefficients de cette résolvante est connue, nous pouvons la reconstruire grâce au théorème chinois. Cette stratégie contribue à prévenir un des problèmes dans les calculs de la méthode exposée : la croissance de la taille des coefficients ; mais elle ne limite pas la croissance des degrés en x_i des polynômes y apparaissant (et donc le nombre de termes).

4.6 Généralisation aux multi-résolvantes

Notation 4.8. Soient $n_1, \ldots, n_p$ des entiers positifs, $\mathfrak{S}_{n_1,\ldots,n_p}$ désigne le *sous-groupe de Young* du groupe $\mathfrak{S}_{\{1,\ldots,n_1+\cdots+n_p\}}$. Il est égal à

$$\mathfrak{S}_{1,\ldots,n_1} \times \cdots \times \mathfrak{S}_{n_1+\cdots+n_{p-1}+1,\ldots,n_1+\cdots+n_{p-1}} \quad .$$

Les multi-résolvantes sont définies dans [45] et [99] ou [85]. Soient A un anneau intègre de caractéristique nulle et $\left(f^{[1]}, \ldots, f^{[p]}\right)$ un p-uplet de polynômes unitaires de $A[x]$ de degrés respectifs $n_1, \ldots, n_p$.

Soient $N_0, \ldots, N_p$ définis par :

$$N_0 = 0 \qquad \text{et} \qquad N_j = \sum_{i=1}^{j} n_i \quad \text{pour} \quad 1 \leq j \leq p. \tag{4.4}$$

Le produit de degré N_p des p polynômes du p-uplet est noté f :

$$f = f^{[1]} \ldots f^{[p]} \quad .$$

Les groupes de Galois sur le corps de fractions K, supposé parfait, de A de $f^{[1]}, \ldots, f^{[p]}$ sont notés $G_1, \ldots, G_p$ ($G_i \subset \mathfrak{S}_{\{N_{i-1}+1,\ldots,N_{i-1}+n_i\}}$ pour $1 \leq i \leq p$) et les racines des $f^{[i]}$, $1 \leq i \leq p$ appartiennent respectivement à l'ensemble ordonné $\Omega_f = \{\Omega_{f^{[1]}}, \ldots, \Omega_{f^{[p]}}\}$ avec $\Omega_{f^{[1]}} = \{\alpha_{N_0+1}, \ldots, \alpha_{N_0+n_1}\}$; $\Omega_{f^{[2]}} = \{\alpha_{N_1+1}, \ldots, \alpha_{N_1+n_2}\}$; $\ldots$; $\Omega_{f^{[p]}} = \{\alpha_{N_{p-1}+1}, \ldots, \alpha_{N_{p-1}+n_p}\}$.

Soient L un sous-groupe de $\mathfrak{S}_{n_1,n_2,\ldots,n_p}$ tel que $G_1 \times G_2 \times \cdots \times G_p$ soit un sous-groupe de L :

$$G_1 \times \cdots \times G_p \subset L \subset \mathfrak{S}_{n_1,\ldots,n_p} \subset \mathfrak{S}_{N_p} \quad ,$$

et H un sous-groupe de L et Θ un H-invariant primitif L-relatif appartenant à $A[x_1, \ldots, x_{N_p}]$.

La *multi-résolvante L-relative de $(f^{[1]}, \ldots, f^{[p]})$ par Θ, pour l'ordre sur les racines Ω_f*, notée $\mathcal{L}^L_{\Theta,\Omega_f}$, est définie par :

$$\mathcal{L}^L_{\Theta,\Omega_f}(t) = \prod_{\Psi \in L.\Theta} \left(t - \Psi\left(\Omega_f\right)\right) \quad . \tag{4.5}$$

$\mathcal{L}^{\mathfrak{S}_{n_1,\ldots,n_p}}_{\Theta,\Omega_f}$ qui ne dépend pas de l'ordre sur les racines Ω_f mais seulement de l'ordre sur les polynômes $f^{[i]}$ (qui correspond à l'ordre sur les paquets de racines $\Omega_{f^{[i]}}$) pour $1 \leq i \leq p$ est appelée *multi-résolvante (absolue) de $(f^{[1]}, \ldots, f^{[p]})$ par l'invariant Θ* et simplement notée $\mathcal{L}_{\Theta,(f^{[1]},\ldots,f^{[p]})}$.

Remarques :

1. La définition des multi-résolvantes coïncide avec celle des résolvantes lorsque $p = 1$.
2. Ces multi-résolvantes sont particulièrement intéressantes lors de la recherche du groupe de Galois d'un polynôme séparable f se factorisant sur $A[x]$ en $f = f^{[1]} f^{[2]} \cdots f^{[p]}$, les $f^{[i]}$ n'étant pas nécessairement irréductibles. Dans ce cas les multi-résolvantes absolues de ses p facteurs $(f^{[1]}, \ldots, f^{[p]})$ correspondent aux résolvantes $\mathfrak{S}_{n_1,\ldots,n_p}$-relatives de f [99] :

$$\mathcal{L}_{\Theta,(f^{[1]},\ldots,f^{[p]})} = \mathcal{L}^{\mathfrak{S}_{n_1,\ldots,n_p}}_{\Theta,f} \quad .$$

Soit L un sous-groupe de $\mathfrak{S}_{n_1,\ldots,n_p}$ et $\Theta \in A[x_1, \ldots, x_{N_p}]$. Le *polynôme Ω_f-caractéristique L-relatif de Θ* est défini par :

$$\mathcal{X}^L_{\Theta,\Omega_f}(t) = \prod_{\sigma \in L} \left(t - \Theta\left(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(N_p)}\right)\right) \quad .$$

$\mathcal{X}^{\mathfrak{S}_{n_1,\ldots,n_p}}_{\Theta,\Omega_f}$ est appelé *polynôme $(f^{[1]}, \ldots, f^{[p]})$-caractéristique de Θ et noté $\mathcal{X}_{\Theta,(f^{[1]},\ldots,f^{[p]})}$* car il ne dépend pas de l'ordre Ω_f sur les racines mais seulement de l'ordre sur les polynômes $f^{[i]}$.

Théorème 4.9. *Soient R un anneau intègre ; $(f^{[1]}, \ldots, f^{[p]})$ un p-uplet de polynômes unitaires de $R[x]$ de degrés respectifs $n_1, \ldots, n_p$; $N_0, \ldots, N_p$ définis par (4.4) ; pour $1 \leq j \leq p$: $f_{N_{j-1}+i}$, pour $1 \leq i \leq n_j$, les n_j modules de Cauchy de $f^{[j]}$ évalués en $(x_{N_{j-1}+i}, \ldots, x_{N_{j-1}+n_j})$:*

$$f_{N_{j-1}+i} = f_i^{[j]}(x_{N_{j-1}+i}, \ldots, x_{N_{j-1}+n_j})$$

et l'invariant $\Theta \in R[x_1, \ldots, x_{N_p}]$.

La suite finie $\{r_0, \ldots, r_{N_p}\}$ définie récursivement par:

$$\begin{aligned} r_0(t) &= t - \Theta \quad ; \\ r_i(t) &= \mathrm{Rés}_{x_i}\left(f_i(x_i), r_{i-1}(t)\right) \quad \textit{pour} \quad 1 \leq i \leq N_p \quad ; \end{aligned}$$

calcule le polynôme $\left(f^{[1]},\dots,f^{[p]}\right)$*-caractéristique de* Θ*:*

$$\mathcal{X}^{\mathfrak{S}_{n_1,\dots,n_p}}_{\Theta,\left(f^{[1]},\dots,f^{[p]}\right)}(t)=r_{N_p}(t)\quad .$$

Preuve : Par récurrence sur p.

Pour $p=1$, il s'agit du théorème 4.6.

Le théorème étant supposé vrai pour un $(p-1)$-uplet sur un anneau intègre, l'hypothèse de récurrence peut être appliquée au $(p-1)$-uplet de polynômes unitaires $\left(f^{[1]},\dots,f^{[p-1]}\right)$ sur l'anneau intègre $R[x_{N_{p-1}+1},\dots,x_{N_p}]$:

$$\mathcal{X}^{\mathfrak{S}_{n_1,\dots,n_{p-1}}}_{\Theta,\left(f^{[1]},\dots,f^{[p-1]}\right)}(t)=r_{N_{p-1}}(t)\quad .$$

Or

$$\mathcal{X}^{\mathfrak{S}_{n_1,\dots,n_p}}_{\Theta,\left(f^{[1]},\dots,f^{[p]}\right)}(t)=\prod_{\sigma\in\mathfrak{S}_{n_1,\dots,n_p}}\left(t-\Theta\left(\alpha_{\sigma(1)},\dots,\alpha_{\sigma(N_{p-1})},\alpha_{\sigma(N_{p-1}+1)},\dots,\alpha_{\sigma(N_p)}\right)\right)$$

soit :

$$\begin{aligned}\mathcal{X}^{\mathfrak{S}_{n_1,\dots,n_p}}_{\Theta,\left(f^{[1]},\dots,f^{[p]}\right)}(t)&=\prod_{(\sigma,\tau)\in\mathfrak{S}_{n_1,\dots,n_{p-1}}\times\mathfrak{S}_{n_p}}\left(t-\Theta\left(\alpha_{\sigma(1)},\dots,\alpha_{\sigma(N_{p-1})},\alpha_{N_{p-1}+\tau(1)},\dots,\alpha_{N_{p-1}+\tau(n_p)}\right)\right)\\&=\prod_{\tau\in\mathfrak{S}_{n_p}}\prod_{\sigma\in\mathfrak{S}_{n_1,\dots,n_p}}\left(t-\Theta\left(\alpha_{\sigma(1)},\dots,\alpha_{\sigma(N_{p-1})},\alpha_{N_{p-1}+\tau(1)},\dots,\alpha_{N_{p-1}+\tau(n_p)}\right)\right)\\&=\prod_{\tau\in\mathfrak{S}_{n_p}}\mathcal{X}^{\mathfrak{S}_{n_1,\dots,n_{p-1}}}_{\Theta(x_1,\dots,x_{N_{p-1}},\alpha_{N_{p-1}+\tau(1)},\dots,\alpha_{N_{p-1}+\tau(n_p)}),\left(f^{[1]},\dots,f^{[p-1]}\right)}(t)\end{aligned}$$

qui s'obtient en appliquant le théorème 4.5 à $f^{[p]}$, à l'anneau intègre $A=R[x_{N_{p-1}+1},\dots,x_{N_p}]$ et à $\Phi=\mathcal{X}^{\mathfrak{S}_{n_1,\dots,n_{p-1}}}_{\Theta,\left(f^{[1]},\dots,f^{[p-1]}\right)}(t)=r_{N_{p-1}}(t)$ c'est à dire :

$$r_i(t)=\mathrm{Rés}_{x_i}\left(f_i,r_{i-1}(t)\right)\quad\text{pour}\quad N_{p-1}+1\leq i\leq N_p$$

calcule $\mathcal{X}^{\mathfrak{S}_{n_1,\dots,n_p}}_{\Theta,\left(f^{[1]},\dots,f^{[p]}\right)}(t)$, ce qui conclut la preuve du théorème. □

Il est maintenant possible d'énoncer une généralisation du théorème 4.4 aux multi-résolvantes :

Théorème 4.10. *Soient A un anneau intègre de caractéristique nulle ; un p-uplet de polynômes unitaires $\left(f^{[1]},\dots,f^{[p]}\right)$ à coefficients dans A, de degrés respectifs $n_1,\dots,n_p$; les entiers $N_0,\dots,N_p$ et les modules de Cauchy $f_{N_{j-1}+i}$, pour $1\leq i\leq n_j$ et $1\leq j\leq p$, définis comme au théorème 4.9 ; H_{N_p} un sous-groupe de $\mathfrak{S}_{n_1,\dots,n_p}$ et Θ un H_{N_p}-invariant primitif $\mathfrak{S}_{n_1,\dots,n_p}$-relatif ($\Theta\in A[x_1,\dots,x_{N_p}]$).*

Soient H_k, pour $1\leq k\leq N_p$ les N_p stabilisateurs sur $\mathfrak{S}_k\cap\mathfrak{S}_{n_1,\dots,n_p}$ de l'invariant Θ considéré comme un élément de $A[x_{k+1},\dots,x_{N_p}][x_1,\dots,x_k]$ définis par :

$$H_{N_{j-1}+i}=\mathrm{Stab}_{\mathfrak{S}_{n_1,\dots,n_{j-1},i}}(\Theta)\quad \textit{pour } 1\leq i\leq n_j \textit{ et } 1\leq j\leq p\quad .$$

Soient les entiers d_k *et* m_k, *pour* $1 \leq k \leq N_p$ *définis par:*

$$m_k = \frac{\operatorname{card}(H_k)}{\operatorname{card}(H_{k-1})} \quad ,$$

en posant $\operatorname{card}(H_0) = 1$ *; et*

$$d_{N_{j-1}+i} = \frac{n_1! \cdots n_{j-1}! i!}{\operatorname{card}\left(H_{N_{j-1}+i}\right)} \quad \textit{pour } 1 \leq i \leq n_j \textit{ et } 1 \leq j \leq p \quad .$$

Alors la suite finie $\{\mathcal{R}_0, \ldots, \mathcal{R}_{N_p}\}$ *définie récursivement par:*

$$\begin{aligned} &\mathcal{R}_0(t) = \operatorname{Récip}(t - \Theta) = 1 - t\Theta \quad ; \\ &\mathcal{R}_k(t) = \sqrt[m_k]{\operatorname{Rés}_{x_k}(f_k, \mathcal{R}_{k-1}(t)) \bmod t^{d_k+1}} \quad \text{pour} \quad 1 \leq k \leq n \quad ; \end{aligned}$$

calcule le polynôme réciproque de la multi-résolvante (absolue) du p-uplet $\left(f^{[1]}, \ldots, f^{[p]}\right)$ *par* Θ *:*

$$\operatorname{Récip}\left(\mathcal{L}^{\mathfrak{S}_{n_1,\ldots,n_p}}_{\Theta,\left(f^{[1]},\ldots,f^{[p]}\right)}\right)(t) = \mathcal{R}_{N_p}(t) \quad .$$

Preuve : Le théorème 4.9 appliqué à l'anneau $R = A[x_{N_{j-1}+i+1}, \ldots, x_{N_j}]$ qui est intègre et au j-uplet $\left(f^{[1]}, \ldots, f^{[j-1]}, f_{N_{j-1}+i}\right)$ de polynômes unitaires de $R[x]$ de degrés respectifs $n_1, \ldots, n_{j-1}, i$ et de modules de Cauchy $f_1, \ldots, f_{N_{j-1}+i}$ définies au théorème 4.9 (ils coïncident avec ceux définis pour $(f^{[1]}, \ldots, f^{[p]})$) montre que:

$$r_{N_{j-1}+i}(x) = \mathcal{X}^{\mathfrak{S}_{n_1,\ldots,n_{j-1},i}}_{\Theta,\left(f^{[1]},\ldots,f^{[j-1]},f_{N_{j-1}+i}\right)} \quad \text{pour } 1 \leq i \leq n_j \text{ et } 1 \leq j \leq p \quad .$$

D'après la proposition 2.8 appliquée au produit $f^{[1]} \ldots f^{[j-1]} f_{N_{j-1}+i}$,

$$\mathcal{X}^{\mathfrak{S}_{n_1,\ldots,n_{j-1},i}}_{\Theta,f^{[1]}\ldots f^{[j-1]}f_{N_{j-1}+i}} = \left(\mathcal{L}^{\mathfrak{S}_{n_1,\ldots,n_{j-1},i}}_{\Theta,f^{[1]}\ldots f^{[j-1]}f_{N_{j-1}+i}}\right)^{\operatorname{card}\left(H_{N_{j-1}+i}\right)} \quad .$$

Or,

$$\mathcal{L}^{\mathfrak{S}_{n_1,\ldots,n_{j-1},i}}_{\Theta,f^{[1]}\ldots f^{[j-1]}f_{N_{j-1}+i}} = \mathcal{L}^{\mathfrak{S}_{\{N_0+1,\ldots,N_0+n_1\}\times\cdots\times\mathfrak{S}_{n_1,\ldots,n_{j-1},i}}}_{\Theta,\left(f^{[1]},\ldots,f^{[j-1]},f_{N_{j-1}+i}\right)}$$

et

$$\mathcal{X}^{\mathfrak{S}_{n_1,\ldots,n_{j-1},i}}_{\Theta,f^{[1]}\ldots f^{[j-1]}f_{N_{j-1}+i}} = \mathcal{X}^{\mathfrak{S}_{n_1,\ldots,n_{j-1},i}}_{\Theta,(f^{[1]},\ldots,f^{[j-1]},f_{N_{j-1}+i})} \quad .$$

D'où, comme au paragraphe 4.4, la formule de récurrence sur les $\mathcal{R}_k$ du théorème 4.10.

Le degré de $\mathcal{L}^{\mathfrak{S}_{n_1,\dots,n_{j-1},i}}_{\Theta,f^{[1]}\dots f^{[j-1]}f_{N_{j-1}+i}}$ est :

$$\begin{aligned} d_{N_{j-1}+i} &= \operatorname{card}\left(\mathfrak{S}_{n_1,\dots,n_{j-1},i}.\Theta\right) \\ &= \frac{\operatorname{card}\left(\mathfrak{S}_{n_1,\dots,n_{j-1},i}\right)}{\operatorname{card}\left(H_{N_{j-1}+i}\right)} \\ &= \frac{n_1!\cdots n_{j-1}!i!}{\operatorname{card}\left(H_{N_{j-1}+i}\right)} \end{aligned}$$

d'où la formule donnée dans le théorème 4.10. □

Les entiers d_k et m_k pour $1 \leq k \leq N_p$ dans le théorème 4.10 peuvent être calculés plus simplement par :

Lemme 4.11. *Sous les hypothèses du théorème 4.10.*

Soit $H_i^{(j)}$ le stabilisateur de Θ considéré comme un élément de l'anneau de polynômes $A[x_{N_0+1},\dots,x_{N_{j-1}},x_{N_{j-1}+i+1},\dots,x_{N_p}][x_{N_{j-1}+1},\dots,x_{N_{j-1}+i}]$:

$$H_i^{(j)} = \operatorname{Stab}_{\mathfrak{S}_{N_{j-1}+1,\dots,N_{j-1}+i}}(\Theta) \quad \textit{pour } 1 \leq i \leq n_j \textit{ et } 1 \leq j \leq p \quad .$$

Alors en posant $H_0^{(j)} = 1$ et $d_{N_0} = 1$:

$$m_{N_{j-1}+i} = \frac{\operatorname{card}\left(H_i^{(j)}\right)}{\operatorname{card}\left(H_{i-1}^{(j)}\right)} \qquad \textit{et} \qquad d_{N_{j-1}+i} = d_{N_{j-1}}\frac{i!}{\operatorname{card}\left(H_i^{(j)}\right)}$$

pour $1 \leq i \leq n_j$ et $1 \leq j \leq p$.

Preuve : Il suffit de remarquer pour les m_k :

$$\operatorname{card}\left(H_{N_{j-1}+i}\right) = \operatorname{card}\left(H_{n_1}^{(1)}\right)\cdots\operatorname{card}\left(H_{n_{j-1}}^{(j-1)}\right)\operatorname{card}\left(H_i^{(j)}\right) \quad .$$

Et pour les d_k qu'en particulier :

$$d_{N_{j-1}} = \frac{n_1!\cdots n_{j-1}!}{\operatorname{card}\left(H_{N_{j-1}}\right)}$$

d'où

$$d_{N_{j-1}+i} = d_{N_{j-1}}\frac{\operatorname{card}\left(H_{N_{j-1}}\right)i!}{\operatorname{card}\left(H_{N_{j-1}+i}\right)} \quad \text{pour } 1 \leq i \leq n_j \text{ et } 1 \leq j \leq p \quad .$$

ce qui conclut la preuve du lemme. □

Remarque : Il est bien sûr possible d'appliquer aux multi-résolvantes les traitements particuliers exposés au paragraphe 4.5.

4.7 Numérotation des variables de l'invariant primitif

Le théorème 4.4 fonctionne évidemment pour toute numérotation des variables de l'invariant primitif, mais pour des raisons d'efficacité du calcul, pour éviter la croissance des coefficients, il est toujours plus intéressant de calculer les racines r-ièmes le plus tôt possible dans les calculs. Cela signifie qu'il est préférable de choisir une numérotation des variables qui fait apparaître rapidement les puissances superflues dans les calculs.

4.8 Conclusion

Nous avons fourni dans le théorème 4.4 une méthode, basée sur l'élimination, permettant de calculer les *résolvantes absolues* en se débarrassant des puissances parasites éventuelles. Cette méthode est étendue aux *multi-résolvantes* par le théorème 4.10.

Les remarques du paragraphe 2.6 s'appliquent particulièrement à cette méthode.

A. VALIBOUZE et N. RENNERT ont suggéré dans [85] d'effectuer ces calculs modulo l'*idéal des relations symétriques entre les racines de* f qui est défini par (voir le paragraphe 2.1.1) :

$$I_f^{\mathfrak{S}_n} = \left\{ \Psi \in A[x_1, \ldots, x_n] \;\middle|\; \begin{array}{c} (\sigma.\Psi)(\alpha_1, \ldots, \alpha_n) = 0 \\ \forall \sigma \in \mathfrak{S}_n \end{array} \right\} \quad ,$$

avec $(f_1, \ldots, f_n)$ qui forme une base standard de cet idéal (voir [98]). Cette approche limite la croissance du nombre de termes à manipuler.

Elle peut encore être étendue aux résolvantes relatives en remplaçant les modules de Cauchy par un idéal triangulaire comme présenté dans [7] et [100] : en utilisant le lemme 2.8, la preuve du théorème 4.4, peut être transposée pour calculer les *polynômes f-caractéristiques L-relatifs*.

Chapitre 5

Factorisation en théorie de Galois effective

Il a été rappelé au chapitre 1, que l'information permettant de distinguer plusieurs partitions possibles lors de la chasse aux résolvantes ne nécessite pas toujours la factorisation complète de la résolvante.

Dans le même temps, la liste des partitions possibles du degré de la résolvante en degrés de facteurs irréductibles, extraite de la matrice de partitions, apporte des informations *a priori* sur la factorisation de la résolvante. Or, les factorisateurs généralistes des logiciels de calcul formel ne sont pas conçus pour tirer parti de ces informations supplémentaires ni pour n'effectuer qu'une factorisation partielle.

Ce chapitre est consacré aux rapports que peut entretenir le processus de factorisation avec la « chasse aux résolvantes ».

Il présente un algorithme général de factorisation d'un polynôme à coefficients entiers. La description de cet algorithme nous permet d'examiner les différentes étapes de la factorisation. Pour chacune de ces étapes, nous précisons les rapports qu'elle entretient avec la théorie de Galois.

Plus précisément, nous détaillons les informations connues *a priori* dont les étapes de la factorisation peuvent bénéficier et de quelle façon s'en servir. Elles sont rassemblées sous les titres **informations utilisées**. Ces informations émanent des matrices de partitions et des matrices des groupes, voire des informations dérivées de ces matrices ou des étapes précédentes de l'algorithme de factorisation.

Nous nous attachons aussi à utiliser au mieux et dès que possible les informations discriminantes que chacune des étapes de cette factorisation peut fournir. Elles sont rassemblées sous les titres **informations recueillies**.

5.1 Un algorithme général de factorisation

Un historique des progrès effectués au cours des deux derniers siècles et plus spécifiquement les trois dernières décennies sur la factorisation des polynômes est détaillé, avec une bibliographie très fournie, dans les trois articles successifs de E. Kaltofen [49], [50] et [51].

Remarque : Les polynômes considérés sont supposés à coefficients entiers. La plupart des techniques et algorithmes auxquels il sera fait référence peuvent être étendus à des anneaux vérifiant les conditions et hypothèses, soit de [79], soit de [104].

La factorisation efficace des polynômes à coefficient entiers repose depuis les publications de E. R. Berlekamp [12] et [13] et H. Zassenhaus [117] sur le schéma communément appelé de Berlekamp-Zassenhaus. En fait la présentation que nous avons retenue ne fait pas directement appel aux travaux de E. R. Berlekamp. Ils sont remplacés par les améliorations qui ont succédé à la méthode de D. G. Cantor et H. Zassenhaus [26] qui est elle-même une amélioration d'une technique classique basée sur la factorisation en degré distincts des facteurs (voir [77] et [54, pages 429–430]).

Le procédé de factorisation de Berlekamp-Zassenhaus ne fonctionne que pour des polynômes **primitifs sans facteur carré**. Il nécessite donc un pré-calcul afin de ne lui proposer que des polynômes vérifiant ces conditions.

5.2 Préparation de la factorisation pour le schéma de Berlekamp-Zassenhaus

Nous notons dans la suite de ce chapitre f le polynôme de $\mathbb{Z}[t]$ que nous cherchons à factoriser.

5.2.1 Partie primitive

Définition 5.1. Le *contenu* d'un polynôme f est le pgcd de ses coefficients.

Le polynôme f est dit *primitif* si son contenu est égal à 1.

La *partie primitive* du polynôme f est le quotient de f par son contenu.

Proposition 5.2. *Un polynôme $f \in \mathbb{Z}[t]$ est primitif si et seulement si ses facteurs le sont.*

Remarque : Si un polynôme est unitaire alors il est primitif. C'est le cas des résolvantes de polynômes à coefficients entiers.

La factorisation du contenu fait appel aux algorithmes de factorisation sur $\mathbb{Z}$, techniques que nous ne détaillerons pas plus ici (voir par exemple [28]). La partie primitive sera factorisée par les algorithmes des paragraphes 5.2.2 et 5.4.

5.2.2 Factorisation sans facteur carré

Le polynôme f de $\mathbb{Z}[t]$ est ici supposé primitif.

La factorisation sans facteur carré est définie au paragraphe 2.5.2. Elle peut être obtenue par l'algorithme de Yun [116]. À l'issue de cet algorithme, le polynôme f de degré n est factorisé sous la forme :

$$f = f_1^1 \cdots f_n^n$$

avec $\text{pgcd}\,(f_i, f_j) = 1$ et $\text{pgcd}\,(f_i, f_i') = 1$, pour $1 \leq i < j \leq n$.

Remarque : La première étape de l'algorithme de Yun consiste à calculer le pgcd du polynôme f avec sa dérivée f'. Cette remarque offre la possibilité, pour calculer le discriminant, d'utiliser à cet endroit l'algorithme des sous-résultants (voir paragraphe 5.5.1).

Informations recueillies

La factorisation sans facteur carré non triviale d'une résolvante indique que cette résolvante est dans le premier cas du théorème 1.38.

La factorisation partielle obtenue permet d'entreprendre le schéma de Berlekamp-Zassenhaus sur chacun des polynômes f_i différents de 1, pour $1 \leq i \leq n$, qui sont eux sans facteur carré. Obtenir une factorisation sans facteur carré non triviale permet ainsi de couper le problème de factorisation en problèmes de tailles plus petites.

5.3 Raffinements à la préparation

5.3.1 Détection de polynômes particuliers

Au début d'un algorithme de factorisation, il est intéressant d'effectuer quelques tests rapides et peu coûteux pour déterminer si le polynôme donné est d'une forme particulière. Si un algorithme de factorisation spécifique et efficace est connu pour cette forme particulière de polynômes, son utilisation est substituée au schéma général de factorisation.

L'appel à ces tests ne sert pas qu'au début de la factorisation, mais est utile pour tout appel récursif de la factorisation: polynôme obtenu comme facteur sans carré (paragraphe 5.2.2), facteurs gauches de décomposition fonctionnelle polynomiale et polynômes obtenus par composition d'une factorisation non triviale (paragraphe 5.3.3), polynôme obtenu par détection de vrais facteurs dont l'irréductibilité n'est pas connue (paragraphe 5.5.2).

Polynômes dont le terme constant est nul

Soit k le plus petit degré des monômes de f :

$$f(t) = a_n t^n + \cdots + a_k t^k \quad .$$

Pour $k \neq 0$, le polynôme f se factorise en t^k et en le polynôme $a_n t^{n-k} + \cdots + a_k$.

Le test de ce cas particulier est à placer juste avant d'entreprendre la factorisation sans facteur carré.

Polynômes de degré 1

Proposition 5.3. *Si f est un polynôme primitif de degré 1 alors il est irréductible.*

Polynômes de degré 2

Il suffit d'adapter les formules classiques de résolution des équations en degré 2 en ne retenant que les solutions entières.

Proposition 5.4. *Soit f un polynôme primitif de degré 2 :*

$$f(t) = at^2 + bt + c \quad .$$

Son discriminant est $\Delta = b^2 - 4ac$. Si Δ n'est pas un carré dans $\mathbb{Z}$, alors f est irréductible.
Sinon, soit δ une racine carrée de Δ et $d = \text{pgcd}\left(\frac{b+\delta}{2}, a\right)$. La factorisation de f sur $\mathbb{Z}[t]$ est alors :

$$f(t) = \left(\frac{a}{d}t + \frac{b+\delta}{2d}\right)\left(d\,t + \frac{d(b-\delta)}{2a}\right) \quad . \tag{5.1}$$

Les deux facteurs sont distincts si f est sans facteur carré ($\Delta \neq 0$).

Preuve : C'est immédiat, en remarquant au passage que l'équation (5.1) est à coefficients entiers car $b + \delta \equiv 0 \bmod 2$. □

Autres polynômes particuliers

Il existe aussi, pour les polynômes très particuliers de la forme $t^n - 1$, des algorithmes de factorisation explicites à l'aide des polynômes cyclotomiques.

Définition 5.5. Soit n un entier strictement positif. Le *n-ième polynôme cyclotomique*, noté $\Phi_n(t) \in \mathbb{Z}[t]$ est, par définition, le polynôme minimal des racines primitives n-ièmes de l'unité :

$$\Phi_n(t) = \prod_{1 \leq k < n \ \mid \ \text{pgcd}(k,n)=1} \left(t - e^{2i\pi\frac{k}{n}}\right) \quad .$$

Proposition 5.6. *Pour n un entier strictement positif, on a :*

$$t^n - 1 = \prod_{d \mid n} \Phi_d(t) \quad .$$

Le cas des polynômes binomiaux, de la forme $t^n - a$, a été exploré par J. H. DAVENPORT dans [36] ainsi que quelques autres classes de polynômes particuliers dont les polynômes « unaires », de coefficients égaux à plus ou moins 1 et les polynômes trinomiaux.

5.3.2 Critères d'irréductibilité

Avant de lancer le schéma de factorisation de Berlekamp-Zassenhaus, l'irréductibilité de certains polynômes peut être détectée par des critères dont le temps de calcul est faible devant celui de la factorisation. Nous rappelons en premier lieu celui d'Eisenstein :

Proposition 5.7 (Einsenstein). *Soient n un entier strictement positif et f un polynôme de degré n donné par :*

$$f(t) = a_n t^n + \ldots + a_0$$

avec $a_i \in \mathbb{Z}$, $0 \leq i \leq n$ et $a_n \neq 0$. S'il existe un entier premier p tel que p ne divise pas a_n, p divise a_i pour $n-1 \geq i \geq 0$ et p^2 ne divise pas a_0 alors le polynôme f est irréductible.

Preuve : Voir [67, page 41]. □

Dans la pratique, ce critère est testé avec les entiers premiers qui apparaissent dans la factorisation sur $\mathbb{Z}$ du pgcd des coefficients $\{a_{n-1}, \ldots a_1\}$ et qui sont faciles à trouver. Ce critère peut être simultanément appliqué au polynôme réciproque de f. Une autre catégorie de critères d'irréductibilité est attachée aux valeurs entières prises par les polynômes. Nous rappelons en premier le critère proposé par W. S. BROWN et R. L. GRAHAM [21] :

Proposition 5.8. *Soient n un entier strictement positif et f un polynôme de degré n :*

$$f(t) = a_n t^n + \ldots + a_0$$

avec $a_i \in \mathbb{Z}$, $0 \leq i \leq n$ et $a_n \neq 0$. Soit $\mathcal{S}(f)$ l'ensemble des valeurs de f évalué sur les entiers :

$$\mathcal{S}(f) = \{\ldots, f(-1), f(0), f(1), \ldots\} \quad .$$

Soient u le nombre d'unités dans $\mathcal{S}(f)$ et p le nombre de premiers y apparaissant (comptés avec leur multiplicité). Si

$$p + 2u > n + 4$$

alors f est irréductible.

Preuve : Voir [21, corollaire 2]. □

Ce critère suppose qu'il existe un grand nombre de premiers parmi les valeurs prises par le polynôme f, ce qui rend assez faibles ses chances de succès.

Par contre un autre critère dû à J. BRILLHART [20] permet de prouver l'irréductibilité d'un polynôme à condition de détecter une seule valeur première suffisamment loin des racines :

Théorème 5.9 (Brillhart). *Soient n un entier strictement positif et f un polynôme de degré n à coefficients entiers. Soit b un entier qui majore strictement les modules des racines complexes de f.*

Soit m un entier tel que $|m| > b$. Si $f(m)$ est premier alors f est irréductible dans $\mathbb{Z}[t]$.

Preuve : Voir plus bas ou [20, théorème 1]. □

Ce résultat a été étendu par J. H. DAVENPORT [38] ainsi que, plus généralement, par M. B. MONAGAN [78].

Lemme 5.10 (Monagan). *Soient n un entier strictement positif et f un polynôme de degré n à coefficients entiers. Soit b un entier qui majore strictement les modules des racines complexes de f.*

Soient g un facteur de degré d strictement positif de f et m un entier avec $|m| > b$. Alors

$$|g(m)| > (|m| - b)^d \quad .$$

Preuve : Voir [78, lemme 2] ou, en notant $\alpha_1, \ldots, \alpha_n$ les racines de f :

le polynôme g dont les racines sont notées $\alpha_{i_1}, \ldots, \alpha_{i_d}$ peut s'écrire sous la forme :

$$g(t) = g_d \prod_{j=1}^{d} (t - \alpha_{i_j})$$

avec $g_d = \mathrm{cd}\,(g)$.

$$\begin{aligned} \text{Alors } |g(m)| &\geq |g_d| \prod_{j=1}^{d} |m - \alpha_{i_j}| \\ &\geq |g_d| \prod_{j=1}^{d} \left(|m| - b + b - |\alpha_{i_j}|\right) \\ &> |g_d| \left(|m| - b\right)^d \\ &> \left(|m| - b\right)^d \quad . \end{aligned}$$

□

Théorème 5.11 (Monagan). *Soient n un entier strictement positif et f un polynôme de degré n à coefficients entiers. Soit b un entier qui majore strictement les modules des racines complexes de f.*

Supposons que tous les facteurs du polynôme f dans $\mathbb{Z}[t]$ soient de degré supérieur ou égal à d. Soit m un entier avec $|m| > b$.

Si v est un entier qui divise $f(m)$ avec $|v| \leq (|m| - b)^d$ et $f(m)/v$ est premier alors le polynôme f est irréductible.

Preuve : Voir [78, théorème 3] ou, en notant $\alpha_1, \ldots, \alpha_n$ les racines de f, ce qui suit.

Soit g un facteur de f. Le polynôme g étant par hypothèse de degré supérieur à d, il vérifie, d'après le lemme 5.10, :

$$|g(m)| > \left(|m| - b\right)^d \quad .$$

Donc, si un entier v divise $f(m)$ avec $|v| \leq (|m| - b)^d$ alors $|g(m)/v| > 1$.
Donc si f se factorise sur $\mathbb{Z}[t]$ alors $f(m)/v$ se factorise sur $\mathbb{Z}$. □

Les tests de primalité sur $\mathbb{Z}$ dans les théorèmes 5.9 et 5.11 peuvent être effectués par le critère de Rabin [83]. Une factorisation partielle qui cherche d'abord les plus petits facteurs de l'évaluation de f en l'entier m convient particulièrement au théorème 5.11.

Deux paramètres importants pour l'efficacité de ces critères sont la valeur de la borne sur les modules des racines des polynômes, et le nombre d'évaluations à effectuer.

Une liste de bornes maximales sur les modules des racines de polynômes peut être trouvée dans [74, théorème 2, pages 154–155]. Il est intéressant d'avoir la plus petite borne possible sur le module des racines en raison de la remarque suivante de J. BRILLHART [20, remarque 4] :

Propriété 5.12. *Soient n un entier strictement positif et f un polynôme de degré n. Soient v et m des entiers. Si v divise $f(m)$ alors, pour tout entier k, v divise $f(m+kv)$.*

Preuve : C'est immédiat avec le développement de Taylor de f :

$$f(m+kv) = f(m) + kvf'(m) + \cdots + \frac{(kv)^n}{n!} f^{(n)}(m) \quad .$$

□

Comme $|f(t)|$ croît quand $|t|$ s'éloigne de la borne b sur les modules des racines, les évaluations de f pour des valeurs entières de t ne seront pas irréductibles sauf peut-être pour les valeurs de k pour lesquelles $|m+kv|$ est le plus proche de b par valeurs supérieures et $|f(m+kv)| = |v|$ est premier. Le test de Brillhart est donc inutile pour toutes les autres valeurs de k.

Le test de Brillhart ne fonctionnera pas toujours, même en évaluant f sur un grand nombre de valeurs entières : certains polynômes comme $x^2 + x + 4$ ne prennent jamais de valeurs premières.

Définition 5.13. Soient n un entier strictement positif et f un polynôme de degré n, $f \in \mathbb{Z}[t]$.

Le *diviseur fixé* de f est le pgcd des entiers $f(m)$ pour tout $m \in \mathbb{Z}$.

Le test de Monagan peut être utilisé avec les polynômes de diviseur fixé différent de 1. Cependant, le succès du test est subordonné à une ancienne conjecture de théorie des nombres (voir [78]) qui, même prouvée, ne semble pas conduire à des bornes réalistes pour un nombre d'évaluations accessible au calcul (voir [4]).

Comme le suggèrent [20] et [78], les tests peuvent être aussi effectués sur les polynômes réciproques.

Dans la pratique, ces tests ne sont utilisables que si les coefficients du polynôme à factoriser ne sont pas trop gros, ou tout au moins si certaines de leurs évaluations en des entiers se prêtent à des tentatives de factorisation sur les entiers (qui peuvent n'être que partielles).

Informations utilisées

Dans le cas de la factorisation d'une résolvante, la liste des partitions possibles du degré de cette résolvante fournit un minorant du degré des facteurs possibles, utilisable avec le théorème 5.11, accroissant ainsi son efficacité (voir paragraphe 5.5.1).

Informations recueillies

Prouver l'irréductibilité par un critère clôt très tôt le processus de factorisation.

Dans le cas de la factorisation d'une résolvante, si les évaluations en un entier m vérifiant les conditions du théorème 5.9 par rapport aux racines de la résolvante conduisent à des factorisations avec peu de facteurs premiers et si ce nombre de facteurs premiers est plus petit que le degré de la résolvante, alors il donne un majorant sur le nombre de facteurs irréductibles de cette résolvante. Cette majoration permet d'éliminer de la liste des partitions possibles les partitions du degré de la résolvante en un plus grand nombre de parts, ainsi que les classes candidates associées à cette partition.

5.3.3 La décomposition polynomiale fonctionnelle

Une propriété fréquemment utilisée par les factorisateurs est celle de la parité du polynôme f, c'est-à-dire s'il s'écrit sous la forme

$$f(t) = g(t^2)$$

où $g \in \mathbb{Z}[t]$. Le polynôme g est d'abord factorisé sur $\mathbb{Z}[t]$ sous la forme :

$$g = g_1 \cdots g_k \quad ,$$

puis chacun des polynômes $g_i(t^2)$ l'est ensuite pour obtenir une factorisation complète du polynôme f sur $\mathbb{Z}[t]$.

Cette remarque est immédiatement applicable aux polynômes de la forme $f = g(t^s)$ avec $s \geq 2$. Elle peut même être généralisée à la recherche d'une décomposition fonctionnelle polynomiale (voir [56]).

Définition 5.14. Soit f un polynôme à coefficients dans $\mathbb{Z}$. La *décomposition fonctionnelle* du polynôme f est la donnée de polynômes $g_1, \ldots, g_k$ appartenant à $\mathbb{Z}[t]$ et vérifiant

$$f = g_1 \circ \cdots \circ g_k \quad .$$

Les polynômes g_i sont les *composants* de la décomposition fonctionnelle de f.

Si le degré du polynôme g_i est 1 alors il est dit *trivial* (les polynômes $ax + b$ et $\frac{1}{a}(x - b)$ sont inverses pour la composition).

Une décomposition est dite *triviale* si l'un des polynômes g_i est trivial.

Si f n'admet que des décompositions triviales alors il est dit *indécomposable*.

Supposons que f admette une décomposition polynomiale fonctionnelle non triviale avec $g, h \in \mathbb{Z}[t]$ tels que

$$f = g(h) \quad .$$

L'intérêt d'une telle décomposition pour l'algorithme de factorisation est de chercher d'abord à factoriser le polynôme g, dont le degré est, parfois de beaucoup, plus petit que celui de f.

Si g se factorise en

$$g = g_1 \cdots g_\ell$$

alors le produit des factorisations des polynômes $g_i(h)$ est une factorisation de f sur $\mathbb{Z}[t]$.

Trouver une décomposition polynomiale fonctionnelle non triviale permet donc parfois de casser le problème de factorisation en problèmes plus petits. Cela ne constitue un gain sur l'algorithme général que lorsque le polynôme g n'est pas irréductible car sinon, le temps passé à en chercher une factorisation est depensé en pure perte.

Pour casser récursivement le problème le plus possible, la recherche d'un polynôme g candidat à la décomposition de f sous la forme $f = g(h)$ se fait en commençant par les plus grands degrés de g possibles (ce degré devant être un diviseur de celui de f). Si g est lui-même décomposable, sa factorisation peut aussi faire appel à sa décomposition, et ainsi de suite jusqu'à ce que le facteur à gauche de la décomposition soit indécomposable.

Le premier algorithme efficace de décomposition polynomiale (en temps polynomial pour les polynômes unitaires) a été proposé par D. Kozen et S. Landau dans [56] puis amélioré par J. Von zur gathen dans [105].

Informations utilisées

D. Kozen et S. Landau ont aussi donné un critère pour l'existence d'une décomposition polynomiale fonctionnelle non triviale [56, théorème 8]. Ce critère montre notamment que l'existence d'une décomposition polynomiale fonctionnelle non triviale implique que le groupe de Galois du polynôme à factoriser est imprimitif. Si l'information déjà connue sur la classe de conjugaison du groupe de Galois de la résolvante permet d'exclure tout sous-groupe imprimitif alors il n'est pas même nécessaire de rechercher une décomposition polynomiale fonctionnelle de cette résolvante.

Par ailleurs, les facteurs irréductibles de g doivent correspondre au degré d'un produit de facteurs de f divisé exactement par le degré de h. Ils sont ainsi limités par la liste des partitions possibles pour les facteurs de f.

Informations recueillies

Si le polynôme f se décompose, alors son groupe de Galois est imprimitif. Cette information peut être utilisée pour rejeter certaines classes de conjugaison possibles pour le groupe de Galois de f.

Si une factorisation non triviale de $g = g_1 g_2$ est trouvée alors $f = g_1(h)g_2(h)$ est une factorisation non triviale de f. Les partitions du degré de f qui ne sont pas compatibles avec cette factorisation sont éliminées de la liste des partitions possibles, ainsi que les classes de conjugaison du groupe de Galois qui leur sont associées.

5.4 La factorisation de Berlekamp-Zassenhaus

Le schéma de factorisation de Berlekamp-Zassenhaus utilise la factorisation de l'image du polynôme f dans $\mathbb{F}_p[t]$. Elle est décrite au paragraphe 5.4.1. Cette factorisation est remontée sur les entiers par un lemme de Hensel effectif en une factorisation modulo un entier m (égal à une puissance de p) de l'image de f dans $\mathbb{Z}/m\mathbb{Z}$. Cette remontée est décrite au paragraphe 5.4.2. Pour m suffisamment grand, les facteurs irréductibles sur $\mathbb{Z}[t]$ de f sont retrouvés à partir des images modulo m de ces facteurs. Ceci est exploré au paragraphe 5.4.3.

Nous supposons qu'à l'issue des opérations décrites au paragraphe 5.2, le polynôme f soumis au schéma de Berlekamp-Zassenhaus est primitif et sans facteur carré.

5.4.1 Factorisation modulaire

La première étape du schéma de Berlekamp-Zassenhaus consiste à choisir un entier premier p tel que l'image $\overline{f}$ de f dans $\mathbb{F}_p[t]$ soit sans facteur carré et de même degré que f.

C'est le cas lorsque l'entier premier p ne divise pas le discriminant du polynôme f (voir le paragraphe 2.3.5). Il n'y a donc qu'un nombre fini de premiers qui ne vérifient pas cette condition. Supposons choisi un tel entier p.

La factorisation de $\overline{f}$ s'effectue alors en trois étapes.

Factorisation sans facteur carré

Cette étape est inutile dans le cas qui nous intéresse puisque nous avons choisi le premier p de façon à ce que l'image $\overline{f}$ de f modulo p soit sans facteur carré.

Elle n'est rappelée ici que pour mémoire de l'algorithme général de factorisation sur $\mathbb{F}_p[t]$. Dans la pratique, les algorithmes de la factorisation sans facteur carré évoqués au paragraphe 5.2.2 sont adaptés à la caractéristique non-nulle.

Un traitement spécifique est réservé aux multiplicités que p divise pour adapter l'algorithme de Musser (rappelé dans [116]) ou supérieures ou égales à p pour adapter l'algorithme de Yun (voir, par exemple, [81, page 201]).

Factorisation en degrés distincts

Cet algorithme n'a pas d'équivalent sur $\mathbb{Z}[t]$. Il consiste à regrouper les facteurs par degré.

Définition 5.15. Soit $\overline{f}$ un polynôme sans facteur carré de degré n à coefficients dans $\mathbb{F}_p$.

La *factorisation en degrés distincts du polynôme* $\overline{f}$ est la donnée des polynômes $f_1, \ldots, f_n$ appartenant à $\mathbb{F}_p[t]$ tels que

$$\overline{f} = f_1 \cdots f_n$$

et que f_i soit le produit des facteurs irréductibles de $\overline{f}$ de degré i sur $\mathbb{F}_p[t]$.

Connaître la factorisation en degrés distincts d'un polynôme revient donc à connaître le degré de chacun de ses facteurs irréductibles sans avoir même déterminé tous les facteurs. Et dans le cas où tous les facteurs de $\overline{f}$ sont de degrés tous distincts, la factorisation en degrés distincts est suffisante pour obtenir la factorisation en facteurs irréductibles sur $\mathbb{F}_p[t]$.

E. KALTOFEN et V. SHOUP ont proposé dans [53] une méthode de factorisation en degrés distincts basée sur le principe « pas de bébé / pas de géant » (*baby step/giant step*).

Elle repose sur le lemme suivant :

Lemme 5.16. *Soit p un premier. Soient i et j des entiers positifs. Le polynôme $t^{p^i} - t^{p^j}$ est divisible par exactement les polynômes irréductibles de $\mathbb{F}_p[t]$ dont le degré divise $i - j$*

Preuve : Voir [53]. □

En notant $f_{[(j-1)\ell+1,(j-1)\ell+\ell]}$ le produit des facteurs irréductibles de $\overline{f}$ dont le degré est compris entre $(j-1)\ell+1$ et $(j-1)\ell+\ell$, l'algorithme de Kaltofen et Shoup s'écrit (d'après [88]) :

Algorithme de factorisation en degrés distincts de Kaltofen et Shoup

Entrée : Un entier $n \geq 0$ et un polynôme $\overline{f} \in \mathbb{F}_p[t]$, sans facteur carré de degré n
Sortie : $f_1, \ldots, f_n$ où f_i est le produit des facteurs irréductibles de degré i de $\overline{f}$.

(1) $B \leftarrow \lfloor n/2 \rfloor$,
$\ell \leftarrow \lfloor \sqrt{B} \rfloor$, {Nombre de « pas de bébé »}
$m \leftarrow \lceil B/\ell \rceil$, {Nombre de « pas de géant »}
$h_0 \leftarrow t$. $\{h_0 \in \mathbb{F}_p[t]\}$
(2) $h_1 \leftarrow t^p \bmod \overline{f}$, $\{h_1 \in \mathbb{F}_p[t]\}$
Pour i passant de 2 à m
$h_i \leftarrow t^{p^i} \bmod \overline{f}$ $\{h_i \in \mathbb{F}_p[t]\}$
(3) Pour j passant de 1 à m
$H_j \leftarrow t^{p^{\ell j}} \bmod \overline{f}$ $\{H_j \in \mathbb{F}_p[t]\}$
(4) Pour j passant de 1 à m
$I_j \leftarrow \prod_{i=0}^{\ell-1} (H_j - h_i) \bmod \overline{f}$ $\{I_j \in \mathbb{F}_p[t]\}$
(5) $f^* \leftarrow \overline{f}$, {« Pas de géant »}
Pour j passant de 1 à m
$f_{[(j-1)\ell+1,(j-1)\ell+\ell]} \leftarrow \operatorname{pgcd}(f^*, I_j)$
$f^* \leftarrow f^* / f_{[(j-1)\ell+1,(j-1)\ell+\ell]}$
(7) Si $f^* \neq 1$ alors $f_{\deg f^*} \leftarrow f^*$
(8) Pour j passant de 1 à m {« Pas de bébé »}
$g^* \leftarrow f_{[(j-1)\ell+1,(j-1)\ell+\ell]}$
Pour i passant de $\ell - 1$ à 0
$f_{(j-1)\ell+\ell-i} \leftarrow \operatorname{pgcd}(g^*, H_j - h_i)$
$g^* \leftarrow g^* / f_{(j-1)\ell+\ell-i}$

La validité de cet algorithme est une conséquence du lemme 5.16.

La méthode de Kaltofen et Shoup a pour intérêt de fournir un bon compromis effectif (et non seulement théorique) entre temps de calcul et besoin d'espace mémoire (voir [88]).

Mais au-delà, elle informe aussi, dès l'étape des « pas de géant », sur les degrés possibles des facteurs irréductibles modulo p.

À l'issue de la factorisation en degrés distincts, quand elle n'a pas directement abouti à factoriser complètement l'image du polynôme f modulo p, les degrés des facteurs modulo p sont au moins connus.

Pour obtenir les facteurs eux-mêmes, il ne reste plus qu'à séparer des produits de polynômes dont les degrés sont tous égaux et connus à l'avance. C'est l'objet du paragraphe suivant.

Factorisation en degrés égaux

Définition 5.17. Soit $\overline{f} \in \mathbb{F}_p[t]$ un polynôme unitaire de degré n produit de n/d polynômes irréductibles distincts de degrés d.

La *factorisation en degrés égaux du polynôme* $\overline{f}$ est la donnée des polynômes $f_1, \ldots, f_{n/d}$ appartenant à $\mathbb{F}_p[t]$ tels que

$$\overline{f} = f_1 \cdots f_{n/d}$$

où f_i pour $1 \leq i \leq n/d$ est un facteur irréductible unitaire de $\overline{f}$ de degré d sur $\mathbb{F}_p[t]$.

La factorisation en degrés égaux peut se faire à partir d'un algorithme probabiliste. Une version possible de cet algorithme est l'amélioration de [108] donnée par V. Shoup dans [88]. Elle repose sur une technique de calcul de trace [108] pour calculer le séparateur de McEliece (voir [10] et [68]).

5.4.2 Remontée des facteurs modulaires par un lemme de Hensel effectif

Une fois connue la factorisation de l'image du polynôme f modulo le premier p, la « remontée de Hensel », technique suggérée par H. Zassenhaus dans [117], consiste à trouver, à partir de f et des facteurs modulaires de l'image de f modulo p, des images des facteurs de f modulo m, où m est une puissance de p. Lorsque m est suffisamment grand, il est possible de retrouver les facteurs sur $\mathbb{Z}[t]$ à l'aide des algorithmes du paragraphe 5.4.3.

Définition 5.18. Soient m un entier primaire (puissance d'un premier) et $f_1, \ldots, f_r$ les r facteurs premiers entre eux de l'image de f modulo m alors

$$f \equiv f_1 \cdots f_r \bmod m \quad ,$$

avec $\mathrm{pgcd}\,(f_i, f_j) = 1 \bmod m$ pour $1 \leq i < j \leq r$.

Un *lemme de Hensel effectif* est un algorithme qui permet de calculer, pour un entier primaire m', puissance du même premier que m, strictement supérieur à m, l'image des facteurs de f modulo m' à partir de f et des facteurs de son image modulo m.

$$f \equiv f'_1 \cdots f'_r \bmod m' \quad ,$$

avec $\text{pgcd}\left(f'_i, f'_j\right) = 1 \bmod m'$ pour $1 \leq i < j \leq r$ et, pour $1 \leq i \leq r$,

$$f'_i \equiv f_i \bmod m \quad .$$

Dans le cas où f n'est pas unitaire, l'image du coefficient dominant de f modulo m sera toujours conservée au cours des calculs comme coefficient dominant du facteur f_1, les autres facteurs f_i pour $i \geq 2$ étant maintenus unitaires.

Typiquement, l'entier m est de la forme $m = q^k$ où q est un entier primaire.

Lorsque $m' = m^2$ le lemme de Hensel effectif est appelé *lemme de Hensel quadratique.*

Lorsque $m' = qm$, il est appelé de *lemme de Hensel linéaire.*

Des exemples de lemmes de Hensel effectifs sont donnés dans [117] ou [79].

5.4.3 Factorisation finale sur les entiers

Nous supposons connue une borne, notée $B(f)$, sur le maximum des valeurs absolues des coefficients de chacun des facteurs irréductibles du polynôme f.

L'algorithme de remontée décrit au paragraphe 5.4.2 est utilisé pour obtenir une factorisation de f en r facteurs modulo m avec m vérifiant $m > 2\text{cd}\,(f)\, B(f)$:

$$f \equiv f_1 \cdots f_r \bmod m \quad ,$$

avec $\text{cd}\,(f_1) \equiv \text{cd}\,(f) \bmod m$ et $\text{cd}\,(f_i) = 1$ pour $2 \leq i \leq r$.

La méthode classique consiste à tester comme diviseur de f sur $\mathbb{Z}[t]$, pour toutes les parties P de cardinal croissant de l'ensemble $\{1, \ldots r\}$, chaque polynôme h_P égal à la partie primitive (sur $\mathbb{Z}$) du produit :

$$\text{cd}\,(f) \prod_{i \in P} f_i \bmod m \quad ,$$

en représentant les coefficients de h_P par des entiers compris entre $-m/2$ et $m/2$.

Lorsque h_P divise exactement f dans $\mathbb{Z}[t]$ c'est un facteur de f et le processus est répété avec les facteurs modulaires restants jusqu'à épuisement et factorisation complète.

Malheureusement cette méthode peut se révéler exponentielle en le degré de f dans le pire des cas : il existe des polynômes irréductibles qui se factorisent en facteurs de degré 2 modulo tous les premiers (voir [52]). Le cardinal des partitions du degré $n/2$ qui correspond alors au nombre d'essais infructueux (car le polynôme est irréductible) est $2^{n/2}$.

Une méthode de complexité polynomiale garantie a été donnée par [63]. Elle consiste à trouver, pour tout facteur modulaire de f choisi, le facteur sur $\mathbb{Z}[t]$ dont il est un diviseur modulo m. Cette méthode utilise une technique de recherche de vecteurs courts dans un réseau. Pour garantir son succès, il faut atteindre un modulo beaucoup plus gros que $B(f)$ ce qui alourdit l'étape de remontée de Hensel du paragraphe 5.4.2. Par ailleurs, la puissance à laquelle est porté, pour la complexité, le degré de f ($O(n^6)$ opérations arithmétiques sur des entiers de taille $O(n^3)$ soit $O(n^{12})$ opérations booléennes, qui peuvent être abaissées avec des techniques de multiplication rapide, voir [106]) n'ôte pas, pour les degrés n les plus bas, l'intérêt de la méthode exponentielle dans le pire des cas.

5.5 Raffinements au schéma de factorisation

Le schéma de factorisation de Berlekamp et Zassenhaus a fait l'objet de nombreuses variantes et améliorations. Nous allons détailler ici quelques points qui interagissent avec la factorisation d'une résolvante dans le cadre d'une recherche de groupe de Galois.

5.5.1 Choix du premier pour les calculs modulaires

Le discriminant

La factorisation modulaire exposée au paragraphe 5.4.1 commence par le choix d'un premier p qui ne divise pas le discriminant (défini au paragraphe 2.3.5).

Cependant le discriminant d'un polynôme de degré élevé avec de gros coefficients (c'est typiquement le cas de certaines résolvantes) peut être extrêmement difficile à calculer en raison de l'explosion de la taille des entiers à manipuler.

Notation 5.19. Soit f un polynôme de degré n :

$$f(t) = a_n t^n + \cdots + a_0$$

La *norme euclidienne* du polynôme f, ou bien du vecteur de $(a_0, \ldots, a_n) \in A^{n+1}$ qui lui est associé est notée $|f|_2$:

$$|f|_2 = \sqrt{a_n^2 + \cdots + a_0^2} \quad .$$

Théorème 5.20 (Inégalité de Hadamard). *Soient n un entier et $b_1, \ldots, b_n$ des vecteurs de A^n, alors*

$$\left| \text{dét} \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} \right| \leq |b_1|_2 \cdots |b_n|_2 \quad .$$

Preuve : Voir par exemple [74, théorème 2, page 319]. □

Corollaire 5.21. *Le discriminant $\Delta(f)$ du polynôme f de degré n,*

$$f(t) = a_n t^n + \cdots + a_0$$

est majoré par :

$$\Delta(f) \leq \sqrt{n}\, |f|_2^{2n-1} / a_n \quad .$$

Preuve : C'est immédiat d'après la définition 2.28 du discriminant, la propriété 2.29 et le théorème 5.20. □

Petits coefficients. Si la borne donnée par le corollaire 5.21 est considérée comme suffisamment petite (ce critère est à ajuster en fonction des moyens de calculs disponibles) et s'il n'est pas déjà connu par une autre méthode le discriminant est calculé.

Informations recueillies

Une technique efficace de calcul du discriminant est d'utiliser les calculs du résultant de f et f' par l'algorithme des sous-résultants et la proposition 2.29. Ce calcul peut même être fait à l'occasion du pgcd nécessaire à la première étape des algorithmes de factorisation sans facteurs carrés (voir paragraphe 5.2.2).

Une fois le discriminant de f calculé, il est facile de voir s'il est un carré parfait, auquel cas le groupe de Galois de f appartient, ou n'appartient pas, à la classe de conjugaison d'un sous-groupe du groupe alterné $\mathfrak{A}_n$.

Comme suggéré par G. KOLESOVA et J. MCKAY dans [55], si l'algorithme des sous-résultants est utilisé pour le calcul du discriminant, la suite de Sturm obtenue permet aussi de fournir les longueurs de cycles de l'élément du groupe de Galois de f correspondant à la conjugaison complexe ce qui permet d'exclure de la liste des classes de conjugaison possibles celles qui ne possèdent aucun groupe avec de telles longueurs de cycles (voir le paragraphe 5.5.1).

Gros coefficients. Si par contre le polynôme à factoriser a de très gros coefficients alors le calcul de son discriminant dans $\mathbb{Z}$ risque d'être trop difficile. Dans ce cas ce calcul est remplacé par le calcul du discriminant modulo le premier p choisi à partir du résultant des polynômes $\overline{f}$ et $\overline{f}'$ et de la proposition 2.29 pour le corps $\mathbb{F}_p$. La taille des entiers rencontrés au cours des calculs reste bornée par p.

Informations recueillies

Si le résultant $\text{Rés}_t\left(f(t), f'(t)\right) \bmod p$ est nul, le premier p choisi ne convient pas car il est un diviseur du discriminant de f.

Un test possible supplémentaire qui permet, lorsque cette information est recherchée, d'affirmer que le discriminant n'est pas un carré parfait est de calculer le résultant de f et f' modulo p^2. Si p divise le discriminant sans que p^2 le divise, alors le discriminant de f n'est pas un carré parfait. Cette remarque peut être étendue à la plus grande puissance impaire de p qui divise le discriminant : si p^k divise le discriminant avec k impair sans que p^{k+1} le divise, alors le discriminant de f n'est pas un carré parfait. Cependant les anneaux $\mathbb{Z}/p^k\mathbb{Z}$ pour $k \geq 2$ ne sont pas intègres ce qui empêche l'utilisation de l'algorithme des sous-résultants pour le calcul du résultant de f et f' modulo p^k (voir le paragraphe 2.6 pour résoudre ce problème).

Les longueurs de cycles d'éléments du groupe de Galois

Il est rappelé au paragraphe 5.4.3 que l'étape de réconciliation des facteurs modulaires en facteurs à coefficients dans $\mathbb{Z}$ a un coût exponentiel dans le pire des cas. Et l'exposant dépend du nombre de facteurs modulaires trouvés et remontés.

Il est donc intéressant de choisir un premier pour lequel le nombre de facteurs modulaires sera le plus petit possible. Pour cela, le polynôme f est factorisé en degrés distincts comme exposé au paragraphe 5.4.1 pour plusieurs premiers dans le but de retenir le premier qui aura produit le moins de facteurs modulaires.

Avec les degrés des facteurs irréductibles de f modulo p, la factorisation en degrés distincts (paragraphe 5.4.1) fournit, des informations sur le groupe de Galois de f.

Notation 5.22. Soit f un polynôme de degré n sans facteur carré et p un premier ne divisant pas le discriminant de f. La partition de n en les degrés des facteurs irréductibles de $\overline{f} \equiv f \bmod p$ est notée $\varpi(f,p)$.

Théorème 5.23. *Soit f un polynôme de degré n sans facteur carré et p un premier ne divisant pas le discriminant de f. Alors il existe un élément du groupe de Galois de f dont les longueurs de cycles sont $\varpi(f,p)$.*

Preuve : Voir [102, pages 190–191], [27] ou [101]. □

En fait les rapports entre partitions du degré et longueurs de cycles d'éléments du groupe de Galois sont plus profonds :

Théorème 5.24 (Frobenius-Čebotarev). *Soit Γ le groupe de Galois du polynôme f séparable de degré n. Soit $\mathcal{C}$ l'ensemble des éléments de Γ dont les longueurs de cycles correspondent à la partition ϖ de n, alors :*

$$\lim_{m\to\infty} \frac{\operatorname{card}\left(\{p \text{ premier}, p \leq m \mid \varpi(f,p) = \varpi\}\right)}{\{p \text{ premier}, p \leq m\}} = \frac{\operatorname{card}(\mathcal{C})}{\operatorname{card}(\Gamma)} \quad .$$

Preuve : Voir [101, théorème 3] ou [92]. □

Cette information très précise sur la fréquence avec laquelle apparaissent les différentes partitions de n en longueurs de cycle invite à distinguer les différentes classes de conjugaison possibles du groupe de Galois de f comme le suggère J. McKay dans [69]. Il faut pour cela connaître une classification des longueurs de cycles possibles pour chaque classe de conjugaison de $\mathfrak{S}_n$. Une telle classification est fournie pour les sous-groupes transitifs jusqu'au degré 7 dans [69] et jusqu'au degré 11 dans [24].

Malheureusement, d'une part les bornes sur le nombre de premiers à tester pour voir apparaître toutes les partitions de n (ce problème est connu sous le nom de théorème de Čebotarev effectif) sont beaucoup trop grandes pour être utilisables, même en admettant l'hypothèse de Riemann généralisée (voir [58], [57] et [82]) ; d'autre part certains groupes de Galois demeurent impossibles à distinguer par cette seule méthode (voir [111, paragraphe 3] et [58]).

Ces longueurs de cycles permettent cependant de souvent reconnaître les groupes de Galois $\mathfrak{S}_n$ et $\mathfrak{A}_n$. Par exemple, un groupe de Galois contenant une transposition et un

$(n-1)$-cycle est le groupe symétrique $\mathfrak{S}_n$. J. H. DAVENPORT et G. C. SMITH [37] ont étudié la probabilité de reconnaître $\mathfrak{S}_n$ et $\mathfrak{A}_n$ en utilisant la k-transitivité.

En fait, les degrés des facteurs modulaires de f modulo différents premiers doivent tous être compatibles avec les degrés des facteurs de f sur $\mathbb{Z}[t]$.

Dans [79] et [80] D. R. MUSSER a introduit un test de compatibilité des degrés des facteurs dont nous allons faire un usage intensif.

Notation 5.25. Soit f un polynôme de degré n.

Le sous-ensemble de $\{0,\ldots,n\}$ des degrés possibles pour les facteurs (non nécessairement irréductibles) de f sur $\mathbb{Z}[t]$ est noté $D_0(f)$ et le sous-ensemble de $\{0,\ldots,n\}$ des degrés possibles pour les facteurs (non nécessairement irréductibles) de $\overline{f} = f \bmod p$ sur $\mathbb{F}_p[t]$ est noté $D_p(f)$.

Il est facile d'obtenir l'ensemble des degrés possibles à partir d'une partition du degré n du polynôme.

Notation 5.26. Soit $\varpi = (m_1,\ldots,m_n)$ une partition de n en degrés de facteurs irréductibles. L'ensemble des degrés possibles pour la partition ϖ, noté $D(\varpi)$ correspond à l'ensemble des sommes de toutes les parties de ϖ :

$$D(\varpi) = \left\{ \sum_{i=0}^{n} j_i\ i \;\middle|\; 0 \le j_i \le m_i \text{ pour } 1 \le i \le n \right\} \quad .$$

Notation 5.27. Soit D un sous-ensemble de $\{0,\ldots,n\} \subset \{0,\ldots,n+m\}$, $D+\{m\}$ désigne le sous-ensemble de $\{0,\ldots,n+m\}$ défini par :

$$D + \{m\} = \{i + m \mid i \in D\} \quad .$$

Si la partition ϖ correspond aux degrés répétés avec leur multiplicité $\{n_1,\ldots,n_r\}$ des r facteurs du polynôme alors $D(\varpi)$ se calcule par l'algorithme suivant :

Algorithme de Musser

Entrée : $\varpi = (m_1,\ldots,m_n) = \{n_1,\ldots,n_r\}$ une partition de n ; $\{n = \sum_{i=1}^{n} m_i\, i = \sum_{j=1}^{r} n_j\}$
Sortie : $D(\varpi) = D$, l'ensemble des degrés possibles pour ϖ.

(1) $D \leftarrow \{0\}$
(2) Pour i passant de 1 à r
$\quad D \leftarrow D \cup (D + \{n_i\})$

Ces sous-ensembles de $\{0,\ldots,n\}$ peuvent être représentés sous une forme très compacte : par des éléments de $\mathbb{F}_2^{n+1} = M(n+1)$ ou par l'écriture en binaire d'entiers compris entre 0 et $2^{n+1}-1$. Les coordonnées ou les chiffres égaux à 1 correspondant à la présence de l'élément dans le sous-ensemble D, l'union des ensembles correspond alors au « ou » logique et

l'opération $D+\{m\}$ devient un décalage de m coordonnées vers la droite ou une multiplication par 2^m.

Il est clair que $D_0(f) \subset D_p(f)$ pour tout premier p ne divisant pas le coefficient dominant de f. Donc, si $\{p_1, \ldots, p_k\}$ sont les k premiers pour lesquels a été obtenue la partition du degré de l'image de f modulo ces premiers en degrés de facteurs irréductibles alors $D_0(f) \subset D_{p_1}(f) \cap \cdots \cap D_{p_k}(f)$.

Si l'ensemble $D_{p_1}(f) \cap \cdots \cap D_{p_k}(f)$ est réduit à $\{0, n\}$, cela prouve l'irréductibilité de f sur $\mathbb{Z}[t]$.

Le test d'irréductibilité de Musser [79] consiste donc à initialiser un ensemble des degrés possibles noté $\mathcal{D}$ à la valeur $\mathcal{D} = \{0, \ldots, n\}$ avant de procéder à plusieurs factorisations modulo des premiers $p_1, \ldots, p_k$ ne divisant pas le discriminant de f. Ces différentes factorisations peuvent être effectuées dans le but de trouver un premier minimisant le nombre r de facteurs modulaires pour contenir autant que possible le coût de la réconciliation des facteurs modulaires sur $\mathbb{Z}[t]$ (exponentiel en r dans le pire des cas) ou pour tenter de reconnaître le groupe de Galois grâce à ses longueurs de cycles. Au passage, $\mathcal{D}$ est intersecté avec chacun des ensembles $D_{p_i}(f)$ trouvés. Si $\mathcal{D}$ se réduit à $\{0, n\}$ alors l'irréductibilité du polynôme f est prouvée. Mais même quand l'irréductibilité n'est pas prouvée, l'ensemble $\mathcal{D}$ fournit une information sur les degrés possibles des facteurs. Cette information peut être utilisée pour éviter des tentatives de réconciliation de facteurs modulaires inutiles : dans la méthode du paragraphe 5.4.3 un polynôme dont le degré n'appartient pas à l'ensemble $\mathcal{D}$ ne peut pas être un diviseur de f sur $\mathbb{Z}[t]$.

Cet ensemble des degrés possibles $\mathcal{D}$ va utiliser et fournir de nombreuses informations dans le cas de la factorisation de résolvantes.

Informations utilisées

Pour factoriser une résolvante dont les degrés des factorisations possibles sont parmi les partitions $\{\varpi_1, \ldots, \varpi_k\}$ qui correspondent à l'extrait d'une ligne de la matrice de partition définie en 1.14 pour les k classes de conjugaison restant à distinguer, l'ensemble $\mathcal{D}$ est initialisé non pas à la valeur $\{0, \ldots, n\}$ mais avec

$$\mathcal{D} \leftarrow \bigcup_{i=1}^{k} D(\varpi_i) \quad .$$

L'ensemble des degrés possibles pour les facteurs de la résolvante peut ainsi parfois être très réduit dès le début de la factorisation.

Informations recueillies

Dès qu'est modifié l'ensemble $\mathcal{D}$ au cours des diverses factorisations modulo différents premiers, les partitions qui contiennent des degrés n'appartenant plus à $\mathcal{D}$ peuvent être rejetées (ainsi que les classes candidates qui leur sont associées). Il est aussi possible de réintersecter $\mathcal{D}$ avec l'union des ensembles de degrés possibles des partitions restantes comme lors de l'initialisation (cela permet de « nettoyer » $\mathcal{D}$ des degrés possibles engendrés par les partitions rejetées).

Remarque : Le nombre de premiers différents à tester est traité par D. R. MUSSER dans [80]. Il considère que 5 premiers distincts suffisent (en moyenne) pour déterminer l'irréductibilité de polynômes de degré n choisis au hasard (et donc de groupe de Galois $\mathfrak{S}_n$ en général) pour $n \leq 200$. Cependant les résolvantes que nous cherchons à factoriser ne sont pas prises au hasard et ont d'autres groupes de Galois. Donc tirer, avec la même probabilité, des factorisations modulaires l'information maximale sur la transitivité du polynôme peut demander un bien plus grand nombre que 5 premiers distincts (voir [39]). Dans la pratique, le nombre de premiers distincts à utiliser va dépendre de l'efficacité des algorithmes de factorisation modulaire. S'ils sont très rapides, un grand nombre de factorisations modulaires peuvent être effectuées, même si à partir d'un certain point, elles ont une très faible probabilité de fournir une partition nouvelle. Par contre, lorsque le coût de la factorisation modulaire croît, leur nombre peut être limité voire réduit à la seule factorisation modulaire nécessaire pour continuer l'algorithme de factorisation sur les entiers.

En fait, la partition du degré de l'image de f modulo p par la factorisation en degrés distincts, notée $\varpi(f,p)$, est porteuse de plus d'informations que le seul ensemble des degrés possibles des facteurs qui s'en déduit par $\mathcal{D}_p(f) = D(\varpi(f,p))$.

Informations recueillies

La partition $\varpi(f,p)$ fournit une borne sur le nombre maximal des facteurs sur $\mathbb{Z}[t]$.

Des partitions de classes candidates pourront ainsi être rejetées parce qu'elles ont un trop grand nombre de parts, ce que n'aurait pas permis la seule comparaison des ensembles de degrés possibles engendrés par chacune d'elle.

Exemple : Soient deux partitions $\varpi_1 = (1^4)$ et $\varpi_2 = (1^2 2)$. Elles vérifient $D(\varpi_1) = D(\varpi_2) = \{0,1,2,3,4\}$ alors que la partition du degré 4 en ϖ_2 est incompatible avec une factorisation correspondant à ϖ_1 (ϖ_2 impose trois facteurs irréductibles au maximum).

En fait, plus généralement, il est possible de définir, pour une partition donnée ϖ l'ensemble des partitions compatibles avec elle.

Définition 5.28. Soit ϖ une partition de l'entier n dont les r parts (répétées avec leurs multiplicités) sont $\{n_1, \ldots, n_r\}$.

Soit « . » l'application, qui à un motif de partition non nul de longueur r, $m \in M(r)$ et à une partition ϖ de n associe la partition $m.\varpi$ de n dont les parts sont les n_i pour lesquels la i-ième coordonnée de m est égale à 0 ainsi que la somme des n_j pour lesquels la j-ième coordonnée est égale à 1.

Exemple : En notant au dessus des parts, la valeur de la coordonnée qui les affecte :

$$0001.\{\overset{0}{1},\overset{0}{1},\overset{0}{1},\overset{1}{2}\} = \{1,1,1,2\} \quad ,$$
$$1001.\{\overset{1}{1},\overset{0}{1},\overset{0}{1},\overset{1}{2}\} = \{1,1,3=2+1\} \quad ,$$
$$0101.\{\overset{0}{1},\overset{1}{1},\overset{0}{1},\overset{1}{2}\} = \{1,1,3=2+1\} \quad ,$$
$$0111.\{\overset{0}{1},\overset{1}{1},\overset{1}{1},\overset{1}{2}\} = \{1,4=2+1+1\} \quad .$$

Définition 5.29. L'*ensemble* $\mathcal{C}(\varpi)$ *des partitions compatibles* avec la partition ϖ de n en r parts $\varpi = \{n_1, \ldots, n_r\}$ est défini par :

$$\mathcal{C}(\varpi) = \{m.\varpi \mid m \in M(r)\} \quad .$$

Pour toute partition $\varpi(f,p)$ trouvée du degré de l'image de f modulo p l'ensemble des classes de conjugaison dont les partitions sont incompatibles avec $\varpi(f,p)$ peuvent être rejetées de l'ensemble des classes candidates pour contenir le groupe de Galois de f.

Cependant dans la pratique le coût combinatoire pour déterminer l'ensemble des partitions compatibles est beaucoup trop élevé dès que les parts sont nombreuses, et nous n'utilisons que l'ensemble des degrés possibles combiné avec la borne sur le nombre maximal de facteurs.

Le nombre total de facteurs irréductibles r (sur $\mathbb{Z}[t]$) correspond en fait à la moyenne du nombre de facteurs linéaires modulo les entiers premiers. Un encadrement de la convergence de cette moyenne vers r quand les premiers tendent vers l'infini a été fourni par P. WEINBERGER. Il donne ainsi dans [111] un algorithme en temps polynomial (avec l'hypothèse de Riemann généralisée) pour déterminer le nombre de facteurs irréductibles. Malheureusement, la taille des constantes, liées au théorème de Čebotarev effectif, ne rend pas l'algorithme praticable.

L'existence de facteurs linéaires

Un critère souvent utile pour la factorisation des polynômes (et la factorisation de résolvantes en particulier) est l'existence ou non de facteurs linéaires.

Si leur inexistence n'a pas été prouvée par le procédé de Musser exposé ci-dessus, les facteurs linéaires peuvent être cherchés par l'algorithme classique de recherche de racines rationnelles du polynôme de degré n, $f(t) = a_n t^n + \cdots + a_0$: les racines rationnelles de f sont recherchées sous la forme p/q où p et q sont premiers entre eux et appartiennent respectivement à l'ensemble des diviseurs (positifs ou négatifs) de a_0 et de a_n. Si p/q est une racine rationnelle de f, alors $qt - p$ est un facteur linéaire de f. Dans la pratique, l'ensemble des tests à effectuer peut être très élevé, et cette méthode n'est utilisable que si le nombre des diviseurs respectifs de a_0 et a_n est faible.

Un résultat attribué à D. H. LEHMER par J. BRILLHART dans [19] peut être signalé :

Théorème 5.30. *Soit n un entier supérieur ou égal à 2 et f un polynôme de degré n,*

$$f(t) = a_n t^n + \cdots + a_0 \quad .$$

Si a_n, a_0 et $f(1)$ sont impairs alors f n'a pas de racines rationnelles.

Preuve : Voir [19]. □

Informations recueillies

> Prouver avec le théorème 5.30 que la résolvante f n'a aucun facteur linéaire permet de retirer le degré 1 de l'ensemble $\mathcal{D}$ des degrés possibles et d'extraire de la liste des classes de conjugaison candidates celles dont la partition du degré de la résolvante f contient la part 1.

La recherche spécifique des facteurs linéaires sur $\mathbb{Z}[t]$ à partir de factorisations modulaires suivies d'un lemme de Hensel effectif a été explorée par R. LOOS (voir [66]).

Le problème inverse (trouver un premier p pour lequel l'image de f modulo p admet une racine simple) pour en tirer une méthode de factorisation a été exploré par G. VIRY dans [103, chapitre IV].

5.5.2 Bornes sur les coefficients des facteurs et reformulation

Les étapes de remontée par un lemme de Hensel effectif (décrit au paragraphe 5.4.2) et de détection des facteurs irréductibles (esquissée au paragraphe 5.4.3) sont les plus coûteuses du schéma de Berlekamp-Zassenhaus.

La taille du modulo m jusqu'auquel les facteurs vont être remontés par un lemme de Hensel effectif dépend à la fois de la qualité de la borne connue sur les coefficients des facteurs du polynôme f sur $\mathbb{Z}[t]$ ainsi que de la méthode retenue pour retrouver les coefficients entiers des facteurs possibles à partir de leurs images modulo m.

Définition 5.31. Soit $f \in \mathbb{Z}[t]$ un polynôme à coefficients entiers. Un *vrai facteur* de f est un facteur de f sur $\mathbb{Z}[t]$ (non nécessairement irréductible).

Soit r le nombre de facteurs irréductibles de f modulo le premier p choisi (au paragraphe 5.5.1) pour la factorisation modulaire, et par conséquence, le nombre de facteurs modulo m où m est le modulo atteint dans la remontée par un lemme de Hensel effectif des facteurs de f modulo p (voir le paragraphe 5.4.2).

$$f \equiv f_1 \cdots f_r \bmod m$$

et

$$f \equiv \overline{f}_1 \cdots \overline{f}_r \bmod p$$

avec, pour $1 \leq i \leq r$,

$$\overline{f}_i \equiv f_i \bmod p \quad .$$

Le paragraphe 5.4.3 a rappelé la méthode (exponentielle dans le pire des cas) qui consiste à d'abord essayer les facteurs f_i modulo m comme candidats pour être de vrais facteurs puis leurs produits deux à deux, trois à trois et ainsi de suite jusqu'à épuisement de toutes les combinaisons possibles.

Candidats pour être vrais facteurs

Pour trouver les vrais facteurs sur $\mathbb{Z}[t]$, à partir des r facteurs de f modulo m, nous avons besoin de méthodes produisant des polynômes $g \in \mathbb{Z}[t]$ à partir de produits de ℓ-uplets de facteurs de f modulo m, pour $1 \leq \ell \leq r$ (ils correspondent à toutes les parties possibles de l'ensemble $\{1, \ldots, r\}$). Ces polynômes g sont considérés comme candidats pour être de vrais facteurs de f. Ils sont de vrais facteurs de f lorsque la division de f par g est exacte. Sinon, ils sont rejetés lorsque la division n'est pas exacte ou par tout autre critère plus facile à calculer que la division des polynômes (incompatibilité du degré avec l'ensemble des degrés possibles défini au paragraphe 5.5.1, compatibilité de l'évaluation en 0 ou 1, si la taille des coefficients reconstitués est supérieure à la borne sur ces coefficients...).

Il existe deux méthodes classiques pour calculer le candidat pour être vrai facteur de f associé au produit c d'un ℓ-uplet ($1 \leq \ell \leq r-1$) de facteurs modulaires dans $\mathbb{Z}/m\mathbb{Z}[t]$.

Reconstruction de rationnels [109] : Soit $u \in \mathbb{Z}/m\mathbb{Z}[t]$ le polynôme

$$u = \mathrm{cd}\,(c)^{-1}\, c \bmod m \in \mathbb{Z}/m\mathbb{Z}[t] \quad ;$$

soit $v \in \mathbb{Q}[t]$ l'unique polynôme, quand il existe, à coefficients rationnels dont les numérateurs et dénominateurs sont bornés par $\sqrt{m/2}$ et égaux à ceux de u modulo m. Soit $g \in \mathbb{Z}[t]$ le polynôme à coefficients entiers égal à v multiplié par le ppcm des dénominateurs des coefficients de v (les dénominateurs sont ainsi supprimés). Le polynôme g est pris comme candidat pour être un vrai facteur de f dans $\mathbb{Z}[t]$.

Restes symétriques : Soit $w \in \mathbb{Z}/m\mathbb{Z}[t]$ le polynôme

$$w = \mathrm{cd}\,(f)\,\mathrm{cd}\,(c)^{-1}\, c \bmod m \in \mathbb{Z}/m\mathbb{Z}[t]$$

dont tous les coefficients sont représentés par des entiers entre $-m/2$ et $m/2$. Soit $g \in \mathbb{Z}[t]$ la partie primitive de w sur $\mathbb{Z}[t]$. Le polynôme g est pris comme candidat pour être un vrai facteur de f dans $\mathbb{Z}[t]$.

Définition 5.32. Soient n un entier positif et $f \in \mathbb{Z}[t]$ un polynôme de degré n à coefficients entiers,

$$f(t) = a_n t^n + \cdots + a_1 t + a_0 \quad .$$

La *hauteur* de f, notée $\mathrm{H}\,(f)$ est le maximum des valeurs absolues de ses coefficients :

$$\mathrm{H}\,(f) = \max_{0 \leq i \leq n} |a_i| \quad .$$

Soit s la hauteur du vrai facteur, noté $\widehat{f}$, que nous cherchons. La méthode de reconstruction de rationnels va le reconstituer dès que

$$m > 2s^2 \quad , \tag{5.2}$$

(voir [109] et [32]) et la méthode des restes symétriques peut être utilisée dès que

$$m > 2\mathrm{cd}\,(f)\ s \quad . \tag{5.3}$$

Nous notons $C(m,s)$ une des conditions (5.2) ou (5.3) selon la méthode retenue pour produire des candidats pour être vrais facteurs.

Comme la hauteur s n'est généralement pas connue *a priori* elle est remplacée par des bornes calculées à partir des degrés possibles des facteurs de f et de leur nombre.

Bornes sur les coefficients des facteurs

Il existe deux type de telles bornes.

Borne sur tous les facteurs : Une telle borne, que nous notons $B(f)$, garantit que chaque coefficient de tout facteur irréductible de f est inférieur à $B(f)$.

Borne sur un seul facteur : Une telle borne, que nous notons $b(f)$, garantit que chaque coefficient d'au moins un des facteurs non triviaux (non nécessairement irréductible) de f est inférieur à $b(f)$.

Les bornes sur tous les facteurs $B(f)$ sont classiques et peuvent être déduites de bornes sur les racines de f. Des exemples de telles bornes sont donnés dans [71], [72], [54], [73], [8], [15] et [1].

Des formules de bornes sur un seul facteur sont données dans [9], [16] et [75]. Certaines situations où les polynômes ont leur hauteur pour borne sur un seul facteur ont été explorées par Z. Liu et P. S. Wang dans [64] et [65].

Informations utilisées

> Certaines formules de bornes sont plus petites dans le cas où le nombre minimum de facteurs est connu et supérieur à 2. C'est le cas de la borne [9, théorème 1 (a)].

Les bornes sur un seul facteur ont été étudiées car elles peuvent être beaucoup plus petites que les bornes sur tous les facteurs. La principale différence lors de leur utilisation est l'étape de combinaison des facteurs modulaires remontés modulo m par un lemme de Hensel effectif.

Si m satisfait la condition $C(m,B(f))$ (s étant majoré par $B(f)$), les combinaisons ne doivent être tentées que jusqu'à des $\lfloor r/2 \rfloor$-uplets de facteurs modulaires car le facteur associé à un $\lfloor r/2 \rfloor$-uplet tout autant que son complément associé à un $(r - \lfloor r/2 \rfloor)$-uplet sont des vrais facteurs. Si m ne satisfait que la condition $C(m,b(f))$, les combinaisons doivent être tentées jusqu'aux $(r-1)$-uplets de facteurs modulaires car le vrai facteur associé à un certain $(r-1)$-uplet peut être le seul à avoir ses coefficients plus petits que $b(f)$.

Détection précoce de vrais facteurs

Malheureusement, dans les applications courantes, ces bornes, qui ont été parfois prouvées optimales car atteintes par certaines classes de polynômes, se révèlent souvent beaucoup trop pessimistes. La conséquence est que la remontée par un lemme de Hensel effectif est poussée jusqu'à un modulo beaucoup plus grand que ne le requiert vraiment le problème. Des méthodes de *détection précoce de vrais facteurs* ont donc été étudiées par WANG dans [109] et [110] ainsi que par [9] et [33].

Dans ces méthodes, les étapes des paragraphes 5.4.2 et 5.4.3 sont entrelacées pour des modulos m croissants. Les combinaisons de ℓ-uplets de facteurs modulaires sont tentées avant même que les conditions $C(m, b(f))$ et $C(m, B(f))$ soient remplies. Des heuristiques sur le modulo auquel les combinaisons doivent être essayées ont été proposées par [109] et [33].

La conséquence d'une détection précoce de vrais facteurs est que lorsqu'un vrai facteur de f, noté $\widehat{f}$, est détecté, son irréductibilité sur $\mathbb{Z}[t]$ peut ne pas être prouvée.

Plus précisément, nous supposons que le polynôme $\widehat{f}$ est associé à un certain ℓ-uplet de facteurs modulaires sur $\mathbb{Z}/m\mathbb{Z}[t]$. Le polynôme $\widehat{f}$ peut être facilement prouvé irréductible dans certains cas spécifiques : quand ℓ est égal à 1 ; si la restriction de l'ensemble des degrés possibles $\mathcal{D}$ du paragraphe 5.5.1 aux degrés plus petits que celui de $\widehat{f}$ est réduit à l'ensemble vide ; et parfois en utilisant les critères d'irréductibilité faciles à calculer exposés au paragraphe 5.3.2 ; sinon, $b(\widehat{f})$ et $B(\widehat{f})$ sont calculées.

Si l'une ou l'autre des conditions $C(m, b(\widehat{f}))$ ou $C(m, B(\widehat{f}))$ est remplie alors $\widehat{f}$ est irréductible, car les vrais facteurs de $\widehat{f}$ auraient été trouvés comme vrais facteurs de f associés à un ℓ'-uplet pour $\ell' < \ell$ à la condition que tous les candidats vrais facteurs associés à de tels ℓ'-uplets aient déjà été rejetés.

Si aucune des conditions $C(m, b(\widehat{f}))$ ni $C(m, B(\widehat{f}))$ ne sont remplies alors il est nécessaire de poursuivre le processus de remontée/combinaison des facteurs modulaires de $\widehat{f}$ jusqu'à ce que des vrais facteurs de $\widehat{f}$ soient trouvés ou qu'un modulo m' plus grand que m soit atteint tel qu'il remplisse la condition $C(m', b(\widehat{f}))$ ou $C(m', B(\widehat{f}))$.

Si les ℓ'-uplets de facteurs modulaires de $\widehat{f}$ n'ont pas été testés alors que la condition $C(m, b(\widehat{f}))$ ou $C(m, B(\widehat{f}))$ est remplie le processus de combinaison des facteurs modulaires de $\widehat{f}$ est poursuivi jusqu'à ce que les vrais facteurs de $\widehat{f}$ soient trouvés ou que tous les candidats pour être vrais facteurs soient rejetés. Ce dernier cas (que tous les ℓ'-uplets n'aient pas été testés) arrive particulièrement pour le facteur complément $\overline{f} = f/\widehat{f}$ qui est associé au $(r-\ell)$-uplet restant de facteurs modulaires modulo m de f.

Il est important de remarquer que chaque fois qu'un vrai facteur $\widehat{f}$ et son complément $\overline{f}$ sont trouvés, même s'il n'est pas facilement montré qu'ils sont irréductibles, le nombre exponentiel de combinaisons restant à tester est fortement réduit (de la forme 2^ℓ et $2^{r-\ell}$ dans le pire des cas au lieu de 2^r).

Si pour $\widehat{f}$ (ou $\overline{f}$) le processus de remontée/combinaison a besoin d'être poursuivi (aucune des conditions $C(m, b(\widehat{f}))$ ni $C(m, B(\widehat{f}))$ ne sont remplies) alors le ℓ-uplet de facteurs modulaires de m doit être remonté par un lemme de Hensel effectif jusqu'à un modulo m'

supérieur à m. Nous pouvons :

1. prendre les images du ℓ-uplet modulo p issues de l'étape 5.4.1 et les remonter jusqu'au modulo m' ;

2. prendre leurs images connues modulo m et les remonter jusqu'au modulo m'.

Comme la remontée de Hensel effective utilise des polynômes secondaires (polynômes de base de remontée, polynômes de correction, voir le chapitre 6) la première méthode signifie que les polynômes secondaires utilisés dans la remontée de Hensel effective doivent être recalculés complètement au cours du nouveau processus de remontée, alors que dans la seconde méthode, nous souhaitons calculer ces polynômes secondaires à partir de ceux utilisés pour la remontée des r facteurs modulaires de f.

Définition 5.33. La *reformulation du problème de remontée de Hensel effective* est l'ensemble des calculs qui donnent les polynômes nécessaires à la poursuite d'une remontée de Hensel effective sur les seuls ℓ facteurs d'un ℓ-uplet fixé des r facteurs modulaires de f modulo m. Ces calculs utilisent comme paramètres les polynômes déjà connus pour les r facteurs modulaires de f modulo m ainsi que le vrai facteur $\widehat{f}$ de f, associé au ℓ-uplet.

Le paragraphe 6.2 expose la reformulation du problème de remontée dans la technique de lemme de Hensel effectif proposée par G. E. COLLINS et M. J. ENCARNACIÓN.

5.5.3 Énumération des combinaisons possibles

Lors des tentatives de combinaisons des r facteurs modulaires de f modulo m, nous souhaitons parcourir l'ensemble des ℓ-uplets possibles. Il est plus intéressant, d'après [31], de les parcourir pour ℓ croissant à partir de 1 plutôt qu'à degré fixé croissant : la complexité des combinaisons est **en moyenne** polynomiale.

De plus, dans le cas où la condition $C(m, B(f))$ est vérifiée, si tous les candidats vrais facteurs associés à des ℓ'-uplets, pour $\ell' \leq \ell$, ont été rejetés alors f n'admet aucun facteur de degré inférieur au minimum des sommes ℓ par ℓ des degrés de facteurs modulaires distincts. L'ensemble $\mathcal{D}$ des degrés possibles pour f est modifié en supprimant tous les degrés inférieurs à ce minimum.

Informations recueillies

En modifiant l'ensemble $\mathcal{D}$ au cours des combinaisons, sont exclues, le cas échéant, de la liste des partitions possibles celles qui possèdent des degrés plus petits que le minimum de $\mathcal{D}$.

Par ailleurs, dès qu'est trouvé un facteur non trivial de f sur $\mathbb{Z}[t]$, peuvent aussi être exclues de la liste des partitions celles qui sont incompatibles avec la présence de ce facteur. En pratique en raison du coût combinatoire du calcul de toutes les partitions compatibles, ce critère n'est utilisé que pour les polynômes irréductibles : leurs degrés doivent forcément apparaître dans les partitions compatibles.

5.6 La factorisation partielle interactive

Nous avons vu dans ce chapitre comment différentes étapes de l'algorithme de factorisation pouvaient fournir des informations discriminantes entre les différentes partitions ou groupes de Galois possibles (rassemblées sous l'intitulé **informations recueillies**). Cependant, comme nous ignorons *a priori* à quel moment elles vont apparaître, parfois pas avant la factorisation finale, il est utile d'avoir ces informations disponibles à tout moment pour le programme de recherche du groupe de Galois qui fait appel à la factorisation.

Comme signalé dans [35, chapitre 4, page 158] l'utilisation d'informations aux origines hétérogènes en cours de factorisation et pouvant en modifier le cours complique l'écriture des algorithmes et plus encore leur implantation.

La solution que nous avons retenue est de rendre disponibles toutes les informations trouvées à tout moment de la factorisation en les gardant dans une structure de données qui contient le polynôme à factoriser. La fonction de factorisation effectue *une étape supplémentaire* dans le processus de factorisation. La nature de cette étape élémentaire n'est pas précisément définie. Ce qui compte, c'est que des informations nouvelles aient été déposées dans la structure de données. Et c'est la structure de données qui est interrogée pour obtenir les informations cherchées.

En fait, pour les opérations de remontée par un lemme de Hensel effectif et des réconciliations, il n'est pas nécessaire que les polynômes modulaires soient irréductibles mais simplement premiers entre eux. Notre factorisation peut donc être limitée aux seuls polynômes susceptibles d'apporter une information discriminante et de laisser inachevée la factorisation des autres facteurs (en les laissant sous forme de produits de facteurs inintéressants).

Pour cela, la *factorisation partielle interactive* repose sur le principe suivant : pour toute factorisation non triviale trouvée, deux nouvelles structures de données sont créées, elles correspondent à un facteur non trivial et à son complément sur lesquels le processus de factorisation pourra être poursuivi si besoin. Chaque factorisation non triviale peut être vue comme un arbre dont ne seront parcourues que les branches qui sont susceptibles de fournir une information discriminante.

Il importe donc d'être capable de pouvoir toujours reformuler les problèmes en cours de traitement et réinitialiser les variables de façon cohérente sur chacun des facteurs.

Le factorisateur est actuellement conçu pour poursuivre la factorisation sur les facteurs non-triviaux dans les cas suivants :

- si le polynôme est susceptible d'avoir des facteurs irréductibles dont les degrés appartiennent à un ensemble de degrés cherchés fixé d'avance (par exemple, tous les facteurs de degré plus petit que 13, ou tous les facteurs dont le degré est multiple de 3, ou tous les facteurs de degrés 2 et 7) ;

- (dans le cas où le premier p modulo lequel les calculs modulaires sont effectués est imposé) si le polynôme est susceptible d'avoir des facteurs irréductibles dont les degrés de leur factorisation en facteurs irréductibles modulo p appartiennent à un ensemble de degrés cherchés fixé d'avance ;

– (dans le cas où le premier p modulo lequel les calculs modulaires sont effectués est imposé) si le polynôme est susceptible d'avoir des facteurs irréductibles dont les partitions des degrés en facteurs irréductibles modulo p appartiennent à un ensemble de partitions cherchées fixées d'avance.

Ces critères sont ceux choisis par défaut mais ils peuvent être modifiés.

Une implantation en `Axiom` de cette façon de considérer la factorisation est détaillée au chapitre 7.

Chapitre 6

Outils de la factorisation interactive

Ce chapitre est destiné à introduire des outils pour améliorer l'étape de reformulation du problème de remontée décrite au paragraphe 5.5.2 du chapitre 5 lors de la factorisation d'un polynôme à coefficients entiers.

Le paragraphe 6.1 définit un ordre sur l'énumération des ℓ-uplets de facteurs modulaires. Pour cela un motif de partition, défini au paragraphe 2.4, est associé à chaque ℓ-uplet de facteurs. Nous associons à l'ordre défini sur les motifs de partition une fonction *successeur* qui permet d'itérer sur l'ensemble des combinaisons possibles. L'ordre est construit pour éviter d'effectuer deux fois les mêmes calculs dans un processus de détection précoce de vrais facteurs (lorsque les facteurs trouvés ne sont pas nécessairement irréductibles).

Le paragraphe 6.2 rappelle la méthode de remontée « à la Hensel » de Collins et Encarnación et fournit une reformulation du problème de remontée dans cette méthode. Notre reformulation permet d'inscrire efficacement la méthode Collins et Encarnación dans le processus de factorisation interactive. L'ordre du paragraphe 6.1 est utilisé au paragraphe 6.2.3.

6.1 Un ordre sur les motifs de partition

Ce paragraphe a pour but de fournir un ordre sur les motifs de partition définis au paragraphe 2.4. Il est utilisé par la procédure de réconciliation des facteurs modulaires décrite aux paragraphes 5.5.2 et 6.2.3 pour éviter de faire deux fois les mêmes calculs et les mêmes tests.

Les preuves de ce paragraphe nous paraissent beaucoup trop techniques par rapport à ce que semble devoir requérir la nature du problème. Nous invitons le lecteur à ne retenir que les énoncés et tout particulièrement la définition 6.24 et les propositions 6.26 et 6.27.

6.1.1 Définitions

Soit r un entier, $r \geq 1$, pour $1 \leq k \leq r$ la k-ième coordonnée d'un motif de partition m de longueur r, défini au paragraphe 2.4, est notée m_k :

$$m = m_1 m_2 \dots m_{r-1} m_r \in M(r) \quad .$$

Définition 6.1. Pour des entiers i et j, si $j > 1$ ou $i < r$, alors $\text{bloc}_r(i,j)$ est le motif nul de $M(r)$ dont toutes les coordonnées sont égales à 0.

Sinon, lorsque i et j vérifient $1 \leq i \leq j \leq r$, la fonction $\text{bloc}_r(i,j)$ est le motif de partition de $M(r)$ avec des 1 pour ses coordonnées entre i et j, bornes comprises, et des 0 ailleurs :

$$\text{bloc}_r(i,j) = \underbrace{0\ldots0}_{i-1}\underbrace{1\ldots1}_{j-i+1}\underbrace{0\ldots0}_{r-j} \quad .$$

Convention : Par la suite, la convention suivante sera adoptée :

$$\max \varnothing = 0 \quad .$$

Définition 6.2. Le *sous-niveau* d'un motif de partition de longueur r est le nombre de ses premières coordonnées consécutives égales à 0 :

$$\text{sous-niveau}(m_1 \ldots m_r) = \max\{k \in [1,r] \mid m_1 = \ldots = m_{k-1} = m_k = 0\} \quad .$$

Définition 6.3. Le *bout* d'un motif de partition de longueur r est le plus grand indice de coordonnée de m égal à 1 :

$$\text{bout}(m_1 \ldots m_r) = \max\{k \in [1,r] \mid m_k = 1\} \quad .$$

Définition 6.4. Pour $k \in \mathbb{Z}$ un entier fixé, $k \leq r$ et m un motif de partition de longueur r, l'*extrait* des k premières coordonnées d'un motif de partition est une application de $M(r) \times \mathbb{Z}$ dans $M(r)$ définie par :

$$\text{extrait}_r(m_1 \ldots m_r, k) = m_1 m_2 \ldots m_k \underbrace{0\ldots0}_{r-k} \quad .$$

Si l'entier k est strictement inférieur à 1, alors $\text{extrait}_r(m,k)$ est le motif nul.

Remarque : $\text{extrait}_r(m, \text{bout}(m)) = m$.

Propriété 6.5. *Les fonctions* sous-niveau, bout *et* extrait *vérifient pour tout* $m \in M(r)$ *et tout entier* $k \leq r$ *:*

$$\text{bout}(\text{extrait}_r(m,k)) \leq k \tag{6.1}$$

$$\textit{et} \qquad \text{sous-niveau}(\text{extrait}_r(m,k)) \geq \text{sous-niveau}(m) \quad . \tag{6.2}$$

Définition 6.6. Le *complément* $\complement_r$ d'un motif de partition de longueur r est l'application qui consiste à échanger les 1 et les 0 de ses coordonnées :

$$\complement_r(m_1 \ldots m_r) = (1-m_1)(1-m_2)\ldots(1-m_{r-1})(1-m_r) \quad .$$

Propriété 6.7. *Pour tout* $m \in M(r)$ *et tout* $k \leq r$*, nous avons :*

$$\text{extrait}_r\left(\complement_r(\text{extrait}_r(m,k)), k\right) = \text{extrait}_r\left(\complement_r(m), k\right) \quad .$$

Définition 6.8. Notons $M_P(r)$ et $M_S(r)$ les ensembles définis par

$$M_P(r) = \bigcup_{\ell=0}^{r-1} M(\ell, r) \quad \text{et} \quad M_S(r) = \bigcup_{\ell=1}^{r} M(\ell, r) \quad ,$$

où $M(\ell, r)$ a été défini au paragraphe 2.4

Propriété 6.9. *L'application* $\complement_r$ *réalise une bijection sur* $M(r)$ *ainsi que de* $M_P(r)$ *sur* $M_S(r)$.

Définition 6.10. Définissons les fonctions d_0, g_0, d_1, g_1, respectivement, zéro de droite, zéro de gauche, un de droite, un de gauche, par :

$$\begin{array}{rccl} \mathrm{d}_0 \ : & M_P(r) & \longrightarrow & \{1, \ldots, r\} \\ & m & \longmapsto & \mathrm{bout}\left(\complement_r(m)\right) \quad ; \end{array}$$

$$\begin{array}{rccl} \mathrm{g}_0 \ : & M_P(r) & \longrightarrow & \{1, \ldots, r\} \\ & m & \longmapsto & \mathrm{bout}\left(\mathrm{extrait}_r\left(m, \mathrm{d}_0(m)\right)\right) + 1 \quad ; \end{array}$$

$$\begin{array}{rccl} \mathrm{d}_1 \ : & M_S(r) & \longrightarrow & \{1, \ldots, r\} \\ & m & \longmapsto & \mathrm{bout}(m) \quad ; \end{array}$$

$$\begin{array}{rccl} \mathrm{g}_1 \ : & M_S(r) & \longrightarrow & \{1, \ldots, r\} \\ & m & \longmapsto & \mathrm{bout}\left(\mathrm{extrait}_r\left(\complement_r(m), \mathrm{d}_1(m)\right)\right) + 1 \quad . \end{array}$$

Propriété 6.11. *Les applications* d_0, g_0, d_1, g_1 *vérifient pour tout* $m \in M_P(r)$ *:*

$$\mathrm{d}_0(m) = \mathrm{d}_1\left(\complement_r(m)\right) \quad ; \tag{6.3}$$

$$\mathrm{g}_0(m) = \mathrm{g}_1\left(\complement_r(m)\right) \quad ; \tag{6.4}$$

$$\mathrm{g}_0(m) \leq \mathrm{d}_0(m) \quad . \tag{6.5}$$

Preuve : Les propriétés (6.3) et (6.4) découlent directement des définitions.

Pour (6.5), en posant,

$$m = m_1 m_2 \ldots m_{r-1} m_r \quad ,$$

par définition de d_0, nous avons $(1 - m_{\mathrm{d}_0(m)}) = 1$ avec $\mathrm{d}_0(m) \geq 1$ car comme $m \in M_P(r)$, par la propriété 6.9, son complément appartient à $M_S(r)$ et n'est pas le motif nul.

Donc $m_{\mathrm{d}_0(m)} = 0$ et $\mathrm{bout}\left(\mathrm{extrait}_r\left(m, \mathrm{d}_0(m)\right)\right) < \mathrm{d}_0(m)$, d'où l'inégalité. □

Propriété 6.12. *Tout motif de partition* $m \in M_S(r)$ *peut s'écrire sous la forme :*

$$m = \mathrm{extrait}_r\left(m, \mathrm{g}_1(m) - 2\right) + \mathrm{bloc}_r\left(\mathrm{g}_1(m), \mathrm{d}_1(m)\right) \quad .$$

Preuve : En posant

$$m = m_1 m_2 \dots m_{r-1} m_r \quad ,$$

d'après la définition de d_1, $m_k = 0$ pour $k > \mathrm{d}_1(m)$; d'après la définition de g_1, $(1 - m_k) = 0$ pour $\mathrm{g}_1(m) \leq k \leq \mathrm{d}_1(m)$ et $(1 - m_{\mathrm{g}_1(m)-1}) = 1$, d'où la propriété. □

Propriété 6.13. *Tout motif de partition $m \in M_P(r)$ peut s'écrire sous la forme :*

$$\begin{aligned} m = {} & \mathrm{extrait}_r(m, \mathrm{g}_0(m) - 2) + \mathrm{bloc}_r(\mathrm{g}_0(m) - 1, \mathrm{g}_0(m) - 1) \\ & + \mathrm{bloc}_r(\mathrm{d}_0(m) + 1, r) \quad . \end{aligned}$$

Preuve : En appliquant la propriété 6.12 au complément de m, on obtient :

$$\complement_r(m) = \mathrm{extrait}_r\left(\complement_r(m), \mathrm{g}_0(m) - 2\right) + \mathrm{bloc}_r(\mathrm{g}_0(m), \mathrm{d}_0(m)) \quad .$$

La formule s'obtient en prenant le complément de cette égalité. □

Exemples : Pour $r = 16$:

m	niveau	sous-niveau	g_1	d_1	g_0	d_0
0001001001111000	6	3	10	13	14	16
1110110110000111	10	0	14	16	10	13
1111111110000000	9	0	1	9	10	16
0000000001111111	8	9	10	16	1	9

Nous utilisons les fonctions définies dans ce paragraphe pour définir un ordre sur les motifs de partition.

6.1.2 Définition d'un ordre sur les motifs de partition

Nous définissons dans ce paragraphe une application successeur sur les motifs de partition. Le paragraphe 6.1.3 établit que cette application est surjective.

Définition 6.14. L'application *successeur* de $M_P(r)$ dans $M_S(r)$ est définie par :

$$\begin{array}{rccl} \mathrm{succ}_r : & M_P(r) & \longrightarrow & M_S(r) \\ & m & \longmapsto & \mathrm{extrait}_r(m, \mathrm{g}_0(m) - 2) + \mathrm{bloc}_r(\mathrm{g}_0(m), r - \mathrm{d}_0(m) + \mathrm{g}_0(m)) \quad . \end{array}$$

Remarque : L'application succ_r est bien définie à image dans $M_S(r)$, car de la définition 6.10 des fonctions g_0 et d_0 ainsi que de l'inégalité 6.5 de la propriété 6.11 nous tirons $1 \leq \mathrm{g}_0(m) \leq r$ et $\mathrm{g}_0(m) \leq r - \mathrm{d}_0(m) + \mathrm{g}_0(m) \leq r$. Le bloc $\mathrm{bloc}_r(\mathrm{g}_0(m), r - \mathrm{d}_0(m) + \mathrm{g}_0(m))$ n'est donc jamais nul.

Lemme 6.15. *L'application composée* $\mathrm{succ}_r \circ \complement_r$ *réalise une involution de* $M_S(r)$.

Preuve : D'après les égalités (6.3) et (6.4) de la propriété 6.11,

$$\mathrm{succ}_r\left(\complement_r(m)\right) = \mathrm{extrait}_r\left(\complement_r(m), \mathrm{g}_1(m) - 2\right) + \mathrm{bloc}_r(\mathrm{g}_1(m), r - \mathrm{d}_1(m) + \mathrm{g}_1(m)) \quad .$$

En passant au complément, et d'après la propriété 6.7, en posant

$$\begin{aligned} m' &= \complement_r\left(\mathrm{succ}_r\left(\complement_r(m)\right)\right) \\ &= \mathrm{extrait}_r(m, \mathrm{g}_1(m) - 2) + \mathrm{bloc}_r(\mathrm{g}_1(m) - 1, \mathrm{g}_1(m) - 1) \\ &\quad + \mathrm{bloc}_r(r - \mathrm{d}_1(m) + \mathrm{g}_1(m) + 1, r) \quad . \end{aligned}$$

nous obtenons

$$\mathrm{g}_0(m') = \mathrm{g}_1(m)$$

et

$$\mathrm{d}_0(m') = r - \mathrm{d}_1(m) + \mathrm{g}_1(m) \quad .$$

D'où,

$$\begin{aligned} \mathrm{succ}_r(m') &= \mathrm{extrait}_r(m', \mathrm{g}_0(m') - 2) + \mathrm{bloc}_r(\mathrm{g}_0(m'), r - \mathrm{d}_0(m') + \mathrm{g}_0(m')) \\ &= \mathrm{extrait}_r(m', \mathrm{g}_1(m) - 2) + \mathrm{bloc}_r(\mathrm{g}_1(m), \mathrm{d}_1(m)) \\ &= \mathrm{extrait}_r(m, \mathrm{g}_1(m) - 2) + \mathrm{bloc}_r(\mathrm{g}_1(m), \mathrm{d}_1(m)) \\ &= m \end{aligned}$$

d'après la propriété 6.12. □

Ce lemme a pour conséquence :

Proposition 6.16. *L'application* succ_r *réalise une bijection de* $M_P(r)$ *sur* $M_S(r)$ *et a pour inverse l'application* $\mathrm{préd}_r$ *définie par*

$$\begin{array}{rccl} \mathrm{préd}_r & : & M_S(r) & \longrightarrow & M_P(r) \\ & & m & \longmapsto & \complement_r\left(\mathrm{succ}_r\left(\complement_r(m)\right)\right) \ . \end{array}$$

6.1.3 Propriétés de l'application succ_r

Ce paragraphe a pour but de montrer que l'application succ_r est bien nommée, c'est à dire que tout élément de $M_S(r)$ est atteint par composition itérée de cette application au motif de partition nul.

Nous montrons cette propriété par une récurrence pour laquelle il est nécessaire d'introduire de nouvelles notations.

Définition 6.17. Soit r un entier supérieur ou égal à 2. L'application de $M(r)$ dans $M(r-1)$ qui associe à $m \in M(r)$ le motif de partition de $M(r-1)$ composé des $r-1$ dernières coordonnées de m est notée $\curvearrowleft$:

$$\begin{array}{rccc} \curvearrowleft : & M(r) & \longrightarrow & M(r-1) \\ & m = m_1 m_2 \dots m_r & \longmapsto & \curvearrowleft(m) = m_2 \dots m_r \end{array}$$

L'application de $M(r-1)$ dans $M(r)$ qui associe à $m \in M(r)$ le motif de partition de $M(r)$ dont les $r-1$ dernières coordonnées sont celles de m et dont la première est 0 est notée $\curvearrowright$:

$$\begin{array}{rccc} \curvearrowright : & M(r) & \longrightarrow & M(r-1) \\ & m = m_1 \dots m_{r-1} & \longmapsto & \curvearrowright(m) = 0 m_1 \dots m_{r-1} \end{array}$$

Propriété 6.18. *Soit r un entier supérieur ou égal à 2.*

Pour tout $m \in M(r-1)$:

$$\curvearrowleft(\curvearrowright(m)) = m$$

et pour tout $m \in M(r)$:

$$m = \mathrm{extrait}_r(m,1) + \curvearrowright(\curvearrowleft(m))$$

ainsi que, pour $k \leq r$,

$$\mathrm{extrait}_r(m,k) = \mathrm{extrait}_r(m,1) + \curvearrowright(\mathrm{extrait}_{r-1}(\curvearrowleft(m), k-1)) \quad . \tag{6.6}$$

Pour tout $m \in M(r)$:

$$\mathbb{C}_{r-1}(\curvearrowleft(m)) = \curvearrowleft(\mathbb{C}_r(m)) \quad .$$

Pour tout $m \in M_P(r)$:

$$\begin{array}{c} \textit{Si } \mathrm{d}_0(m) \geq 2, \textit{ alors } \mathrm{d}_0(\curvearrowleft(m)) = \mathrm{d}_0(m) - 1 \textit{ ;} \\ \textit{Si } \mathrm{g}_0(m) \geq 2, \textit{ alors } \mathrm{g}_0(\curvearrowleft(m)) = \mathrm{g}_0(m) - 1. \end{array}$$

Et pour tout $m \in M_P(r-1)$:

$$\mathrm{d}_0(\curvearrowright(m)) = \mathrm{d}_0(m) + 1 \qquad \textit{et} \qquad \mathrm{g}_0(\curvearrowright(m)) = \mathrm{g}_0(m) + 1 \quad .$$

Preuve : Ces propriétés se déduisent directement des définitions 6.17, 6.4, 6.6 et 6.10. □

Définition 6.19. Soient ℓ et r des entiers vérifiant $0 \leq \ell \leq r$. L'ensemble $M_P(\ell, r)$ est défini par :

$$M_P(\ell, r) = M(\ell, r) \setminus \mathrm{bloc}_r\,(r - \ell + 1, r) \quad .$$

Propriété 6.20. *Soient un entier $r \geq 3$, $m \in M_P(r)$.*
Si $\mathrm{\ulcorner}\,(m) \in M_P(\mathrm{niveau}\,(\mathrm{\ulcorner}\,(m)), r-1)$ alors :

$$\mathrm{succ}_r\,(m) = \mathrm{extrait}_r\,(m, 1) + \mathrm{\urcorner}\,(\mathrm{succ}_{r-1}\,(\mathrm{\ulcorner}\,(m))) \quad .$$

Preuve : D'après la définition 6.14 du successeur, et l'égalité (6.6) de la propriété 6.18, on a, si $\mathrm{g}_0\,(m) \geq 3$:

$$\begin{aligned}
\mathrm{succ}_r\,(m) &= \mathrm{extrait}_r\,(m, 1) + \mathrm{\urcorner}\,(\mathrm{extrait}_{r-1}\,(\mathrm{\ulcorner}\,(m), \mathrm{g}_0\,(m) - 3)) \\
&\quad + \mathrm{bloc}_r\,(\mathrm{g}_0\,(m), r - \mathrm{d}_0\,(m) + \mathrm{g}_0\,(m)) \\
&= \mathrm{extrait}_r\,(m, 1) + \mathrm{\urcorner}\,(\mathrm{extrait}_{r-1}\,(\mathrm{\ulcorner}\,(m), \mathrm{g}_0\,(\mathrm{\ulcorner}\,(m)) - 2)) \\
&\quad + \mathrm{\urcorner}\,(\mathrm{bloc}_{r-1}\,(\mathrm{g}_0\,(\mathrm{\ulcorner}\,(m)), r - \mathrm{d}_0\,(\mathrm{\ulcorner}\,(m)) + \mathrm{g}_0\,(\mathrm{\ulcorner}\,(m)))) \\
&= \mathrm{extrait}_r\,(m, 1) + \mathrm{\urcorner}\,(\mathrm{succ}_{r-1}\,(\mathrm{\ulcorner}\,(m))) \quad .
\end{aligned}$$

Il n'y a plus qu'à monter que l'hypothèse

$$\mathrm{\ulcorner}\,(m) \in M_P(\mathrm{niveau}\,(\mathrm{\ulcorner}\,(m)), r-1)$$

implique $\mathrm{g}_0\,(m) \geq 3$ ou, ce qui est équivalent d'après la propriété 6.18, $\mathrm{g}_0\,(\mathrm{\ulcorner}\,(m)) \geq 2$.

Supposons que $\mathrm{g}_0\,(\mathrm{\ulcorner}\,(m)) = 1$, alors, d'après la propriété 6.13 appliquée à $\mathrm{\ulcorner}\,(m)$, nous obtenons : $\mathrm{\ulcorner}\,(m) = \mathrm{bloc}_{r-1}\,(\mathrm{d}_0\,(\mathrm{\ulcorner}\,(m)) + 1, r-1)$ avec forcément $r - 1 - \mathrm{d}_0\,(\mathrm{\ulcorner}\,(m)) = \mathrm{niveau}\,(\mathrm{\ulcorner}\,(m))$, or cet élément n'appartient pas à $M_P(\mathrm{niveau}\,(\mathrm{\ulcorner}\,(m)), r-1)$ par définition de $M_P(\mathrm{niveau}\,(\mathrm{\ulcorner}\,(m)), r-1)$. Nous avons donc bien, pour $\mathrm{\ulcorner}\,(m) \in M_P(\mathrm{niveau}\,(\mathrm{\ulcorner}\,(m)), r-1)$, $\mathrm{g}_0\,(\mathrm{\ulcorner}\,(m)) \geq 2$. □

Soit r un entier strictement positif. Nous montrons que tout élément de $M(r)$ est un successeur du motif nul. Pour cela nous adoptons la convention suivante :

Définition 6.21. Pour $m \in M(r)$, nous posons

$$\mathrm{succ}_r^0\,(m) = m$$

et, pour i un entier strictement positif, si $\mathrm{succ}_r^{i-1}\,(m) \in M_P(r)$ alors :

$$\mathrm{succ}_r^i\,(m) = \mathrm{succ}_r\left(\mathrm{succ}_r^{i-1}\,(m)\right) \quad .$$

Proposition 6.22. *Soit r un entier strictement positif, pour ℓ, k des entiers vérifiant $0 \leq \ell \leq k \leq r$, on a :*

$$M(\ell, k) = \left\{ \mathrm{succ}_k^{i-1}\,(\mathrm{bloc}_k\,(1, \ell)) \;\middle|\; 1 \leq i \leq \binom{k}{\ell} \right\} \quad ,$$

avec

$$\mathrm{succ}_k^{\binom{k}{\ell}-1}\left(\mathrm{bloc}_k\left(1,\ell\right)\right)=\mathrm{bloc}_k\left(k-\ell+1,k\right)$$

et pour ℓ,k *vérifiant* $0\leq\ell<k\leq r$, *nous avons :*

$$\mathrm{succ}_k^{\binom{k}{\ell}}\left(\mathrm{bloc}_k\left(1,\ell\right)\right)=\mathrm{bloc}_k\left(1,\ell+1\right)$$

Preuve : Nous prouvons cette proposition par récurrence.

Pour $r=1$:

$$M(0,1)=\{0\}\qquad\text{et}\qquad M(1,1)=\left\{\mathrm{succ}_1^1\left(0\right)\right\}\quad.$$

Pour $r=2$:

$$M(0,2)=\{00\}\quad,$$

or, d'après la définition 6.14 de l'application successeur :

$$\mathrm{succ}_2^1\left(00\right)=10,\qquad\mathrm{succ}_2^1\left(10\right)=01\quad\text{et}\quad\mathrm{succ}_2^1\left(01\right)=11\quad.$$

Il est aisé de voir que, avec ces valeurs, la proposition 6.22 est vérifiée pour $r=2$.

Pour $r\geq3$:

Supposons que la proposition 6.22 est vérifiée pour $r-1$. Le cas $\ell=0$ est trivial et ressort de la définition 6.21 et de la définition 6.14 de l'application successeur.

Supposons $1\leq\ell<r$. Par induction de la propriété 6.20, dont les conditions sont vérifiées, pour $1\leq i\leq\binom{r-1}{\ell-1}$,

$$\mathrm{succ}_r^{i-1}\left(\mathrm{bloc}_r\left(1,\ell\right)\right)=\mathrm{bloc}_r\left(1,1\right)+\mathrm{r}\left(\mathrm{succ}_{r-1}^{i-1}\left(\mathrm{bloc}_{r-1}\left(1,\ell-1\right)\right)\right)\quad.\tag{6.7}$$

Soit, en posant

$$\begin{aligned}m'&=\mathrm{succ}_r^{\binom{r-1}{\ell-1}-1}\left(\mathrm{bloc}_r\left(1,\ell\right)\right)\\&=\mathrm{bloc}_r\left(1,1\right)+\mathrm{r}\left(\mathrm{bloc}_{r-1}\left(r-\ell+1,r-1\right)\right)\\&=\mathrm{bloc}_r\left(1,1\right)+\mathrm{bloc}_r\left(r-\ell+2,r\right)\end{aligned}$$

alors

$$\mathrm{succ}_r^{\binom{r-1}{\ell-1}}\left(\mathrm{bloc}_r\left(1,\ell\right)\right)=\mathrm{succ}_r\left(m'\right)\quad,$$

or $\mathrm{g}_0\left(m'\right)=2$ et $\mathrm{d}_0\left(m'\right)=r-\ell+1$, d'où par la définition 6.14 du successeur,

$$\begin{aligned}\mathrm{succ}_r^{\binom{r-1}{\ell-1}}\left(\mathrm{bloc}_r\left(1,\ell\right)\right)&=\mathrm{bloc}_r\left(2,\ell+1\right)\\&=\mathrm{r}\left(\mathrm{bloc}_{r-1}\left(1,\ell\right)\right)\quad.\end{aligned}$$

D'où par induction de la propriété 6.20, pour $1 \leq i \leq \binom{r-1}{\ell}$,

$$\mathrm{succ}_r^{\binom{r-1}{\ell-1}+i-1}\left(\mathrm{bloc}_r\left(1,\ell\right)\right) =\!\!\nearrow \left(\mathrm{succ}_{r-1}^{i-1}\left(\mathrm{bloc}_{r-1}\left(1,\ell\right)\right)\right) \quad . \tag{6.8}$$

Soit, en posant

$$\begin{aligned} m'' &= \mathrm{succ}_r^{\binom{r-1}{\ell-1}+\binom{r-1}{\ell}-1}\left(\mathrm{bloc}_r\left(1,\ell\right)\right) \\ &=\!\!\nearrow \left(\mathrm{bloc}_{r-1}\left(r-1-\ell+1,r-1\right)\right) \\ &= \mathrm{bloc}_r\left(r-\ell+1,r\right) \end{aligned}$$

alors

$$\mathrm{succ}_r^{\binom{r-1}{\ell-1}+\binom{r-1}{\ell}}\left(\mathrm{bloc}_r\left(1,\ell\right)\right) = \mathrm{succ}_r\left(m''\right) \quad ,$$

or $\mathrm{g}_0\left(m''\right)=1$ et $\mathrm{d}_0\left(m''\right)=r-\ell$, d'où par la définition 6.14 du successeur,

$$\mathrm{succ}_r^{\binom{r}{\ell}}\left(\mathrm{bloc}_r\left(1,\ell\right)\right) = \mathrm{bloc}_r\left(1,\ell+1\right) \quad .$$

Les équations (6.7) et (6.8) montrent que si l'hypothèse de récurrence est vérifiée pour $r-1$, et qu'on a donc,

$$M(\ell,r-1) = \left\{ \mathrm{succ}_{r-1}^{i-1}\left(\mathrm{bloc}_{r-1}\left(1,\ell\right)\right) \;\middle|\; 1 \leq i \leq \binom{r-1}{\ell} \right\} \quad ,$$

et

$$M(\ell-1,r-1) = \left\{ \mathrm{succ}_{r-1}^{i-1}\left(\mathrm{bloc}_{r-1}\left(1,\ell-1\right)\right) \;\middle|\; 1 \leq i \leq \binom{r-1}{\ell-1} \right\}$$

alors l'ensemble

$$\left\{ \mathrm{succ}_r^{i-1}\left(\mathrm{bloc}_r\left(1,\ell\right)\right) \;\middle|\; 1 \leq i \leq \binom{r}{\ell} \right\}$$

est bien constitué de $\binom{r}{\ell}$ éléments tous distincts, de niveau ℓ et est donc égal à $M(\ell,r)$. Ceci conclut la récurrence pour r. □

Corollaire 6.23. *La proposition 6.22 a pour conséquence immédiate, pour r un entier strictement positif:*

$$M(r) = \left\{ \mathrm{succ}_r^{i-1}\left(0\right) \;\middle|\; 1 \leq i \leq 2^r \right\} \quad .$$

Ce corollaire nous donne la possibilité de parcourir itérativement l'ensemble $M(r)$ en définissant un ordre sur les motifs de réconciliation.

Définition 6.24. Les motifs de partition appartenant à $M(r)$ sont ordonnés par $\leq$ définie par $m \leq m'$ si m' est parmi les successeurs de m.

Plus précisément, avec les notations de 6.21, et d'après le corollaire 6.23, en posant $m = \mathrm{succ}_r^i\left(0\right)$ et $m' = \mathrm{succ}_r^j\left(0\right)$; alors $m \leq m'$ si $i \leq j$.

6.1.4 Propriétés de l'ordre sur les motifs de partition

Lemme 6.25. *Pour $m \in M_P(r)$, nous avons :*

$$\text{niveau}\,(\text{succ}_r\,(m)) = \text{niveau}\,(m) + 1 - \text{niveau}\,(\text{bloc}_r\,(\text{g}_0\,(m) - 1, \text{g}_0\,(m) - 1)) \quad .$$

Preuve : D'après la définition 6.14 du successeur de m,

$$\text{niveau}\,(\text{succ}_r\,(m)) = \text{niveau}\,(\text{extrait}_r\,(m, \text{g}_0\,(m) - 2)) + r - \text{d}_0\,(m) + 1 \quad .$$

Or d'après la proposition 6.13,

$$\begin{aligned}\text{niveau}\,(m) = {} & \text{niveau}\,(\text{extrait}_r\,(m, \text{g}_0\,(m) - 2)) \\ & + \text{niveau}\,(\text{bloc}_r\,(\text{g}_0\,(m) - 1, \text{g}_0\,(m) - 1)) \\ & + r - \text{d}_0\,(m) \quad .\end{aligned}$$

d'où l'égalité. □

Proposition 6.26. *Le niveau est une fonction croissante de $M(r)$ pour l'ordre défini en 6.24.*

Preuve : Il suffit de montrer que pour $m \in M_P(r)$,

$$\text{niveau}\,(\text{succ}_r\,(m)) \geq \text{niveau}\,(m) \quad . \tag{6.9}$$

Comme

$$\text{niveau}\,(\text{bloc}_r\,(\text{g}_0\,(m) - 1, \text{g}_0\,(m) - 1)) \leq 1 \quad ,$$

l'inégalité (6.9) est obtenue en utilisant l'égalité du lemme 6.25.

Cette proposition peut aussi être vue comme une conséquence immédiate de la proposition 6.22. □

Proposition 6.27. *À niveau ℓ fixé, avec $0 \leq \ell \leq r$, le sous-niveau est une fonction croissante de $M(\ell, r)$ pour l'ordre défini en 6.24.*

Preuve : Il n'y a rien à démontrer pour $\ell = 0$ ou $\ell = r$ car dans ces deux cas $\text{card}\,(M(\ell, r)) = 1$. Supposons $m \in M(\ell, r)$ pour $1 \leq \ell \leq r - 1$.

Il suffit de montrer que $\text{sous-niveau}\,(\text{succ}_r\,(m)) \geq \text{sous-niveau}\,(m)$ pour un motif m tel que $\text{niveau}\,(\text{succ}_r\,(m)) = \text{niveau}\,(m) = \ell$.

Or d'après l'égalité du lemme 6.25, si $\text{niveau}\,(\text{succ}_r\,(m)) = \text{niveau}\,(m) = \ell$, alors on a nécessairement $\text{niveau}\,(\text{bloc}_r\,(\text{g}_0\,(m) - 1, \text{g}_0\,(m) - 1)) = 1$, le bloc $\text{bloc}_r\,(\text{g}_0\,(m) - 1, \text{g}_0\,(m) - 1)$ n'est donc pas le bloc nul.

La propriété 6.13 et la définition 6.2 du sous-niveau, entraînent donc :

$$\text{sous-niveau}\,(m) \leq \text{g}_0\,(m) - 2 \leq \text{g}_0\,(m) - 1 \quad . \tag{6.10}$$

La définition 6.14 de l'application succ_r entraîne :

$$\text{sous-niveau}\,(\mathrm{succ}_r\,(m)) = \min\{\text{sous-niveau}\,(\mathrm{extrait}_r\,(m, \mathrm{g}_0\,(m) - 2))\,, \mathrm{g}_0\,(m) - 1\} \quad ,$$

la proposition 6.5 :

$$\text{sous-niveau}\,(\mathrm{extrait}_r\,(m, \mathrm{g}_0\,(m) - 2)) \geq \text{sous-niveau}\,(m) \quad ,$$

et l'inégalité (6.10) impliquent le résultat :

$$\text{sous-niveau}\,(\mathrm{succ}_r\,(m)) \geq \text{sous-niveau}\,(m) \quad .$$

□

Remarque : La proposition 6.22 nous indique aussi qu'à niveau ℓ fixé, $1 \leq \ell \leq r$, le plus petit élément de $M(\ell, r)$ est $\mathrm{bloc}_r\,(1, \ell)$, de sous-niveau 0 et que le plus grand élément de $M(\ell, r)$ est $\mathrm{bloc}_r\,(r - \ell + 1, r)$ de sous-niveau ℓ.

La propriété de l'ordre 6.24 donnée par la proposition 6.27 sera utilisée dans le paragraphe 6.2.3 pour éviter de refaire des calculs inutiles lors de la reformulation du problème de remontée lorsque sont détectés des vrais facteurs qui ne sont pas nécessairement irréductibles.

6.2 Reformulation du problème de remontée

Bien qu'il existe un algorithme en temps polynomial pour la factorisation des polynômes à coefficients dans les entiers [63], le schéma classique de Berlekamp-Zassenhaus de complexité exponentielle dans le pire des cas [52] est toujours utilisé et implanté. Il l'est en raison de son efficacité pratique sur les problèmes accessibles au calcul formel. C'est pourquoi il est toujours intéressant de l'améliorer.

Le schéma de Berlekamp-Zassenhaus a récemment été modifié sur les points suivants : *détection précoce de vrais facteurs* [109], [110], utilisation des *bornes sur un seul facteur* (*single factor bounds*) [9] et de nouvelles *méthodes de remontée* des facteurs modulaires [33], [110].

Soit f un polynôme à coefficients entiers, séparable (sans racines multiples), primitif et dont le coefficient dominant $\mathrm{cd}\,(f)$ est positif. Fixons p, un premier impair ne divisant pas le discriminant de f (défini au paragraphe 2.3.5), ni $\mathrm{cd}\,(f)$: $f \bmod p$ est aussi séparable sur $\mathbb{F}_p[t]$ et son degré est le même que celui de f sur $\mathbb{Z}[t]$.

Nous traitons de la reformulation du problème de remontée (paragraphe 5.5.2) lorsque sont trouvés des *vrais facteurs* qui ne sont pas forcément irréductibles.

P. S. Wang a proposé dans [110] un algorithme de lemme de Hensel effectif en l'accompagnant d'une reformulation du problème de remontée dans sa méthode.

G. E. Collins et M. J. Encarnación ont proposé dans [33] un nouvel algorithme de lemme de Hensel effectif mais ils n'indiquent pas comment effectuer la reformulation du problème de remontée dans leur méthode. Or cette reformulation est indispensable pour

implanter efficacement leur algorithme. La méthode de remontée de Collins et Encarnación est rappelée au paragraphe 6.2.1. Nous fournissons au paragraphe 6.2.2 tous les calculs nécessaires pour la reformulation du problème de remontée dans la méthode de Collins et Encarnación. Elle est accompagnée d'une étude de complexité qui montre que la reformulation est plus intéressante qu'un re-calcul.

Nous y avons aussi ajouté au paragraphe 6.2.3 l'utilisation de l'algorithme d'énumération des motifs de partitions du paragraphe 6.1 pour éviter des calculs superflus.

6.2.1 Méthode de remontée de Collins et Encarnación

Dans notre présentation de la méthode de Collins et Encarnación [33] nous avons fait de légers changements dans leurs notations pour unifier la présentation des différentes étapes de leur algorithme.

Notation 6.28. Soient $A_1, \ldots, A_r \in \mathbb{Z}/p\mathbb{Z}[t]$ les facteurs du polynôme f modulo le premier impair p. Le coefficient dominant du facteur A_1 est égal à $\mathrm{cd}\,(f) \bmod p$ et les autres facteurs A_i, pour $2 \leq i \leq r$, sont unitaires.

$$f \equiv \prod_{i=1}^{r} A_i \bmod p \quad .$$

Notation 6.29. Soit q la plus grande puissance de p rentrant dans un mot machine. Pour tout entier $j \geq 1$, les anneaux quotients $\mathbb{Z}/q^j\mathbb{Z}$ sont représentés par l'ensemble des entiers

$$\mathbb{Z}(q,j) = \{-(q^j-1)/2, \ldots, (q^j-1)/2\} \quad .$$

Lorsque nous considérons des polynômes dans $\mathbb{Z}/q^j\mathbb{Z}[t]$ comme des polynômes dans $\mathbb{Z}[t]$, nous gardons ces valeurs signées sur $\mathbb{Z}$. Lorsque nous effectuons des opérations modulo q^j nous conservons les résidus dans le même ensemble de valeurs.

Définition 6.30. La *division euclidienne symétrique* d'un polynôme $C \in \mathbb{Z}[t]$ par q est le calcul d'un quotient $D \in \mathbb{Z}[t]$ et d'un reste symétrique E avec ses coefficients dans $\mathbb{Z}(q,1)$ tels que :

$$C = D\,q + E \quad .$$

Notation 6.31. Pour tout entier $j \geq 1$, $A_{1,j}, \ldots, A_{r,j} \in \mathbb{Z}(q,j)[t]$ désignent les facteurs modulaires remontés modulo q^j de f et pour $0 \leq i \leq r$, les polynômes unitaires $B_{i,j} \in \mathbb{Z}(q,j)[t]$ sont les produits modulo q^j des $r-i$ derniers facteurs de f modulo q^j.

Nous avons ainsi pour $j \geq 1$:

$$f \equiv \prod_{i=1}^{r} A_{i,j} \bmod q^j \quad ,$$

$$\begin{aligned} A_i &\equiv A_{i,j} \bmod p \qquad \text{pour} \quad 1 \leq i \leq r \quad , \\ \text{avec} \qquad & \mathrm{cd}\,(A_{1,j}) = \mathrm{cd}\,(f) \bmod q^j \\ & \mathrm{cd}\,(A_{i,j}) = 1 \qquad \text{pour} \quad 2 \leq i \leq r \quad , \end{aligned}$$

$$\begin{aligned} B_{i,j} &\equiv \prod_{h=i+1}^{r} A_{h,j} \bmod q^j \qquad \text{pour} \quad 0 \leq i \leq r \quad , \\ B_{i-1,j} &\equiv A_{i,j} B_{i,j} \bmod q^j \qquad \text{pour} \quad 1 \leq i \leq r \quad , \end{aligned}$$

$$\begin{aligned} f &\equiv A_{1,j} \cdots A_{i,j} B_{i,j} \bmod q^j \\ A_{r,j} &= B_{r-1,j} \quad . \end{aligned}$$

Comme G. E. COLLINS et M. J. ENCARNACIÓN le suggèrent dans [33], la factorisation modulo p peut être remontée par un lemme de Hensel quadratique (comme [79]) en une factorisation modulo q qui donne pour $1 \leq i \leq r-1$, les polynômes $A_{i,1}$ et $B_{i,1}$. Au cours du processus de remontée quadratique (Algorithme Q dans [79]), des polynômes de *base de remontée* $S_i, T_i \in \mathbb{Z}(q,1)[t]$ sont calculés.

Définition 6.32. Les polynômes de *base de remontée* $S_i, T_i \in \mathbb{Z}(q,1)[t]$ satisfont pour $1 \leq i \leq r-1$:

$$A_{i,1} S_i + B_{i,1} T_i \equiv 1 \bmod q \tag{6.11}$$

$$\deg(S_i) < \deg(B_{i,1}) \tag{6.12}$$

$$\deg(T_i) < \deg(A_{i,1}) \quad . \tag{6.13}$$

Notation 6.33. Pour $j \geq 1$, les polynômes $Y_{0,j} \in \mathbb{Z}(q,1)[t]$ désignent les termes du développement q-adique de f. Ils sont définis par:

$$f = Y_{0,1} + Y_{0,2}\, q + \cdots + Y_{0,j}\, q^{j-1} + \cdots \quad , \tag{6.14}$$

avec $\deg(Y_{0,j}) \leq \deg(f)$.

Pour $j \geq 1$, nous avons $B_{0,j} \in \mathbb{Z}(q,j)[t]$ qui est égal à ce développement modulo q^j:

$$B_{0,j} = Y_{0,1} + Y_{0,2}\, q + \cdots + Y_{0,j}\, q^{j-1} \equiv f \bmod q^j \quad .$$

Notation 6.34. Nous notons $R_j \in \mathbb{Z}[t]$, $\deg(R_j) \leq \deg(f)$, les quotients successifs des divisions euclidiennes symétriques f par q associées aux restes $Y_{0,j} \in \mathbb{Z}(q,1)[t]$:

$$R_j = R_{j+1}\, q + Y_{0,j+1} \qquad \text{et} \qquad f = B_{0,j} + q^j R_j \quad .$$

Dans le but de remonter les facteurs modulaires du polynôme f du modulo q^j au modulo q^{j+1}, la méthode de Collins et Encarnación calcule de façon inductive les termes du développement q-adique des facteurs modulaires du polynôme f.

Notation 6.35. Pour $j \geq 2$ et $1 \leq i \leq r-1$, nous notons $Z_{i,j} \in \mathbb{Z}(q,1)[t]$ et $Y_{i,j} \in \mathbb{Z}(q,1)[t]$ respectivement, les termes des développements q-adiques symétriques de $A_{i,1}$ et $B_{i,1}$.

Comme $A_{i,j} \in \mathbb{Z}(q,j)[t]$ et $B_{i,j} \in \mathbb{Z}(q,j)[t]$ sont égaux à ces développements modulo q^j, nous avons :

$$\begin{aligned} A_{i,j} &= A_{i,1} + Z_{i,2}\, q + \cdots + Z_{i,j}\, q^{j-1} \\ B_{i,j} &= B_{i,1} + Y_{i,2}\, q + \cdots + Y_{i,j}\, q^{j-1} \\ A_{i,j} &= A_{i,j-1} + Z_{i,j}\, q^{j-1} \\ B_{i,j} &= B_{i,j-1} + Y_{i,j}\, q^{j-1} \end{aligned}$$

avec

$$\begin{aligned} &\deg(Z_{1,j}) \leq \deg(A_1) \\ &\deg(Z_{i,j}) < \deg(A_i) \qquad \text{pour} \quad 2 \leq i \leq r-1 \\ &\deg(Y_{i,j}) < \deg(B_i) \qquad \text{pour} \quad 1 \leq i \leq r-1 \end{aligned}$$

et $\deg(Z_{1,j}) < \deg(A_1)$ si f est unitaire ou $q^j > 2\mathrm{cd}\,(f)$.

Dans les calculs inductifs de la méthode de Collins et Encarnación, des polynômes correctifs sont nécessaires.

Définition 6.36. Nous définissons, pour $j \geq 2$ et $1 \leq i \leq r-1$, les *polynômes correctifs* $U_{i,j} \in \mathbb{Z}[t]$ par

$$U_{i,j} = \frac{B_{i-1,j} - A_{i,j-1} B_{i,j-1}}{q^{j-1}} \quad .$$

Leurs calculs sont effectués sur $\mathbb{Z}[t]$ avec

$$\begin{aligned} &\deg(U_{1,j}) \leq \deg(f) \\ &\deg(U_{i,j}) < \deg(B_{i-1}) \qquad \text{pour} \quad 2 \leq i \leq r-1 \end{aligned}$$

et $\deg(U_{1,j}) < \deg(f)$ si f est unitaire ou $q^j > 2\mathrm{cd}\,(f)$.

Dans les formules qui apparaissent dans la suite de ce chapitre, une variable inconnue prend une valeur induite par la formule dans laquelle son nom est écrit en caractère gras.

Étape inductive

Nous rappelons d'abord l'étape inductive de la méthode de Collins et Encarnación. Nous prenons $j \geq 1$ et supposons que la remontée des facteurs modulaires du polynôme f a été effectuée jusqu'à q^j. Ainsi, les polynômes $A_{i,j-1}, B_{i,j-1} \in \mathbb{Z}(q,j)[t]$, $Z_{i,j}, Y_{i,j} \in \mathbb{Z}(q,1)[t]$,

$A_{i,j}, B_{i,j} \in \mathbb{Z}(q,j)[t], U_{i,j} \in \mathbb{Z}[t]$, pour $1 \leq i \leq r-1$ et $R_j \in \mathbb{Z}[t]$, sont supposés calculés dans l'étape précédente de la remontée. Nous allons calculer les facteurs modulaires de f modulo q^{j+1}.

Tout d'abord, les variables $R_{j+1} \in \mathbb{Z}[t]$ et $Y_{0,j+1} \in \mathbb{Z}(q,1)[t]$ sont calculées par une division euclidienne symétrique de R_j par q:

$$R_j = \boldsymbol{R}_{j+1}\, q + \boldsymbol{Y}_{0,j+1} \quad . \tag{6.15}$$

Pour i croissant de 1 à $r-1$:

$$\boldsymbol{V}_{i,j} := (A_{i,j-1}Y_{i,j} + B_{i,j-1}Z_{i,j} - U_{i,j})/q \quad ; \tag{6.16}$$

où les opérations sont effectuées dans $\mathbb{Z}[t]$ (la division par q est exacte) et

$$\boldsymbol{U}_{i,j+1} := Y_{i-1,j+1} - V_{i,j} - Z_{i,j}Y_{i,j}\, q^{j-2} \quad ; \tag{6.17}$$

soit

$$\widetilde{U}_{i,j+1} :\equiv U_{i,j+1} \bmod q \quad , \tag{6.18}$$

ils trouvent des polynômes $Y_{i,j+1}, Z_{i,j+1} \in \mathbb{Z}(q,1)[t]$ (par l'algorithme P dans [79]) tels que :

$$A_{i,1}\boldsymbol{Y}_{i,j+1} + B_{i,1}\boldsymbol{Z}_{i,j+1} \equiv \widetilde{U}_{i,j+1} \bmod q \quad ; \tag{6.19}$$

la remontée est achevée par :

$$\boldsymbol{A}_{i,j+1} := A_{i,j} + Z_{i,j+1}\, q^j \qquad \text{et} \qquad \boldsymbol{B}_{i,j+1} := B_{i,j} + Y_{i,j+1}\, q^j \quad . \tag{6.20}$$

La preuve de l'algorithme ci-dessus est dans [33]. Elle est basée sur

$$U_{i,j+1} = \frac{B_{i-1,j+1} - A_{i,j}B_{i,j}}{q^j} \tag{6.21}$$

pour $j \geq 1$ et $1 \leq i \leq r-1$.

Étape d'initialisation

La remontée des facteurs modulaires de q à q^2 est considérée comme un cas spécial dans [33]. Ce cas peut en fait être traité par les opérations inductives du paragraphe 6.2.1 à la condition de prendre certaines valeurs d'initialisation *ad hoc* (irrégulières) données ci-dessous.

Nous calculons d'abord la division euclidienne symétrique de f par q:

$$f = \boldsymbol{R}_1\, q + \boldsymbol{B}_{0,1} \tag{6.22}$$

et, pour $1 \leq i \leq r-1$,

$$\boldsymbol{U}_{i,1} := B_{i-1,1} - A_{i,1}B_{i,1} \quad . \tag{6.23}$$

Ensuite, nous posons

$$\boldsymbol{A}_{i,0} := 0, \qquad \boldsymbol{B}_{i,0} := 0, \qquad \boldsymbol{Z}_{i,1} := 0, \qquad \boldsymbol{Y}_{i,1} := 0 \quad . \tag{6.24}$$

Ces valeurs utilisées dans (6.16) et (6.17) donnent bien

$$U_{i,2} = Y_{i-1,2} + \frac{B_{i-1,1} - A_{i,1}B_{i,1}}{q} = \frac{B_{i-1,2} - A_{i,1}B_{i,1}}{q}$$

comme exigé par la méthode de Collins et Encarnación ou la formule (6.21).

La division par q dans (6.16) est exacte car par construction, $B_{i-1,1} \equiv A_{i,1}B_{i,1} \bmod q$.

6.2.2 Reformulation du problème de remontée

Ce paragraphe a trait à la reformulation du problème de remontée. Nous supposons connu le polynôme $\widehat{f}$, un vrai facteur du polynôme f associé au ℓ-uplet de facteurs modulaires $A_{i_1,k}, \ldots, A_{i_\ell,k}$ remontés modulo q^k, avec $\{i_1, \ldots, i_\ell\} \subset \{1 \ldots, r\}$, et $i_1 < \ldots < i_\ell$, $\ell < r$. Il peut avoir été aussi bien trouvé par reconstruction de rationnels que par les restes symétriques, méthodes qui sont exposées au paragraphe 5.5.2 ou par toute autre méthode.

Notation 6.37. L'injection strictement croissante de l'ensemble $\{1, \ldots, \ell\}$ dans $\{1, \ldots, r\}$ qui envoie $e \in \{1, \ldots, \ell\}$ sur i_e est notée $\widehat{}$.

Le but de ce paragraphe est de calculer les polynômes (que nous notons avec des chapeaux) associés à $\widehat{f}$ qui peuvent être utilisés par l'étape inductive du paragraphe 6.2.1 avec $\widehat{f}$ au lieu de f et en partant du modulo q^k. Nous évitons ainsi de recalculer la remontée à partir du modulo p jusqu'à q^k pour les nouveaux facteurs modulaires associés au ℓ-uplet.

Reformulation de la base de remontée

Comme p ne divise pas le coefficient dominant de f, $\mathrm{cd}\,(f)$ et $\mathrm{cd}\left(\widehat{f}\right)$ (qui est un facteur de $\mathrm{cd}\,(f)$) sont inversibles dans $\mathbb{Z}(q, j)$ pour $j \geq 1$.

Soit

$$\widehat{\boldsymbol{A}}_{1,1} :\equiv \mathrm{cd}\left(\widehat{f}\right) \mathrm{cd}\left(A_{\widehat{1},1}\right)^{-1} A_{\widehat{1},1} \bmod q$$

et, pour $2 \leq i \leq \ell$,

$$\widehat{\boldsymbol{A}}_{i,1} := A_{\widehat{i},1} \quad .$$

Alors

$$\widehat{\boldsymbol{B}}_{\ell-1,1} := \widehat{A}_{\ell,1}$$

et pour i décroissant de $\ell - 2$ à 1,

$$\widehat{\boldsymbol{B}}_{i,1} :\equiv \widehat{B}_{i+1,1}\widehat{A}_{i+1,1} \bmod q \quad . \tag{6.25}$$

Nous calculons maintenant une base de remontée reformulée $\widehat{S}_i, \widehat{T}_i \in \mathbb{Z}(q,1)[t]$, pour $1 \leq i \leq \ell - 1$ satisfaisant :

$$\widehat{A}_{i,1}\widehat{S}_i + \widehat{B}_{i,1}\widehat{T}_i \equiv 1 \bmod q \tag{6.26}$$

$$\deg(\widehat{S}_i) < \deg(\widehat{B}_{i,1}) \tag{6.27}$$

$$\deg(\widehat{T}_i) < \deg(\widehat{A}_{i,1}) \tag{6.28}$$

Lemme 6.38. *Soient $Q_{S,1}$ et $\widehat{S}_1$ les quotient et reste de la division euclidienne modulo q du polynôme* $\mathrm{cd}\left(A_{\widehat{1},1}\right) \mathrm{cd}\left(\widehat{f}\right)^{-1} S_{\widehat{1}}$ *par le polynôme unitaire $\widehat{B}_{i,1}$:*

$$\mathrm{cd}\left(A_{\widehat{1},1}\right) \mathrm{cd}\left(\widehat{f}\right)^{-1} S_{\widehat{1}} \equiv \boldsymbol{Q}_{S,i}\widehat{B}_{1,1} + \widehat{\boldsymbol{S}}_1 \bmod q \quad .$$

Pour $2 \leq i \leq \ell - 1$, soient $Q_{S,i}$ et $\widehat{S}_i$ les quotient et reste de la division euclidienne mod$q$ *du polynôme $S_{\widehat{i}}$ par le polynôme unitaire $\widehat{B}_{i,1}$:*

$$S_{\widehat{i}} \equiv \boldsymbol{Q}_{S,i} \widehat{B}_{i,1} + \widehat{\boldsymbol{S}}_i \bmod q \quad .$$

Soit, pour $1 \leq i \leq \ell - 1$, $Q_{B,i}$ le quotient de la division exacte mod$q$ *du polynôme $B_{\widehat{i},1}$ par le polynôme unitaire $\widehat{B}_{i,1}$:*

$$B_{\widehat{i},1} \equiv \boldsymbol{Q}_{B,i} \widehat{B}_{i,1} \bmod q \quad ,$$

et soit $\widehat{T}_i$, défini par:

$$\widehat{\boldsymbol{T}}_i :\equiv \widehat{A}_{i,1} Q_{S,i} + Q_{B,i} T_{\widehat{i}} \bmod q \quad .$$

Alors $\widehat{S}_i, \widehat{T}_i \in \mathbb{Z}(q,1)[t]$ pour $1 \leq i \leq \ell - 1$ forment une base de remontée satisfaisant les conditions (6.26), (6.27), et (6.28).

Preuve :

$$\begin{aligned}
\widehat{A}_{1,1}\widehat{S}_1 + \widehat{B}_{1,1}\widehat{T}_1 &\equiv \widehat{A}_{1,1}\left(\mathrm{cd}\left(A_{\widehat{1},1}\right)\mathrm{cd}\left(\widehat{f}\right)^{-1} S_{\widehat{1}} - Q_{S,1}\widehat{B}_{1,1}\right) \\
&\quad + \widehat{B}_{1,1}\left(\widehat{A}_{1,1}Q_{S,1} + Q_{B,1}T_{\widehat{1}}\right) \bmod q \\
&\equiv A_{\widehat{1},1}S_{\widehat{1}} + \widehat{B}_{1,1}Q_{B,1}T_{\widehat{1}} \bmod q \\
&\equiv A_{\widehat{1},1}S_{\widehat{1}} + B_{\widehat{1},1}T_{\widehat{1}} \bmod q \\
&\equiv 1 \bmod q
\end{aligned}$$

et, pour $2 \leq i \leq \ell - 1$,

$$\begin{aligned}
\widehat{A}_{i,1}\widehat{S}_i + \widehat{B}_{i,1}\widehat{T}_i &\equiv \widehat{A}_{i,1}(S_{\widehat{i}} - Q_{S,i}\widehat{B}_{i,1}) + \widehat{B}_{i,1}(\widehat{A}_{i,1}Q_{S,i} + Q_{B,i}T_{\widehat{i}}) \bmod q \\
&\equiv A_{\widehat{i},1}S_{\widehat{i}} + \widehat{B}_{i,1}Q_{B,i}T_{\widehat{i}} \bmod q \\
&\equiv A_{\widehat{i},1}S_{\widehat{i}} + B_{\widehat{i},1}T_{\widehat{i}} \bmod q \\
&\equiv 1 \bmod q \quad .
\end{aligned}$$

Nous avons $\deg(\widehat{S}_i) < \deg(\widehat{B}_{i,1})$ pour $1 \leq i \leq \ell - 1$ ce qui entraîne $\deg(\widehat{T}_i) < \deg(\widehat{A}_{i,1})$ pour $1 \leq i \leq \ell - 1$. □

Étape de ré-initialisation

Comme au paragraphe 6.2.1, nous devons calculer des valeurs d'initialisation convenables pour $\widehat{f}$ afin de reprendre la remontée simplement en appelant l'étape inductive du paragraphe 6.2.1.

Soit

$$\widehat{\boldsymbol{A}}_{1,k} :\equiv \mathrm{cd}\left(\widehat{f}\right)\mathrm{cd}\left(A_{\widehat{1},k}\right)^{-1} A_{\widehat{1},k} \bmod q^k$$

et, pour $2 \leq i \leq \ell$,

$$\widehat{\boldsymbol{A}}_{i,k} := A_{\widehat{i},k} \quad .$$

Alors

$$\widehat{\boldsymbol{B}}_{\ell-1,k} := A_{\widehat{\ell},k}$$

et pour i décroissant de $\ell - 2$ à 1,

$$\widehat{\boldsymbol{B}}_{i,k} :\equiv \widehat{B}_{i+1,k}\widehat{A}_{i+1,k} \bmod q^k \quad . \tag{6.29}$$

Nous calculons le développement q-adique symétrique de $\widehat{f}$ jusqu'au modulo q^k:

$$\widehat{f} = \widehat{\boldsymbol{Y}}_{0,1} + \widehat{\boldsymbol{Y}}_{0,2}\, q + \cdots + \widehat{\boldsymbol{Y}}_{0,k}\, q^{k-1} + \widehat{\boldsymbol{R}}_k \tag{6.30}$$

et dans le même temps :

$$\widehat{\boldsymbol{B}}_{0,k} := \widehat{Y}_{0,1} + \widehat{Y}_{0,2}\, q + \cdots + \widehat{Y}_{0,k}\, q^{k-1} \quad . \tag{6.31}$$

Et alors, pour $1 \leq i \leq \ell - 1$, comme au paragraphe 6.2.1 nous prenons des valeurs (irrégulières) d'initialisation *ad hoc* :

$$\widehat{\boldsymbol{U}}_{i,k} := \frac{\widehat{B}_{i-1,k} - \widehat{A}_{i,k}\widehat{B}_{i,k}}{q^{k-1}} \tag{6.32}$$

(la division est exacte) et

$$\widehat{\boldsymbol{A}}_{i,k-1} := 0, \qquad \widehat{\boldsymbol{B}}_{i,k-1} := 0, \qquad \widehat{\boldsymbol{Z}}_{i,k} := 0, \qquad \widehat{\boldsymbol{Y}}_{i,k} := 0 \quad . \tag{6.33}$$

Ces valeurs utilisées dans (6.16) et (6.17) donnent

$$\widehat{U}_{i,k+1} = \widehat{Y}_{i-1,k+1} + \frac{\widehat{B}_{i-1,k} - \widehat{A}_{i,k}\widehat{B}_{i,k}}{q^k} = \frac{\widehat{B}_{i-1,k+1} - \widehat{A}_{i,k}\widehat{B}_{i,k}}{q^k}$$

comme attendu par la méthode de Collins et Encarnación ou la formule (6.21).

Les deux divisions par q^{k-1} et q dans (6.32) et (6.16) sont exactes car par construction, $\widehat{B}_{i-1,k} \equiv \widehat{A}_{i,k}\widehat{B}_{i,k} \bmod q^k$.

Complexité de la reformulation comparée au re-calcul

Comme dans l'article [33] nous comptons et bornons le temps pour les opérations arithmétiques en nombre de mots, q étant la plus grande puissance de p rentrant dans un seul mot machine.

Notations et complexités élémentaires: Afin de continuer le processus de détection de vrais facteurs en calculant les facteurs $\widehat{f}$ modulo q^k (les $\widehat{A}_{i,k}$ pour $1 \leq i \leq \ell - 1$) nous pouvons soit reformuler le problème de remontée comme décrit dans le paragraphe 6.2.2 ou les recalculer ainsi que les polynômes secondaires nécessaires $\widehat{A}_{i,1}$, $\widehat{B}_{i,1}$, $\widehat{S}_i$, $\widehat{T}_i$, $\widehat{A}_{i,k-1}$, $\widehat{B}_{i,k-1}$, $\widehat{Z}_{i,k}$, $\widehat{Y}_{i,k}$ et $\widehat{U}_{i,k}$ à partir des facteurs de $\widehat{f}$ modulo p, que nous notons $\widehat{A}_1, \ldots, \widehat{A}_\ell \in \mathbb{Z}/p\mathbb{Z}[t]$. Nous les remontons par un lemme de Hensel quadratique jusqu'au modulo q et ensuite par les étapes inductives du paragraphe 6.2.1 jusqu'au modulo q^k.

Soit

$$\widehat{\boldsymbol{A}}_1 :\equiv \mathrm{cd}\left(\widehat{f}\right) \mathrm{cd}\,(A_{\widehat{1}})^{-1}\, A_{\widehat{1}} \bmod p$$

et, pour $2 \leq i \leq \ell$,

$$\widehat{\boldsymbol{A}}_i := A_{\widehat{i}} \quad .$$

Et alors, pour $0 \leq i \leq \ell - 1$,

$$\widehat{\boldsymbol{B}}_i := \prod_{h=i+1}^{\ell} \widehat{A}_h \bmod p \quad .$$

Soit $n := \deg(f)$ le degré de f et $n_i := \deg(A_i)$ pour $1 \leq i \leq r$. Soient $m_i := \deg(B_i) = \sum_{h=i+1}^{r} n_h$ pour $0 \leq i \leq r-1$. Soit c la longueur en mots du coefficient maximum de f.

Similairement, soit $\widehat{n} := \deg(\widehat{f})$ le degré de $\widehat{f}$ et $\widehat{n}_i := \deg(\widehat{A}_i)$ pour $1 \leq i \leq \ell$. Soient $\widehat{m}_i := \deg(\widehat{B}_i) = \sum_{h=i+1}^{\ell} \widehat{n}_h$ pour $0 \leq i \leq \ell - 1$. Soit $\widehat{c}$ la longueur en mots du coefficient maximum de $\widehat{f}$.

Calculer $\widehat{Y}_{0,j+1}$ et $\widehat{R}_j$ dans (6.15) coûte $O(\widehat{c}\,\widehat{n})$ opérations. Calculer $\widehat{U}_{i,j+1}$ dans (6.17) coûte $O(j\,\widehat{n}_i\,\widehat{m}_i)$ opérations. Résoudre les congruences (6.19) coûte $O(\widehat{m}_i^2 + \widehat{n}_i^2)$ opérations. Le coût de ces calculs domine celui des autres, ce qui conduit, en sommant sur i pour $1 \leq i \leq \ell - 1$ au coût $O(j\,\widehat{n}^2 + \ell\,\widehat{n}^2 + \widehat{c}\,\widehat{n})$ pour remonter tous les facteurs de q^j à q^{j+1}. Ensuite, en sommant sur j pour $1 \leq j \leq k$ le coût total pour remonter du modulo q au modulo q^k est

$$O(k^2\,\widehat{n}^2 + \ell\,k\,\widehat{n}^2 + k\,\widehat{c}\,\widehat{n}) \tag{6.34}$$

comme donné dans [33].

Comme le coût de la remontée de Hensel quadratique du modulo p au modulo q est $O(\ell\,\widehat{n}^2 + \widehat{c}\,\widehat{n})$ (voir [33]), le coût total du re-calcul pour remonter la factorisation du modulo p au modulo q^k est donné par (6.34).

Si nous utilisons à la place la reformulation du problème de remontée du paragraphe 6.2.2 pour remonter les facteurs jusqu'au modulo q^k, alors nous devons calculer $\widehat{Y}_{0,j+1}$ et $\widehat{R}_j$, ce qui a un coût de $O(\widehat{c}\,\widehat{n})$ opérations arithmétiques, pour $1 \leq j \leq k$, ce qui donne un coût de $O(k\,\widehat{c}\,\widehat{n})$ opérations. Calculer selon le lemme 6.38, $\widehat{S}_i$ et $\widehat{T}_i$ coûte respectivement $O((m_{\widehat{i}} - \widehat{m}_i)\widehat{m}_i)$ et $O((m_{\widehat{i}} - \widehat{m}_i)\widehat{n}_i)$ opérations ce qui donne en sommant sur les i un coût de $O((n - \widehat{n})\ell\,\widehat{n})$ opérations. Calculer $\widehat{B}_{i,k}$ pour tout i demande $O(k\,\widehat{n}^2 + \widehat{n}\,\ell\,k^2)$ opérations. Calculer $\widehat{U}_{i,k}$ pour tout i demande $O(k\,\widehat{n}^2 + \widehat{n}\,\ell\,k^2)$ opérations. Les coûts de ces calculs dominent les autres calculs restants. Le coût total de la reformulation est donc:

$$O(k\,\widehat{c}\,\widehat{n} + (n - \widehat{n})\ell\,\widehat{n} + k\widehat{n}^2 + \widehat{n}\ell k^2) \tag{6.35}$$

opérations arithmétiques.

Coût du calcul pour le vrai facteur et son complément : En général deux calculs doivent être effectués : à la fois pour le vrai facteur trouvé et le ℓ-uplet de facteurs modulaires remontés qui lui est associé, mais aussi pour le facteur complément $\overline{f} := f/\widehat{f}$ de degré $n - \widehat{n}$ et les $r - \ell$ facteurs modulaires remontés qui lui sont associés. Soit $\overline{c}$ la longueur en mots du coefficient maximum de $\overline{f}$.

Cela conduit au coût suivant pour le re-calcul :

$$O(k^2\widehat{n}^2 + \ell\, k\, \widehat{n}^2 + k\, \widehat{c}\, \widehat{n}) + O(k^2(n-\widehat{n})^2 + (r-\ell)k(n-\widehat{n})^2 + k\, \overline{c}(n-\widehat{n}))$$

soit un coût total pour le re-calcul de :

$$O(k^2(\widehat{n}^2 + (n-\widehat{n})^2) + k(\ell\, \widehat{n}^2 + (r-\ell)(n-\widehat{n})^2) + k(\widehat{c}\, \widehat{n} + \overline{c}(n-\widehat{n})))$$

opérations arithmétiques.

Pour la reformulation pour les deux facteurs, nous avons un coût de :

$$\begin{aligned} & O(k\, \widehat{c}\, \widehat{n} + (n-\widehat{n})\ell\, \widehat{n} + k\, \widehat{n}^2 + \widehat{n}\, \ell\, k^2) \\ & \quad + \quad O(k\, \overline{c}(n-\widehat{n}) + (n-\widehat{n})(r-\ell)\widehat{n} + k(n-\widehat{n})^2 + (n-\widehat{n})(r-\ell)k^2) \quad , \end{aligned}$$

soit un coût total pour la reformulation de :

$$O(k(\widehat{c}\, \widehat{n} + \overline{c}(n-\widehat{n})) + (n-\widehat{n})r\, \widehat{n} + k(\widehat{n}^2 + (n-\widehat{n})^2) + k^2(\widehat{n}\, \ell + (n-\widehat{n})(r-\ell)))$$

opérations arithmétiques.

Pour comparer, nous faisons l'hypothèse $\widehat{n} = O(n)$, $\ell = O(r)$, $\widehat{c} = O(c)$ et $\overline{c} = O(c)$. Elle conduit à un coût de $O(k^2n^2 + k\, r\, n^2 + k\, c\, n)$ opérations arithmétiques pour le re-calcul contre seulement $O(k^2 n\, r + r\, n^2 + k\, n^2 + k\, c\, n)$ opérations pour la reformulation.

6.2.3 Améliorations

Le paragraphe 5.5.3 a montré comment, en recherchant des vrais facteurs associés à des ℓ-uplets de facteurs modulo $m = q^j$ pour ℓ croissant à partir de 1, des degrés pouvaient être exclus de l'ensemble des degrés possibles pour les facteurs irréductibles du polynôme f.

Par ailleurs, lors de la détection précoce d'un vrai facteur $\widehat{f}$ de f, exposée au paragraphe 5.5.2, il est parfois nécessaire de poursuivre la recherche de vrais facteurs de $\widehat{f}$ (c'est le cas lorsque qu'aucune des conditions $C(m, b(\widehat{f}))$ et $C(m, B(\widehat{f}))$ définies au paragraphe 5.5.2 ne sont vérifiées). Il faut alors poursuivre la remontée pour un modulo m' supérieur à m. Par contre, il est fréquent (si $r - \ell > \ell$) que le facteur complément $\overline{f} = f/\widehat{f}$ ne puisse être prouvé irréductible : lorsque tous les polynômes candidats pour être vrais facteurs associés aux ℓ'-uplets de *ses* $r - \ell$ facteurs modulaires n'ont pas été testés. Il importe alors de les tester tous après reformulation soit pour prouver l'irréductibilité de $\overline{f}$ soit pour en trouver un vrai facteur.

Nous utilisons dans ce paragraphe l'ordre sur les motifs de partition défini au paragraphe 6.1 (définition 6.24) pour énumérer les ℓ-uplets de facteurs modulaires. Les propriétés de cet ordre permettent d'énumérer les ℓ-uplets pour ℓ croissant et d'éviter des calculs inutiles après reformulation.

Extraction des facteurs

Nous définissons d'abord l'injection utilisée au paragraphe 6.2.2 en fonction d'un motif de partition m.

Définition 6.39. Soit $m \in M(r)$ un motif de partition de longueur r de niveau ℓ. L'injection $\widehat{\ }$ de $\{1,\ldots,\ell\}$ dans $\{1,\ldots,r\}$ correspondant au motif de partition m associe à e, $1 \leq \ell$ l'indice de la e-ième coordonnée de m égale à 1.

Exemple : Pour $m = 01011$, $\widehat{\ }$ est définie par :

$$\begin{array}{rccc} \widehat{\ }: & \{1,2,3\} & \hookrightarrow & \{1,2,3,4,5\} \\ & 1 & \longmapsto & 2 \\ & 2 & \longmapsto & 4 \\ & 3 & \longmapsto & 5 \end{array}$$

Le polynôme candidat pour être vrai facteur de f associé au motif de partition m est donc construit à partir des facteurs modulaires $A_{i,j}$ dont les indices i sont dans l'image de $\widehat{\ }$.

Remarque : Le motif de partition correspondant au polynôme complément d'un vrai facteur associé au motif de partition m est le complément du motif de partition m, $\complement_r(m)$.

Éviter les combinaisons inutiles

Définition 6.40. Soit $m \in M(\ell, r)$ un motif de partition de longueur r et de niveau $1 \leq \ell \leq r$. Soit $m' \in M(r)$ un motif de partition de longueur r. La *trace* de m' sous m notée $\mathrm{trace}_m(m')$ est le motif de partition de $M(\ell)$ composé des ℓ coordonnées de m' pour lesquelles les coordonnées de m sont égales à 1 :

en posant $m' = m'_1 m'_2 \ldots m'_r$ et $\mathrm{trace}_m(m') = t_1 t_2 \ldots t_\ell$, alors $t_i = m'_{\widehat{i}}$ où $\widehat{\ }$ est donnée par la définition 6.39.

Exemples :

$$\begin{aligned} \mathrm{trace}_{11111}(01010) &= 01010 \\ \mathrm{trace}_{10101}(01010) &= 000 \\ \mathrm{trace}_{00011}(01010) &= 10 \\ \mathrm{trace}_{11010}(01010) &= 011 \quad . \end{aligned}$$

Proposition 6.41. *Soit m un motif de partition de longueur r et de niveau ℓ correspondant à un vrai facteur $\widehat{f}$ de f extrait parmi les r facteurs modulaires de f.*

Si les tests de polynômes candidats vrais facteurs de f ont été effectués pour les motifs de partitions ordonnés suivant l'ordre défini au paragraphe 6.1, alors pour le polynôme complément $\overline{f} = f/\widehat{f}$ il est suffisant pour poursuivre la détection précoce des vrais facteurs de $\overline{f}$ de ne tester les polynômes candidats vrais facteurs de $\overline{f}$, après reformulation du problème de remontée que parmi les successeurs du motif de partition

$$\mathrm{bloc}_r(\text{sous-niveau}(m) + 1, \text{sous-niveau}(m) + \ell)$$

pour le même ordre défini au paragraphe 6.1.

Preuve : Nous supposons que les candidats pour être vrais facteurs ont été testés en suivant l'ordre sur les motifs de partition défini au paragraphe 6.1. Soit m le motif de partition associé au premier vrai facteur trouvé.

Comme, d'après la proposition 6.26, le niveau est une fonction croissante pour l'ordre 6.24, cela signifie qu'ont été rejetés tous les ℓ'-uplets associés au motifs de partition de niveau ℓ', pour $\ell' < \ell$.

Comme, d'après la proposition 6.27, le sous-niveau est une fonction croissante à niveau ℓ fixé pour l'ordre 6.24, cela signifie qu'ont aussi été rejetés tous les ℓ'-uplets associés aux motifs de partition de niveau ℓ et de sous-niveau s, pour $s <$ sous-niveau (m).

Les motifs de partition associés aux candidats vrais facteurs de $\overline{f}$ de niveau strictement inférieur à ℓ ou de niveau égal à ℓ et de sous-niveau strictement inférieur à s apparaissent comme traces sur le complément de m de motifs de partitions déjà testés pour f. Les candidats vrais facteurs de $\overline{f}$ associés à ces motifs auraient donc dû déjà être détectés comme vrais facteurs de f (les vrais facteurs de $\overline{f}$ sont évidemment aussi des vrais facteurs de f) or, par hypothèse, ils ont été rejetés.

Il suffit donc pour poursuivre le processus de détection de vrais facteurs de $\overline{f}$ après reformulation de commencer directement par le plus petit motif de partition de niveau $\ell =$ niveau (m) et de sous-niveau sous-niveau (m) qui est

$$\text{bloc}_r\,(\text{sous-niveau}\,(m)+1, \text{sous-niveau}\,(m)+\ell)\,.$$

□

Détection de « probables » vrais facteurs

Au cours de l'algorithme de Collins et Encarnación, les termes $Z_{i,j+1}$ et $Y_{i,j+1}$ des développements q-adiques respectifs de $A_{i,1}$ et $B_{i,1}$ peuvent être séparément ou simultanément nuls (c'est notamment le cas si $\widetilde{U}_{i,j+1}$ est égal à 0 modulo q) dans (6.19).

Cela signifie respectivement que $A_{i,j}$ et $A_{i,j+1}$ ou que $B_{i,j}$ et $B_{i,j+1}$ sont égaux modulo q^{j+1} (par construction ils sont égaux modulo q^j). Comme leurs coefficients dans $\mathbb{Z}(q,j)$ sont représentés par les entiers dans $\{-(q^j-1)/2, \ldots, (q^j-1)/2\}$, cela peut signifier qu'ils sont effectivement égaux à des vrais facteurs sur $\mathbb{Z}[t]$. Dans un tel cas, nous considérons respectivement $A_{i,j+1}$ ou $B_{i,j+1}$ (selon les termes du développement q-adique qui sont nuls) comme de « *probables* » *vrais facteurs*. Sans pousser plus loin la remontée, nous testons par division de f sur $\mathbb{Z}[t]$ les polynômes associés respectivement à $A_{i,j+1}$ ou $B_{i,j+1}$ (voir le paragraphe 5.5.2), correspondant respectivement aux motifs de partitions $\text{bloc}_r\,(i,i)$ et $\text{bloc}_r\,(i+1,r)$.

Si un vrai facteur est trouvé, nous reformulons, lorsque c'est nécessaire, le problème de remontée selon le paragraphe 6.2.2.

Ce type de détection précoce convient particulièrement à la factorisation de facteurs unitaires (c'est le cas des résolvantes de Lagrange).

6.3 Conclusion

Nous avons fourni dans ce chapitre toutes les formules nécessaires à la reformulation du problème de remontée dans la méthode de remontée de Collins et Encarnación. Cette reformulation est moins coûteuse que de recalculer les facteurs modulaires à partir des facteurs modulo p des facteurs extraits.

Nous avons fourni au paragraphe 6.1 un ordre sur les motifs de partitions qui permet de parcourir itérativement l'ensemble des combinaisons possibles entre facteurs modulaires. Les propriétés de cet ordre nous permettent au paragraphe 6.2.3 d'éviter des calculs redondants après reformulation du problème de remontée.

Nous suggérons aussi de vérifier si les facteurs modulaires remontés sont de vrais facteurs dès qu'ils semblent converger modulo q^k et cela avant même d'avoir atteint une borne quelconque pour la détection de vrais facteurs.

Chapitre 7

Implantation des factorisations interactives en AXIOM

Nous exposons dans ce chapitre l'implantation réalisée en AXIOM du factorisateur interactif décrit au paragraphe 5.6. Le paragraphe 7.1 rappelle les principes généraux qui sous-tendent la factorisation partielle interactive ainsi que les choix techniques retenus pour AXIOM. Les paragraphes 7.2 et 7.3 précisent respectivement les spécificités de l'implantation de la factorisation modulaire et de la factorisation sur les entiers.

7.1 Principes généraux

Le chapitre 5 a rappelé le schéma général de factorisation de Berlekamp-Zassenhaus.

Un des objectifs principaux du travail présenté ici est la spécialisation des algorithmes généraux de factorisation des polynômes à coefficients dans $\mathbb{Z}$ au cas de la factorisation de polynômes particuliers appelés résolvantes. Ces résolvantes apparaissent dans le cadre de la théorie de Galois effective qui fournit des informations *a priori* sur le degré, le nombre de facteurs irréductibles, ou le groupe de Galois de ces facteurs (voir le chapitre 1 ou [5] et [96]). Or, les factorisateurs des logiciels de calcul formel ne sont pas conçus pour tirer parti de ces informations supplémentaires. Par ailleurs, en théorie de Galois effective, pour distinguer les différents types de factorisation possibles pour les résolvantes, l'information discriminante peut être obtenue au cours du processus de factorisation, bien avant d'avoir achevé la factorisation complète du polynôme, tout particulièrement s'il est de degré élevé (supérieur à 100). Ces informations discriminantes peuvent être de différentes natures :

1. irréductibilité du polynôme ;
2. existence, ou non, de facteurs d'un certain degré ;
3. nombre de facteurs ;
4. détermination de certains facteurs de degrés fixés (en vue de calculer leur groupe de Galois).

Il est donc souhaitable de disposer d'un factorisateur sur $\mathbb{Z}[t]$ qui rende accessible à tout moment de la factorisation les informations recueillies sur celle-ci et qui soit capable de n'effectuer que des factorisations partielles en fonction des informations cherchées. L'accès à ces informations doit se faire de façon automatique, le factorisateur étant destiné à être appelé par un programme de détermination du groupe de Galois, tel celui de N. RENNERT.

Le langage `AXIOM` a été retenu comme support de cette implantation en raison de son système de typage qui se prête particulièrement à la manipulation d'informations. Son successeur `Aldor` n'était pas suffisamment développé à l'époque de l'implantation (absence de bibliothèques algébriques). Toutes précautions ont été prises lors de l'implantation pour faciliter un futur basculement du code en `AXIOM` vers un code en `Aldor`.

Il nous est apparu intéressant que ce programme interactif fasse lui-même appel à une factorisation modulaire interactive.

Nous avons donc implanté deux factorisateurs interactifs. Un pour la factorisation sur $\mathbb{Z}[t]$ et un autre sur $\mathbb{F}_p[t]$.

Nous décrivons dans ce paragraphe les principes communs aux deux factorisateurs. Leurs spécificités et différences sont précisées dans les paragraphes 7.2 et 7.3.

Nous n'explicitons pas ici l'implantation des algorithmes : ce sont ceux du chapitre 5. Nous n'indiquons que la nature des informations accessibles en cours et à la fin de la factorisation. Nous donnons aussi la correspondance entre les variables internes et les notations du chapitre 5 dans l'implantation actuelle du factorisateur.

7.1.1 Un nouveau domaine `AXIOM`

Le *principe d'implantation* retenu pour les factorisateurs interactifs (modulaire ou sur les entiers) est le suivant : le polynôme à factoriser est stocké dans un nouveau domaine `AXIOM` [48] nommé :

- `InteractivelyFactoredModularPolynomialDomain` (abrégé ci-après en `IFMPD`) pour la factorisation modulaire,
- `InteractivelyFactoredPolynomialDomain` (abrégé ci-après en `IFPD`) pour la factorisation sur les entiers.

Dans ce seul paragraphe qui décrit les propriétés communes des deux factorisateurs, nous les noterons indifféremment avec l'abréviation `IF*PD`.

Ce domaine `IF*PD` correspond à une structure de données où sont stockés le polynôme ainsi que toutes les informations s'y rapportant.

L'accès aux données se fait par l'intermédiaire de fonctions d'interrogation. Elles rendent le domaine indépendant de la représentation choisie. Par exemple, pour `u` un élément de ce domaine `IF*PD`, l'appel

```
-> factorMinimumDegree(u)
```

retourne le degré minimum des facteurs irréductibles possibles de `u`. Les valeurs de ces données sont modifiées par des instructions de la forme

```
-> setFactorMinimumDegree!(u,3)
```

qui donne la valeur 3 à `factorMinimumDegree(u)` ou

```
-> updateFactorMinimumDegree!(u,3)
```
qui modifie en conséquence et de façon cohérente les autres données associées au polynôme `u`.

La factorisation s'effectue pas à pas par appel de la fonction qui modifie `u` par effet de bord
```
-> factor!(u)  .
```

Cette fonction effectue une étape supplémentaire dans le processus de factorisation et rend la main dès qu'elle a trouvé une information nouvelle, c'est-à-dire, dès qu'une donnée de `IF*PD` a été modifiée.

Par exemple, la factorisation complète s'effectue par

```
while not factorsKnown?(u) repeat factor!(u)
u
```

alors que la recherche de l'irréductibilité s'effectue par

```
while not irreducibleKnown?(u) repeat factor!(u)
irreducible?(u)
```

Le choix de ce mode de fonctionnement impose des contraintes d'implantation :

- il est nécessaire de stocker dans la structure de données `IF*PD` toutes les variables utilisées par les algorithmes se succédant dans le processus de factorisation ;
- la fonction de factorisation interactive doit pouvoir toujours, à partir de ces seules variables stockées dans les structure de données, déterminer à quelle étape en est la factorisation et pouvoir la poursuivre (jusqu'à factorisation complète).

Pour cela nous avons ordonné les opérations de la factorisation générale. Un pas dans l'algorithme de factorisation correspond donc à une modification de l'état de la structure de données. Nous faisons l'hypothèse, vérifiée dans notre implantation que le coût d'un appel de la fonction `factor!` est négligeable devant le coût d'une étape élémentaire de la factorisation (c'est en général le cas lorsque `u` est passé par variable à la fonction `factor!`).

7.1.2 Les données stockées

Les données stockées dans les éléments du domaine `IF*PD` sont de deux types distincts nommés local et global.

Les *données locales* de l'élément `u` de l'`IF*PD` correspondent à toutes les informations relatives à sa factorisation ainsi que toutes les variables utilisées par les différents algorithmes employés successivement dans le processus de factorisation. Ces variables locales sont propres à `u`.

Les *données globales* de `u` correspondent plutôt à des paramètres qui conditionnent et orientent les différents algorithmes de factorisation. Par exemple :
```
-> requiredFactorDegrees(u)
```
contient un ensemble de degrés de facteurs recherchés (lors de la factorisation partielle). Ces variables globales sont partagées avec les facteurs de `u` ainsi que les `IF*PD` dont `u` est facteur.

Par la suite nous appelons *variable* chacune des données stockées dans les `IF*PD` et accessibles par les diverses fonctions d'interrogation.

7.1.3 Factorisation non triviale

Au cours du processus de factorisation de l'`IF*PD u`, les facteurs trouvés, non nécessairement irréductibles, seront eux-mêmes du type `IF*PD`. La fonction `factor!` peut s'appliquer à chacun d'eux pour achever sa factorisation en facteurs irréductibles.

Lorsqu'une factorisation non triviale est trouvée, elle correspond à un certain vrai facteur f_1 de f (défini au paragraphe 5.5.2) ainsi qu'en même temps le quotient f_2 de f par ce facteur.

Pour chacun d'eux un nouvel `IF*PD` est créé, par exemple `v1` et `v2`, par les fonctions
`-> v1 := newFactor(u,f1)`
et
`-> v2 := newFactor(u,f2)` .
En ce cas `v1` et `u` partagent leurs variables globales. La modification d'une de ces variables sur l'un des facteurs va aussi modifier l'`IF*PD u` de départ. La variable `product(v1)` (tout comme `product(v2)`) pointe vers l'`IF*PD u`. Le caractère de facteur ou non d'un `IF*PD` s'obtient par `factor?(u)` qui sera vrai si la variable `product(u)` est affectée.

Les facteurs non triviaux `v1` et `v2` de `u` sont stockés dans la liste d'`IF*PD`
`-> factorization(u)` .
Dès que cette liste est non vide, la fonction `factor!` est appliquée aux `IF*PD` qu'elle contient. La factorisation se poursuit sur ces facteurs non triviaux et non sur l'`IF*PD u` jusqu'à ce qu'elle soit complète pour chacun d'eux. La fin de la factorisation est signalée par
`-> factorsKnown?(u)` .

Certaines nouvelles propriétés trouvées sur les facteurs (comme le nombre maximum ou minimum de facteurs, le degré maximum ou minimum de ces facteurs, l'ensemble des degrés possibles) sont transmises le cas échéant de façon cohérente du facteur `v` à
`-> product(v)` .
Cette transmission est subordonnée à la variable globale
`-> productInheritProperties(v)` .
Si cette transmission a modifié les précédentes valeurs connues, le drapeau
`-> inheritedFromFactor?(u)`
est positionné. Cela signifie qu'une information supplémentaire a été obtenue. Il peut permettre d'exclure certaines factorisations possibles pour le polynôme stocké dans `u`. Cette information à destination des programmes appelant `factor!` signifie qu'il est intéressant de vérifier si certaines factorisations possibles peuvent être exclues (une nouvelle information discriminante a été apportée).

Réciproquement, lorsqu'un nouveau facteur de `u` est créé il hérite, de façon cohérente de certaines autres propriétés du polynôme de départ (l'ensemble des degrés possibles, le nombre maximum de facteurs, la borne sur tous les facteurs). Cette transmission est subordonnée à la variable globale
`-> factorInheritProperties(u)` ,
elle ne sera effectuée que si cette variable booléenne est positionnée *vraie* (`true`).

La correspondance entre informations disponibles et variables internes est susceptible de modifications pour prendre en compte toute amélioration algorithmique de la factorisation. Il est donc préférable pour avoir une compatibilité avec les modifications futures du factorisateur de ne faire appel qu'à des fonctions standard qui utiliseront, elles, les informations présentes dans la structure de donnée.

Exemple : Plutôt que d'utiliser directement les informations recueillies sur la factorisation pour savoir si une partition `dp` est compatible avec l'état actuel de la factorisation de l'`IF*PD` `u`, la fonction
`-> compatibleDegreePartition?(dp,u)`
sera utilisée (elle retourne *vrai* tant que la partition du degré n'est pas incompatible avec les informations stockées).

7.1.4 La factorisation partielle

Les factorisateurs ont été entièrement conçus de façon à pouvoir effectuer des factorisations partielles (typiquement, recherche de tous les facteurs de degré inférieur à un degré donné ou de certains degrés choisis à l'avance).

L'utilisation de la factorisation partielle est subordonnée à la variable globale
`-> partialFactorization?(u)` .
En ce cas des variables globales sont consultées.

La factorisation des facteurs qui ne sont pas susceptibles (parce que leur degré minimal est supérieur par exemple) de fournir les facteurs recherchés n'est pas menée à son terme alors que les autres facteurs susceptibles d'être ou de contenir des facteurs recherchés de f sont eux complètement factorisés.

La **fonction** qui indique si tout `IF*PD` facteur de `u` est un facteur non désiré de `u` est stockée dans la variable globale
`-> unrequiredTrial(u)` .

7.2 Factorisation modulaire interactive

Nous décrivons dans ce paragraphe l'enchaînement ordonné d'opérations qui conduit à la factorisation modulaire complète.

La factorisation modulaire est appelée interactivement par la factorisation sur les entiers (paragraphe 7.3). Elle conserve cependant son intérêt et ses caractéristiques propres et peut être utilisée indépendamment.

Nous suivons les opérations décrites au paragraphe 5.4.1.

7.2.1 Initialisation

Elle se fait par
`-> u := new(f,p)`
où `f` est le polynôme à factoriser et `p` le modulo selon lequel tous les calculs seront menés.

Sont alors affectées les variables :
`-> polynomial(u)`
qui contient `f`,
`-> prime(u)`
qui contient `p`, ainsi que les variables ayant trait aux degrés des facteurs irréductibles (`factorMaximumDegree(u)`).

7.2.2 Mise sous forme unitaire

Si le polynôme n'est pas unitaire, il est d'abord factorisé en un `IFMPD` de degré zéro contenant le coefficient dominant de f modulo p ainsi qu'un `IFMPD` contenant le quotient unitaire de f par ce coefficient dominant.

7.2.3 Factorisation sans facteur carré

L'algorithme utilisé est celui de Musser [116] adapté au modulo p qui détermine, par pgcd, les polynômes $f_1, \ldots, f_k$ dont les indices $1 \leq i \leq k$ sont premiers avec p et tels que

$$f = f_1 f_2^2 f_3^3 \ldots f_k^k \quad .$$

Dès que `factor!` trouve un f_i différent de 1, pour i premier avec p, il place dans
`-> factorization(u)`
les `IFMPD`

- `ui` correspondant respectivement au facteur f_i avec `power(ui)` égal à i et
- `vi` correspondant au produit des polynômes de multiplicité strictement supérieure à i avec ceux dont p divise la multiplicité (`minimumCoprimeMultiplicity(vi)` est égal à $i+1$).

Une fois tous les f_i trouvés avec i premier avec p, il ne reste qu'un produit de facteurs dont la multiplicité est un multiple de p. Une fois calculée la racine p-ième de ce polynôme (factorisation non triviale), le processus est répété sur la racine.

Les variables concernées par la factorisation sans facteur carré sont
`-> squareFreeKnown?(u)`
et
`-> squareFree?(u)`
qui indiquent respectivement si la présence d'un facteur carré dans `u` est connue, et auquel cas, s'il a pour facteur des facteurs multiples.
Les autres variables utilisées sont
`-> minimumCoprimeMultiplicity(u)` ,

`-> musserPolynomialKnown?(u)` ,

`-> musserLeftPolynomial(u)`
et
`-> musserRightPolynomial(u)` .

7.2.4 Factorisation en degrés distincts

Arrivé à ce point de l'algorithme, les polynômes à factoriser sont sans facteur carré. Une information qui peut être obtenue avant même d'achever la factorisation modulo p en facteurs irréductibles est la factorisation de f sous la forme :

$$f = F_1 \dots F_k$$

où F_i est le produit des facteurs irréductibles de degré i de f.

Le *type* de la factorisation (la liste des degrés des facteurs irréductibles) est donc connu dès ce moment. La variable drapeau qui signale que les degrés des facteurs irréductibles sont connus est
`-> factorDegreesKnown?(u)` .

Nous avons retenu pour cette étape la méthode « pas de bébé/pas de géant » (*baby step/giant step*) de Kaltofen et Shoup [53] qui factorise les polynômes en degrés distincts en deux étapes :

- factorisation par intervalles de degrés : par pgcd avec les polynômes stockés dans `intervalPolynomials(u)` (ils sont calculés de façon paresseuse, c'est-à-dire à la demande, par la fonction `intervalPolynomial!(u,j)`), nous obtenons les produits des polynômes dont les degrés sont dans l'intervalle $[j\ell - (\ell - 1), j\ell]$;
- factorisation en degrés distincts : à partir de ces produits, sont extraits les facteurs en degrés distincts.

Lorsqu'un pgcd non trivial est obtenu, une factorisation non triviale de f vient d'être trouvée (voir paragraphe 7.2.6). Sinon, les degrés minimaux et maximaux des facteurs sont modifiés en conséquence.

À la fin de cette factorisation, les facteurs de `polynomial(u)` sont au pire des produits de polynômes de même degré et si un facteur est de degré égal au degré de ses facteurs, alors sa factorisation est achevée, il est irréductible.

7.2.5 Factorisation en degrés égaux

À cette étape de l'algorithme, les polynômes à factoriser sont des produits de facteurs de degrés tous égaux. La factorisation s'effectue de façon probabiliste avec le pgcd du polynôme avec un séparateur de McEliece [10] calculé par la méthode de Shoup [88] à partir d'un polynôme aléatoire.

La fin de la factorisation en facteurs irréductibles est signalée par la variable drapeau `factorsKnown?(u)`.

7.2.6 Factorisation non triviale

En fait, lors de la factorisation non triviale pour un vrai facteur `f1`, le factorisateur fait appel à la fonction
`-> newFactor(u,f1)`
qui calcule les restes modulo `f1` modulo `prime(u)` les différents polynômes internes (dont `babySteps(u)`, `giantSteps(u)`, `intervalPolynomials(u)`, `separator(u)`).

7.2.7 La factorisation partielle

Dans le cas de la factorisation modulaire la fonction appliquée à tout facteur `v` de `u` utilisée par défaut (placée dans `unrequiredTrial(u)`) est
`-> unrequiredFactor?(v)`
qui rejette les `IFMPD` dont les degrés n'appartiennent pas à
`-> requiredFactorDegrees(u)` .

7.2.8 Les informations disponibles

Sur la factorisation

`-> factorMaximumDegree(u)`
contient le degré maximum des facteurs de `u`.
`-> factorMinimumDegree(u)`
contient le degré minimum des facteurs de `u`.
`-> inheritedFromFactor?(u)`
est une variable drapeau indiquant qu'une des deux variables ci-dessus a été modifiée par une factorisation plus poussée de la factorisation non triviale stockée dans `factorization(u)`.
`-> polynomial(u)`
contient le polynôme à factoriser.
`-> polynomialKnown?(u)`
indique si la variable `polynomial(u)` a été affectée.
`-> power(u)`
est la puissance à laquelle `u` est représenté.
`-> irreducibleKnown?(u)`
indique si l'irréductibilité de `u` est connue auquel cas la réponse à cette question est dans `irreducible?(u)`.
`-> squareFreeKnown?(u)`
indique si l'absence de facteur carré parmi ceux de `polynomial(u)` est connue auquel cas la réponse à cette question est dans `squareFree?(u)`.

La variable `factor?(u)` indique si `u` est le facteur d'un `IFMPD` auquel cas, `product(u)` pointe sur cet `IFMPD`.

Si une factorisation non triviale de `u` a été trouvée, alors la liste d'`IFMPD` contenue dans `factorization(u)` est non vide et contient ces facteurs.

La liste des `IFMPD` facteurs de `u` trouvés jusque-là s'obtient récursivement par la fonction `factors(u)`.

Factorisation sans facteur carré

`-> minimumCoprimeMultiplicity(u)`
retourne la multiplicité minimum première avec `prime(u)` des facteurs de `u`.

Les variables suivantes sont utilisées par l'algorithme de factorisation sans facteur carré de Musser [116] et lui sont spécifiques.

`-> musserPolynomialsKnown?(u)`

`-> musserLeftPolynomial(u)`

`-> musserRightPolynomial(u)`

Variables internes liées à l'ordre de la factorisation

`-> factorDegreesKnown?(u)`
indique que la factorisation de `u` en facteurs de degrés distincts est achevée (conformément à la fonction dans `unrequiredTrial(u)` dans le cas d'une factorisation partielle).

`-> factorsKnown?(u)`
indique que la factorisation de `u` est achevée (conformément à la fonction stockée dans la variable `unrequiredTrial(u)` dans le cas d'une factorisation partielle).

Variables internes liées à la factorisation en degrés distincts

`-> babySteps(u)`
contient une liste de polynômes (déjà calculés) pour la factorisation en degrés distincts selon la méthode de Shoup [88].

`-> giantSteps(u)`
contient une liste de polynômes (déjà calculés) pour la factorisation en degrés distincts selon la méthode de Shoup [88].

`-> intervalPolynomials(u)`
contient une liste de polynômes (déjà calculés) pour la factorisation en degrés distincts selon la méthode de Shoup [88].

Variables internes liées à la factorisation en degrés égaux

`-> separator(u)`
contient le séparateur utilisé pour factoriser en degrés égaux.

`-> separationAttempsNumber(u)`
indique le nombre de tentatives de factorisation en degrés égaux avec le séparateur dans `separator(u)`.

Paramètres globaux communs à un `IFMPD` et ses facteurs

`-> prime(u)`
contient le premier modulo lequel sont effectués tous les calculs.
`-> productInheritProperties?(u)`
indique si les conséquences d'une factorisation modifiée d'un facteur se transmettent au polynôme produit (pointé par `product(u)`).
`-> factorInheritProperties?(u)`
indique si les propriétés déjà connues de `u` doivent être transmises à ses facteurs.
`-> unrequiredFactorFound?(u)`
est une variable drapeau qui indique qu'un facteur ne vérifiant pas les critères de la fonction `unrequiredFactor?` a été trouvé. Elle doit être remise à faux pour continuer la détection d'autres facteurs non cherchés.
`-> babyStepsNumber(u)`
indique le nombre de polynômes à calculer dans `babySteps(u)` et
`-> intervalPolynomials(u)` .

`-> giantStepsNumber(u)`
indique le nombre de polynômes à calculer dans `giantSteps(u)`.
`-> partialFactorization?(u)`
indique que la factorisation attendue n'est que partielle. Le factorisateur ne factorise que les facteurs `v` pour lesquels la fonction `unrequiredTrial(u)(v)` est fausse. La fonction utilisée par défaut est
`-> unrequiredFactor?(v)`
qui vérifie l'appartenance du degré de `v` à
`-> requiredFactorDegrees(u)`
qui contient l'ensemble des degrés des facteurs cherchés.

7.3 Factorisation interactive sur les entiers

Ce paragraphe décrit l'enchaînement ordonné d'opérations qui conduit à la factorisation complète.

7.3.1 Initialisation

Elle se fait par
`-> u := new(f)`
où $f \in \mathbb{Z}[t]$ est le polynôme à coefficients entiers à factoriser. Les données stockées dans l'`IFPD u` sont positionnées par `new` :
`-> polynomial(u)`
contient le polynôme f et
`-> degreeSet(u)`
reçoit l'ensemble des degrés possibles pour les facteurs de f, c'est-à-dire $\{0, \ldots, \deg_t(f)\}$.

Les variables ayant trait au nombre de facteurs irréductibles (`factorMaximumNumber(u)`, `factorMinimumNumber(u)`) sont aussi affectées en conséquence.

7.3.2 Détection de polynômes particuliers

Évoqués au paragraphe 5.3.1 les cas de polynômes particuliers n'ont pas encore fait l'objet d'une implantation mais ces tests doivent être placés au tout début de la factorisation.

7.3.3 Primitivité du polynôme

La primitivité des polynômes a été définie au paragraphe 5.2.1.

Les variables concernées sont

`-> primitiveKnown?(u)`

qui indique si la primitivité est connue, et

`-> primitive?(u)`

qui indique si le polynôme est primitif.

Si le polynôme est primitif, les facteurs de degré 0 sont exclus de l'ensemble

`-> degreeSet(u)`

des degrés possibles, sinon, la factorisation obtenue entre le contenu c de f et sa partie primitive f/c est stockée dans `factorization(u)`.

Comme la factorisation du contenu dans $\mathbb{Z}$ peut être très coûteuse (s'il est très gros), cette opération est subordonnée à la valeur de la variable globale

`-> factorContent?(u)` .

Le contenu ne sera factorisé que si elle est vraie sinon il sera conservé comme produit non factorisé de facteurs de degré 0.

7.3.4 Factorisation sans facteur carré

L'algorithme utilisé est celui de Yun (voir paragraphe 5.2.2) qui détermine, par pgcd, les polynômes $f_1, \ldots, f_k$ tels que

$$f = f_1 f_2^2 f_3^3 \ldots f_k^k \quad .$$

Dès que `factor!` trouve un f_i non égal à 1, il place dans `factorization(u)` les `IFPD`

- `ui` représentant les facteurs f_i avec `power(ui)` égal à i et
- `vi` correspondant à $f_{i+1}^{i+1} \ldots f_k^k$ avec `minimumMultiplicity(vi)` égal à $i+1$.

Les variables concernées sont

`-> squareFreeKnown?(u)`

et

`-> squareFree?(u)`

qui indiquent respectivement si la présence d'un facteur carré dans `u` est connue, et auquel cas, s'il a pour facteur des facteurs multiples.

Les autres variables utilisées sont

`-> minimumMultiplicity(u)` ,

`-> yunPolynomialKnown?(u)` ,

`-> yunLeftPolynomial(u)`
et
`-> yunRightPolynomial(u)` .

7.3.5 Critères d'irréductibilité

Les variables
`-> irreducibleKnown?(u)`
et
`-> irreducible?(u)`
indiquent si l'irréductibilité est connue et auquel cas si le polynôme est ou non irréductible. Cette variable est bien sûr positionnée fausse dès qu'un facteur non trivial de `u` est obtenu, c'est-à-dire dès que `factorization(u)` est non vide.

Avant d'attaquer la factorisation par un algorithme général, quelques critères rapides d'exécution permettent parfois de prouver très vite l'irréductibilité.

Les deux critères actuellement retenus sont le critère d'Eisenstein et le critère de Brillhart étendu par J. H. DAVENPORT et M. B. MONAGAN (voir paragraphe 5.3.2). L'intérêt de ce dernier critère est qu'il lie les valeurs entières prises par les polynômes à leur factorisation dans $\mathbb{Z}[t]$ permettant ainsi dans certains cas favorables de borner le nombre de facteurs irréductibles (le polynôme étant évalué en un entier suffisamment loin de ses racines, les facteurs irréductibles ne peuvent être en plus grand nombre que les facteurs premiers de la valeur du polynôme en cet entier).

Leur utilisation est respectivement subordonnée aux valeurs des variables globales
`-> useEisensteinCriterion?(u)`
et
`-> useBrillhartCriterion?(u)` .
Ils affectent les variables `einsensteinDone?(u)` et `brillhartDone?(u)` qui indiquent que ces tests ont été effectués pour `u` et le cas échéant `irreducibleKnown?(u)` et `irreducible?(u)`.

7.3.6 La décomposition fonctionnelle polynomiale

La recherche de décomposition polynomiale est subordonnée à la variable globale
`-> tryFunctionalDecomposition?(u)` .

C'est l'algorithme de [56] (voir le paragraphe 5.3.3) qui est actuellement utilisé par la fonction `factor!`.

Si un polynôme g tel que $f = g \circ h$ est trouvé, il est stocké sous forme d'`IFPD` dans
`-> decompositionLeftFactor(u)`
et le h correspondant (de type polynôme) dans

`-> decompositionRightFactor(u)` .

L'utilisation de cette décomposition pour la factorisation ultérieure en factorisant d'abord `w := decompositionLeftFactor(u)` est subordonnée à la variable globale
`-> useFunctionalDecomposition?(u)` .
Dès qu'une factorisation non triviale $g = g_1 g_2$ de `w` est trouvée (stockée dans la variable `factorization(w)`) elle conduit immédiatement à une factorisation partielle non triviale de f :

$$f = g_1(h)g_2(h) \quad ,$$

qui est stockée dans `factorization(u)`. Cette action est subordonnée à la variable globale `compositionInheritProperties?(w)`, le pointeur sur `u` étant stocké dans la variable `composition(w)`.

7.3.7 Les factorisations modulaires

Arrivés à ce point de l'algorithme, les polynômes à factoriser sont primitifs, sans facteur carré et leur irréductibilité n'a pu être prouvée par critère sinon ils sont concernés par l'une des étapes précédentes.

Seront effectuées encore autant de factorisations modulo certains premiers qu'indiqué par `modularFactorizationsNumber(u)`. Un « bon » premier impair p choisi, dans `prime(u)`, pour la factorisation modulaire en cours, stockée dans `currentModularFactorization(u)`, ne doit pas diviser le discriminant de f. Si ce n'est pas le cas, p est ajouté à la liste
`-> discriminantPrimes(u)`
de diviseurs du discriminant de f.

La factorisation modulo p est menée jusqu'à ce que les degrés des facteurs modulo p soient connus (c'est le cas avec la factorisation en degrés distincts). Cette factorisation modulaire de `currentModularFactorization(u)` est effectuée par le factorisateur qui fonctionne lui-même en mode interactif décrit au paragraphe 7.2.

Une fois connue la partition ϖ du degré de f mod p, l'ensemble $\mathcal{D}(\varpi)$ défini au paragraphe 5.5.1 est calculé et intersecté avec `degreeSet(u)`. La variable `degreeSet(u)` prend ensuite la valeur de cette intersection. Si l'ensemble `degreeSet(u)` se réduit au degré de f alors f est irréductible.

Sont aussi ajustées par la même occasion les informations sur le nombre maximum de facteurs de f (`factorMaximumNumber(u)`).

Cette partition du degré de f correspond aussi aux longueurs de cycles d'une permutation du groupe de Galois de f. Elle permet d'exclure de l'ensemble des groupes de Galois possibles pour f ceux qui n'ont pas de tels éléments. Ces partitions sont stockées dans `cycletypes(u)` (voir le paragraphe 5.5.1 pour leur utilisation probabiliste).

Les factorisations modulaires interactives sont itérées jusqu'à épuisement de
`-> modularFactorizationsNumber(u)`
et sont stockées, si la variable globale `keepModularFactorizations?(u)` est positionnée en ce but, dans `modularFactorizations(u)`. Dans tous les cas, la « meilleure » factorisation modulaire, au sens de celle qui va donner le plus petit nombre de facteurs irréductibles et

diminuer ainsi la difficulté de la combinaison des facteurs modulaires pour obtenir les facteurs sur $\mathbb{Z}[t]$, est stockée dans
`-> roughestModularFactorization(u)`
et la « pire » (le plus grand nombre de facteurs) dans
`-> finestModularFactorization(u)` .

7.3.8 Remontée de Hensel des facteurs modulaires

Une fois achevées les factorisations modulaires, la factorisation retenue dans
`-> roughestModularFactorization(u)`
est menée à son terme (obtention des facteurs modulaires et non simplement de leurs degrés) par le factorisateur modulaire interactif (voir le paragraphe 7.2).

Ce sont ces facteurs qui seront remontés modulo p^k par un lemme de Hensel effectif (voir les paragraphes 5.4.2 et 5.4.3).

Deux types de bornes sont calculées par `setFactorBounds!(u)` :

La borne sur un seul facteur : Elle est stockée dans
`-> singleFactorBound(u)` .

La borne sur tous les facteurs : Elle est stockée dans
`-> allFactorBound(u)` .

Nous avons implanté l'algorithme de remontée de Collins et Encarnación décrit au paragraphe 6.2. Il combine une remontée quadratique (implantée d'après Musser [79]) jusqu'à atteindre la plus grande puissance q du « bon » premier choisi qui rentre dans un mot machine, puis une remontée linéaire par rapport à q.

Les paramètres de cette remontée sont initialisés par
`-> initializeHenselLifting!(u,p,l,q)`
où `p` est `prime(roughestModularFactorization(u))`, `l` est la liste de ses facteur (simples) premiers entre eux deux à deux modulo `p` et `q` la plus grande puissance de `p` rentrant dans un mot machine. Ils pourraient l'être à partir de données obtenues par d'autres moyens que par la factorisation modulaire interactive. La remontée implantée est indépendante des calculs modulaires déjà effectués. Les polynômes modulaires fournis peuvent avoir une autre origine que le factorisateur modulaire interactif.

Une fois les paramètres initialisés, le modulo atteint par les opérations de remontée se trouve dans `linearHenselModulus(u)`. Le calcul de remontée des polynômes se fait à la demande par
`-> henselLiftedFactor!(u,i)`
où `i` est le i-ième facteur de `modularFactors(u)` remonté modulo
`-> linearHenselModulus(u)` .
Le numéro du dernier facteur remonté est stocké dans

`-> linearCurrentLiftedFactorNumber(u)` .
L'étape suivante dans la remontée : incrémenter
`-> linearCurrentLiftedFactorNumber(u)`
ou faire passer `linearHenselModulus(u)` de q^k à q^{k+1} (une fois que tous les facteurs ont été remontés modulo q^k) est provoquée par `henselLifting!(u)`.

7.3.9 Combinaison des facteurs

L'énumération des produits des parties des r facteurs modulaires pour en tirer les vrais facteurs sur $\mathbb{Z}[t]$ de f est l'étape exponentielle de la factorisation : $2^{\lfloor r/2 \rfloor}$ en utilisant une borne sur tous les facteurs, et 2^{r-1} en utilisant une borne sur un seul facteur.

Nous avons cherché à éviter tous les calculs inutiles : lorsque le degré du polynôme réconcilié (calculé à partir des degrés de `modularFactors(u)`, sans même remonter le facteur modulo `linearHenselModulus`) n'appartient pas à l'ensemble des degrés possibles de f (dans `degreeSet(u)`) le produit n'est pas même calculé. Si le degré est celui d'un facteur possible, l'évaluation à 0 du polynôme est effectuée ce qui permet de rejeter tous les polynômes dont le terme constant ne divise pas celui de f.

Lorsqu'un facteur modulaire est inchangé lorsqu'il est remonté de q^k à q^{k+1}, il est considéré comme vrai facteur probable et son numéro est stocké dans `probableTrueFactor(u)`. À l'appel suivant de `factor!` il est essayé comme diviseur de f (voir le paragraphe 7.3.8). En cas de succès une factorisation non triviale est trouvée (voir le paragraphe 7.1.3) sinon, `probableTrueFactor(u)` est remis à 0.

Lors de la recherche de vrais facteurs pour f, le motif de partition associé à la combinaison des facteurs est stocké dans `recombinationPattern(u)` et le dernier sous-niveau atteint dans `recombinationSubLevel(u)`.

Lorsque qu'un vrai facteur g est détecté, correspondant au motif de partition
`-> recombinationPattern(u)` ,
un nouveau facteur est créé (par la fonction `ug := newPrimitiveFactor(u,w,g)`) où `w` est le motif de partition correspondant. Si le modulo atteint, `linearHenselModulus(u)`, est supérieur à la borne b requise pour g sur tous ses facteurs, qui dépend de la méthode retenue pour le coefficient dominant, (voir section 7.3.8, b correspondant à `allFactorBound(ug)` ou à `singleFactorBound(ug)` si la technique de borne sur un seul facteur est utilisée) alors g est irréductible si les combinaisons ont été effectuées dans l'ordre du paragraphe 6.1 (voir le paragraphe 5.5.3). Sinon, le problème de remontée est reformulé (voir le paragraphe 6.2.2) pour permettre de poursuivre cette remontée jusqu'à dépasser les bornes nécessaires pour prouver (ou non) l'irréductibilité de ce facteur par de futures itérations de `factor!`.

Le quotient de f par ce vrai facteur g est aussi un vrai facteur qui correspond au complément du motif de partition `w`. Il n'est en général pas irréductible (c'est le produit des facteurs irréductibles restant) sauf si le modulo atteint est supérieur à sa borne sur tous les facteurs et que le nombre de ses facteurs modulaires est inférieur ou égal au niveau de combinaison atteint.

Quand il ne peut être prouvé irréductible, la reformulation du problème de remontée est aussi effectuée pour lui. Cependant, dans le cas où la remontée n'a pas à être poussée plus haut, mais qu'il suffit de reprendre les combinaisons des facteurs modulaires correspon-

dant à ce facteur, les combinaisons ne sont pas redémarrées à partir du début (certaines ont déjà été effectuées pour f) mais à partir du plus petit motif de partition de niveau `recombinationLevel(u)` et de sous-niveau `subLevel(u)` (voir la proposition 6.41 du paragraphe 6.2.3).

7.3.10 La factorisation partielle

En fait, toutes les opérations décrites ci-dessus (factorisations modulaires, remontées à la Hensel, combinaison) peuvent être menées pour des polynômes qui ne sont pas nécessairement irréductibles (il suffit qu'ils soient premiers entre eux deux à deux). Le factorisateur à donc été entièrement conçu de façon à pouvoir effectuer des factorisations partielles (typiquement, recherches de tous les facteurs de degré inférieur à un degré donné ou de certains degrés choisis à l'avance).

En plus de la fonction `unrequiredTrial(u)`, la fonction `factor!` utilise aussi une fonction `unrequiredModularTrial(u)` qui rejette les facteurs modulaires qui ne sont pas susceptibles de fournir les facteurs recherchés. Cette fonction est passée au factorisateur modulaire interactif qui retourne donc une factorisation partielle. Le produit des facteurs modulaires qui ne sont pas susceptibles de fournir les facteurs recherchés est placé en tête de la liste `l` des facteurs modulaires (en portant le modulo du coefficient dominant le cas échéant) lors de l'initialisation de la remontée par

`-> initializeHenselLifting!(u,p,l,q)`

(voir le paragraphe 7.3.8) alors que les autres facteurs susceptibles d'être des facteurs modulaires de vrais facteurs recherchés de f sont eux complètement factorisés par le factorisateur modulaire et placés dans la suite de cette liste `l`. En ce cas la variable

`-> skipFirstModularFactor?(u)`

est positionnée pour que les motifs de partitions qui prennent en compte ce facteur non irréductible soient évités.

La fonction appliquée à tout facteur `v` de `u` utilisée par défaut (placée dans la variable `unrequiredTrial(u)`) est

`-> unrequiredFactor?(v)`

qui rejette les `IFPD` dont les degrés des facteurs modulaires ne sont pas tous dans

`-> requiredFactorModularDegreePartitions(u)` ,

lorsque cette variable n'est pas vide, ou dont aucun des degrés des facteurs modulaires n'est dans `requiredFactorModularDegrees(u)` ou bien, lorsque les degrés possibles pour les facteurs ne sont pas dans `requiredFactorDegrees(u)`.

La fonction appliquée à tout facteur `mv` d'une factorisation modulaire `mu` de `u` utilisée par défaut (placée dans `unrequiredModularTrial(u)`) est

`-> unrequiredFactor?(mv)`

qui rejette les `IFMPD` dont les degrés n'appartiennent pas à

`-> requiredFactorDegrees(mu)`

des facteurs modulaires `mu`, c'est-à-dire en fait

`-> requiredFactorModularDegrees(u)` .

Il s'agit de la même fonction que celle sur les factorisations modulaires du paragraphe 7.2.7.

7.3.11 Les informations disponibles

Sur le groupe de Galois

`-> cycleTypes(u)`
retourne une liste de longueurs de cycles de permutations appartenant au groupe de Galois de `polynomial(u)`. Elles sont obtenues à partir de multiples factorisations modulaires en degrés distincts (voir théorème 5.23).

Sur la factorisation

`-> factorMinimumNumber(u)`
contient le nombre minimum de facteurs irréductibles.
`-> factorMaximumNumber(u)`
contient le nombre maximum de facteurs irréductibles.
`-> degreeSet(u)`
contient l'ensemble des degrés possibles pour les facteurs irréductibles de `u`.
`-> irreducibleFactorDegrees(u)`
contient les degrés de facteurs irréductibles de `polynomial(u)` trouvés jusque là.
`-> inheritedFromFactor?(u)`
est une variable drapeau indiquant qu'une des quatre variables ci-dessus a été modifiée par une factorisation plus poussée de la factorisation non triviale présente dans la variable `factorization(u)`.
`-> polynomial(u)`
contient le polynôme à factoriser.
`-> polynomialKnown?(u)`
indique si la variable `polynomial(u)` a été affectée.
`-> power(u)`
est la puissance à laquelle `u` est représenté.
`-> primitive?(u)`
indique si le polynôme est primitif quand cette primitivité a été déterminée (voir la description de `primitiveKnown?(u)` au paragraphe 7.3.3).
`-> irreducibleKnown?(u)`
indique si l'irréductibilité de `u` est connue auquel cas la réponse à cette question est dans
`-> irreducible?(u)` .

`-> squareFreeKnown?(u)`
indique si l'absence de facteur carré parmi ceux de `polynomial(u)` est connue auquel cas la réponse à cette question est dans `squareFree?(u)`.

La variable `factor?(u)` indique si `u` est le facteur d'un `IFPD` auquel cas, `product(u)` pointe sur cet `IFPD`.

Si une factorisation non triviale de `u` a été trouvée, alors la liste d'`IFPD` contenue dans `factorization(u)` est non vide et contient ces facteurs.

La liste des `IFPD` facteurs de `u` trouvés jusque-là s'obtient récursivement par la fonction `factors(u)`.

Bornes sur les coefficients

`-> allFactorBound(u)`
lorsqu'elle n'est pas nulle, contient une borne sur la taille maximale des coefficients de tous les facteurs de `u`.

`-> singleFactorBound(u)`
lorsqu'elle n'est pas nulle, contient une borne sur la taille maximale des coefficients d'un facteur non trivial de `u`.

Factorisation sans facteur carré

`-> minimumMultiplicity(u)`
retourne la multiplicité minimum des facteurs de `u`.

Les variables suivantes sont utilisées par l'algorithme de factorisation sans facteur carré de Yun [116] et lui sont spécifiques.

`-> yunPolynomialsKnown?(u)`

`-> yunLeftPolynomial(u)`

`-> yunRightPolynomial(u)`

Variables internes liées à l'ordre de la factorisation

`-> primitiveKnown?(u)`
indique si le caractère primitif de `polynomial(u)` est connu.

`-> composedKnown?(u)`
indique si la recherche de décomposition polynomiale est achevée.

`-> eisensteinDone?(u)`
indique si le test d'irréductibilité de Einsenstein a été effectué.

`-> brillhartDone?(u)`
indique si le test d'irréductibilité de Brillhart a été effectué.

`-> modularFactorizationsNumber(u)`
indique le nombre de factorisations modulaires qui restent à effectuer. Une fois ce nombre tombé à zéro, les étapes de remontée et de combinaison commencent.

`-> primes(u)`
Si cette liste de premiers est non vide, elle impose les premiers modulo lesquels sont effectuées les factorisations modulaires. Le nombre de premiers restant dans `primes(u)` prend le pas sur `modularFactorizationsNumber(u)`.

`-> factorsKnown?(u)`
indique que la factorisation de `u` est achevée (conformément aux fonctions contenues dans les variables `unrequiredTrial(u)` et `unrequiredModularTrial(u)` dans le cas d'une factorisation partielle).

Décomposition polynomiale fonctionnelle

`-> decompositionLeftFactor?(u)`
indique si l'`IFPD` actuellement factorisé est le facteur à gauche d'une décomposition polynomiale. En ce cas, la variable `composition(u)` pointe sur l'`IFPD` dont `u` est le facteur à gauche.

Si `u` est décomposable (appel de `composed?(u)`) alors le facteur à gauche trouvé de `u` se trouve sous forme d'`IFPD` dans
`-> decompositionLeftFactor(u)`
et le facteur à droite se trouve sous forme de polynôme dans
`-> decompositionRightFactor(u)` .

Variables internes liées à la factorisation modulaire

`-> currentModularFactorization(u)`
contient la factorisation modulaire en cours (modulo `prime(u)`). Cette variable est vidée une fois la factorisation modulaire menée jusqu'à une factorisation en degrés distincts, totale ou partielle.
`-> prime(u)`
contient le premier modulo lequel a été effectuée la dernière factorisation modulaire si `currentModularFactorization(u)` est vide, ou bien le premier modulo lequel elle est effectuée si une factorisation modulaire est en cours.
`-> finestModularFactorization(u)`
contient la factorisation modulaire qui jusque là factorise `u` en le plus de facteurs modulaires possibles (pour une utilisation future avec des idées du types de celles exposées dans [43]).
`-> roughestModularFactorization(u)`
contient la factorisation modulaire qui jusque là factorise `u` en le moins de facteurs modulaires possibles, c'est à la fin des factorisations modulaires (voir l'utilisation de la variable `modularFactorizationsNumber(u)` au paragraphe 7.3.7) celle qui sera complètement achevée et retenue pour les opérations de remontée.
`-> modularFactorizations(u)`
stocke la liste de toutes les factorisations modulaires entreprises lorsque la valeur de la variable `keepModularFactorizations?(u)` le demande.

Variables internes liées à la remontée

`-> skipFirstModularFactor?(u)`
n'est utilisé qu'en factorisation partielle et signifie que le premier des facteurs modulaires de `modularFactors(u)` contient un produit de facteurs inintéressants et qu'il est inutile d'essayer de les réconcilier avec les facteurs modulaires restants.
`-> henselPrime(u)`
contient le premier modulo lequel s'effectuent les opérations de remontée à partir des facteurs stockés dans `modularFactors(u)`.

`-> quadraticHenselModulus(u)`
contient le modulo atteint par la remontée quadratique qui sert de base à la remontée linéaire.
`-> linearHenselModulus(u)`
contient le modulo atteint par la remontée.

Le `i`-ième facteur remonté modulo `linearHenselModulus(u)` correspond au `i`-ième élément stocké dans
`-> modularFactors(u)` .
Il s'obtient par appel à la fonction `linearLiftedFactor!(u,i)`. Celle-ci fonctionne sur un mode paresseux et n'effectue les calculs nécessaires que lorsqu'ils sont utilisés (le numéro du dernier facteur calculé est stocké dans `linearCurrentLiftedFactorNumber(u)`).

Nous ne donnons pour les variables suivantes que leurs correspondances avec les notations du paragraphe 6.2.

`-> modularFactors(u)`	$A_i, \quad 1 \leq i \leq r$
`-> modularFactorProducts(u)`	$B_i, \quad 1 \leq i \leq r-1$
`-> quadraticLiftedFactors(u)`	$A_{i,1}, \quad 1 \leq i \leq r-1$
`-> quadraticLiftedFactorProducts(u)`	$B_{i,1}, \quad 1 \leq i \leq r-1$
`-> quadraticProductPolynomial(u)`	voir [79]
`-> quadraticRightCofactors(u)`	$S_i, \quad 1 \leq i \leq r-1$
`-> quadraticLeftCofactors(u)`	$T_i, \quad 1 \leq i \leq r-1$
`-> linearFactorExpansionTerms(u)`	$Z_{i,j+1}, \quad 1 \leq i \leq r-1$
`-> linearFactorProductExpansionTerms(u)`	$Y_{i,j+1}, \quad 1 \leq i \leq r-1$
`-> linearLastButOneHenselModulus(u)`	q^{j-1}
`-> linearLastButTwoHenselModulus(u)`	q^{j-2}
`-> linearLastHenselModulus(u)`	q^{j}
`-> linearHenselModulus(u)`	q^{j+1}
`-> linearLastLiftedFactors(u)`	$A_{i,j}, \quad 1 \leq i \leq r-1$
`-> linearLastLiftedFactorProducts(u)`	$B_{i,j}, \quad 1 \leq i \leq r-1$
`-> linearLiftedFactors(u)`	$A_{i,j+1}, \quad 1 \leq i \leq r-1$
`-> linearLiftedFactorProducts(u)`	$B_{i,j+1}, \quad 1 \leq i \leq r-1$
`-> linearPolynomialQuotient(u)`	R_{j+1}
`-> linearPolynomialRemainder(u)`	$Y_{0,j+1}$
`-> linearRightHandSidePolynomials(u)`	$U_{i,j+1}, \quad 1 \leq i \leq r-1$

`-> currentRecombinationLevelFactorMinimumDegree(u)`
contient le plus petit degré rencontré pour les combinaisons de facteurs modulaires du niveau de combinaison courant, jusqu'au motif de partition dans `recombinationPattern(u)`. Une fois le niveau achevé, l'ensemble des degrés possibles `degreeSet(u)` est modifié en conséquence et, pour le niveau de combinaison suivant, cette valeur est stockée dans
`-> lastRecombinationLevelFactorMinimumDegree(u)` .

`-> lastRecombinationLevelFactorMinimumDegree(u)`
contient le degré minimum atteint pour les combinaisons du niveau précédent. Si la valeur de `linearHenselModulus(u)` est suffisamment grande (voir le paragraphe 5.4.3), alors aucun facteur irréductible n'est de degré plus petit que cette variable.

`-> probableTrueFactor(u)`
est une variable drapeau qui donne le numéro du dernier facteur qui n'a pas été modifié par le passage de `linearHenselModulus(u)` à une puissance supérieure. Les coefficients de ce facteur modulaire sont soupçonnés de correspondre en fait aux coefficients d'un vrai facteur de `polynomial(u)`.
`-> recombinationLevel(u)`
contient le niveau courant de combinaison (nombre de facteurs multipliés ensemble).
`-> recombinationPattern(u)`
contient le motif de partition atteint jusque là. Les indices des coordonnées égales à 1 correspondent en croissant aux numéros des facteurs de `modularFactors(u)`. Il est vide à la fin du niveau courant.
`-> recombinationSubLevel(u)`
contient le sous-niveau du dernier motif de partition calculé.

Autour du discriminant

`-> discriminantKnown?(u)`
indique si le discriminant de `polynomial(u)` est connu et auquel cas, il est stocké dans `discriminant(u)`.

Les premiers trouvés comme diviseurs du discriminant de `polynomial(u)`, lors des tentatives de factorisations modulaires, sont dans la liste
`-> discriminantPrimes(u)` .

`-> squareDiscriminantKnown?(u)`
indique s'il est connu si le discriminant de `polynomial(u)` est un carré auquel cas la réponse à cette question est dans `squareDiscriminant?(u)`.

Paramètres globaux communs à un `IFPD` et ses facteurs

`-> compositionInheritProperties?(u)`
indique si les conséquences de la factorisation d'un facteur à gauche d'une décomposition polynomiale se transmettent au polynôme composé (pointé par `composition(u)`).
`-> decompositionInheritProperties?(u)`
indique si les propriétés requises pour `u` sont traduites pour tout facteur à gauche de décomposition polynomiale qui puisse être trouvé (par exemple, la factorisation du facteur à gauche sera aussi partielle quand celle de `u` l'est).
`-> computeInheritedFactorCycleTypes?(u)`
indique si les longueurs de cycles des facteurs de `u` doivent être calculées à partir des factorisations modulaires en degrés distincts déjà connues qui sont stockées dans la variable `modularFactorizations(u)`.
`-> factorContent?(u)`
indique si le contenu des polynômes non primitifs doit être factorisé (factorisation sur les entiers).
`-> productInheritProperties?(u)`
indique si les conséquences d'une factorisation modifiée d'un facteur se transmettent au

polynôme produit (pointé par `product(u)`).

`-> factorInheritProperties?(u)`
indique si les propriétés déjà connues de `u` doivent être transmises à ses facteurs (par exemple `allFactorBound(u)`, `factorMaximumNumber(u)`, etc.).

`-> keepModularFactorizations?(u)`
indique si les différentes factorisations modulaires entreprises doivent être stockées dans `modularFactorizations(u)`.

`-> linearHenselLifting?(u)`
indique si la remontée linéaire doit être utilisée.

`-> quadraticHenselLifting?(u)`
indique si la remontée quadratique doit être utilisée.

`-> sortModularFactorsByRequirements?(u)`
indique si les facteurs modulaires dans `modularFactors(u)` doivent être classés (à l'exception du premier dans le cas d'une factorisation partielle) selon qu'ils sont plus susceptibles de fournir des facteurs répondant aux contraintes des variables

`-> requiredFactorModularDegreePartitions(u)`
et
`-> requiredFactorModularDegrees(u)` .

`-> tryFunctionalDecomposition?(u)`
indique si la recherche d'une décomposition polynomiale doit être entreprise.

`-> unrequiredFactorFound?(u)`
est une variable drapeau qui indique qu'un facteur ne vérifiant pas les critères de la fonction `unrequiredFactor?` a été trouvé. Elle doit être remise à `false` pour continuer la détection d'autres facteurs non cherchés.

`-> useBrillhartCriterion?(u)`
indique si le critère d'irréductibilité de Brillhart doit être essayé.

`-> useEisensteinCriterion?(u)`
indique si le critère d'irréductibilité d'Eisenstein doit être essayé.

`-> useFunctionalDecomposition?(u)`
indique si la factorisation du facteur à gauche doit être entreprise lorsqu'a été trouvée une décomposition fonctionnelle.

`-> useSingleFactorBound?(u)`
indique si la technique de recherche de factorisation à partir d'une borne sur un seul facteur doit être utilisée. Dans l'implantation actuelle, cette technique est incompatible avec la factorisation partielle car nous ne pouvons pas assurer que le vrai facteur (ou l'un de ses facteurs irréductibles) dont l'existence est garantie par la borne ne se trouve pas dans le produit des polynômes considérés comme inintéressants par la factorisation partielle.

`-> partialFactorization?(u)`
indique que la factorisation attendue n'est que partielle. Le factorisateur ne factorise que les facteurs `v` pour lesquels la fonction `unrequiredTrial(u)(v)` est fausse. Lorsqu'il utilise la factorisation partielle du factorisateur interactif, il lui transmet la fonction contenue dans la variable
`-> unrequiredModularTrial(u)` .

La fonction utilisée par défaut est `unrequiredFactor?(v)`, qui vérifie l'appartenance du degré de `v` à
`-> requiredFactorDegrees(u)`
qui contient l'ensemble des degrés des facteurs cherchés et lorsque ces listes sont non vides respectivement l'appartenance de la partition de son degré en degrés de facteurs modulaires à
`-> requiredFactorModularDegreePartitions(u)`
qui contient l'ensemble des partitions en degrés de facteurs modulaires cherchés des degrés des facteurs irréductibles ou simplement l'appartenance des degrés de ses facteurs modulaires à
`-> requiredFactorModularDegrees(u)`
qui contient l'ensemble des degrés cherchés pour les facteurs modulaires.

Remarque : L'utilisation de ces deux dernières variables n'est vraiment pertinente que lorsque `polynomial(u)` est factorisé modulo un unique premier fixé d'avance par `primes(u)`.

Annexe A

Calculs de résolvantes

Nous avons regroupé dans cet annexe des exemples illustrant nos deux algorithmes de calcul de résolvantes. Les paragraphes A.1 et A.2 traitent le calcul symbolique de la résolvante $\mathcal{L}_{x_1+x_2+x_3,x^5+ax+b} = \mathcal{L}_{x_3+x_4+x_5,x^5+ax+b}$. Elle est calculée respectivement par les méthodes des chapitres 3 et 4. Nous n'avons pas choisi cette résolvante pour son intérêt en théorie de Galois effective. Sa factorisation n'apporte pas plus d'informations sur le groupe de Galois que la factorisation de $\mathcal{L}_{x_1+x_2,x^5+ax+b}$. Mais les résultats de calculs symboliques avec des invariants de plus grande arité ou pour des polynômes moins creux que $x^5 + ax + b$ occuperaient de nombreuses pages. Nous avons choisi la « petite » résolvante $\mathcal{L}_{x_1+x_2+x_3,x^5+ax+b}$ car l'intérêt de ne conserver, par les calculs modulo y^s, que les seuls coefficients indispensables à son calcul apparaît déjà, même s'il y est moins spectaculaire que pour des résolvantes plus utiles.

A.1 Calcul récursif de résolvante

Nous prenons :

$$f(x) = x^5 + a\ x + b \qquad \text{et} \qquad \Theta = x_1 + x_2 + x_3 \quad .$$

Avec les notations du chapitre 3, nous avons :

$$\begin{aligned} \text{Récip}\left(\mathcal{L}_{x_1,f}\right)(t) &= \text{Récip}\left(f\right)(t) \\ &= 1 + a\ t^4 + b\ t^5 \end{aligned}$$

soit, immédiatement,

$$\begin{aligned} \mathcal{L}_{x_1,f}(t) &= f(t) \\ &= t^5 + a\ t + b \quad . \end{aligned}$$

La résolvante $\mathcal{L}_{x_1+x_2,f}$ s'obtient par :

$$\text{Récip}\left(\mathcal{L}_{x_1+x_2,f}\right)(t) = \sqrt[2]{\text{Rés}_y\left(f(y), \left\lceil \frac{\widetilde{N}_1(t,y)}{\widetilde{D}_1(t,y)} \right\rceil^t\right) \bmod t^{s_1+1}}$$

avec, pour $j_1 = n-1 = 4$ et $s_1 = n(n-1)/2 = 10$,

$$\begin{aligned}\widetilde{N}_1(t,y) &= (1-yt)^n \times \text{Récip}\,(\mathcal{L}_{x_1,f})\left(\frac{t}{1-yt}\right) \bmod t^{j_1+1}\\ &= 1 - 5\ y\ t + 10\ y^2\ t^2 - 10\ y^3\ t^3 + \left(5\ y^4 + a\right)\ t^4 \mod t^5\end{aligned}$$

$$\text{et} \qquad \widetilde{D}_1(t,y) = 1 - 2\ y\ t$$

$$\left[\frac{\widetilde{N}_1(t,y)}{\widetilde{D}_1(t,y)}\right]^t = 1 - 3\ y\ t + 4\ y^2\ t^2 - 2\ y^3\ t^3 + \left(y^4 + a\right)\ t^4$$

$$\text{Rés}_y\left(f(y), \left[\frac{\widetilde{N}_1(t,y)}{\widetilde{D}_1(t,y)}\right]^t\right) = 1 - 6\ a\ t^4 - 22\ b\ t^5 + a^2\ t^8 + 74\ a\ b\ t^9 + 119\ b^2\ t^{10} \bmod t^{11}$$

d'où

$$\text{Récip}\,(\mathcal{L}_{x_1+x_2,f})\,(t) = 1 - 3\ a\ t^4 - 11\ b\ t^5 - 4\ a^2\ t^8 + 4\ a\ b\ t^9 - b^2\ t^{10}$$

soit

$$\mathcal{L}_{x_1+x_2,f}(t) = t^{10} - 3\ a\ t^6 - 11\ b\ t^5 - 4\ a^2\ t^2 + 4\ a\ b\ t - b^2 \quad .$$

Le calcul direct, sans modulo, aurait demandé :

$$\mathcal{L}_{x_1+x_2,f} = \sqrt[2]{\text{Rés}_y\left(f(y), \left\lfloor\frac{N_1(t,y)}{D_1(t,y)}\right\rfloor_t\right)}$$

avec

$$\begin{aligned}N_1(t,y) &= \mathcal{L}_{x_1,f}\,(t-y)\\ &= t^5 - 5\ y\ t^4 + 10\ y^2\ t^3 - 10\ y^3\ t^2 + \left(5\ y^4 + a\right)\ t - y^5 - a\ y + b\end{aligned}$$

$$\text{et} \qquad D_1(t,y) = t - 2\ y$$

$$\left\lfloor\frac{N_1(t,y)}{D_1(t,y)}\right\rfloor_t = t^4 - 3\ y\ t^3 + 4\ y^2\ t^2 - 2\ y^3\ t + y^4 + a$$

$$\begin{aligned}\text{Rés}_y\left(f(y), \left\lfloor\frac{N_1(t,y)}{D_1(t,y)}\right\rfloor_t\right) &= t^{20} - 6\ a\ t^{16} - 22\ b\ t^{15} + a^2\ t^{12} + 74\ a\ b\ t^{11} + 119\ b^2\ t^{10}\\ &\quad + 24\ a^3\ t^8 + 64\ a^2\ b\ t^7 - 82\ a\ b^2\ t^6 + 22\ b^3\ t^5 + 16\ a^4\ t^4\\ &\quad - 32\ a^3\ b\ t^3 + 24\ a^2\ b^2\ t^2 - 8\ a\ b^3\ t + b^4\\ &= \left(t^{10} - 3\ a\ t^6 - 11\ b\ t^5 - 4\ a^2\ t^2 + 4\ a\ b\ t - b^2\right)^2\end{aligned}$$

La résolvante $\mathcal{L}_{x_1+x_2+x_3,f}$ s'obtient par :

$$\text{Récip}\left(\mathcal{L}_{x_1+x_2+x_3,f}\right)(t) = \sqrt[3]{\text{Rés}_y\left(f(y), \left\lceil \frac{\widetilde{N}_2(t,y)}{\widetilde{D}_2(t,y)} \right\rceil^t\right) \bmod t^{s_2+1}}$$

avec, pour $j_2 = (n-1)(n-2)/2 = 6$ et $s_2 = n(n-1)(n-2)/6 = 10$,

$$\begin{aligned}\widetilde{N}_2(t,y) &= (1-yt)^{n(n-1)/2} \times \text{Récip}\left(\mathcal{L}_{x_1+x_2,f}\right)\left(\frac{t}{1-yt}\right)\\ &\quad \times (1-3yt) \bmod t^{j_2+1}\\ &= 1 - 13\ y\ t + 75\ y^2\ t^2 - 255\ y^3\ t^3 + \left(570\ y^4 - 3\ a\right)\ t^4\\ &\quad + \left(-882\ y^5 + 27\ a\ y - 11\ b\right)\ t^5\\ &\quad + \left(966\ y^6 - 99\ a\ y^2 + 88\ b\ y\right)\ t^6 \bmod t^7\end{aligned}$$

$$\begin{aligned}\text{et} \qquad \widetilde{D}_2(t,y) &= (1-2yt)^n \times \text{Récip}\left(\mathcal{L}_{x_1,f}\right)\left(\frac{t}{1-2yt}\right)\\ &= 1 - 10\ y\ t + 40\ y^2\ t^2 - 80\ y^3\ t^3 + \left(80\ y^4 + a\right)\ t^4\\ &\quad + \left(-32\ y^5 - 2\ a\ y + b\right)\ t^5\end{aligned}$$

$$\begin{aligned}\left\lceil \frac{\widetilde{N}_2(t,y)}{\widetilde{D}_2(t,y)} \right\rceil^t &= 1 - 3\ y\ t + 5\ y^2\ t^2 - 5\ y^3\ t^3 - 4\ a\ t^4\\ &\quad + \left(-10\ y^5 - 8\ a\ y - 12\ b\right)\ t^5 + \left(-30\ y^6 - 30\ a\ y^2 - 29\ b\ y\right)\ t^6\end{aligned}$$

$$\text{Rés}_y\left(f(y), \left\lceil \frac{\widetilde{N}_2(t,y)}{\widetilde{D}_2(t,y)} \right\rceil^t\right) = 1 - 9\ a\ t^4 + 33\ b\ t^5 + 15\ a^2\ t^8 - 210\ a\ b\ t^9 + 360\ b^2\ t^{10} \bmod t^{11}$$

d'où

$$\text{Récip}\left(\mathcal{L}_{x_1+x_2+x_3,f}\right)(t) = 1 - 3\ a\ t^4 + 11\ b\ t^5 - 4\ a^2\ t^8 - 4\ a\ b\ t^9 - b^2\ t^{10}$$

soit

$$\mathcal{L}_{x_1+x_2+x_3,f}(t) = t^{10} - 3\ a\ t^6 + 11\ b\ t^5 - 4\ a^2\ t^2 - 4\ a\ b\ t - b^2 \quad .$$

Le calcul direct, sans modulo, aurait demandé :

$$\mathcal{L}_{x_1+x_2+x_3,f} = \sqrt[3]{\text{Rés}_y\left(f(y), \left\lfloor \frac{N_2(t,y)}{D_2(t,y)} \right\rfloor_t\right)}$$

avec

$$\begin{aligned}
N_2(t,y) &= \mathcal{L}_{x_1+x_2,f}\,(t-y)\\
&= t^{11} - 13\ y\ t^{10} + 75\ y^2\ t^9 - 255\ y^3\ t^8 + \left(570\ y^4 - 3\ a\right)\ t^7\\
&\quad + \left(-882\ y^5 + 27\ a\ y - 11\ b\right)\ t^6\\
&\quad + \left(966\ y^6 - 99\ a\ y^2 + 88\ b\ y\right)\ t^5\\
&\quad + \left(-750\ y^7 + 195\ a\ y^3 - 275\ b\ y^2\right)\ t^4\\
&\quad + \left(405\ y^8 - 225\ a\ y^4 + 440\ b\ y^3 - 4\ a^2\right)\ t^3\\
&\quad + \left(-145\ y^9 + 153\ a\ y^5 - 385\ b\ y^4 + 20\ a^2\ y + 4\ a\ b\right)\ t^2\\
&\quad + \left(31\ y^{10} - 57\ a\ y^6 + 176\ b\ y^5 - 28\ a^2\ y^2 - 16\ a\ b\ y - b^2\right)\ t\\
&\quad - 3\ y^{11} + 9\ a\ y^7 - 33\ b\ y^6 + 12\ a^2\ y^3 + 12\ a\ b\ y^2 + 3\ b^2\ y
\end{aligned}$$

et
$$\begin{aligned}
D_2(t,y) &= t^5 - 10\ y\ t^4 + 40\ y^2\ t^3 - 80\ y^3\ t^2 + \left(80\ y^4 + a\right)\ t\\
&\quad - 32\ y^5 - 2\ a\ y + b
\end{aligned}$$

$$\begin{aligned}
\left\lfloor \frac{N_2(t,y)}{D_2(t,y)} \right\rfloor_t &= t^6 - 3\ y\ t^5 + 5\ y^2\ t^4 - 5\ y^3\ t^3 - 4\ a\ t^2\\
&\quad + \left(-10\ y^5 - 8\ a\ y - 12\ b\right)\ t - 30\ y^6 - 30\ a\ y^2 - 29\ b\ y
\end{aligned}$$

$$\begin{aligned}
\mathrm{Rés}_y\left(f(y), \left\lfloor \frac{N_2(t,y)}{D_2(t,y)} \right\rfloor_t\right) &= t^{30} - 9\ a\ t^{26} + 33\ b\ t^{25} + 15\ a^2\ t^{22} - 210\ a\ b\ t^{21} + 360\ b^2\ t^{20}\\
&\quad + 45\ a^3\ t^{18} + 105\ a^2\ b\ t^{17} - 1335\ a\ b^2\ t^{16} + 1265\ b^3\ t^{15}\\
&\quad - 60\ a^4\ t^{14} + 780\ a^3\ b\ t^{13} - 615\ a^2\ b^2\ t^{12} - 1230\ a\ b^3\ t^{11}\\
&\quad + \left(-360\ b^4 - 144\ a^5\right)\ t^{10} + 240\ a^4\ b\ t^9 + 840\ a^3\ b^2\ t^8\\
&\quad + 720\ a^2\ b^3\ t^7 + \left(255\ a\ b^4 - 64\ a^6\right)\ t^6 + \left(33\ b^5 - 192\ a^5\ b\right)\ t^5\\
&\quad - 240\ a^4\ b^2\ t^4 - 160\ a^3\ b^3\ t^3 - 60\ a^2\ b^4\ t^2 - 12\ a\ b^5\ t - b^6\\
&= \left(t^{10} - 3\ a\ t^6 + 11\ b\ t^5 - 4\ a^2\ t^2 - 4\ a\ b\ t - b^2\right)^3
\end{aligned}$$

A.2 Calcul par les modules de Cauchy

Nous prenons :

$$f(x) = x^5 + a\ x + b \qquad \text{et} \qquad \Theta = x_3 + x_4 + x_5 \quad .$$

Avec les notations du théorème 4.4 (page 68) :

$H_3 = \mathfrak{S}_2 \times \mathfrak{S}_{\{3\}}$	$H_4 = \mathfrak{S}_2 \times \mathfrak{S}_{\{3,4\}}$	$H_5 = \mathfrak{S}_2 \times \mathfrak{S}_{\{3,4,5\}}$
$d_3 = 3$	$d_4 = 6$	$d_5 = 10$
$m_3 = 1$	$m_4 = 2$	$m_5 = 3$

L'invariant Θ étant d'arité 3 (voir paragraphe 4.5) nous avons :

$$\mathcal{R}_0 = \mathcal{R}_1 = \mathcal{R}_2 = \text{Récip}\,(t - \Theta) \quad .$$

Il n'est nécessaire de calculer que les trois dernières étapes des formules récursives en commençant par :

$$\begin{aligned}\mathcal{R}_2(t) &= \text{Récip}\,(t - \Theta) \\ &= 1 - (x_3 + x_4 + x_5)\ t \quad .\end{aligned}$$

Les trois derniers modules de Cauchy de f sont :

$$f_3 = x_3^3 + (x_4 + x_5)\ x_3^2 + \left(x_4^2 + x_5\ x_4 + x_5^2\right)\ x_3 + x_4^3 + x_5\ x_4^2 + x_5^2\ x_4 + x_5^3$$

$$f_4 = x_4^4 + x_5\ x_4^3 + x_5^2\ x_4^2 + x_5^3\ x_4 + x_5^4 + a$$

$$f_5 = x_5^5 + a\ x_5 + b \quad .$$

Nous calculons la première étape modulo $t^{d_3+1} = t^4$:

$$\mathcal{R}_2 = -\left(t \mod t^4\right)\ x_3 + \left(1 - (x_5 + x_4)\ t \mod t^4\right)$$

$$\begin{aligned}\text{Rés}_{x_3}\,(f_3, \mathcal{R}_2) &= 1 - (2\ x_5 + 2\ x_4)\ t + \left(2\ x_5^2 + 3\ x_4\ x_5 + 2\ x_4^2\right)\ t^2 \\ &\quad - \left(x_4\ x_5^2 + x_4^2\ x_5\right)\ t^3 \mod t^4\end{aligned}$$

$$\begin{aligned}\mathcal{R}_3 &= \sqrt[m_3]{\text{Rés}_{x_3}\,(f_3, \mathcal{R}_2) \bmod t^4} \\ &= 1 - (2\ x_5 + 2\ x_4)\ t + \left(2\ x_5^2 + 3\ x_4\ x_5 + 2\ x_4^2\right)\ t^2 - \left(x_4\ x_5^2 + x_4^2\ x_5\right)\ t^3 \quad .\end{aligned}$$

Nous calculons la seconde étape modulo $t^{d_4+1} = t^7$:

$$\begin{aligned}\mathcal{R}_3 &= \left(2\ t^2 - x_5\ t^3 \mod t^7\right)\ x_4^2 - \left(2\ t - 3\ x_5\ t^2 + x_5^2\ t^3 \mod t^7\right)\ x_4 \\ &\quad + \left(1 - 2\ x_5\ t + 2\ x_5^2\ t^2 \mod t^7\right)\end{aligned}$$

$$\begin{aligned}\text{Rés}_{x_4}\,(f_4, \mathcal{R}_3) &= 1 - 6\ x_5\ t + 19\ x_5^2\ t^2 - 40\ x_5^3\ t^3 + \left(55\ x_5^4 - 8\ a\right)\ t^4 \\ &\quad - \left(46\ x_5^5 - 32\ a\ x_5\right)\ t^5 + \left(11\ x_5^6 - 66\ a\ x_5^2\right)\ t^6 \mod t^7\end{aligned}$$

$$\begin{aligned}\mathcal{R}_4 &= \sqrt[m_4]{\text{Rés}_{x_4}\,(f_4, \mathcal{R}_3) \bmod t^7} \\ &= 1 - 3\ x_5\ t + 5\ x_5^2\ t^2 - 5\ x_5^3\ t^3 - 4\ a\ t^4 + \left(2\ x_5^5 + 4\ a\ x_5\right)\ t^5 \\ &\quad - \left(x_5^6 + a\ x_5^2\right)\ t^6\end{aligned}$$

alors que le calcul complet de $\text{Rés}_{x_4}\,(f_4, \mathcal{R}_3)$ aurait donné :

$$\begin{aligned}\text{Rés}_{x_4}\,(f_4, \mathcal{R}_3) = {} & 1 - 6\ x_5\ t + 19\ x_5^2\ t^2 - 40\ x_5^3\ t^3 + \left(55\ x_5^4 - 8\ a\right)\ t^4 \\ & - \left(46\ x_5^5 - 32\ a\ x_5\right)\ t^5 + \left(11\ x_5^6 - 66\ a\ x_5^2\right)\ t^6 \\ & + \left(26\ x_5^7 + 86\ a\ x_5^3\right)\ t^7 - \left(30\ x_5^8 + 50\ a\ x_5^4 - 16\ a^2\right)\ t^8 \\ & + \left(10\ x_5^9 - 6\ a\ x_5^5 - 32\ a^2\ x_5\right)\ t^9 + \left(4\ x_5^{10} + 24\ a\ x_5^6 + 24\ a^2\ x_5^2\right)\ t^{10} \\ & - \left(4\ x_5^{11} + 12\ a\ x_5^7 + 8\ a^2\ x_5^3\right)\ t^{11} + \left(x_5^{12} + 2\ a\ x_5^8 + a^2\ x_5^4\right)\ t^{12} \quad .\end{aligned}$$

Et la dernière étape modulo $t^{d_5+1} = t^{11}$:

$$\begin{aligned}\mathcal{R}_4 = {} & - \left(t^6 \mod t^{11}\right)\ x_5^6 + \left(2\ t^5 \mod t^{11}\right)\ x_5^5 - \left(5\ t^3 \mod t^{11}\right)\ x_5^3 \\ & + \left(5\ t^2 - a\ t^6 \mod t^{11}\right)\ x_5^2 - \left(3\ t - 4\ a\ t^5 \mod t^{11}\right)\ x_5 \\ & + \left(1 - 4\ a\ t^4 \mod t^{11}\right)\end{aligned}$$

$$\text{Rés}_{x_5}\,(f_5, \mathcal{R}_4) = 1 - 9\ a\ t^4 + 33\ b\ t^5 + 15\ a^2\ t^8 - 210\ a\ b\ t^9 + 360\ b^2\ t^{10} \mod t^{11}$$

$$\begin{aligned}\mathcal{R}_5 = {} & \sqrt[m_5]{\text{Rés}_{x_5}\,(f_5, \mathcal{R}_4) \bmod t^{11}} \\ = {} & 1 - 3\ a\ t^4 + 11\ b\ t^5 - 4\ a^2\ t^8 - 4\ a\ b\ t^9 - b^2\ t^{10}\end{aligned}$$

alors que le calcul complet de $\text{Rés}_{x_5}\,(f_5, \mathcal{R}_4)$ aurait donné :

$$\begin{aligned}\text{Rés}_{x_5}\,(f_5, \mathcal{R}_4) = {} & 1 - 9\ a\ t^4 + 33\ b\ t^5 + 15\ a^2\ t^8 - 210\ a\ b\ t^9 + 360\ b^2\ t^{10} + 45\ a^3\ t^{12} \\ & + 105\ a^2\ b\ t^{13} - 1335\ a\ b^2\ t^{14} + 1265\ b^3\ t^{15} - 60\ a^4\ t^{16} + 780\ a^3\ b\ t^{17} \\ & - 615\ a^2\ b^2\ t^{18} - 1230\ a\ b^3\ t^{19} - \left(360\ b^4 + 144\ a^5\right)\ t^{20} + 240\ a^4\ b\ t^{21} \\ & + 840\ a^3\ b^2\ t^{22} + 720\ a^2\ b^3\ t^{23} + \left(255\ a\ b^4 - 64\ a^6\right)\ t^{24} \\ & + \left(33\ b^5 - 192\ a^5\ b\right)\ t^{25} - 240\ a^4\ b^2\ t^{26} - 160\ a^3\ b^3\ t^{27} - 60\ a^2\ b^4\ t^{28} \\ & - 12\ a\ b^5\ t^{29} - b^6\ t^{30}\end{aligned}$$

Une transformation réciproque par rapport à la variable t achève le calcul :

$$\begin{aligned}\mathcal{L}_{x_3+x_4+x_5,f}(t) = {} & \text{Récip}\,(\mathcal{R}_5)\,(t) \\ = {} & t^{10} - 3\ a\ t^6 + 11\ b\ t^5 - 4\ a^2\ t^2 - 4\ a\ b\ t - b^2 \quad .\end{aligned}$$

Annexe B

Fonctionnement du factorisateur interactif

Cette annexe est consacrée à quelques exemples de factorisation de polynômes à coefficients entiers par notre factorisateur interactif. L'intérêt de la factorisation interactive sera évalué relativement au mode d'implantation provisoirement retenu : en `AXIOM` pour le prototypage de nos algorithmes.

B.1 Un exemple très particulier

Nous commençons les exemples de fonctionnement du factorisateur interactif par une factorisation complète. Le polynôme à factoriser a été construit *ad hoc* pour exploiter certaines des capacités du factorisateur et pour que l'exemple tienne en une page :

$$\begin{aligned} f(t) = {} & t^{20} + 4\,t^{17} - 4\,t^{16} + 6\,t^{14} - 12\,t^{13} + 6\,t^{12} + 4\,t^{11} \\ & - 12\,t^{10} + 12\,t^{9} - 3\,t^{8} - 4\,t^{7} + 6\,t^{6} - 4\,t^{5} + t^{4} - 1 \quad . \end{aligned}$$

Ce polynôme est décomposable :

$$f(t) = (g \circ h)(t)$$

avec

$$\begin{aligned} g(t) &= t^4 - 1 \\ &= (t-1)(t+1)(t^2+1) \end{aligned}$$

et

$$h(t) = t^5 + t^2 - t \quad .$$

Le polynôme f se factorise :

$$\begin{aligned} f(t) = {} & \left(t^3 - t + 1\right)\,\left(t^2 + 1\right)\,(t+1)\,(t-1)\,\left(t^3 + t + 1\right) \\ & \times\ \left(t^{10} + 2\,t^7 - 2\,t^6 + t^4 - 2\,t^3 + t^2 + 1\right) \quad . \end{aligned}$$

Dans le dessin suivant, les flèches pleines indiquent qu'un polynôme « est le facteur de » celui dont est issue la flèche. Les flèches en pointillés signifient que le polynôme sur lequel elles pointent est le composé fonctionnel du polynôme dont elles sont issues avec le polynôme $h(t) = t^5 + t^2 - t$.

Le factorisateur interactif obtient les facteurs de f dans l'ordre où sont numérotées les flèches (en 237 appels à la fonction `factor!`) :

1. le factorisateur découvre que le polynôme f est fonctionnellement décomposable (en 6 appels) et entreprend la factorisation du facteur à gauche $g(t) = t^4 - 1$ de la décomposition fonctionnelle ;

2. le factorisateur trouve une factorisation non triviale de g (en 50 appels) ;

3. cette factorisation est immédiatement traduite en une factorisation non triviale de f, deux facteurs :

$$f_1(t) = t^{10} + 2\ t^7 - 2\ t^6 + t^4 - 2\ t^3 + t^2 - 1$$

et $$f_2(t) = t^{10} + 2\ t^7 - 2\ t^6 + t^4 - 2\ t^3 + t^2 + 1 \quad ;$$

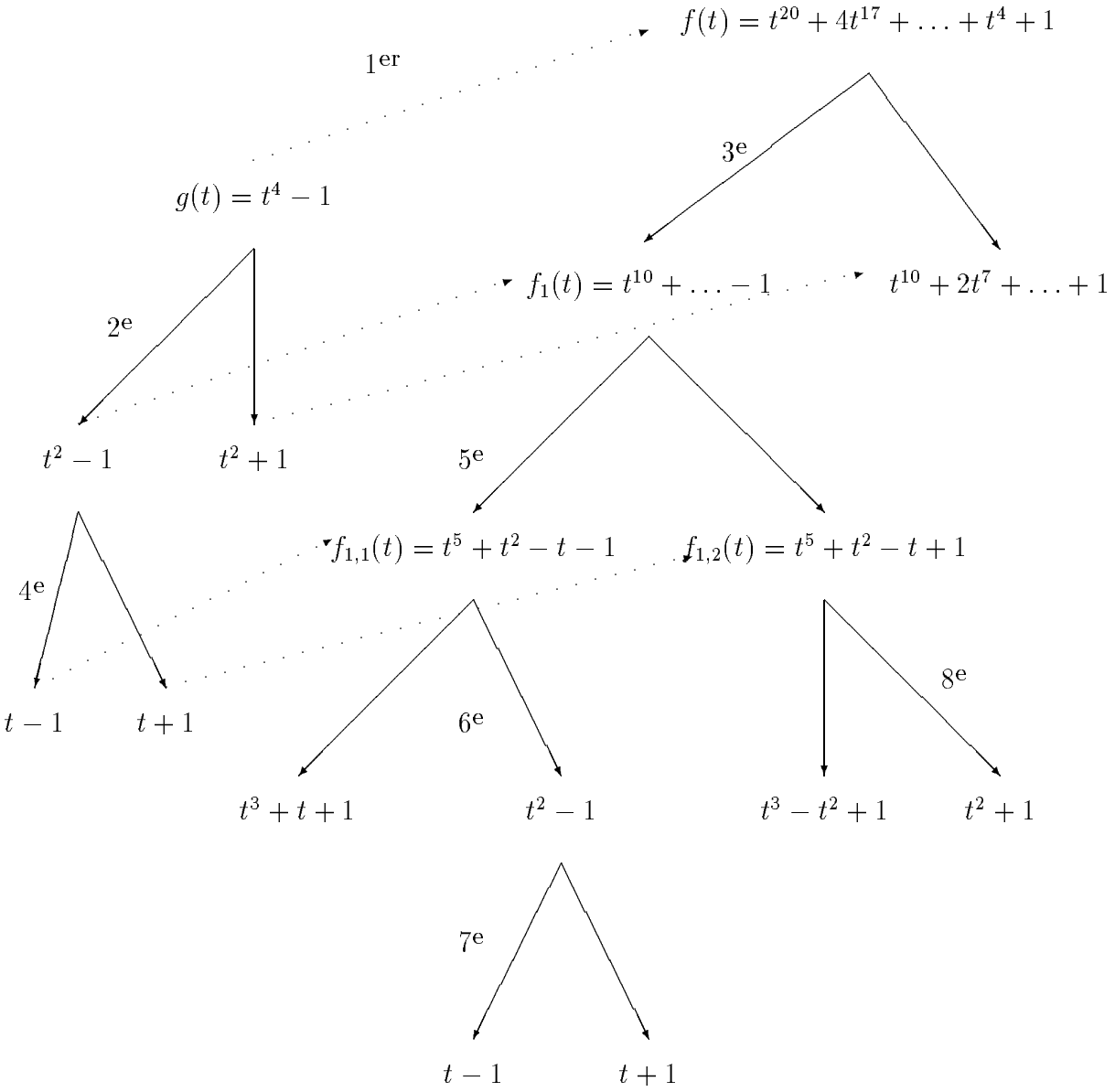
4. la factorisation de g n'étant pas achevée le factorisateur interactif la poursuit jusqu'à trouver deux facteurs irréductibles de degré 1 (en 45 appels) ;

5. cette factorisation est immédiatement traduite en une factorisation non triviale de f_1 ($f_{1,1}(t) = t^5 + t^2 - t - 1$ et $f_{1,2}(t) = t^5 + t^2 - t + 1$ de degré 5) ;

6. la factorisation de g (ainsi que les factorisations non triviales qui en découlent) est achevée mais les facteurs $f_{1,1}$, $f_{1,2}$ ou f_2 ne sont peut-être pas irréductibles, leur factorisation est donc entreprise jusqu'à trouver une factorisation non triviale (irréductibilité de f_2 : 4 appels ; factorisation non triviale de $f_{1,1}$: 59 appels) ;

7. qui est elle-même achevée à cette étape (en 5 appels) ;

8. seule la factorisation de $f_{1,2}$ restait à effectuer ; une fois faite, tous les facteurs irréductibles de f sont connus (en 68 appels).



Cet exemple illustre la façon dont le factorisateur interactif poursuit « à la demande » la factorisation du facteur gauche de la décomposition polynomiale de f. La détermination de facteurs de g est immédiatement traduite en facteurs de f qui ne sont pas nécessairement irréductibles.

Lorsque la factorisation de f_1 est poursuivie, le factorisateur utilise la décomposition polynomiale de f_1 et son facteur gauche. Dans le cas d'une factorisation partielle, si f_1 ne correspond pas à certains critères recherchés comme les degrés possibles de ses facteurs, les degrés de ses facteurs modulo certains premiers ou son irréductibilité, sa factorisation peut être abandonnée au profit d'autres facteurs à factoriser, par exemple f_2 dont l'irréductibilité est facilement prouvée par un critère (par le critère de Brillhart, voir le théorème 5.9, page 83).

Le factorisateur d'`AXIOM`, comme d'autres implantations de factorisateurs dans des logiciels de calcul formel ou dans des bibliothèques `C`, n'utilise pas la décomposition polynomiale fonctionnelle à l'exception du cas particulier, le plus fréquemment rencontré où

$h(t)$ est de la forme t^k. Nous avons ajouté au début du processus de factorisation la recherche d'une décomposition polynomiale au prix d'un surcoût de calcul. Ce surcoût de calcul lorsqu'aucune factorisation n'est trouvée est marginal, sans être négligeable, devant le coût total de la factorisation mais lorsque qu'une décomposition est trouvée sa factorisation fait avancer à pas de géants dans la factorisation du polynôme f. Le cas le pire est lorsque le facteur à gauche de la décomposition polynomiale est irréductible et que cette irréductibilité n'est prouvée qu'à l'issue d'une factorisation coûteuse de ce facteur gauche. Pour éviter des calculs dont l'inutilité peut être connue d'avance, la recherche d'une décomposition fonctionnelle polynomiale ainsi que son utilisation peuvent être débrayées par les paramètres `tryFunctionalDecomposition?` et `useFunctionalDecomposition?` (voir le paragraphe 7.3.6, page 142).

La factorisation sans facteur carré est traitée d'une manière analogue à la décomposition polynomiale : la factorisation n'est entreprise que sur les parties sans facteur carré après qu'elles ont été déterminées et seulement sur certains d'entre eux, qui répondent à nos critères, en cas de factorisation partielle.

B.2 Autres exemples

Nous traitons dans ce paragraphe des exemples qui correspondent effectivement à des résolvantes. Le but recherché en théorie de Galois effective n'est pas la factorisation complète du polynôme mais la détermination de la partition de son degré en degrés de facteurs irréductibles parmi une liste de partitions possibles.

L'information partielle dont nous disposons sur chacune des factorisations correspond à une liste de partitions de leurs degrés extraites de matrices de partition ou de toute autre source d'information. Nous comparons sur chaque exemple le comportement de notre factorisateur pour les trois stratégies suivantes :

1. la factorisation complète sans utiliser l'information (comptée en nombres d'appels à la fonction `factor!`) ;

2. la factorisation partielle avec information : nous donnons le nombre d'appels nécessaires pour déterminer la bonne partition du degré parmi la liste des partitions ;

3. la factorisation partielle avec information et stratégie : dans ce cas nous précisons au factorisateur quels sont les facteurs dont les degrés seront discriminants pour qu'il ne cherche que ceux-là en priorité.

En raison du caractère probabiliste de la factorisation modulaire (ainsi que des faiblesses d'`AXIOM`, voir paragraphe B.3) nous ne considérons dans ce paragraphe que l'étape de remontée à la Hensel et de combinaison des facteurs modulaires pour trouver des vrais facteurs. Nous ne comptons que les appels de `factor!` y ayant trait. En effet, les algorithmes que nous avons implantés l'ont été de façon à tirer parti de toute information trouvée dès qu'elle apparaît (voir, par exemple, la détection de « probables » vrais facteurs, au paragraphe 6.2.3, ou la suppression de la liste des degrés possibles pour les facteurs les degrés qui ne peuvent plus être engendrés par combinaisons de facteurs déjà testées, au paragraphe 5.5.3). Ils sont

de ce fait très dépendants de l'ordre (aléatoire) dans lequel ont été obtenus les facteurs modulaires. Deux factorisations successives peuvent donc conduire à des exécutions très différentes. Une très rapide si des vrais facteurs sont détectés précocement. C'est le cas si nous avons été « chanceux » dans l'ordre des facteurs modulaires issus de l'étape probabiliste de factorisation modulaire. Une plus longue si les vrais facteurs ne sont détectés que vers la fin de l'énumération des combinaisons.

Pour comparer plusieurs factorisations à factorisation modulaire identique nous « chargeons » le factorisateur interactif avec une factorisation modulaire fixée pour chaque exemple dont nous donnons la liste des degrés.

Un polynôme de degré 276

Ce polynôme vient de [112]. Il s'agit de la résolvante $\mathcal{L}_{x_1x_2,f}$ du polynôme

$$\begin{aligned} f(x) = {} & x^{24} - 18x^{23} + 141x^{22} - 630x^{21} + 1774x^{20} - 3318x^{19} + 4373x^{18} - 4632x^{17} \\ & + 4817x^{16} - 5082x^{15} + 4698x^{14} - 3298x^{13} + 1946x^{12} - 528x^{11} - 330x^{10} \\ & + 492x^9 + 648x^7 - 606x^6 + 234x^5 + 115x^4 - 132x^3 + 39x^2 - 6x + 1 \quad . \end{aligned}$$

Elle se factorise en deux facteurs de degrés respectifs 12 et 264.

Les degrés des facteurs modulaires, modulo 31, sont :

$$[36, 36, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 9, 4, 4, 4, 2, 1] \quad .$$

Dans ce cas, la factorisation complète par notre factorisateur demande 33822 appels pour les remontées et la factorisation (576 secondes).

La détermination de la partition de la résolvante entre

$$[12, 264] \qquad \text{et} \qquad [276]$$

ne se fait plus qu'en 1160 appels (159 secondes) nécessaires pour trouver le facteur de degré 12.

Lorsque le factorisateur, en factorisation partielle, ne recherche que le facteur de degré 12, il détermine la partition $[12, 264]$ en seulement 126 appels (18 secondes).

Un polynôme de degré 126

Ce polynôme vient de [41, exemple 4, page 44]. Il s'agit de la résolvante

$$\mathcal{L}_{x_1x_2x_3x_4x_5+x_6x_7x_8x_9x_{10},f}$$

du polynôme

$$f(x) = x^{10} - 9x^8 + 27x^6 - 2x^5 - 27x^4 + 9x^3 - 8x + 1 \quad .$$

Elle se factorise en deux facteurs de degrés respectifs 36 et 90.

Les degrés des facteurs modulaires, modulo 17, sont :

$$[6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 3, 3, 3, 3] \quad .$$

Dans ce cas, la factorisation complète par notre factorisateur demande 264059 appels pour les remontées et la factorisation (764 secondes).

La détermination de la partition de la résolvante entre

$$[36, 90] \qquad \text{et} \qquad [126]$$

ne se fait plus qu'en 142375 appels (389 secondes) nécessaires pour trouver le facteur de degré 36.

La factorisation partielle en ne recherchant que le facteur de degré 36 conduit au même résultat. C'est logique car les degrés des facteurs modulaires, tous diviseurs de 36, ne conduisent pas à rejeter précocement des combinaisons possibles.

B.3 Problèmes posés par `AXIOM`

Le lecteur pourrait s'étonner de l'absence de comparaison avec les logiciels existants, ne serait-ce que par rapport aux fonctions déjà implantées en `AXIOM`.

Il ne s'agit pas d'une omission involontaire. La cause en est des spécificités inattendues du logiciel `AXIOM` que nous n'avons découvertes que récemment à notre corps défendant.

Leur conséquence est que la comparaison avec les fonctions `AXIOM` préexistantes d'un logiciel compilé par un programmeur en `AXIOM` utilisant des opérations sur les listes, les polynômes et les entiers est particulièrement délicate sinon impossible. C'est le cas de notre factorisateur.

C'est à l'occasion de la mise au point des exemples de cette thèse que nous avons découvert (et signalé à NAG qui les a reconnues) certaines spécifications **non documentées** du logiciel `AXIOM` (version 2.1).

La première concerne les types d'entiers en `AXIOM`.

`AXIOM` propose trois types d'entiers dans la catégorie `IntegerNumberSystem` :

- `Integer` qui implante les « vrais » entiers, c'est à dire des entiers de taille arbitraire qui n'est limitée que par la mémoire disponible ;
- `SingleInteger` qui est censé implanter les entiers aisément manipulables par les processeurs, à savoir, typiquement, ceux compris entre -2^{31} et $2^{31}-1$, car ils tiennent sur un seul « mot machine » ;
- `MachineInteger` dont l'utilisation n'est pas recommandée par NAG et qui ne semblent être utilisés que pour l'interfaçage avec les bibliothèques numériques de NAG.

Avant d'aller plus loin, nous motivons l'intérêt d'utiliser des mots machine.

Dans le processus de factorisation de polynômes à coefficients entiers plusieurs étapes (la factorisation modulaire, la remontée à la Hensel) font appel à des polynômes dont les coefficients sont plus petits qu'un certain modulo. Traditionnellement, ce modulo est choisi de façon à ce que tous les produits d'entiers, ou les inverses par rapport au modulo, intervenant dans les calculs sur les polynômes tiennent dans un seul mot machine. Il suffit pour cela de choisir, quand c'est possible, et cela l'est toujours en général, un nombre dont la taille est un peu plus petite que la moitié d'un mot machine. Un gain important est attendu

de cette stratégie car l'utilisation d'opérations sur les mots machine permet de s'affranchir totalement du mécanisme (d'origine Lisp) de manipulation d'entiers de taille arbitraire. Ce mécanisme est plus ou moins lourd suivant l'implantation utilisée (par exemple `LIP`, `PARI` ou `GMP`) mais seule la garantie de coefficients théoriquement bornés grâce au modulo permet d'utiliser pleinement la puissance du processeur et de ses registres.

Nous nous sommes particulièrement intéressé à ce problème d'utilisation des mots machine en `AXIOM` car ce procédé en était absent. En effet, les domaines `AXIOM` ne peuvent être compilés lorsqu'ils dépendent d'un paramètre, un élément d'un domaine, qui ne sera connu qu'à l'exécution. En clair, les domaines `PrimeField(p)` qui implantent les corps finis peuvent être utilisés pour des valeurs fixées à l'avance lors de la compilation mais pas pour une variable `p` dont la valeur est déterminée à l'exécution.

La factorisation modulaire nécessaire à la factorisation de polynômes à coefficients entiers implantée dans le logiciel `AXIOM` ne fait donc pas appel aux paquetages de factorisation sur les corps finis qui existent déjà mais y substitue des fonctions factorisant modulairement des polynômes à coefficients **entiers** (type `Integer`) modulo un entier transmis en paramètre. Les entiers machine ne sont donc pas utilisés. Qui plus est l'implantation de ces paquetages (en `AXIOM` 2.1) n'est pas particulièrement optimisée. Par exemple, la division d'un polynôme par un polynôme modulo p est d'abord effectuée dans les entiers, puis `AXIOM` calcule les restes des coefficients du résultat modulo p. Cette stratégie est particulièrement inefficace, lorsque p ou les degrés augmentent, en raison d'une inutile croissance des coefficients dans les calculs intermédiaires. Le caractère économique du modulo n'est donc pas pleinement utilisé.

Dans notre implantation d'un factorisateur nous avons utilisé des entiers machine (type `SingleInteger`) pour la représentation des coefficients des polynômes modulaires et privilégié des algorithmes comme celui de Collins et Encarnación (voir le paragraphe 6.2, page 117), qui déplacent un maximum de calculs sur les entiers vers les (rapides) entiers machine.

Or, lors des premiers tests en degrés élevés, c'est à dire supérieurs à 100, de notre factorisateur la lenteur suspecte de la partie modulaire de la factorisation (plus de 100 fois plus lente que `Maple` ou que du `C++`) nous a amenés à des investigations supplémentaires. Le résultat de cet examen est que les entiers machine **n'existent pas** en `AXIOM` 2.1. Le type `SingleInteger` (tout comme `MachineInteger`) est en fait **simulé** au niveau du Lisp sous-jacent en `AXIOM` par des entiers de taille arbitraire. La coercition (fonction `coerce`) d'un entier trop grand vers un entier machine n'est impossible qu'à cause de la présence de tests dans la fonction de coercition qui interdisent à l'entier de dépasser 2^{31}. Mais physiquement, il n'en est rien.

Au-delà du fait que les temps de calculs en `SingleInteger` et `Integer` sont identiques il est facile de le prouver : il est possible à partir de n'importe quel petit entier d'engendrer des « entiers machine » de taille quelconque (bien au-delà de 2^{32}). Voilà certainement la raison pour laquelle les concepteurs d'`AXIOM` n'avaient pas optimisé l'implantation des calculs modulaires : il n'y avait rien à y gagner.

NAG interrogé sur ce point a confirmé cette étrange particularité (non documentée) mais n'a pu me fournir aucune solution pour accéder au type entier machine à partir d'`AXIOM` (sauf à utiliser leur nouveau compilateur `Aldor`, encore en développement, qui, lui, fournit cet accès aux entiers machine).

En conclusion de ce premier point, il n'y a donc pas de différence d'implantation (ni

d'efficacité) entre les types `Integer` et `SingleInteger` et pas d'autre possibilité d'accès aux entiers machine en `AXIOM` 2.1.

La deuxième spécification non documentée d'`AXIOM` que nous avons découverte concerne la compilation.

La division de polynômes modulaires est utilisée de façon intensive aussi bien lors de la factorisation modulaire, lors des remontées à la Hensel que lors des combinaisons des facteurs modulaires. Dans le but d'avoir cette division de polynômes modulaires plus efficace que celle déjà présente en `AXIOM`, évoquée ci-dessus, nous en avons programmé une version qui maintient les coefficients dans la taille d'un mot machine (sans répéter des modulos à chaque calcul). Or celle-ci n'a commencé à dépasser en efficacité la fonction d'`AXIOM` que pour des modulos déjà assez conséquents. Elle était par contre plus lente pour les petits modulos. À nouveau nous avons recherché la cause de cette inefficacité mystérieuse. Nous avons par exemple recopié ligne à ligne la fonction d'`AXIOM` en y ajoutant seulement les modulos bien placés : la fonction nouvellement compilée est plus de deux fois plus lente. En fait sans ajouter aucun modulo et en la recopiant par un copier-coller à partir du code source disponible, la fonction `AXIOM` recompilée par l'utilisateur est deux fois plus lente que son original.

Ce phénomène, qui ne se produit que pour pour certaines fonctions vient à nouveau de spécifications non documentées d'`AXIOM`. Ce comportement est apparu avec la version 2.1 d'`AXIOM` pour laquelle le Lisp sous-jacent a été changé. Alors que dans les versions antérieures le code d'`AXIOM` était compilé en code natif de la machine, donc dépendant de l'architecture utilisée, depuis la version 2.1, le code est désormais compilé en « byte code », code mi-compilé mi-exécuté qui a l'avantage d'être indépendant de l'architecture sur laquelle `AXIOM` est utilisé. Ce gain en portabilité d'`AXIOM` s'est fait au prix d'une légère baisse d'efficacité (lorsqu'`AXIOM` 2.1 a été distribué, tous les programmeurs en `AXIOM` ont remarqué que le nouveau code compilé était environ deux fois plus lent). Cependant, NAG a sélectionné (sélection non rendue publique dans `AXIOM` 2.1) certaines opérations critiques comme le pgcd, certaines opérations sur les listes ou l'accès aux coefficients ainsi que... la division des polynômes. Ces opérations spécifiquement choisies sont en fait, elles, toujours compilées en code natif et incorporées au « noyau » d'`AXIOM` spécifique à chaque architecture. La différence d'efficacité entre la fonction ainsi compilée et sa copie recompilée (pour peu que son nom ait été modifié pour désactiver le phénomène) vient donc de ce que l'une l'a été en code natif directement exécutable et l'autre en « byte code » dont l'exécution est environ deux fois plus lente. NAG contacté a confirmé toutes ces affirmations et a apporté les précisions suivantes : il n'est pas possible à l'utilisateur d'ajouter des fonctions à la liste des fonctions spécifiquement sélectionnées pour être compilées en code natif au lieu de byte code, seul NAG a le pouvoir de les choisir, et il n'est pas possible de débrayer le mécanisme sauf à changer de nom toutes les fonctions concernées. Cette dernière solution équivaudrait à recompiler quasiment tout `AXIOM` en « byte code ». Il y a 233 fonctions concernées dans la liste rendue publique, à notre demande, par NAG dans la version 2.2 d'`AXIOM` (mars 1999). Nous n'avons pas même cherché à entreprendre cette tâche de recompilation, car si c'est pour obtenir un clone d'`AXIOM` plus que deux fois plus lent alors que nos tests poussent déjà `AXIOM` et notre patience à leurs limites...

En conclusion de ce deuxième point, il n'est quasiment pas possible de comparer en

efficacité un algorithme AXIOM avec sa version modifiée par l'utilisateur si la fonction qui l'implante appartient à la liste des fonctions clefs car dans ce cas ils ne seront pas compilés par le compilateur AXIOM de la même façon.

Le troisième point m'a été révélé par R. Rioboo (LIP6) et est mineur par rapport aux précédents mais il n'aide pas le programmeur AXIOM à se faire une bonne idée du fonctionnement de ses programmes : le temps affiché par AXIOM 2.1 est tout simplement faux. Précisément, il se compose en quatre types de temps qui sont sommés pour donner le temps total d'exécution d'une fonction :

- IN : INput time, temps d'initialisation de la fonction ;
- OT : OuTput time, temps de l'affichage du résultat ;
- EV : EValuation time, temps d'évaluation de la fonction ;
- GC : Garbage Collection, temps passé dans le ramasse-miettes.

Les temps IN et OT sont en général négligeables devant EV et GC. Le problème vient de ce que le temps GC est **déjà** compté dans le temps EV. Le calcul IN + OT + EV + GC effectué par AXIOM est faux. Le temps total correspond simplement à IN + OT + EV alors que le temps passé à l'évaluation hors GC est égal à EV − GC.

B.4 Conclusions provisoires et perspectives

Les leçons que nous tirons de notre expérience en AXIOM sont les suivantes. Le système de typage d'AXIOM a certainement largement contribué à la faisabilité du projet et AXIOM avait été choisi dans un but de prototypage des algorithmes pouvant évoluer vers des exécutables rapides grâce au nouveau compilateur Aldor. Cependant, les spécificités non-documentées d'AXIOM nuisent à l'exploitation des résultats obtenus ainsi qu'aux possibilités de comparaison avec les autres logiciels existants. En fait AXIOM n'avait lui même été retenu que par défaut devant l'indisponibilité d'Aldor qui devait, lui, jouir de toutes les propriétés adéquates (accès aux entiers machine, compilations en C) à l'exception, rédhibitoire, d'une bibliothèque mathématique suffisamment développée à l'époque où nous avons produit du code (en 1997-1998). Actuellement (1999), Aldor commence seulement à fournir une bibliothèque de programme plus complète, BasicMath, dont sont, pour le moment, absentes les fonctions de factorisation.

Finalement, notre implantation en AXIOM, bien que montrant l'intérêt de nos méthodes dans certains cas, ne nous permet pas d'évaluer pleinement leur apport pratique. Nous avons notamment, dans cette annexe, fait l'impasse sur la partie modulaire des calculs. Pour exploiter de manière vraiment efficace les résultats de notre travail de prototypage sur les algorithmes de factorisation, une autre implantation, dans un autre langage qu'AXIOM, plus près de l'architecture des machines sera indispensable dans le futur.

Bibliographie

[1] Abbott (John). – Univariate factorization over the integers. – mars 1998. http://lancelot.dima.unige.it/research/publications.html.

[2] Abdeljaouad (Ines). – *Calcul d'invariants primitifs de groupes finis.* – Rapport technique, LIP 6, 1997.

[3] Abdeljaoued (Jounaïdi). – Algorithmes rapides pour le calcul du polynôme caractéristique, mars 1997. Thèse de Doctorat, U.F.R. des sciences et techniques de l'Université de Franche-Comté, Besançon, France.

[4] Adleman (L. M.) et Odlyzko (A. M.). – Irreducibility testing and factorization of polynomials. *Mathematics of Computation*, vol. 41, 1983, pp. 699–709.

[5] Arnaudiès (Jean-Marie) et Valibouze (Annick). – *Calculs de Résolvantes.* – Rapport interne n° 94/46, Laboratoire Informatique Théorique et Programmation, juillet 1994.

[6] Arnaudiès (Jean-Marie) et Valibouze (Annick). – Lagrange resolvents. *Journal of Pure and Applied Algebra*, vol. 117 & 118, 1997, pp. 23–40.

[7] Aubry (Philippe) et Valibouze (Annick). – *Computing characteristic polynomials associated to some quotient rings.* – Rapport de Recherche n° 1997/020, LIP6- UPMC-4, Place Jussieu F-75252 Paris Cedex 05, LIP 6, février 1998.

[8] Beauzamy (Bernard). – Products of polynomials and a priori estimates for coefficients in polynomial decompositions: a sharp result. *Journal of Symbolic Computation*, vol. 13, n° 5, 1992, pp. 463–472.

[9] Beauzamy (Bernard), Trevisan (Vilmar) et Wang (Paul S.). – Polynomial factorization: sharp bounds, efficient algorithms. *Journal of Symbolic Computation*, vol. 15, n° 4, 1993, pp. 393–413.

[10] Ben-Or (M.). – Probabilistic algorithms in finite fields. *Proceedings 22nd IEEE Symposium on Foundations of Computer Science*, 1981, pp. 394–398.

[11] Berkowitz (Stuart J.). – On computing the determinant in small parallel time using a small number of processors. *Information Processing Letter*, vol. 18, 1984, pp. 147–150.

[12] Berlekamp (E. R.). – Factoring polynomials over finite fields. *Bell System Tech. J.*, vol. 46, 1967, pp. 1853–1859.

[13] Berlekamp (E. R.). – Factoring polynomials over large finite fields. *Math. Comp.*, vol. 24, 1970, pp. 713–735.

[14] Berwick (W. E. H.). – On soluble sextic equations. *Proceedings of the London Mathematical Society*, vol. 29, n° 1674, décembre 1929, pp. 1–28.

[15] Boyd (David W.). – Bounds for the height of a factor of a polynomial in terms of Bombieri's norms. I. The largest factor. *Journal of Symbolic Computation*, vol. 16, n° 2, 1993, pp. 115–130.

[16] Boyd (David W.). – Bounds for the height of a factor of a polynomial in terms of Bombieri's norms. II. The smallest factor. *Journal of Symbolic Computation*, vol. 16, n° 2, 1993, pp. 131–145.

[17] Brent (R. P.) et Kung (H. T.). – Fast algorithms for manipulating formal power series. *Journal of the Association for Computing Machinery*, vol. 25, n° 4, octobre 1978, pp. 581–595.

[18] Brent (Richard P.). – Multiple-precision zero-finding methods and the complexity of elementary function evaluation. *In : Analytic Computational Complexity*, éd. par Traub (J. F.), pp. 151–176. – New York, Academic Press, 1975.

[19] Brillhart (John). – On the Euler and Bernoulli polynomials. *J. Reine Angew. Math.*, vol. 234, 1969, pp. 45–64. – MR39:4117.

[20] Brillhart (John). – Note on irreducibility testing. *Mathematics of Computation*, vol. 35, n° 152, 1980, pp. 1379–1381. – MR83g:12002.

[21] Brown (W. S.) et Graham (R. L.). – An irreducibility criterion for polynomials over the integers. *American Mathematical Monthly*, vol. 76, 1969, pp. 795–797. – MR40:2652.

[22] Butler (Gregory). – *Fundamental algorithms for permutation groups.* – Springer-Verlag, 1991, *Lecture Notes in Computer Science*, volume 559. ISBN 3-540-54955-2, Zbl785.20001.

[23] Butler (Gregory). – The transitive groups of degree fourteen and fifteen. *Journal of Symbolic Computation*, vol. 16, 1993, pp. 413–422.

[24] Butler (Gregory) et McKay (John). – The transitive groups of degree up to eleven. *Communincations in Algebra*, vol. 11, 1983, pp. 863–911.

[25] Cantor (David G.) et Kaltofen (Erich). – On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, vol. 28, n° 7, 1991, pp. 693–701.

[26] Cantor (David G.) et Zassenhaus (Hans). – A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, vol. 36, n° 154, 1981, pp. 587–592.

[27] Cockayne (E. J.). – Computation of Galois group elements of a polynomial equation. *Math. Comp.*, vol. 23, n° 106, 1969, pp. 425–428. – MR39:3733.

[28] Cohen (Henri). – *A Course in Computational Algebraic Number Theory.* – Springer Verlag, 1993, *Graduate Texts in Mathematics*, volume 138.

[29] Colin (Antoine). – Formal computation of Galois groups with relative resolvents. *In : proc. AAECC'95, Lecture Notes in Computer Science*, éd. par Cohen (G.), Giusti (M.) et Mora (T.), pp. 169–182. – Springer Verlag, 1995.

[30] Colin (Antoine). – Théorie des invariants effective. Application à la théorie de Galois et à la résolution de systèmes algébriques. Implantation en AXIOM, juin 1997. Thèse de Doctorat, École Polytechnique.

[31] Collins (George E.). – Factoring univariate integral polynomials in polynomial average time. *In : Symbolic and algebraic computation, EUROSAM '79, conférence internationale*, pp. 317–329. – Marseille, 1979. Zbl409.68021.

[32] Collins (George E.) et Encarnación (Mark J.). – Efficient rational number reconstruction. *Journal of Symbolic Computation*, vol. 20, n° 3, 1995, pp. 287–297.

[33] Collins (George E.) et Encarnación (Mark J.). – Improved techniques for factoring univariate polynomials. *Journal of Symbolic Computation*, vol. 21, n° 3, 1996, pp. 313–327.

[34] Conway (John H.), Hulpke (Alexander) et McKay (John). – On transitive permutation groups. *LMS Journal of Computation and Mathematics*, vol. 1, 1998, pp. 1–8 (electronic). – MR99g:20011.

[35] Davenport (J.), Siret (Y.) et Tournier (E.). – *Calcul formel, systèmes et algorithmes de manipulations algébriques.* – Masson, 1987, *Études et recherches en informatique.*

[36] Davenport (J. H.). – Factorisation of sparse polynomials. *In : Computer algebra (London, 1983)*, pp. 214–224. – Berlin, Springer, 1983. MR86i:11073.

[37] Davenport (J. H.) et Smith (G. C.). – Fast recognition of alternating and symmetric Galois groups. – Prépublication.

[38] Davenport (James H.). – On Brillhart irreducibility. – 1995. Communication privée.

[39] Dèvenport (Dzh.). – Galois groups and the factorization of polynomials. *Rossiĭskaya Akademiya Nauk. Programmirovanie*, vol. 1, 1997, pp. 43–58. – MR98m:12002.

[40] Ducos (Lionel). – Effectivité en théorie de Galois. Sous-résultants. – Université de Poitiers, novembre 1997. Thèse de Doctorat.

[41] Eichenlaub (Yves). – Problèmes effectifs de théorie de Galois en degrés 8 à 11. – Université de Bordeaux 1, 1996. Thèse de Doctorat.

[42] Eichenlaub (Yves) et Olivier (Michel). – Computation of Galois groups for polynomials with degree up to eleven. – Prépublication.

[43] Encarnación (Mark J.). – Factoring polynomials over algebraic number fields via norms. *In : Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, éd. par Küchlin (Wolfgang). The Association for Computing Machinery, pp. 265–270. – ACM.

[44] Girstmair (K.). – On invariant polynomials and their application in field theory. *Mathematics of Computation*, vol. 48, n° 178, 1987, pp. 781–797.

[45] Giusti (Marc), Lazard (Daniel) et Valibouze (Annick). – Algebraic transformations of polynomial equations, symmetric polynomials and elimination. *In : Proceedings ISSAC'88*, éd. par Gianni (P.). International Symposium on Symbolic and Algebraic Computation, pp. 309–314. – Springer Verlag.

[46] Henrici (Peter). – Automatic computations with power series. *Journal of the Association for Computing Machinery*, vol. 3, n° 1, January 1956, pp. 10–15.

[47] Henrici (Peter). – *Discrete Fourier Analysis – Cauchy Integrals – Construction of Conformal Maps – Univalent Functions.* – New York, John Wiley and Sons, 1986, *Applied and Computational Complex Analysis*, volume 3.

[48] Jenks (Richard D.) et Sutor (Robert S.). – *Axiom : The scientific computation system.* – NAG, New-York, 1992.

[49] Kaltofen (Erich). – Factorization of polynomials. *In : Computer algebra, symbolic and algebraic computation, Comput. Suppl. 4*, pp. 95–113. – Springer, 1982.

[50] Kaltofen (Erich). – Polynomial factorization 1982–1986. *In : Computers in mathematics (Stanford, CA, 1986)*, pp. 285–309. – New York, Dekker, 1990. MR92f:12001.

[51] Kaltofen (Erich). – Polynomial factorization 1987–1991. *In : LATIN '92 (São Paulo, 1992)*, pp. 294–313. – Berlin, Springer, 1992.

[52] Kaltofen (Erich), Musser (David R.) et Saunders (B. David). – A generalized class of polynomials that are hard to factor. *SIAM Journal on Computing*, vol. 12, n° 3, 1983, pp. 473–483.

[53] Kaltofen (Erich) et Shoup (Victor). – Subquadratic-time factoring of polynomials over finite fields. *Mathematics of Computation*, vol. 67, n° 223, 1998, pp. 1179–1197.

[54] Knuth (Donald E.). – *The art of computer programming. Vol. 2.* – Addison-Wesley Publishing Co., Reading, Mass., 1981, seconde édition, xiii+688p. Seminumerical algorithms, Addison-Wesley Series in Computer Science and Information Processing.

[55] Kolesova (G.) et McKay (J.). – Practical strategies for computing Galois groups. *In : Computational group theory, Proc. Symp.*, pp. 297–299. – Durham/Engl., 1982. Zbl538.12005.

[56] Kozen (Dexter) et Landau (Susan). – Polynomial decomposition algorithms. *Journal of Symbolic Computation*, vol. 7, 1989, pp. 445–456.

[57] Lagarias (J.C.), Montgomery (H.L.) et Odlyzko (A.M.). – A bound for the least prime ideal in the Chebotarev density theorem. *Invent. Math.*, vol. 54, 1979, pp. 271–296. – Zbl413.12011.

[58] Lagarias (J.C.) et Odlyzko (A.M.). – Effective versions of the Chebotarev density theorem. *In : Algebr. Number Fields, Proc. Symp. London math. Soc. 1975*, pp. 409–464. – Univ. Durham, 1977. Zbl362.12011.

[59] Lagrange (Joseph-Louis). – *Réflexions sur la résolution algébrique des équations*, pp. 205–421. – Paris, Gauthier-Villars, 1869, *Œuvres de Lagrange*, volume 3. Nouveaux Mémoires de l'Académie royale des Sciences et Belles-Lettres de Berlin, années 1770 et 1771.

[60] Lefton (Phyllis). – Galois resolvents of permutation groups. *Am. Math. Mon.*, vol. 84, 1977, pp. 642–644. – Zbl374.12013.

[61] Lehobey (Frédéric). – *Algorithmic Methods and Practical Issues in the Computation of Galois Group of Polynomials.* – DEA, Université de Rennes 1, 1994.

[62] Lehobey (Frédéric). – Resolvent computations by resultants without extraneous powers. *In : Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, éd. par Küchlin (Wolfgang). The Association for Computing Machinery, pp. 85–92. – ACM.

[63] Lenstra (A. K.), Lenstra, H. W. (Jr.) et Lovász (L.). – Factoring polynomials with rational coefficients. *Mathematische Annalen*, vol. 261, n° 4, 1982, pp. 515–534.

[64] Liu (Zhuojun) et Wang (P. S.). – Height as a coefficient bound for univariate polynomial factors (part I). *SIGSAM Bulletin (ACM Special Interest Group on Symbolic and Algebraic Manipulation)*, vol. 28, n° 2, août 1994, pp. 20–27.

[65] Liu (Zhuojun) et Wang (P. S.). – Height as a coefficient bound for univariate polynomial factors (part II). *SIGSAM Bulletin (ACM Special Interest Group on Symbolic and Algebraic Manipulation)*, vol. 28, n° 3/4, décembre 1994, pp. 1–9.

[66] Loos (Ruediger). – Computing rational zeros of integral polynomials by p-adic expansion. *SIAM J. Comput.*, vol. 12, 1983, pp. 286–293. – Zbl536.68045.

[67] Malliavin (M.-P.). – *Algèbre commutative, applications en géométrie et théorie des nombres.* – Masson, 1985.

[68] McEliece (R.J.). – Factorization of polynomials over finite fields. *Math. Comput.*, vol. 23, 1969, pp. 861–867. – Zbl185.11301.

[69] McKay (J.). – Some remarks on computing Galois groups. *SIAM Journal on Computing*, vol. 8, n° 3, août 1979, pp. 344–347. – MR80m:12005.

[70] McKay (John) et Soicher (Leonard). – Computing Galois groups over the rationals. *Journal of Number Theory*, vol. 20, 1985, pp. 273–281.

[71] Mignotte (Maurice). – An Inequality About Factors of Polynomials. *Mathematics of Computation*, vol. 28, n° 128, octobre 1974, pp. 1153–1157.

[72] Mignotte (Maurice). – Some inequalities about univariate polynomials. *In : Symbolic and algebraic computation, Proceedings of ACM Symposium, Snowbird/Utah 1981*, pp. 195–199.

[73] Mignotte (Maurice). – Some useful bounds. *In : Computer algebra, symbolic and algebraic computation, Comput. Suppl. 4*, pp. 259–263. – Springer, 1982.

[74] Mignotte (Maurice). – *Mathématiques pour le calcul formel.* – Presses Universitaires de France, 1989.

[75] Mignotte (Maurice) et Glesser (Philippe). – On the smallest divisor of a polynomial. *Journal of Symbolic Computation*, vol. 17, n° 3, 1994, pp. 277–282.

[76] Miller (G. A.). – On the transitive substitution groups of degree thirteen and fourteen. *Quarterly Journal of Pures and Applied Maths*, vol. 29, 1898, pp. 224–249.

[77] Moenck (Robert T.). – On the efficiency of algorithms for polynomials factoring. *Math. Comput.*, vol. 31, 1977, pp. 235–250. – Zbl348.65045.

[78] Monagan (Michael B.). – A heuristic irreducibility test for univariate polynomials. *Journal of Symbolic Computation*, vol. 13, n° 1, 1992, pp. 47–57.

[79] Musser (David R.). – Multivariate polynomial factorization. *Journal of the Association for Computing Machinery*, vol. 22, n° 2, avril 1975, pp. 291–308.

[80] Musser (David R.). – On the efficiency of a polynomial irreducibility test. *Journal of the Association for Computing Machinery*, vol. 25, n° 2, avril 1978, pp. 271–282.

[81] Naudin (Patrice) et Claude (Quitté). – Cours de DEA 1993–1994, algorithmique en théorie des corps. – Prépublication du Département de Mathématiques de l'Université de Poitiers, 40, Avenue du Recteur Pineau, 86022 POITIERS cedex, février 1994.

[82] Oesterlé (Joseph). – Versions effectives du théorème de Chebotarev sous l'hypothèse de Riemann generalisée. *Astérisque*, vol. 61, 1979, pp. 165–167. – Zbl418.12005.

[83] Rabin (Michael O.). – Probabilistic algorithm for testing primality. *J. Number Theory*, vol. 12, 1980, pp. 128–138. – Zbl426.10006.

[84] Ree (Rimhak). – Lie elements and an algebra associated with shuffles. *Ann. of Math. (2)*, vol. 68, 1958, pp. 210–220.

[85] Rennert (Nicolas) et Valibouze (Annick). – Calcul de résolvantes avec les modules de Cauchy. – À paraître dans Experimental Mathematics.

[86] Royle (G. F.). – The transitive groups of degree twelve. *Journal of Symbolic Computation*, vol. 4, 1987, pp. 225–268.

[87] Schönert (Martin). – *Groups, Algorithms and Programming (GAP).* – RWTH, Aachen, 1992, lehrstuhl d für mathematik édition.

[88] Shoup (Victor). – A new polynomial factorization algorithm and its implementation. *Journal of Symbolic Computation*, vol. 20, n° 4, 1995, pp. 363–397.

[89] Soicher (Leonard). – The computation of Galois groups, avril 1981. Thèse de Master, Concordia University, Montréal.

[90] Soicher (Leonard H.). – An algorithm for computing Galois groups. *In : Computational Group Theory*, pp. 291–296. – London, Academic Press, 1984.

[91] Stauduhar (Richard P.). – The determination of Galois groups. *Mathematics of Computation*, vol. 27, n° 124, octobre 1973, pp. 981–996.

[92] Stevenhagen (P.) et Lenstra, H. W. (Jr.). – Chebotarëv and his density theorem. *The Mathematical Intelligencer*, vol. 18, n° 2, 1996, pp. 26–37. – MR97e:11144.

[93] Trager (Barry M.). – Algebraic factoring and rational function integration. *In : Symbolic and algebraic computation, Proc. 1976 ACM Symp.*, pp. 219–226. – Yorktown Heights/N.Y., 1976. Zbl498.12005.

[94] Valibouze (Annick). – Galois groups of all polynomials with applications up to degree 7. – Prépublication.

[95] Valibouze (Annick). – Théorie de Galois constructive. – Université de Paris 6, décembre 1994. Mémoire d'habilitation.

[96] Valibouze (Annick). – Computation of the Galois groups of the resolvent factors for the direct and inverse Galois problems. *In : proceedings AAECC'95, Applied algebra, algebraic algorithms and error-correcting codes. 11th international symposium, AAECC-11, Paris, France July 17-22*, éd. par Cohen (G.), Giusti (M.) et Mora (T.), pp. 456–468. – Berlin, Springer Verlag, 1995. (ISBN 3-540-60114-7) [ISSN 0302-9743].

[97] Valibouze (Annick). – Cours du DEA algorithmique, Université Paris VI. – 1996. Université de Paris VI.

[98] Valibouze (Annick). – *Modules de Cauchy, polynômes caractéristiques et résolvantes.* – Rapport interne n° 95/62, Laboratoire Informatique Théorique et Programmation, janvier 1996.

[99] Valibouze (Annick). – *Construction de l'idéal des relations entre les racines d'un polynôme.* – Rapport technique, LIP6- UPMC- 4, Place Jussieu F-75252 Paris Cedex 05, LIP 6, 1997.

[100] Valibouze (Annick). – Étude des relations algébriques entre les racines d'un polynôme d'une variable. *Bulletin of the Belgian Mathematical Society Simon Stevin*, à paraître.

[101] van der Linden (F. J.). – The computation of Galois groups. *In : Computational methods in number theory, Part II*, éd. par Lenstra Jr (H. W.) et Tijdeman (R.), pp. 199–211. – Amsterdam, Mathematisch Centrum, 1984. MR85f:11078.

[102] van der Waerden (B. L.). – *Modern Algebra.* – New York, Ungar, 1949. Traduction anglaise.

[103] Viry (Guy). – Algorithme de factorisation des polynômes à coefficients entiers. – thèse de Doctorat de l'Université de Nancy I, septembre 1989.

[104] von zur Gathen (Joachim). – Hensel and Newton methods in valuation rings. *Mathematics of Computation*, vol. 42, n° 166, 1984, pp. 637–661.

[105] von zur Gathen (Joachim). – Functional decomposition of polynomials: the tame case. *Journal of Symbolic Computation*, vol. 9, n° 3, 1990, pp. 281–299.

[106] von zur Gathen (Joachim) et Gerhard (Jürgen). – *Modern computer algebra.* – New York, Cambridge University Press, 1999, xiv+753p.

[107] von zur Gathen (Joachim), Kozen (Dexter) et Landau (Susan). – Functional decomposition of polynomials. *In : Proceedings of 28th IEEE Symposium on Foundations of Computer Science*, pp. 127–131. – Los Angeles CA, 1987.

[108] von zur Gathen (Joachim) et Shoup (Victor). – Computing Frobenius maps and factoring polynomials. *Computational Complexity*, vol. 2, n° 3, 1992, pp. 187–224.

[109] Wang (Paul S.). – Early detection of true factors in univariate polynomial factorization. *In : Computer algebra (London, 1983)*, pp. 225–235. – Berlin, Springer, 1983.

[110] Wang (Paul S.). – Parallel univariate p-adic lifting on shared-memory multiprocessors. *Proceedings of ISSAC'92*, 1992, pp. 168–176.

[111] Weinberger (P. J.). – Finding the number of factors of a polynomial. *Journal of Algorithms*, vol. 5, n° 2, 1984, pp. 180–186. – MR86h:11110.

[112] Williamson (Clifton J.). – On the algebraic construction of tri-diagonal matrices with given characteristic polynomial. – Prépublication.

[113] Williamson (Clifton J.). – Polynomials with dihedral Galois groups II. – Prépublication.

[114] Wilson (R.L.). – A method for the determination of the Galois group. *Duke Math. J.*, vol. 17, 1950, pp. 403–408. – Zbl039.01101.

[115] Yokoyama (Kazuhiro), Noro (Masayuki) et Takeshima (Taku). – Solutions of systems of algebraic equations and linear maps on residue class rings. *Journal of Symbolic Computation*, vol. 14, 1992, pp. 399–417.

[116] Yun (David Y. Y.). – On squarefree decomposition algorithms. *Proceedings SYMSAC*, 1976, pp. 26–35.

[117] Zassenhaus (H.). – On Hensel factorization I. *J. Number Theory*, vol. 1, 1969, pp. 291–311. – Zbl188.33703.

Index

Computation and interactive factorization of Lagrange resolvents in computational Galois theory

Computational Galois theory aims at determination, up to conjugation, of the Galois group of a polynomial f.

This computation may be performed from the factorization of polynomials, depending on the polynomial f, which have same coefficient field as the polynomial f : Lagrange resolvents.

Degrees of factors of well chosen resolvents and properties of their Galois groups ensure to always determine Galois group of polynomial f among all possible groups. They are known from the classification of sub-groups of the symmetric group which has been performed, currently, up to order 31.

Efficient computation of interesting Lagrange resolvents and factorization of these resolvents are key points in determination of Galois group by resolvents.

The methods for computing Lagrange resolvents which are based on elimination (resultant) generate parasitic factors and powers.

Group theory gives some information on possible degrees of factors of resolvents as they are not random.

Our work improves two methods for symbolic computation of resolvents which are based on resultant and remove parasitic factors and degrees.

We also show how to adapt polynomial factorization algorithms in order to take into account information known in advance on resolvents.

Factorization being for us a tool more than a goal, we introduce the concept of interactive factorization in order to make quickly available any new information on factorization found along the factorization process.

This concept is not specific to resolvent factorization in computational Galois theory. It may be used for any problem of factorization for which is required less than a complete factorization.

Calcul et factorisation interactive de résolvantes de Lagrange en théorie de Galois effective

La théorie de Galois effective cherche à déterminer, à conjugaison près, le groupe de Galois d'un polynôme f.

Cette recherche peut se faire à partir de la factorisation de polynômes déduits du polynôme f, les résolvantes de Lagrange, qui ont même corps des coefficients que le polynôme f.

Les degrés des facteurs de résolvantes bien choisies, ainsi que les propriétés des groupes de Galois de ces facteurs, permettent de toujours déterminer le groupe de Galois du polynôme f parmi les groupes possibles. Ils sont connus par une classification des sous-groupes du groupe symétrique qui, à ce jour, a été effectuée jusqu'à l'ordre 31.

Le calcul efficace de résolvantes de Lagrange intéressantes et la factorisation de ces résolvantes constituent les points clefs de la recherche du groupe de Galois par les résolvantes.

Les méthodes de calcul de résolvantes de Lagrange par l'élimination (le résultant) engendrent des facteurs et des puissances parasites.

La théorie des groupes fournit des informations sur les degrés possibles, qui ne sont donc pas quelconques, des facteurs des résolvantes.

Notre travail améliore deux méthodes de calcul symbolique des résolvantes, basées sur le résultant, qui suppriment facteurs et puissances parasites.

Nous montrons aussi comment adapter les algorithmes de la factorisation des polynômes pour utiliser les informations connues *a priori* sur les résolvantes.

La factorisation étant pour nous un moyen et non un but, nous introduisons le concept de factorisation interactive pour rendre immédiatement accessibles les nouvelles informations sur la factorisation des résolvantes trouvées au cours du processus de factorisation.

Ce concept n'est pas spécifique à la factorisation de résolvantes en théorie de Galois effective. Il peut être utilisé pour toute factorisation de polynômes lorsque l'information cherchée ne demande pas forcément une factorisation complète.