Algorithmes efficaces pour l' étude des zéros réels des systèmes polynomiaux

Fabrice Rouillier

Table des Matières

Liste des Figures

Introduction

Les objectifs

Cette thèse est intitulée *Algorithmes efficaces pour l'étude des zéros réels des systèmes polynomiaux.*

Notre objectif est de fournir des méthodes de calcul automatiques et exactes permettant de déterminer, pour un système polynomial :

- nombre de zéros réels ou complexes,
- signes de polynômes en les zéros réels d'un système,
- codage des zéros réels / simplification du système.

Par *algorithmes*, nous entendons des méthodes systématiques, nécessitant un nombre fini d'étapes où tous les cas particuliers sont traités. Dans le cas où une technique ne s'applique que sous certaines conditions, (par exemple idéal radical), la vérification de celles-ci doit faire partie intégrante de l'algorithme, et il en sera tenu compte pour tout étude de complexité.
Le terme *efficace* englobe plusieurs notions :

- efficacité théorique :
  - nombre d'opérations arithmétiques nécessaires
  - taille des coefficients dans les calculs intermédiaires
  - taille des résultats
- efficacité pratique
  - performances chronométriques de l'implantation : les outils développés dans cette thèse ont tous été implantés dans un logiciel que nous avons nommé *RealSolving* (décrit en annexe), développé pendant le projet européen PoSSo[1]. Les mesures ont été effectuées sur un panel d'exemples issus de la *PoSSo-Test-Suite* (décrite en annexe) et de problèmes pratiques.
  - utilisation pour des problèmes autres que ceux conçus spécialement pour tester les performances des logiciels de ce type.

[1] Polynomial System Solving : Projet ESPRIT BRA contract 6846 POSSO

Les limites de l'étude

Un de nos principes de base à été de réutiliser au maximum les résultats connus.

En amont, nous supposons que l'idéal engendré par le système considéré est donné par une base de Groebner. En fait, lorsque K est un corps et $\mathcal{I}$ l'idéal engendré par un système admettant un nombre fini de zéros, le point d'entrée que nous fixons est constitué de :

- une base $\mathcal{A} = K[X_1, \ldots, X_n]/\mathcal{I}$ (qui est alors un K-espace vectoriel de dimension finie)
- les endomorphismes de multiplication par une variable dans $\mathcal{A}$

Dans le cas de systèmes à coefficients rationnels, les tailles des entiers intervenant dans ces expressions nous serviront de mesure étalon pour étudier le comportement de nos algorithmes (croissance des coefficients en cours de calcul, taille des résultats).

En aval, nous disposons d'une série d'algorithmes pour le traitement des polynômes en une variable. Nous rappelons, dans le chapitre 4, deux algorithmes standards pour le comptage (Sturm-Habicht) ou l'isolation par intervalles rationnels (Uspensky), utilisés dans ce travail.

En ce qui concerne l'implantation des algorithmes, nous avons décidé d'interfacer notre code avec les outils les plus performants du moment pour le calcul de bases de Groebner:

- Le logiciel *Gb*, écrit par Jean-Charles Faugère[2]
- La librairie PoSSo, écrite en C++, dont nous avons utilisé les outils de base (arithmétique - gestion de mémoire), *RealSolving* faisant par ailleurs partie intégrante de la distribution officielle de cette librairie.

Par souci d'homogénéité, les algorithmes pour le traitement des polynômes en une variable ont été implantés eux aussi dans *RealSolving*, bien que sortant du cadre que nous nous sommes fixé.

Résultats et résumés des différents chapitres

Première Partie : Systèmes zéro-dimensionnels

La première partie de ce document représente le cœur de la thèse. Nous établissons un certain nombre d'outils pour le traitement des systèmes zéro-dimensionnels (admettant un nombre fini de racines).

[2] Chargé de recherches CNRS au LITP - université de Paris IV

Chapitre 1 : Préliminaires

La partie *Préliminaires* est réservée au rappel des résultats d'algèbre linéaire les plus couramment utilisés pour l'étude des systèmes polynomiaux (si K est un corps et $\mathcal{I} \subset K[X_1, \ldots, X_n]$ un idéal zéro-dimensionnel, $K[X_1, \ldots, X_n]/\mathcal{I}$ est un K-espace vectoriel).

Chapitre 2 : Calculs dans le quotient

Comme annoncé dans la partie préliminaire, nous allons utiliser un certain nombre de techniques d'algèbre linéaire. Il va sans dire que le comportement de nos algorithmes sera conditionné par la qualité de la représentation choisie pour $K[X_1, \ldots, X_n]/\mathcal{I}$.

Un outil incontournable à l'heure actuelle, pour effectuer des calculs modulo un idéal $\mathcal{I} \subset K[X_1, \ldots, X_n]$ (zéro-dimensionnel ou pas), est l'algorithme de Buchberger. Il permet de calculer une famille agréable de générateurs de $\mathcal{I}$ (base de Groebner), dont les propriétés sont très utiles pour travailler dans $K[X_1, \ldots, X_n]/\mathcal{I}$ (par exemple, il existe un algorithme pour déterminer, de manière unique, la classe d'un polynôme modulo $\mathcal{I}$). Sur ce sujet, nous ne ferons que rappeler les résultats désormais standards. En fait, étant donné un idéal zéro-dimensionnel $\mathcal{I} \subset K[X_1, \ldots, X_n]$, notre but est de calculer la *table de multiplication* de $K[X_1, \ldots, X_n]/\mathcal{I}$. C'est à dire :

- Construire une base $\mathcal{B} = \{\omega_1, \ldots, \omega_D\}$ du K-espace vectoriel $K[X_1, \ldots, X_n]/\mathcal{I}$
- Donner l'ensemble des produits réduits $\overrightarrow{\omega_i\omega_j}$ où $\overrightarrow{\omega_i\omega_j}$ est le vecteur dont les coordonnées sont celles de la classe de $\omega_i\omega_j$ dans $K[X_1, \ldots, X_n]/\mathcal{I}$, exprimées dans la base $\mathcal{B}$.

La motivation initiale de ce travail était de fournir un outil efficace pour construire les matrices de formes quadratiques particulières nécessaires à pour l'un de nos algorithmes (méthode de Hermite).

Nous avons rapidement remarqué qu'une stratégie naïve ne permettait pas de calculer efficacement ces formes quadratiques (nécessitant la connaissance des expressions $\overrightarrow{\omega_i\omega_j}$) pour deux raisons :

- Le coût de l'opération *réduction d'un polynôme* modulo $\mathcal{I}$ via l'algorithme NF (*Normal Form* décrit dans les rappels sur les bases de Groebner) est important.
- Beaucoup de calculs redondants sont effectués.

Dans un premier temps, nous avons utilisé un procédé informatique général pour éviter les redondances : une table de *Hachage*. Cette solution nous ayant permis de réduire de façon considérable le nombre de calculs.
Toutefois, des calculs inutiles apparaissent, même si l'on utilise un tel procédé. Ils sont intrinsèques à l'utilisation de la fonction NF.

Nous montrons, dans cette partie, comment combiner quelques propriétés algébriques et un procédé informatique pour calculer l'ensemble des réductions $\overrightarrow{\omega_i\omega_j}$ (formant ce que nous appelons une table de multiplication de $K[X_1,\ldots,X_n]/\mathcal{I}$), pour optimiser le nombre et la complexité des calculs, l'idée principale étant de ne pas dissocier le procédé informatique du problème mathématique.

Chapitre 3 : Résolution des systèmes zéro-dimensionnels

Nous abordons, dans ce chapitre, le problème de la simplification des systèmes polynomiaux et de l'élimination de variables.

On trouve, dans la littérature, beaucoup de résultats utilisant ce que nous avons appelé les polynômes séparants. Si K est un corps de caractéristique 0 de clôture algébrique C, pour un idéal $\mathcal{I} \subset K[X_1,\ldots,X_n]$ dont l'ensemble des zéros $V(\mathcal{I}) \subset C^n$ est fini et non vide, l'idée est de trouver une fonction polynomiale, injective de $V(\mathcal{I})$ dans C, de sorte à se ramener à l'étude d'un ensemble fini de points (algébriques) de C.

Dans le cas d'un idéal radical, ceci revient à déterminer un élément primitif (voir par exemple [GH91b] ou [BW93]), mais, dans le cas général, le problème est plus complexe. On peut imaginer se ramener systématiquement au cas radical (des solutions algorithmiques existent d'ailleurs à cet effet - voir [GM89] par exemple), mais on perd alors toute information sur les multiplicités des racines.

Dans [Ca88], un calcul de u-résultant fournit, après déformation du système (ajout d'une variable), une représentation agréable des zéros, les coordonnées des solutions étant exprimées comme fractions rationnelles en les racines d'un unique polynôme en une variable. Un procédé identique est présent dans [Re92], utilisant une arithmétique à base d'infinitésimaux.

Une autre solution est proposée dans [ABRW94], autorisant, en particulier, la représentation de $\mathcal{I}$ par une base de Groebner. Cette fois, aucune déformation n'est utilisée et les calculs s'en trouvent simplifiés.

Nous proposons une définition générale de la représentation univariée, indépendante de toute méthode de calcul, où les expressions ne dépendent pas des multiplicités. Nous montrons que tout idéal zéro-dimensionnel $\mathcal{I} \subset K[X_1,\ldots,X_n]$ est équivalent, en un certain sens, à un idéal $\mathcal{J} \subset K[T]$, définissant ainsi une notion simple de *Résolution d'un système zéro-dimensionnel*.

Nous montrons ensuite qu'il existe un algorithme permettant de calculer explicitement un isomorphisme d'ensembles algébriques $\phi : V(\mathcal{I}) \longrightarrow V(\mathcal{I})$, préservant les multiplicités des racines et défini par un ensemble de $n+1$ polynômes de $K[T]$ (Représentation univariée rationnelle). Cet isomorphisme possède un certain nombre de propriétés agréables comme la préservation des racines réelles, ou encore que les polynômes de la représentation univariée rationnelle permettent, dans certains cas, de calculer simplement une base de Groebner de $\mathcal{I}$ pour l'ordre lexicographique.

Pour le calcul explicite de cette représentation, nous introduisons une extension des notions habituelles de fonctions symétriques élémentaires et de sommes de Newton. Une généralisation de la formule de Newton nous donne alors un algorithme efficace (RUR) pour le calcul de notre isomorphisme, lorsqu'un polynôme séparant est connu. Pour le cas général, nous utilisons une idée présente dans [Ca88], [Re92] ou encore [ABRW94], dont le principe est de déterminer une famille finie de polynômes dont l'un sépare assurément $V(\mathcal{I})$. Un test nouveau et efficace en pratique, nous permet de vérifier, a posteriori, si un élément choisi arbitrairement est séparant ou non.

Une attention toute particulière est accordée au cas des systèmes à coefficients entiers, pour lesquels nous proposons une méthode modulaire pour la détermination d'un élément séparant $V(\mathcal{I})$.

Une analyse détaillée de la complexité montre que le calcul de notre représentation univariée rationnelle par les algorithmes décrits dans cette thèse est :

- Efficace (nombre d'opérations arithmétiques optimisé, taille des calculs intermédiaires, pas de déformation du système initial - l'ajout d'une variable complique fortement les calculs),

- Utilisable (taille de la sortie).

Dans le cas particulier des systèmes à coefficients rationnels, nous montrons de plus que le fait de ne pas connaître de polynôme séparant $V(\mathcal{I})$ n'influe que peu, en pratique, sur le comportement de l'algorithme.

Chapitre 4 : Algorithmes pour le comptage des zéros réels

Le comptage des zéros réels était notre principal souci en débutant ce travail, le premier algorithme mis au point ayant d'ailleurs été l'algorithme de Hermite.

Comme montré dans [PRS93] ou [BW92], le nombre de zéros réel (resp. complexes) d'un idéal zéro-dimensionnel $\mathcal{I} \subset K[X_1, \ldots X_n]$ peut être simplement exprimé par la signature d'une forme quadratique (forme quadratique de Hermite généralisée), définie en termes de traces d'endomorphismes simples de $K[X_1, \ldots, X_n]/\mathcal{I}$. Deux problèmes se posaient en pratique :

- La construction de la forme quadratique, les outils standards ne permettant pas d'effectuer ce calcul de manière efficace,

- La réduction de la forme quadratique.

La table de multiplication décrite dans le chapitre 2 apporte une réponse au premier point.

En ce qui concerne le deuxième point, l'algorithme standard de Gaus n'est pas suffisant pour effectuer certains calculs. En supposant que K soit le corps des fractions d'un anneau Z, un simple changement de variables nous permet de supposer que la forme quadratique à réduire est à coefficients dans Z. En généralisant les identités de Bareiss (voir [B68]) au cas des formes quadratiques, nous proposons une nouvelle méthode permettant, pour une matrice de dimension D, d'effectuer la réduction en $O(D^3)$ opérations arithmétiques dans Z (et non dans K, ce qui diminue très nettement le coût des opérations de base), tout en contrôlant efficacement la taille des calculs intermédiaires.

En combinant la table de multiplication et ce nouvel algorithme, nous obtenons une méthode très efficace, en pratique, pour compter le nombre de zéros réels (ou complexes) d'un idéal zéro-dimensionnel donné. Nous obtenons une complexité théorique meilleure que dans [Moel93] ($O(D^4+nD^3)$ opérations arithmétiques contre $O(D^5)$) et plus précise que celle présentée dans [PRS93].

Comme l'illustre le chapitre 3, une partie de notre travail à consisté à réduire l'étude de systèmes zéro-dimensionnels à celle de polynômes en une variable, pour lesquels beaucoup d'outils ont déjà été développés. Nous avons choisi de rappeler brièvement (des études plus détaillées sont présentes dans [Rio], [BCL82], ou encore [GLRR90]) deux méthodes, éprouvées en pratique, nous semblant pour l'heure parmi les plus efficaces (Sturm-Habicht et Uspensky). Ceci nous servira de base pour comparer, en termes de complexité mais également d'un point de vue pratique, deux types de stratégies pour le comptage des zéros réels d'un système zéro-dimensionnel :

- Traitement direct par l'algorithme de Hermite
- Réduction à un problème plus simple (représentation univariée rationnelle - rappelons que les racines réelles sont préservées par cette transformation), puis application d'un d'un algorithme traitant les polynômes en une variable.

Enfin, nous présentons une évolution de l'algorithme *Inégalités simultanées* proposé dans [RS90], dont le but est de déterminer, pour une famille de polynômes $Q_1, \ldots, Q_s \subset K[X_1, \ldots, X_n]$ donnée, le cardinal de toutes les conditions de signes réalisables par les polynômes Q_i en les zéros d'un idéal $I \subset K[X_1, \ldots, X_n]$ (zéro-dimensionnel), c'est à dire le cardinal de tous les ensembles de la forme

$$\{\alpha \in V(\mathcal{I}) \ , \ et \ , \ \forall i \ , \ Sign(Q_i(\alpha)) = \epsilon_i\}$$

où les ϵ_i sont à valeurs dans $\{+, 0, -\}$. Cette méthode, mise au point en collaboration avec Guadaluppe Trujillo (Université de Cantabria), nécessite l'utilisation d'une *boite noire* permettant de calculer les quantités de la forme :

$$\sharp\{\alpha \in V(I) \cap R \ , \ Q(\alpha) > 0\} - \sharp\{\alpha \in V(I) \cap R \ , \ Q(\alpha) < 0\},$$

pour $Q \in K[X_1, \ldots, X_n]$, soit par exemple l'algorithme de Sturm-Habicht lorsque $n = 1$ ou l'algorithme de Hermite, lorsque $n > 1$.

Deuxième Partie : Applications

La deuxième partie de cette thèse est dédiée à l'étude de trois applications issues de domaine différents. Il était important de vérifier que nos méthodes s'appliquaient sur des problèmes autres que ceux que nous construisions nous mêmes.

La vocation de la boite à outils RealSolving est de permettre l'utilisation d'algorithmes de calcul formel sophistiqués par des utilisateurs non spécialistes, et d'automatiser le plus possible de taches calculatoires.

Chapitre 5 : Robots parallèles

Le problème qui nous à été proposé par J.P. Merlet et L. Tancredi (projet Prisme de l'INRIA Nice), ainsi que par D. Lazard et J.C. Faugère (Université de Paris 6) est l'étude du problème dit *Direct* pour les robots parallèles : on se donne les configurations de deux ensembles rigides munis chacun de 6 articulations (notés $\{M_1, \ldots, M_6\}$ et $\{A_1, \ldots, A_6\}$). En supposant connues les longueurs des liaisons $(A_i, B_i)_{i=1\ldots6}$ (supposées rectilignes et rigides), le problème consiste à déterminer le nombre positions possibles de la plate-forme ($\{M_1, \ldots, M_6\}$) dans un repère lié à la base ($\{A_1, \ldots, A_6\}$).

Bien sur, il est possible par certains algorithmes numériques de déterminer avec une bonne précision les solutions de tels problèmes. L'utilisation du calcul exact devient intéressante pour les spécialistes dans la recherche du nombre de positions dans l'espace d'un robot donné.

Bien que le nombre maximum de positions d'un robot parallèle dans l'espace complexe soit connu (40) et ait été calculé par divers auteurs (voir par exemple [Mou] ou [Laz92b]), le problème reste entier pour le nombre de solutions réelles. Jusqu'à ce jour, le nombre maximum de positions réelles obtenu sur un robot sans articulation double est de 16 (voir [Mer90]) et de 24 pour un robot ayant une articulation double (voir [IPC93]). Quelques essais de calcul de bornes ont été tentés. En particulier, D. Lazard conjecture dans [Laz92b] que le nombre maximum de positions réelles est majoré par 16 ou 32, selon la configuration du robot.

Nous montrons, dans la deuxième partie de ce chapitre, comment nous avons, en collaboration avec L.Tancredi et Jean-Charles Faugère, construit un robot général (sans articulation double) admettant 24 positions (réelles).

Chapitre 6 : Synthèse de filtres

Nous décrivons, dans ce chapitre, une application en traitement de signal. Celle - ci met en œuvre les bases de Groebner (Gb) et les méthodes développées dans cette thèse (RealSolving), qui se révèlent utilisables, en pratique.

L'étude à été menée en collaboration avec Jean-Charles Faugère et Francois Moreau, ingénieur au CCETT (Rennes).

Nous avons cherché à synthétiser des filtres *passe bas* générant des une base d'ondelettes orthogonale. Pour ne pas rentrer dans les détails, disons que ceci revient à déterminer un polynôme (filtre) satisfaisant quelques conditions algébriques et optimal pour certains critères différentiels.

Nous nous intéressons au cas de signaux bidimensionnels (servant par exemple pour le traitement des images), pour lesquels peu de résultats tenant compte de ce type de contraintes sont connus. En particulier, nous prenons comme base d'étude, les filtres déterminés par la cascade de Vetterli-Kovacevic (voir [KV95]).

Nous montrons, dans un premier temps, comment, par le calcul formel, nous avons pu retrouver les meilleurs résultats traités à ce jour dans la littérature spécialisée.

L'utilisation de Gb et de RealSolving a permis ensuite de construire des filtres dont les performances sont nettement supérieures à celles des filtres connus pour la cascade de Vetterli-Kovacevic.

Chapitre 7 : Le problème d'interpolation de Birkhoff

Dans ce chapitre, nous montrons comment le calcul formel peut aider à l'étude de méthodes numériques.

Étant donnée une fonction $f : \mathbb{R} \to \mathbb{R}$, le problème d'interpolation de Birkhoff consiste à décider si la connaissance d'un nombre fini de valeurs de f où de certaines de ses dérivées en un ensemble de points fixé, suffit à déterminer un polynôme p interpolant f.

Une première partie sera consacrée à l'établissement d'un critère simple (proposé dans [G94]), permettant de ramener le problème à une étude d'hypersurfaces. Plus précisément, nous montrons que ce problème se résume à déterminer si une hypersurface admet ou non des points réels dont les coordonnées sont non nulles.

La deuxième partie décrira un certain nombre de techniques permettant, dans des cas favorables, de se ramener à l'étude de systèmes zéro-dimensionnels. Le principe de base étant de compter le nombre de points critiques réels d'une application particulière (fonction distance).

Enfin, une partie expérimentale montrera que l'on peut résoudre, en combinant ces différentes techniques, le problème de Birkhoff pour certaines classes d'exemples qui n'avaient pas pu être traitées jusqu'à ce jour.

Première Partie

Systèmes zéro-dimensionnels

Chapitre 1

Préliminaires

Résumé

La partie *Préliminaires* est réservée au rappel des résultats d'algèbre linéaire les plus couramment utilisés pour l'étude des systèmes polynomiaux (si K est un corps et $\mathcal{I} \subset K[X_1, \ldots, X_n]$ un idéal zéro-dimensionnel, $K[X_1, \ldots, X_n]/\mathcal{I}$ est un K-espace vectoriel).

Soient K un corps ordonné de clôture algébrique C, $\mathcal{S} \subset K[X_1, \ldots, X_n]$ un système polynomial. Nous désignerons par $\mathcal{I}$ (resp. $\mathcal{I}_C$), l'idéal de $K[X_1, \ldots, X_n]$ (resp. $C[X_1, \ldots, X_n]$) engendré par $\mathcal{S}$. Rappelons que l'ensemble des zéros $V(\mathcal{I}) \subset C^n$ de $\mathcal{I}$ est fini si et seulement si le K-espace vectoriel $\mathcal{A} = K[X_1, \ldots, X_n]/\mathcal{I}$ est de dimension finie. Dans ce cas,

$$\mathcal{A}_C = C[X_1, \ldots, X_n]/\mathcal{I}_C = C \otimes_K \mathcal{A} = \prod_{\alpha \in V(\mathcal{I})} A_\alpha$$

où A_α est le localisé de $C[X_1, \ldots, X_n]/\mathcal{I}_C$ en α. Le localisé A_α est alors un sous espace vectoriel de $\mathcal{A}_C$ dont la dimension est égale, par définition, à la multiplicité de α que nous noterons $\mu(\alpha)$.

A toute base $\mathcal{B}$ du K-espace vectoriel $\mathcal{A}$, on peut associer canoniquement une base une base $\mathcal{B}_C$ du C-espace vectoriel $C \otimes_K \mathcal{A}$. En conséquence, $dim_K(K[X_1, \ldots, X_n]/\mathcal{I}) = dim_C(C[X_1, \ldots, X_n]/\mathcal{I}_C)$. Les propriété que nous utiliserons le plus souvent sont résumées par la proposition suivante :

Proposition 1.0.1 *Soit $\mathcal{I}$ un idéal de $K[X_1, \ldots, X_n]$, différent de $[1]$, admettant un nombre fini de zéros dans C^n. Alors :*

$$dim_K(K[X_1, \ldots, X_n]/\mathcal{I}) = dim_C(C[X_1, \ldots, X_n]/\mathcal{I}_C) = \sum_{\alpha \in V(\mathcal{I})} \mu(\alpha).$$

L'idéal $\mathcal{I}$ (resp. le système polynomial $\mathcal{S}$) est dans ce cas un idéal zéro-dimensionnel *(resp.* système zéro-dimensionnel*) de $K[X_1, \ldots, X_n]$.*

1.1 Le théorème de Stickelberger

Nous rappelons, dans ce paragraphe, quelques propriétés importantes de l'application *multiplication par un polynôme dans* $K[X_1, \ldots, X_n]/\mathcal{I}$, lorsque l'idéal $\mathcal{I}$ est zéro-dimensionnel.

Notation 1.1.1 *Pour tout polynôme $h \in K[X_1, \ldots, X_n]$ nous noterons m_h l'endomorphisme de $K[X_1, \ldots, X_n]/\mathcal{I}$:*

$$\begin{array}{rccc} m_h : & K[X_1, \ldots, X_n]/\mathcal{I} & \longrightarrow & K[X_1, \ldots, X_n]/\mathcal{I} \\ & f & \longmapsto & \bar{h} \cdot f \end{array}$$

où $\bar{h}$ est la classe de h dans $K[X_1, \ldots, X_n]/\mathcal{I}$.

Le théorème suivant récapitule quelques propriétés de m_h. Il énonce des résultats couramment utilisés pour l'étude des systèmes zéro-dimensionels (voir par exemple [BW92],[Ped91],[Yoko92],[PRS93]):

Théorème 1.1.1 *(Théorème de Stickelberger) Soient $\mathcal{I}$ un idéal zéro-dimensionnel de $K[X_1, \ldots, X_n]$ admettant un nombre fini de zéros, et pour tout $h \in K[X_1, \ldots, X_n]$, m_h l'opérataur* multiplication par h *dans $K[X_1, \ldots, X_n]/\mathcal{I}$. Les valeurs propres de m_h sont exactement les $h(\alpha)$, $\alpha \in \mathcal{I}$, avec pour multiplicités respectives $\mu(\alpha)$.*
Cette propriété a beaucoup de conséquences, en particulier :

- *$Det(m_h) = \prod_{\alpha \in V(\mathcal{I})} h(\alpha)^{\mu(\alpha)}$,*
- *$Trace(m_h) = \sum_{\alpha \in V(\mathcal{I})} \mu(\alpha) h(\alpha) \cdot$,*
- *Le polynôme caractéristique de m_h est : $\prod_{\alpha \in V(\mathcal{I})} (T - h(\alpha))^{\mu(\alpha)}$.*

Preuve Nous nous intéressons ici aux valeurs propres de m_h, aussi, sans perte de généralité, nous étendons cannoniquement m_h en un endomorphisme de $C[X_1, \ldots, X_n]/\mathcal{I}_C$.
Supposons $\alpha \in V(\mathcal{I})$ fixé. Vu comme sous-espace vetoriel sur C de $\mathcal{A}_C$, le localisé $\mathcal{A}_\alpha$ est stable sous l'action de m_h. Nous noterons alors

$$m_h^\alpha : \mathcal{A}_\alpha \longrightarrow \mathcal{A}_\alpha$$

la restriction de m_h à $\mathcal{A}_\alpha$.
Posons

$$h_\alpha(X_1, \ldots, X_n) = h(X_1, \ldots, X_n) - h(\alpha) \in C[X_1, \ldots, X_n]$$

En notant m_{h_α}, l'endomorphisme de multiplication par h_α dans $\mathcal{A}_C$ et $m_{h_\alpha}^\alpha$ sa restriction à $\mathcal{A}_\alpha$, le polynôme h_α s'annulant en α, $m_{h_\alpha}^\alpha$ est alors nilpotent. En particulier, le polynôme minimal de $m_{h_\alpha}^\alpha$ admet 0 pour unique racine. Comme C est algébriquement clos, on en déduit que, pour tout $\alpha \in V(\mathcal{I})$, il existe une base de A_α dans laquelle $m_{h_\alpha}^\alpha$ a pour matrice :

$$\begin{pmatrix} 0 & \star & \cdots & \star \\ 0 & 0 & \cdots & \star \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

et par conséquent dans laquelle m_h^α a pour matrice :

$$\begin{pmatrix} h(\alpha) & \star & \cdots & \star \\ 0 & h(\alpha) & \cdots & \star \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & h(\alpha) \end{pmatrix}$$

On peut alors voir que $h(\alpha)$ est une valeur propre de m_h^α de multiplicité $\mu(\alpha) = dim_C(A_\alpha)$. Comme $\mathcal{A}_C = \prod_{\alpha \in V(\mathcal{I})} A_\alpha$, le résultat est démontré. □

1.2 La forme quadratique de Hermite généralisée

Lorsque $\mathcal{I} \subset K[X_1, \ldots, X_n]$ est zéro-dimensionnel, nous avons pu voir que $A = K[X_1, \ldots, X_n]/\mathcal{I}$ est un K-espace vectoriel de dimension finie. Il est donc possible de prendre les traces d'éléments de $End_K(A)$ pour définir une forme bilinéaire symétrique $B : A \times A \longrightarrow K$

$$B(f, g) = Trace(m_{fg})$$

En particulier, à tout polynôme $h \subset K[X_1, \ldots, X_n]$, on associe une forme quadratique Q_h comme suit :

Definition 1.2.1 *Soit $h \in K[X_1, \ldots, X_n]$ on définit une forme bilinéaire $B_h : A \times A \longrightarrow K$ en posant :*

$$B_h(f, g) = B(hf, g) = Trace(m_{fgh})$$

Nous noterons Q_h, la forme quadratique associée à B_h et définie par :

$$Q_h(f) = B_h(f, f)$$

Théorème 1.2.1 *Soient un corps ordonné K, un idéal zéro-dimensionnel $\mathcal{I} = \langle f_1, \ldots, f_s \rangle$ de $K[X_1, \ldots, X_n]$, R la clôture réelle de K, C la clôture algébrique de R et h un polynôme de $K[X_1, \ldots, X_n]$. Si $V_R(\mathcal{I}) = V(\mathcal{I}) \bigcap R^n$, alors:*

A. $\sigma(Q_h) = \sharp\{\alpha \in V_R(I) | h(\alpha) > 0\} - \sharp\{\alpha \in V_R(I) | h(\alpha) < 0\}$

B. $\rho(Q_h) = \sharp\{\alpha \in V(I) | h(\alpha) \neq 0\}$

où $\sigma(Q_h)$ (resp. $\rho(Q_h)$) désigne la signature (resp. le rang) de Q_h.

Ce théorème n'est pas nouveau (voir [BW92],[Ped91],[PRS93]). Nous proposons ici une variante de sa démonstration nous permettant d'introduire une notion de polynôme séparant, utilisée par divers auteurs sous des formes variées (voir par exemple [Ca88], [BW93], [Laz92], [GH91b], [GHMP95], [GVT95], [Re92], [ABRW94]) :

Definition 1.2.2 *Un polynôme $t \in K[X_1, \ldots, X_n]$ sépare $V(\mathcal{I})$ si*

$$\forall \alpha, \beta \in V(\mathcal{I}) \,, \alpha \neq \beta \Rightarrow t(\alpha) \neq t(\beta).$$

L'existence de tels éléments est montrée par le lemme suivant :

Lemme 1.2.1 *Soit $\mathcal{X}$ un ensemble de points de C^n tel que $\sharp\mathcal{X} = d$. La famille de formes linéaires $\{X_1 + iX_2 + \ldots + i^{n-1}X_n$ pour $0 \leq i \leq (n-1)d(d-1)/2\}$ contient au moins un élément séparant $\mathcal{X}$.*

Preuve : Notons $u_i(X) = X_1 + iX_2 + \ldots + i^{n-1}X_n$ et supposons que $(x, y) = ((x_1, \ldots, x_n), (y_1, \ldots, y_n))$ soit un couple de points distincts de $\mathcal{X}$. Comme le polynôme $\sum_{i=1}^{n}(x_i - y_i)X^{i-1}$ admet au plus $n-1$ racines distinctes (il n'est pas indentiquement nul car $x \neq y$), la famille $\{u_0, \ldots, u_{n-1}\}$ contient donc au moins un élément u_k tel que $u_k(x) \neq u_k(y)$. Comme le nombre total de couples de points distincts de $\mathcal{X}$ n'excède pas $d(d-1)/2$, la famille de polynômes $\{X_1 + iX_2 + \ldots + i^{n-1}X_n\ ,\ 0 \leq i \leq (n-1)d(d-1)/2\}$ contient au moins un élément séparant $\mathcal{X}$. □

Ces familles d'éléments séparants ont beaucoup de propriétés utiles pour l'étude des algèbres $K[X_1, \ldots, X_n]/\mathcal{I}$. En particulier :

Lemme 1.2.2 *Soient $\mathcal{I} \subset K[X_1, \ldots X_n]$ un idéal zéro-dimensionnel et $t \subset K[X_1, \ldots X_n]$ un polynôme séparant $V(\mathcal{I})$. Si nous notons $d = \sharp V(\mathcal{I})$, $\{1, t, \ldots, t^{d-1}\}$ est une famille libre de $K[X_1, \ldots, X_n]/\mathcal{I}$.*

Preuve : Soient $a_0, \ldots, a_{d-1}$ des scalaires ($\in K$) tels que

$$g(t) = \sum_{i=0}^{d-1} a_i t^i = 0 \ mod\ \mathcal{I}$$

Pour tout, $\alpha \in V(\mathcal{I})$, $t(\alpha)$ est alors une racine de $g(T) = \sum_{i=0}^{d-1} a_i T^i$. Comme t sépare $V(\mathcal{I})$, le polynôme $g(T)$ admet par conséquent d racines (les $t(\alpha)$, $\alpha \in V(\mathcal{I})$) et est donc identiquement nul. La famille $\{1, t, \ldots, t^{d-1}\}$ est donc libre dans $K[X_1, \ldots, X_n]/\mathcal{I}$. □

Preuve du théorème: Soit t un élément séparant $V(\mathcal{I}) = \{\alpha_1, \ldots, \alpha_d\}$. Si $d = \sharp V(\mathcal{I})$, la famille $\{1, t, \ldots, t^{d-1}\}$ est libre dans $A = K[X_1, \ldots X_n]/\mathcal{I}$. Il est donc possible de trouver des polynômes $\omega_{d+1}, \ldots, \omega_D$ tels que l'ensemble $\mathcal{B}' = \{\omega_1 = 1, \omega_2 = t, \ldots, \omega_d = t^{d-1}, \omega_{d+1}, \ldots, \omega_D\}$ soit une base de A. Pour un polynôme $f \in K[X_1, \ldots, X_n]$ donné, notons $Y_1, \ldots, Y_D$ les coordonnées de la classe de f dans $K[X_1, \ldots, X_n]$, exprimées dans la base $\mathcal{B}'$. Dans ce cas, d'après le théorème 1.1.1, $Q_h(f) = \sum_{i=1}^{d} \mu(\alpha_i)h(\alpha_i)\left(\sum_{j=1}^{D} \omega_j(\alpha_i)Y_j\right)^2$

Comme $\alpha_1, \ldots, \alpha_d$ sont des points distincts de C^n et comme t est un élément séparant $V(\mathcal{I})$, la matrice

$$\begin{pmatrix} 1 & t(\alpha_1) & \ldots & t(\alpha_1)^{d-1} \\ \vdots & & & \vdots \\ 1 & t(\alpha_d) & \ldots & t(\alpha_d)^{d-1} \end{pmatrix}$$

est une matrice de Vandermonde (donc inversible) sous-matrice de la matrice associée aux formes linéaires induites par le changement de variables :

$$Z_i = \sum_{j=1}^{D} \omega_j(\alpha_i) \cdot Y_j\ ,\ i = 1 \ldots d,$$

qui sont donc linéairement indépendantes. Par conséquent :

$$Q_h(f) = \sum_{i=1}^{d} \mu_i h(\alpha_i) Z_i^2$$

et donc :

$$\rho(Q_h) = \sharp\{\alpha \in V(I) | h(\alpha) \neq 0\}$$

Rappelons que C peut-être considéré comme un R-espace vectoriel:

$$C = \{a + ib \ , \ i^2 = -1 \ , \ (a, b) \in R^2\}$$

ce qui nous permet d'utiliser pour la suite de la preuve la terminologie et les notations habituellement réservées aux nombres complexes. Nous noterons donc pour tout $c = a + ib \in C$:

- Partie Réelle : $Re(c) = a$
- Partie Imaginaire : $Im(c) = b$
- Conjugaison : $\overline{c} = a - ib$

Comme dans le cas complexe, l'opération de conjugaison est clairement un automorphisme de R-algèbres laissant R invariant et, par conséquent, $\overline{p(c)} = p(\overline{c})$ pour tout $p \in C[X_1, \ldots, X_n]$, ce qui montre en particulier que si $\alpha \in V(\mathcal{I})$, alors $\overline{\alpha} \in V(\mathcal{I})$. Ainsi, il existe un entier positif ou nul s tel que $\sharp(V(\mathcal{I}) \backslash V_R(\mathcal{I})) = 2s$. Quitte à permuter l'ordre des variables, on peut alors supposer que :

- $\alpha_i \in V(\mathcal{I}) \backslash V_R(\mathcal{I})$ si et seulement si $i \leq 2s$
- $\forall i = 1 \ldots s, \ \alpha_{i+s} = \overline{\alpha_i}$.

Nous avons alors :

$$Q_h = \sum_{i=1}^{s} \mu_i (h(\alpha_i) Z_i^2 + \overline{h(\alpha_i)} Z_{s+i}^2) + \sum_{i=2s+1}^{d} \mu_i h(\alpha_i) Z_i^2$$

Si nous posons :

$$U_i = (X_i + X_{i+s})/2 \ , \ V_i = (X_i + X_{i+s})/(2 \cdot i) \ , \ i = 1 \ldots s$$

et :

$$Q_h^i = h(\alpha_i) Z_i^2 + \overline{h(\alpha_i)} Z_{s+i}^2$$

alors :

$$Q_h^i = h(\alpha_i)(U_i + iU_{i+s})^2 + \overline{h(\alpha_i)}(\overline{U_i + iU_{i+s}})^2 = 2Re(h(\alpha_i)(U_i + iU_{i+s})^2)$$

Si $h(\alpha_i) = a + ib$:

$$Q_h^i = 2aU_i^2 - 2aV_i^2 - 4bU_iV_i$$

d'où :

- $Q_h^i = \frac{2}{a}((aU_i - 2bU_{i+s})^2 - (a^2+4b^2)U_{i+s}^2)$ si $a \neq 0$
- $Q_h^i = -b((U_i + U_{i+s})^2 - (U_i - U_{i+s})^2)$ si $a = 0$.

Les signatures des formes quadratiques Q_h^i sont donc nulles et par conséquent, les formes linéaires associées aux changements de variables effectués étant clairement linéairement indépendantes,

$$\sigma(Q_h) = \sigma(\sum_{i=2s+1}^{d} \mu_i h(\alpha_i) Z_i^2) = \sharp\{\alpha \in V_R(f) \mid h(\alpha) > 0\} - \sharp\{\alpha \in V_R(f) \mid h(\alpha) < 0\}.$$

$\square$

Nous décrivons maintenant quelques conséquences du théorème 1.2.1. En particulier, nous rapelons un critère utile pour tester l'appartenance d'un polynôme au radical d'un idéal.

Le théorème 1.2.1 montre qu'un polyôme h s'annule sur $V(\mathcal{I})$ si et seulement si le rang de la forme quadratique de Hermite généralisée Q_h qui lui est associée est nul c'est à dire si et seulement si Q_h est nulle.

Notation 1.2.1 *Soit Q_h la forme quadratique de Hermite généralisée associée à un polynôme $h \in K[X_1,\ldots,X_n]$, de matrice q_h dans une base $\mathcal{B} = \{\omega_1 = 1, \ldots \omega_D\}$ de $\mathcal{A} = K[X_1,\ldots,X_n]/\mathcal{I}$. Nous noterons $q_h[i,j]$ l'élément situé à l'intersection de la i^e colonne et de la j^e ligne de q_h, $q_h[i]$ la i^e ligne de q_h.*
Pour tout polynôme $h \in K[X_1,\ldots,X_n]$, on définit :

- *le vecteur $\overrightarrow{h}$ de K^n dont les coordonnées sont celles de la classe de h dans $K[X_1,\ldots,X_n]/\mathcal{I}$, exprimées dans la base $\mathcal{B}$.*
- *le vecteur de K^n : $Vtr(h) = [Trace(m_{h\omega_1}), \ldots, Trace(m_{h\omega_D})]$*

Lemme 1.2.3 *Pour tout couple de polynômes (f,g) de $K[X_1,\ldots,X_n]$,*

$$Trace(m_{fg}) = \overrightarrow{f} \cdot Vtr(g)$$

Preuve : En exprimant fg dans la base $\mathcal{B}$ et en posant $\overrightarrow{f} = \sum_{i=1}^{D} a_i\omega_i$, il vient :

$$Trace(m_{fg}) = Trace(m_{\sum_{i=1}^{D} a_i\omega_i g}) = \sum_{i=1}^{D} a_i Trace(m_{g\omega_i})$$

et donc,

$$Trace(m_{fg}) = [a_1, \ldots, a_D] \cdot [Trace(m_{h\omega_1}), \ldots, Trace(m_{h\omega_D})] = \overrightarrow{f} \cdot Vtr(g)$$

$\square$

Ceci nous donne alors une propriété intéressante sur la forme quadratique de Hermite généralisée :

Proposition 1.2.1 *Q_h est nulle si et seulement si $q_h[1]$ est nulle.*

Preuve : Pour tout $i > 1$,

$$q_h[i] = [Trace(m_{\omega_i\omega_1}), \ldots, Trace(m_{\omega_i\omega_D})].$$

Par conséquent, d'après le lemme 1.2.3,

$$q_h[i,j] = \overrightarrow{\omega_i\omega_j} \cdot Vtr(1) = \overrightarrow{\omega_i\omega_j} \cdot q_h[1]$$

$\square$

Nous retrouvons ainsi un critère d'appartenance d'un polynôme à $\sqrt{\mathcal{I}}$, ne nécessitant pas le calcul explicite de $\sqrt{\mathcal{I}}$ (voir [Ron90], [GMT88],[GV92]) , ce qui présentera plus tard un avantage algorithmique certain :

Lemme 1.2.4 *(Lemme de Dickson) Soient $\mathcal{I} \subset K[X_1, \ldots, X_n]$ un idéal zéro-dimensionnel, et $\{\omega_1, \ldots, \omega_D\}$ une base de l'espace vectoriel de dimension finie $K[X_1, \ldots, X_n]/\mathcal{I}$. Un polynôme $h \in K[X_1, \ldots, X_n]$ appartient à $\sqrt{\mathcal{I}}$ si et seulement si*

$$Trace(m_{h\omega_i}) = 0 \ , \ \forall i = 1, \ldots, D.$$

Chapitre 2

Calculs dans le quotient

Résumé

Comme annoncé dans la partie préliminaire, nous allons utiliser un certain nombre de techniques d'algèbre linéaire. Il va sans dire que le comportement de nos algorithmes sera conditionné par la qualité de la représentation choisie pour $K[X_1, \ldots, X_n]/\mathcal{I}$.

Un outil incontournable à l'heure actuelle, pour effectuer des calculs modulo un idéal $\mathcal{I} \subset K[X_1, \ldots, X_n]$ (zéro-dimensionnel ou pas), est l'algorithme de Buchberger. Il permet de calculer une famille agréable de générateurs de $\mathcal{I}$ (base de Groebner), dont les propriétés sont très utiles pour travailler dans $K[X_1, \ldots, X_n]/\mathcal{I}$ (par exemple, il existe un algorithme pour déterminer, de manière unique, la classe d'un polynôme modulo $\mathcal{I}$). Sur ce sujet, nous ne ferons que rappeler les résultats désormais standards. En fait, étant donné un idéal zéro-dimensionnel $\mathcal{I} \subset K[X_1, \ldots, X_n]$, notre but est de calculer la *table de multiplication* de $K[X_1, \ldots, X_n]/\mathcal{I}$. C'est à dire :

- Construire une base $\mathcal{B} = \{\omega_1, \ldots, \omega_D\}$ du K-espace vectoriel $K[X_1, \ldots, X_n]/\mathcal{I}$
- Donner l'ensemble des produits réduits $\overrightarrow{\omega_i\omega_j}$ où $\overrightarrow{\omega_i\omega_j}$ est le vecteur dont les coordonnées sont celles de la classe de $\omega_i\omega_j$ dans $K[X_1, \ldots, X_n]/\mathcal{I}$, exprimées dans la base $\mathcal{B}$.

La motivation initiale de ce travail était de fournir un outil efficace pour construire les matrices de formes quadratiques particulières nécessaires à pour l'un de nos algorithmes (méthode de Hermite).

Nous avons rapidement remarqué qu'une stratégie naïve ne permettait pas de calculer efficacement ces formes quadratiques (nécessitant la connaissance des expressions $\overrightarrow{\omega_i\omega_j}$) pour deux raisons :

- Le coût de l'opération *réduction d'un polynôme* modulo $\mathcal{I}$ via l'algorithme NF (*Normal Form* décrit dans les rappels sur les bases de Groebner) est important.
- Beaucoup de calculs redondants sont effectués.

Dans un premier temps, nous avons utilisé un procédé informatique général pour éviter les redondances : une table de *Hachage*. Cette solution nous ayant permis de réduire de façon considérable le nombre de calculs.
Toutefois, des calculs inutiles apparaissent, même si l'on utilise un tel procédé. Ils sont intrinsèques à l'utilisation de la fonction NF.

Nous montrons, dans cette partie, comment combiner quelques propriétés algébriques et un procédé informatique pour calculer l'ensemble des réductions $\overrightarrow{\omega_i\omega_j}$ (formant ce que nous appelons une table de multiplication de $K[X_1,\ldots,X_n]/\mathcal{I}$), pour optimiser le nombre et la complexité des calculs, l'idée principale étant de ne pas dissocier le procédé informatique du problème mathématique.

2.1 Rappels sur les bases de Groebner

Dans cette partie, K désigne un corps de caractéristique 0.

L'objet de cette section est de résumer quelques propriétés élémentaires sur les bases de Groebner. Une description simple et complète peut être trouvée dans [LCO], contenant, en particulier, les démonstrations des principaux théorèmes énoncés dans cette partie.

Les bases de Groebner constituent pour l'heure un outil incontournable pour l'étude des systèmes polynomiaux. Nous ne donnons ici que les éléments nécessaires à la bonne compréhension de cette thèse.

La réduction d'un polynôme modulo un idéal $\mathcal{I} \subset K[X_1, \ldots, X_n]$ n'est pas une opération évidente. Une base de Groebner est un système de générateurs particulier permettant de généraliser l'algorithme de division euclidienne pour l'appliquer au cas des polynômes à plusieurs variables.

La notion de degré usuelle ne suffit plus pour pour ordonner les monômes de unitaires de $K[X_1, \ldots, X_n]$. En effet, celle ci ne permet pas de comparer les monômes ayant un même degré total. Par exemple: xy^2 et x^2y .
En considérant la bijection:

$$\begin{array}{c} \mathcal{M}[X_1, \ldots, X_n] \longrightarrow \mathbb{N}^n \\ X^\alpha = X_1^{\alpha_1}, \ldots, X_n^{\alpha_n} \longmapsto \alpha \end{array}$$

où $\mathcal{M}[X_1, \ldots, X_n]$ est l'ensemble des monômes unitaires de $K[X_1, \ldots, X_n]$, on remarque qu'ordonner l'ensemble des éléments x^α de $\mathcal{M}[X_1, \ldots, X_n]$ revient exactement à ordonner les éléments de $\mathbb{N}^n$. Nous avons alors la définition suivante:

Definition 2.1.1 Un ordre admissible sur les monômes *(unitaires) de $K[X_1, \ldots, X_n]$ est une relation " $>$ " binaire sur $\mathbb{N}^n$ telle que:*

A. $>$ est une relation d'ordre total sur $\mathbb{N}^n$.

B. si $\alpha > \beta$ et $\gamma \in \mathbb{N}^n$, alors $\alpha + \beta > \beta + \gamma$.

C. pour l'ordre $>$, tout ensemble non vide admet un plus petit élément. sur $\mathbb{N}^n$.

Nous donnons maintenant la définition des ordres admissibles les plus couramment utilisés.

Definition 2.1.2 (Ordre Lexicographique)
Soient $\alpha = (\alpha_1, \cdots, \alpha_n)$ et $\beta = (\beta_1, \cdots, \beta_n) \in \mathbb{N}^n$, on définit l'ordre lexicographique :

$$\alpha >_{lex} \beta \;\Leftrightarrow \exists i \in [1, \cdots, n] \;, \left\{ \begin{array}{l} \alpha_j - \beta_j = 0 \quad \forall j \in [1 \cdots i[\\ \alpha_i > \beta_i \end{array} \right.$$

Exemple : $(1,2,3) >_{lex} (1,1,4)$ car $\alpha - \beta = (0,1,-3)$

Definition 2.1.3 (Ordre du Degré lexicographique).
Soient $\alpha, \beta \in \mathbb{N}^n$. On dit que α est plus grand que β pour l'ordre du degré lexicographique, et on note $\alpha >_{dlex} \beta$, si:

$$(|\alpha| = \textstyle\sum_{i=1}^n \alpha_i > |\beta| = \sum_{i=1}^n \beta_i) \text{ , } ou\ si : (|\alpha| = |\beta| \ et\ \alpha >_{lex} \beta)$$

Exemple : $(1,2,4) >_{dlex} (1,1,5)$.

Definition 2.1.4 (Ordre du Degré Lexicographique Inverse) .
Soient α , $\beta \in \mathbb{N}^n$. On dit que α est plus grand que β pour l'ordre du degré lexicographique inverse, et on note $\alpha >_{drl} \beta$, si:

$$|\alpha| = \sum_{i=1}^n \alpha_i > |\beta| = \sum_{i=1}^n \beta_i$$

ou si :

$$\begin{cases} |\alpha| = |\beta| \\ \exists \mathcal{I} \ , \ i \in [1 \cdots n] & \begin{cases} \alpha_j = \beta_j \ \forall j \ , \ j \in]i \cdots n] \\ \alpha_i < \beta_i \end{cases} \end{cases}$$

Exemples :

- Comme $|(4,7,1)| = 12$ et $|(4,2,3)| = 9$, alors

$$(4,7,1) >_{drl} (4,2,3)$$

- Comme $(1,5,2,2) - (4,1,3,2) = (-3,4,-1,0)$, alors

$$(1,5,2,2) >_{drl} (4,1,3,2)$$

Il est maintenant possible de donner la série de définitions suivante:

Definition 2.1.5 *Soient $p = \sum_\alpha c_\alpha x^\alpha \in K[X_1, \ldots, X_n]$ et $>$ un ordre sur les monômes de $K[\tilde{x}]$.*

A. Le **multi-degré** *de p est :*

$$multideg(p) = max\{\alpha \in \mathbb{N}^n | c_\alpha \neq 0\}$$

B. Le **coefficient de plus haut degré** *de p est :*

$$lc(p) = c_{multideg(p)} \in K$$

C. Le **monôme de plus haut degré** *de p est :*

$$lm(p) = x^{multideg(p)}$$

D. Le **terme initial** *de p est :*

$$in(p) = lc(p) \times lm(p)$$

Exemple :

Soit $p = 4xy^2z + 4z^2 - 5x^3 + 7x^2z^2$, on a alors pour l'ordre *lex* :

$$multideg(p) = (3, 0, 0)$$

$$lc(p) = -5$$
$$lm(p) = x^3$$
$$in(p) = -5x^3$$

Pour toute la suite de cette section, on suppose fixé un ordre admissible sur les monômes de $K[X_1, \ldots, X_n]$.

Théorème 2.1.1 (Réduction modulo une famille de polynômes). *On suppose que l'on s'est fixé un ordre admissible sur les monômes unitaires de $K[X_1, \ldots, X_n]$. Si $F = \{f_1, \cdots, f_s\}$ est une famille de polynômes de $K[X_1, \cdots, X_n]$, alors tout $f \in K[X_1, \cdots, X_n]$ peut s'écrire :*

$$f = a_1 f_1 + \cdots + a_s f_s + r$$

où : $\forall i$, $i \in [1 \cdots s]$, $a_i \in K[X_1, \cdots, X_n]$ et $r \in K[X_1, \cdots, X_n]$ tel que aucun de ses monômes de r ne soit divisible par l'un des $in(f_i)$, $i \in [1 \cdots s]$.

En fait, la preuve de ce théorème peut être établie en exhibant un algorithme calculant le reste d'un polynôme modulo une liste ordonnée.

Algorithme de réduction :

ALGORITHME NF(f,f_1,$\cdots$,f_s)
  $a_1 = 0$; $\cdots$; $a_s = 0$; $r = 0$
  $q = f$
  TANT QUE $q \neq 0$ FAIRE
    $i := 1$
    TermeEnlevé:=FAUX

```
      TANT QUE i ≤ s ET NON(TermeEnlevé) FAIRE
        SI in(f_i) divise in(q) ALORS
          a_i := a_i + in(q)/in(f_i)
          q := q − (in(q)/in(f_i))f_i
          TermeEnlevé:= VRAI
        SINON
          i := i + 1
        FIN SI
      FIN TANT QUE
      SI NON(TermeEnlevé) ALORS
        r := r + in(q)
        q := q − in(q)
      FIN SI
    FIN TANT QUE
    RETOURNER(a_1, ···, a_s, r)
  FIN ALGORITHME
```

Cet algorithme est un peu naif et a fait l'objet de nombreuses optimisations (voir [Fau94]), mais toutefois, le principe de base reste d'actualité.
Si $f, f_1, \cdots, f_s$ des polynômes de $K[X_1, \ldots, X_n]$, l'expression de f donnée par le théorème 2.1.1 (calculée par l'algorithme **NF**) dépend de l'ordre dans lequel les polynômes $f_1, \cdots, f_s$ interviennent dans l'algorithme de réduction.
Exemple : Posons :

$$f_1 = x^3 - xy \ , \ f_2 = x - y^4 \ , \ f = x^5y^2,$$

et supposons que l'ordre choisi soit l'ordre *lex*. Nous avons alors :

$$f = (x^2y^2 + y^3)f_1 + y^4 f_2 + y^8$$

$$f = (x^4y^2 + x^3y^6 + x^2y^10 + xy^14 + y^18)f_2 + 0f_1 + y^2,$$

ces deux expressions ayant été respectivement obtenues en utilisant l'algorithme de réduction avec les familles ordonnées (f_1, f_2) et (f_2, f_1).
Plus préocupant, cet algorithme ne permet a priori pas de décider si un polynôme appartient ou non à un idéal donné.

En effet, si $\mathcal{I}$ est un idéal de $K[X_1, \ldots, X_n]$ engendré par les polynômes $f_1, \cdots, f_s$ et f un elément de $K[X_1, \ldots, X_n]$, le fait que la réduction de f modulo $f_1, \cdots, f_n$ (pris dans cet ordre) ne soit pas nul ne constitue pas une condition suffisante pour que f ne soit pas élément de $\mathcal{I}$.
Exemple : Posons : $f_1 = x + 1$, $f_2 = y^2 - 1$, $f = xy^2 - x$, ordonnés selon l'ordre lexicographique.

Si nous appliquons l'algorithme de réduction pour $F = (f_1, f_2)$, nous obtenons la décomposition suivante:

$$f = yf_1 + 0 \cdot f_2 + (-x - y),$$

alors que pour $F = (f_2, f_1)$ on obtient :

$$f = xf_2$$

et donc $f \in I = \langle f_1, f_2 \rangle$.
Ces diverses remarques ont amené l'étude de systèmes de générateurs particuliers : les bases de Groebner, dont une définition peut être donnée par :

Definition 2.1.6 *Un famille génératrice finie $G = (g_1, \cdots, g_t)$ d'éléments d'un idéal $\mathcal{I}$ de $K[X_1, \ldots, X_n]$ est une* **base de Groebner** *si :*

$$\langle in(g_1), \cdots, in(g_t) \rangle = \langle in(\mathcal{I}) \rangle$$

La principale propriété des bases de Groebner peut être résumée par :

Proposition 2.1.1 *Soit $G = (g_1, \cdots, g_t)$ une base de Groebner d'un idéal $\mathcal{I} \subset K[X_1, \ldots, X_n]$. Pour tout $f \in K[X_1, \ldots, X_n]$, le reste de f modulo G est déterminé de manière unique (par l'algorithme NF), indépendemment de l'ordre dans lequel les polynômes $g_i, i = 1 \ldots t$ sont utilisés. En particulier, f est dans l'idéal $\mathcal{I}$ si et seulement si $NF(f, \mathcal{G}) = 0$. Dans ce cas le résultat de la fonction $NF(f, \mathcal{G})$ est appelé forme normale de f modulo $\mathcal{G}$.*

Buchberger (voir [Buch85]) a proposé un algorithme qui depuis à été dérivé en de nombreux clones (voir, par exemple, [Tra88],[Fau94]) basé sur la propriété suivante :

Théorème 2.1.2 *Soient $f, g \in K[X_1, \ldots, X_n]$ et x^γ le plus petit commun multiple de $lm(f)$ et $lm(g)$. Le* **S-polynôme** *de f et g est défini par:*

$$S(f, g) = \frac{x^\gamma}{in(f)}.f - \frac{x^\gamma}{in(g)}.g$$

Soit $\mathcal{I}$ un idéal de $K[X_1, \ldots, X_n]$. Une famille génératrice $G = \{g_1, \cdots, g_s\} \subset K[X_1, \ldots, X_n]$ sera une base de Groebner de $\mathcal{I}$, si et seulement si, pour tout couple d'entiers (i, j) tel que $i \neq j$, $NF(S(g_i, g_j), G) = 0$.

L'algorithme de Buchberger consiste donc à rajouter des S-polynômes à la famille de générateurs de $\mathcal{I}$, jusqu'à ce que tous les S-polynômes se réduisent à zéro. Evidement, si aucune précaution n'est prise, ce procédé construit une famille de générateurs dont le nombre peut être extrêmement important. Cette famille peut cependant être simplifiée en ce qui est appelé une base de Groebner Réduite :

Definition 2.1.7 *Une base de Groebner réduite pour un idéal $\mathcal{I}$ de $K[X_1, \ldots, X_n]$ est une base de Groebner pour $\mathcal{I}$ dont les polynômes sont constitués de monômes irréductibles modulo $\mathcal{I}$.*

L'algorithme de Buchberger est de complexité au moins doublement exponnentielle dans les meilleurs cas mais s'avère toutefois efficace en pratique grace à de nombreuses optimisations. En particulier, dans le cas des systèmes à coefficients entiers, une prédiction modulaire en parallèle (voir [Tra88],[Fau94]) permet d'éviter de nombreux calculs de S-polnômes inutiles (c'est à dire n'amenant aucun nouveau générateur). D'autre par, lorsque K est un anneau intègre, en multipliant les S-polynômes par des scalaires biens choisis, il est possible d'effectuer tous les calculs dans $K[X_1, \ldots, X_n]$ et, en particulier, de fournir un système de générateurs dans $K[X_1, \ldots, X_n]$.

Les bases de Groebner calculées pour l'ordre lexicographique donnent une bonne représentation de solutions d'un système zéro-dimensionnel, l'un des polynômes de la base étant dans ce cas en une seule variable. Le calcul d'une telle base par l'algorithme de Buchberger devenant très vite insurmontable ([Fau94]), il est, en général, effectué en deux étapes dans le cas zéro-dimensionnel :

- Calcul d'une base de Groebner pour un ordre du degré (par exemple $D.R.L$) par l'algorithme de Buchberger.
- Calcul de la base de Groebner lexicographique par changement d'ordre via des techniques d'algèbre linéaire : l' algorithme *FGLM* (voir [FGLM94]) par exemple, utilise la structure d'espace vectoriel de $K[X_1, \ldots, X_n]/\mathcal{I}$.

2.2 La table de multiplication

Dans toute cette partie, K désigne un corps de caractéristique 0.
Nous avons vu, dans le premier chapitre, comment ramener certains problèmes liés à l'étude des systèmes zéro-dimensionnels à des problèmes d'algèbre linéaire simples (par exemple le comptage du nombre de zéros réels ou encore l'appartenance d'un polynôme au radical d'un idéal). Dans le chapitre précédent, nous avons rappelé un outil fondamental pour le calcul modulo un idéal de polynômes, il s'agit maintenant de rendre possible, d'un point de vue pratique, l'utilisation de ces divers objets.

Étant donné un idéal zéro-dimensionnel $\mathcal{I} \subset K[X_1, \ldots, X_n]$, notre but est de construire la *table de multiplication* de $K[X_1, \ldots, X_n]/\mathcal{I}$. C'est à dire :

- Construire une base $\mathcal{B} = \{\omega_1, \ldots, \omega_D\}$ du K-espace vectoriel $K[X_1, \ldots, X_n]/\mathcal{I}$
- Donner l'ensemble des produits réduits $\overrightarrow{\omega_i\omega_j}$ où $\overrightarrow{\omega_i\omega_j}$ est le vecteur dont les coordonnées sont celles de la classe de $\omega_i\omega_j$ dans $K[X_1, \ldots, X_n]/\mathcal{I}$, exprimées dans la base $\mathcal{B}$.

La motivation initiale de ce travail était de fournir un outil efficace pour construire la matrice, dans $\mathcal{B}$, de la forme quadratique de Hermite généralisée associée à un polynôme de $K[X_1, \dots X_n]$. Par exemple :

$$q_1[i,j] = Trace(m_{\omega_i \omega_j}) = \sum_{k=1}^{D} (\omega_i \overrightarrow{\omega_j} \omega_k)_k$$

(où $(\omega_i \overrightarrow{\omega_j} \omega_k)_k$ est la k^e coordonnée de $\omega_i \overrightarrow{\omega_j} \omega_k$). Nous avons rapidement remarqué qu'une stratégie naïve ne permettait pas de construire efficacement ces formes quadratiques pour deux raisons :

- Le coût de l'opération *réduction d'un polynôme* modulo $\mathcal{I}$ via l'algorithme NF est important.
- Beaucoup de calculs redondants sont effectués.

Prenons un exemple simple :
Soit $\mathcal{I} = \langle X_1^3 - 2, X_2^3 - 3 \rangle \subset K[X_1, \dots X_2]$. Une base de $K[X_1, \dots X_2]/\mathcal{I}$ est alors donnée par :

$$\mathcal{B} = \left\{ \begin{array}{l} \omega_1 = 1, \omega_2 = X_1, \omega_3 = X_1^2, \omega_4 = X_2, \omega_5 = X_2^2, \\ \omega_6 = X_1 X_2, \omega_7 = X_1^2 X_2, \omega_8 = X_1 X_2^2, \omega_9 = X_1^2 X_2^2 \end{array} \right\}$$

D'après le lemme 1.2.3, la connaissance des produits $\overrightarrow{\omega_i \omega_j}$ est suffisante pour construire de q_1 (i.e. $q_1[i,j] = \overrightarrow{\omega_i \omega_j} \cdot [Trace(m_{\omega_1}), \dots, Trace(\omega_D)]$). Dans le cas présent, les monômes à réduire pour obtenir la table de multiplication de $K[X_1, \dots X_2]/\mathcal{I}$ sont représentés par le tableau :

$\times$	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9
ω_1	1	X_1	X_1^2	X_2	X_2^2	X_1X_2	$X_1^2X_2$	$X_1X_2^2$	$X_1^2X_2^2$
ω_2	X_1	X_1^2	X_1^3	X_1X_2	$X_1X_2^2$	$X_1^2X_2$	$X_1^3X_2$	$X_1^2X_2^2$	$X_1^3X_2^2$
ω_3	X_1^2	X_1^3	X_1^4	$X_1^2X_2$	$X_1^2X_2^2$	$X_1^3X_2$	$X_1^4X_2$	$X_1^3X_2^2$	$X_1^4X_2^2$
ω_4	X_2	X_1X_2	$X_1^2X_2$	X_2^2	X_2^3	$X_1X_2^2$	$X_1^2X_2^2$	$X_1X_2^3$	$X_1^2X_2^3$
ω_5	X_2^2	$X_1X_2^2$	$X_1^2X_2^2$	X_2^3	X_2^4	$X_1X_2^3$	$X_1^2X_2^3$	$X_1X_2^4$	$X_1^2X_2^4$
ω_6	X_1X_2	$X_1^2X_2$	$X_1^3X_2$	$X_1X_2^2$	$X_1X_2^3$	$X_1^2X_2^2$	$X_1^3X_2^2$	$X_1^2X_2^3$	$X_1^3X_2^3$
ω_7	$X_1^2X_2$	$X_1^3X_2$	$X_1^4X_2$	$X_1^2X_2^2$	$X_1^2X_2^3$	$X_1^3X_2^2$	$X_1^4X_2^2$	$X_1^3X_2^3$	$X_1^4X_2^3$
ω_8	$X_1X_2^2$	$X_1^2X_2^2$	$X_1^3X_2^2$	$X_1X_2^3$	$X_1X_2^4$	$X_1^2X_2^3$	$X_1^3X_2^3$	$X_1^2X_2^4$	$X_1^3X_2^4$
ω_9	$X_1^2X_2^2$	$X_1^3X_2^2$	$X_1^4X_2^2$	$X_1^2X_2^3$	$X_1^2X_2^4$	$X_1^3X_2^3$	$X_1^4X_2^3$	$X_1^3X_2^4$	$X_1^4X_2^4$

Ce tableau contient 81 monômes de la forme $\omega_i \omega_j$. En ne tenant compte que de la commutativité dans $K[X_1, \dots, X_n]$, nous en dénombrons encore 41 alors que le nombre d'éléments distincts est 25.

Dans un premier temps, nous avons utilisé un procédé informatique général pour éviter les redondances : une table de *Hachage*. Cette solution nous ayant permis de réduire de façon considérable le nombre de calculs.
Toutefois, des calculs inutiles apparaissent, même si l'on utilise un tel procédé. Ils sont intrinsèques à l'utilisation de la fonction NF décrite au chapitre précédent. En reprenant notre exemple, supposons que nous désirions calculer $\overrightarrow{\omega_7\omega_8} = \overrightarrow{X_1^3X_2^3}$. Deux réductions seront nécessaires pour obtenir la valeur de $X_1^3X_2^3 \; mod \, \mathcal{I}$:

$$NF(X_1^3X_2^3) = NF(3X_1^3) = NF(6) = 6$$

Par contre si, avant ce calcul, $NF(X_1^2X_2^3) = 3X_1^2$ est connu, en écrivant

$$NF(X_1^3X_2^3) = NF(NF(X_1)NF(X_1^2X_2^3)) = NF(3X_1^3) = 6,$$

une seule réduction est nécessaire.

Nous montrons dans cette partie comment combiner quelques propriétés algébriques et un procédé informatique pour calculer l'ensemble des réductions $\overrightarrow{\omega_i\omega_j}$ (formant ce que nous appelons une table de multiplication de $K[X_1,\ldots,X_n]/\mathcal{I}$), pour optimiser le nombre et la difficulté des calculs, l'idée principale étant de **ne pas dissocier le procédé informatique du problème mathématique**.

2.2.1 Principes de base

En nous appuyant sur les résultats de la partie *Bases de Groebner* , nous déterminons, pour commencer, une base de monômes $\mathcal{B}$ de $K[X_1,\ldots,X_n]$.

Definition 2.2.1 *Soient $\mathcal{I} \subset K[X_1,\ldots X_n]$ un idéal zéro-dimensionnel et $\mathcal{G} = \{g_1,\ldots,g_u\}$ une base de Groebner de $\mathcal{I}$, calculée pour un ordre admissible (quelconque) sur les monômes de $K[X_1,\ldots X_n]$. On définit :*

$$\mathcal{B}^C = \{m \in \mathcal{M}[X_1,\ldots,X_n] \mid \exists g \in \mathcal{G} \;, m' \in \mathcal{M}[X_1,\ldots,X_n] \;, m = lm(g_i)m'\}$$

où $\mathcal{M}[X_1,\ldots,X_n]$ est l'ensemble des monômes unitaires de $K[X_1,\ldots X_n]$ et $\mathcal{B}^C$ le complémentaire de $\mathcal{B}$ dans $\mathcal{M}[X_1,\ldots,X_n]$.

Lemme 2.2.1 *$\mathcal{B}$ est une base du K-espace vectoriel $\mathcal{A} = K[X_1,\ldots,X_n]/\mathcal{I}$.*

Preuve : Pour tout $m \in \mathcal{B}$, m n'est divisible par aucun des $lm(g)$, $g \in \mathcal{G}$. Ainsi, $NF(m,\mathcal{G}) = m$, et donc les éléments de $\mathcal{B}$ sont irréductibles modulo $\mathcal{I}$.

Soient $m_1,\ldots m_D \in \mathcal{B}$ et $a_1,\ldots,a_D \in K$ tels que $\sum_{i=1}^D a_im_i = 0$ modulo $\mathcal{I}$. Comme $\sum_{i=1}^D a_im_i$ est irréductible modulo $\mathcal{I}$, nous avons alors $a_i = 0$, $\forall i = 1\ldots D$, ce qui montre que $\mathcal{B}$ est une famille libre de $K[X_1,\ldots,X_n]/\mathcal{I}$.

Supposons maintenant que $p \in K[X_1,\ldots X_n]$ soit irréductible modulo $\mathcal{I}$. Il existe alors un scalaire $a_1 \in K$ est un monôme $m_1 \in \mathcal{B}$ tels que $in(p) = a_1 \cdot m_1$. En utilisant récursivement ce procédé, on peut trouver des scalaires $a_1,\ldots,a_D \in K$ et des monômes $m_1,\ldots,m_D \in \mathcal{B}$ tels que $NF(p,\mathcal{G}) = \sum_{i=1}^D a_im_i$. □

Remarque 2.2.1 *La définition 2.2.1 et le lemme 2.2.1 donnent un algorithme simple pour décider si un idéal est zéro-dimensionnel ou pas, les affirmations suivantes étant équivalentes :*

- *Pour tout* $i = 1 \dots n$, $\exists g \in \mathcal{G}$ *et* $a_i \in \mathbb{N}$ *tels que* $lm(g) = X_i^{a_i}$
- $\mathcal{B}$ *est un ensemble fini de monômes.*

La propriété suivante exprime une forme de connexité de $\mathcal{B}$ et sera à la base de l'algorithme :

Lemme 2.2.2 *Soit* $m \in \mathcal{B}$. *Si,* $m \neq 1$, $\exists k \in [1, \dots n]$ *et* $m' \in \mathcal{B}$ *tels que* $m = X_k m'$.

Preuve Soient $m \neq 1 \in \mathcal{B}$ et $\alpha_1, \dots \alpha_n$ des entiers positifs tels que $m = X_1^{\alpha_1} \dots X_n^{\alpha_n}$. Comme $m \neq 1$, il existe k tel que $\alpha_k \neq 0$ et donc $m' = \frac{m}{X_k} \in \mathcal{M}[X_1, \dots, X_n]$. Comme m n'est divisible par aucun élément de l'ensemble $\{lm(g) \ , \ g \in \mathcal{G}\}$, a fortiori, il en est de même pour m' et donc $m' \in \mathcal{B}$. □

Cette propriété se généralise alors facilement à l'ensemble des éléments de la table de multiplication :

$$MT(\mathcal{B}) = \{\omega_i \omega_j \ , \ \omega_i, \omega_j \in \mathcal{B}\}$$

Lemme 2.2.3 *Soit* $m \in MT(\mathcal{B})$, $m \neq 1$. *Il existe un monôme* $m' \in MT(\mathcal{B})$ *et un entier* $k \in [1, \dots, n]$ *tels que :* $m = X_k m'$

Preuve Soit $m \in MT(\mathcal{B})$, $m = \omega_i \omega_j$. Comme $m \neq 1$, on peut supposer que $\omega_i \neq 1$. D'après le lemme 2.2.2, il existe X_k et ω_l tels que $\omega_i = X_k \omega_l$. Ainsi, $m = X_k \omega_l \omega_j = X_k m'$ ($m' \in MT(\mathcal{B})$). □

Les lemmes 2.2.2 et 2.2.3 montrent que l'ensemble des monômes à calculer peut être représenté par un arbre $\mathcal{A}$ tel que :

- la racine de $\mathcal{A}$ soit 1.
- un noeud de $\mathcal{A}$ est la donnée d'un monôme m dont les descendants sont les $\{X_i m \ , \ i = 1 \dots n\}$.

Notre but est alors de construire et d'élaguer $\mathcal{A}$, de sorte qu'un monôme n'y apparaisse qu'une seule fois, et de transformer $\mathcal{A}$ en arbre de calcul. Cette dernière opération devient simple dès lors que nous connaissons un moyen de multiplier par une variable dans $K[X_1, \dots, X_n]/\mathcal{I}$ puisque si m est un noeud de $\mathcal{A}$, ses descendants sont de la forme $\{X_i m \ , \ i = 1 \dots n\}$.
On peut dors et déja remarquer que l'ensemble des monômes de la forme :

$$\{X_i \omega_j \ , \ i = 1 \dots n \ , \ j = 1 \dots D\}$$

joue un role particulier, puisque qu'il permet de définir les matrices de multiplication par chacune des variables. Il devra être calculé en priorité.

Definition 2.2.2 *Soient $I \subset K[X_1, \ldots X_n]$ un idéal zéro-dimensionnel et $\mathcal{B}$ une base de monômes du K-espace vectoriel A, on note $Border(\mathcal{B})$ le sous - ensemble de $\mathcal{M}[X_1, \ldots, X_n]$ défini par :*

$$Border(\mathcal{B}) = \{m \in \mathcal{B}^C \ , \exists k, i \ , \ \ m = X_k \omega_i\}$$

Remarque 2.2.2 *Si $\mathcal{G}$ est une base de Groebner réduite de $\mathcal{I}$, alors, pour tout $g \in \mathcal{G}$, $lm(g) \in Border(\mathcal{B})$. En effet, $lm(g) = X_1^{e_1} \ldots X_n^{e_n}$ où les e_i sont non tous nuls. En supposant $e_k \neq 0$, $m = lm(g)/X_k$ est en effet un élément de $\mathcal{B}$ car divisible par aucun des termes de tête des éléments de l'ensemble $\{lm(g) \ , \ g \in \mathcal{G}\}$ ($\mathcal{G}$ est une base de Groebner réduite).*

D'après la remarque précédente, une partie de $Border(\mathcal{B})$ peut-être facilement déduite de la base de Groebner :

Definition 2.2.3 *Une partie de $Border(\mathcal{B})$ est constituée par les termes de tête des éléments de $\mathcal{G}$:*

$$lm(\mathcal{G}) = \{lm(g) \ , g \in \mathcal{G}\}$$

En supposant que $\mathcal{G}$ soit une base de Groebner réduite, pour tout $g \in \mathcal{G}$, $g - in(g)$ est alors une combinaison linéaire déléments de $\mathcal{B}$ et par conséquent nous pouvons supposer connues les expressions $\overrightarrow{lm(g)}$, $g \in \mathcal{G}$.

Lemme 2.2.4 *Soit $m \in Border(\mathcal{B})$,$m \notin lm(\mathcal{G})$. Il existe $m' \in Border(\mathcal{B})$ et $i \in [1, \ldots, n]$ tels que $m = X_i m'$*

Preuve : Soit $m \in Border(\mathcal{B})$. Comme $m \notin \mathcal{B}$, on peut trouver un polynôme $g \in \mathcal{G}$ et un monôme $m_1 \in \mathcal{M}[X_1, \ldots, X_n]$ tels que $m = m_1 \cdot lm(g)$. En utilisant la condition : $m \notin lm(\mathcal{G})$, on déduit alors que $m_1 \neq 1$ et par conséquent, il existe $i \in [1, \ldots, n]$ tel que $m_1 = X_i m_2$, $m_2 \in \mathcal{M}[X_1, \ldots, X_n]$. Ainsi, m peut être exprimé différement: $m = X_i m_2 lm(g_l)$. D'après la definition 2.2.2, $\exists j \in [1, \ldots, n]$ et $k \in [1, \ldots, D]$ tels que $m = X_j \omega_k$. Il vient alors, $X_i m_2 lm(g_l) = X_j \omega_k$ et $m_2 lm(g_l) = X_j \frac{\omega_k}{X_i}$. D'après le lemme 2.2.1, $m_2 lm(g_l) \notin \mathcal{B}$ et comme $m_2 lm(g_l) \in \mathcal{M}[X_1, \ldots, X_n]$, alors, $\frac{\omega_k}{X_i} \in \mathcal{B}$, $i \neq j$ et $m' = m_2 lm(g_l) \in Border(\mathcal{B})$. □

2.2.2 Modélisation

En tenant compte des diverses remarques effectuées jusqu'ici, la table de multiplication MT de $K[X_1, \ldots, X_n]/\mathcal{I}$ sera modélisée par :

- Une liste de monômes $\mathcal{B} = \{\omega_1, \ldots, \omega_D\}$
- Un arbre $\mathcal{A}$ représentant l'arbre de calcul de l'ensemble des éléments distincts de la forme $\overrightarrow{\omega_i \omega_j}$.

- Un tableau $tcalc$ destiné à recevoir les expressions réduites distinctes de la forme $\overrightarrow{\omega_i\omega_j}$ calculées lors du parcours de $\mathcal{A}$.
- Une matrice $refMult$ telle que $refMult[i,j]$ soit la position de $\overrightarrow{\omega_i\omega_j}$ dans $tcalc$.

D'après les lemmes 2.2.2 et 2.2.3, et par définition de $\mathcal{A}$, tout noeud $m \in \mathcal{A}$ admet pour descendants des monômes de la forme :

$$X_i m \ , \ i = 1 \ldots, n$$

Par conséquent un noeud m de $\mathcal{A}$ sera la donnée de :

- Un monôme $m.val$
- Trois entiers $m.pos$, $m.var$ et $m.prec$ tels que :
  - $tcalc[m.pos]$ contienne l'expression de $\overrightarrow{m.val}$
  - $m.prec$ soit la position dans $tcalc$ de la réduction d'un monôme m' tel que $m.val = X_{m.var}m'$.

Enfin, l'arbre $\mathcal{A}$ sera modélisé par une liste ordonnée de noeuds ne contenant pas d'éléments redondants. Ce type de représentation n'est pas des plus standard, mais il est toutefois adapté à notre problème. N'oublions pas, en effet, qu'un ordre admissible sur les monômes est compatible avec la multiplication. Ainsi, si N_1 et N_2 sont deux noeuds de profondeurs différentes situés sur une même branche, disons que N_2 est de profondeur supérieure à N_1, alors $N_1.val < N_2.val$, ce qui montre en particulier que tous les monômes apparaissant sur une branche de $\mathcal{A}$ sont parcourus par ordre croissant. La modélisation de $\mathcal{A}$ par une simple liste ne compliquera pas, par conséquent, le calcul de MT mais facilitera par contre la gestion des éléments redondants : l'arbre de calcul sera parfaitement élagué (i.e. aucune redondance) dès lors que les éléments $e_1, \ldots, e_k$ de la liste triée modélisant $\mathcal{A}$ vérifient la relation $e_i.val < e_{i+1}.val$, condition facile à maintenir puisque vérifiable par un simple test à chaque insertion d'un élément nouveau. De plus, nous avons vu que le procédé se déroulera en trois étapes (calcul de $\mathcal{B}$, calcul de $Border(\mathcal{B})$, reste de la table). La structure de liste ordonnée facilitera alors la fusion de sous arbres.
Cette modélisation est décrite par la figure 2.1.

2.2.3 L'algorithme

Il est clair que, si nous ordonnons les monômes à calculer par ordre croissant (en utilisant l'ordre pour lequel la base de Groebner à été calculée), les formes réduites peuvent se déduire par induction, par des combinaisons linéaires pour le calcul du bord et par des multiplications matrice/vecteur pour le reste de la table.

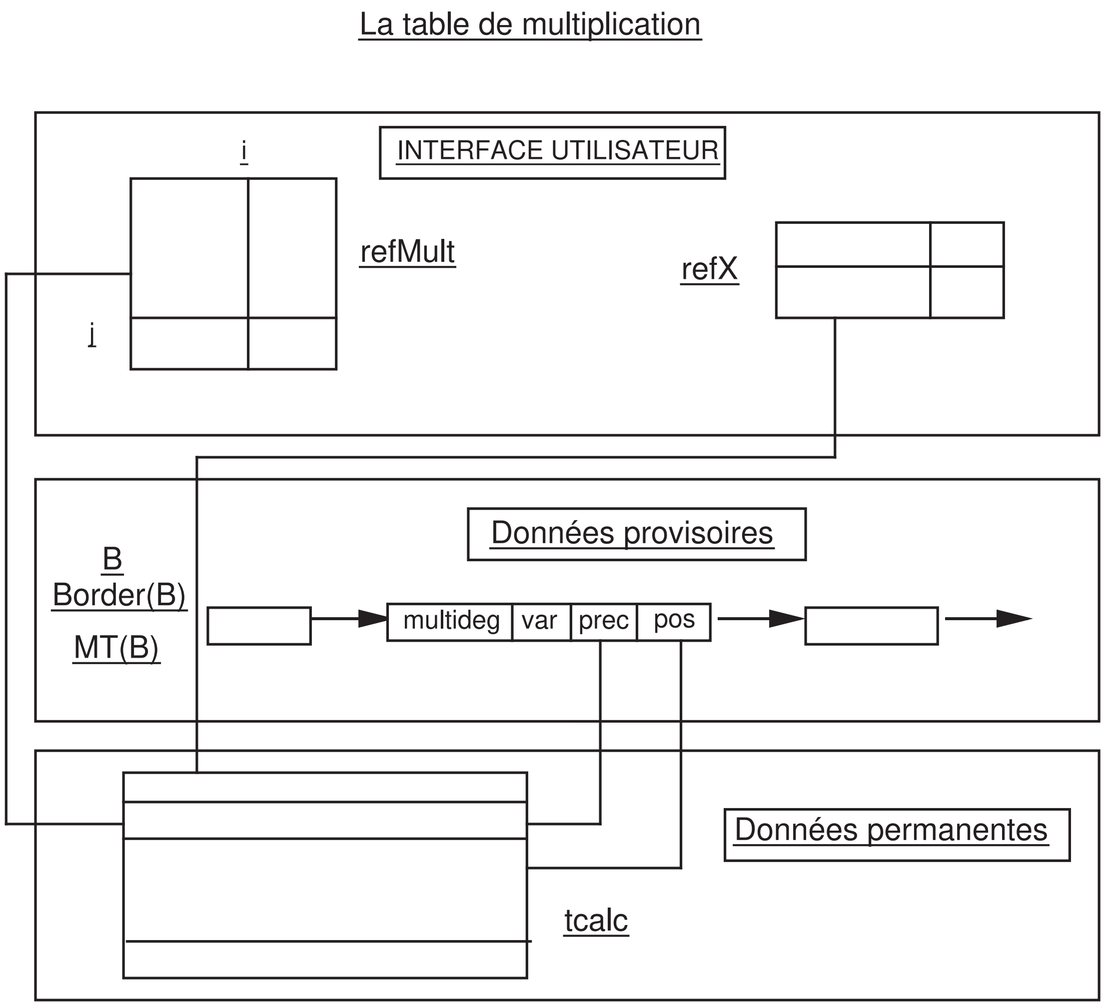

Figure 2.1: Table de Multiplication - Modélisation

Ainsi, la stratégie que nous utiliserons, consiste à calculer d'abord la liste des monômes à réduire, de la stocker dans une liste triée en définissant, pour chaque monôme, une variable X_k et un autre monôme m' tels que $m = X_k m'$. Puis, après avoir défini un moyen de multiplier par une quelconque variable, il nous suffira alors de parcourir cette liste pour calculer toutes les réductions que nous cherchons.

Comme nous avons établi trois relations principales dans la partie précédente, l'algorithme se découpe donc naturellement en trois phases :

- Le calcul de $\mathcal{B}$
- Le calcul de $Border(\mathcal{B})$
- Le calcul de $MT(\mathcal{B})$,

chaque phase se divisant elle - même en deux sous-parties :

- Une partie préparatoire qui consiste à calculer les monômes sans les réduire mais en affectant éventuellement les champs *val*, *pos* et *prec*.
- Une partie calcul qui consiste à calculer la forme réduite de chaque monôme (vecteur de A à coordonnées dans $\mathcal{B}$).

Pour optimiser l'implantation, nous commencerons par toute les parties *préparatoires* de manière à connaitre, par exemple, la taille du tableau *tcalc* avant d'y stocker quoi que ce soit.

Nous aurons besoin, par la suite, de fonctions élémentaires pour la manipulation des listes (arbres).

- Pour toute liste L, nous supposons qu'il existe un pointeur courant sur l'un de ses éléments. Lors de la création de L celui - ci pointe sur le premier terme de la liste. La fonction $Next(L)$ retourne l'élément suivant celui qui est couramment pointé (NULL si l'élément courrant est le dernier) et place le pointeur courant sur celui-ci.
- $InsertInList(L, el, st)$ insère le noeud *el* dans la liste ordonnée L pour l'ordre $>$ en suppposant que $el.val \geq st.val$.
  - retourne *True* si il existe déjà un élément $el' \in L$ tel que $el'.val = el.val$.
  - retourne *False* sinon
- Une premiere fonction d'insertion dans l'arbre modélisé par une liste L est : $InsertInListAndSet(L, el)$. Celle-ci permet d'insérer *el* dans la liste triée L, d'affecter les champs *el.var*, *el.prec* et *el.pos* et retourne un entier de la façon suivante :

  - Si il existe $m \in L$ tel que $m.val = el.val$, la fonction retourne $m.pos$.
  - Sinon, si il existe $i \in \mathbb{N}$ et $m \in L$ tels que $el.val = X_i \cdot m.val$, alors $el.prec := m.pos$, $el.var := i$ et la fonction insère el dans L puis retourne $\sharp L + 1$.

- Une deuxième fonction d'insertion dans l'arbre modélisé par une liste L est : $InsertInListAndSet(L, el, st)$. Celle ci permet d'insérer el dans la liste triée L sachant que $el.val > st.val$, mais cette fois en ne modifiant aucun champ de el. Nous imposons, de plus, que l'arbre modélisé par L avant l'appel soit un sous arbre de L après l'appel de cette fonction. Ainsi, la règle est la suivante:
  - Si il existe $m \in L$ tel que $el.val = m.val$ alors la fonction retourne $m.pos$.
  - Sinon, el est inséré dans la liste triée L et la fonction retourne $\sharp L + 1$.

Nous supposerons qu'il existe une fonction $Divisible(\mathcal{G}, m)$ qui retourne *True* si il existe $g \in \mathcal{G}$ tel que $lm(g)$ divise $m.val$ et une fonction $fusion(L_1, L_2)$ qui réalise la fusion de deux listes triées dont les éléments sont distincts.

Preparation de $\mathcal{B}$

L'algorithme suivant est une conséquence directe de la définition de $\mathcal{B}$:

Procédure Prepare-$\mathcal{B}$

- **Entrée :** $\mathcal{G}$
- **Sortie :** $\mathcal{B}$

```
Begin
    B := {1};
    m.val := 1;m.pos = 1;m.prec = 0;m.var = 0;
    /* Une variable pour affecter de nouvelles positions dans tcalc */
    FirstFree=2
    While not (m = NULL) do
        for i in 1..n do
            /* calcul des fils de m */
            tmp.val:=X_i * m.val
            tmp.prec:=m.pos
            tmp.pos:=FirstFree
            If not(Divisible(tmp,G)) then
                /* Insertion des fils de m dans B */
                If InsertIntList(B,tmp,m)
                    FirstFree=FirstFree+1
```

m=Next($\mathcal{B}$)
End.

Preuve de l'algorithme : Il est facile de remarquer que la liste retournée par Prepare-$\mathcal{B}$ est constituée par les éléments de l'ensemble :

$$\{m \in \mathcal{M}[X_1, \ldots, X_n]\ ,\ Divisible(m, \mathcal{G}) = False\}$$

ce qui coincide avec la définition de $\mathcal{B}$. □

Remarque 2.2.3 *Si m et m' sont deux éléments de $\mathcal{B}$, $m < m'$ n'implique pas forcément $m \cdot pos < m' \cdot pos$.*

Pour des raisons pratiques, nous choisirons la numérotation suivante pour les éléments de la base :

Notation 2.2.1 *$\omega_i = m.val$ si $m \cdot pos = i$.*

Préparation de $Border(\mathcal{B})$

Commençons par introduire une nouvelle table (matrice) $refX$, de dimension $n \times D$ telle que $refX[i,j]$ soit la position dans $tcalc$ de $\overrightarrow{X_i \omega_j}$ ($refX$ sera utilisée pour le calcul de $Border(\mathcal{B})$).

D'après ce qui précède, nous pouvons dors et déja initialiser quelques uns des éléments de $refX$:

$$for\ m\ in\ \mathcal{B}\ refX[m \cdot var, m \cdot prec] = m \cdot pos$$

En supposant les autres éléments de $refX$ initialisés à zéro, nous proposons l'algorithme suivant pour la *préparation* de $Border(\mathcal{B})$:

Procédure Prepare-$Border(\mathcal{B})$

- **Entrée :** $\mathcal{B}$ et $refX$
- **Sortie :** $\overline{\mathcal{B}} = \mathcal{B} \bigcup Border(\mathcal{B})$ et $refX$ (réactualisée).

Begin
  $Border(\mathcal{B}) := \{lm(g)\ g \in \mathcal{G}\}$;
  pos:= $\sharp\mathcal{B}$
  Sort($Border(\mathcal{B}), <$)
  for m in $\mathcal{B}$ do
    for j in 1..n do
      If refX[j,m.pos] =0 then /* $X_j\omega_{pos} \in Border(\mathcal{B})$ */
        refX[j,m.pos]:=pos+InsertInListAndSet(Border($\mathcal{B}$),$X_j \cdot m$)
  $\overline{\mathcal{B}}$ =fusion($\mathcal{B}$, Border($\mathcal{B}$))
End.

Preuve de l'algorithme : Pour prouver l'algorithme, il est suffisant de montrer que chaque fois que la fonction $InsertInListAndSet(Border(\mathcal{B}), X_j \cdot m)$ est appelée, soit il existe un élément de $m' \in Border(\mathcal{B})$ tel que $m = m'$, soit il existe X_k tel que $m = X_k \cdot m'$ (résultat donné par le lemme 2.2.4). □

Préparation de $MT(\mathcal{B})$

L'algorithme suivant est une conséquence directe du lemme 2.2.3 :

$\boxed{\textbf{Procédure Prepare-}MT(\mathcal{B})}$

- **Entrée :** $\mathcal{B}$ et $\overline{\mathcal{B}} = \mathcal{B} \bigcup Border(\mathcal{B})$ triées pour $<$.
- **Sortie :** L'arbre $MT(\mathcal{B})$ constitué des monômes de la table de multiplication et $refMult$.

$Begin$
  $MT(\mathcal{B}) = \overline{\mathcal{B}}$
  for $m_1 \in \mathcal{B}$ do
    fm=m_1
    for $m_2 \in \mathcal{B}$, $m_2.pos \geq m1.pos$ do
      $m.val := m_1.val * m_2.val$
      $m.var := m_2.var$
      $m.prec := refMult[m1 \cdot pos, m2 \cdot prec]$
      $refMult[m1.pos, m2.pos] := InsertIntListAndSet(MT(\mathcal{B}), m, fm)$
      /* commutativite */
      $refMult[m2.pos, m1.pos] := refMult[m1.pos, m2.pos]$
      $fm = m$
$End.$

Preuve de l'algorithme : Le test $m_2.pos \geq m1.pos$ traduit le fait que nous calculons $\omega_i\omega_j$ seulement pour $i \geq j$, utilisant la commutativité dans $K[X_1, \ldots, X_n]$ pour déduire $\omega_j\omega_i$.

Comme $\mathcal{B}$ à été préalablement calculé, liant les différents monômes à l'aide du lemme 2.2.4, cet algorithme utilise le lemme 2.2.3 pour construire $MT(\mathcal{B})$. □

Calcul de $\mathcal{B}$

Comme $\overrightarrow{\omega_i}$ est stocké à la ième position dans $tcalc$, on peut facilement initialiser les D premières valeurs de $tcalc$ (représentant les formes réduites des monômes de la base) :

$$for\ i\ from\ 1\ to\ D\ do\ tcalc[i] := [\underbrace{0, \ldots, 0}_{i-1}, 1, 0, \ldots, 0]$$

Calcul de $Border(\mathcal{B})$

On suppose que la liste $\mathcal{G} = \{g_1, \ldots, g_s\}$ est ordonnée pour l'ordre sur les monômes choisi. D'après l'algorithme Prepare-$\mathcal{B}$, $\overrightarrow{lm(g_i)}$ est stocké à la $(D+i)^e$ position dans $tcalc$. On initialise alors ces termes par :

$$for\ i\ from\ 1\ to\ \sharp\mathcal{G}\ do\ tcalc[i+D] := -\frac{1}{lc(g_i)}\overrightarrow{rest(g_i)}$$

(Le calcul de $\overrightarrow{rest(g_i)}$ est trivial lorsque la base de Groebner $\mathcal{G}$ est réduite).

On suppose que l'on dispose d'un moyen pour détecter si un élément de $tcalc$ à déjà été calculé ou non (grace à une initialisation particulière des éléments de $tcalc$) et on considère alors l'algorithme suivant :

Procédure Compute-$Border(\mathcal{B})$

- **Entrée :** $\overline{\mathcal{B}} = \mathcal{B} \bigcup Border(\mathcal{B})$ (ordonnée)
- **Sortie :** $tcalc$ réactualisée.

```
Begin
    for m ∈ B̄ do
        if not computed(tcalc[m.pos]) do
            tmp:=tcalc[m.prec]
            tcalc[m.pos]:=[0,..., 0]
            for i from 1 to D do
                tcalc[m.pos]=tcalc[m.pos]+tmp[i]*tcalc[refX[m.var,i]]
End.
```

Preuve de l'algorithme : Le lemme 2.2.4 nous assure que tmp est toujours bien défini. Comme $\mathcal{B}$ à été calculé précédement et comme $\mathcal{B} \bigcup Border(\mathcal{B})$ est ordonné pour $<$, toutes les données $tcalc[refX[m.var, i]]$ $(\overrightarrow{X_{m.var} \cdot tmp})$ nécessaires au calcul de $tcalc[m.pos]$ ont donc été calculées auparavant. Ainsi, en utilisant le lemme 2.2.4, on peut voir que l'algorithme calcule correctement $\mathcal{B} \bigcup Border(\mathcal{B})$. □

Calcul de $MT(\mathcal{B})$

On peut maintenant construire simplement les matrices M_{X_i}, $i = 1, \ldots n$ (multiplication par X_i dans $K[X_1, \ldots, X_n]/\mathcal{I}$), dont les colonnes sont constituées des vecteurs $X_i\omega_j$, $i = 1 \ldots n$, $j = 1 \ldots D$ ($X_i\omega_j$ à été calculé par la procédure précédente et stocké dans $tcalc[refX[i,j]]$).

Procédure Compute-$MT(\mathcal{B})$

- **Entrée :** $MT(\mathcal{B})$ (ordonnée)
- **Sortie :** $tcalc$ réactualisée.

Begin
    for $m \in MT(\mathcal{B})$ do
        if not computed(tcalc[m.pos]) do
            tcalc[m.pos]:=$M_{m.var} \cdot tcalc[m.prec]$
End.

Preuve de l'algorithme : Comme $MT(\mathcal{B})$ est ordonnée pour $<$, $tcalc[m.prec]$ est bien défini au moment où l'on calcule $tcalc[m.pos]$. La preuve est alors évidente grâce au lemme 2.2.3. □

2.2.4 Comportement en pratique

Dans cette partie, nous détaillons, sur une panoplie d'exemples, le comportement de l'algorithme et de toutes ses étapes intermédiaires.

Mesures Brutes

Légende :

- D. : dimension de l'espace vectoriel.
- V. : nombre de variables dans le problème.
- R. : rapport (nombre de calculs effectués)/(D^2)
- L. : taille binaire du plus grand entier apparaissant dans le résultat. (i.e : plus petit entier i tel que 2^{32*i} soit supérieur au plus grand entier apparaissant dans le résultat).
- T : temps de calcul en secondes sur station DEC Alpha-600 - 175 Mhz
- G.B. : calcul de la base de Groebner pour l'ordre DRL par l'algorithme *modular trace* de Gb.
- M.T. : table de multiplication :
  - Prep. : préparation de la table et calcul de $\mathcal{B}$
  - Bord : calcul de $Border(\mathcal{B})$
  - Table : calcul de $MT(\mathcal{B})$

- Nb : nombre de combinaisons linéaires (Border) ou de produits Matrice/vecteur (Table) effectivement calculés.

	GB	M.T.									D.	V
		Prep.	Bord			Table			Total			
Ex.	L.	T.	T.	L.	Nb.	T.	L.	Nb.	T.	R.		
e1	1	0,1	0,31	1	52	0,9	1	367	1,31	0,13	56	4
e2	5	0,04	0,16	5	20	0,04	7	37	0,24	0,22	16	4
e3	3	0,06	0,09	3	8	0,58	15	110	0,73	0,06	46	2
e4	6	0,02	1,42	6	76	1,24	14	124	2,68	0,20	32	6
e5	10	0,69	83,05	11	402	317,78	15	1489	401,52	0,12	128	8
e6	1	0,11	1,05	1	137	1,15	1	357	2,31	0,10	70	5
e7	2	0,44	50,19	3	217	93,16	123	1250	143,79	0,07	144	5
e8	9	0,81	48,213	9	448	174,79	13	1611	223,813	0,13	128	8
e9	7	0,08	2,61	21	177	3,83	27	235	6,52	0,26	40	9
e10	51	0,017	4,88	51	55	1,38	72	35	6,277	0,35	16	9
e11	43	0,06	14,23	44	112	12,23	66	175	26,52	0,28	32	9
e12	65	0,036	15,013	65	65	11,23	101	92	26,279	0,27	24	9
e13	91	0,116	46,94	91	114	133,58	-	253	180,636	0,23	40	9
e14	101	0,06	58,52	101	93	161,11	-	255	219,69	0,22	40	9
e15	5	0,08	5,61	5	177	4,18	-	238	9,87	0,26	40	8

On peut remarquer que notre algorithme fait gagner un nombre important de calculs grace à l'élimination des redondances. Chaque calcul étant lui-même réduit à sa plus simple expression (combinaison linéaire de vecteurs ou produit matrice / vecteur) le gain par rapport à un algorithme naif est très important.
Le comportement général de notre algorithme en termes de temps de calcul est donné par la figure 2.2 (les temps ont été normalisés). .
Nous pouvons estimer avoir rempli notre cahier des charges en ce sens que la gestion des données (construction de l'arbre de calcul - élimination des redondances) est totalement négligeable par rapport aux étapes de calcul, la principale part du temps de calcul étant réservée aux opérations arithmétiques.
Il peut apparaitre étonant que le temps de calcul dédié au bord de la base soit aussi important. Ceci est essentiellement dû aux choix effectués pour la représentation des données. Il faut en effet noter qu'en théorie, même si la base de Groebner (ou le système initial) est à coefficients dans $\mathbb{Z}$, les calculs effectués dans la table de multiplication sont dans $\mathbb{Q}$. Pour optimiser les opérations arithmétiques, nous avons choisi de représenter un vecteur (resp. une matrice carrée) de $\mathbb{Q}^D$ par :

- Un rationnel
- Un vecteur (resp. une matrice carrée) à coefficients dans $\mathbb{Z}$

Cette représentation limite très fortement les calculs de pgcd mais favorise la multiplication Matrice/Vecteur face à la combinaison linéaire de vecteurs.

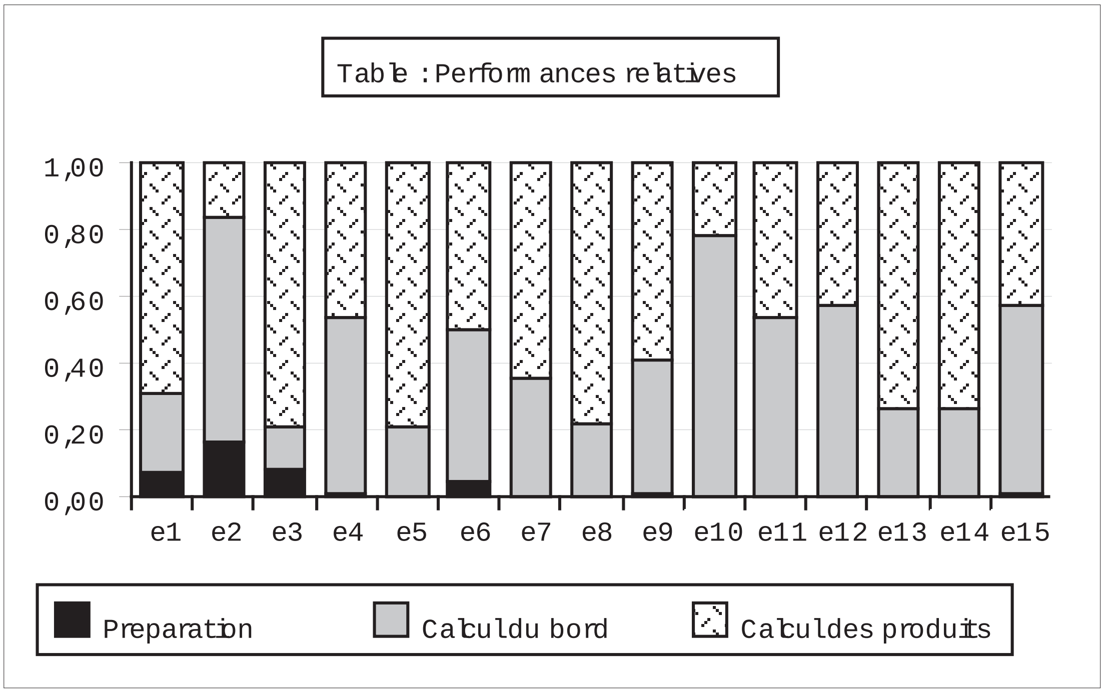

Figure 2.2: Table : temps de calculs normalisés

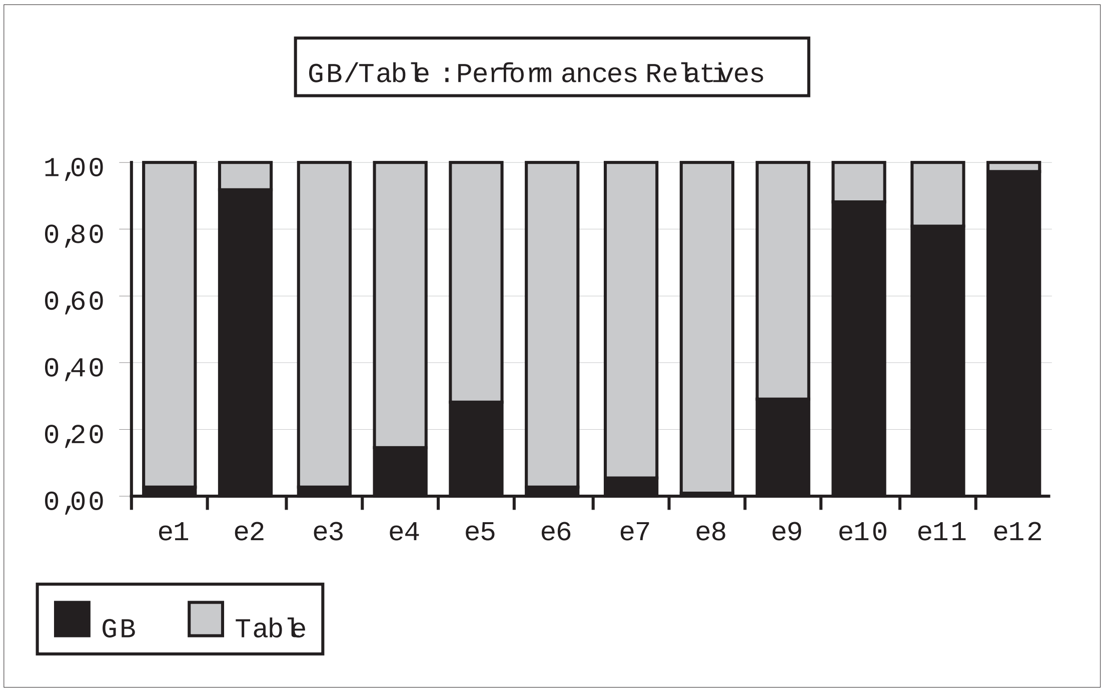

Figure 2.3: Temps de calcul relatifs - Base de Groebner / Table

À titre indicatif, la figure 2.3 donne le rapport des temps de calcul entre la base de Groebner (pour l'ordre DRL) par l'algorithme *modular-trace* du logiciel Gb (prédiction modulaire en parallèle) et celui de la table, pour quelques exemples. Aucune conclusion précise ne peut être faite à ce sujet, mais il semble que, en général, la table soit calculable (en des temps décents) dès qu'une base de Groebner peut être obtenue.

2.2.5 Complexité

Nous ne nous attarderons pas sur la complexité des opérations de préparation de la table étant donné que leur coût apparait à peine dans les tableaux de mesures.

Proposition 2.2.1 *Soit $\mathcal{I}$ un idéal zéro-dimensionnel de $K[X_1, \ldots, X_n]$ dont une base de Groebner (pour un ordre admissible) est notée $\mathcal{G}$. Si D est la dimension du K-espace vectoriel $K[X_1, \ldots, X_n]/\mathcal{I}$, le nombre d'opérations arithmétiques élémentaires (dans K) nécessaires pour calculer la table de multiplication est $O(D^4 + nD^3)$.*

Preuve : Chaque élément de $Border(\mathcal{B})$ est calculé grâce à une combinaison linéaire de D vecteurs de dimension D. Le coût d'un telle opération étant $O(D^2)$, le nombre d'éléments de $Border(\mathcal{B})$ étant majoré par nD, le calcul de l'ensemble :

$$\{\overrightarrow{X_i\omega_j}\ ,\ i = 1 \ldots n\ ,\ j = 1 \ldots D\}$$

nécessitera par conséquent $O(nD^3)$ opérations arithmétiques dans le corps de base.

Le nombre d'éléments de type $\omega_i\omega_j$ présents dans la table est majoré par D^2. Chaque élément étant calculé par un produit Matrice/vecteur, le nombre d'opérations arithmétiques nécessaires pour obtenir les réductions des éléments de l'ensemble :

$$\{\omega_i\omega_j\ ,\ i = 1 \ldots D\ ,\ j = 1 \ldots D\}$$

est donc $O(D^4)$ (dans le corps de base). □

D'après le tableau de mesures, nous pouvons remarquer que $O(D^4 + nD^3)$ est un ordre de grandeur très large.

Nous regardons maintenant le cas des systèmes à coefficients entiers (où rationnels), pour mesurer la taille des représentations binaires.

Le degré total d'un élément de $\mathcal{B}$ ne peut exéder D car, si $\omega \in \mathcal{B}$, tout diviseur de ω est dans $\mathcal{B}$. Ainsi, le degré total d'un élément de la table de multiplication est inférieur à $2D$, et donc, le nombre de multiplications Matrice/vecteur successives utilisées pour obtenir la réduction d'un $\omega_i\omega_j$ est $O(D)$. Si la taille des coefficients présents dans les matrices de multiplication $(M_{X_1}, \ldots M_{X_n})$ est notée $\mathcal{T}$, la taille maximale d'un coefficient apparaissant dans l'expression réduite d'un élément de la table sera donc $O(D(\mathcal{T} + log(D)))$.

La taille des coefficients apparaissant dans les matrices de multiplication $(M_{X_1}, \ldots M_{X_n})$ nous servira de point de repère dans nos calculs de complexité futurs. Des représentations de $K[X_1, \ldots, X_n]/\mathcal{I}$ autres que celles induites par les bases de Groebner pourraient en effet être considérées, et la donnée de ces matrices nous semble être un prérequis minimal. En particulier, il est possible que les techniques développées dans [Car93] puissent à l'avenir nous fournir l'information nécessaire.

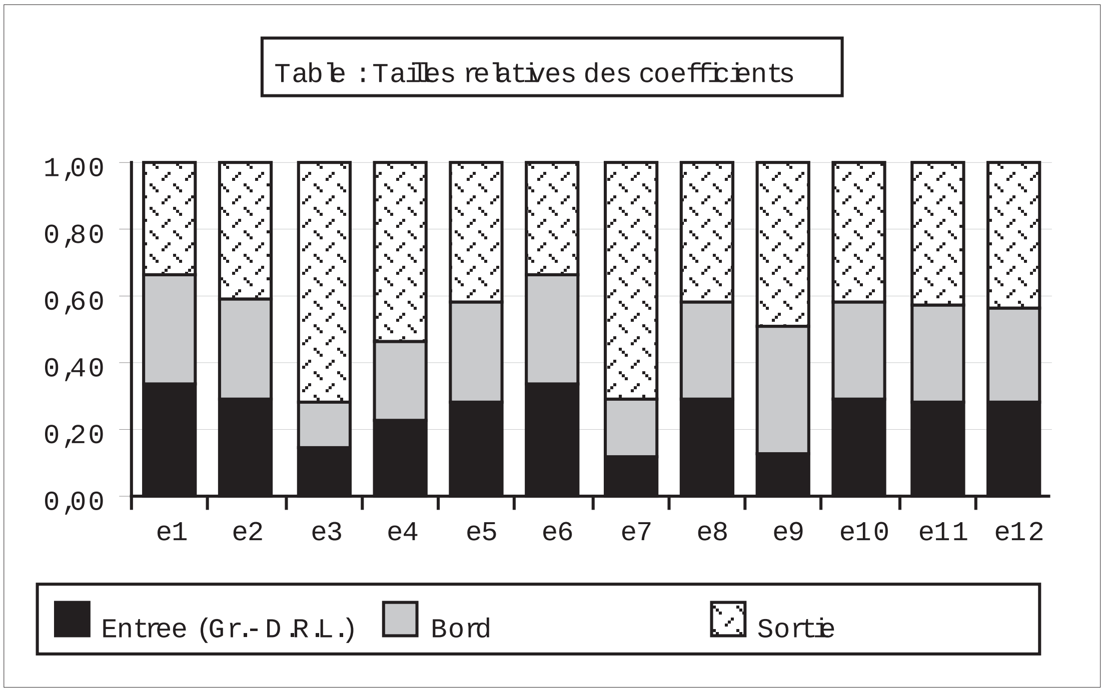

Figure 2.4: Table : Croissance des coefficients

Enfin, la figure 2.4 montre que, en pratique, la taille des coefficients reste parfaitement contrôlée lors du calcul de la table, malgré une complexité théorique en $O(D(\mathcal{T}+log(D)))$. Il faut,à ce sujet, noter l'influence de l'allure des matrices de multiplication qui conditionnent l'ensemble du calcul. Le caractère creux de certaines d'entre elles n'est pas pris en compte dans le calcul de complexité, mais contribue au bon comportement de l'algorithme.

2.2.6 Une application : Le calcul des formes quadratiques de Hermite généralisées

On suppose, dans toute cette partie, que $\mathcal{I}$ est un idéal zéro-dimensionnel de $K[X_1, \ldots, X_n]$.

Étant donnée une table de multiplication de $K[X_1,\ldots,X_n]/\mathcal{I}$, il est facile, si $\mathcal{B}=\{\omega_1,\ldots,\omega_D\}$ est la base choisie pour le calcul de la table de multiplication, de calculer le vecteur :

$$Vtr(1)=[Trace(m_{\omega_1}),\ldots,Trace(m_{\omega_D})]$$

Si l'expression de $\overrightarrow{h}$ dans $\mathcal{B}$ est connue (on peut l'obtenir en appliquant l'algorithme NF avec la base de Groebner de $\mathcal{I}$ utilisée pour le calcul de la table de multiplication) et si q_h est la matrice de Q_h dans $\mathcal{B}$, le lemme 1.2.3 nous donne la relation suivante :

$$\forall g,f\in K[X_1,\ldots,X_n]/\mathcal{I}\ ,\ Trace(m_{fg})=\overrightarrow{f}\cdot Vtr(g)$$

avec $Vtr(g)=[Trace(m_{g\omega_1}),\ldots,Trace(m_{h\omega_D})]$.
En particulier, nous aurons pour tout $i,j=1,\ldots D$:

$$q_1[i,j]=Trace(m_{\omega_i\omega_j})=\overrightarrow{\omega_i\omega_j}\cdot Vtr(1)$$

Proposition 2.2.2 *Si une table de multiplication de $K[X_1,\ldots,X_n]/\mathcal{I}$ est connue, la matrice q_1 peut être calculée en $O(D^3)$ opérations arithmétiques élémentaires dans K.*

Dans le cas où K est le corps des rationnels, et si $\mathcal{T}'$ est la taille binaire du plus grand entier apparaissant dans les expressions des vecteurs de la table, la taille maximale d'un coefficient apparaissant dans q_1 est $O(\mathcal{T}'+log(D))$.

Ainsi, si $\mathcal{T}$ est la taille binaire du plus grand entier apparaissant dans les matrices M_{X_i} , $i=1,\ldots,n$, la taille maximale d'un coefficient apparaissant dans q_1 est $O(D(\mathcal{T}+log(D)))$.

Preuve : En remarquant que, pour tout $i=1\ldots D$, $Trace(m_{\omega_i})=\sum_{j=1\ldots D}(\overrightarrow{\omega_i\omega_j})_k$, on s'aperçoit que $Vtr(1)$ peut être calculé en $O(D)$ opérations arithmétiques, et que, dans le cas où K est le corps des rationnels, la taille des coefficients apparaissant dans $Vtr(1)$ est en $O(\mathcal{T}'+log(D))$.

La construction de q_1 nécessite alors D^2 produits scalaires de vecteurs de dimension D ce qui entraînera $O(D^3)$ opérations arithmétiques limitant, dans le cas où K est le corps des rationnels, la taille des coefficients apparaissant dans le résultat à $O(\mathcal{T}'+log(D))$. □

Ainsi, il en coûtera, en incluant le calcul de table de multiplication, $O(D^4+nD^3)$ opérations arithmétiques pour la construction de la forme quadratique Q_1 si le point d'entrée de l'algorithme est une base de Groebner.

Dans le cas général ($h\neq 1$), nous pouvons utiliser les mêmes arguments que précédemment pour construire q_h :

$$q_h[i,j] = Trace(m_{h\omega_i\omega_j}) = \overrightarrow{\omega_i\omega_j} \cdot Vtr(h)$$

Si on note $q_1[i]$ la i^e ligne de q_1, il vient :

$$Trace(m_{h\omega_i}) = \overrightarrow{h}\,Vtr(\omega_i) = \overrightarrow{h}\,q_1[i]$$

Par conséquent, $Vtr(h) = q_1\overrightarrow{h}$ et donc :

$$q_h[i,j] = \overrightarrow{\omega_i\omega_j} \cdot (q_1\overrightarrow{h}),$$

ce qui nous permet d'établir la proposition :

Proposition 2.2.3 *Soit $h \in K[X_1,\ldots,X_n]$. Si une table de multiplication de $K[X_1,\ldots,X_n]/\mathcal{I}$ est connue, la matrice q_h peut être calculée en $O(D^3)$ opérations arithmétiques élémentaires.*

Dans le cas où K est le corps des rationnels, si $\mathcal{T}'$ est la taille binaire du plus grand entier apparaissant dans les expressions des vecteurs de la table et si $\mathcal{T}'_h$ est la taille du plus grand entier apparaissant dans l'expression de $\overrightarrow{h}$, la taille maximale d'un coefficient apparaissant dans q_h est $O(max(\mathcal{T}',\mathcal{T}'_h) + log(D))$.

Chapitre 3

Résolution des systèmes zéro-dimensionnels

Résumé

Nous abordons, dans ce chapitre, le problème de la simplification des systèmes polynomiaux et de l'élimination de variables.

On trouve, dans la littérature, beaucoup de résultats utilisant ce que nous avons appelé les polynômes séparants. Si K est un corps de caractéristique 0 de clôture algébrique C, pour un idéal $\mathcal{I} \subset K[X_1, \ldots, X_n]$ dont l'ensemble des zéros $V(\mathcal{I}) \subset C^n$ est fini et non vide, l'idée est de trouver une fonction polynomiale, injective de $V(\mathcal{I})$ dans C, de sorte à se ramener à l'étude d'un ensemble fini de points (algébriques) de C.

Dans le cas d'un idéal radical, ceci revient à déterminer un élément primitif (voir par exemple [GH91b] ou [BW93]), mais, dans le cas général, le problème est plus complexe. On peut imaginer se ramener systématiquement au cas radical (des solutions algorithmiques existent d'ailleurs à cet effet - voir [GM89] par exemple), mais on perd alors toute information sur les multiplicités des racines.

Dans [Ca88], un calcul de u-résultant fournit, après déformation du système (ajout d'une variable), une représentation agréable des zéros, les coordonnées des solutions étant exprimées comme fractions rationnelles en les racines d'un unique polynôme en une variable. Un procédé identique est présent dans [Re92], utilisant une arithmétique à base d'infinitésimaux.

Une autre solution est proposée dans [ABRW94], autorisant, en particulier, la représentation de $\mathcal{I}$ par une base de Groebner. Cette fois, aucune déformation n'est utilisée et les calculs s'en trouvent simplifiés.

Nous proposons une définition générale de la représentation univariée, indépendante de toute méthode de calcul, où les expressions ne dépendent pas des multiplicités. Nous montrons que tout idéal zéro-dimensionnel $\mathcal{I} \subset K[X_1, \ldots, X_n]$ est

équivalent, en un certain sens, à un idéal $\mathcal{J} \subset K[T]$, définissant ainsi une notion simple de *Résolution d'un système zéro-dimensionnel.*

Nous montrons ensuite qu'il existe un algorithme permettant de calculer explicitement un isomorphisme d'ensembles algébriques $\phi : V(\mathcal{I}) \longrightarrow V(\mathcal{I})$, préservant les multiplicités des racines et défini par un ensemble de $n+1$ polynômes de $K[T]$ (Représentation univariée rationnelle). Cet isomorphisme possède un certain nombre de propriétés agréables comme la préservation des racines réelles, ou encore que les polynômes de la représentation univariée rationnelle permettent, dans certains cas, de calculer simplement une base de Groebner de $\mathcal{I}$ pour l'ordre lexicographique.

Pour le calcul explicite de cette représentation, nous introduisons une extension des notions habituelles de fonctions symétriques élémentaires et de sommes de Newton. Une généralisation de la formule de Newton nous donne alors un algorithme efficace (RUR) pour le calcul de notre isomorphisme, lorsqu'un polynôme séparant est connu. Pour le cas général, nous utilisons une idée présente dans [Ca88], [Re92] ou encore [ABRW94], dont le principe est de déterminer une famille finie de polynômes dont l'un sépare assurément $V(\mathcal{I})$. Un test nouveau et efficace en pratique, nous permet de vérifier, a posteriori, si un élément choisi arbitrairement est séparant ou non.

Une attention toute particulière est accordée au cas des systèmes à coefficients entiers, pour lesquels nous proposons une méthode modulaire pour la détermination d'un élément séparant $V(\mathcal{I})$.

Une analyse détaillée de la complexité montre que le calcul de notre représentation univariée rationnelle par les algorithmes décrits dans cette thèse est :

- Efficace (nombre d'opérations arithmétiques optimisé, taille des calculs intermédiaires, pas de déformation du système initial - l'ajout d'une variable complique fortement les calculs),

- Utilisable (taille de la sortie).

Dans le cas particulier des systèmes à coefficients rationnels, nous montrons de plus que le fait de ne pas connaître de polynôme séparant $V(\mathcal{I})$ n'influe que peu, en pratique, sur le comportement de l'algorithme.

3.1 μ-résolution, représentation univariée d'un système zéro-dimensionnel

Dans toute cette partie, K désigne un corps de caractéristique 0 et de clôture algébrique C.

Nous proposons de ramener l'étude d'un idéal zéro-dimensionnel de $K[X_1, \ldots, X_n]$ à celle d'une famille de fractions rationnelles de $K(T)$. Nous définissons, pour commencer, une équivalence entre idéaux faisant appel à des morphismes d'ensembles algébriques particuliers :

Definition 3.1.1 *Soit $\mathcal{E} \subset C^n$ un ensemble algébrique. Une application $\phi^\star : \mathcal{E} \subset C^n \longrightarrow C$ sera dite K-régulière si il existe $p \in K[X_1, \ldots, X_n]$ tel que $\forall \alpha \in \mathcal{E}$, $p(\alpha) = \phi^\star(\alpha)$*

De la même manière, si $\mathcal{E} \subset C^n$ et $\mathcal{F} \subset C^m$ sont deux ensembles algébriques, l'application $\phi^\star : \mathcal{E} \subset C^n \longrightarrow \mathcal{F} \subset C^m$ sera un K-morphisme d'ensembles algébriques si chacune de ses fonctions coordonnée est K-régulière.

L'application $\phi^\star : \mathcal{E} \subset C^n \longrightarrow \mathcal{F} \subset C^m$ sera un K-isomorphisme d'ensembles algébriques si il existe un K-morphisme d'ensembles algébriques $\psi^\star : \mathcal{F} \subset C^m \longrightarrow \mathcal{E} \subset C^n$ tel que $\psi^\star \circ \phi^\star = Id_{\mathcal{E}}$ et $\phi^\star \circ \psi^\star = Id_{\mathcal{F}}$.

A tout morphisme de K-algèbres $\phi : K[X_1, \ldots, X_n] \longrightarrow K[Y_1, \ldots, Y_m]$, on associe canoniquement un K-morphisme d'ensembles algébriques en posant :

$$\begin{array}{cccc} \phi^\star : & C^m & \longrightarrow & C^n \\ & \alpha & \longmapsto & (\phi(X_1)(\alpha), \ldots, \phi(X_n)(\alpha)) \end{array}$$

Nous insistons sur le fait que ces morphismes sont définis sur K (et non sur C), ce qui a, comme nous le verrons plus tard, un grand intéret algorithmique.
Il résulte de ces définitions quelques propriétés simples qui nous seront utiles par la suite :

Proposition 3.1.1 *Soient $\phi : K[X_1, \ldots, X_n] \longrightarrow K[Y_1, \ldots, Y_m]$ un morphisme de K-algèbres , $\mathcal{E} \subset C^n$ et $\mathcal{F} \subset C^m$ deux ensembles algébriques tels que $\phi^\star$ induise un K-isomorphisme de $\mathcal{F}$ dans $\mathcal{E}$.*

Pour tout morphisme de K-algèbres $\psi : K[Y_1, \ldots, Y_m] \longrightarrow K[X_1, \ldots, X_n]$ tel que $\psi^\star$ coincide avec $(\phi^\star)^{-1}$ sur $\mathcal{E}$,

- $\forall \alpha \in \mathcal{E}$, $\forall p \in K[X_1, \ldots, X_n]$, $p(\alpha) = \phi(p)(\psi^\star(\alpha))$
- $\forall \alpha \in \mathcal{E}$, $\forall p \in K[X_1, \ldots, X_n]$, $p(\alpha) = (\psi \circ \phi)(p)(\alpha)$

Preuve : Pour tout $\beta \in \mathcal{F}$,

$$p(\phi^\star(\beta)) = p(\phi(X_1)(\beta), \ldots, \phi(X_n)(\beta)).$$

Comme ϕ est un morphisme d'algèbres, ceci nous donne :

$$p(\phi^\star(\beta)) = \phi(p)(\beta).$$

Pour tout $\beta \in \mathcal{F}$, $\phi^\star$ étant un isomorphisme de $\mathcal{F}$ dans $\mathcal{E}$, il existe $\alpha \in \mathcal{E}$ tel que $\alpha = \phi^\star(\beta)$. Par définition de $\psi^\star$, nous avons alors $\beta = \psi^\star(\alpha)$ et par conséquent,

$$p(\alpha) = \phi(p)(\psi^\star(\alpha)).$$

De plus, $\forall \alpha \in \mathcal{E}$, $\forall p \in K[X_1, \ldots, X_n]$,

$$(\psi \circ \phi)(p)(\alpha) = (\psi(\phi(p))(\phi^\star(\psi^\star(\alpha))) = \phi(p)(\psi^\star(\alpha)) = p(\alpha).$$

□

Introduisons maintenant une relation d'équivalence entre idéaux zéro-dimensionnels :

Definition 3.1.2 *Soient $\mathcal{I} \subset K[X_1, \ldots, X_n]$ et $\mathcal{J} \subset K[Y_1, \ldots, Y_m]$ deux idéaux zéro-dimensionnels. Nous dirons que $\mathcal{I}$ est μ-équivalent à $\mathcal{J}$, et nous noterons $\mathcal{I} \simeq_\mu \mathcal{J}$, si il existe un K-ismorphisme $\phi^\star : V(\mathcal{J}) \longrightarrow V(\mathcal{I})$ tel que :*

$$\forall \beta \in V(\mathcal{J}) \ , \ \mu(\phi^\star(\beta)) = \mu(\beta)$$

Nous avons pu voir que l'application *Trace* est à la base de résultats utiles pour l'étude des systèmes zéro-dimensionnels. Montrons que la μ-équivalence *préserve les traces* :

Proposition 3.1.2 *Soient $\mathcal{I} \subset K[X_1, \ldots, X_n]$ et $\mathcal{J} \subset K[Y_1, \ldots, Y_m]$ deux idéaux zéro-dimensionnels. Les idéaux $\mathcal{I}$ et $\mathcal{J}$ sont μ-équivalents si et seulement si il existe un morphisme de K algèbres $\phi : K[X_1, \ldots, X_n] \longrightarrow K[Y_1, \ldots, Y_m]$ tel que :*

- *$\phi^\star$ induise un K-isomorphisme d'ensembles algébriques de $V(\mathcal{J})$ dans $V(\mathcal{I})$*
- $\forall p \in K[X_1, \ldots, X_n]$, $\sum_{\alpha \in V(\mathcal{I})} \mu(\alpha) p(\alpha) = \sum_{\beta \in V(\mathcal{J})} \mu(\beta)\phi(p)(\beta)$

En reprenant la notation 1.1.1 et en utilisant le théorème 1.1.1, la dernière assertion peut se traduire par :

$$\forall p \in K[X_1, \ldots, X_n] \ , \ Trace(m_p) = Trace(m_{\phi(p)})$$

Preuve :

- Si $\mathcal{I}$ et $\mathcal{J}$ sont μ-équivalents , il existe un morphisme de K-algèbres ϕ : $K[X_1,\ldots,X_n] \longrightarrow K[Y_1,\ldots,Y_m]$ tel que $\phi^\star$ vérifie les propriétés de la définition 3.1.2. Soit $p \in K[X_1,\ldots,X_n]$. L'application $\phi^\star$ étant un K-isomorphisme d'ensembles algébriques, nous avons :

$$\sum_{\beta\in V(\mathcal{J})} \mu(\beta)\phi(p)(\beta) = \sum_{\alpha\in V(\mathcal{I})} \mu((\phi^\star)^{-1}(\alpha))\phi(p)((\phi^\star)^{-1}(\alpha))$$

  D'après la proposition 3.1.1 il vient alors :

$$\sum_{\beta\in V(\mathcal{J})} \mu(\beta)\phi(p)(\beta) = \sum_{\alpha\in V(\mathcal{I})} \mu((\phi^\star)^{-1}(\alpha))p(\alpha)$$

  et enfin, d'après la définition 3.1.2 :

$$\sum_{\beta\in V(\mathcal{J})} \mu(\beta)\phi(p)(\beta) = \sum_{\alpha\in V(\mathcal{I})} \mu(\alpha)p(\alpha).$$

- En appliquant les formules classiques d'interpolation de Lagrange, on définit, pour tout $\gamma \in V(\mathcal{I})$, un polynôme $p_\gamma \in C[X_1,\ldots,X_n]$ tel que $p_\gamma(\gamma) = 1$ et $\forall\alpha \neq \gamma$, $\alpha \in V(\mathcal{I})$ $p_\gamma(\alpha) = 0$.

  L'application $\phi^\star$ étant une bijection de $V(\mathcal{J})$ dans $V(\mathcal{I})$,

$$\forall\beta \in V(\mathcal{J}) \ , \ \exists\alpha \in V(\mathcal{I}) \ , \ \alpha = \phi^\star(\beta)$$

  Ainsi, $\sharp V(\mathcal{J}) = \sharp V(\mathcal{I})$ et, en étendant cannoniquement ϕ en un morphisme de C-algèbres $\phi_C : C[X_1,\ldots,X_n] \longrightarrow C[Y_1,\ldots,Y_m]$ ($\phi^\star$ et $\phi_C^\star$ coincident alors sur $V(\mathcal{J})$), la proposition 3.1.1 nous donne :

$$\sum_{\alpha\in V(\mathcal{I})} \mu(\alpha)P_\gamma(\alpha) = \sum_{\alpha\in V(\mathcal{I})} \mu((\phi^\star)^{-1}(\alpha))\phi_C(P_\gamma)((\phi^\star)^{-1}(\alpha))$$

  Comme $\phi_C(P_\gamma)((\phi^\star)^{-1}(\alpha)) = P_\gamma(\alpha)$ nous avons alors $\mu(\gamma) = \mu((\phi^\star)^{-1}(\gamma))$, pour tout $\gamma \in V(\mathcal{I})$.

□

Remarque 3.1.1 *Soient $\mathcal{I} \subset K[X_1,\ldots,X_n]$ et $\mathcal{J} \subset K[Y_1,\ldots Y_m]$, deux idéaux μ-équivalents. Les K-algèbres $K[X_1,\ldots,X_n]/\mathcal{I}$ et $K[Y_1,\ldots,Y_m]/\mathcal{J}$ ne sont pas , en général, isomorphes.*

Considérons par exemple $\mathcal{I} = \langle X_1^2, X_1X_2, X_2^2\rangle \subset K[X_1,X_2]$ et $\mathcal{J} = \langle Y_1^3\rangle \subset K[Y_1]$. En posant :

$$\begin{array}{rccc} \phi : & K[X_1,X_2] & \longrightarrow & K[Y_1] \\ & X_1 & \longmapsto & Y_1 \\ & X_2 & \longmapsto & 0 \end{array}$$

on peut voir que $\phi^\star$ est un K-isomorphisme d'ensembles algébriques vérifiant les hypothèses de la définition 3.1.2 et nous avons alors $\mathcal{I} \simeq_\mu \mathcal{J}$.
Supposons que $\psi : K[Y_1]/\mathcal{J} \longrightarrow K[X_1, X_2]/\mathcal{I}$ soit un morphisme de K-algèbres, et posons $\psi(Y_1) = aX_1 + bX_2$, $a, b \in K$.
Dans ce cas,

$$\psi(Y_1^2) = \psi(Y_1)^2 = a^2X_1^2 + b^2X_2^2 + 2abX_1X_2 = 0 \; mod \, \mathcal{I}$$

et ψ n'est pas injectif.
Il n'existe donc pas d'isomorphisme de K-algèbres entre $K[Y_1]/\mathcal{J}$ et $K[X_1, X_2]/\mathcal{I}$.
Pour des raisons évidentes de conservation des multiplicités, l'existence d'un isomorphisme entre $K[X_1, \ldots, X_n]/\sqrt{\mathcal{I}}$ et $K[Y_1, \ldots, Y_m]/\sqrt{\mathcal{J}}$ n'implique pas que les deux idéaux soient μ-équivalents. La réciproque est par contre vraie :

Proposition 3.1.3 *Soient $\mathcal{I} \subset K[X_1, \ldots, X_n]$ et $\mathcal{J} \subset K[Y_1, \ldots, Y_m]$ deux idéaux zéro -dimensionnels tels que $\mathcal{I} \simeq_\mu \mathcal{J}$. Notons alors $\phi : K[X_1, \ldots, X_n] \longrightarrow K[Y_1, \ldots, Y_m]$ un morphisme de K-algèbres vérifiant les hypothèses de la définition 3.1.2.*
Le morphisme de K-algèbres $\tilde{\phi} : K[X_1, \ldots, X_n] \longrightarrow K[Y_1, \ldots, Y_m]/\sqrt{\mathcal{J}}$, canoniquement déduit de ϕ, induit un isomorphisme de K-algèbres :

$$\phi_{red} : K[X_1, \ldots, X_n]/\sqrt{\mathcal{I}} \longrightarrow K[Y_1, \ldots, Y_m]/\sqrt{\mathcal{J}}$$

Preuve : Comme $\phi^\star$ est, par définition, un K-isomorphisme d'ensembles algébriques, pour tout $\alpha \in V(\mathcal{I})$, $\mu(\alpha) = \mu((\phi^\star)^{-1}(\alpha))$.
De plus, pour tout $p \in \sqrt{\mathcal{I}}$ et tout $\beta \in V(\mathcal{J})$, il existe $\alpha \in V(\mathcal{J})$ tel que $\phi(p)(\beta) = \phi(p)((\phi^\star)^{-1}(\alpha)) = p(\alpha) = 0$. Par conséquent, $\phi(p) \in \sqrt{\mathcal{J}}$.
Inversement, soit $p \in K[X_1, \ldots, X_n]$ tel que $\phi(p) \in \sqrt{\mathcal{J}}$. Pour tout $\alpha \in V(\mathcal{I})$, il existe $\beta \in V(\mathcal{J})$ tel que $(\phi^\star)^{-1}(\alpha) = \beta$. Dans ce cas,

$$p(\alpha) = \phi(p)((\phi^\star)^{-1}(\alpha)) = \phi(p)(\beta) = 0$$

Ainsi, $ker(\tilde{\phi}) = \sqrt{\mathcal{I}}$ et donc $\tilde{\phi}$ se factorise en un morphisme injectif

$$\phi_{red} : K[X_1, \ldots, X_n]/\sqrt{\mathcal{I}} \longrightarrow K[Y_1, \ldots, Y_m]/\sqrt{\mathcal{J}}$$

Comme $\sharp V(\mathcal{J}) = \sharp V(\mathcal{I})$, nous avons alors

$$dim_K(K[X_1, \ldots, X_n]/\sqrt{\mathcal{I}}) = dim_K(K[Y_1, \ldots, Y_m]/\sqrt{\mathcal{J}})$$

ce qui montre que ϕ_{red} est bijectif. □

Les diverses propriétés de la μ-équivalence nous conduisent alors à définir ce que nous entendons désormais par *résoudre* un système zéro-dimensionnel :

Definition 3.1.3 *Un idéal zéro-dimensionnel $\mathcal{I} \subset K[X_1,\dots,X_n]$ sera dit μ-résolu, si il existe un idéal $\mathcal{J} \subset K[T]$ tel que $\mathcal{I} \simeq_\mu \mathcal{J}$. Si ϕ est un morphisme de K-algèbres vérifiant les propriétes de la définition 3.1.2, nous dirons alors que $(\mathcal{J}, \phi^\star)$ est une μ-résolution de $\mathcal{I}$.*

Remarque 3.1.2 *Si $(\mathcal{J}, \phi^\star)$ et $(\mathcal{J}, (\phi')^\star)$ sont deux μ-résolutions de $\mathcal{I}$, les K-isomorphismes $\phi^\star$ et $(\phi')^\star$ coincident sur $V(\mathcal{J})$. Par abus de notation, nous écrirons que $\mathcal{J}$ est une μ-résolution de $\mathcal{I}$.*

Le terme de résolution ne parait pas abusif, car la donnée d'une μ-résolution permet de se ramener à un problème en une variable pour lequel beaucoup d'outils algorithmiques sont disponibles tant pour l'approximation que pour le codage des racines. La donnée d'un K-isomorphisme d'ensembles algébriques nous permet de *remonter* les résultats.
Le résultat que nous nous proposons de montrer dans la suite de cette partie est le suivant :

Théorème 3.1.1 *Tout idéal zéro-dimensionnel $\mathcal{I} \subset K[X_1,\dots,X_n]$ admet une μ-résolution.*

Pour tout $h \in K[X_1,\dots,X_n]$, rappelons que m_h désigne l'opérateur *multiplication* par h dans $\mathcal{A} = K[X_1,\dots,X_n]/\mathcal{I}$ (notation 1.1.1).

Notation 3.1.1 *Pour tous $t, v \in K[X_1,\dots,X_n]$ et tout $u \in C$, on définit les ensembles :*

$$V_t^u(\mathcal{I}) = \{\beta \in V(\mathcal{I})\ ,\ t(\beta) = u\}$$

$$t(V(\mathcal{I})) = \{t(\alpha)\ ,\ \alpha \in V(\mathcal{I})\}$$

ainsi que les polynômes :

$$f(t,T) = \prod_{\alpha \in V(\mathcal{I})} (T - t(\alpha))^{\mu(\alpha)}$$

$$g(v,t,T) = \sum_{\alpha \in V(\mathcal{I})} \mu(\alpha) v(\alpha) \prod_{y \in t(V(\mathcal{I})) \setminus \{t(\alpha)\}} (T - y)$$

et on pose :

$$\mu_t(u) = \sum_{\beta \in V_t^u(\mathcal{I})} \mu(\beta)$$

Lemme 3.1.1 *Les polynômes $f(t,T)$ et $g(v,t,T)$ sont dans $K[X_1,\dots,X_n]$. Plus précisément, si S est une variable indépendante de $X_1,\dots,X_n$, et si , pour $h \in K[X_1,\dots,X_n]$ fixé, $P_h(T)$ désigne le polynôme caractéristique de m_h,*

- *$f(t,T)$ est le polynôme caractéristique de m_t.*
- *En considérant l'extension $\mathcal{A}\otimes_K K(S)$, on note $P_{t+Sv}(T)$, le polynôme caractéristique de m_{t+Sv} (multiplication par $t+Sv$ dans $\mathcal{A}\otimes_K K(S)$), et*

$$h(v,t,T) = -\left(\frac{\partial}{\partial S}P_{t+Sv}(T)\right)_{S=0}.$$

Dans ce cas, $\forall v \in K[X_1,\ldots,X_n]$, $g(v,t,T) = h(v,t,T)/gcd(f(t,T),f'(t,T))$

- *$g(t,T) = g(1,t,T) = f'(t,T)/gcd(f(t,T),f'(t,T))$*

Preuve : D'après le théorème 1.1.1, le polynôme caratéristique de m_t peut s'exprimer par :

$$\prod_{\alpha\in V(\mathcal{I})}(T-t(\alpha))^{\mu(\alpha)},$$

ce qui montre que $f(t,T)\in K[X_1,\ldots,X_n]$.
D'après les notations 3.1.1, $f(t,T) = \prod_{y\in t(V(\mathcal{I}))}(T-y)^{\mu_t(y)}$. Par conséquent, $gcd(f(t,T),f'(t,T)) = \prod_{y\in t(V(\mathcal{I}))}(T-y)^{\mu_t(y)-1}$.
Toujours d'après le théorème 1.1.1, en considerant m_{t+Sv} en tant qu'endomorphisme de $\mathcal{A}\otimes_K K(S)$:

$$P_{t+Sv}(T) = \prod_{\alpha\in V(\mathcal{I})}(T-t(\alpha)-Sv(\alpha))^{\mu(\alpha)} \in K[T,S].$$

Ainsi :

$$\left(\frac{\partial}{\partial S}P_{t+Sv}(T)\right)_{S=0} = -\sum_{\alpha\in V(\mathcal{I})}\mu(\alpha)v(\alpha)(T-t(\alpha))^{\mu(\alpha)-1}\prod_{\beta\in V(\mathcal{I})\setminus\{\alpha\}}(T-t(\beta))^{\mu(\beta)}$$

En remarquant que

$$\prod_{\beta\in V(\mathcal{I})\setminus\{\alpha\}}(T-t(\beta))^{\mu(\beta)} = (T-t(\alpha))^{\mu_t(t(\alpha))-\mu(\alpha)}\prod_{y\in t(V(\mathcal{I}))\setminus\{t(\alpha)\}}(T-y)^{\mu_t(y)},$$

$$\left(\frac{\partial}{\partial S}P_{t+Sv}(T)\right)_{S=0} = -\sum_{\alpha\in V(\mathcal{I})}\mu(\alpha)v(\alpha)\prod_{y\in t(V(\mathcal{I}))}(T-y)^{\mu_t(y)-1}\prod_{y\in t(V(\mathcal{I}))\setminus\{t(\alpha)\}}(T-y)$$

et donc $\left(\frac{\partial}{\partial S}P_{t+Sv}(T)\right)_{S=0} = -g(v,t,T)\cdot gcd(f(t,T),f'(t,T))$.
Comme $\left(\frac{\partial}{\partial S}P_{t+Sv}(T)\right)\in K[T,S]$ et comme $gcd(f(t,T),f'(t,T))\in K[T]$,

$$g(v,t,T) = h(v,t,T)/gcd(f(t,T),f'(t,T))\in K[T].$$

Enfin, en groupant differément les termes $\mu(\alpha)$ dans l'expression de $g(1,t,T)$,

$$g(t,T) = \sum_{z\in t(V(\mathcal{I}))}\mu_t(z)\prod_{y\in t(V(\mathcal{I}))\setminus\{z\}}(T-y)$$

et donc $g(t,T) = g(1,t,T) = f'(t,T)/gcd(f(t,T),f'(t,T))$ □

Lemme 3.1.2 *Pour tout $\alpha \in V(\mathcal{I})$, et tous polynômes $t, v \in K[X_1, \dots, X_n]$,*

$$\frac{\sum_{\beta \in V_t^{t(\alpha)}(\mathcal{I})} \mu(\beta) v(\beta)}{\sum_{\beta \in V_t^{t(\alpha)}(\mathcal{I})} \mu(\beta)} = \frac{g(v, t, t(\alpha))}{g(t, t(\alpha))}$$

avec $g(t, T) = g(1, t, T)$.

Preuve : Par définition,

$$g(v, t, t(\alpha)) = \sum_{\beta \in V(\mathcal{I})} \mu(\beta) v(\beta) \prod_{y \in t(V(\mathcal{I})) \setminus \{t(\beta)\}} (t(\alpha) - y)$$

Or , on peut remarquer que $\mu(\beta)v(\beta)\left(\prod_{y \in t(V(\mathcal{I})) \setminus \{t(\beta)\}} (t(\alpha) - y)\right)$ est nul si et seulement si il existe $y \in t(V(\mathcal{I})) \setminus \{t(\beta)\}$ tel que $y = t(\alpha)$ et donc, si et seulement si $\beta \notin V_t^{t(\alpha)}(\mathcal{I})$. Ainsi, $g(v, t, t(\alpha))$ peut s'écrire :

$$g(v, t, t(\alpha)) = \sum_{\beta \in V_t^{t(\alpha)}(\mathcal{I})} \mu(\beta) v(\beta) \prod_{y \in t(V(\mathcal{I})) \setminus \{t(\alpha)\}} (t(\alpha) - y)$$

ou encore :

$$g(v, t, t(\alpha)) = \left(\prod_{y \in t(V(\mathcal{I})) \setminus \{t(\alpha)\}} (t(\alpha) - y)\right) \left(\sum_{\beta \in V_t^{t(\alpha)}(\mathcal{I})} \mu(\beta) v(\beta)\right)$$

Comme $g(t, T) = g(1, t, T)$, nous avons de même :

$$g(t, t(\alpha)) = \sum_{\beta \in V_t^{t(\alpha)}(\mathcal{I})} \mu(\beta) \prod_{y \in t(V(\mathcal{I})) \setminus \{t(\alpha)\}} (t(\alpha) - y)$$

et donc

$$g(t, t(\alpha)) = \left(\prod_{y \in t(V(\mathcal{I})) \setminus \{t(\alpha)\}} (t(\alpha) - y)\right) \left(\sum_{\beta \in V_t^{t(\alpha)}(\mathcal{I})} \mu(\beta)\right)$$

ce qui démontre la formule. □

Proposition 3.1.4 (représentation univariée rationnelle) *Soient $\mathcal{S} = \{f_1, \dots, f_s\}$ un système zéro-dimensionel de $K[X_1, \dots, X_n]$, $\mathcal{I} \subset K[X_1, \dots, X_n]$ l'idéal endendré par $\mathcal{S}$ et $V(\mathcal{I}) = \{\alpha_1, \dots \alpha_d\}$ l'ensemble des racines distinctes de $\mathcal{S}$. Pour tout polynôme $t \in K[X_1, \dots, X_n]$ séparant $V(\mathcal{I})$, il existe des polynômes :*

$$f(t, T)\ ,\ g(t, T)\ ,\ g_1(t, T), \dots, g_n(t, T) \in K[X_1, \dots X_n]$$

tels que :

- *Les racines de $f(t,T)$ soient les $t(\alpha_1),\ldots,t(\alpha_d)$ avec pour multiplicités respectives $\mu(\alpha_1),\ldots,\mu(\alpha_d)$.*
- *Si α est une racine de $\mathcal{S}$, pour tout $i=1\ldots n$:*

$$X_i(\alpha)=\frac{g_i(t,t(\alpha))}{g(t,t(\alpha))}$$

Nous dirons que $\{f(t,T)\ ,\ g(t,T)\ ,\ g_1(t,T),\ldots,g_n(t,T)\}$ est la **représentation univariée rationnelle** *de $\mathcal{I}$ associée à t.*

Preuve : Soit $t\in K[X_1,\ldots,X_n]$ un polynôme séparant $V(\mathcal{I})$ (son existence est prouvée par le lemme 1.2.1). Nous pouvons remarquer que $V_t^{t(\alpha)}(\mathcal{I})=\{\alpha\}$ et par conséquent, pour tout $v\in K[X_1,\ldots,X_n]$ et tout $\alpha\in V(\mathcal{I})$, le lemme 3.1.2 nous donne :

$$\frac{\mu(\alpha)v(\alpha)}{\mu(\alpha)}=v(\alpha)=\frac{g(v,t,t(\alpha))}{g(t,t(\alpha))}$$

Il ne reste alors qu'à poser $g_i(t,T)=g(X_i,t,T)\ ,\ i=1\ldots n$ pour obtenir le résultat énoncé. □

Le résultat suivant achèvera la démonstration du théorème 3.1.1 :

Proposition 3.1.5 *Soient $\mathcal{I}\subset K[X_1,\ldots,X_n]$ un idéal zéro-dimensionnel et*

$$\{f(t,T)\ ,\ g(t,T)\ ,\ g_1(t,T),\ldots,g_n(t,T)\}$$

une représentation univariée rationnelle de $\mathcal{I}$. L'application

$$\begin{array}{llll}\gamma^\star: & V(f) & \longrightarrow & V(\mathcal{I})\\ & \beta & \longmapsto & 1/g(t,\beta)\cdot(g_1(t,\beta),\ldots,g_n(t,\beta))\end{array}$$

définit un K-isomorphisme de $V(f)$ dans $V(\mathcal{I})$ et $(f,\gamma^\star)$ est une μ-résolution de $\mathcal{I}$.

Preuve : D'après le lemme 3.1.1, le polynôme $g(t,T)$ est premier avec $f(t,T)$. Il existe donc des polynômes $u(T)$ et $v(T)$ tels que

$$u(T)g(t,T)+v(T)f(t,T)=1=u(t)g(t,t)+v(t)f(t,t)$$

D'après le lemme 3.1.1, $f(t,T)$ est le polynôme caractéristique de m_t. En appliquant le théorème de Cayley, on peut voir que $f(t,t)\in\mathcal{I}$. Le polynôme $g(t,t)$ est donc inversible modulo $\mathcal{I}$ d'inverse $u(t)$.

En posant, pour tout $i = 1, \ldots, n$, $u_i(t,T) = u(T)g_i(t,T)$, on définit deux morphismes de K-algèbres :

$$\begin{array}{rlcl} \phi : & K[X_1, \ldots, X_n] & \longrightarrow & K[T] \\ & X_i & \longmapsto & u_i(t,T)\ ,\ i = 1 \ldots n \\ \psi : & K[T] & \longrightarrow & K[X_1, \ldots, X_n] \\ & T & \longmapsto & t \end{array}$$

et on note $\mathcal{J}$ l'idéal de $K[T]$ engendré par $f(t,T)$.
Pour tout $\alpha \in V(\mathcal{I})$, $\phi^\star(\psi^\star)(\alpha) = \phi^\star(t(\alpha)) = (u_1(t,t(\alpha)), \ldots, u_n(t,t(\alpha)))$. Par construction des polynômes $u_1, \ldots, u_n$, $\phi^\star(\psi^\star)(\alpha) = \alpha$ et donc $\phi^\star \circ \psi^\star|_{V(\mathcal{I})} = Id_{V(\mathcal{I})}$.
De la même façon, pour tout $\beta \in V(\mathcal{J})$, il existe $\alpha \in V(\mathcal{I})$ tel que $\beta = t(\alpha)$. Ainsi,

$$\psi^\star(\phi^\star)(\beta) = t(u_1, t(\alpha), \ldots, u_n(t, t(\alpha))) = t(\alpha) = \beta,$$

et donc $\psi^\star \circ \phi^\star|_{V(\mathcal{J})} = Id_{V(\mathcal{J})}$.
Comme $\phi^\star$ est un K-isomorphisme de $V(f)$ dans $V(\mathcal{I})$, coincidant avec $\gamma^\star$ sur $V(\mathcal{I})$, l'application $\gamma^\star$ est donc un K-isomorphisme de $V(f)$ dans $V(\mathcal{I})$.
Enfin, d'après la proposition 3.1.4,

$$\forall \alpha \in \mathcal{I}\ ,\ (\gamma^\star)^{-1}(\alpha) = t(\alpha)$$

et

$$\mu(\alpha) = \mu(t(\alpha)) = \mu((\gamma^\star)^{-1}(\alpha)),$$

$(\mathcal{J}, \gamma^\star)$ est par conséquent une μ-résolution de $\mathcal{I}$.

□

Definition 3.1.4 *En reprenant les notations des propositions 3.1.4 et 3.1.5, nous dirons que $(f, \gamma^\star)$ est la* μ-résolution induite par la représentation univariée rationnelle de $\mathcal{I}$ associée à t.

Nous avons indroduit un procédé (représentation univariée rationnelle) de construction pour définir une μ-résolution. Montrons maintenant que toute μ-résolution peut s'exprimer par la donnée d'une représentation univariée rationnelle :

Proposition 3.1.6 *Soit $(\langle g(T)\rangle, \phi^\star)$ une μ-résolution d'un idéal zéro dimensionnel $\mathcal{I} \subset K[X_1, \ldots, X_n]$. Le polynôme $g(T)$ est alors le polynôme caractéristique d'un élément $t \in K[X_1, \ldots, X_n]$ séparant $V(\mathcal{I})$. Si de plus $(\mathcal{J}', \gamma^\star)$ est la μ-résolution induite par la représentation univariée rationnelle de $\mathcal{I}$ associée à t, $\mathcal{J}' = \langle g(T)\rangle$ et $\gamma^\star$ coincide avec $\phi^\star$ sur $V(g)$.*

Preuve : Notons $\phi : K[X_1, \ldots, X_n] \longrightarrow K[T]$ un morphisme de K-algèbres vérifiant les hypothèses de la définition 3.1.2.
D'après la proposition 3.1.3 ϕ se factorise en un isomorphisme ϕ_{red} de $A_{red} = K[X_1, \ldots, X_n]/\sqrt{\mathcal{I}}$ dans $B_{red} = K[T]/\sqrt{\langle g(T) \rangle}$ et il existe alors $t \in K[X_1, \ldots, X_n]$ tel que $\phi_{red}(t) = T$. En particulier, $\phi(t) = T + q(T)$ qvec $q(T) \in \sqrt{\langle g(T) \rangle}$.

Notons P_t (resp. P_T) le polynôme caractéristique de m_t (resp. m_T) dans $A = K[X_1, \ldots, X_n]/\mathcal{I}$ (resp. $B = K[T]/\langle g(T) \rangle$). Comme $dim_K(A) = dim_K(B)$, P_t et P_T ont mêmes degrés. Les sommes de Newton élémentaires de P_t s'écrivent, d'après le théorème de Stickelberger :

$$N_i(P_t) = \sum_{\alpha \in V(\mathcal{I})} \mu(\alpha) t(\alpha)^i.$$

De même,

$$N_i(P_T) = \sum_{\beta \in V(g)} \mu(\beta) \beta^i = \sum_{\beta \in V(g)} \mu(\beta)(\phi(t)(\beta))^i.$$

D'après la proposition 3.1.2, nous avons alors $N_i(P_t) = N_i(P_T)$ pour tout $i = 1 \ldots D$ où $D = \sum_{\alpha \in V(\mathcal{I})} \mu(\alpha) = \sum_{\beta \in V(g)} \mu(\beta)$, ce qui montre en particulier que P_t et P_T ont mêmes fonctions symétriques élémentaires et par conséquent mêmes coefficients.
Le polynôme P_T étant le polynôme caractéristique de m_T, $P_T(T) \in \langle g(T) \rangle$ et donc $g(T)$ divise $P_T(T)$. Comme de plus $g(T)$ et P_T sont de mêmes degrés, il sont égaux (à multiplication par un scalaire près).
Supposons maintenant que t ne soit pas un élément séparant $V(\mathcal{I})$. Dans ce cas , le nombre de racines distinctes de $P_t = \sum_{\alpha \in V(\mathcal{I})} (T - t(\alpha))^{\mu(\alpha)}$ est strictement inférieur à $\sharp V(\mathcal{I})$ et par conséquent, le nombre de racines de $P_T = \sum_{\beta \in V(g)} (T - \beta)^{\mu(\beta)}$ est strictement inférieur à $\sharp V(g) = \sharp V(\mathcal{I})$, ce qui est impossible car $g(T)$ divise P_T.
En reprenant les notations de l'énoncé, ceci prouve en particulier que, par construction de la représentation univariée rationnelle , $\mathcal{J}' = \langle g(T) \rangle$ et que, en utilisant la remarque 3.1.2, $\gamma^\star$ coincide avec $\phi^\star$ sur $V(g)$. □

3.2 Propriétés de la Représentation Univariée Rationnelle

Nous avons vu dans la partie précédente que la représentation univariée rationnelle permet de réduire l'étude d'un système zéro-dimensionnel à celle d'un polynôme en une variable. Nous montrons maintenant que la μ-résolution permet de préserver le caractère *réel* des racines.

Proposition 3.2.1 *Nous supposons que K est un corps ordonné. Soient $\mathcal{I} \subset K[X_1, \ldots, X_n]$ un idéal zéro-dimensionnel, $(\mathcal{J} \subset K[T], \phi^\star)$ une μ-résolution de $\mathcal{I}$ et R la clôture réelle de K. Alors, $\alpha \in V(\mathcal{I}) \bigcap R^n$ si et seulement si $(\phi^\star)^{-1}(\alpha) \in R$.*

Preuve : D'après la proposition 3.1.6, nous pouvons supposer que $(\mathcal{J}, \phi^\star)$ est la μ-résolution induite par la représentation univariée rationnelle associée à un polynôme $t \in K[X_1, \ldots, X_n]$ séparant $V(\mathcal{I})$. Par conséquent, toujours d'après la proposition 3.1.6, $(\phi^\star)^{-1}$ coincide sur $V(\mathcal{I})$ avec le K-morphisme $\psi^\star$ déduit du morphisme d'algèbres :

$$\begin{array}{llll} \psi : & K[T] & \longrightarrow & K[X_1, \ldots, X_n] \\ & T & \longmapsto & t \end{array}$$

Ainsi, $\forall \alpha \in V(\mathcal{I})$, $(\phi^\star)^{-1}(\alpha) = t(\alpha) \in V(\mathcal{J})$.
Comme $t \in K[X_1, \ldots, X_n]$, $\forall \alpha \in V(\mathcal{I}) \bigcap R^n$, $t(\alpha) \in R \bigcap V(\mathcal{J})$.
Inversement, supposons que $t(\alpha) \in R \bigcap V(\mathcal{J})$, pour $\alpha \in V(\mathcal{I})$ donné. Si α est un zéro de $\mathcal{I}$, son expression conjuguée $\bar{\alpha}$ est également un zéro de $\mathcal{I}$. Comme $t(\alpha) \in R$, $t(\alpha) = \overline{t(\alpha)} = t(\bar{\alpha})$. Comme de plus t sépare $\mathcal{I}$, $\bar{\alpha} = \alpha$ et donc $\alpha \in R^n$. □

Ce résultat est d'une grande importance puisqu'il permet l'utilisation des nombreux outils réservés à l'étude des polynômes en une variable pour compter, isoler où encore coder *à la Thom* les racines réelles d'un système zéro-dimensionnel, comme nous le verrons dans le prochain chapitre.

Par les algorithmes existants et en particulier les bases de Groebner, le calcul du radical d'un idéal n'est pas une opération facile en pratique. Nous montrons, dans la fin de cette partie, que le calcul d'une μ-résolution du radical $\sqrt{\mathcal{I}}$ d'un idéal zéro-dimensionnel n'est pas plus difficile que celui d'une μ-résolution de $\mathcal{I}$, puis nous établissons le lien, dans le cas particulier d'idéaux radicaux en position générique, entre la représentation univariée rationnelle et une base de Groebner pour l'ordre lexicographique.

Definition 3.2.1 *Un ideal $\mathcal{I} \subset K[X_1, \ldots, X_n]$ est en* position générique *si X_1 est un polynôme séparant $V(\mathcal{I})$.*

Proposition 3.2.2 *Soit $\mathcal{I} \subset K[X_1, \ldots, X_n]$ un idéal zéro-dimensionnel. Les conditions suivantes sont équivalentes :*

- *(1) $\mathcal{I}$ est radical.*
- *(2) Pour toute μ-résolution $(h(T), \phi^\star)$ de $\mathcal{I}$, h est sans facteurs carrés.*

En particulier, si $\mathcal{I}$ est radical et si $(h(T), \phi^\star)$ est une μ-résolution de $\mathcal{I}$, $K[X_1, \ldots, X_n]/\mathcal{I}$ est isomorphe à $K[T]/h$.

Preuve : Sans perte de généralité, nous pouvons supposer que X_1 est un polynôme séparant $V(\mathcal{I})$. En effet , d'après le lemme 1.2.1, il existe un changement linéaire de variables tel que $\mathcal{I}$ soit en position générique.
(1) $\Rightarrow$ (2) : Soit $(h(T), \phi^\star)$ une μ-résolution de $\mathcal{I}$. Si $\mathcal{I}$ est radical, d'après la proposition 3.1.3, $K[X_1, \ldots, X_n]/\mathcal{I}$ est isomorphe à $K[T]/\sqrt{\langle h(T)\rangle}$. Comme de plus

$$dim_K(K[X_1, \ldots, X_n]/\mathcal{I}) = \sum_{\alpha \in V(\mathcal{I})} \mu(\alpha) = \sum_{\beta \in V(h)} \mu(\beta) = dim_K(K[T]/\langle h(T)\rangle$$

alors $\sqrt{\langle h(T)\rangle} = \langle h(T)\rangle$ et $h(T)$ est donc sans facteurs carrés.
(2) $\Rightarrow$ (1) : Soit $(f(T), \phi^\star)$ la μ-résolution induite par la représentation univariée rationnelle associée à X_1. Rappelons que dans ce cas, $f(T)$ est le polynôme caractéristique de l'application m_{X_1} et, en particulier, si d est le degré de f, $d = dim_K(K[X_1, \ldots, X_n]/\mathcal{I})$. Comme f est supposé être sans facteurs carrés, f est également le polynôme minimal de m_{X_1}, ce qui montre que $\{1, X_1, \ldots, X_1^{d-1}\}$ est une une base de $K[X_1, \ldots, X_n]/\mathcal{I}$. D'après le théorème de Cayley, $f(X_1) \in \mathcal{I}$, donc, le morphisme de K-algèbres de $K[X_1, \ldots, X_n]/\mathcal{I}$ dans $K[T]/f$ qui à X_1 associe T est un isomorphisme. L'idéal $\mathcal{I}$ est donc radical.
La dernière assertion de la proposition est une application directe de la proposition 3.1.3. □

Ceci nous donne alors un moyen simple de trouver une μ-résolution du radical d'un idéal zéro-dimensionnel donné :

Corollaire 3.2.1 *Soit $\mathcal{I} \subset K[X_1, \ldots, X_n]$ un idéal zéro-dimensionnel. Si $(f, \phi^\star)$ une μ-résolution de $\mathcal{I}$, alors $(f/gcd(f, f'), \phi^\star)$ est une μ-résolution de $\sqrt{\mathcal{I}}$.*

Preuve : D'après la proposition 3.1.6, il existe un polynôme t séparant $V(\mathcal{I})$ tel que $(f, \phi^\star)$ soit la μ-résolution induite par la représentation univariée rationnelle de $\mathcal{I}$ associée à t. En remarquant que t sépare également $V(\sqrt{\mathcal{I}})$, si $(g, \psi^\star)$ est la μ-résolution induite par la représentation univariée rationnelle de $\sqrt{\mathcal{I}}$ associée à t, g est sans facteurs carrés d'après la proposition précédente et g à mêmes racines que f (les $t(\alpha)$, $\alpha \in V(\mathcal{I})$) par construction de la représentation univariée rationnelle . Par conséquent g est égal (à multiplication par un scalaire près) à la partie sans facteurs carrés de f. □

Remarquons en particulier que le calcul d'une représentation univariée rationnelle du radical d'un idéal zéro-dimensionnel $\mathcal{I}$ est immédiat si une représentation univariée rationnelle de $\mathcal{I}$ est connue (le calcul de $f/gcd(f, f')$ est en effet nécessaire pour calculer la représentation univariée rationnelle de $\mathcal{I}$).
Les bases de Groebner lexicographiques sont souvent considérées comme étant une *résolution* des systèmes zéro-dimensionnels. Dans le cas général, aucune relation

simple n'apparait entre la représentation univariée rationnelle et les bases de Groebner lexicographiques. Toutefois, nous allons montrer que dans le cas particulier d'un idéal radical en position générique, la représentation univariée rationnelle permet de retrouver une base de Groebner lexicographique.
Rappelons deux résultats classiques concernant les bases de Groebner lexicographiques (voir [BMMT93],[Fau94],[GM89]):

Proposition 3.2.3 *Soit $\mathcal{I} \subset K[X_1, \ldots, X_n]$ un idéal zéro dimensionnel en position générique. Les conditions suivantes sont équivalentes :*

- (1) *: $\mathcal{I}$ est radical.*
- (2) *: $\exists f \in K[X_1]$, $\forall i = 1, \ldots, n \exists f_i \in K[X_1]$ tels que :*
  - *f est sans facteurs carrés.*
  - $\forall i = 1, \ldots, n$, $deg(f_i) < deg(f)$
  - $\mathcal{I} = \{f(X_1), X_2 - f_2, \ldots, X_n - f_n\}$

Preuve :

- (2) $\Rightarrow$ (1) : Comme $\mathcal{I} = \langle f, X_1 - u_1, \ldots, X_n - u_n \rangle$, alors $K[X_1, \ldots, X_n]/\mathcal{I}$ est isomorphe à $K[X_1]/f$ et donc si f est radical, $\mathcal{I}$ l'est aussi.
- (1) $\Rightarrow$ (2) : Soit f un générateur de $\mathcal{I} \bigcap K[X_1]$. L'idéal $\mathcal{I}$ étant radical, $\mathcal{I} \bigcap K[X_1]$ l'est donc également et f est par conséquent sans facteurs carrés. Comme $\mathcal{I}$ est en position générique, $dim_K(K[X_1]/f) = dim_K(K[X_1, \ldots, X_n]/\mathcal{I})$. Ainsi, si d est le degré de f en X_1, $1, X_1, \ldots, X_1^{d-1}$ est une base de $K[X_1, \ldots, X_n]/\mathcal{I}$. Il ne reste plus alors qu'à exprimer les coordonnées de $X_2, \ldots, X_n$ dans cette nouvelle base pour conclure.

$\square$

Corollaire 3.2.2 *Soit $\mathcal{I} \subset K[X_1, \ldots, X_n]$ un idéal zéro dimensionnel en position générique. Les conditions suivantes sont équivalentes :*

- (1) *: $\mathcal{I}$ est radical.*
- (2) *: Sa base de Groebner réduite pour l'ordre lexicographique est donnée par*

  $$\{f(X_1), X_2 - f_2(X_1), \ldots, X_n - f_n(X_1)\}$$

  avec f sans facteurs carrés et $X_1 < X_2 < \ldots < X_n$.

Preuve : L'ensemble $\{f(X_1), X_1 - f_1(X_1), \ldots, X_n - f_n(X_1)\}$ est clairement une base de Groebner pour l'ordre lexicographique avec $X_1 < X_2 < \ldots < X_n$. Elle est réduite par définition ($deg(f_i) < deg(f)$) et d'après la proposition précédente, c'est un système de générateurs de $\mathcal{I}$. □

Etablissons maintenant la relation entre représentation univariée rationnelle et base de Groebner lexicographique :

Corollaire 3.2.3 *Soit $\mathcal{I} \subset K[X_1, \ldots, X_n]$ un idéal zéro-dimensionnel radical en position générique de base de Groebner lexicographique $\{f(X_1), X_2 - f_2(X_1), \ldots, X_n - f_n(X_1)\}$ ($X_1 < \ldots < X_n$) et dont la représentation univariée rationnelle associée à X_1 s'écrit $\{h(X_1, X_1)\ ,\ g(X_1, X_1)\ ,\ g_1(X_1, X_1), \ldots, g_n(X_1, X_1)\}$. Alors :*

- $f = h$
- $f' = g$
- $g(X_1)$ *est inversible modulo* $\mathcal{I}$ *et* $\forall i = 2 \ldots n\ f_i = g^{-1} g_i\ mod\ f$

Preuve : Par construction, f est le polynôme caractéristique de m_{X_1} dans $K[X_1, \ldots, X_n]/\mathcal{I}$. Il coincide donc avec le premier polynôme de la représentation univariée rationnelle de $\mathcal{I}$ associée à X_1. Comme f est sans facteurs carrés, nous avons également $f' = g$. Par conséquent

$$g f_i = g_i\ mod\ \mathcal{I},$$

car d'une part, $\forall \alpha \in V(\mathcal{I})\ ,\ X_i(\alpha) = f_i(X_1(\alpha))$ et d'autre part $\forall \alpha \in V(\mathcal{I})\ ,\ X_i(\alpha) = g_i(X_1, X_1(\alpha))/g(X_1, X_1(\alpha))$. Comme $g = f'$ est inversible modulo f et donc modulo $\mathcal{I}$ (voir preuve du théorème 3.1.1), $f_i = g^{-1} g_i\ mod\ \mathcal{I}$. Comme $f, g, g_2, \ldots, g_n, f_2, \ldots, f_n \subset K[X_1]$ alors

$$f_i = g^{-1} g_i\ mod\ (\mathcal{I} \bigcap K[X_1])\ , \forall i = 2 \ldots n$$

$\{f, f_2, \ldots, f_n\}$ étant une base de Groebner, il vient alors :

$$f_i = g^{-1} g_i\ mod\ f\ , \forall i = 2 \ldots n$$

□

3.3 Fonctions symétriques étendues et calcul de représentation univariée rationnelle

Pour un idéal zéro-dimensionnel $\mathcal{I} \subset K[X_1, \ldots, X_n]$ donné, nous supposerons connue, dans cette section, une table de multiplication MT de $K[X_1, \ldots, X_n]/\mathcal{I}$, c'est à dire l'ensemble des produits $\overrightarrow{\omega_i \omega_j}, \overrightarrow{X_k \omega_j} \in K[X_1, \ldots, X_n]/\mathcal{I}$ où $\mathcal{B} = \{\omega_1, \ldots, \omega_D\}$ est une base de $K[X_1, \ldots, X_n]/\mathcal{I}$ donnée.

Notation 3.3.1 *En reprenant les notations 3.1.1, on pose, pour tout* $i = 1, \ldots, n$ *et pour tout* $t \in K[X_1, \ldots, X_n]$ *:*

$$f(t,T) = \prod_{\alpha \in V(\mathcal{I})} (T - t(\alpha))^{\mu(\alpha)}$$

$$g_i(t,T) = g(X_i,t,T) = \sum_{\alpha \in V(\mathcal{I})} \mu(\alpha) X_i(\alpha) \prod_{y \in t(V(\mathcal{I})) \setminus \{t(\alpha)\}} (T - y)$$

$$g(t,T) = g(1,t,T)$$

D'après le lemme 3.1.1, ces polynômes sont dans $K[T]$ et rappelons que, lorsque t sépare $V(\mathcal{I})$, $\{f(t,T), g(t,T), g_1(t,T), \ldots, g_n(t,T)\}$ est la représentation univariée rationnelle de $\mathcal{I}$ associée à t (proposition 3.1.4).

L'objet de ce qui suit est de fournir un algorithme efficace pour calculer la famille de polynômes $\{f(t,T), g_i(t,T)\ ,\ i = 1 \ldots n\}$, pour un polynôme t donné.

Une méthode de calcul est donnée par le lemme 3.1.1. D'un point de vue pratique, hormis la difficulté de calculer le polynôme caractéristique d'une matrice à coefficients polynomiaux, un nombre important d'informations inutiles sont fournies par cette méthode. En effet seuls les termes de degré 1 en S des polynômes $det(M_{t+SX_i} - TI) \in K[S,T]$ sont réellement utilisés alors que tous les termes sont calculés.

L'idée est d'effectuer ce calcul à l'aide d'outils combinatoires pour utiliser, autant que possible, les informations contenues dans MT. Nous étendons les notions classiques de sommes de Newton et de fonctions symétriques élémentaires aux ensembles de points à plusieurs coordonnées et nous établissons alors une formule généralisant la relation de Newton.

La considération de fonctions symétriques et de sommes de Newton à plusieurs coordonnées est déjà présente dans [J93] et plus récemment dans [Dal95] et [RRS94] où des formules très générales sont proposées. Dans l'esprit de [GVT95], nous présentons ici des résultats particuliers, adaptés au calcul de la représentation univariée rationnelle .

3.3.1 Utilisation de fonctions symétriques étendues

D'après le théorème de Stickelberger, pour $h \in K[X_1, \ldots X_n]$ fixé, les valeurs propres de l'endomorphisme $m_{h^j}\ ,\ j \geq 0$ sont exactement les $h(\alpha)^j\ ,\ \alpha \in V(\mathcal{I})$ avec pour multiplicités respectives $\mu(\alpha)$. Par conséquent, les sommes de Newton associées au polynôme $f(t,T)$ peuvent être facilement obtenues si l'on connaît les traces : $Trace(m_{t^i})$. Ainsi, on peut déduire facilement, par la formule de Newton, les coefficients de $f(t,T)$ à partir de ces traces.

Le but de cette partie, est de montrer que l'on peut étendre ce processus pour le calcul des polynômes $g_1(t,T), \ldots, g_n(t,T)$ (voir notation 3.3.1).
Nous aurons besoin, dans ce qui suit de la notion de multi-ensemble:

Definition 3.3.1 *Un multi-ensemble $\mathcal{Y} = (E(\mathcal{Y}), \mu_{\mathcal{Y}})$ est la donnée d'un ensemble fini $E(\mathcal{Y})$ et d'une application $\mu_{\mathcal{Y}}$ de $E(\mathcal{Y})$ dans $\mathbb{N}$.*
La cardinalité d'un multi-ensemble $\mathcal{Y}$ est alors définie par : $\sharp\mathcal{Y} = \sum_{y \in E(\mathcal{Y})} \mu_{\mathcal{Y}}(y)$.
Un multi-ensemble $\mathcal{X}$ est inclus dans $\mathcal{Y}$ si:

- $E(\mathcal{X}) \subset E(\mathcal{Y})$
- *Pour tout* $x \in \mathcal{X}$, $\mu_{\mathcal{X}}(x) \leq \mu_{\mathcal{Y}}(x)$.

Pour tout entier i, $\mathcal{P}_i(\mathcal{Y})$ est l'ensemble des multi-ensembles de cardinal i contenus dans $\mathcal{Y}$ et $\mathcal{Y}_y$ est le multi-ensemble défini par $(E(\mathcal{Y}), \mu_{\mathcal{Y}_y})$, avec $\mu_{\mathcal{Y}_y}(z) = \mu_{\mathcal{Y}}(z)$ si $z \neq y$, et $\mu_{\mathcal{Y}_y}(y) = max(\mu_{\mathcal{Y}}(y) - 1, 0)$.
Enfin, pour toute famille $\{f_y \ , \ y \in E(\mathcal{Y})\}$ d'éléments de C^n on notera :

$$\sum_{y \in \mathcal{Y}} f_y = \sum_{y \in E(\mathcal{Y})} \mu_{\mathcal{Y}}(y) f_y \quad et \quad \prod_{y \in \mathcal{Y}} f_y = \prod_{y \in E(\mathcal{Y})} f_y^{\mu_{\mathcal{Y}}(y)}.$$

Definition 3.3.2 *Soient t un polynôme de $K[X_1, \ldots, X_n]$ et $\mathcal{Y}$ un multi-ensemble de C^n. Nous noterons :*

- $S_i(t, \mathcal{Y}) = \sum_{\mathcal{Z} \in \mathcal{P}_i(\mathcal{Y})} \prod_{y \in \mathcal{Z}} t(y)$ *la i^e fonction symétrique élémentaire associée au multi-ensemble* $(\{t(\beta), \beta \in E(\mathcal{Y})\}, \mu_{t(\beta)} : t(\beta) \longmapsto \sum_{\substack{\gamma \in E(\mathcal{Y}) \\ t(\gamma) = t(\beta)}} \mu_{\mathcal{Y}}(\gamma))$.
- $N_i(t, \mathcal{Y}) = \sum_{y \in \mathcal{Y}} t(y)^i$ *la i^esomme de Newton associée au multi-ensemble* $(\{t(\beta), \beta \in E(\mathcal{Y})\}, \mu_{t(\beta)})$.

Pour les besoins de notre algorithme, nous étendons ces définitions classiques :
Étant donnés des polynômes t et $v \in K[X_1, \ldots, X_n]$ et un multi-ensemble $\mathcal{Y} \in C^n$ on définit :

- $S_i(v, t, \mathcal{Y}) = \sum_{y \in \mathcal{Y}} v(y) S_i(t, \mathcal{Y}_y)$
- $N_i(v, t, \mathcal{Y}) = \sum_{y \in \mathcal{Y}} v(y) t(y)^i$

En développant $f(t, T)$ et $h(v, t, T)$ (notations du lemme 3.1.1) nous pouvons remarquer qu'il est possible d'exprimer leurs coefficients en fonction des fonctions symétriques étendues introduites plus haut :

$$f(t, T) = \sum_{i=0}^{D} (-1)^i S_i(t, \mathcal{Y}) T^{D-i}$$

$$h(v,t,T) = \sum_{i=0}^{D-1} (-1)^i S_i(v,t,\mathcal{Y}) T^{D-i-1}$$

La formule classique de Newton reliant sommes de Newton et fonctions symétriques élémentaires s'écrit :

$$(D-i) S_i(t,\mathcal{Y}) = \sum_{j=0}^{i} (-1)^j N_j(t,\mathcal{Y}) S_{i-j}(t,\mathcal{Y})$$

avec la convention $S_0(t,\mathcal{Y}) = 1$. En utilisant la définition 3.3.2, la formule établie dans la proposition suivante en est une généralisation :

Proposition 3.3.1

$$S_i(v,t,\mathcal{Y}) = \sum_{j=0}^{i} (-1)^j N_j(v,t,\mathcal{Y}) S_{i-j}(t,\mathcal{Y})$$

Pour prouver ce résultat, nous utiliserons le lemme suivant :

Lemme 3.3.1 *Pour* $0 \leq k < i$,

$$\begin{aligned}\sum_{y\in\mathcal{Y}} v(y)t(y)^k S_{i-k}(t,\mathcal{Y}_y) = \quad & N_k(v,t,\mathcal{Y}) S_{i-k}(t,\mathcal{Y}) - \\ & \sum_{y\in\mathcal{Y}} v(y)t(y)^{k+1} S_{i-k-1}(t,\mathcal{Y}_y)\end{aligned}$$

Preuve du lemme : Pour tout entier k, $0 \leq k < i$,

$$N_k(v,t,\mathcal{Y}) S_{i-k}(t,\mathcal{Y}) = \left(\sum_{y\in\mathcal{Y}} v(y)t(y)^k \right) \left(\sum_{I\in\mathcal{P}_{i-k}(\mathcal{Y})} \prod_{z\in I} t(z) \right)$$

$$= \sum_{y\in\mathcal{Y}} \sum_{I\in\mathcal{P}_{i-k}(\mathcal{Y})} v(y)t(y)^k \textstyle\prod_{z\in I} t(z)$$

$$= \sum_{y\in\mathcal{Y}} \left(\sum_{I\in\mathcal{P}_{i-k-1}(\mathcal{Y})} v(y)t(y)^k \prod_{z\in I_y} t(z) + \sum_{I\in\mathcal{P}_{i-k-1}(\mathcal{Y})} v(y)t(y)^{k+1} \prod_{z\in I_y} t(z) \right)$$

$$= \sum_{y\in\mathcal{Y}} v(y)t(y)^k S_{i-k}(t,\mathcal{Y}_y) + \sum_{y\in\mathcal{Y}} v(y)t(y)^{k+1} S_{i-k-1}(t,\mathcal{Y}_y)$$

□

Preuve de la proposition 3.3.1 : D'après le lemme 3.3.1, nous avons :

$$S_i(v,t,\mathcal{Y}) = \sum_{y\in\mathcal{Y}} v(y) S_i(t,\mathcal{Y}_y) = N_0(v,t,\mathcal{Y}) S_i(t,\mathcal{Y}) - \sum_{y\in\mathcal{Y}} v(y)t(y) S_{i-1}(t,\mathcal{Y}_y)$$

En utilisant le même argument, il vient alors par induction :

$$S_i(v,t,\mathcal{Y}) = \sum_{j=0}^{i-1}(-1)^j N_j(v,t,\mathcal{Y})S_{i-j}(t,\mathcal{Y}) + (-1)^i \sum_{y\in\mathcal{Y}} v(y)t(y)^i S_0(t,\mathcal{Y}_y)$$

Comme $S_0(t,\mathcal{Y}_y) = 1$ le résultat est prouvé. □

D'après le théorème de Stickelberger,

$$N_i(v,t,\mathcal{Y}) = Trace(m_{vt^i})$$

$$N_i(t,\mathcal{Y}) = Trace(m_{t^i})$$

ce qui amène alors naturellement la proposition :

Proposition 3.3.2 *Soient $S = \{f_1,\ldots,f_s\}$ un système zéro-dimensionnel de $K[X_1,\ldots,X_n]$, D la dimension du K-espace vectoriel $K[X_1,\ldots,X_n]/\langle\mathcal{S}\rangle$ et $t \in [X_1,\ldots,X_n]$.*
Il est possible de calculer les polynômes (notations du lemme 3.1.1)

$$f(t,T), h(X_1,t,T), \ldots h(X_n,t,T)$$

à partir des traces :

$$Trace(m_{X_j t^i})\ ,\ i = 1\ldots D-1\ , j = 1,\ldots n$$

$$Trace(m_{t^i})\ ,\ i = 1\ldots D$$

en résolvant n systèmes linéaires triangulaires.
D'après le lemme 3.1.1, en divisant les polynômes $h(X_i,t,T)$ ainsi que $f'(t,T)$ par $gcd(f,f')$, nous obtenons alors les polynômes de $K[T]$:

$$f(t,T)\ ,\ g(t,T)\ ,\ g_1(t,T), \ldots g_n(t,T)$$

définissant, lorsque t est séparant, la représentation univariée rationnelle de $\mathcal{I}$ associée à t (proposition 3.1.4).

3.3.2 Calcul des sommes de Newton étendues

Soit $\mathcal{I} \subset K[X_1,\ldots,X_n]$ un Idéal zéro-dimensionnel. D'après la partie précédente, il existe un moyen simple et efficace de calculer les polynômes $f(t,T), h(v,t,T), g(v,t,T)$ définis dans le lemme 3.1.1, pour $t, v \in K[X_1,\ldots,X_n]$ et ainsi, lorsque t sépare $V(\mathcal{I})$, de calculer la représentation univariée rationnelle de $\mathcal{I}$ associée à t (proposition 3.1.4), à partir des traces : $Trace(m_{X_j t^i})$, $i = 1,\ldots,D-1$, $j = 1,\ldots,n$ et $Trace(m_{t^i})$, $i = 1,\ldots,D$.

Pour une base $\mathcal{B} = \{(\omega_1 \ldots \omega_D\}$ de $\mathcal{A} = K[X_1, \ldots, X_n]/\mathcal{I}$ donnée, en supposant que nous disposions d'une table contenant tous les produits $\overrightarrow{\omega_i \omega_j}$, il est possible de calculer simplement toutes les traces de la forme $Trace(m_{X_j t^i})$, en utilisant le lemme 1.2.3 .
Pour tout $h \in K[X_1, \ldots X_n]/\mathcal{I}$, notons M_h la matrice de m_h dans la base $\mathcal{B}$.
D'après le lemme 1.2.3, en posant $Vtr(1) = [Trace(m_{\omega_1}), \ldots, Trace(m_{\omega_D})]$, pour tout $i = 1 \ldots D$ et tout $j = 1 \ldots n$ on peut écrire :

$$Trace(m_{t^i}) = \overrightarrow{t^i} \cdot Vtr(1) = (M_t \cdot \overrightarrow{t^{i-1}}) \cdot Vtr$$

$$Trace(m_{X_j t^i}) = \overrightarrow{X_j t^i} \cdot Vtr(1) = (M_{X_j} \cdot \overrightarrow{t^i}) \cdot Vtr(1)$$

Ainsi, connaissant $Vtr(1)$, un algorithme simple s'écrira :

Algorithme TracesToExtNewton-0

Entrée: $Vtr(1)$, M_t, $M_{X_1}, \ldots, M_{X_n}$.

- **Etape 1 :** $t^0 = 1$
- **Etape 2 :** Pour $i = 0 \ldots (D-1)$,
  - $Trace(m_{t^i}) = \overrightarrow{t^i} \cdot Vtr(1)$
  - Pour $j = 1 \ldots n$, $Trace(m_{X_j t^i}) = (M_{X_j} \cdot \overrightarrow{t^i}) \cdot Vtr(1)$
  - $\overrightarrow{t^{i+1}} = M_t \, \overrightarrow{t^i}$
- **Etape 3 :** $Trace(m_{t^D}) = \overrightarrow{t^D} Vtr(1)$

Sortie: $Trace(m_{t^i})$, $i = 0 \ldots D$ et $Trace(m_{X_j t^i})$, $i = 0 \ldots (D-1)$, $j = 1 \ldots n$

Cette version naïve de l'algorithme entraîne un nombre d'opérations arithmétiques en $O(n \cdot D^3)$. Il est toutefois possible de combiner plus astucieusement les propriétés de linéarité de l'application $Trace$, pour réduire de façon significative le nombre d'opérations arithmétiques nécessaires pour obtenir un méthode en $O(D^3 + n \cdot D^2)$. En effet, pour tout $i = 1 \ldots (D-1)$ et tout $j = 1 \ldots n$,

$$Trace(m_{t^{i+1}}) = \overrightarrow{t} \cdot Vtr(t^i)$$

$$Trace(m_{X_j t^i}) = \overrightarrow{X_j} \cdot Vtr(t^i)$$

où $Vtr(t^i) = [Trace(m_{t^i\omega_1}), \ldots, Trace(m_{t^i\omega_D})]$. Pour i fixé, il est donc possible de calculer simultanément $Trace(m_{t^{i+1}})$ et $Trace(m_{X_j t^i})$, $j = 1 \ldots n$, dès lors que $Vtr(t^i)$ est connu.
D'après le lemme 1.2.3,

$$Trace(m_{t^i\omega_k}) = \overrightarrow{t\omega_k} \cdot Vtr(t^{i-1})$$

Donc, si $(M_t)^T$ est la transposée de M_t, nous aurons :

$$Vtr(t^i) = (M_t)^T \cdot Vtr(t^{i-1})$$

Ces diverses relations nous donnent alors un algorithme pour obtenir les sommes de Newton étendues nécessaires au calcul des polynômes

$$\{f(t,T), h(X_1,t,T), \ldots, h(X_1,t,T)\},$$

avec $t \in K[X_1, \ldots, X_n]$ quelconque, définis dans le lemme 3.1.1 :

Algorithme TracesToExtNewton

Entrée: $Vtr(1)$, M_t, $M_{X_1}, \ldots, M_{X_n}$.

- **Etape 1 :** $Trace(m_{t^0}) = D$, $Vtr(t^0) = Vtr(1)$
- **Etape 2 :** Pour $i = 0 \ldots (D-1)$,
  - $Trace(m_{t^{i+1}}) = \overrightarrow{t} \cdot Vtr(t^i)$
  - Pour $j = 1 \ldots n$, $Trace(m_{X_j t^i}) = \overrightarrow{X_j} \cdot Vtr(t^i)$
  - $Vtr(t^{i+1}) = (M_t)^T \cdot Vtr(t^i)$

Sortie: $Trace(m_{t^i})$, $i = 0 \ldots D$ et $Trace(m_{X_j t^i})$, $i = 0 \ldots (D-1)$, $j = 1 \ldots n$

3.3.3 Le problème de l'élément séparant

Dans cette section, nous proposons un algorithme complet pour le calcul d'une représentation univariée rationnelle d'un idéal zéro-dimensionnel. Nous avons vu dans la section précédente comment calculer efficacement les divers polynômes définis dans le lemme 3.1.1. Une représentation univariée rationnelle pourra alors être calculée dès lors qu'un élément séparant est connu (proposition 3.1.4).

Il n'apparaît pas raisonnable d'espérer trouver un polynôme séparant sans aucun calcul. Nous avons pu voir que l'ensemble des formes linéaires

$$\mathcal{U} = \{X_1 + iX_2 + \ldots + i^{n-1}X_n \ , \ i = 1, \ldots, D\}$$

contient au moins un polynôme séparant $V(\mathcal{I})$.

Une idée simple, mais peu économique, consisterait à calculer tous les polynômes caractéristiques de tous les éléments de $\mathcal{U}$, pour garder un de ceux dont le polynôme caractéristique admet un nombre maximum de racines (i.e : dont le degré de la partie sans facteurs carrés est maximum).

Comme un polynôme générique est séparant, notre but , dans l'esprit des résultats exposés dans [Ca88], est ici de fournir un test de validité a posteriori, venant après le calcul de la représentation univariée rationnelle associée à un polynôme quelconque, pour détecter si celui-ci est séparant où non.

Soient $t \in K[X_1, \ldots, X_n]$ et $f(t,T), g(t,T), g_1(t,T), \ldots, g_n(t,T)$ les polynômes introduits dans la notation 3.3.1. Le lemme suivant sera à la base des divers résultats de cette partie :

Lemme 3.3.2 *Un polynôme $t \in K[X_1, \ldots, X_n]$ est séparant pour $V(\mathcal{I})$ si et seulement si:*

$$\forall i = 1 \ldots n \ , \ u_i(t) = g(t,t)X_i - g_1(t,t) \in \sqrt{\mathcal{I}}.$$

Preuve :

- $\Rightarrow$: Supposons que t soit séparant. Nous avons alors (proposition 3.1.4),

  $$\forall \beta \in V(\mathcal{I}) \ , \ X_i(\beta) = \frac{g_i(t, t(\beta))}{g(t, t(\beta))}.$$

  Ainsi, $u_i(t(\beta)) = 0$ et donc, $u_i(t) \in \sqrt{\mathcal{I}}$.

- $\Leftarrow$: Supposons que t ne soit pas séparant et que $\forall i = 1, \ldots, n \ , \ u_i(t) \in \sqrt{\mathcal{I}}$. Soient alors α et β deux zéros distincts de $\mathcal{I}$ tels que $t(\alpha) = t(\beta)$. Comme $\alpha \neq \beta$, il existe un indice i tel que $X_i(\alpha) \neq X_i(\beta)$. Toutefois, $u_i(t) \in \sqrt{\mathcal{I}}$ et donc, $u_i(t(\alpha)) = u_i(t(\beta)) = 0$. Par définition de u_i et comme $g(t,t)$ ne s'annule pas sur $V(\mathcal{I})$, nous avons :

  $$X_i(\beta) = \frac{g_i(t, t(\beta))}{g(t, t(\beta))}$$

  $$X_i(\alpha) = \frac{g_i(t, t(\alpha))}{g(t, t(\alpha))}$$

  Mais $t(\alpha) = t(\beta) \Rightarrow X_i(\beta) = X_i(\alpha)$, ce qui contredit es hypothèses.

□

Le problème de vérification est maintenant ramené à un test d'appartenance au racial d'un idéal. En adaptant le lemme 1.2.4 à notre problème , nous obtenons alors le corollaire suivant :

Corollaire 3.3.1 *Soient $t \in K[X_1, \ldots, X_n]$ et $f(t,T), g(t,T), g_1(t,T), \ldots, g_n(t,T)$ les polynômes introduits dans la notation 3.3.1. En posant, pour tout $h \in K[X_1, \ldots, X_n]$, $Vtr(h) = [Trace(m_{h\omega_1}), \ldots Trace(m_{\omega_D})]$, d'après la proposition 1.2.4, l'élément t est séparant (en d'autres termes $\{f(t,T), g(t,T), g_1(t,T), \ldots, g_n(t,T)\}$ est une représentation univariée rationnelle de $\mathcal{I}$) si et seulement si :*

$$Vtr(u_i(t)) = 0$$

pour tout $i = 1 \ldots n$. En posant :

- $d = degree(f(t))$
- $g(t,t) = \sum_{j=0}^{d-1} a_j t^j$ *et* $g_i(t,t) = \sum_{j=0}^{d-1} a_j^i t^j$.

cette condition peut être traduite par :

$$M_{X_i}^T \left(\sum_{j=0}^{d-1} a_j Vtr(t^j) \right) - \sum_{j=0}^{d-1} a_j^i Vtr(t^j) = 0$$

pour tout $i = 1 \ldots n$.

Preuve : En utilisant la linéarité de l'application $Trace$, on peut écrire :

$$\begin{aligned} Vtr(u_i(t)) &= Vtr(g(t)X_i) - Vtr(g_i(t)) \\ &= [Trace(m_{g(t)X_1\omega_1}), \ldots, Trace(m_{g(t)X_1\omega_D})] - \textstyle\sum_{j=0}^{d-1} a_j^i Vtr(t^j) \\ &= [\overrightarrow{X_i\omega_1} Vtr(g(t)), \ldots, \overrightarrow{X_i\omega_D} Vtr(g(t))] - \textstyle\sum_{j=0}^{d-1} a_j^i Vtr(t^j) \\ &= M_{X_i}^T Vtr(g(t)) - \textstyle\sum_{j=0}^{d-1} a_j^i Vtr(t^j) \\ &= M_{X_i}^T \left(\textstyle\sum_{j=0}^{d-1} a_j Vtr(t^j) \right) - \textstyle\sum_{j=0}^{d-1} a_j^i Vtr(t^j) \end{aligned}$$

□

Remarque 3.3.1 *Si l'algorithme* **TracesToExtNewton** *à été utilisé pour calculer les polynômes $f(t,T), g(t,T), g_1(t,T), \ldots, g_n(t,T)$, tous les vecteurs $Vtr(t^j)$ sont déjà connus (voir preuve de l'algorithme) ce qui rend ce test performant.*
Si $g(t,T)$ est de degré $D-1$, $f(t,T)/gcd(f(t,T), f'(t,T))$ est forcément de degré maximum. Dans ce cas t est séparant et l'on sait de plus que $\mathcal{I}$ est radical.

Ceci nous donne un algorithme pour tester, a postériori, si un polynôme arbitrairement choisi est séparant où non :

Algorithme TestSep

Entrée: Une table de multiplication MT de $K[X_1, \ldots, X_n]/\mathcal{I}$, des polynômes (voir notation3.3.1) :

$$\{f(t,T), g(t,T), g_1(t,T), \ldots, g_n(t,T)\}$$

(avec : $g(t,T) = \sum_{j=0}^{d-1} a_j T^j$ et $g_i(t,T) = \sum_{j=0}^{d-1} a_j^i t^j$, pour tout $i = 1 \ldots n$), ainsi que les vecteurs $Vtr(t^j)$ pour $j = 0, \ldots, D-1$.

- **Etape 0** : Si $g(t,T)$ est de degré $D-1$, t est séparant.
- **Etape 1** : Calcul de $Vtr(g(t,T)) = \sum_{j=0}^{d-1} a_j Vtr(t^j)$.
- **Etape 2** : Pour $i = 1, \ldots, n$, calcul de :
  - $Vtr(g(t,t)X_i) = M_{X_i}^T Vtr(g(t,t))$
  - $Vtr(g_i(t)) = \sum_{j=0}^{d-1} a_j^i Vtr(t^j)$
  - $Vtr(u_i(t)) = Vtr(g(t,T)X_i) - Vtr(g_i(t))$

Sortie: **Vrai** si t est séparant (c'est à dire si $Vtr(u_i(t)) = 0$, pour tout $i = 1 \ldots n$ ou si $g(t,T)$ est de degré $D-1$), **Faux** sinon.

Si nous faisons un bilan des différents résultats établis dans ce chapitre, nous pouvons exhiber un algorithme complet pour le calcul de la représentation univariée rationnelle d'un système zéro-dimensionnel. En supposant que le point d'entrée est une table de multiplication MT de $K[X_1, \ldots, X_n]/\mathcal{I}$, c'est à dire la donnée d'une base $\mathcal{B} = \{\omega_1, \ldots, \omega_D\}$ de $K[X_1, \ldots, X_n]/\mathcal{I}$ et de tous les produits réduits $\overrightarrow{\omega_i \omega_j}$, nous suggérons l'algorithme :

Algorithme RUR

Entrée: MT

- **Etape 1** : Choix d'une forme linéaire (lemme 1.2.1) t parmi :

$$\{X_1 + iX_2 + \ldots + i^{n-1}X_n \ , \ i = 1, \ldots, D\}$$

- **Etape 2** : Calcul des Traces (Compute-Traces) $Trace(m_{t^i})$, $i = 1, \ldots, D$ et $Trace(m_{X_j t^i})$, $i = 1, \ldots, D-1$, $j = 1, \ldots, n$
- **Etape 3** : Déduction des polynômes (formule de Newton étendue)

$$f(t,T), h(X_1,t,T), \ldots, h(X_n,t,T)$$

(notations du lemme 3.1.1) (

- **Etape 4** : Calcul de :

$$g(t,T) = f'/gcd(f(t,T), f'(t,T))$$

et, pout tout $i = 1, \ldots, n$, de :

$$g_i(t,T) = h(X_i,t,T)/gcd(f(t,T), f'(t,T))$$

- **Etape 5** : Test de l'élément séparant (TestSep). Si t n'est pas séparant, retourner à l'étape **1**.

Sortie: Un polynôme t séparant $V(\mathcal{I})$ et la représentation univariée rationnelle $\{f(t,T), g(t,T), g_1(t,T), \ldots, g_n(t,T)\}$ correspondante.

3.4 Complexité

L'entrée de l'algorithme représentation univariée rationnelle étant une table de multiplication MT associée à l'idéal zéro-dimensionnel que nous étudions, il est naturel de ne considérer que les paramêtres liés à cette représentation.

En particulier, pour l'étude du comportement en termes de croissance des coefficients, l'unité que nous choisirons pour les calculs est la taille des représentation binaires des entiers intervenant dans l'expression des matrices M_{X_i} (multiplication par une variable dans $K[X_1, \ldots, X_n]/\mathcal{I}$. Deux raisons à cela : d'une part, aucun lien évident (ou du moins précisément établi) n'existe entre les coefficients des polynômes définissant l'idéal et ceux intervenant dans l'expression d'une base de Groebner de ce même idéal, et d'autre part, on peut imaginer d'autres procédés pour représenter l'algèbre quotient qu'une base de Groebner (nous comptons, dans le futur, pouvoir exploiter certains résultats de [BCRS94] ou de [Car93]) et la donnée de ces matrices nous parrait être un prérequis minimum.

Proposition 3.4.1 *Soient $\mathcal{I} \subset K[X_1, \ldots, X_n]$ un idéal zéro-dimensionnel et MT une table de multiplication de $K[X_1, \ldots, X_n]/\mathcal{I}$. Notons D la dimension du K-espace vectoriel $K[X_1, \ldots, X_n]/\mathcal{I}$, dont une base sera notée $\{\omega_1 = 1, \ldots, \omega_D\}$, et supposons donné un polynôme t séparant $V(\mathcal{I})$. La représentation univariée rationnelle de $\mathcal{I}$ associée à t se déduit de MT en*

$$O(D^3 + nD^2)$$

opérations arithmétiques.
Dans le cas où aucun vecteur séparant n'est connu a priori, le processus utilisera alors au maximum

$$O(n^2d^2(D^3 + nD^2))$$

opérations arithmétiques où d est le nombre de racines distinctes du système.

Preuve : Enumérons une à une les diverses étapes décrites plus haut pour le calcul de la représentation univariée rationnelle de $\mathcal{I}$ associée à t :

- Le calcul préalable à l'algorithme TracesToExtNewton consiste à former le vecteur
$$[Trace(m_{\omega_1}), \ldots, Trace(m_{\omega_D})]$$
Chaque trace $Trace(m_{\omega_i})$ requiert D additions lorsque la table MT est connue. Le nombre d'opérations induites par cette étape est donc $O(D^2)$.

- Calcul des sommes de Newton étendues par l'algorithme TracesToExtNewton utilise clairement D multiplications Matrice-Vecteur de dimension D et $D(n+1)$ produits scalaires de deux vecteurs de dimension D. Le nombre d'opérations induites par cette étape est donc $O(D^3 + nD^2)$.

- Le calcul des coefficients des différents polynômes $\{f, h(X_1, t, T), \ldots, h(X_n, t, T)\}$ (voir lemme 3.1.1) s'effectue en résolvant $n + 1$ systèmes triangulaires. Le nombre d'opérations induites par cette étape est donc $O(nD^2)$ et, d'après le lemme 3.1.1, les polynômes de la représentation univariée rationnelle se déduisent en $O(nD^2)$ opérations arithmétiques.

En ce qui concerne le processus global, il faut inclure :

- Le test a postériori de validité de l'élément séparant qui est en $O(nD^2)$ et ne modifie en rien la complexité de l'algorithme.

- Le nombre de polynômes potentiellement non séparants que nous pouvons avoir à tester:
$$(n-1)d(d-1)/2$$
si $d \leq D$ est le nombre de racines complexes distinctes.

□

Dans le cas des systèmes à coefficients entiers où rationnels, nous pouvons également mesurer les tailles des représentations binaires des entiers apparaissant dans le résultat d'une représentation univariée rationnelle. En reprenant les notations du lemme 3.1.1, nous nous attachons dans un premier temps à l'étude du calcul de la famille de polynômes :

$$\mathcal{R} = \{f, h(X_1, t, T), \ldots, h(X_n, t, T)\}$$

Proposition 3.4.2 *Soient $\mathcal{I} \subset \mathbb{Q}[X_1, \ldots, X_n]$ un idéal zéro-dimensionnel et MT une table de multiplication de $\mathbb{Q}[X_1, \ldots, X_n]/\mathcal{I}$. Notons D la dimension de l'espace vectoriel $\mathbb{Q}[X_1, \ldots, X_n]/\mathcal{I}$ et $\mathcal{T}$ la taille maximale d'un coefficient apparaissant dans le sous-ensemble de MT :*

$$\{\overrightarrow{X_i \omega_j} \ , \ i = 1..n \ , \ j = 1 \ldots D\}.$$

La taille maximale d'un coefficient apparaissant dans les polynômes de $\mathcal{R}$ est en

$$O(D(\mathcal{T} + log(D)))$$

lorsque l'un des X_i sépare $V(\mathcal{I})$, et en

$$O(D(\mathcal{T} + nlog(nD)))$$

lorsqu'un polynôme t séparant $\mathcal{I}$ est choisi dans l'ensemble

$$\{X_1 + iX_2 + \ldots + i^{n-1}X_n \ i = 0 \ldots (n-1)d(d-1)/2\}.$$

Preuve : D'après le lemme 3.1.1, les polynômes $\{f, h(X_1, t, T), \ldots, h(X_n, t, T)\}$ peuvent être obtenus par des calculs de polynômes caractéristiques. Si $\mathcal{T}_1$ est la taille maximale des coefficients apparaissant dans les matrices $M_t, M_{X_1}, \ldots, M_{X_n}$, la taille d'un coefficient apparaissant dans les determinants de la forme $Det(M_{t+SX_i} - TI)$ est alors en $O(D(\mathcal{T}_1 + log(D)))$ et par conséquent la taile maximum d'un coefficient apparaissant dans $\mathcal{R}$, (c'est à dire dans les expressions $\left(\frac{\partial}{\partial S} Det(M_{t+SX_i} - TI)\right)_{S=0}$) est en $O(D(\mathcal{T}_1 + log(D)))$. Si t est l'un des X_i, $\mathcal{T}_1 = \mathcal{T}$. Dans le cas contraire, la matrice M_t est une combinaison linéaire des matrices M_{X_i} de la forme $\sum_{i=1}^{n} a_i M_{X_i}$ où les a_i sont des entiers inférieurs à $(nD^2)^n$, par conséquent $\mathcal{T}_1 = O(\mathcal{T} + nlog(nD))$.
□

Nous avons calculé le nombre d'opérations arithmétiques utilisées lorsque l'algorithme **RUR** est appliqué pour calculer la représentation univariée rationnelle d'un système et nous avons fourni une borne sur la taille des coefficients du résultat. Nous montrons maintenant le bon comportement de cet algorithme en calculant la taille des coefficients intervenant en cours de calcul :

Proposition 3.4.3 *En reprenant les notations de la proposition précédente, la taille des coefficients apparaissant dans le calcul de $\mathcal{R}$ par l'algorithme RUR est*

$$O(D(\mathcal{T} + log(D)))$$

lorsque l'un des X_i sépare $V(\mathcal{I})$, et en

$$O(D(\mathcal{T} + nlog(nD)))$$

dans le cas général.

Preuve : Reprenons une à une les différentes étapes du calcul de $\mathcal{R}$ par l'algorithme RUR pour étudier la croissance des coefficients dans les calculs intermédiaires dans le cas général :

- Le calcul préalable à l'algorithme TracesToExtNewton consiste à former le vecteur
$$Vtr(1) = [Trace(m_{\omega_1}), \ldots, Trace(m_{\omega_D})].$$
Chaque trace $Trace(m_{\omega_i}) = \sum_{i=1}^{D} (\overrightarrow{\omega_i \omega_j})_j$ requiert D additions de coefficients dont la taille est $O(D\mathcal{T})$ (taille des coeficients apparaissant dans la table de multiplication), les coefficients apparaissant dans $Vtr(1)$ sont par conséquent de taille $O(D\mathcal{T} + log(D))$.

- Dans le calcul des sommes de Newton étendues, la construction des vecteurs $Vtr(t^i)$ s'effectue itérativement par la formule :
$$Vtr(t^i) = M_t^T \cdot Vtr(t^{i-1}).$$
D'après la démonstration de la proposition précédente, la taille d'un coefficient apparaissant dans M_t est en $O(\mathcal{T} + nlog(nD))$ et par conséquent la taille maximale d'un coefficient apparaissant dans $Vtr(t^i)$, $i = 1 \ldots D$ est en $O(D(\mathcal{T} + nlog(nD)))$. Les traces $Trace(m_{X_j t^i})$ et $Trace(m_{t^{i+1}})$ étant calculées par la formules : $Trace(m_{X_j t^i}) = \overrightarrow{X_i} \cdot Vtr(t^i)$ et $Trace(m_{t^{i+1}}) = \overrightarrow{t} \cdot Vtr(t^i)$, leur taille est également en $O(D(\mathcal{T} + nlog(nD)))$.

- Les coefficients des différents polynômes de $\mathcal{R}$ sont obtenus en résolvant $n+1$ systèmes triangulaires par subsitutions successives (par une méthode sans fractions). Comme la taille des entiers intervenant dans le résultat est en $O(D(\mathcal{T} + nlog(nD)))$ et que les coefficients apparaissant dans l'expression des systèmes sont de taille $O(D(\mathcal{T} + nlog(nD)))$, les calculs intermédiaires feront intervenir des entiers de taille maximale $O(D(\mathcal{T} + nlog(nD)))$.

Le même raisonnement s'applique dans le cas où l'un des X_i sépare $V(\mathcal{I})$. La taille maximum d'un coefficient apparaissant dans M_t étant dans ce cas en $O(\mathcal{T} + log(D))$, les calculs intermédiaires feront alors intervenir des entiers de taille maximale $O(D(\mathcal{T} + log(D)))$. □

Lorsque toutes les racines du système sont simples et que t sépare $V(\mathcal{I})$, l'ensemble $\mathcal{R}$ coincide avec une représentation univariée rationnelle de $\mathcal{I}$, mais dans le cas contraire, rappelons que la représentation univariée rationnelle est obtenue en divisant les polynômes $f', h(X_1, t, T), \ldots, h(X_n, t, T)$ par $gcd(f, f')$. Une analyse rapide de la complexité nous donne une taille maximale de coefficients en $O(D^2(\mathcal{T} + nlog(nD)))$ dans le cas général.

Notons que dans ce cas, le calcul de pgcd ($gcd(f, f')$), les n divisions nécessaires pour le calcul des polynômes de la représentation univariée rationnelle ainsi que le test (TestSep) sont effectués en utilisant des entiers de taille $O(D^2(\mathcal{T} + nlog(nD)))$. Ces $O(nD^2)$ opérations ne peuvent par conséquent plus être négligés par rapport au reste ($O(D^3 + nD^2)$ opérations arithmétiques sur des entiers de taille $O(D(\mathcal{T} + nlog(nD))))$.
En pratique toutefois, les coefficients d'un pgcd sont en général plus petits que ceux des polynômes considérés et cette analyse ne correspond donc pas avec les résultats observées, les coefficients des polynômes de la représentation univariée rationnelle étant en général plus petits que ceux des polynômes de $\mathcal{R}$.

3.5 Cas des systèmes à coefficients entiers

Nous revenons dans cette partie sur certaines étapes de l'algorithme RUR dans le cas de systèmes à coefficients entiers, et en particulier sur le problème de l'élément séparant. L'idéal de $\mathbb{Z}[X_1, \ldots X_n]$ considéré sera noté $\mathcal{I}$ et supposé zéro-dimensionnel dans $\mathbb{Q}[X_1, \ldots X_n]$, D sera la dimension de $\mathbb{Q}[X_1, \ldots, X_n]/\mathcal{I}$ et le nombre de racines distinctes du système (dans $\mathbb{C}^n$) sera noté $d = \sharp V(\mathcal{I})$.
La pratique montre que, pour obtenir des expressions finales plus simples, nous devons considérer en priorité les polynômes de la forme $X_i \ , i = 1 \ldots n$ comme séparant potentiellement $V(\mathcal{I})$. Il est toutefois fréquent qu'un polynôme séparant $V(\mathcal{I})$ ne soit pas de la forme $X_i \ , i = 1 \ldots n$. Dans ce cas, l'algorithme RUR sera exécuté au moins $n + 1$ fois dans sa totalité, en prenant t dans la famille $\{X_1, \ldots, X_n, X_1 + iX_2 + \ldots + i^{n-1}X_n \ , i = 1 \ldots (n-1)d(d-1)/2\}$ jusqu'à l'obtention d'une représentation univariée de $\mathcal{I}$.
Nous proposons ici une prédiction, par un calcul modulaire (donc rapide), d'un polynôme séparant.
Comme pour tout algorithme utilisant une technique modulaire, il nous faut, dans un premier temps définir une notion d'*entier premier de bonne réduction*, en nous limitant toutefois au cas où l'idéal $\mathcal{I}$ est représenté par une base de Groebner pour un ordre arbitraire (que nous noterons $>$) sur les monômes.

3.5.1 Entiers de bonne réduction pour la représentation univariée rationnelle

Si on applique l'algorithme de Buchberger classique pour calculer la base de Groebner d'un idéal de $\mathbb{Q}[X_1, \ldots, X_n]$ (pour un ordre fixé $>$) , nous obtenons une famille de polynômes de $\mathcal{G} \subset \mathbb{Q}[X_1, \ldots, X_n]$. Toutefois, nous pouvons remarquer que pour tout $a \in \mathbb{Z}$, $a \neq 0$, si $\mathcal{G}$ est une base de Groebner de $\mathcal{I}$, l'ensemble $\{ag \ , \ g \in \mathcal{G}\}$ est encore une base de Groebner de $\mathcal{I}$ pour $>$. En choisissant cet entier de façon correcte, nous pouvons donc supposer que, pour l'ordre $>$, $\mathcal{G} \subset \mathbb{Z}[X_1, \ldots, X_n]$.

Malgré ce changement, pour tout $h \in \mathbb{Z}[X_1, \ldots, X_n]$, le résultat de la réduction de h modulo $\mathcal{G}$ est encore un élément de $\mathbb{Q}[X_1, \ldots, X_n]$. Pour tout $g \in \mathcal{G}$, en considérant un entier n_g suffisement grand, on peut toutefois remarquer que le reste de la division de $lc(g)^{n_g}h$ par g est un polynôme de $\mathbb{Z}[X_1, \ldots, X_n]$, ce qui nous conduit à énoncer le lemme :

Lemme 3.5.1 *Pour tout polynôme $h \in \mathbb{Z}[X_1, \ldots, X_n]$, il existe des entiers positifs n_g , $g \in \mathcal{G}$ tels que ;*

$$NF((\prod_{g\in\mathcal{G}} lc(g)^{n_g}) \cdot h, \mathcal{G}) \in \mathbb{Z}[X_1, \ldots, X_n]$$

et des polynômes q_g , $g \in \mathcal{G}$, $q_g \in \mathbb{Z}[X_1, \ldots, X_n]$ tels que

$$(\prod_{g\in\mathcal{G}} lc(g)^{n_g}) \cdot h = NF((\prod_{g\in\mathcal{G}} lc(g)^{n_g}) \cdot h, \mathcal{G}) + \sum_{g\in\mathcal{G}} q_g \cdot g$$

Notation 3.5.1 *Pour tout $p \in \mathbb{Z}$ premier, nous noterons $\mathbb{Z}_p$ le localisé de $\mathbb{Z}$ en $p\mathbb{Z}$, c'est à dire le sous anneau de $\mathbb{Q}$:*

$$\mathbb{Z}_p = \{a/b\ ,\ a \in \mathbb{Z}\ , b \in \mathbb{Z}\ |p \textit{ ne divise pas } b\},$$

$GF(p)$ le corps fini d'ordre p ($\mathbb{Z}/p\mathbb{Z}$) et $\overline{GF(p)}$ sa clôture algébrique.
$\phi_p : \mathbb{Z}_p[X_1, \ldots, X_n] \longrightarrow GF(p)[X_1, \ldots, X_n]$ sera alors le morphisme canonique tel que $\forall a/b \in \mathbb{Z}_p$, $\phi_p(a/b) = (a \bmod p)(b \bmod p)^{-1}$.

Definition 3.5.1 *Nous dirons qu'un entier premier p est $\mathcal{G}$-compatible (pour l'ordre $>$), si p ne divise aucun des $lc(g)$, $g \in \mathcal{G}$.*

Proposition 3.5.1 *Soit $\mathcal{G} \subset \mathbb{Z}[X_1, \ldots, X_n]$ une base de Groebner réduite d'un idéal $\mathcal{I} \subset \mathbb{Q}[X_1, \ldots, X_n]$ pour un ordre $>$ admissible. Si p est un entier premier $\mathcal{G}$-compatible pour $>$, alors $\phi_p(\mathcal{G})$ est une base de Groebner de $\langle\phi_p(\mathcal{G})\rangle$ (pour $>$) et $\phi_p(NF(f, \mathcal{G})) = NF(\phi_p(f), \phi_p(\mathcal{G}))$*

Pour établir cette proposition, nous aurons besoin d'un résultat intermédiaire :

Lemme 3.5.2 *Soit $\mathcal{G} \in \mathbb{Z}[X_1, \ldots, X_n]$ une base de Groebner réduite d'un idéal $\mathcal{I} \subset \mathbb{Q}[X_1, \ldots, X_n]$ pour $>$ et p un entier premier $\mathcal{G}$-compatible pour $>$. Pour tout $f \in \mathbb{Z}_p[X_1, \ldots, X_n]$, rappelons que l'expression suivante est unique :*

$$f = \sum_{g\in\mathcal{G}} q_g g + r_f$$

avec r_f tel que ses monômes ne soient divisibles par aucun des $lm(g)$, $g \in \mathcal{G}$. De plus :

- *(1)* $\forall g \in \mathcal{G}$, $q_g \in \mathbb{Z}_p[X_1, \ldots, X_n]$
- *(2)* $r_f \in \mathbb{Z}_p[X_1, \ldots, X_n]$
- *(3) Pour tout* $f \in \mathbb{Z}_p[X_1, \ldots, X_n]$ *, on peut ordonner les polynômes de* $\mathcal{G}$ *de telle sorte que :*
$$\phi_p(NF(f, \mathcal{G})) = NF(\phi_p(f), \phi_p(\mathcal{G}))$$

Preuve : Supposons dans un premier temps que f soit un polynôme de $\mathbb{Z}[X_1, \ldots X_n]$. Comme $\mathcal{G}$ est une base de Groebner réduite de $\mathcal{I}$, l'expression

$$f = \sum_{g \in \mathcal{G}} q_g g + r_f$$

est unique. En reprenant les notations du lemme 3.5.1, il existe des entiers positifs n_g , $g \in \mathcal{G}$ tels que ;

$$NF((\prod_{g \in \mathcal{G}} lc(g)^{n_g}) \cdot f, \mathcal{G}) \in \mathbb{Z}[X_1, \ldots, X_n]$$

et des polynômes q_g , $g \in \mathcal{G}$, $q_g \in \mathbb{Z}[X_1, \ldots, X_n]$ tels que

$$(\prod_{g \in \mathcal{G}} lc(g)^{n_g}) \cdot f = NF((\prod_{g \in \mathcal{G}} lc(g)^{n_g}) \cdot f, \mathcal{G}) + \sum_{g \in \mathcal{G}} q_g \cdot g.$$

Comme p est $\mathcal{G}$-compatible, $NF(f, \mathcal{G}) \in \mathbb{Z}_p[X_1, \ldots, X_n]$. Si f est maintenant un polynôme de $\mathbb{Z}_p[X_1, \ldots X_n]$, on peut trouver un entier a non divisible par p et un polynôme $f' \in \mathbb{Z}[X_1, \ldots X_n]$ tels que $f = af'$. La $\mathbb{Q}$-linéarité de l'application $NF(., \mathcal{G})$ nous permet alors de montrer les points (1) et (2).
Nous supposons que p est $\mathcal{G}$-compatible et que les polynômes de $\mathcal{G}$ sont unitaires dans $\mathbb{Z}_p[X_1, \ldots, X_n]$ ($\forall g \in \mathcal{G}$, $\phi_p(g)$ est donc bien défini). L'algorithme de réduction de f modulo $g \in \mathcal{G}$ fait apparaître une suite de polynômes : $f_0 = f$, $f_i = f_{i-1} - lc(f_{i-1}) t_i g$ si $f_{i-1} \neq 0$ et si $lm(f_{i-1})$ est divisible par $lm(g)$ et $f_i = f_{i-1}$ sinon, où t_i est un monôme unitaire de $\mathbb{Z}[X_1, \ldots, X_n]$. Si f est un élément de $\mathbb{Z}_p[X_1, \ldots, X_n]$ la suite (f_i) est donc dans $\mathbb{Z}_p[X_1, \ldots, X_n]$.
Plus généralement, l'algorithme de réduction modulo la famille ordonnée $\mathcal{G}$ fait apparaître une suite de polynômes de $\mathbb{Z}_p[X_1, \ldots, X_n]$:

$$f_0 = f \ , \ f_i = f_{i-1} - lc(f_{i-1}) t_i g_{k_i}$$

où les g_{k_i} sont des éléments de $\mathcal{G}$. Le système de générateurs $\mathcal{G}$ étant une base de Groebner, nous pouvons supposer la liste $\mathcal{G} = [g_1, \ldots, g_u]$ ordonnée de telle sorte que $i \geq j \Rightarrow k_i > k_j$.
Soit e l'indice maximal ainsi défini. Nous avons alors $NF(f, \mathcal{G}) = f_e$.

De la même façon, on définit la suite $(f_i^{(p)})$ par :

$$f_0 = \phi_p(f)\ ,\ f_i^{(p)} = f_{i-1}^{(p)} - lc(f_{i-1}^{(p)})t_i\phi_p(g_{l_i}),$$

et on note s l'indice le plus élevé dans la suite $(f_i^{(p)})$ tel que $f_i^{(p)} = \phi_p(f_i)$.
Comme $\mathcal{G}$ est une base de Groebner réduite, aucun des $lm(g)$, $g \in \mathcal{G}$, ne divise l'un des monômes de f_e. Comme p est $\mathcal{G}$-compatible, aucun des $lm(\phi_p(g))$, $g \in \mathcal{G}$, ne divise l'un des monômes de $\phi_p(f_e)$. Par conséquent $s \leq e$.
Si $\phi_p(lc(f_s)) \neq 0$, alors $f_{s+1}^{(p)} = f_s^{(p)} - lc(f_s^{(p)})\phi_p(g_{l_s}) = \phi_p(f_s) - \phi_p(lc(f_s))\phi_p(g_{k_s})$ et donc $f_{s+1}^{(p)} = \phi_p(f_{s+1})$ (s n'est donc pas maximal). Alors, $\phi_p(lc(f_s)) = 0$. Dans ce cas, $f_s = f'_s + pf_p$ avec $\phi_p(lc(f'_s)) \neq 0$ et $\phi_p(NF(f,\mathcal{G})) = \phi_p(NF(f'_s,\mathcal{G}))$ ce qui montre que $s = e$ et donc que $\phi_p(NF(f,\mathcal{G})) = NF(\phi_p(f),\phi_p(\mathcal{G}))$. □

Ce résultat montre en particulier que si $f \in \mathcal{I}$, alors $\phi_p(f) \in \langle\phi_p(\mathcal{G})\rangle$. Nous pouvons maintenant montrer la proposition 3.5.1:

Preuve de la proposition 3.5.1: Soient g et g' deux polynômes de $\mathcal{G}$ et $S(g,g')$ le S-polynôme qui leur est associé. Comme $\phi_p(S(g,g')) = S(\phi_p(g),\phi_p(g'))$, et comme $\mathcal{G}$ est une base de Groebner pour $>$, $NF(S(\phi_p(g),\phi_p(g'),\mathcal{G}) = 0$ et $\phi_p(\mathcal{G})$ est donc une base de Groebner de $\langle\phi_p(\mathcal{G})\rangle$ pour $>$. Enfin si $\mathcal{G}$ est une base de Groebner réduite, $\phi_p(\mathcal{G})$ est a fortiori réduite (p est $\mathcal{G}$-compatible). □

Rappelons qu'une représentation univariée rationnelle peut être calculée à l'aide de polynômes caractéristiques d'endomorphismes de multiplication. Il convient alors de regarder le comportement de ces quantités lors d'une projection dans $GF(p)[X_1,\ldots,X_n]$.

Proposition 3.5.2 *Soient $\mathcal{I}$ un idéal zéro-dimensionnel de $\mathbb{Q}[X_1,\ldots,X_n]$, $\mathcal{G} \subset \mathbb{Z}[X_1,\ldots,X_n]$ une base de Groebner réduite de $\mathcal{I}$ (pour un ordre $>$), et $p > D$ un entier premier $\mathcal{G}$-compatible pour $>$. Notons $\langle\phi_p(\mathcal{G})\rangle$ l'idéal engendré par $\phi_p(\mathcal{G})$ et, pour tout $t \in \mathbb{Z}[X_1,\ldots,X_n]$, P_t (resp. $P_{\phi_p(t)}$) le polynôme caractéristique de m_t (supposé unitaire) (resp. $m_{\phi_p(t)}$) dans $\mathbb{Q}[X_1,\ldots,X_n]/\mathcal{I}$ (resp. $GF(p)[X_1,\ldots,X_n]/\langle\phi_p(\mathcal{G})\rangle$). Alors :*

- $dim_{GF(p)}(GF(p)[X_1,\ldots,X_n]/\langle\phi_p(\mathcal{G})\rangle) = dim_{\mathbb{Q}}(\mathbb{Q}[X_1,\ldots,X_n]/\mathcal{I})$
- *Pour tout $t \in \mathbb{Z}[X_1,\ldots,X_n]$, le polynôme P_t est à coefficients dans $\mathbb{Z}_p$ et $\phi_p(P_t) = P_{\phi_p(t)}$.*

Preuve : Rappelons qu'une base de $\mathbb{Q}[X_1,\ldots,X_n]/\mathcal{I}$ peut être définie par l'ensemble des monômes unitaires $\mathcal{B} = \{\omega_1,\ldots,\omega_D\} \in \mathcal{M}[X_1,\ldots,X_n]$ ne divisant aucun des $\{lc(g)\ ,\ g \in \mathcal{G}\}$. Comme p est un entier premier $\mathcal{G}$-compatible, $\phi_p(\mathcal{G})$ est donc une base de Groebner réduite de $\langle\phi_p(\mathcal{G})\rangle$ et par conséquent $\mathcal{B}$ est également une base de $GF(p)[X_1,\ldots,X_n]/\langle\phi_p(\mathcal{G})\rangle$. En particulier, si $NF(f,\mathcal{G})=\sum_{i=1\ldots D}a_i\omega_i$ ($a_i\in\mathbb{Z}_p$ pour tout $i=1,\ldots,D$), alors

$$\phi_p(NF(f,\mathcal{G}))=\sum_{i=1\ldots D}\phi_p(a_i)\omega_i=NF(\phi_p(f),\phi_p(\mathcal{G})).$$

Pour tout $t\in\mathbb{Z}[X_1,\ldots,X_n]$, soit M_t (resp. $M_{\phi_p(t)}$) la matrice de m_t (resp. $m_{\phi_p(t)}$) dans $\mathbb{Q}[X_1,\ldots,X_n]/\mathcal{I}$ (resp. $GF(p)[X_1,\ldots,X_n]/\langle\phi_p(\mathcal{G})\rangle$). Dans ce cas,

$$\phi_p(NF(t\omega_i,\mathcal{G}))=NF(\phi_p(t)\omega_i,\phi_p(\mathcal{G}))$$

pour tout $i=1,\ldots D$. La matrice M_t prend ses valeurs dans $\mathbb{Z}_p$, P_t est donc à coefficients dans $\mathbb{Z}_p$ et $\phi_p(M_t)=M_{\phi_p(t)}$. Par linéarité de ϕ_p, on obtient également $\phi_p(Trace(M_{t^i}))=Trace(M_{(\phi_p(t))^i})$ pour tout $i=0\ldots D$, ce qui montre que les sommes de Newton de P_t ont pour images, par ϕ_p, les sommes de Newton de $P_{\phi_p(t)}$. Par la formule de Newton, les coefficients de P_t peuvent s'obtenir à partir des sommes de Newton de P_t en résolvant un système triangulaire tel que le déterminant ($D!$) de la matrice associée est non nul modulo p (car $p>D$). Les coefficients de $P_{\phi_p(t)}$ s'obtiendront, par conséquent, en résolvant l'image par ϕ_p de ce système, et donc $\phi_p(P_t)=P_{\phi_p(t)}$. □

Corollaire 3.5.1 *Si $p>D$ est un entier premier $\mathcal{G}$-compatible, alors, pour tout $t\in\mathbb{Z}[X_1,\ldots,X_n]$:*

$$gcd(P_t,P_t')\in\mathbb{Z}_p[X_1,\ldots,X_n],$$
$$deg(gcd(P_t,P_t'))\le deg(gcd(P_{\phi_p(t)},P'_{\phi_p(t)}))$$

et par conséquent,

$$deg(P_t/gcd(P_t,P_t'))\ge deg(P_{\phi_p(t)}/gcd(P_{\phi_p(t)},P'_{\phi_p(t)}))$$

En particulier, $\sharp V(\mathcal{I})\ge\sharp V(\langle\phi_p(\mathcal{G})\rangle)$.

Preuve: Comme $P_t\in\mathbb{Z}_p[X_1,\ldots,X_n]$ (supposé unitaire), il existe un entier a tel que $\phi_p(a)\neq 0$, $aP_t\in\mathbb{Z}[X_1,\ldots,X_n]$ et tel que aP_t soit de contenu égal à 1. Dans ce cas $gcd(P_t,P_t')=1/a(gcd(aP_t,(aP_t)')$, ce qui montre que $Q=gcd(P_t,P_t')\in\mathbb{Z}_p[X_1,\ldots,X_n]$.
Comme $Q\in\mathbb{Z}_p[X_1,\ldots,X_n]$ divise P_t et P_t', $\phi_p(Q)$ divise $P_{\phi_p(t)}$ et $P'_{\phi_p(t)}$ est donc un diviseur de $gcd(P_{\phi_p(t)},P'_{\phi_p(t)})$.

Rappelons que si t (resp. $\phi_p(t)$) sépare $V(\mathcal{I})$ (resp. $V(\langle\phi_p(\mathcal{G})\rangle)$) alors le degré de la partie sans facteurs carrés de P_t (resp. $P_{\phi_p(t)}$) est maximal et égal à $\sharp V(\mathcal{I})$. Ainsi, $\sharp V(\mathcal{I})\ge\sharp V(\langle\phi_p(\mathcal{G})\rangle)$.

□

Ceci nous donne alors un résultat important pour la recherche d'éléments séparants :

Proposition 3.5.3 *Soit $p > D$ un entier premier $\mathcal{G}$-compatible. Si $\sharp V(\mathcal{I}) = \sharp V(\phi_p(\mathcal{G}))$ et si $t_p \in GF(p)[X_1, \dots, X_n]$ sépare $V(\phi_p(\mathcal{G}))$ alors, tout $t \in \mathbb{Z}[X_1, \dots, X_n]$ tel que $\phi_p(t) = t_p$, sépare $V(\mathcal{I})$.*

Preuve : Comme t_p sépare $V(\phi_p(\mathcal{G}))$, $P_{t_p}/gcd(P_{t_p}, P'_{t_p})$ est de degré maximal égal à $\sharp V(\phi_p(\mathcal{G}))$. Pour $t \in \mathbb{Z}[X_1, \dots, X_n]$,

$$deg(P_t/gcd(P_t, P'_t)) \leq \sharp V(\mathcal{I}).$$

Or, d'après la proposition 3.5.1,

$$deg(P_t/gcd(P_t, P'_t)) \geq deg(P_{t_p}/gcd(P_{t_p}, P'_{t_p})).$$

Comme $\sharp V(\mathcal{I}) = \sharp V(\phi_p(\mathcal{G}))$, $deg(P_t/gcd(P_t, P'_t)) = \sharp V(\mathcal{I})$ et t sépare $V(\mathcal{I})$. □

Dès lors qu'il est possible de trouver un entier premier p tel que l'égalité de la proposition précédente soit vérifiée, nous avons alors une technique modulaire pour déterminer un élément séparant $V(\mathcal{I})$. Attachons nous maintenant à la caractérisation des entiers p premiers pour lesquels cette égalité n'est pas vérifiée. En particulier, la proposition 3.5.3 fournira un algorithme complet si ce nombre d'entiers est fini.

Definition 3.5.2 *Soit $\mathcal{I} \subset \mathbb{Q}[X_1, \dots, X_n]$ un idéal zéro-dimensionnel, et $\mathcal{G} \subset \mathbb{Z}[X_1, \dots, X_n]$ une base de Groebner de $\mathcal{I}$. Un entier premier sera de bonne réduction pour le calcul d'une représentation univariée rationnelle de $\mathcal{I}$ si il est $\mathcal{G}$-compatible et si $\sharp V(\mathcal{I}) = \sharp V(\phi_p(\mathcal{G}))$.*

Rappelons que l'algorithme de Hermite permet, dans le cas d'un système à coefficients dans un corps de caractéristique nulle, de calculer le nombre de racines, nous donnant par conséquent le degré de la partie sans facteurs carrés du polynôme caractéristique d'un élément séparant.

Nous n'allons pas reprendre ici la démonstration complète du théorème de Hermite, nous nous contentons de présenter les conditions sur p pour qu'il puisse s'appliquer avec des systèmes zéro-dimensionnels à coefficients dans $GF(p)$. Les deux principaux théorèmes utilisés dans la démonstration pour des coefficients dans un corps de caractéristique zéro sont le théorème des zéros de Hilbert et celui de Stickelberger. Le théorème des zéros de Hilbert restant valide en caractéristique quelconque , il nous suffit de remarquer que celui de Stickelberger s'applique avec des coefficients dans $GF(p)$ pour tout premier $p > dim_{GF(p)}(GF(p)[X_1, \dots, X_n]/I_p)$, I_p étant l'idéal considéré, pour énoncer la proposition suivante :

Proposition 3.5.4 *Soient, un idéal zéro-dimensionnel* $I_p \subset GF(p)[X_1, \ldots, X_n]$ *et un polynôme* $h \in \mathbb{Z}[X_1, \ldots, X_n]$. *Notons* $V(I_p) \subset \overline{GF(p)}^n$ *l'ensemble des zéros de* I_p. *Si* $p > dim_{GF(p)}(GF(p)[X_1, \ldots, X_n]/I_p)$,

$$\rho(Q_{\phi_p(h)}) = \sharp\{\alpha \in V(I_p) | \phi_p(h)(\alpha) \neq 0\}$$

où $\rho(Q_{\phi_p(h)})$ *désigne le rang de* $Q_{\phi_p(h)}$.

Remarque 3.5.1 *En utilisant les mêmes arguments que dans les démonstrations du lemme 3.5.2 et de la proposition 3.5.2, et en posant* $I_p = \phi_p(\mathcal{I})$, *on peut montrer que si* $p > D$ *est* $\mathcal{G}$*-compatible, alors*

$$\phi_p(Q_h) = Q_{\phi_p(h)}$$

où Q_h *est la forme quadratique de Hermite généralisée définie sur* $\mathbb{Q}[X_1, \ldots, X_n]/\mathcal{I}$ *et associée à* $h \in \mathbb{Z}_p[X_1, \ldots, X_n]$.

De même, nous pouvons dériver le lemme 1.2.1 :

Lemme 3.5.3 *Soit* $\mathcal{X}$ *un ensemble de points de* $\overline{GF(p)}^n$ *tel que* $\sharp\mathcal{X} = d$. *Si* $p \geq (n-1)\frac{d(d-1)}{2}$, *la famille de formes linéaires* $\mathcal{U}_p = \{X_1 + \phi_p(i)X_2 + \ldots + \phi_p(i)^{n-1}X_n\ ,\ 0 \leq i \leq n\frac{d(d-1)}{2}\}$ *contient au moins un élément séparant* $\mathcal{X}$.

Le résultat suivant ne sera démontré que dans le prochain chapitre (partie 4.2.2) mais il nous sera toutefois nécessaire pour l'étude de la complexité de notre algorithme :

Proposition 3.5.5 *Soit* $\mathcal{I} \subset \mathbb{Q}[X_1, \ldots, X_n]$ *un idéal zéro-dimensionnel, et* $\mathcal{G} \subset \mathbb{Z}[X_1, \ldots, X_n]$ *une base de Groebner de* $\mathcal{I}$. *Si* $\mathcal{T}$ *désigne la taille maximale d'un entier intervenant dans l'expression des matrices* M_{X_i} *(multiplication par* X_i *dans* $\mathbb{Q}[X_1, \ldots, X_n]/\mathcal{I}$*), le nombre d'entiers premiers de taille* $O(1)$ *de mauvaise réduction pour le calcul d'une représentation univariée rationnelle de* $\mathcal{I}$ *est* $O(D^2\mathcal{T})$.

3.5.2 L'algorithme

La définition 3.5.2 nous indique quels sont exactement les entiers premiers de mauvaise réduction pour l'algorithme RUR lorsque les polynômes potentiellement séparants sont choisis dans une famille finie $\mathcal{U}$ contenant au moins un élément séparant $V(\mathcal{I})$. La proposition 3.5.5 montre en particulier que le nombre d'entiers premiers de mauvaise réduction est fini et que, pour tout entier premier $\mathcal{G}$-compatible de mauvaise réduction, $\sharp V(\mathcal{I}) > \sharp V(\phi_p(\mathcal{G}))$.
La proposition 3.5.3 montre que si de plus p est de bonne réduction et si il existe $t_0 \in \mathcal{U}$ tel que $\phi_p(t_0)$ sépare $V(\phi_p(\mathcal{G}))$, alors t_0 sépare $V(\mathcal{I})$.

Enfin, le lemme 3.5.3 montre que si

$$\mathcal{U} = \{X_1 + iX_2 + \ldots + i^{n-1}X_n \ , \ 0 \leq i \leq n\frac{d(d-1)}{2}\},$$

pour tout $p > nD^2$, $\phi_p(\mathcal{U})$ contient un tel élément t_0.
Bien entendu, nous ne pouvons pas trouver d'entier premier de bonne réduction a priori (leur caractérisation se faisant en fonction du résultat). Toutefois, nous disposons d'un test efficace (a posteriori - voir corollaire 3.3.1) permettant de vérifier si un polynôme t choisi arbitrairement sépare $V(\mathcal{I})$ ou pas.
Le principe de base de l'algorithme que nous proposons est de supposer que le nombre d'entiers premiers de mauvaise réduction est très faible (ce qui est d'ailleurs vérifié en pratique).
L'idée est de fixer un entier premier $p > nD^2$ $\mathcal{G}$-compatible (test facile à réaliser et peu coûteux), de trouver un élément séparant $V(\phi_p(\mathcal{G}))$ de la forme $\phi_p(t)$, $t \in \mathcal{U}$ (le degré de la partie sans facteurs carrés de $P_{\phi_p(t)}$ doit être égal au cardinal de $V(\phi_p(\mathcal{G}))$ lui-même étant égal au rang de $\phi_p(Q_1)$), puis de calculer les polynômes $f(t,T), g(t,T), g_1(t,T), \ldots, g_n(t,T)$ (notation 3.3.1) par les fonctions symétriques étendues.

En utilisant alors le test **TestSep**, nous vérifions alors que l'ensemble

$$\{f(t,T), g(t,T), g_1(t,T), \ldots, g_n(t,T)\}$$

est une représentation univariée rationnelle de $\mathcal{I}$. Dans le cas contraire, sommes assurés que p est un entier premier de mauvaise réduction, que $\sharp V(\mathcal{I}) > \sharp V(\phi_p(\mathcal{G}))$ et que t ne sépare pas $V(\mathcal{I})$. Nous changeons alors d'entier premier en prenant soin de choisir un entier p' tel que $\sharp V(\phi_{p'}(\mathcal{G})) > \sharp V(\phi_p(\mathcal{G}))$ et nous réitérons le procédé en retirant t de $\mathcal{U}$. Nous sommes sûr que l'algorithme s'arrête puisque que le nombre d'entiers premiers de mauvaise réduction est fini (proposition 3.5.5).
En langage de description, ceci nous donne l'algorithme :

$\boxed{\textbf{Algorithme RUR-Int}}$

Entrée : MT

Sortie : Un élément t séparant $V(\mathcal{I})$ et la représentation univariée rationnelle de $\mathcal{I}$ correspondante.

- **Étape 0** : Initialisation : $d_0 = 1$ (cardinal minimum de $V(\mathcal{I})$),

  $$\mathcal{U} = \{X_1, \ldots, X_n, X_1 + iX_2 + \ldots + i^{n-1}X_n \ , \ i = 1, \ldots, nD(D-1)/2\}$$

- **Étape 1** : Prédiction modulaire.

  - **Étape a** : Choix d'un entier premier $p > nD^2$ $\mathcal{G}$-compatible.
  - **Étape b** : Calcul du nombre de racines d_p de $\langle \phi_p(\mathcal{G}) \rangle$ par Hermite. Si $d_p < d_0$, retourner à l'étape **1-a**, sinon, $d_0 := d_p + 1$.
  - **Étape c** : Calcul de $\phi_p(M_{X_i})$, $i = 1, \ldots, n$.
  - **Étape d** : Choix d'un polynôme t (lemme 1.2.1) parmi les polynômes de $\mathcal{U}$.
  - **Étape e** : Calcul de $P_{\phi_p(t)}$ (polynôme caractéristique de d'une combinaison linéaire de matrices de la forme $\sum_{i=1}^{n} \phi_p(a_i)\phi_p(M_{X_i})$ par la méthode de Hessenberg). Si $deg(P_{\phi_p(t)}) \neq d_p$ retourner à l'étape **1-d**.

- **Étape 2** : On prend pour élément séparant t tel que $\phi_p(t)$ sépare $V(\phi_p(\mathcal{I}))$.
- **Étape 3** : Calcul des Traces (Compute-Traces) $Vtr(t^i)$, $i = 1, \ldots, D$ et $Vtr(X_j t^i)$, $i = 1, \ldots, D-1$, $j = 1, \ldots, n$
- **Étape 4** : Déduction des polynômes (formule de Newton étendue)

$$f(t,T), h(X_1,t,T), \ldots, h(X_n,t,T)$$

(notations du lemme 3.1.1) et de

$$g(t,t) = f'/gcd(f(t,T), f'(t,T))$$

- **Étape 5** : Pour $i = 1, \ldots, n$, calcul de

$$g_i(t,T) = h(X_i,t,T)/gcd(f(t,T), f'(t,T))$$

- **Étape 6** : Si $degre(g) = D - 1$, t est forcément séparant. Sinon, on effectue le test de l'élément séparant (lemme 3.3.1). Si t n'est pas séparant, retirer t de $\mathcal{U}$ et retourner à l'étape **1** (Entier premier de mauvaise réduction).

3.5.3 Complexité

L'étude de complexité de l'algorithme RUR-Int est un peu plus ardue que celle de l'algorithme RUR. Pour distinguer le coût des opérations modulaires de celui des opérations sur les entiers, nous sommes conduits à parler *d'opérations machine*. Le coût d'une opération de base sur des entiers de taille $\mathcal{T}$ sera noté $M(\mathcal{T}) > \mathcal{T}$, celui des opérations de base dans $GF(p)$ sera supposé unitaire et celui d'une opération faisant intervenir un entier de taille $\mathcal{T}$ et une entier premier p sera supposé proportionnel à $\mathcal{T}$ (voir par exemple [DST93]).

Rappelons que si $\mathcal{T}$ désigne la taille des entiers intervenant dans les M_{X_i} , $i = 1, \ldots, n$, l'algorithme RUR exige $O(D^3M(D\mathcal{T})+nD^2M(D^2\mathcal{T}))$ opérations machine si un vecteur séparant est connu et $O(nD^2(D^3M(D\mathcal{T}) + nD^2M(D^2\mathcal{T})))$ sinon (on néglige les termes logarithmiques). Rappelons également que si l'on suppose que les coefficients d'un pgcd sont en général plus petits que ceux des polynômes considérés, cette complexité devient $O(M(D\mathcal{T})(D^3 + nD^2))$ lorsqu'un élément séparant est connu et $O(nD^2M(D\mathcal{T})(D^3 + nD^2))$ sinon.

En pratique, pour obtenir des expressions finales plus simples, nous devons considérer en priorité les polynômes de la forme X_i , $i = 1 \ldots n$ comme séparant potentiellement $V(\mathcal{I})$. Ainsi, même si un polynôme t générique sépare $V(\mathcal{I})$, il est toutefois fréquent que celui-ci ne soit pas de la forme X_i , $i = 1 \ldots n$ (l'idéal n'est pas en position générique). Dans ce cas, l'algorithme RUR sera exécuté (au moins $n+1$ fois) dans sa totalité en prenant t dans la famille $\mathcal{U} = \{X_1, \ldots, X_n, X_1 + iX_2 + \ldots + i^{n-1}X_n$, $i = 1 \ldots (n-1)d(d-1)/2\}$ jusqu'à l'obtention d'une représentation univariée de $\mathcal{I}$. Une bonne estimation de la complexité pratique de l'algorithme RUR est alors $O(nM(D\mathcal{T})(D^3 + nD^2))$ opérations machine.

Proposition 3.5.6 *Si $p > nD^2$ est un entier premier de bonne réduction pour la représentation univariée rationnelle (c'est à dire si il est $\mathcal{G}$-compatible pour $>$ et si $\forall t \in \mathcal{U}$, $deg(gcd(P_t, P'_t)) = deg(gcd(\phi_p(P_t), \phi_p(P'_t))))$ alors, en effectuant la prédiction d'un élément séparant dans $GF(p)$, le coût de l'algorithme RUR-Int est $O((D^3 + nD^2)M(\mathcal{T}))$ pour ce choix de p.*

Preuve : Supposons qu'un entier premier de bonne réduction soit connu. Si on ne tient pas compte des manipulations dues au calcul modulaire, la complexité de l'algorithme est $O(D^3M(D\mathcal{T}) + nD^2M(D^2\mathcal{T}))$ opérations machine (complexité de l'algorithme RUR-Int dans le cas ou un élément séparant est connu).

Nous pouvons dors et déjà négliger le coût des calculs occasionnés par l'application de ϕ_p sur :

- La forme quadratique de Hermite : réduction modulo p de D^2 entiers de taille $O(D\mathcal{T})$, car si la taille de p est $O(1)$, le coût de cette opération est alors $O(D^3\mathcal{T})$ (voir par exemple [DST93]).

- Les matrices $\phi_p(M_{X_i})$: réduction modulo p de nD^2 entiers de taille $O(\mathcal{T})$.

Vérifier que p est $\mathcal{G}$-compatible nécessite de diviser le terme de tête (de taille $O(\mathcal{T})$ de chaque élément de la base de Groebner par p. Le nombre d'éléments de la base de Groebner étant majoré par nD le coût de cette opération est $O(nD\mathcal{T})$. La recherche d'un élément séparant $V(\langle\phi_p(\mathcal{G})\rangle)$ nécessite le calcul du nombre de zéros $V(\langle\phi_p(\mathcal{G})\rangle)$ par la méthode de Hermite, soit un coût de $O(D^3)$ opérations machine. Il faut ensuite calculer au plus nD^2 polynômes caractéristiques pour déterminer l'élément séparant, ce qui requiert nD^2D^3 opérations arithmétiques machine par la méthode de Hessenberg.

En terme d'opérations machine, toutes ces opérations peuvent être négligées par rapport à $O(D^3M(D\mathcal{T}) + nD^2M(D^2\mathcal{T}))$. □

En pratique, le nombre d'entiers premiers de mauvaise réduction est très faible lorsque les entiers premiers considérés sont suffisamment grands (codables toutefois sur un mot machine - i.e inférieurs à 2^{32} sur une station 32 bits) et cette complexité est à l'image des observations. En effet, sur notre panel d'exemples, jamais aucun entier de mauvaise réduction n'a été détecté. Il est intéressant alors de constater que, **en pratique, le problème de l'élément séparant n'influe pas sur le comportement global de l'algorithme**.
Dans l'absolu, le nombre d'entiers premiers de mauvaise réduction est en $O(D^2\mathcal{T})$, ainsi les étapes **1-a** et **1-b** seront répétées $O(D^2\mathcal{T})$ alors que les autres le seront au plus D fois (le nombre d'éléments de $V(\mathcal{G})$ croit à chaque étape). Ceci nous donne alors une complexité théorique en $O(D^3(D^2\mathcal{T}^2 + DM(D\mathcal{T}) + nM(D^2\mathcal{T})))$ opérations machine.

3.6 Comportement pratique

Dans tous les exemples qui suivent, la première variable est séparante. Nous pouvons donc comparer le calcul de la représentation univariée rationnelle à celui d'une base de Groebner lexicographique (voir proposition 3.2.3). Nous nous attardons bien sûr sur les temps de calcul mais surtout sur la taille de la représentation univariée fournie, qui conditionera l'utilisation de la sortie de l'algorithme.

Le calcul direct d'une base de Groebner lexicographique par l'algorithme de Buchberger étant en pratique très coûteux, nous comparerons les temps de calcul de la représentation univariée rationnelle à celui d'un algorithme calculant les bases de Groebner lexicographiques par changement d'ordre. Le principe est de calculer un base de Groebner pour un ordre du degré (efficace en pratique - voir [Fau94]) et d'en déduire une base lexicographique par des techniques d'algèbre linéaire (algorithme FGLM - [FGLM94]).

Pour que cette comparaison soit juste, la méthode utilisée pour le calcul de la table de multiplication nécessaire à notre algorithme prend en entrée la même base de Groebner que l'algorithme FGLM.
Attention : Cette première série de test à été effectuée sur une station Sun/sparc 40 Mhz nettement moins puissante que la station DEC Alpha-600 - 175 Mhz utilisée dans toutes les autres mesures et avec des versions de Gb et RealSolving datant de 1994. Compte tenu des temps de calculs observés, nous n'avons pas tenu à recommencer les mesures. En particulier, la version de l'algorithme RUR utilisée n'incluait pas la recherche modulaire d'éléments séparants.

Legende :

- G.B : Calcul de bases de Gröbner pour l'ordre *degré lexicographique inverse* via Gb.
- MT : Une table de multiplication de $K[X_1, \ldots, X_n]/\mathcal{I}$.
- RUR : Calcul d'une représentation univariée rationnelle du système.
- Racines :
  - R : nombre de racines réelles distinctes.
  - C : nombre de racines complexes distinctes.
- FGLM : Algorithme FGLM pour le calcul des bases de Groebner lexicographiques par changement d'ordre (fonction *totolex* de Gb).
- T. : temps de calcul en secondes
- X : Calcul arrêté après plus de 12 heures.
- L. : taille binaire du plus grand entier apparaissant dans le résultat (i.e: le plus petit entier i tel que 2^{32i} soit supérieur à la valeur absolue de l'entier considéré).

	G.B.		MT		RUR		MT+RUR	FGLM		Racines	
Nom	T.	L.	T.	L.	T.	L.	T.	T.	L.	R	C
cassou	28	5	0.8	6	0.2	4	1	1.6	12	4	16
sendra	0.2	5	4	16	4	5	8	2379	78	6	46
katsura 5	1.2	7	9	15	10	4	19	1065	73	18	32
katsura 7	631	10	1672	15	662	19	2334	X		44	128
discret 3	6	9	930	13	376	26	1306	X		128	128
planM2I	6	16	15	26	23	26	38	2846	274	0	32
planMV	4	19	3	27	6	23	9	92	127	0	16
planN3I	15	18	34	27	29	32	63	12711	431	0	40
spatV2A	221	21	30	72	60	30	90	3088	365	0	16
spatAV2I	5035	65	144	101	144	48	288	48432	543	0	24
spat2A2I	757	43	186	66	85	32	171	X		0	32

On peut remarquer que la taille du résultat à la sortie de la représentation univariée rationnelle est nettement inférieure à celle obtenue à la sortie de *FGLM*, le rapport allant de 1 à 10. Si $(f, g, g_1, \ldots, g_n)$ est une représentation univariée rationnelle de $\mathcal{I}$, il est possible de déduire une base lexicographique en multipliant les polynômes g_i intervenant dans la représentation univariée rationnelle par l'inverse du polynôme g modulo f (corollaire 3.2.3). Par exemple, l'application de l'algorithme d'Euclide pour faire ce calcul entraine une croissance polynômiale des coefficients, cette observation n'est donc pas surprenante.

Les techniques d'algèbre linéaire employées lors du calcul de $FGLM$ sont de complexités (en nombre d'opérations arithmétiques) sensiblement équivalentes aux méthodes employées dans le calcul de la représentation univariée rationnelle . Par conséquent, les fortes différences de temps de calcul ne peuvent s'expliquer que par un meilleur contrôle de la taille des entiers dans l'algorithme représentation univariée rationnelle .

Remarquons enfin que le calcul des prérequis (base de Groebner + table de multiplication) est, dans bon nombre de cas, au moins aussi coûteux en temps que le calcul de la représentation univariée rationnelle.
Nous nous attachons maintenant à étudier de façon plus détaillée le comportement de l'algorithme RUR-Int.
Les performances actuelles, sur station station DEC Alpha-600 - 175 Mhz , de l'algorithme RUR-Int sur notre série d'exemples habituelle sont données par la table suivante :

	GR	MT	RUR		MT+RUR
Ex.	L.	T.	T	L.	T.
e1	1	1,31	3,65	9	4,96
e2	5	0,24	0,43	1	0,67
e3	3	0,73	2,1	5	2,83
e4	6	2,68	3,4	4	6,08
e5	10	401,52	114,93	19	516,45
e6	1	2,31	3,15	5	5,46
e7	2	143,79	116,75	20	260,54
e8	9	223,81	84,1	26	307,92
e9	7	6,52	5,71	32	12,23
e10	51	6,28	3,46	30	9,74
e11	43	26,52	16,79	38	43,31
e12	65	26,28	27,54	53	53,82
e13	91	180,64	460	40	640,64
e14	101	219,69	135,2	77	354,89
e15	5	9,87	3,65	8	13,52

La figure 3.1 montre que la répartition des temps de calculs entre la table de multiplication et le calcul de la représentation unvariée rationnelle est relativement équitable.

En ce qui concerne les tailles de coefficients, la figure 3.2 montre une croissance des coefficients est plus forte dans l'expression de la représentation univariée rationnelle lorsque l'élément séparant est non trivial (exemples e1,e6,e8). Enfin, on peut également remarquer que la taille des coefficients intervenant dans la représentation univariée rationnelle peut être inférieure à celle des coefficients intervenant

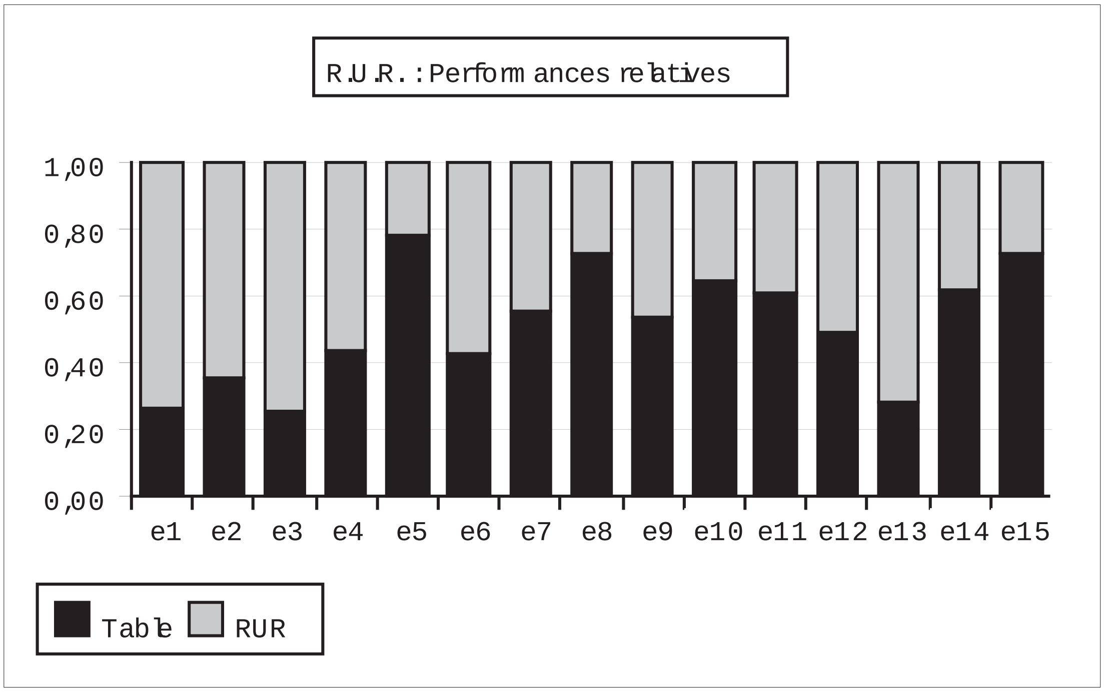

Figure 3.1: Table / RUR : temps de calcul

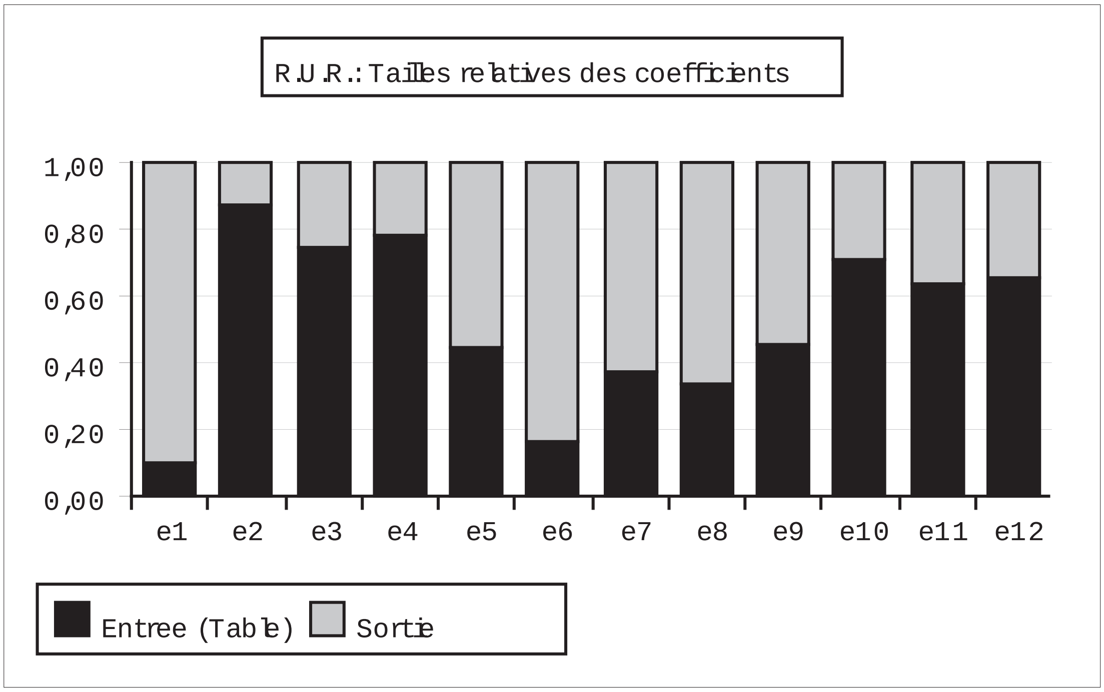

Figure 3.2: Table / RUR : croissance des coefficients

dans la base de Groebner.

Chapitre 4

Algorithmes pour le comptage des racines réelles et complexes

Résumé

Le comptage des zéros réels était notre principal souci en débutant ce travail, le premier algorithme mis au point ayant d'ailleurs été l'algorithme de Hermite.

Comme montré dans [PRS93] ou [BW92], le nombre de zéros réel (resp. complexes) d'un idéal zéro-dimensionnel $\mathcal{I} \subset K[X_1, \ldots X_n]$ peut être simplement exprimé par la signature d'une forme quadratique (forme quadratique de Hermite généralisée), définie en termes de traces d'endomorphismes simples de $K[X_1, \ldots, X_n]/\mathcal{I}$. Deux problèmes se posaient en pratique :

- La construction de la forme quadratique, les outils standards ne permettant pas d'effectuer ce calcul de manière efficace,
- La réduction de la forme quadratique.

La table de multiplication décrite dans le chapitre 2 apporte une réponse au premier point.

En ce qui concerne le deuxième point, l'algorithme standard de Gaus n'est pas suffisant pour effectuer certains calculs. En supposant que K soit le corps des fractions d'un anneau Z, un simple changement de variables nous permet de supposer que la forme quadratique à réduire est à coefficients dans Z. En généralisant les identités de Bareiss (voir [B68]) au cas des formes quadratiques, nous proposons une nouvelle méthode permettant, pour une matrice de dimension D, d'effectuer la réduction en $O(D^3)$ opérations arithmétiques dans Z (et non dans K, ce qui diminue très nettement le coût des opérations de base), tout en contrôlant efficacement la taille des calculs intermédiaires.

En combinant la table de multiplication et ce nouvel algorithme, nous obtenons une méthode très efficace, en pratique, pour compter le nombre de zéros réels (ou

complexes) d'un idéal zéro-dimensionnel donné. Nous obtenons une complexité théorique meilleure que dans [Moel93] ($O(D^4+nD^3)$ opérations arithmétiques contre $O(D^5)$) et plus précise que celle présentée dans [PRS93].

Comme l'illustre le chapitre 3, une partie de notre travail à consisté à réduire l'étude de systèmes zéro-dimensionnels à celle de polynômes en une variable, pour lesquels beaucoup d'outils ont déjà été développés. Nous avons choisi de rappeler brièvement (des études plus détaillées sont présentes dans [Rio], [BCL82], ou encore [GLRR90]) deux méthodes, éprouvées en pratique, nous semblant pour l'heure parmi les plus efficaces (Sturm-Habicht et Uspensky). Ceci nous servira de base pour comparer, en termes de complexité mais également d'un point de vue pratique, deux types de stratégies pour le comptage des zéros réels d'un système zéro-dimensionnel :

- Traitement direct par l'algorithme de Hermite
- Réduction à un problème plus simple (représentation univariée rationnelle - rappelons que les racines réelles sont préservées par cette transformation), puis application d'un d'un algorithme traitant les polynômes en une variable.

Enfin, nous présentons une évolution de l'algorithme *Inégalités simultanées* proposé dans [RS90], dont le but est de déterminer, pour une famille de polynômes $Q_1, \ldots, Q_s \subset K[X_1, \ldots, X_n]$ donnée, le cardinal de toutes les conditions de signes réalisables par les polynômes Q_i en les zéros d'un idéal $I \subset K[X_1, \ldots, X_n]$ (zéro-dimensionnel), c'est à dire le cardinal de tous les ensembles de la forme

$$\{\alpha \in V(\mathcal{I}) \ , \ et \ , \ \forall i \ , \ Sign(Q_i(\alpha)) = \epsilon_i\}$$

où les ϵ_i sont à valeurs dans $\{+, 0, -\}$. Cette méthode, mise au point en collaboration avec Guadaluppe Trujillo (Université de Cantabria), nécessite l'utilisation d'une *boite noire* permettant de calculer les quantités de la forme :

$$\sharp\{\alpha \in V(I) \cap R \ , \ Q(\alpha) > 0\} - \sharp\{\alpha \in V(I) \cap R \ , \ Q(\alpha) < 0\},$$

pour $Q \in K[X_1, \ldots, X_n]$, soit par exemple l'algorithme de Sturm-Habicht lorsque $n = 1$ ou l'algorithme de Hermite, lorsque $n > 1$.

4.1 Cas des polynômes en une variable

Lorsqu'une représentation univariée rationnelle d'un idéal zéro dimensionnel est connue, le comptage des zéros réels peut s'effectuer sur un polynôme en une variable (proposition 3.2.1). Cette partie est donc dédiée à la description des algorithmes les plus performants du moment pour le comptage des zéros réels de polynômes en une variable, utilisant une arithmétique exacte.
Deux points de vus sont abordés :

- Le comptage par une méthode *à la Sturm*.
- L'isolation des racines (par exemple par intervalles rationnels dans le cas de polynômes à coefficients entiers).

Le traitement des polynômes en une variable sort du cadre de cette thèse. Nous ne rappelons donc ici que les deux algorithmes, dont le comportement pratique est déjà éprouvé, et qui nous semblent les plus efficaces. Notons, à ce sujet, qu'un comparatif très détaillé est effectué dans [Rio].

Par exemple, bien que nous en ayons présenté une généralisation, la méthode de Hermite est plus connue pour les polynômes en une variable. Dans ce cas, la forme quadratique de Hermite est de type *Hankel*, ce qui rend sa réduction plus aisée. Une solution utilisant la base de Horner peut également être proposée, mais dans tous les cas, la complexité théorique, autant que pratique, de cette méthode est au mieux équivalente à celle de la méthode de Sturm-Habicht que nous allons rappeler (voir [Her53], [Roy1]).

Comme les théorèmes cités ne seront pas re-démontrés, des références précises seront indiquées.

4.1.1 Comptage

Dans cette partie, Z est un anneau ordonné, K le corps des fractions de Z, de clôture réelle R.

Pour $P = \sum_{i=0,\ldots,d_P} a_i T^{d_P-i}$ et $Q = \sum_{i=0,\ldots,d_Q} b_i T^{d_Q-i}$ dans $Z[T]$, on définit, pour $i = 0\ldots, K < d_P + d_Q$:

- $P_0 = P$
- $P_1 = QP'$
- $P_{i+1} = -reste(P_{i-1}, P_i)$

et on pose : $Stu(P,Q) = [P_0, \ldots, P_K]$ avec $K = sup(i = 0, \ldots, d_P + d_Q - 1 \mid P_i \neq 0)$ (suite de Sturm).

Pour $a \in R$ tel que $P(a) \neq 0$, on note $V_{Stu(P,Q)}(a)$ le nombre de changements de signes observés dans la suite $P_0(a), \ldots, P_K(a)$, en ne tenant pas compte des quantités nulles.
On rappelle alors le théorème de Sturm :

Théorème 4.1.1 *$P = \sum_{i=0,\ldots,d_P} a_i T^{d_P-i}$ et $Q = \sum_{i=0,\ldots,d_Q} b_i T^{d_Q-i}$ deux polynômes de $Z[T]$ et $a < b$ deux éléments de R tels que $P(a)$ et $P(b)$ soient non nuls. Dans ce cas, en notant $V_R(P)$ l'ensemble des zéros de P dans R :*

$$\begin{aligned} V_{Stu(P,Q)}(a) - V_{Stu(P,Q)}(b) = \ & \sharp\{x \in V_R(P) \cap]a,b[\mid Q(x) > 0\} \\ & -\sharp\{x \in V_R(P) \cap]a,b[\mid Q(x) < 0\} \end{aligned}$$

En particulier,

$$\begin{aligned} V_{Stu(P,Q)}(-\infty) - V_{Stu(P,Q)}(+\infty) = \ & \sharp\{x \in V_R(P) \mid Q(x) > 0\} \\ & -\sharp\{x \in V_R(P) \mid Q(x) < 0\} \end{aligned},$$

ou encore :

$$V_{Stu(P,1)}(-\infty) - V_{Stu(P,1)}(+\infty) = \sharp\{x \in V_R(P)\}$$

Lorsque P et Q sont dans $Z[T]$, l'inconvénient du calcul de la suite $Stu(P,Q)$ est que les opérations sont effectuées dans $K[T]$, ce qui rend cette méthode inefficace en pratique.
Ceci peut être évité, en modifiant légèrement la définition des polynômes P_i :

Si $p = sup(deg(P), deg(Q)+1)$, pour tout $j = 1\ldots,p$, on considère la suite $[H_p, \ldots, H_0]$ définie par :

- $H_p = P$,
- $H_{p-1} = P'Q$,

Itérativement, pour j décroissant de p à 2, en supposant que $deg(H_j) > 0$, et en posant $k = deg(H_{j-1}) \leq j-1$,

- $lc(H_j)^{j-1-k} H_k = \delta_{j-k} Co_k(H_{j-1})^{j-1-k} H_{j-1}$, si $k < j-1$,
- $H_l = 0$ si $k < j-2$ et $k < l < j-2$,
- $lc(H_j)^{j+1-k} H_{k-1} = \delta_{j-k+2} PReste(H_j, H_{j-1})$

où :

- $Co_k(P)$ désigne le terme de degré k d'un polynôme P,
- $\delta_k = (-1)^{(k(k-1)/2)}$,
- $PReste(H_{i-1}, H_i) = reste(|lc(H_{i-1})|^{max(deg(H_{i-1})+1, deg(H_i))-deg(H_{i-1})+1} H_i, H_{i-1})$,

- $lc(P)$ désigne le coefficient du terme de plus haut degré d'un polynôme P

On définit ainsi la suite de *Sturm-Habicht* (dont les éléments sont cette fois dans $\mathbb{Z}[T]$) associée à P et Q :

$$StuHa(P,Q) = [P_0 = H_p, \ldots, P_d = H_0].$$

Pour $a \in R$ tel que $P(a) \neq 0$, en comptant de façon différente le nombre de changements de signes dans la suite $[P_0(a), \ldots, P_d(a)]$ (i.e: les éléments nuls ne sont plus systématiquement négligés : $[+,0,0,-]$ et $[-,0,0,+]$ comptent pour 1 changement et $[-,0,0,-]$ et $[-,0,0,-]$ comptent pour 2 changements), on définit $V_{StuHa(P,Q)}(a)$, le nombre de changements de signes observés dans la suite $P_0(a), \ldots, P_d(a)$. Le théorème de Sturm devient alors (voir par exemple [Roy1]) :

Théorème 4.1.2 *$P = \sum_{i=0,\ldots,d_P} a_i T^{d_P - i}$ et $Q = \sum_{i=0,\ldots,d_Q} b_i T^{d_Q-i}$ deux polynômes de $Z[T]$ et $a < b$ deux éléments de R tels que $P(a)$ et $P(b)$ soient non nuls. Dans ce cas, en notant $V_R(P)$ l'ensemble des zéros de P dans R :*

$$\begin{aligned} V_{StuHa(P,Q)}(a) - V_{StuHa(P,Q)}(b) = \; & \sharp\{x \in V_R(P) \cap]a,b[\mid Q(x) > 0\} \\ & \sharp - \{x \in V_R(P) \cap]a,b[\mid Q(x) < 0\} \end{aligned}$$

En particulier,

$$\begin{aligned} V_{StuHa(P,Q)}(-\infty) - V_{StuHa(P,Q)}(+\infty) = \; & \sharp\{x \in V_R(P) \mid Q(x) > 0\} \\ & -\sharp\{x \in V_R(P) \mid Q(x) < 0\}, \text{'} \end{aligned}$$

ou encore :

$$V_{StuHa(P,1)}(-\infty) - V_{StuHa(P,1)}(+\infty) = \sharp\{x \in V_R(P)\}$$

Ainsi, l'algorithme calculant la suite $StuHa(P,Q)$, et par conséquent le nombre de zéros réels de P si $Q = 1$, nécessitera $O((d_P + d_Q)^2)$ opérations arithmétiques élémentaires dans Z.

Il esiste un lien évident entre la suite de Sturm de deux polynômes et leur pgcd. De la même façon, il existe un lien entre la suite de Sturm-Habicht et les sous-résultants. En effet, on peut montrer (voir par exemple [Roy1]) que les coefficients des polyômes de la suite $StuHa(P,Q)$ s'expriment simplement (au signe près) comme des déterminants de matrices extraites de la matrice de Sylvester :

$$\begin{pmatrix}
a_0 & a_1 & a_2 & \ldots & . & . & . & a_{d_P} & 0 & . & \ldots & 0 \\
0 & a_0 & a_1 & \ldots & . & . & . & a_{d_P-1} & a_{d_P} & . & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & . & . & \ldots & 0 & a_0 & a_1 & a_2 & . & . & \ldots & a_{d_P} \\
b_0 & b_1 & . & \ldots & . & . & b_{d_Q} & 0 & . & . & \ldots & 0 \\
0 & b_0 & . & \ldots & . & . & b_{d_Q-1} & b_{d_Q} & 0 & . & \ldots & 0 \\
0 & . & . & \ldots & . & 0 & b_0 & b_1 & . & . & \ldots & b_{d_Q}
\end{pmatrix}$$

Dans le cas de polynômes à coefficients entiers, si on note $\mathcal{T}$ la taille de la représentation binaire du plus grand entier (en valeur absolue) apparaissant dans P et Q, en utilisant l'inégalité d'Hadamard, on peut alors montrer que les coefficients apparaissant au cours de calcul de la suite $StuHa(P,Q)$ sont de taille maximale $O((d_P + d_Q)(\mathcal{T} + log(d_P + d_Q)))$.

4.1.2 Isolation

Nous ne nous intéressons, dans cette partie, qu'aux polynômes à coefficients entiers. La méthode de Uspensky nécessitant le calcul de la partie sans facteurs carrés du polynôme étudié, elle n'apporte aucun progrès en pratique dans le cas de polynômes plus généraux, comparé à l'algorithme de Sturm-Habicht. En effet, la meilleure façon, en pratique, à l'heure actuelle, de calculer le pgcd de deux polynômes à coefficients dans un anneau Euclidien est d'appliquer l'algorithme des sous-résultants dont la complexité est équivallente à celle de la méthode de Sturm-Habicht (le calcul est par ailleurs pratiquement identique). Dans le cas de polynômes à coefficients entiers (ou rationnels), nous utilisons une méthode multi-modulaire extrêmement efficace (voir par exemple [DST93]), rendant ce prérequis négligeable, dans la plupart des cas, en termes de temps de calculs.

La méthode de Uspensky est basée sur la règle de Descartes : si $P = \sum_{i=0}^{d} a_i x^i$, le nombre de changements de signes $var(P)$ dans la suite $a_0, a_1, \ldots, a_d$ majore le nombre de racines réelles positives, comptées avec multiplicités, de P. Evidement, si $var(P) = 0$, P n'a pas pas de racines réelles positives, et on peut montrer de plus que si $var(P) = 1$, P admet exactement une racine réelle positive (voir [BCL82]).

L'idée de base de l'algorithme que nous présentons est due à un théorème que Uspensky ([Us48]) attribue à Vincent ([Vi36]), montrant que, après un nombre fini de transformations de la forme :

$$\hat{P}(x) = P(x+1)$$

et

$$P^{\star}(x) = (x+1)^n P(1/(x+1)),$$

le nombre de changements de signes dans les suites de coefficients apparaissant dans les divers polynômes est 0 ou 1, dès lors que P est sans facteurs carrés.

L'algorithme induit impliquait des temps de calcul exponnentiels. Ce comportement à ensuite été sensiblement amélioré par Collins et Akritas (voir [CA76]) en introduisant des opérations de bissection. Le principe est simple : on modifie un peu les changements de variables cités plus haut de sorte à ce que les polynômes considérés aient leurs racines dans $[0,1]$. Pour un polynôme P sans facteurs carrés donné,

- soit $var(P) \geq 2$ et dans ce cas, on réalise 2 changements de variables visant à étudier le nombre de zéros réels de P dans $(0, 1/2)$ et $(1/2, 1)$. Puis, modulo une deuxième série de changements de variables, on se ramène alors à l'étude du nombre de zéros réels de deux polynômes dans $(0, 1)$.
- soit $var(P) = 0$ et alors P n'admet aucune racine,
- soit $var(P) = 1$ et alors P n'admet une unique racine réelle positive.

Algorithme de Uspensky modifié

Entrée : Un polynôme sans facteurs carrés.

- **Etape 1** (borne) : Calcul de k tel que 2^k borne la valeur absolue des racines de P (par Landau-Mignotte par exemple)
- **Etape 2** (transformation) : On définit $B(x) = P(2^k x)$ si $k \geq 1$, et $B(x) = 2^{-kd} P(2^k x)$ sinon. Dans tous les cas, les racines de B sont majorées en valeur absolue par 1.
- **Etape 3** (traitement) : On applique la routine *Traitement* à B, et on note L' la liste des intervalles d'isolation des racines de B dans $[0, 1]$.
- **Etape 4** (remontée) : Dans L', on remplace chaque intervalle (a_i, b_i) par $(2^k a_i, 2^k b_i)$ et on appelle le résultat L

Sortie : L, une liste d'intervalles isolant les racines réelles positives de P.

Traitement

Entrée : Un polynôme A, de degré n, sans facteurs carrés, et ayant toutes ses racines dans $(0, 1)$

- **Etape 1** : On pose $A^\star = (x+1)^n A(1/(x+1))$ (les zéros de A dans $(0, 1)$ sont les zéros de $A^\star$ dans $(0, +\infty)$).

  Si $var(A^\star) = 0$, $L := []$ et on sort de la procédure.

  Si $var(A^\star) = 1$, $L := [(0, 1)]$ et on sort de la procédure.
- **Etape 2** : Si $A(1/2) = 0$, on rajoute $(1/2, 1/2)$ dans L et on remplace A par $A/(2x-1)$.

- **Etape 3** : On pose $A' = 2^n A(x/2)$ (les zéros de A dans $(0, 1/2)$ sont par conséquent ceux de A' dans $(0,1)$). On note alors L' la liste retournée par l'algorithme *Traitement* appelé avec A'. On remplace chaque intervalle (a_i, b_i) de L' par $(a_i/2, b_i/2)$ et on le rajoute à L.

- **Etape 4** : On pose $A'' = (x+1)A'(x)$ (les zéros de A dans $(1/2, 0)$ sont par conséquent ceux de A'' dans $(0,1)$). On note alors L' la liste retournée par l'algorithme *Traitement* appelé avec A''. On remplace chaque intervalle (a_i, b_i) de L' par $((a_i+1)/2, (b_i+1)/2)$ et on le rajoute à L.

Sortie : L, une liste d'intervalles isolant les racines réelles positives de A dans $(0, 1)$.

4.2 Cas des systèmes polynomiaux

Dans tout ce chapitre, Z désigne un anneau euclidien, dont le corps de fractions sera noté K.

Si $\mathcal{I} \subset K[X_1, \ldots, X_n]$ est un idéal zéro-dimensionnel, rappelons que la forme quadratique de Hermite généralisée associée à un polynôme $h \in K[X_1, \ldots, X_n]$ est définie sur $K[X_1, \ldots, X_n]/\mathcal{I}$ par :

$$Q_h(f) = Trace(hf^2)$$

et permet de calculer les quantités suivantes :

$$\sharp\{\alpha \in V_R(I) | h(\alpha) > 0\} - \sharp\{\alpha \in V_R(I) | h(\alpha) < 0\}$$

(signature de Q_h), avec $V_R(\mathcal{I}) = V(\mathcal{I}) \bigcap R^n$, R étant la clôture réelle de K et

$$\sharp\{\alpha \in V(I) | h(\alpha) \neq 0\}$$

(rang de Q_h).
Calculées pour $h = 1$, ces quantités nous donnent en particulier le nombre de zéros dans R^n ou dans C^n de $\mathcal{I}$.

Nous avons vu, dans le chapitre 2, comment calculer la forme quadratique de Hermite. Notre but est maintenant de montrer comment la réduire (en les supposant, modulo une petite transformation, à coefficients dans Z) pour en obtenir la signature (lorsque Z est ordonné) et le rang de façon efficace, les calculs dans Z étant en général moins coûteux que dans K.

4.2.1 Réduction de formes quadratiques

Le but initial de cette étude était de construire un algorithme rapide pour le calcul du rang et de la signature des formes quadratiques à coefficients entiers pour optimiser le plus possible la méthode de Hermite. En fait nous avons produit un algorithme général pour la réduction des formes quadratiques à coefficients dans une anneau euclidien.
L'idée est de modifier la méthode d'orthonormalisation de Gauss pour que tous les calculs se déroulent dans Z (et non K), en conservant un nombre d'opération arithmétiques de l'ordre de $O(D^3)$ (si D est la dimension de la matrice), tout en contrôlant correctement la taille des calculs intermédiaires. Une telle méthode existant pour le calcul des déterminants (méthode de Bareiss), notre travail a consisté à la généraliser pour obtenir un algorithme de réduction de formes quadratiques.

Pour toute cette partie, Z sera un anneau euclidien, K le corps des fractions qui lui est associé et $M = (a_{i,j})$ une matrice de dimension $d \times d$ à coefficients dans Z.

Notation 4.2.1 *Pour tout entier k , $0 \leq k \leq d$, et tout couple d'entiers (i,j) $i > k$, $j > k$, nous noterons $D^k_{i,j}(M)$, le déterminant :*

$$D^k_{i,j}(M) = Det \begin{pmatrix} a_{1,1} & \dots & a_{1,k} & a_{1,j} \\ \dots & \dots & \dots & \dots \\ a_{k,1} & \dots & a_{k,k} & a_{k,j} \\ a_{i,1} & \dots & a_{i,k} & a_{i,j} \end{pmatrix},$$

et nous supposerons que $D^0_{i,j} = a_{i,j}$, $D^{-1}_{0,0} = 1$.

Pour éviter d'alourdir les notations, nous travaillerons *à permutation de lignes et de colonnes près*, en utilisant la convention suivante :

Convention 4.2.1 *On peut supposer sans perte de généralité que:*

- $D^{k-1}_{k,k}(M) = 0 \Rightarrow D^{k-1}_{i,i}(M) = 0$, $\forall k \leq i \leq d$ *(i.e. Si le k^e mineur principal de M est nul, les suivants le sont également).*
- $(D^{k-1}_{k,k}(M) = 0 \;\&\; D^{k-1}_{k+1,k}(M) = 0) \Rightarrow D^{k-1}_{i,j}(M) = 0$, $\forall k \leq i,j \leq d$

Méthode de Bareiss pour le calcul des Déterminants

Dans l'article [B68], Bareiss décrit un algorithme pour le calcul des déterminants, basé sur la relation de Sylvester :

Proposition 4.2.1 *Pour tout indice k tel que $D^{k-2}_{k-1,k-1}(M) \neq 0$,*

$$D^k_{i,j}(M) = \frac{1}{D^{k-2}_{k-1,k-1}(M)} det \begin{pmatrix} D^{k-1}_{k,k}(M) & D^{k-1}_{k,j}(M) \\ D^{k-1}_{i,k}(M) & D^{k-1}_{i,j}(M) \end{pmatrix}$$

Comme $D^d_{d-1,d-1}(M) = det(M)$, cette formule nous donne alors un algorithme simple pour le calcul de déterminant (en tenant compte de la convention 4.2.1) :

Procedure Déterminant
  Entrée : M_0, *une matrice de dimension d à coefficients dans Z.*
  Sortie : *Le déterminant de* M_0
Début
  $M := M_0$;i:=1;PrecPivot:=1;Fin:=Faux
  tant que ($i < d$ et non(Fin)) faire
    Si ($M(i,i) \neq 0$) alors
      Pivot:=$M(i,i)$;
      Pour (k:=i+1 jqa d) et (l:=i+1 jqa d) faire
        $M(l,k) := \frac{M(k,l)*Pivot - M(i,k)*M(i,l)}{PrecPivot}$;
      PrecPivot:=Pivot;
    Sinon Fin:=Vrai;
  Si ($i = d$ et $M(d,d) \neq 0$) alors Retourner($M(d,d)$);
  Sinon Retourner(0);
Fin.

Preuve de l'algorithme : En utilisant la proposition 4.2.1,on remarque (par récurrence), qu'après le i^e passage dans la boucle pincipale (i.e. *tant que*) $M(l,k) = D^i_{l,k}(M)$. Si la matrice M_0 respecte la convention 4.2.1, la preuve de l'algorithme est alors immédiate. □

La proposition 4.2.1 assure que tous les calculs effectués dans l'algorithme précédent sont dans Z. Dans le cas où Z est l'anneau des entiers, rappelons que le déterminant d'une matrice $M = (a_{i,j})^{j=1,\ldots,d}_{i=1,\ldots,d}$ est borné, en valeur absolue, par $\prod_{i=1,\ldots,d} \sum_{j=1,\ldots,d} a^2_{i,j}$ (inégalité d'Hadamard - voir par exemple [BCL82] ou [DST93]). Dans l'algorithme *Determinant* décrit ci-dessus, les éléments de toutes les matrices intervenant dans les calculs intermédiaires sont des déterminants extraits de la matrice initiale, ceci nous permet alors de d'évaluer la taille de la représentation binaire des entiers intervenant en cours de calcul :

Proposition 4.2.2 *Soit M est une matrice à coefficients entiers de dimension d. Si $\mathcal{T}$ est la taille maximale des coefficients de M (représentation binaire), la taille maximale des entiers intervenant dans l'algorithme Déterminant est en $O(d(\mathcal{T} + log(d)))$.*

Orthonormalisation de Gauss

Soit $Q = \sum_{i=1}^d \sum_{j=1}^d b_{i,j} Y_i Y_j$ une forme quadratique à coefficients dans Z.

Les deux transformations suivantes seront à la base de toute notre étude :

Definition 4.2.1 *Soit Δ_k la transformation bilinéaire inversible définie par :*

$$\begin{cases} \Delta_k(Y_{k+1}) = & Y_{k+1} - Y_k \\ \Delta_k(Y_i) = & Y_i \ , \ \forall i \neq (k+1) \end{cases}$$

On note alors :

- $\Delta_k(b_{k,k}) = 2 \cdot b_{k,k+1} + b_{k+1,k+1} + b_{k,k}; \forall i \neq k$
- $\Delta_k(b_{k,i}) = b_{k,i} + b_{k+1,i}; \forall i, j \neq k$
- $\Delta_k(b_{i,j}) = b_{i,j}$

de telle sorte que : $Q(\Delta_k(X)) = \sum_{i=1}^{d} \sum_{j=1}^{d} \Delta_k(b_{i,j}) \Delta_k(Y_i) \Delta_k(Y_j)$.
Par abus de notations, nous écrirons $\Delta_k(M) = (\Delta_k(b_{i,j}))_{i,j}$ *pour toute matrice* $M = (b_{i,j})_{i,j}$.

Definition 4.2.2 *Si* $b_{k,k} \neq 0$*, on définit un changement de variables inversible* Γ_k *:*

$$\begin{cases} \Gamma_k(Y_k) & = b_{k,k}^{-1} \sum_{i=k}^{d} b_{k,i} Y_i \\ \Gamma_k(Y_i) & = Y_i \ \forall i \neq k \end{cases}$$

On distingue deux cas dans le procédé d'orthogonalisation de Gauss appliqué à F :

- Le cas favorable : $(a_{1,1}, \ldots a_{d,d}) \neq (0, \ldots, 0)$. Conformément aux conventions (4.2.1), $a_{1,1} \neq 0$, F peut alors s'écrire : $F = a_{1,1} \cdot Y_1^2 + F'$, avec : $F' = \sum_{i=2}^{d} \sum_{j=2}^{d} b_{i,j} Y_i Y_j$, $b_{ij} = a_{ij} - \frac{a_{1i} a_{1j}}{a_{11}}$ et $Y = \Gamma_1(X)$.
- Le cas défavorable : $(a_{1,1}, \ldots a_{d,d}) = (0, \ldots, 0)$. En appliquant Δ_1 à F, la condition $a_{12} \neq 0$ assure que nous nous retrouverons alors dans le cas favorable.

La suite de la réduction se fait par induction, en traitant à l'étape suivante la forme quadratique F' (en utilisant Δ_k et Γ_k lors de la k^{ieme} étape).

Lorque nous nous trouvons dans un cas favorable, les formules sont similaires à celles utilisées dans l'algorithme du pivot de Gauss pour le calcul des déterminants. Ainsi, dans le cas particulier où aucune étape défaforable n'est impliquée dans la méthode de Gauss, une adaptation immédiate de la méthode de Bareiss (pour le calcul des déterminants) fournit un algorithme permettant d'effectuer tous les calculs dans Z.

Modification de la matrice initiale

L'idée est d'étudier le comportement des mineurs $D_{i,j}^k(M)$ si l'on effectue préalablement les changements de variables induits par les cas défavorables de l'algorithme de Gauss.

Definition 4.2.3 *Soit $F = \sum_{i=1}^{d}\sum_{j=1}^{d} a_{i,j}X_iX_j$ une forme quadratique à coefficients dans Z de matrice M. On définit:*

- *Pour toute matrice M de dimension d donnée, on note $l(M) = Inf\{l = 1\dots d\ ,\ D_{l,l}^{l-1}(M) = 0\}$ en posant $l(M) = d+1$ si $D_{l,l}^{l-1}(M) \neq 0\ ,\ \forall l = 1\dots d$.*
- *$G = \sum_{i=1}^{d}\sum_{j=1}^{d} \Delta_{l(M)}(a_{i,j})\Delta_{l(M)}(X_i)\Delta_{l(M)}(X_j)$ et N la matrice de G.*

Lemme 4.2.1 *Pour tout k , $0 \leq k \leq d$, $k \neq l(M)$ et tout couple $i, j > k$ on a :*

A. $D_{i,j}^{k}(N) = \Delta_l(D_{i,j}^{k}(M))$, $\forall i, j > k$

B. $l(N) > l(M) + 1$.

Preuve : Notons $l = l(M)$.

A. Ces relations sont immédiates pour $i, j \neq l$, par définition des $D_{i,j}^{k}(N)$.
En développant $D_{i,l}^{k}(N)$ et $D_{l,l}^{k}(N)$, obtient : $D_{i,l}^{k}(N) = D_{i,l}^{k}(M) + D_{i,l+1}^{k}(M)$.
et $D_{l,l}^{k}(N) = D_{l,l}^{k}(M) + 2\cdot D_{l,l+1}^{k}(M) + D_{l+1,l+1}^{k}(M)$

B. D'après la définition de l et la convention (4.2.1), $D_{l,l}^{l-1}(N) = 2\cdot D_{l,l+1}^{l-1}(M) \neq 0$, donc $l(N) > l$. De plus, $D_{l-1,l-1}^{l-2}(N) = D_{l-1,l-1}^{l-2}(M) \neq 0$, la proposition 4.2.1 nous donne alors $D_{l+1,l+1}^{l}(N) = D_{l+1,l+1}^{l}(M) \neq 0$, ce qui implique $l(N) > l(M) + 1$.

□

Etendons maintenant ce résultat :

Definition 4.2.4 *En posant $M_0 = M$ et $l_0 = l(M)$, on peut définir récursivement des indices l_m et des matrices M_m, $m > 0$:*

- *$M_{m+1} = \Delta_{l_m}(M_m)$, $l_{m+1} = l(M_{m+1})$*
- *$l_m \leq d+1$*

On définit également d', le plus grand indice tel que $D_{d',d'}^{d'-1}(M_{l_s}) \neq 0$, avec $s = max\{m\ , l_m < d\}$.

En considérant cette défintion, on généralise le lemme (4.2.1) :

Proposition 4.2.3 *En utilisant la definition 4.2.4, $\forall k\ ,\ 0 < k < d\ ,\ k \neq l_m\ ,\forall i, j > k$, $D_{i,j}^{k}(M_{m+1}) = \Delta_{l_m}(D_{i,j}^{k}(M_m))$, et $l_{m+1} > l_m + 1$.*

L'Algorithme

D'après la proposition 4.2.3, et la définition 4.2.4, la matrice M_s déduite de M n'engendrerait aucun cas défavorable si on lui appliquait l'algorithme de Gauss. Le but est maintenant de simuler la construction et le traitement de toutes les matrices M_m en conservant un nombre d'opérations arithmétiques de base en $O(d^3)$.

Definition 4.2.5 *On définit* $m(k) = inf\{m \in [0 \dots s]\ ,\ k < l_m\}$, $N_k = M_{m(k)}$,

$$f_k = \sum_{i=k+1}^{d} \sum_{j=k+1}^{d} D_{i,j}^{k}(N_k) X_i^{(k)} X_j^{(k)}$$

Definition 4.2.6 *On définit un changement de variables inversible* Φ_k *en posant :* $\Phi_k = \Gamma_k \circ \Delta_k$ *si* $k = l_m$, $m = 1 \dots s$ *et* $\Phi_k = \Gamma_k$ *sinon.*

Lemme 4.2.2 *Pour* $0 \leq k < d'$, *en posant* $X^{k+1} = \Phi_{k+1}(X^k)$ *et* $D_{0,0}^{(-1)}(M) = 1$,

$$f_k(\Phi_{k+1}(X^{(k)})) = D_{k+1,k+1}^{k}(N_{k+1}) \left(X_{k+1,k+1}^{(k+1)}\right)^2 + \frac{D_{k,k}^{k-1}(N_{k+1})}{D_{k+1,k+1}^{k}(N_{k+1})} f_{k+1}$$

Preuve : On distingue deux cas :

- Si $k \neq l_m - 1$, $m = 0 \dots s$, on pose $Y = X^{k-1}$ et $b_{i,j} = D_{i,j}^{k}(N_k)$. Comme $k + 1 < l_m$, $b_{i,j} = D_{i,j}^{k}(N_{k+1})$.
- Si $k = l_m - 1$, $m \in \{0 \dots s\}$, d'après la definition de Δ_{l_m} et la proposition 4.2.3,

$$f_{l_m-1}(\Delta_{l_m}(X^{(l_m-1)})) = \sum_{i=l_m}^{d} \sum_{j=l_m}^{d} D_{i,j}^{l_m-1}(N_m) \Delta_{l_m}(X_i^{(l_m-1)}) \Delta_{l_m}(X_j^{(l_m-1)})$$

  On pose alors : $b_{i,j} = D_{i,j}^{k}(N_{k+1})$ et $Y = \Delta_{k+1}(X^k)$.

A l'aide de ces notations, on réécrit f_k de la façon suivante :

$$\begin{aligned} f_k = \ & b_{k+1,k+1}^{-1} \left(\left(\textstyle\sum_{i=k+1}^{d} b_{k+1,i} Y_i \right)^2 - \textstyle\sum_{i=k+2}^{d} \textstyle\sum_{j=k+2}^{d} b_{k+1,i} b_{k+1,j} Y_i Y_j \right) \\ & + \textstyle\sum_{i=k+2}^{d} \textstyle\sum_{j=k+2}^{d} b_{i,j} Y_i Y_j \end{aligned}$$

En appliquant maintenant $\Phi_{k+1} = \Gamma_{k+1}$ à cette expression et en utilisant la proposition 4.2.1, on obtient immédiatement le résultat annoncé. □

Regardons comment se comporte la forme quadratique de départ :

Definition 4.2.7 *Soit* $F_0 = \sum_{i=1}^{d}\sum_{j=1}^{d} a_{i,j}X_i^{(0)}X_j^{(0)} = F(X^{(0)})$*, on construit récursivement une famille de formes quadratiques* F_k *pour* $k = 1 \ldots d'$ *à l'aide de la relation :* $F_{k+1} = F_k(\Phi_{k+1}(X^{(k)}))$*, où* Φ_k *est le changement de variables de la definition 4.2.6.*

Proposition 4.2.4 *En utilisant les définitions 4.2.7 et 4.2.5, on obtient :*

$$F_k = \sum_{i=1}^{k} \frac{D_{i,i}^{i-1}(N_i)}{D_{i-1,i-1}^{i-2}(N_{i-1})}(X_{i,i}^{(k)})^2 + \frac{1}{D_{k,k}^{k-1}(N_k)} f_k$$

Dans ce cas, par définition de d' *et compte tenu des conventions (4.2.1),* $f_{d'} = 0$*. Par conséquent :* $F(\Phi_{d'} \circ \ldots \circ \Phi_1(X^{(0)})) = F_{d'} = \sum_{i=1}^{d'} \frac{D_{i,i}^{i-1}(N_i)}{D_{i-1,i-1}^{i-2}(N_{i-1})}(X_{i,i}^{(d')})^2$*.*

Preuve : Raisonnons par récurrence sur k. La relation est triviale pour $k = 0$, supposons qu'elle soit vraie pour $k > 0$. Si nous appliquons Φ_k à F_k nous obtenons :

$$F_{k+1} = \sum_{i=1}^{k} \frac{D_{i,i}^{i-1}(N_i)}{D_{i-1,i-1}^{i-2}(N_i)}(X_{i,i}^{(k)})^2 + \frac{1}{D_{k,k}^{k-1}(N_{k+1})} f_k(\Phi_{k+1}(X^{(k)}))$$

la formule finale peut alors être vérifiée en utilisant le lemme 4.2.2. □

En appliquant la proposition 4.2.4 à F, $F = F_{d'}$ et en considérant le changement de variables $(D_{i-1,i-1}^{i-2}(N_i)Y_i \longrightarrow X_i^{d'})_{i=0\ldots d'}$, nous obtenons le théorème :

Théorème 4.2.1 *Etant donnée une forme quadratique* $F = \sum_{i=1}^{d}\sum_{j=1}^{d} a_{i,j}X_iX_j$ *à coefficients dans* Z*, il existe une transformation bilinéaire inversible* $(X \to Y)$*, un indice* d' *et des matrices* $N_i = M_{m(i)}$ *(où* $m(i) = inf\{m \in [1, \ldots, s],\ i < l_m\}$*) à coefficients dans* Z *tels que :*

$$F = \sum_{i=1}^{d'} D_{i,i}^{i-1}(N_i) \cdot D_{i-1,i-1}^{i-2}(N_{i-1})Y_i^2$$

Ainsi, la donnée d'un algorithme calculant toutes les $D_{i,i}^{i-1}(N_i)$ du théorème 4.2.1 permet de calculer facilement le rang, la signature (si Z est ordonné), et de réduire une forme quadratique à coefficients dans Z (avec changement de variables dans K) en effectuant tous les calculs dans Z. Considérons alors l'algorithme suivant :

Procedure Réduction
Entrée : M_0*, une matrice symétrique de dimension* d *à* $coefficients$ *dans* Z*.*
Sortie : La liste $[D_{i,i}^{(i-1)}(N_i)\ ,\ i = 0, \ldots d']$ *associée à* M_0
Fonction annexe : Ajoute(Liste, elt) ajoute elt a la fin de Liste
Début
    $M := M_0$;i:=1;PrecPivot:=1;ListPivots:=[1];Succes:=Faux
    tant que ($i < d$ et non(Succes)) faire

Si ($M(i,i) = 0$ et $M(i,i+1) \neq 0$) alors
        Pour k:=i jqua d faire
            $M(k,i) := M(i,k) := M(i,k) + M(i+1,k)$;
Si $M(i,i) \neq 0$ alors
    Pivot:=$M(i,i)$;
    Pour (k:=i+1 jqa d) et (l:=k jqa d) faire
        $M(l,k) := M(k,l) := \frac{M(k,l)*Pivot - M(i,k)*M(i,l)}{PrecPivot}$;
    Ajoute(ListPivots,Pivot);PrecPivot:=Pivot;i:=i+1;
Sinon Succes:=Vrai;
Si ($i = d$ et $M(d,d) \neq 0$) alors Ajoute(ListPivots,M(d,d));
Retourner(ListPivots);
Fin.

La preuve de cet algorithme est évidente (compte tenu des conventions (4.2.1)) si l'on remarque qu'après chaque passage dans la boucle principale, $(M(l,m))_{l,m \geq i}$ est exactement la matrice de f_i, pour $i = 1..inf(d', d-1)$. Enfin, d'après la proposition 4.2.1, on peut voir que tous les calculs sont effectués dans Z.

Remarque 4.2.1 *Pour fournir un algorithme général, il suffit de remarquer que de simples permutations de lignes et de colonnes permettent , à chaque passage dans la boucle prncipale, de se ramener au cas décrit par les conventions (4.2.1). Les valeurs des différents mineurs demeurent inchangées, seuls leurs indices sont éventuellement permutés.*

Performances

Dans cette partie, nous comparons notre algorithme de réduction avec l'algorithme de Gauss, et nous observons son comportement lorqu'il est couplé à l'algorithme de calcul de la table de multiplication, pour le calcul du nombre de zéros réels (ou complexes) d'un système zéro-dimensionnel.

- Gauss : Orthonormalisation de Gauss classique.
- Red. : Le nouvel algorithme.
- T : temps de calcul en secondes sur une station DEC Alpha-600 - 175 Mhz
- L. : taille binaire du plus grand entier apparaissant dans le résultat (i.e: le plus petit entier i tel que 2^{32i} soit supérieur à la valeur absolue de l'entier considéré).
- O.O.M. : "Out Of Memory"
- D. : dimension de la matrice traitée.

	Gauss	Red.		D.
Name	T.	T.	L.	
e1	0,3	0,15	10	56
e2	0,08	0,03	29	16
e3	83,41	23,58	442	46
e4	8,09	5,71	316	32
e5	O.O.M.	1003,61	584	128
e6	1,28	0,49	11	70
e7	313,11	127,41	123	144
e8	O.O.M.	405,82	332	128
e9	23,66	20,86	654	40
e10	16,63	2,03	590	16
e11	220,29	25,46	894	32
e12	164,17	65,24	2125	24
e13	824	1277	2348	40
e14	2227	554,11	590	40
e15	22,12	3,6	132	40

La figure 4.1 montre que notre algorithme est plus efficace pour tous les exemples (sauf pour l'exemple e13). On peut d'autre part remarquer une variation sensible du rapport de temps entre le nouvel algorithme et l'algorithme de Gauss d'un exemple à l'autre. Ceci est dû à de bonnes réductions des fractions apparaissant dans l'algorithme de Gauss pour certains cas.

4.2.2 Résultats de complexité

Compte tenu de la modification apportée à la méthode d'orthonormalisation de Gauss pour obtenir notre algorithme, le nombre d'opérations arithmétiques de base (dans Z) effectuées dans *Réduction* est en $O(D^3)$ si D est la dimension de la matrice considérée.

Proposition 4.2.5 *Soit F une forme quadratique à coefficients entiers de matrice M. Si $\mathcal{T}$ est la taille des coefficients (représentation binaire) de M et D la dimension de M, la taille des entiers intervenant dans l'algorithme Réduction est en $O(D(\mathcal{T} + log(D)))$.*

Preuve : D'après la proposition 4.2.3, $l_{k+1} > l_k + 1$, $\forall k = 0 \ldots s$. Donc, la taille des coefficients des N_k est de l'ordre de $\mathcal{T} + 2$ pour $m = 0 \ldots D'$. Ainsi, comme les calculs effectués dans l'algorithme sont des déterminants extraits des matrices N_k. En utilisant la borne d'Hadamard sur la taille des déterminants, la taille maximum des coefficients apparaissant dans *Réduction* est donc $O(D(\mathcal{T} + log(D)))$. □

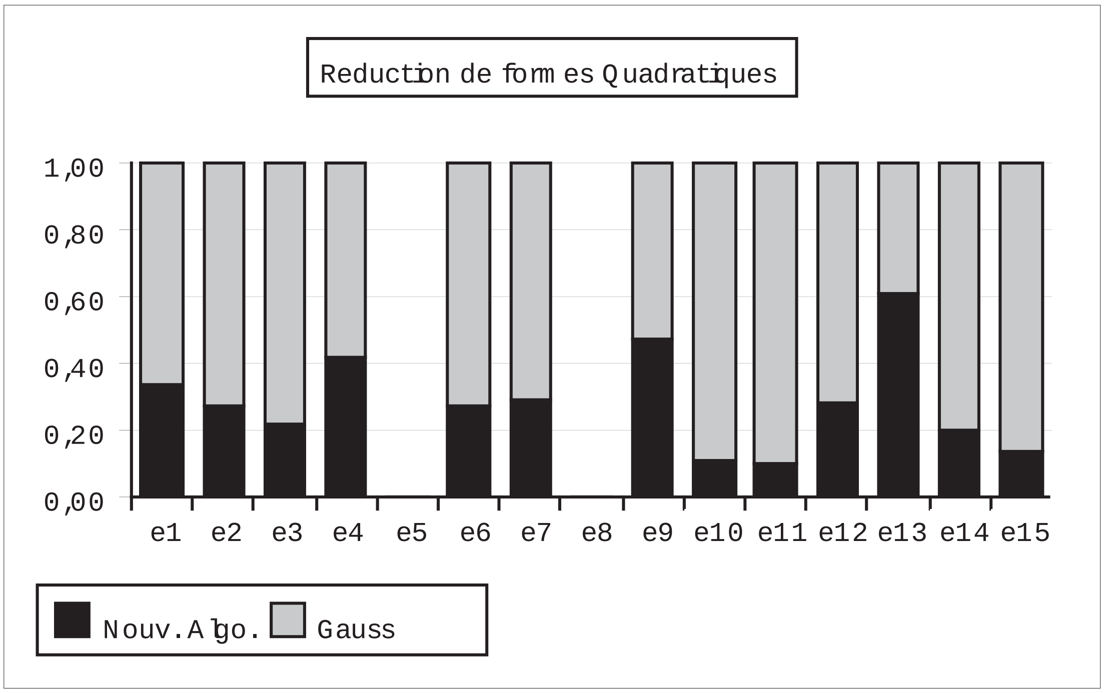

Figure 4.1: Performances comparées des algorithmes de réduction de formes quadratiques

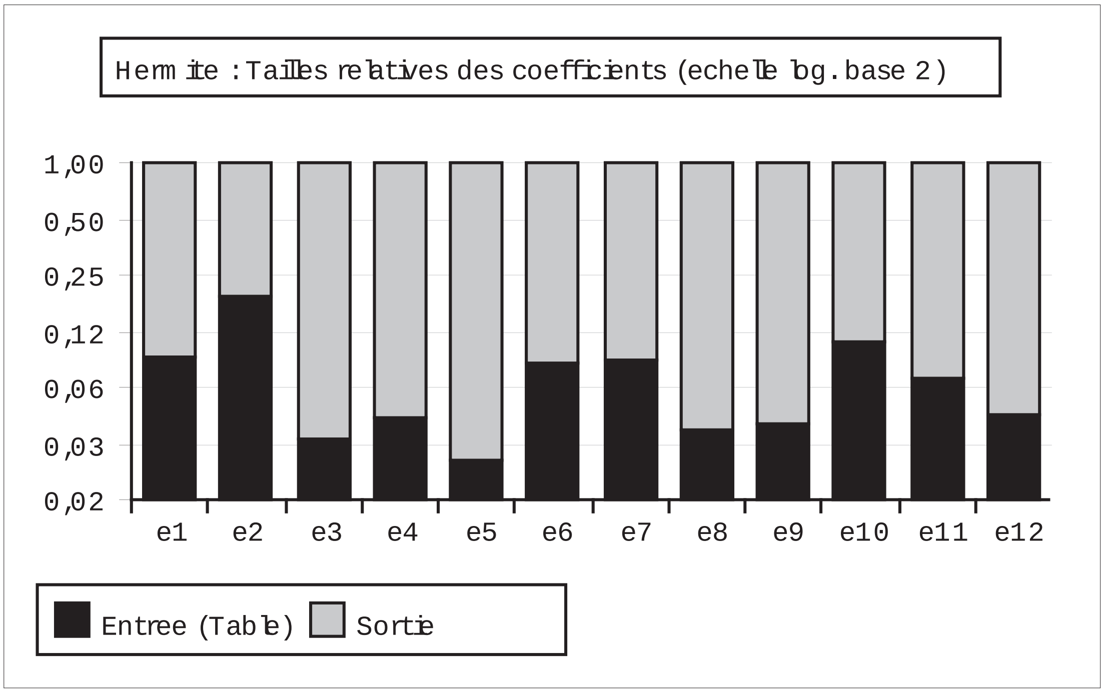

Figure 4.2: Formes quadratiques : Taille des coefficients

Le bon contrôle de la taille des coefficients lors de la réduction de la forme quadratique de Hermite généralisée est confirmé par la figure 4.2.

Nous revenons maintenant sur un résultat énoncé dans le chapitre *Résolution des systèmes zéro-dimensionnels* (proposition 3.5.5):

Proposition : *Soient $\mathcal{I} \subset \mathbb{Q}[X_1, \ldots, X_n]$ un idéal zéro-dimensionnel, et $\mathcal{G} \subset \mathbb{Z}[X_1, \ldots, X_n]$ une base de Groebner de $\mathcal{I}$. Si $\mathcal{T}$ désigne la taille maximale d'un entier intervenant dans l'expression des matrices M_{X_i} (multiplication par X_i dans $\mathbb{Q}[X_1, \ldots, X_n]/\mathcal{I}$), le nombre d'entiers premiers de taille $O(1)$ de mauvaise réduction pour le calcul d'une représentation univariée rationnelle de $\mathcal{I}$ est $O(D^2\mathcal{T})$.*

Preuve : Soit Q la forme quadratique de Hermite généralisée associée au polynôme unitaire, M la matrice (de dimension D) de Q dans une base de monômes $\mathcal{B}$ de $\mathbb{Z}[X_1, \ldots, X_n]/\mathcal{I}$. Notons $Q^p = \phi_p(Q)$ son image modulo un entier premier plus grand que D $\mathcal{G}$-compatible et $M^p = \phi_p(M)$ la matrice de $\phi_p(Q)$ dans la base $\mathcal{B}$ de $GF(p)[X_1, \ldots, X_n]/\langle \phi_p(\mathcal{G}) \rangle$ (voir proposition 3.5.1).

En reprenant les notations du théorème 4.2.1, on peut réécrire Q et Q^p comme suit :

$$\Gamma(Q) = \sum_{i=1}^{d} D_{i,i}^{i-1}(N_i) D_{i-1,i-1}^{i-2}(N_{i-1}) Y_i^2$$

$$\Gamma(Q^p) = \sum_{i=1}^{d_p} D_{i,i}^{i-1}(N_i^p) D_{i-1,i-1}^{i-2}(N_{i-1}^p) Y_i^2$$

Avec les notations de la définition 4.2.3 et en supposant les conditions de la convention 4.2.1 vérifiées pour Q on définit $l(M) = Inf\{l\ ,\ D_{l,l}^{l-1}(M) = 0\}$ (event. $l(M) = D+1$) et $l(M^p) = Inf\{l\ ,\ D_{l,l}^{l-1}(M^p) = 0\}$ (event. $l(M^p) = D+1$). Ainsi, si p ne divise aucun des mineurs $D_{k,k}^{k-1}(M)$, $k = 1, \ldots, l(M)-1$, $l(M) = l(M^p)$ et $\phi_p(D_{l,l}^{l-1}(M)) = D_{l,l}^{l-1}(M^p)$. En itérant cet argument, si p ne divise aucun des $D_{i,i}^{i-1}(N_i)$, $i = 1, \ldots, d$ alors $d = d_p$, $\phi_p(D_{i,i}^{i-1}(N_i)) = D_{l,l}^{l-1}(N_i^p)$, $i = 1, \ldots, d$ et donc $\phi_p(\Gamma(Q)) = \Gamma(Q^p)$. En particulier, ceci implique que Q et Q^p ont même rang. D'après la proposition 2.2.2 les coefficients de Q sont de taille $O(D\mathcal{T})$ ce qui montre que les mineurs $D_{i,i}^{i-1}(N_i)$, $i = 1, \ldots, d$ sont de taille $O(D^2\mathcal{T})$ (proportionnelle au nombre d'entiers premiers de mauvaise réduction pour le calcul d'une représentation univariée rationnelle de $\mathcal{I}$). □

4.3 Les inégalités simultanées

Dans toute cette partie, K désigne un corps ordonné de clôture réelle R

La méthode décrite, dans cette section, est fortement inspirée par celles présentes dans [RS90] et [Roy1]. Nous avons toutefois rajouté un certain nombre de raffinements algorithmiques.

Soient $\mathcal{I} \in K[X_1, \ldots, X_n]$ un idéal zéro-dimensionnel, $\mathcal{X} = V_{\mathbf{R}}(\mathcal{I})$ l'ensemble des zéros réels de $\mathcal{I}$ et $Q_1, \ldots, Q_m$ une liste de de polynômes de $K[X_1, \ldots, X_n]$.

Definition 4.3.1 *Pour une suite de signes $\epsilon = \{\epsilon_1, \ldots, \epsilon_m\}$ $(\epsilon_i \in \{+, 0, -\})$ donnée, la réalisation de la condition de signes ϵ par les polynômes $Q_1, \ldots, Q_m$ en les solutions réelles de $\mathcal{I}$ est un sous-ensemble de $\mathcal{X}$:*

$$C_\epsilon(\mathcal{I}; Q_1, \ldots, Q_n) = \{\alpha \in \mathcal{X} \mid (\forall i)\ Sign(Q_i(\alpha)) = \epsilon_i\}.$$

Notre but est de calculer, pour une famille de polynômes $Q_1, \ldots, Q_m$ et un idéal zéro-dimensionnel $\mathcal{I}$ fixés, le cardinal de toutes les réalisations de conditions de signes possibles par les les polynômes $Q_1, \ldots, Q_m$ en les zéros réels de $\mathcal{I}$, c'est à dire, pour tout $\epsilon \in \{+, 0, -\}^m$ les quantités :

$$c_\epsilon(\mathcal{I}; Q_1, \ldots, Q_n) = \mathrm{card}(C_\epsilon(\mathcal{I}; Q_1, \ldots, Q_n)).$$

Notre algorithme nécessite l'existence d'une *boite noire* permettant de calculer les quantités :

$$V_{\mathbf{R}}(\mathcal{I}; Q) = c_{\{+\}}(\mathcal{I}; Q) - c_{\{-\}}(\mathcal{I}; Q).$$

Ainsi, selon le nombre de variables n, il sera possible, par exemple, d'utiliser indifféremment l'algorithme de Sturm-Habicht (cas $n = 1$) où la méthode de Hermite (cas $n > 1$).

4.3.1 Les principes de base

Definition 4.3.2 *Une* pseudo-partition *de hauteur l de $\mathcal{X}$ est une liste $C = [C_1, \ldots, C_l]$ de l sous-ensembles de $\mathcal{X}$ tels que :*

- $\mathcal{X} = \bigcup_{i=1}^{l} C_i$
- *pour tout couple (i, j) tel que $i \neq j$, $C_i \bigcap C_j = \emptyset$*

Une liste de polynômes $H = [H_1, \ldots, H_m]$ est adaptée *à la pseudo-partition $C = [C_1, \ldots, C_l]$ de $\mathcal{X}$ si, sur chaque C_i , $i = 1 \ldots l$, les signes des polynômes de H sont constants.*

Pour toute liste de polynômes H adaptée à C, on définit la matrice des signes de H sur C, *de dimension $n \times m$, $A(H, C)$ par : $A(H, C)(i, j) =$ signe de H_j sur C_i.*

Par exemple $H = [1, Q, Q^2]$ est adaptée à $C = [R_{\{0\}}(\mathcal{I}; Q), R_{\{+\}}(\mathcal{I}; Q), R_{\{-\}}(\mathcal{I}; Q)]$ et :

$$A(H, C) = A^{(0)} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \\ 0 & 1 & 1 \end{pmatrix}$$

Definition 4.3.3 *Soient $C = [C_1, \ldots, C_l]$ une pseudo partition de $\mathcal{X}$ et $H = [H_1, \ldots, H_m]$ une liste de polynômes adaptée à C. On définit :*

- $c(C) = [\sharp C_1, \ldots, \sharp C_l]^T$
- $\mathrm{sign}(H(x)) = [sign(H_1(x)), \ldots, sign(H_1(x))]$
- $V_{\mathbf{R}}(\mathcal{I}; H) = [V_{\mathbf{R}}(\mathcal{I}; H_1), \ldots, V_{\mathbf{R}}(\mathcal{I}; H_m)]^T$

Proposition 4.3.1 *Soient une pseudo-partition C de $\mathcal{X}$ et H une liste de polynômes adaptée à C.*

$$A(C, H) \cdot c(C) = V_{\mathbf{R}}(\mathcal{I}; H)$$

Preuve : La j^e ligne de $A(C, H)$ est exactement le vecteur :

$$[sign(H_j(x \in C_1), \ldots, sign(H_j(x \in C_l)]$$

(rappelons que le signe de H_j sur C_i est constant). par conséquent, nous avons immédiatement la relation :

$$[sign(H_j(x \in C_1), \ldots, sign(H_j(x \in C_l)] \cdot c(C)^T = V_{\mathbf{R}}(\mathcal{I}; H_j)$$

et donc :

$$A(C, H) \cdot c(C) = V_{\mathbf{R}}(\mathcal{I}; H).$$

□

Dans le cas particulier où $C = [R_{\{0\}}(\mathcal{I}; Q), R_{\{+\}}(\mathcal{I}; Q), R_{\{-\}}(\mathcal{I}; Q)]$ et $H = [1, Q, Q^2]$, nous obtenons :

Corollaire 4.3.1 *Étant donné $Q \in K[X_1, \ldots, X_n]$, en posant :*

$$C = [R_{\{0\}}(\mathcal{I}; Q), R_{\{+\}}(\mathcal{I}; Q), R_{\{-\}}(\mathcal{I}; Q)]$$

et $H = [1, Q, Q^2]$, la relation $A(C, H) \cdot c(C) = V_{\mathbf{R}}(\mathcal{I}; H)$ s'écrit :

$$A^{(0)} \cdot \begin{pmatrix} c_{\{0\}}(\mathcal{I}; Q) \\ c_{\{+\}}(\mathcal{I}; Q) \\ c_{\{-\}}(\mathcal{I}; Q) \end{pmatrix} = \begin{pmatrix} V_{\mathbf{R}}(\mathcal{I}; 1) \\ V_{\mathbf{R}}(\mathcal{I}; Q) \\ V_{\mathbf{R}}(\mathcal{I}; Q^2) \end{pmatrix}$$

Definition 4.3.4

- *Si $C = [C_1, \cdots, C_l]$ et $C' = [C'_1, \cdots, C'_{l'}]$ sont deux pseudo-partitions de $\mathcal{X}$, on note $C \cap C'$ la pseudo-partition de $\mathcal{X}$ définie par :*

$$[C_1 \cap C'_1, \cdots, C_n \cap C'_1, \cdots, C_1 \cap C'_{n'}, \cdots, C_n \cap C'_{n'}]$$

- *Si* $H = [H_1, \cdots, H_m]$ *et* $H' = [H'_1, \cdots, H'_{m'}]$ *sont deux listes de polynômes, on note* $[H \cdot H']$ *la liste de polynômes (de cardinal de* mm'*) :*

$$[H_1 \cdot H'_1, \cdots, H_n \cdot H'_1, \cdots, H_1 \cdot H'_{n'} \cdots, H_n \cdot H'_{n'}].$$

- *Si* A *et* A' *sont deux matrices de dimensions respectives* $l \times m$ *et* $l' \times m'$*, on note* $A \otimes A'$ *le produit tensoriel des matrices* A *et* A'*, c'est à dire la matrice de dimension* $ll' \times mm'$ *obtenue en remplaçant le terme* $a'_{i,j}$ *de* A' *par la matrice* $a'_{i,j}A$*.*

Lemme 4.3.1 *Si* H *et* H' *sont deux listes de polynômes adaptées respectivement aux deux pseudo-partitions* C *et* C'*, alors* $[H \cdot H']$ *est adaptée à* $C \cap C'$*.*

Preuve: Soient $H_i \in H$ et $H'_j \in H'$. Pour tout $C_l \in C$ et tout $C'_m \in C'$, H_i (resp. H'_j) est de signe constant sur C_l (resp. C'_m), par conséquent, $H_i \cdot H'_j$ est de signe constant sur $C_l \cap C'_m$. □

D'après le lemme précédent, on peut définir une matrice $A(C \cap C', H \cdot H')$ telle que

$$A(C \cap C', H \cdot H') \cdot c(C \cap C') = V_{\mathbf{R}}(\mathcal{I}, H \cdot H').$$

Proposition 4.3.2 *Soient deux pseudo-partitions* C *et* C' *de* $\mathcal{X}$ *et deux listes de polynômes* H *et* H' *telles que* H *(resp.* H'*) soit adaptée à* C *(resp.* C'*). Alors :*

$$A(C, H) \otimes A(C', H') = A(C \cap C', H \cdot H')$$

Preuve : Ce résultat est immédiat à partir des définitions :

$$\begin{aligned} A(C, H) \otimes A(C', H')(j \cdot m' + l, i \cdot l' + m) &= sign(H_j(x \in C_i)) \cdot sign(H'_l(x \in C'_m)) \\ &= sign(H_j \cdot H'_l(y \in C_i \cap C'_m)) \\ &= A(C \cap C', H \cdot H')(j \cdot m' + l, i \cdot l' + m) \end{aligned}$$

□

En combinant les deux résultats précédents, nous obtenons le résultat principal de cette partie :

Corollaire 4.3.2 *Soient deux pseudo-partitions* C *et* C' *de* $\mathcal{X}$ *et deux listes de polynômes* H *et* H' *telles que* H *(resp.* H'*) soit adaptée à* C *(resp.* C'*). Alors :*

$$A(C, H) \otimes A(C', H') \cdot c(C \cap C') = V_{\mathbf{R}}(\mathcal{I}; H \cdot H')$$

4.3.2 L'algorithme naif

Supposons que nous disposons d'une *boite noire* suceptible de calculer $V_{\mathbf{R}}(\mathcal{I}, Q)$ pour tout polynôme $Q \in K[X_1, \dots X_n]$ (Sturm-Habicht ou Hemite selon le nombre de variables).
En combinant les corollaires (4.3.1) et (4.3.2) , nous obtenons le résultat suivant:

Corollaire 4.3.3 *Soient une pseudo-partitions C de $\mathcal{X}$, une liste H de polynômes de $K[X_1, \dots, X_n]$ adaptée à C et Q un polynôme de $K[X_1, \dots, X_n]$. Alors :*

$$A(C, H) \otimes A(C_Q, Q) \cdot c(C \cap C_Q) = V_{\mathbf{R}}(\mathcal{I}; H \cdot [1, Q, Q^2])$$

où $C_Q = [C_{\{0\}}(\mathcal{I}; Q)\ ,\ C_{\{+\}}(\mathcal{I}; Q)\ ,\ C_{\{-\}}(\mathcal{I}; Q)]$.

Ce corollaire nous donne alors un algorithme itératif pour résoudre notre problème :

Algorithme **SI-Seq-Naif**

- **Entrée :** $\mathcal{I}$, $\{Q_1, \dots, Q_s\}$, $Q \to V_{\mathbf{R}}(\mathcal{I}, Q)$
- **Sortie :** Le cardinal de chaque condition de signes réalisée par $\{Q_1, \dots, Q_s\}$ sur $\mathcal{X}$.
- **Initialisations :**
  - $H^{(0)} = [1, Q_1, Q_1^2]$
  - $V^{(0)} = [V_{\mathbf{R}}(\mathcal{I}, 1), V_{\mathbf{R}}(\mathcal{I}, Q_1), V_{\mathbf{R}}(\mathcal{I}, Q_1^2)]$
  - $S^{(0)} = [\{0\}, \{-\}, \{+\}]$
- **Etape 1 :**
  Pour $i = 1, \dots s$:
  - $A^{(i)} = A^{(0)} \otimes A^{(i-1)}$
  - $H^{(i)} = [H^{(i-1)} \cdot [1, Q_i, Q_i^2]]$
  - $V^{(i)} = [V_{\mathbf{R}}(\mathcal{I}, H_k^{(i-1)}), k = 1, \dots, \sharp H^{(i-1)}]$
  - $S^{(i)} = [S_k^{(i-1)} \cup \{0\}, S_k^{(i-1)} \cup \{-\}, S_k^{(i-1)} \cup \{+\}\quad , \quad k = 1, \dots, log_3(\sharp S^{(i-1)})]$
- **Etape 2 :** Résoudre $A^{(s)} \cdot c = V^{(s)}$

Résultat : deux vecteurs c et $S^{(s)}$ tels que pour tout $i = 1 \dots 3^s$, le nombre de points de $\mathcal{X}$ réalisant la condition de signe $S_i^{(s)}$ soit c_i.

Preuve de l'algorithme : Si l'on note $C^{(i)}$ la pseudo-partition associée cannoniquement à l'ensemble des conditions de signes $S^{(i)}$, la définition (4.3.4) et le lemme (4.3.1) nous assurent que $H^{(i)}$ est adaptée à $C^{(i)}$ et que $A^{(i)} = A(C^{(i)}, H^{(i)})$ pour tout $i = 1, \dots, s$.

On peut montrer facilement, par récurrence, que les matrices $A^{(i)}$ sont inversibles grâce aux relations :

- $det(A^{(0)}) = -2$
- Pour $i > 0$, $det(A^{(i)}) = det(A^{(0)} \otimes A^{(i-1)}) = det(A^{(0)}) \cdot det(A^{(i-1)})$

Il ne reste plus qu'à appliquer le corollaire (4.3.2) pour montrer que l'unique solution c du système $A^{(s)} \cdot c = V^{(s)}$ est $c = c(C^{(s)})$ et donc conclure. $\square$

En observant cette méthode, on s'apperçoit immédiatement que le système linéaire de l'étape 2 est de dimension 3^s et par conséquent que le nombre de conditions de signes ainsi déterminées est 3^s. C'est en ce sens que l'algorithme est naif, car le nombre de conditions de signes non vides définies par les polynômes Q_i sur $\mathcal{X}$ ne peut en aucun cas éxéder $\sharp\mathcal{X}$.

4.3.3 L'algorithme

Nous avons vu dans la partie précédente un algorithme de base pour déterminer le cardinal de **TOUTES** les réalisations de signes d'une famille de polynômes en les zéros réels d'un système zéro-dimensionel. Pour éviter un nombre d'appel exponentiel à la *boite noire* (V) nous allons nous attacher à *épurer* les pseudo-partitions utilisées, en partant du principe que seules les réalisations de signes non vides sont à dénombrer, les autres se déduisant par complémentarité.

Definition 4.3.5 *Soit C une pseudo-partition dont le nombre de sous-ensembles non vides est noté $l(C)$. On définit alors la liste $co(C) = [co(1), \ldots, co(l(C))]$ des indices des ensembles C_i de C de cardinal non nul et on note*

$$Red(C) = [C_{co(1)}, \ldots, C_{co(l(C))}]$$

la partition ainsi obtenue.

Le but de cette partie est de modifier l'algorithme **SI-Seq-Naif** pour pouvoir travailler avec des partitions au lieu de pseudo-partitions. La principale difficulté est de contourner le problème énoncé dans la remarque suivante :

Remarque 4.3.1 *Soient C et C' deux partitions, $C \cap C'$ (voir définition 4.3.4) n'est pas forcément une partition.*

La simplification

Soient C une pseudo-partition et H une liste de polynômes, adaptée à C, telles que $A(C, H)$ soit carrée et inversible.
H est clairement adaptée à $Red(C)$ et nous avons de plus :

$$A(Red(C), H) \cdot c(Red(C)) = V_{\mathbf{R}}(\mathcal{I}, H)$$

où $A(Red(C), H)$ se définit simplement par :

$$A(Red(C), H) = (A(C, H)[i, j])_{i=1,\ldots,\sharp H}^{j\in co(C)}$$

Comme $A(C, H)$ est supposée inversible, il existe une liste d'indices $li(H)$ de cardinal $\sharp Red(C)$ telle que la matrice :

$$Red(A(Red(C), H), li(H)) = (A(C, H)[i, j])_{i\in li(H)}^{j\in co(C)}$$

soit inversible. Si l'on note maintenant $Red(H, li(H)) = [H_i\ ,\ i \in li(H)]$, la liste $Red(H, li(H))$ est clairement adaptée à $Red(C)$ et nous avons par construction :

- $Red(A(Red(C), H), li(H)) = A(Red(C), Red(H, Li(H)))$
- $A(Red(C), Red(H, Li(H)) \cdot c(Red(C)) = V_{\mathbf{R}}(\mathcal{I}, Red(H, Li(H)))$

L'algorithme

En notant $MaxInvert(A(C, H), co(C))$ la fonction qui calcule la liste d'indices $li(H)$ associée à C et $A(C, H)$ comme décrit plus haut (lorsque $A(C, H)$ est carrée et inversible), nous proposons l'algorithme suivant (Inégalités simultanées, version séquentielle) :

Algorithme **SI-Seq**

- **Entrée :** $\mathcal{I}$, $\{Q_1, \ldots, Q_s\}$, $Q \to V_{\mathbf{R}}(\mathcal{I}, Q)$
- **Sortie :** Le cardinal de chaque condition de signes réalisée par $\{Q_1, \ldots, Q_s\}$ sur $\mathcal{X}$.
- **Fonction Annexe :** $Simplifie(c, A, H, S)$

  (Remarque : Si on note C la pseudo-partition cannoniquement associée à la liste de conditions de signes S, on peut alors , par abus de notation, définir $Red(S)$).

  $Simplifie(c, A, H, S)$ effectue les substitutions suivantes :

  - $co(c) := [i\ ,\ n_i \in c\ ,\ n_i \neq 0]$ (exactement equivallent à $co(C)$)
  - $li(H) := MaxInvert(A, co(c))$
  - $A := (A[i, j])_{i\in li(H)}^{j\in co(c)}$
  - $S := Red(S)$
  - $c := [c_i \in c\ ,\ i \in co(c)]$
  - $H := Red(H, li(H))$
- **Initialisations :**
  - $H^{(0)} = [1, Q_1, Q_1^2]$

- $V^{(0)} = [V_{\mathbf{R}}(\mathcal{I}, 1), V_{\mathbf{R}}(\mathcal{I}, Q_1), V_{\mathbf{R}}(\mathcal{I}, Q_1^2)]$
- $S^{(0)} = [\{0\}, \{-\}, \{+\}]$

• **Boucle Principale :**

Pour $i = 1, \ldots s$:

- $c^{(i-1)} = (A^{(i-1)})^{-1} \cdot V_{\mathbf{R}}(\mathcal{I}, H^{(i-1)})$
- $Simplifie(c^{(i-1)}, A^{(i-1)}, H^{(i-1)}, S^{(i-1)})$
- $H^{(i)} = [H^{(i-1)} \cdot [1, Q_i, Q_i^2]]$
- $S^{(i)} = [S_k^{(i-1)} \cup \{0\}, S_k^{(i-1)} \cup \{-\}, S_k^{(i-1)} \cup \{+\}\ ,\ k = 1, \ldots, \sharp S^{(i-1)}]$
- $A^{(i)} = A^{(0)} \otimes A^{(i-1)}$

Résultat : deux listes $c^{(s)}$ et $S^{(s)}$ tels que pour tout $i = 1 \ldots \sharp c^{(s)}$, le nombre de points de $\mathcal{X}$ réalisant la condition de signe $S_i^{(s)}$ soit $c_i^{(s)}$.

Preuve de l'algorithme : Tous les éléments nécessaires à la preuve ont été développés plus haut où dans la preuve de la version naive de de l'algorithme : pour tout $i = 1, \ldots, s$, si $C^{(i)}$ est la pseudo partition associée canoniquement à $S^{(i)}$, $H^{(i)}$ est adapté à $C^{(i)}$, $c^{(i)} = c(C^{(i)})$ et $A^{(i)}$ est carrée et inversible. □

Optimisations

Nous disposons maintenant d'un bon algorithme de base, que nous allons , dans cette partie, faire évoluer. Deux points sont importants :

- On peut remarquer que $H^{(i)}$ est systématiquement calculé par la formule $H^{(i)} = [H^{(i-1)} \cdot [1, Q_i, Q_i^2]]$, même si certains polynômes de $[1, Q_i, Q_i^2]]$ n'apportent n'apportent aucune information supplémentaire (par exemple si Q_i est toujours positif, l'addition de Q_i^2 apparait superflue). Il convient donc de rajouter une autre étape de simplification.

- La liste $li(H)$ utilisée pour épurer les pseudo-partitions n'est pas définie de manière unique. Il nous faudra donc déterminer des stratégies pour que celle-ci entraine le moins possible d'appels à la boite noire V.

Nombre d'appels à V : Attachons nous dans un premier temps à étudier le premier point. Il faut remarquer que la liste de polynômes $H^{(i-1)}$ contient (du moins nous nous arrangerons pour cela) en général le polynôme unitaire. Ainsi, si on suppose que le processus est à l'étape i et que nous désirons rajouter un polynôme Q, les quantités $V_{\mathbf{R}}(\mathcal{I}, Q)$ et $V_{\mathbf{R}}(\mathcal{I}, Q^2)$ seront dans tous les cas calculées. Nous nous proposons de faire ce calcul à part et de montrer que selon les valeurs obtenues, nous pouvons optimiser efficacement le processus de calcul en évitant, dans certains cas, d'appliquer la boite noire V sur la totalité des polynômes de la liste : $[H^{(i-1)} \cdot [1, Q_i, Q_i^2]]$.

- En calculant $V_{\mathbf{R}}(\mathcal{I};Q)$, deux cas de figure peuvent se produire :

  – *Cas 1:* $V_{\mathbf{R}}(\mathcal{I};Q) = \sharp\mathcal{X}$.
  Le polynôme Q est positif en tous les zéros réels de $\mathcal{I}$. Nous pouvons alors stopper l'étape de construction, en rajoutant le signe $+$ à toutes les conditions de signes :

  $$S^{(i)} = [S_k^{(i-1)} \cup \{+\}\ ,\ k = 1 \dots \sharp S_k^{(i-1)}]$$

  On pose alors $H^{(i)} = H^{(i-1)}$, $A^{(i)} = A^{(i-1)}$ et $c^{(i)} = c^{(i-1)}$

  – *Cas 2:* $V_{\mathbf{R}}(\mathcal{I};Q) = -\sharp\mathcal{X}$.
  Le polynôme Q est négatif en tous les z'eros réels de $\mathcal{I}$. Nous pouvons alors stopper l'étape de construction, en rajoutant le signe $-$ à toutes les conditions de signes :

  $$S^{(i)} = [S_k^{(i-1)} \cup \{-\}\ ,\ k = 1 \dots \sharp S_k^{(i-1)}]$$

  On pose alors $H^{(i)} = H^{(i-1)}$, $A^{(i)} = A^{(i-1)}$ et $c^{(i)} = c^{(i-1)}$

- Si $V_{\mathbf{R}}(\mathcal{I};Q) \neq \sharp\mathcal{X}$ et $V_{\mathbf{R}}(\mathcal{I};Q) \neq -\sharp\mathcal{X}$, nous avons alors forcément besoin de l'information $V_{\mathbf{R}}(\mathcal{I};Q^2)$. Celà dit, plusieurs cas particuliers peuvent encore être distingués :

  – *Cas 3:* $V_{\mathbf{R}}(\mathcal{I};Q^2) = 0$. Le polynôme Q est nul en tous les zéros réels de $\mathcal{I}$. Nous pouvons alors stopper létape de construction, en rajoutant le signe 0 à toutes les conditions de signes :

  $$S^{(i)} = [S_k^{(i-1)} \cup \{0\}\ ,\ k = 1 \dots \sharp S_k^{(i-1)}]$$

  On pose alors $H^{(i)} = H^{(i-1)}$, $A^{(i)} = A^{(i-1)}$ et $c^{(i)} = c^{(i-1)}$

  – *Cas 4:* $V_{\mathbf{R}}(\mathcal{I};Q^2) = V_{\mathbf{R}}(\mathcal{I};Q)$. Le polynôme Q est nul ou positif en tous les zéros réels de $\mathcal{I}$.

  Nous pouvons alors traduire ceci par la relation matricielle :

  $$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} c_{\{0\}}(\mathcal{I};Q) \\ c_{\{+\}}(\mathcal{I};Q) \end{pmatrix} = \begin{pmatrix} V_{\mathbf{R}}(\mathcal{I};1) \\ V_{\mathbf{R}}(\mathcal{I};Q) \end{pmatrix}.$$

  Dans ce cas, $H^{(i)}, A^{(i)}, S^{(i)}$ seront donnés par :

$$A^{(i)} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \otimes A^{(i-1)},$$

$$S^{(i)} = [S_k^{(i-1)} \cup \{0\} \ , \ S_k^{(i-1)} \cup \{+\} \ , \ k = 1 \dots \sharp S_k^{(i-1)}], \cdot$$

$$H^{(i)} = [H^{(i-1)} \cdot [1, Q]]$$

– *Cas 5:* $V_{\mathbf{R}}(\mathcal{I}; Q^2) = -V_{\mathbf{R}}(\mathcal{I}, Q)$. Le polynôme Q est nul ou négatif en tous les zéros réels de $\mathcal{I}$.

  Nous pouvons alors traduire ceci par la relation matricielle :

$$\begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} c_{\{0\}}(\mathcal{I}; Q) \\ c_{\{-\}}(\mathcal{I}; Q) \end{pmatrix} = \begin{pmatrix} V_{\mathbf{R}}(\mathcal{I}; 1) \\ V_{\mathbf{R}}(\mathcal{I}; Q) \end{pmatrix}.$$

  Dans ce cas, $H^{(i)}, A^{(i)}, S^{(i)}$ seront donnés par :

$$A^{(i)} = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix} \otimes A^{(i-1)},$$

$$S^{(i)} = [S_k^{(i-1)} \cup \{0\} \ , \ S_k^{(i-1)} \cup \{-\} \ , \ k = 1 \dots \sharp S_k^{(i-1)}], \cdot$$

$$H^{(i)} = [H^{(i-1)} \cdot [1, Q]]$$

– *Cas 6:* $V_{\mathbf{R}}(\mathcal{I}; Q^2) = \sharp \mathcal{X}$. Le polynôme Q n'est jamais nul en les zéros réels de $\mathcal{I}$.

  Nous pouvons alors traduire ceci par la relation matricielle :

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} c_{\{+\}}(\mathcal{I}; Q) \\ c_{\{-\}}(\mathcal{I}; Q) \end{pmatrix} = \begin{pmatrix} V_{\mathbf{R}}(\mathcal{I}; 1) \\ V_{\mathbf{R}}(\mathcal{I}; Q) \end{pmatrix}.$$

  Dans ce cas, $H^{(i)}, A^{(i)}, S^{(i)}$ seront donnés par :

$$A^{(i)} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes A^{(i-1)},$$

$$S^{(i)} = [S_k^{(i-1)} \cup \{+\} \ , \ S_k^{(i-1)} \cup \{-\} \ , \ k = 1 \dots \sharp S_k^{(i-1)}], \cdot$$

$$H^{(i)} = [H^{(i-1)} \cdot [1, Q]]$$

Algèbre Linéaire : Le deuxième point à développer est la partie *algèbre linéaire* utilisée dans l'algorithme. Rappelons qu'à chaque étape, étant donnés une matrice $A^{(i-1)}$ et un vecteur $c^{(i-1)}$, nous devons résoudre un système linéaire (**SISolve**), puis opérer une étape de simplification (**Simplifie**).

Bien qu'optimisées (utilisation de méthodes de type Bareiss), ces deux fonctions nous entrainent dans certains cas à utiliser une arithmétique multi-précision sur les entiers. En effet, la taille d'une matrice $A^{(i)}$ pouvant atteindre $3D$ (si D est le cardinal de $V_{\mathbf{R}}(\mathcal{I})$), son déterminant est alors en $O(3D + log(D))$. Ainsi, en utilisant une machine dont la longueur d'un mot mémoire est de 32 bits, une arithmétique multi-précision pourra être nécessaire dès lors que $\sharp V_{\mathbf{R}}(\mathcal{I}) > 10$.

Dans le cadre d'une utilisation avec l'algorithme de Sturm-Habicht ou Hermite, ceci n'est pas un facteur limitant. Cela dit, nous pouvons imaginer que la machinerie puisse être utilisée avec des algorithmes numériques extrèmement rapides, et dans ce cas, les deux fonctions décrites plus haut pourraient devenir les principales consommatrices de temps et de mémoire.
Une idée consiste à utiliser une arithmétique modulaire. Regardons pour commencer les tailles des entiers utilisés tout au long des diverses étapes:

- Les coordonnées du vecteur solution d'un système du type
$$A^{(i)}X = V_{\mathbf{R}}(\mathcal{I}, H^{(i)})$$
sont les nombres des racines réalisant un certain nombre de conditions de signes. Ce sont dons des entiers positifs dont la somme est majorée par le nombre total de racines.

- De la même façon, toute coordonnée v de $V_{\mathbf{R}}(\mathcal{I}, H^{(i)})$ vérifie l'inégalité :
$$-\sharp V_{\mathbf{R}}(\mathcal{I}) \leq v \leq \sharp V_{\mathbf{R}}(\mathcal{I})$$

- Les coefficients intervenant dans les matrices $A^{(i)}$ sont pris dans l'ensemble $\{0, 1, -1\}$.

Supposons que p soit un entier premier plus grand que $2 \cdot \sharp V_{\mathbf{R}}(\mathcal{I})$ et etudions le système $\overline{A^{(i)}}X_p = \overline{V_{\mathbf{R}}(\mathcal{I}, H^{(i)})})$ image de $A^{(i)}X = V_{\mathbf{R}}(\mathcal{I}, H^{(i)})$ dans $\mathbb{Z}/p\mathbb{Z}$. Si $\overline{A^{(i)}}$ est inversible modulo p, il existe un unique X_p tel que $\overline{A^{(i)}}X_p = \overline{V_{\mathbf{R}}(\mathcal{I}, H^{(i)})}$. Mais comme $p > 2 \cdot \sharp V_{\mathbf{R}}(\mathcal{I})$, X_p est également solution du système de départ.

Pour appliquer cette stratégie, il suffit par conséquent de s'assurer que $\overline{A^{(i)}}$ est inversible modulo un entier premier p pour tout i. En supposant $p > 2 \cdot \sharp V_{\mathbf{R}}(\mathcal{I})$ fixé, nous pouvons montrer ceci simplement par récurrence : la matrice $A^{(0)}$ est prise parmi les matrices :

$$M \in \{(1)\ , \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} , \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix} , \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} , \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \\ 0 & 1 & 1 \end{pmatrix}\}$$

qui sont toutes inversibles modulo un entier premier $p > 2 \cdot \sharp V_{\mathbf{R}}(\mathcal{I})$, car de déterminant égal à 2.

En supposant que $A^{(i-1)}$ soit inversible modulo p, la matrice $A^{(i)}$ étant construite (avant simplification) à l'aide d'une identité du type : $B \otimes A^{(i-1)}$ ou B est est choisie dans l'ensemble M, elle est inversible modulo p car

$$det(B \otimes A^{(i-1)}) = det(B)^{dim(B)} det(A^{(i-1)})^{dim(A^{(i-1)})}$$

Bien évidement, pour pouvoir appliquer cette stratégie modulaire à chaque étape, il faut également s'assurer que $A^{(i)}$ est encore inversible après l'étape de simplification, en remplaçant la fonction **MaxInvert** par un équivalent modulaire que nous noterons **ZpMaxInvert**. Notons d'ailleurs que les polynômes utilisés dans cet algorithme ne seront pas forcément les mêmes que ceux utilisés dans **SI-Seq**, toute sous matrice de $A^{(i)}$ n'étant pas inversible modulo p).
Ainsi, les fonctions **MaxInvert** et **SISolve** peuvent se transformer aisément en fonction modulaires **ZpMaxInvert** et **ZpSISolve** garanties à condition de considérer un entier p suffisement grand ($> 2\sharp V_{\mathbf{R}}(\mathcal{I})$).

4.3.4 Une application : Le codage à la Thom des racines d'un polynôme

L'algorithme *inégalités simultanées* a beaucoup d'applications pratiques liées à la détermination du nombre de racines dans une région de l'espace. Cette stratégie peu également être considéré comme outil de base pour le codage des nombres algébriques réels. Rappelons à ce sujet une conséquence du lemme de Thom (voir par exemple [Roy1]):

Lemme 4.3.2 *Soit P un polynôme à coefficients réels de degré D, dont les dérivées successives sont notées $P', P^{(2)}, \ldots, P^{(D-1)}$. Si α et β sont deux racines de P telles que pour tout $i = 1 \ldots D-1$ $sign(P^{(i)}(\alpha) = sign(P^{(i)}(\beta)$, alors $\alpha = \beta$.*

Ainsi, la connaissance des signes des dérivées de P en les racines de P permet de coder de manière unique ces nombres réels. Ce codage pourra être obtenu en appliquant l'algorithme *Inégalités simultanées* en considérant pour idéal, l'idéal engendré par P et comme famille de polynômes dont les conditions de signes doivent être étudiées, l'ensemble des dérivées de P.

Bien que la machinerie générale puisse être appliquée, nous avons effectué quelques optimisations visant à réduire le nombre et la complexité des calculs :

- L'algorithme s'arrête dès que toutes les racines ont été séparées, c'est à dire dès lors que chaque condition de signe est de cardinal égal à 1. Nous obtenons dans ce cas un codage tronqué puisque ne faisant pas intervenir la totalité des dérivées.

- Les dérivées sont ajoutées par ordre décroissant (i.e: $P^{(D-1)}, \ldots, P'$) pour traiter des polynômes de degrés les plus petits possibles et par conséquent réduire le coût de la fonction V.

4.4 Stratégies de calcul

Nous avons décrit jusqu'ici une certaine quantité d'algorithmes pour le calcul du nombre de solutions réelles d'un système zéro-dimensionnel. Nous nous attachons, dans cette partie, à en étudier le comportement en pratique.

D'après la partie précédente, nous pouvons distinguer 2 familles de méthodes pour le comptage des solutions réelles d'un système zéro-dimensionnel :

- Les méthodes de type *univarié*, réduisant le problème à l'étude d'un polynôme en une variable (représentation univariée rationnelle), puis appliquant un algorithme approprié (Sturm-Habicht, Uspensky)

- Les méthodes directes (Hermite) (de type *multivarié*)

L'ensemble des stratégies possibles est décrit par la figure 4.3.

Pour ne pas surcharger cette partie, nous ne revenons pas sur l'algorithme *Inégalités simultanées* puisque son comportement est directement conditionné par celui des méthodes de comptage et nous n'incluons pas dans notre étude la méthode de Hermite dans le cas de polynômes en une variable. Nous nous bornerons par conséquent à étudier trois algorithmes :

- Algorithme I : Table + Hermite

- Algorithme II : Table + RUR-Int + Sturm-Habicht

- Algorithme III : Table + RUR-Int + Uspensky

Nous ne reviendrons pas sur les temps de calcul absolus déjà détaillés dans les précédents chapitres, mais nous nous attarderons sur les performances relatives de ces trois méthodes.

L'algorithme III à été rajouté à cette série de tests comme alternative à l'algorithme II. L'un des principaux buts de cette étude étant de déterminer quelle famille de stratégies (*multivariées,univariées*) semble la plus efficace pour le comptage des zéros, il était nécessaire de considérer plusieurs méthodes univariées.

4.4.1 Un bilan sur les algorithmes

L'idée de cette partie est d'étudier séparément les performances des trois algorithmes choisis.

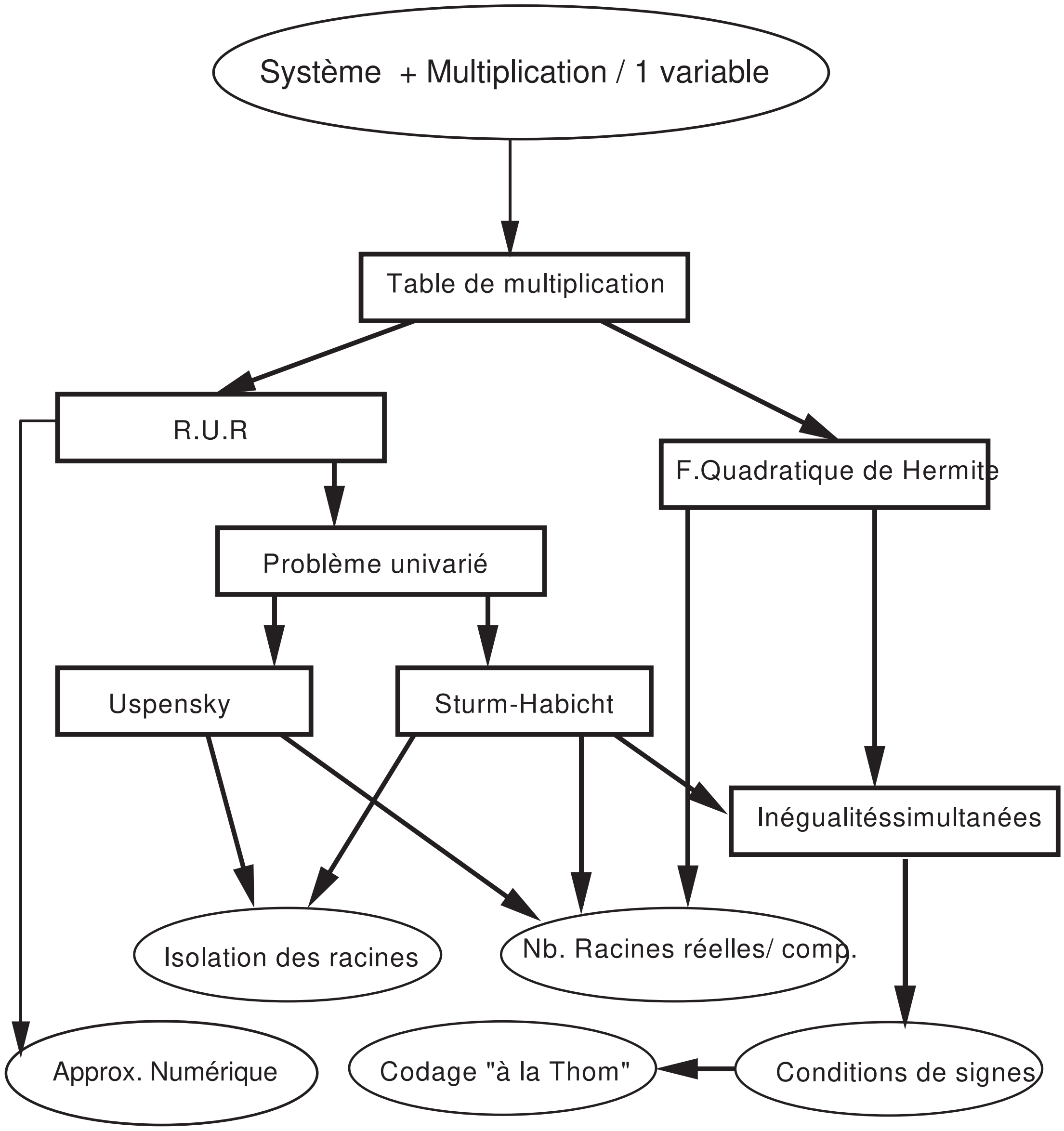

Figure 4.3: Que peut-on calculer ?

En outre, nous effectuons des rappels complets sur les complexités. Ceci ne donne qu'une indication sur le comportement en pratique des différentes méthodes, aussi nous nous limiterons au cas des systèmes à coefficients entiers pour permettre de mesurer facilement la taille des coefficients intervenant dans les calculs intermédiaires.

Comme indiqué dans le chapitre *représentation univariée rationnelle*, la complexité théorique est souvent en décalage avec la complexité que nous avons appelé *pratique*, surtout lorsque des méthodes modulaires sont appliquées. Par complexité pratique nous entendrons : complexité hors cas exceptionnels.

Pour que cette étude soit la plus précise possible, nous supposons que l'entrée des différents programmes est une partie de la table de multiplication : les matrices de multiplication par les $X_1, \ldots, X_n$ dans $\mathbb{Q}[X_1, \ldots, X_n]/\mathcal{I}$ et nous noterons $\mathcal{T}$ la taille binaire maximale observée pour les entiers apparaissant dans ces matrices.

Pour chaque méthode , nous précisons la taille des coefficients intervenant aussi bien en cours d'algorithme que dans le résultat, ainsi que le nombre d'opérations arithmétiques induit, pour faciliter l'analyse des résultats.

Pour mesurer les nombres d'opérations *machine* consommés par ces différents algorithmes, nous supposerons que le coût d'une multiplication de deux entiers (ou rationnels) de taille $\mathcal{T}$ est $M(\mathcal{T}) > \mathcal{T}$ (en pratique, nous pouvons supposer d'ailleurs que $M(\mathcal{T}) = \mathcal{T}^{log_2(3)}$, valeur calculée pour le package d'entiers utilisé pour ces mesures, celui-ci utilisant algorithme *Karatsuba* pour les multiplications).

Les pré-requis : Les trois méthodes décrites plus haut utilisent la table de multiplication comme entrée. Rappelons que celle-ci nécessite $O(D^4 + nD^3)$ opérations arithmétiques dans $\mathbb{Q}$ et que la taille des coefficients observée dans le résultat est $O(D\mathcal{T})$. Ceci nous donne un ordre de grandeur sur le nombre d'opération machine nécessaires : $O((D^4 + nD^3)M(D\mathcal{T})$.

Algorithme I : Hermite La construction de la forme quadratique Q_1 nécessite $O(D^3)$ opérations arithmétiques et entraîne le calcul de coefficients dont la taille est $O(D\mathcal{T} + log(D))$. En appliquant notre nouvel algorithme pour réduire Q_1, nous consommons $O(D^3)$ opérations arithmétiques sur des coefficients dont la taille est en $O(D(D\mathcal{T} + log(D)))$ soit $O(D^2\mathcal{T})$, en négligeant les termes logarithmiques. Ceci nous amène alors à un nombre d'opérations machine en $O(D^3 M(D^2\mathcal{T}))$.

La figure 4.4 montre que la part de temps consommée par la réduction de forme quadratique semble croître proportionnellement à un facteur de type $D^\alpha\mathcal{T}$.

Il n'en n'est pas de même en ce qui concerne la taille des coefficients. En effet, la figure 4.2 montre que la croissance des coefficients est contrôlée de façon homogène.

L'algorithme I semble par conséquent avoir un comportement sans surprise, le temps de calcul pouvant presque être estimé à la vue des tailles des coefficients apparaissant dans la table de multiplication et de la dimension de l'algèbre.

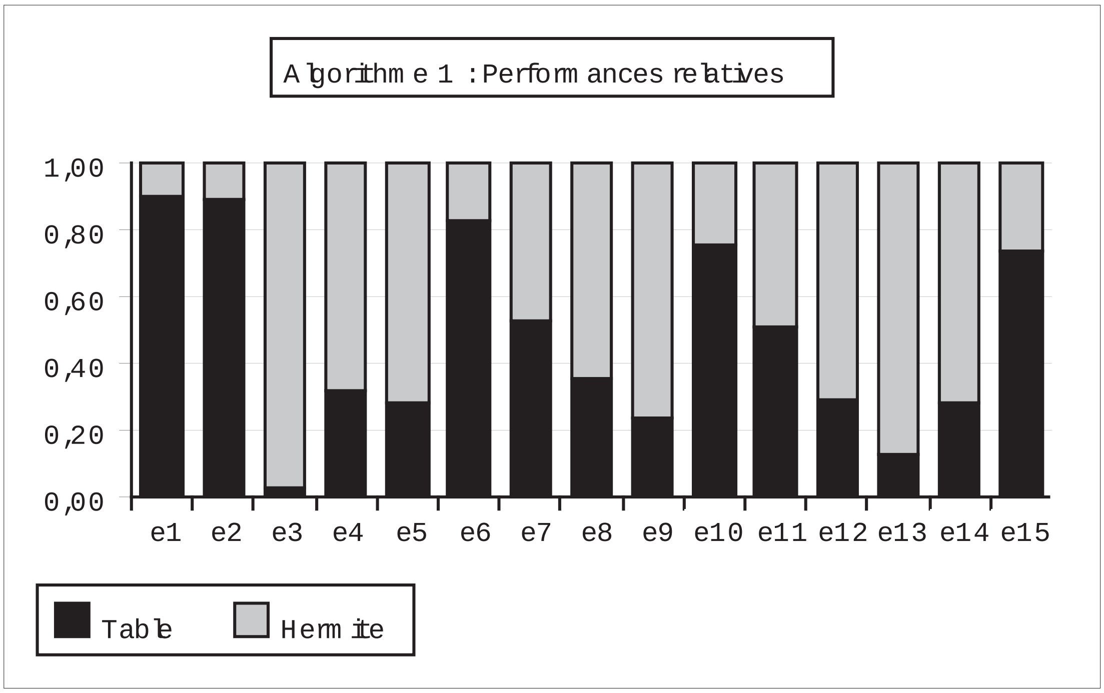

Figure 4.4: Algorithme I : Temps de calcul relatifs

Algorithmes II et III Dans sa version avec prédiction modulaire (utilisée pour les tests), l'algorithme RUR-int utilise $O(D^3M(D\mathcal{T}) + nD^2M(D^2\mathcal{T}))$ (on néglige les termes logarithmiques) opérations arithmétiques machine, en supposant connu un entier premier de bonne réduction. D'un point de vue pratique, nous pouvons considérer ceci comme un bon ordre de grandeur compte tenu du faible nombre d'entiers de mauvaise réduction apparaissant en général. En théorie toutefois, en tenant vraiment compte des mauvais entiers, nous obtenons un nombre d'opérations arithmétiques machines en $O(D^3(D^2\mathcal{T}^2 + DM(D\mathcal{T}) + nM(D^2\mathcal{T})))$.

Une discussion est également possible au sujet de l'ordre de grandeur des coefficients apparaissant dans l'expression finale. Théoriquement, les tailles des entiers observés peuvent être en $O(D^2\mathcal{T})$, mais nous avons souligné que rarement les coefficients d'un pgcd étaient supérieurs, en taille, à ceux des deux polynômes considérés ce qui nous permet d'évaluer une taille en pratique en $O(D\mathcal{T})$.

On peut remarquer que le rapport de temps entre l'algorithme calculant la table de multiplication et l'algorithme RUR-Int est relativement équitable (figure 3.1). En ce qui concerne la taille des coefficients apparaissant dans les résultats (figure 3.2), elle est en général plus petite à la sortie de l'algorithme RUR-Int qu'à l'entrée, lorsque l'une des variables sépare les zéros du système (Les exemples e1, e6, e8 ont des éléments séparants non triviaux), confirmant ainsi les hypothèses faites sur la complexité *pratique* de l'algorithme, d'autant qu'aucun entier premier de mauvaise réduction n'à été remarqué.

Dans tous les cas, l'application de l'algorithme de Sturm Habicht sur le premier polynôme de la représentation univariée rationnelle donne un nombre d'opérations arithmétiques (dans $\mathbb{Z}$) de l'ordre de $O(D^2)$ avec des tailles de coefficients atteignant $O(D\mathcal{T}_r)$ si $\mathcal{T}_r$ (resp. D) est la taille des coefficients (resp. le degré) du polynôme initial.

Sur la figure 4.5 (attention, l'échelle est logarithmique), on peut voir que coût du traitement du polynôme en une variable croit de façon dramatique lorsque le degré du polynôme augmente (exemples e5,e7,e8,e9), et la figure 4.6 montre une explosion systématique de la taille des coefficients apparaissant dans le résultat.

Pour indication, la figure 4.7 montre le comportement relatif des composantes de l'algorithme III. On peut noter que l'algorithme d'Uspensky semble réagir indépendamment de la taille des coefficients de départ, mais est cependant sensible au degré du polynôme traité. Cette méthode étant une méthode d'isolation, ceci n'a rien d'étonnant dans l'absolu.

4.4.2 Performances relatives des algorithmes

La figure 4.8 confirme la bonne tenue de la stratégie *multivariée*.
La mauvaise surprise vient du comportement de l'algorithme Sturm-Habicht qui étonnamment est plus coûteux, sur bon nombre d'exemples, que la réduction de

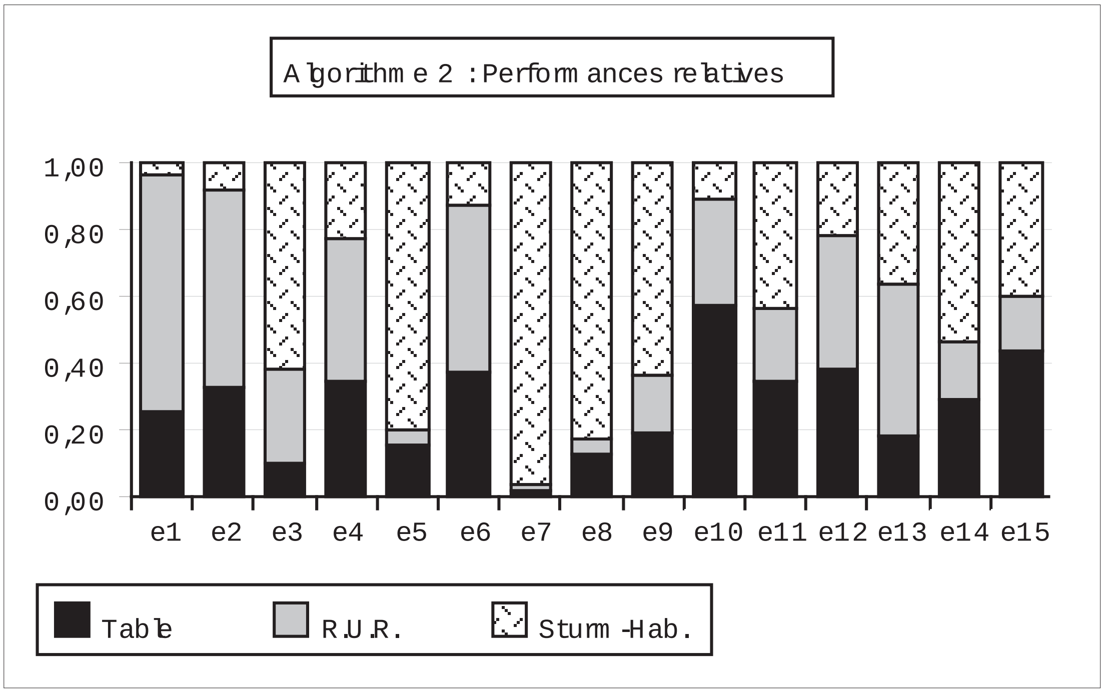

Figure 4.5: Algorithme II : Temps de calcul relatifs

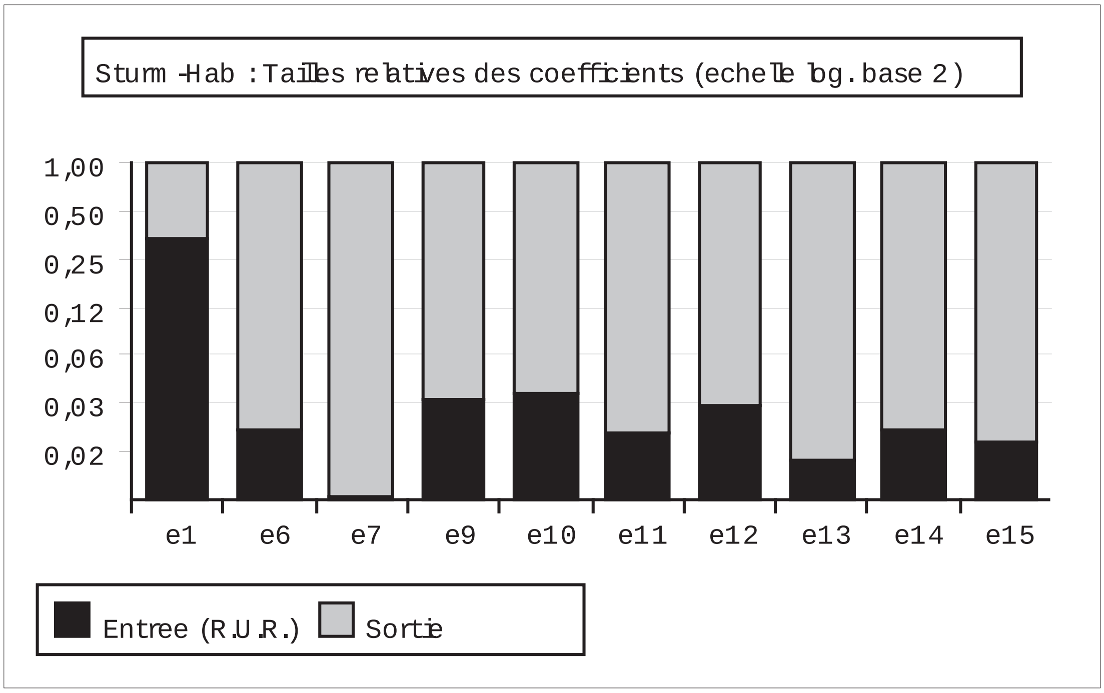

Figure 4.6: Sturm-Habicht : Croissance des coefficients

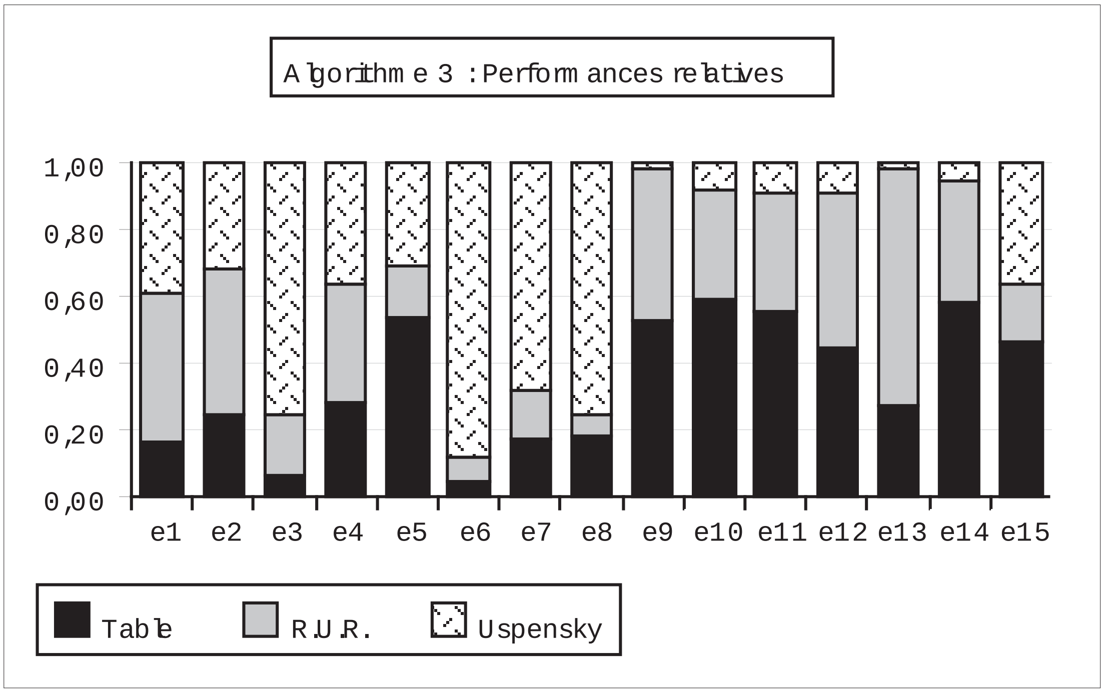

Figure 4.7: Algorithme III : Temps de calcul relatifs

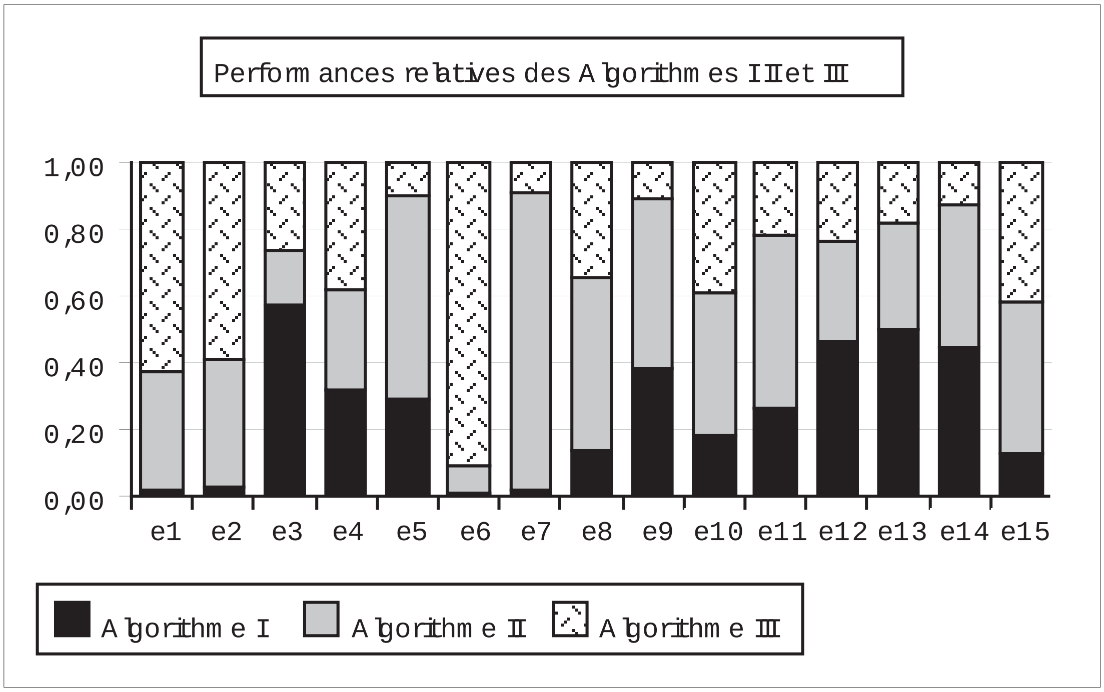

Figure 4.8: Algorithmes I, II et III : Temps de calcul relatifs

formes quadratiques. En théorie, le nombre d'opérations arithmétiques dans $\mathbb{Z}$ nécessaires pour le calcul de la suite de Sturm-Habicht est en $O(D^2)$ alors que celui de Hermite est en $O(D^3)$ sur des coefficients dont la taille maximale est équivalente. Cela dit, il faut modérer cette analyse en observant que les coefficients apparaissant dans une représentation univariée rationnelle sont relativement homogènes en taille, ce qui n'est pas le cas dans la forme quadratique de Hermite, $q_1[1, 1]$ étant par exemple égal à D dans tous les cas.

L'application de l'algorithme de Uspensky apporte une amélioration conséquente sur les problèmes où les racines des polynômes étudiés sont bien séparées, et s'avère complémentaire (voir figure 4.9) de la méthode de Sturm-Habicht sur notre panel d'exemples.

4.4.3 Conclusion

Notons le comportement sans faille de la table de multiplication tant au niveau de la croissance des coefficients qu'au niveau des temps de calculs, malgré une complexité théorique relativement élevée, en termes d'opérations arithmétiques. Quelle que soit la stratégie employée, ce pré-requis ne semble être en aucun cas un facteur limitant, sauf pour les exemples de large dimension où la place mémoire consommée peut

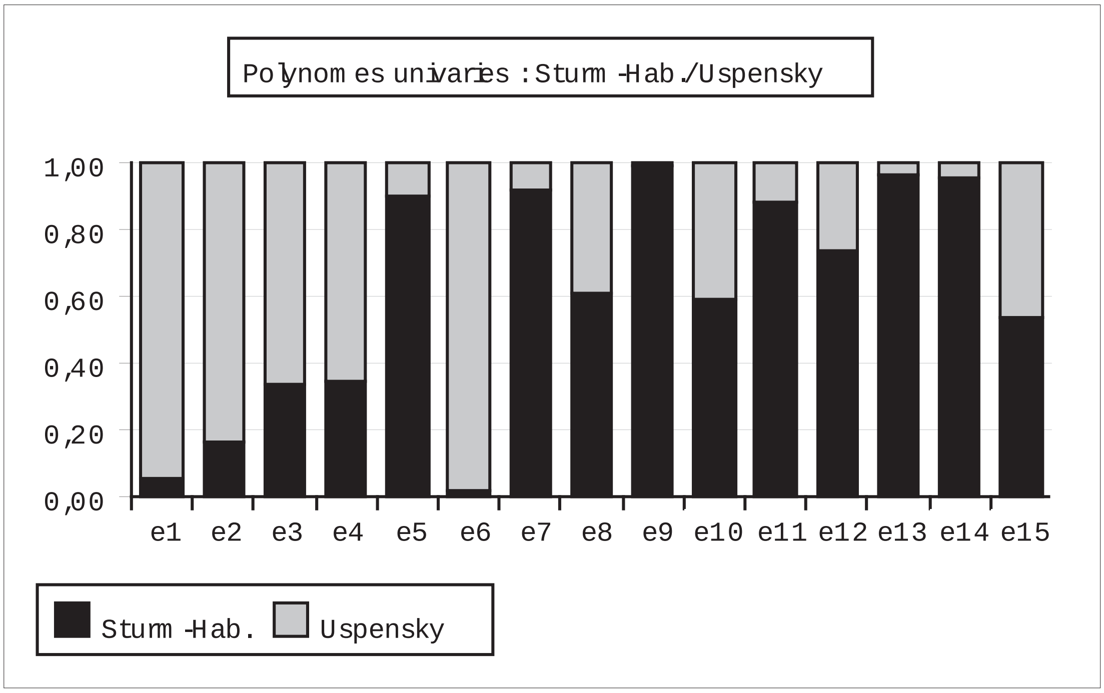

Figure 4.9: Méthodes *univariées* : Performances relatives

devenir excessive (pour l'exemple e7, l'occupation mémoire avoisine les 90Mo à la fin du calcul de table).

Le comportement le plus surprenant est celui de l'algorithme **RUR**. Présentant un nombre important d'opérations arithmétiques de base, il s'avère d'une grande efficacité pratique. Ceci est explicable par un bon contrôle de la taille des coefficients au cours des calculs, et confirme les remarques effectuées sur le faible nombre d'entiers premiers de mauvaise réduction (aucun n'a été détecté durant les tests) ainsi que sur la taille de la sortie (les coefficients intervenant dans les parties sans facteurs carrés étant, en général, plus petits que ceux des polynômes bruts).

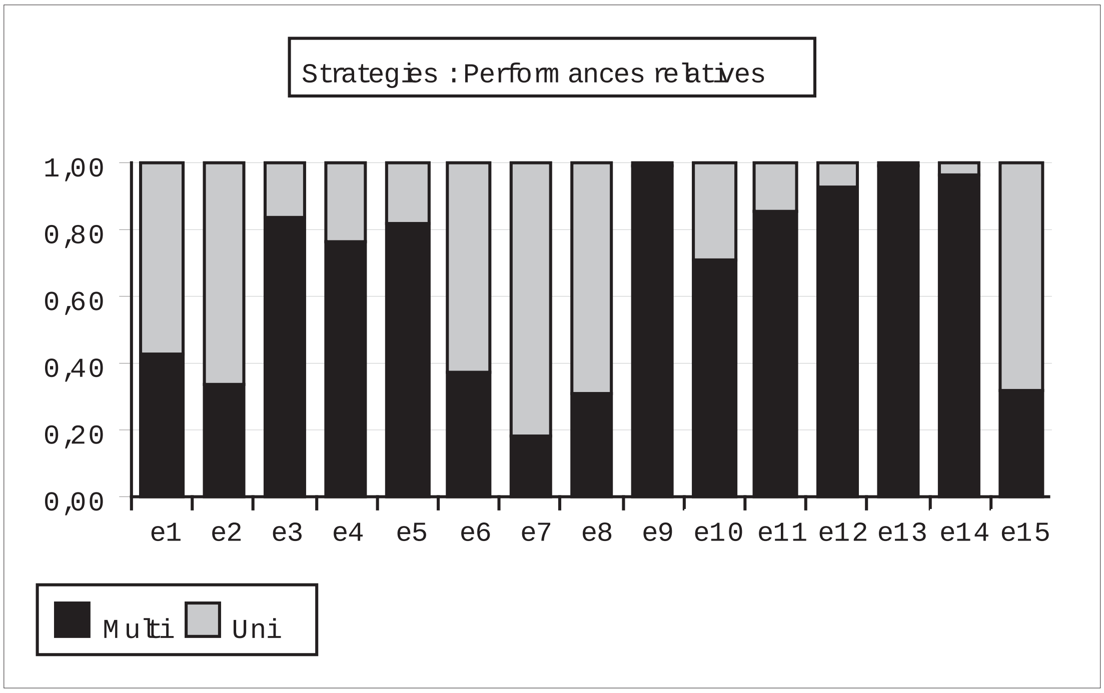

Figure 4.10: Stratégies : Temps de calcul relatifs

Sur la figure 4.10, pour chaque exemple, la méthode *multivariée* est comparée avec la méthode *univariée* la plus performante. On peut remarquer qu'une stratégie de type *univariée* se comporte en général mieux que son homologue *multivariée*. On peut toutefois remarquer que le rapport s'inverse lorsque aucune des variables ne sépare l'ensemble des zéros (exemples e1, e6 ,e8 , e15).

Compte tenu du nombre d'informations contenues dans la représentation univariée rationnelle , nous préférerons l'algorithme II ou III, d'autant que l'algorithme I ne permet apparemment pas de faire des calculs inaccessibles par une stratégie de type *univariée*.

Deuxième Partie

Applications

Chapitre 5

Robots parallèles

Les Robots parallèles passionnent un certain nombre de chercheurs de différents domaines depuis que Stewart en a proposé dès 1965 une application en tant que simulateur de vol.
Un robot parallèle est défini par :

- un groupe d'articulations (que nous noterons $\{A_1, \ldots, A_6\}$) situées sur une base fixe,

- un groupe d'articulations (que nous noterons $\{M_1, \ldots, M_6\}$) situées sur une plate-forme mobile par rapport à un référentiel lié à la base,

tels que tout couple d'articulations $(A_i, B_i)_{i=1\ldots6}$ soit relié par une *chaîne cinématique élémentaire équipée d'un seul moteur permettant d'actionner l'articulation correspondante de la plate-forme.*

Le problème qui nous à été proposé par J.P. Merlet et L. Tancredi (projet Prisme de l'INRIA Nice), ainsi que par D. Lazard et J.C. Faugère (Université de Paris 6) est l'étude du problème dit *Direct* :

> En supposant connues :
>
> - la géométrie de la plate-forme (c'est à dire la position des points $M_1, \ldots, M_s$ dans un référentiel lié à la plate-forme),
> - la géométrie de la base,
> - la longueur des liaisons $(A_i, B_i)_{i=1\ldots6}$ (supposées droites et rigides),
>
> **Quelles sont les positions possibles de la plate-forme ?** En supposant que ni les articulations $M_1, \ldots, M_6$ ni les articulations $A_1, \ldots, A_6$ ne soient alignées (dans la cas contraire, la plate-forme ou la base pourrait tourner librement autour d'un axe et le robot aurait pas conséquent une infinité de positions possibles).

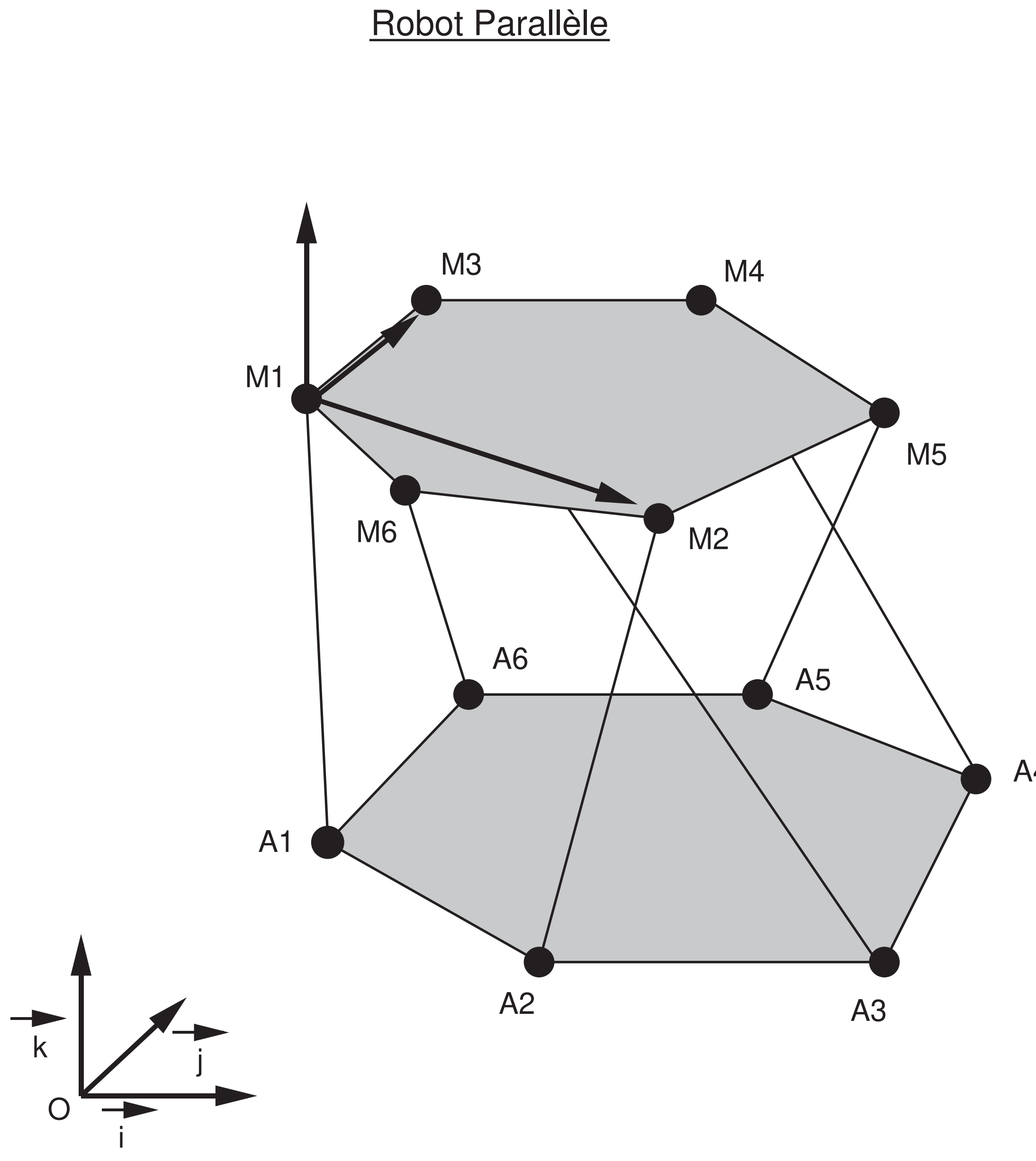

Figure 5.1: Un exemple de robot parallèle

Bien sur, il est possible, par certains algorithmes numériques, de déterminer avec une bonne précision les solutions de tels problèmes, mais l'utilisation du calcul exact devient intéressante pour les spécialistes dans la recherche du nombre de positions dans l'espace d'un robot donné.

Bien que le nombre maximum de positions d'un robot parallèle dans l'espace complexe soit connu (40), et ait été calculé par divers auteurs (voir par exemple [Mou] ou [Laz92b]), le problème reste entier pour le nombre de solutions réelles. Jusqu'à ce jour, le nombre maximum de positions réelles obtenu sur un robot sans articulation double est de 16 (voir [Mer90]) et de 24 sur un robot particulier ayant une articulation double (voir [IPC93]). Quelques calculs de bornes ont été tentés. En particulier, D. Lazard conjecture dans [Laz92b] que le nombre maximum de positions réelles est majoré par 16 ou 32, selon la configuration du robot.

5.1 Les équations

Ce type de problème à entraîné énormément de complications du coté du calcul de bases de Groebner qui constituent, rappelons le, le point d'entrée de la plupart de nos algorithmes. Aussi, nous avons choisi la mise en équations proposée par D. Lazard dans [Laz92b], celle-ci ayant déjà permis, via un calcul de base de Groebner, de vérifier de façon calculatoire et effective, la borne (40) sur le nombre de positions dans l'espace complexe d'un robot parallèle.

Comme énoncé en introduction, nous supposons que ni les points $M_1, \ldots, M_6$, ni les points $A_1, \ldots, A_6$ ne sont alignés, et quitte à les numéroter différemment, nous supposons que les vecteurs $\overrightarrow{M_1M_2}$ et $\overrightarrow{M_1M_3}$ (resp. $\overrightarrow{A_1A_2}$ et $\overrightarrow{A_1A_3}$) ne sont pas colinéaires. Dans ce cas, nous pouvons prendre comme base liée à la plate-forme, le système :

$$\{\overrightarrow{M_1M_2}, \overrightarrow{M_1M_3}, \overrightarrow{M_1M_2} \wedge \overrightarrow{M_1M_3}\}$$

La position de la plate-forme dans le référentiel fixe (lié à la base) est entièrement déterminée lorsque l'on connaît les points M_1, M_2, M_3.

Le but est alors de construire un nombre suffisant d'équations polynomiales qui nous permettront de déterminer les coordonnées de ces points. Nous notons désormais (x_i, y_i, z_i) les coordonnées de M_i $(i = 1, 2, 3)$ dans le référentiel fixe.

La géométrie de la plate-forme étant connue, nous pouvons alors, pour $i = 4 \ldots 6$, déterminer des scalaires e_i, f_i, g_i tels que :

$$\overrightarrow{M_1M_i} = e_i\overrightarrow{M_1M_2}, f_i\overrightarrow{M_1M_3}, g_i\overrightarrow{M_1M_2} \wedge \overrightarrow{M_1M_3}.$$

Les longueurs d_i des segments (A_i, M_i), $i = 1 \ldots 6$ étant connues, nous avons, sans effort, une première série de 6 équations, dont les inconnues sont les coordonnées des points M_1, M_2, M_3:

$$||\overrightarrow{A_iM_i}||^2 = d_i^2,$$

ce qui nous donne :

$$||\overrightarrow{A_iM_1} + \overrightarrow{M_1M_i}||^2 = ||\overrightarrow{A_iM_1} + e_i\overrightarrow{M_1M_2}, f_i\overrightarrow{M_1M_3}, g_i\overrightarrow{M_1M_2} \wedge \overrightarrow{M_1M_3}||^2 = d_i^2.$$

Trois autres équations sont obtenues en exprimant, à l'aide des coordonnées des points M_1, M_2, M_3 dans le référentiel fixe, les quantités connues par hypothèses (la géométrie de la plate-forme est connue) :

- Les carrés des normes des vecteurs $\overrightarrow{M_1M_2}$ et $\overrightarrow{M_1M_3}$
- Le produit scalaire $\overrightarrow{M_1M_2} \cdot \overrightarrow{M_1M_3}$

Quelques raffinements, que nous ne décrirons pas ici, visant à diminuer le nombre de monômes présents dans le système, sont également envisageables.

Un exemple de système : Pour donner une idée de la difficulté du problème à résoudre, nous donnons un exemple de système généré à partir du procédé décrit plus haut : Il s'agit d'un problème de type général où tous les joints sont distincts et où la plate-forme et la base ne sont pas planes :

```
[y1^2+z1^2+x1^2-74529,
y2^2+z2^2-33241775,
10*x3^2+10*z3^2+10*y3^2+100*x3-1799*y3-21964,
y2*y1+z1*z2-1271*x1+675424,
80*x3*x1+80*y1*y3+80*z3*z1+2000*x3-35980*y3-805427,
25*y2*y3+25*z3*z2-23775*x3-143920*y3+1585944,
11380000*x3*y2*z1-11380000*x3*y1*z2-
11380000*y2*z3*x1+11380000*y3*z2*x1-
5172210000*x3*y2+66204288000*x3*y1+644563200*x3*z1-
50356500*x3*z2+1547680000*y2*z3-309536000*y2*z1+
1034442000*y2*x1-34274284000*y1*z3+309536000*y1*z2-
644563200*z3*x1+34274284000*z1*y3-1547680000*y3*z2-
66204288000*y3*x1+10071300*z2*x1+23403907968000*x3+
1844972775*y2+1503728861040*y1+66004792984500*z3-
19394922411300*z1-72595431092400*y3-189500593500*z2-
5807371468800*x1-137594268684108,
-97200*x3*y2*z1+97200*x3*y1*z2+97200*y2*z3*x1-
97200*y3*z2*x1-44031600*x3*y2+563604480*x3*y1+
426124800*x3*z1-33291000*x3*z2+25174800*y2*z3-
5034960*y2*z1+8806320*y2*x1-198696240*y1*z3+
5034960*y1*z2-426124800*z3*x1+198696240*z1*y3-
25174800*y3*z2-563604480*y3*x1+6658200*z2*x1+
15862691520*x3-418891200*y2+4959464240*y1-
11604317160*z3+16706998248*z1-48693763320*y3+
554037120*z2+4713700544*x1+621265957475];
```

5.2 La résolution dans la pratique

L'étude du nombre de positions réelles de robots parallèles à été effectuée en deux temps. La première phase à été une phase de validation de nos méthodes, visant à montrer qu'elles peuvent s'appliquer dans des cas jugés extrêmes pour les calculs de bases de Groebner.

Une deuxième phase à consisté à étudier le comportement de diverses stratégies pour le comptage des zéros réels, sur des problèmes significatifs pour les équipes travaillant dans ce domaine, dans le but de proposer une solution *clés en mains*.

Nous n'exhibons ici que la deuxième partie des tests visant à comparer la méthode de comptage directe (algorithme I) des racines réelles, par l'algorithme de Hermite, avec des stratégies se ramenant à un problème en une variable (algorithmes II - RUR+Sturm - et III - RUR + Uspensky -).
Ces exemples sont tirés d'études spécialisées ([Dit95], [Fau94c], [Mer90], ...)
Légende :

- RUR : Calcul de la représentation univariée rationnelle.
- St-Habicht : Algorithme de Sturm-Habicht appliqué sur le premier polynôme de la représentation univariée rationnelle.
- Uspensky : Algorithme de Uspensky appliqué sur le premier polynôme de la représentation univariée rationnelle.
- Hermite : Algorithme de Hermite.
- Racines :
  - R : nombre de racines réelles.
  - C : nombre de racines complexes.
- T. : temps de calcul en secondes sur station DEC Alpha-600 - 175 Mhz .

Nous ne revenons pas ici sur les détails concernant les pré-requis de nos divers algorithmes (base de Groebner, table de multiplication), notre but étant essentiellement de choisir une stratégie pour le comptage des zéros réels.

Nom	RUR	St-Habicht	Uspensky	Hermite	R	C
e9	5,71	21,77	0,17	20.8	0	40
e10	3,4	1,21	0,85	2.03	0	16
e11	16,8	33,5	4,37	25,5	0	32
e12	27,5	14,7	5,36	65,2	0	24
e13	460,2	360,2	11,9	1277	2	40
e14	135,2	407,7	21,25	554,11	24	40
e15	3,65	8,88	7,79	3,6	8	40

Plusieurs constatations ont pu être effectuées au travers de cette panoplie de tests :

- Les exemples *génériques* pour lesquels un nombre maximum de positions complexes peuvent être dénombrées (voir [Laz92b] - exemples $e9$ à $e12$), n'ont en général pas ou peu de solutions réelles.

- En ce qui concerne la stratégie à adopter pour le calcul du nombre de racines réelles, il semble que les algorithmes II et III se comportent asymptotiquement mieux, une forte proportion du temps de calcul étant réservée à la résolution du problème en une variable dans le cas de l'algorithme II (RUR + Sturm-Habicht). A priori, les racines sont facilement séparables puisque l'algorithme III se révèle d'une grande efficacité.

5.3 Histoire d'un robot

Après avoir déterminé les méthodes les plus favorables pour compter le nombre de positions réelles d'un robot parallèle, en collaboration avec L.Tancredi, nous avons voulu étudier le nombre maximal de positions d'une configuration générale (ne présentant pas de joints confondus et telle que la base ainsi que la plate-forme ne soient pas planes).

Jusqu'à ce jour, quelques exemples de ce type ayant 16 positions étaient connus (voir [Mer90]). Notre idée à été de modifier un robot particulier ayant 24 positions (nombre maximum mis en évidence) (voir [IPC93]).

La configuration considérée est décrite dans le tableau suivant où les coordonnées des $A_i, (i = 1, \ldots, 6)$ sont données dans le repère de référence et celles des $M_i, (i = 1, \ldots, 6)$ sont données dans le repère de la plate-forme ; la troisième colonne correspond à la longueur des segments.

i	A_i	M_i	ρ_i
1	0 0 0	0 0 0	1.0925
2	1 0 0	-2.5073 0 0	2.2804
3	-0.1 1.7982 0	-0.333 0.5913 0	2.03125
4	0.5439 -0.0176875 1.8184	-0.1948 -0.9060375 0.5692	2.5243
5	-1.036 1.3698 -1.8127	-0.9234 0.1692 -1.9444	3.1355
6	0 0 0	-2.5073 0 0	1.844974

On peut remarquer que ce robot possède deux joints confondus (A_1 et A_6).

Pour construire un robot général à partir de cet exemple, il suffit, en théorie, de considérer une déformation suffisamment petite de sa base, pour préserver le nombre de positions.

Dans un premier temps, nous nous sommes limités à définir, en pratique, cette notion de *petite déformation* en agissant sur le module des coordonnées des divers joints, et nous nous sommes aperçus qu'en effet, pour toutes les configurations testées pour lesquelles les points modélisant les joints restaient dans des boules de rayon 10^{-3} centrées sur les valeurs initiales, le nombre de positions réelles restait constant et égal à 24. Ceci nous a permis, entre autres, de réduire la taille des coefficients intervenant dans nos calculs, de remarquer que le nombre de solutions a tendance à diminuer par paquets de 8 lorsque la déformation est trop importante, mais surtout de construire le premier robot parallèle de configuration générale ayant 24 positions réelles :

i	A_i	M_i	ρ_i
1	0 0 0	0 0 0	1.092
2	1 0 0	-2.507 0 0	2.280
3	-0.1 1.799 0	-0.333 0.591 0	2.031
4	0.544 -0.0177 1.818	-0.195 -0.906 0.569	2.524
5	-1.036 1.370 -1.813	-0.923 0.169 -1.944	3.135
6	0 0 0.001	-2.508 0 0	1.845

Enfin, un dernier travail nous a permis, à partir de la représentation univariée rationnelle des racines du système, de reconstituer numériquement avec une grande précision (de l'ordre de 10^{-40}) les 24 positions de ce robot, en utilisant les intervalles rationnels fournis par la méthode de Uspensky pour guider les recherches numériques.

Ces problèmes de robots parallèles nous ont permis de tester le comportement d'algorithmes expérimentaux utilisant massivement de nouvelles techniques alliant calcul parallèle et modulaire dans un environnement de calcul de type distribué. Nos observations à ce sujet sont très encourageantes tant au niveau du calcul de bases de Groebner (J.C. Faugère) qu'au niveau des algorithmes pour l'étude des zéros réels. Notre exemple test qui prenait plus de 20 heures cpu à pu être effectué, lors d'une démonstration, en quelques secondes.

Pour conclure ce chapitre, nous terminerons par une petite note d'humour au sujet d'une remarque de D.Lazard dans [Laz92b], décrivant le calcul de bases de Groebner sur les problèmes de robots parallèles :

> *First of all, it appears that exact computations involve integers which are to big to be usefull in practical computations*

... effectivement, nul n'aurait pensé pouvoir réaliser ce type de calcul en des temps décents il y a quelques années (1993), bon nombre de progrès ayant d'ailleurs été rendus possibles grâce aux nombreuses contributions du groupe de D.Lazard et en particulier de J.C. Faugère.

Chapitre 6

Synthèse de filtres

Dans ce chapitre, nous décrivons une application en traitement de signal dont l'étude à été menée en collaboration avec Francois Moreau (C.C.E.T.T. Rennes) et Jean-Charles Faugère (L.I.T.P. Paris VI).

Après un bref résumé des définitions de base servant dans la formulation du problème, nous décrivons, sans rentrer dans les détails, le problème posé, puis nous montrons comment les diverses stratégies développées dans cette thèse ont contribué à faire progresser les choses.

6.1 Éléments de base

Un signal modélise une grandeur physique variant dans le temps (son, image). Nous supposerons ici qu'un signal est la donnée d'une suite discrète d'informations, ce qui correspond à la modélisation standard en pratique (transmission de données, compression, ...).

La théorie de l'information permet de dire que l'on peut, sous certaines conditions, échantillonner un signal (de nature continue $x(t)$) sans perdre d'information. C'est le cas, par exemple, des signaux à bande limitée (transformée de Fourier à support compact).

Un signal sera, en ce qui nous concerne, la donnée d'une suite d'éléments $X = (x_i)_{i\in\mathbb{Z}}$ de $\mathbb{R}^n$ que nous supposerons sommable (i.e. : $\sum_{i=-\infty}^{+\infty} |x_i| < \infty$) et de carré sommable. Ceci n'est pas, en pratique, une contrainte, les volumes d'informations traités restant forcément finis.

Cette représentation , dite temporelle, admet une forme duale (*représentation fréquentielle*). En théorie, la *représentation fréquentielle* d'un signal en temps continu (suffisamment régulier) est la donnée de sa transformée de Fourier. Dans le cas d'un signal discret de représentation temporelle $X = (x_i)_{i\in\mathbb{Z}}$, celle-ci peut se déduire simplement de la transformée en Z de X :

$$\hat{X}(z) = \sum_{k=-\infty}^{+\infty} x_{-k} z^{-k}$$

(La *représentation fréquentielle* de X est donnée par la fonction $f \longmapsto \hat{X}(e^{2i\pi f})$).

Lorsque l'on veut, par exemple, compresser l'information contenue dans un signal X donné, on applique une transformation, de préférence réversible, définie sous le terme générique de filtrage.

Une opération de filtrage consiste à transformer un signal X en un autre signal. Mathématiquement, ceci se définit par la convolution de X et d'une mesure de Radon. Dans notre cas (signaux discrets), il s'agit simplement de la convolution de deux suites et nous nous limiterons même au cas de filtres linéaires à support fini, traduisant ainsi l'opération de filtrage par la convolution de X et d'un polynôme de Laurent.

La transformée de Fourier d'une convolution étant égale au produit des transformées de Fourier (lorsque les fonctions sont suffisamment régulières), lorsque l'on considère les représentations fréquentielles, l'image de $\hat{X}(z)$ par un filtre linéaire à support fini est donc de la forme

$$\hat{X}(z)H(z),$$

où $H(z)$ est un polynôme de Laurent.

Une catégorie particulière de filtres nous intéresse : les filtres *passe-bande* et en particulier les filtres *passe-haut* et *passe-bas*.

Un filtre *passe-bas* (resp. *passe-haut*) est une transformation permettant d'atténuer la contribution des fréquences élevées (resp. faibles). Rappelons que l'expression fréquentielle d'un signal est une fonction périodique ($f \longmapsto \hat{X}(e^{2i\pi f})$), et que nous entendons par *basse fréquence* (resp. *haute fréquence*), les valeurs de f proches de $0 + n$, $n \in \mathbb{Z}$ (resp. $1/2 + n$, $n \in \mathbb{Z}$).

Un exemple simple :

Considérons le filtre transformant tout signal de représentation fréquentielle $\hat{X} = (x_i)_{i\in\mathbb{Z}}$ en $Y_0 = (y_{0,i} = (x_i - x_{i-1})/2)_{i\in\mathbb{Z}}$. D'après ce qui précède, cette transformation peut s'exprimer par :

$$Y_0 = \hat{X}(z)H_0(z)$$

avec $H_0(z) = (1 - z)/2$. Si l'on exprime Y_0 en fonction de f ($e^{2i\pi f} = z$), on obtient alors :

$$Y_0(f) = e^{i\pi f} sin(\pi f)\hat{X}(f)$$

ce qui montre que ce filtre est de type *passe-haut* (on peut voir, par exemple, que la réponse du filtre est nulle pour $f = 0$).

De la même façon, le filtre transformant tout signal de représentation fréquentielle $\hat{X} = (x_i)_{i\in\mathbb{Z}}$ en $Y_1 = (y_{1,i} = (x_i + x_{i-1})/2)_{i\in\mathbb{Z}}$ (i.e. : $Y_1 = \hat{X}(z)H_1(z)$, $H_1(z) = (1+z)/2$) est un filtre de type *passe-bas*.
Par construction de Y_0 et Y_1, nous avons :

$$\begin{cases} y_{0,2n} + y_{1,2n} = x_{2n} \\ y_{0,2n} - y_{1,2n} = x_{2n-1} \end{cases},$$

montrant ainsi que seule la connaissance des termes pairs des suites Y_0 et Y_1 est nécessaire pour reconstituer le signal initial. Ceci peut être considéré comme un shéma de base pour la compression de données si on suppose nulle, par exemple, l'information contenue dans Y_1.

Pour tout signal $X = (x_i)_{i\in\mathbb{Z}}$, nous noterons $X \downarrow 2 = (y_i = x_{2i})_{i\in\mathbb{Z}}$ (sous-échantillonnage) et $X \uparrow 2 = (y_2 i = x_i \ , \ y_{2i+1} = 0)_{i\in\mathbb{Z}}$ (sur-échantillonnage).

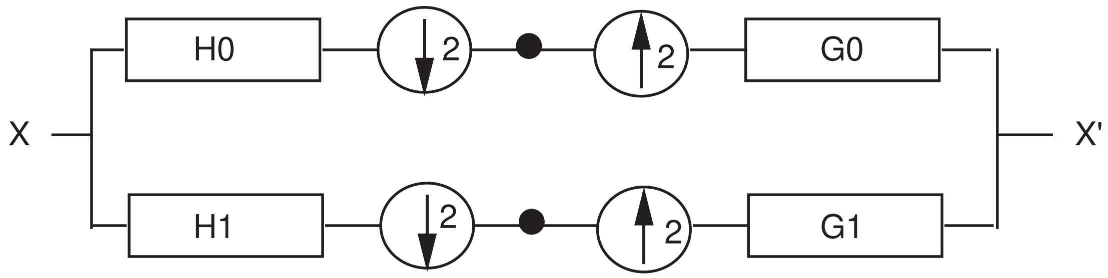

Figure 6.1: Décomposition par des filtres *passe-bande*

La figure 6.1 résume une opération de filtrage élémentaire (dans le cas de signaux uni-dimensionnels), pour laquelle le but est de trouver des filtres G_i, H_i tels que $\hat{X}' = \hat{X}$.
L'expression algébrique résumant le shema étant :

$$\begin{aligned} \hat{X}' = \ & 1/2G_0(Z)[\hat{X}(Z)H_0(Z) + \hat{X}(-Z)H_0(-Z)] \\ & +1/2G_1(Z)[\hat{X}(Z)H_1(Z) + \hat{X}(-Z)H_1(-Z)] \end{aligned},$$

cette condition s'exprime simplement en annulant le coefficient de $\hat{X}(-Z)$ dans :

$$\begin{aligned} \hat{X}' = \ & 1/2\hat{X}(Z)[G_0(Z)H_0(Z) + G_1(Z)H_1(Z)]+ \\ & 1/2\hat{X}(-Z)[G_0(Z)H_0(-Z) + G_1(Z)H_1(-Z)] \end{aligned}$$

En posant :

$$G_0(Z)H_0(-Z) + G_1(Z)H_1(-Z) = 0 \ , \tag{6.1}$$

nous obtenons, à un facteur retard près (terme de la forme Z^{-d} , d impair) :

$$G_1(Z) = H_0(-Z) \;\; H_1(Z) = -G_0(-Z) \tag{6.2}$$

En réalisant une décomposition *polyphase* des filtres H_i et G_i, c'est à dire en posant :

$$G_i(Z) = G_{i0}(Z^2) + ZG_{i1}(Z^2)\ ,\ i = 0, 1$$

$$H_i(Z) = H_{i0}(Z^2) + ZH_{i1}(Z^2)\ ,\ i = 0, 1$$

on peut exprimer la relation (6.1) sous forme matricielle :

$$\begin{bmatrix} G_{00} & G_{01} \\ G_{10} & G_{11} \end{bmatrix} \begin{bmatrix} H_{00} & H_{10} \\ H_{01} & H_{11} \end{bmatrix} = IdZ^{-d'}$$

Si on impose que la base sur laquelle on décompose le signal (expression obtenue après filtrage et sous échantillonnage) est *orthogonale*, l'expression précédente conduit alors à une égalité de la forme :

$$H_0(Z) = Z^L G_0(1/Z)$$

Le problème décrit par la figure 6.1 se résume par conséquent à l'étude d'un unique polynôme de Laurent, par exemple H_0 (*synthèse de filtre*).

La décomposition d'un signal $\hat{X}$ par le biais de filtres *passe-haut* et *passe-bas* (figure 6.1) peut être itérée comme décrit par la figure 6.2, on définit ainsi un *banc de filtres itérés*.

Par des lois classiques de permutations entre filtrages et sous/sur-échantilonnages, il est possible de modéliser chaque branche par une composée plus simple. Par exemple la branche la plus *haute* (ne faisant intervenir que les filtres H_0 et G_0) peut être résumée par un diagramme identique à celui de la figure 6.2, où, dans le cas de k itérations, le filtre itéré H s'exprimera simplement par :

$$H(z) = H_0(Z)H_0(Z^2)\ldots H_0(Z^{2k-1})$$

La fonction limite du processus itéré établit alors le lien entre l'étude du banc de filtres itérés et la théorie des *ondelettes*.

6.2 Le problème

On cherche à synthétiser un filtre *passe-bas* générant une base d'ondelettes orthogonale. Sans entrer dans les détails, disons que ceci revient à définir un polynôme (filtre) H_0, satisfaisant un certain nombre de relations algébriques et optimum pour un certain critère que nous préciserons selon le cas étudié.

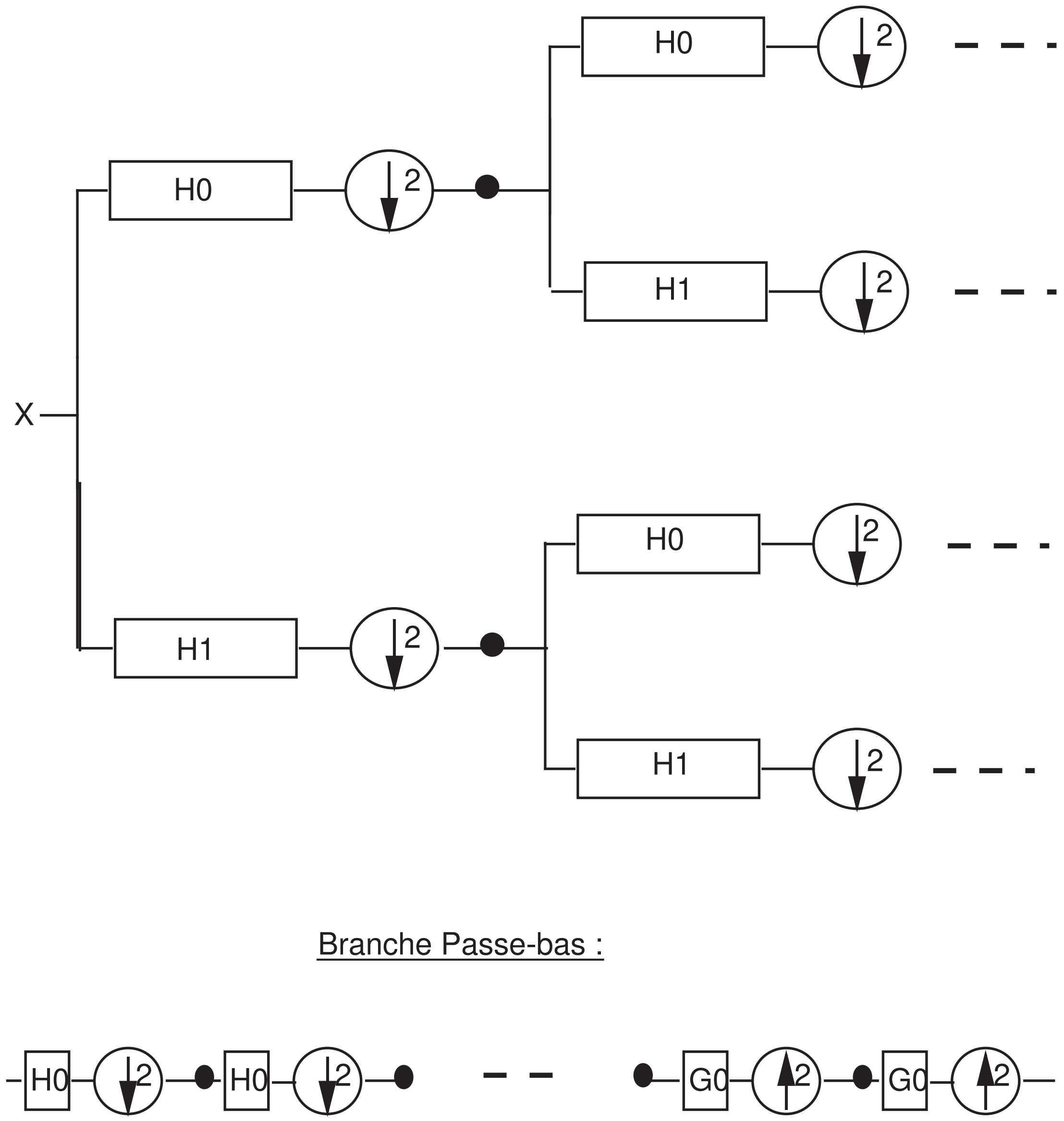

Figure 6.2: Itération de filtres *passe-bande*

Par exemple, dans le cas de signaux $1D$ (à valeurs dans $\mathbb{R}$), les contraintes algébriques caractérisant les filtres générant une base d'ondelettes orthogonale permettent détablir une paramétrisation des filtres H_0 sous la forme :

$$H_0(z) = \begin{bmatrix} \cos\theta_0 & \sin\theta_0 \\ -\sin\theta_0 & \cos\theta_0 \end{bmatrix} \left(\prod_{k=1}^{K-1} \begin{bmatrix} 1 & 0 \\ 1 & z^2 \end{bmatrix} \begin{bmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{bmatrix} \right) \begin{bmatrix} 1 \\ z \end{bmatrix} \tag{6.3}$$

La décomposition des solutions sous forme d'une telle *cascade* est largement reconnue par les spécialistes, à cause de la simplicité de la paramétrisation, pour l'efficacité des implantations hard-ware qui l'utilisent, et enfin et surtout parce que c'est une paramétrisation complète de l'espace sur lequel on effectue l'optimisation, sans aucune redondance.

Nous nous intéressons ici au cas des signaux $2D$ (bidimensionnels) pour lesquelles les choses se présentent de façon un peu plus complexe.

La décomposition d'un signal $\hat{X}$ par le biais de filtres (polynômes de Laurent en deux variables) *passe-bande* itérés s'effectue de la même manière que dans le cas $1D$.

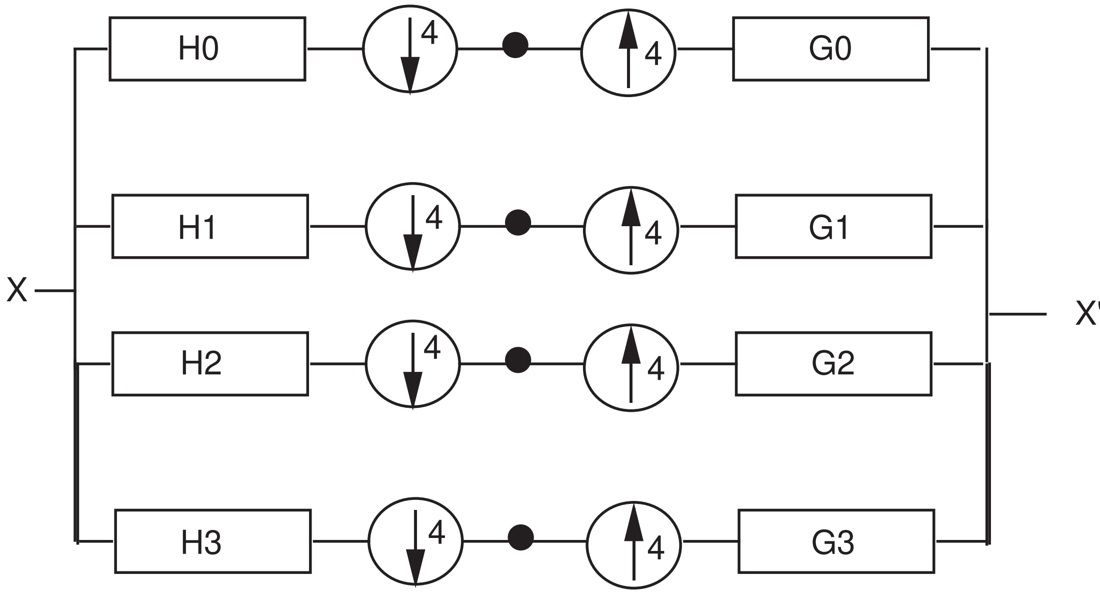

Figure 6.3: Décomposition d'un signal $2D$

Si l'on se place dans le cadre d'un échantillonnage de type $2I$ (système à 4 sous bandes séparable), la figure 6.3 résume une situation que nous pouvons décrire de la façon suivante :

- La représentation temporelle d'un signal $2D$ est la donnée d'une suite $X = (x_{i,j})_{(i,j)\in\mathbb{Z}^2}$ d'éléments de $\mathbb{R}^2$

- Sa représentation fréquentielle est alors déductible à partir de l'expression :

$$\sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} x_{i,j} z_0^{-i} z_1^{-j}$$

- Les sous-échantillonnages consistent alors à prendre simplement les termes de la forme $x_{2i,2j}$ dans l'expression de $\hat{X}$.
- Dans le cas $2D$, la notion de filtre *passe-bas* est similaire à celle du cas $1D$ (réponse en fréquence faible sauf autour des points entiers), la définition d'un filtre *passe-haut* n'ayant, en général, pas de sens précis.

Comme dans le cas $1D$, l'expression fréquentielle du signal filtré s'exprime par une égalité de la forme :

$$\hat{X}' = A_1 X(z_1, z_2) + A_2 X(-z_1, z_2) + A_3 X(z_1, -z_2) + A_4 X(-z_1, -z_2)$$

et la *reconstruction parfaite* impose alors

$$A_2 = A_3 = A_4 = 0$$

Toujours comme dans le cas $1D$, on peut itérer le processus (voir figure 6.4), définissant ainsi un filtre H_0, équivalent à la branche *passe-bas* et s'exprimant sous la forme d'un polynôme dépendant de $(z_1, z_2, z_1^{-1}, z_2^{-1})$ et de k (nombre d'itérations), dont la limite, lorsque $k \to \infty$, établit le lien avec la théorie des *ondelettes*.

Dans le cas $2D$, nous n'avons plus de description complète de l'espace (simplement paramétrée) des filtres à étudier et nous limitons l'étude à celle d'une cascade de type **Vetterli-Kovacevic PL** (voir [KV95]) présentant un certain nombre de *bonnes* propriétés :

- orthogonalité de la base de décomposition.
- filtres (i.e. les fonctions de base) symétriques ou antisymétriques par rapport au centre de leur support.

On note:

$$R = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i & 0 & 0 \\ \sin\alpha_i & \cos\alpha_i & 0 & 0 \\ 0 & 0 & \cos\beta_i & -\sin\beta_i \\ 0 & 0 & \sin\beta_i & \cos\beta_i \end{bmatrix} \tag{6.4}$$

$$W = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{6.5}$$

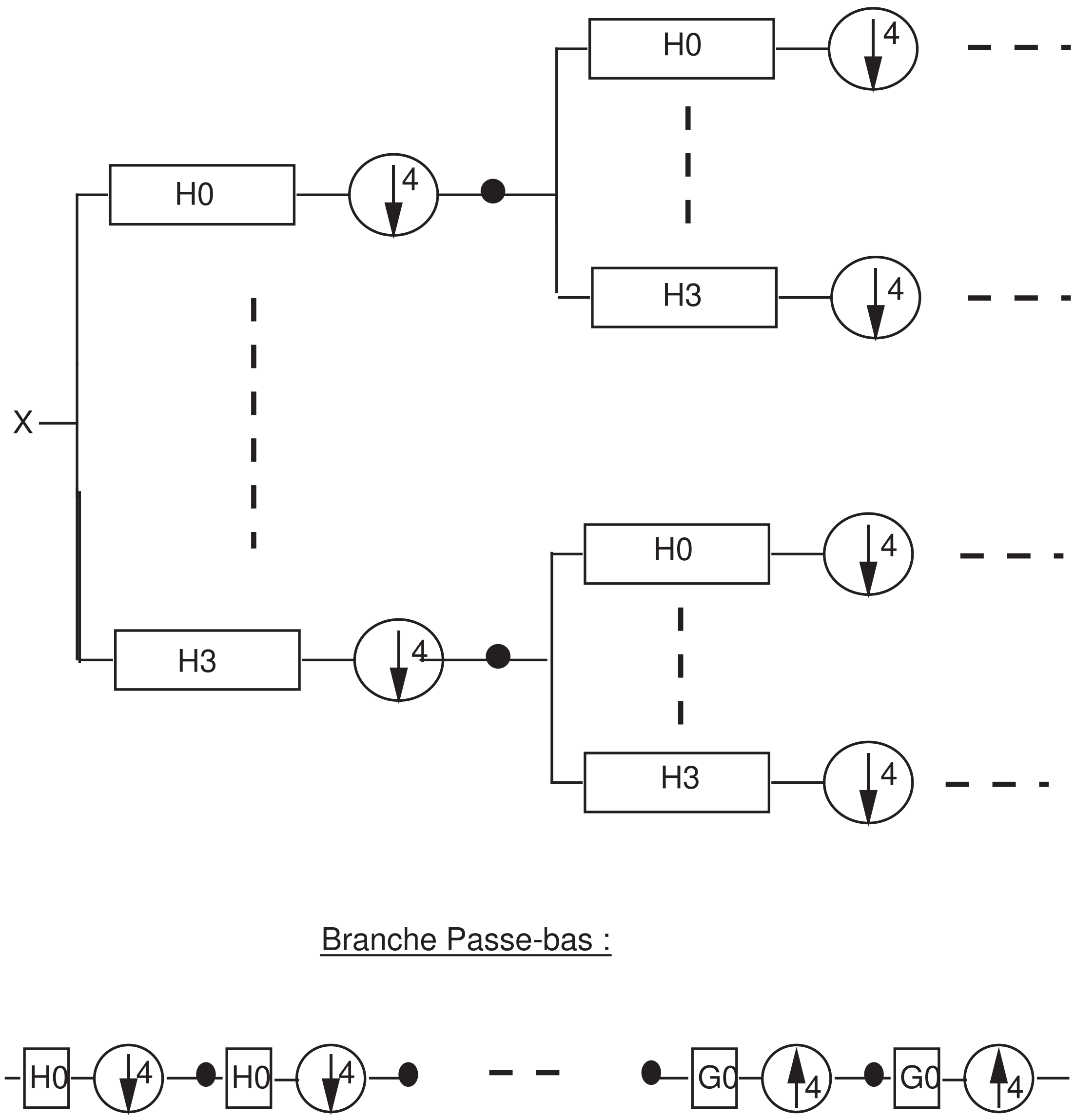

Figure 6.4: Filtres itérés, cas $2D$

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.6}$$

$$D(z_1, z_2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & z_1 & 0 & 0 \\ 0 & 0 & z_2 & 0 \\ 0 & 0 & 0 & z_1 z_2 \end{bmatrix} \tag{6.7}$$

Avec ces définitions, on construit des matrices MH_0 en posant :

$$MH_0(z) = R_0 W P \prod_{i=1}^{k} D(z_1, z_2) P W R_i W P \tag{6.8}$$

et on considère les filtres H_0 définis par :

$$H_0(z) = H_{0,0}(z_1^2, z_2^2) + H_{0,1}(z_1^2, z_2^2) z_1 + H_{0,2}(z_1^2, z_2^2) z_2 + H_{0,3}(z_1^2, z_2^2) z_1 z_2 \tag{6.9}$$

(les filtres H_1 , H_2 et H_3 sont définis de la même manière)
La synthèse consiste alors à déterminer les angles α_i, β_i pour que la matrice MH_0 satisfasse certaines propriétés ou bien soit optimale par rapport à certains critères.

Le critère que nous avons choisi d'étudier est celui de platitude maximale : les caractéristiques de platitude (directement reliées à la régularité des ondelettes sous-jacentes) se rattachent a deux notions :

- régularité de la fonction limite du processus de subdivision. Pour cela, il faut imposer un certain nombre de propriétés sur H_0. En particulier, il est souhaitable qu'un maximum de dérivées partielles du polynôme H_0 s'annulent en $(-1, 1)$, $(-1, -1)$ et $(1, -1)$. Si N_l caractérise l'ordre de platitude du filtre, cela signifie que les équations suivantes sont satisfaites:

$$\forall k_1, k_2,\ k_1 + k_2 \leq N_l, \qquad \frac{\partial^{k_1+k_2} H_0}{\partial z_1^{k_1} \partial z_2^{k_2}}(-1, -1) = 0, \tag{6.10}$$

$$\frac{\partial^{k_1+k_2} H_0}{\partial z_1^{k_1} \partial z_2^{k_2}}(-1, 1) = 0, \tag{6.11}$$

$$\frac{\partial^{k_1+k_2} H_0}{\partial z_1^{k_1} \partial z_2^{k_2}}(1, -1) = 0 \tag{6.12}$$

- nombre maximum de composantes nulles sur les bandes hautes fréquences. Pour cela, il est souhaitable qu'un maximum de dérivées partielles du polynôme H_1, (resp. H_2, H_3) s'annulent aux points $\{(1, 1),\ (-1, 1)$ et $(1, -1)\}$ (resp. $\{(-1, -1),\ (-1, 1)$ et $(1, 1)\}$, $\{(-1, -1),\ (1, 1)$ et $(1, -1)\}$) Comme pour le point précédent, ceci nous fournit alors un ensemble de N_h équations.

Pour fixer les idées sur la démarche à suivre, les équations indexées par N_h engendrent plus de *bonnes* propriétés sur les ondelettes sous-jacentes et seront à considérer en priorité.
Notre but sera, par conséquent, pour K fixé, de trouver la valeur maximale de N_l telle que le système constitué par la première série d'équations admette des solutions réelles. Dans un second temps, c'est N_h que nous ferons varier pour *affiner* les filtres possibles.

6.3 La résolution en pratique

Notre première démarche a été de vérifier que nous pouvions retrouver des résultats connus avant de nous lancer dans des cas nouveaux. Pour l'instant, seul le cas $K = 3$ à été traité dans la littérature (voir [KV95]), pour la cascade de Vetterli-Kovacevic et une seule famille de filtres à été exhibée (correspondant aux valeurs $N_l = 2$ et $N_h = 0$. Les figures 6.5 et 6.6 illustrent respectivement la représentation fréquentielle du filtre H_0 obtenu et le filtre itéré limite induit.

Nous avons entrepris la démarche évoquée à la fin de la partie précédente en utilisant les outils développés par J.C. Faugère pour le calcul de bases de Groebner ainsi que ceux décrits dans cette thèse.

Calcul d'une valeur limite pour N_l :
Notre première idée a été de vérifier que le problème n'admettait pas de solutions pour $N_l = 2$ et $N_h = 0$. La base de Groebner du système correspondant étant égale à [1], le système n'admettait donc pas de solutions complexes et a fortiori pas de solutions réelles.
Nous nous avons alors fixé $N_l = 2$ et fait varié N_h jusqu'à trouver un système zéro-dimensionnel ($N_h = 2$). En appliquant l'algorithme de Hermite, nous avons alors dénombré 64 racines réelles.

En appliquant l'algorithme RUR, nous avons pu déterminer, avec une précision de l'ordre de 10^{-20} l'ensemble des solutions du système et par conséquent, nous avons reconstitué 64 filtres possibles.

Sur des dessins de ces filtres, nous avons alors remarqué qu'en fait seuls deux filtres semblaient réellement distincts, modulo certaines symétries. Pour permettre la détection, de manière exacte, de telles symétries il aurait fallu déterminer de façon exacte (non numérique) les coefficients des polynômes les définissant, et par conséquent les coordonnées du système étudié. En remarquant que certaines de celles-ci semblaient pouvoir s'exprimer à l'aide de radicaux, une manière de les déterminer semblait consister à factoriser le premier polynôme de la RUR. Malheureusement, celui-ci est de degré 64 et présente des coefficients de tailles suffisamment importantes pour que les algorithmes de factorisation ne puissent effectuer l'opération.

Lorsqu'un idéal n'est pas en position générique, le calcul d'ensembles triangulaires (voir [Laz92], [Fau94]) permet, dans certains cas , de décomposer un système

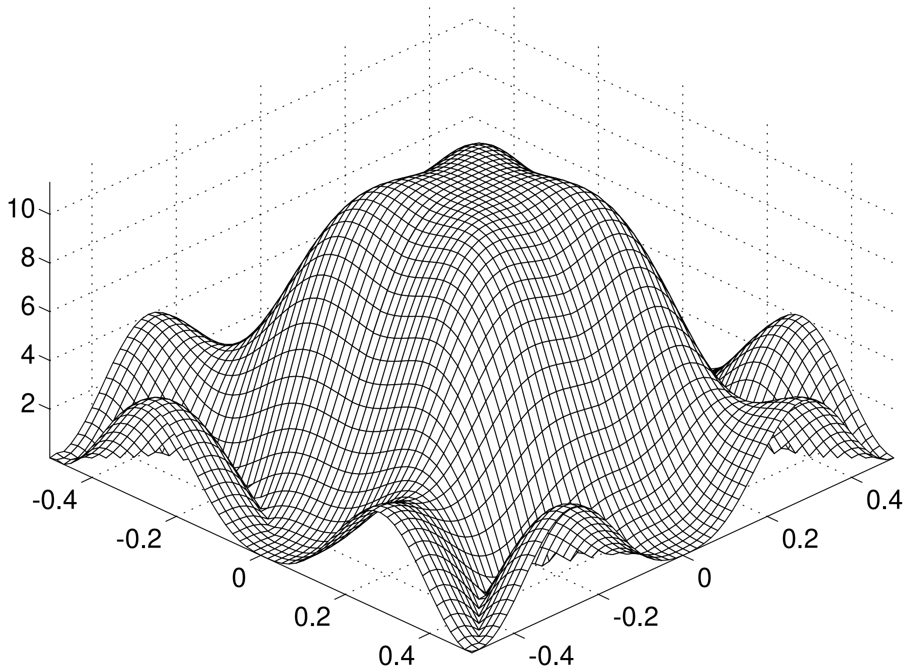

Figure 6.5: Filtre *passe-bas* connu pour $K = 3$

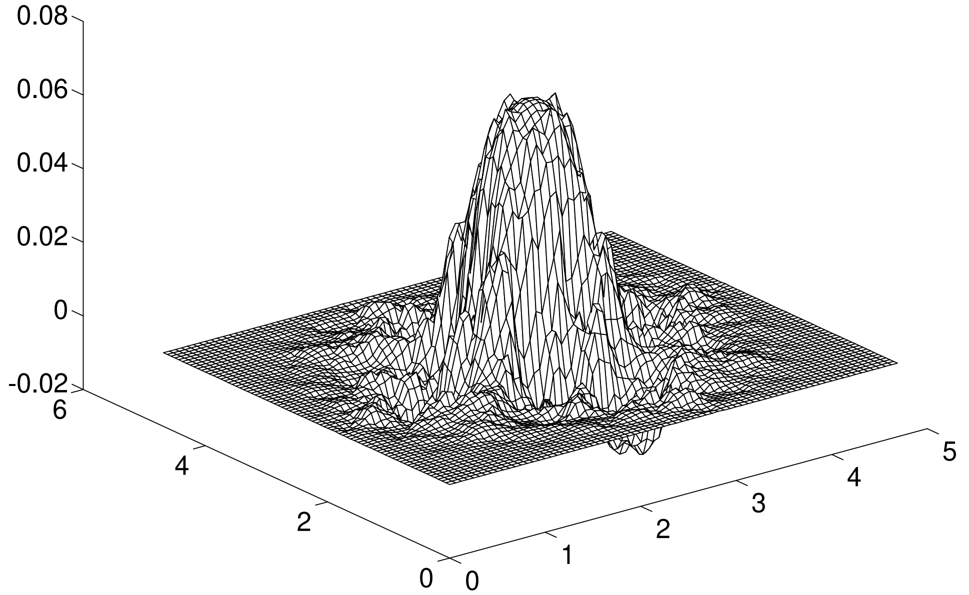

Figure 6.6: Fonction limite (filtre itéré) connu pour $K = 3$

en une union de systèmes triangulaires. En remarquant que l'élément séparant déterminé pour le calcul de la RUR n'est pas trivial, ce qui sous-entend que le système n'est pas en position générique, nous avons alors appliqué cette méthode et obtenu deux systèmes triangulaires ayant chacun 32 solutions complexes. Sur chacune de ces composantes, nous avons alors appliqué l'algorithme RUR et cette fois, nous avons pu factoriser chaque premier polynôme en facteurs de degrés 4, permettant ainsi la détermination explicite des coordonnées des 64 solutions possibles.
Par exemple, l'une des solutions obtenues est la suivante :

$$[\frac{\sqrt{2}}{8}+\frac{\sqrt{15}\sqrt{2}}{8},\frac{7}{8},1/4,-\frac{\sqrt{2}}{8}+\frac{\sqrt{15}\sqrt{2}}{8},\frac{\sqrt{15}}{8},\frac{\sqrt{15}}{4},-\frac{\sqrt{2}}{2},-1,-1,\frac{\sqrt{2}}{2},0,0]$$

En reconstituant alors les polynômes H_0 correspondants, nous obtenons alors 2 solutions distinctes (modulo certaines symétries), dont la solution déjà connue (les caractéristiques du nouveau filtre sont données par la figure 6.7).

Ce nouvel exemple de filtre n'apporte pas de réel progrès en pratique puisque la figure 6.7 tend à montrer que sa représentation fréquentielle est moins *plate* que celle de l'exemple déjà connu et surtout que la fonction limite du banc de filtres itérés est moins régulière. Il nous a permis toutefois de vérifier que nos outils sont applicables à ce genre de problèmes puisque, les temps de calcul, pour la RUR, n'excédant pas quelques secondes (base de Groebner comprise) par la boite à outils *RealSolving* et le logiciel Gb.

Aucun exemple n'était connu pour pour la cascade de Vetterli lorsque $K > 3$, y compris dans la littérature spécialisée. En ce qui nous concerne, nous avons obtenu pour le moment les valeurs maximales de N_l pour $K = 4, 5, 6, 7, 8$, mais aucune des valeurs de N_h testées n'a permis de retrouver un système zéro-dimensionnel à l'image du cas $K = 3$. Il faut souligner au passage que le fichier contenant les polynômes pour le cas $K = 8$, $N_l = 6$, $N_h = 6$ occupe plusieurs dizaines de Méga-octets sur disque.

Ceci nous à conduit à rajouter, au cas par cas, quelques conditions supplémentaires, comme l'annulation du filtre en certains points, pour exhiber des solutions particulières. A l'heure actuelle, les résultats sont convaincants, du point de vue traitement de signal, puisqu'un nombre non négligeable de filtres nouveaux et performants ont été produits.

En particulier nous sommes en mesure de travailler sur des systèmes correspondant à $K = 5$ (voir figure 6.8), le meilleur filtre déterminé à ce jour, compte tenu des critères considérés, ayant toutefois été obtenu pour $k = 7$, $N_l = 4$ et $N_h = 4$. Ses caractéristiques sont données par la figure 6.9.

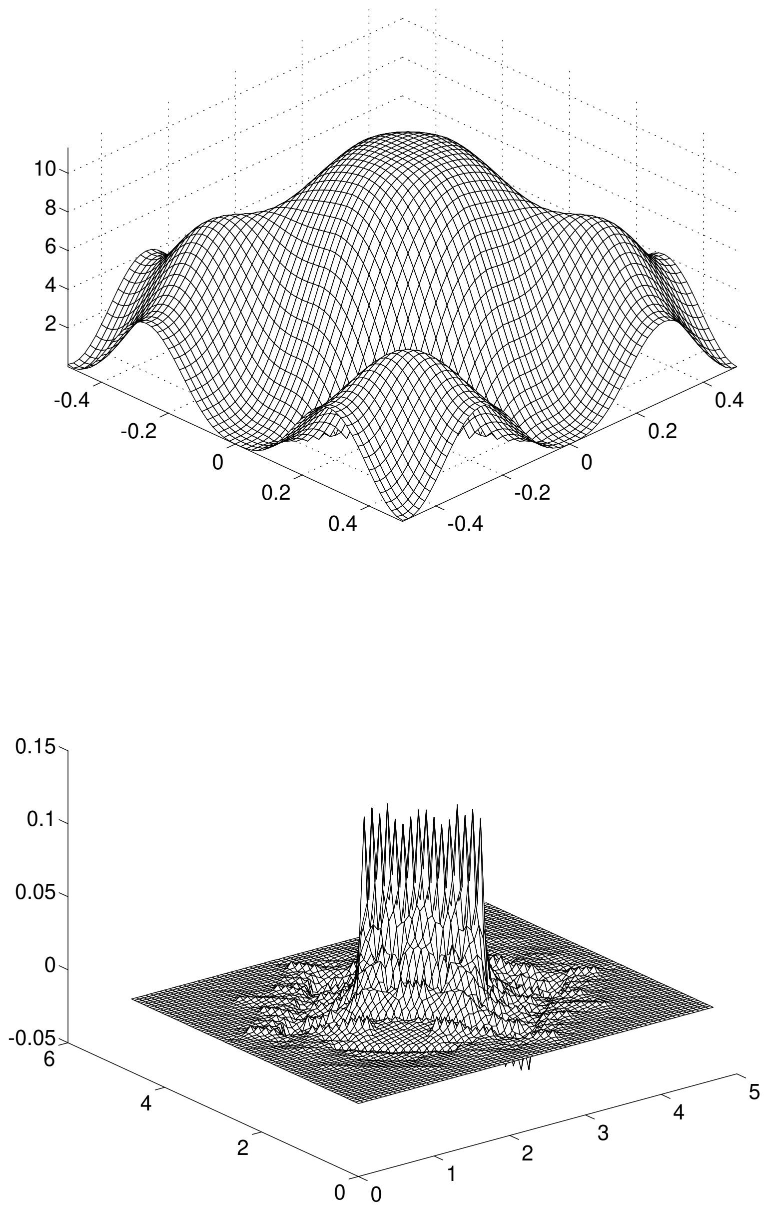

Figure 6.7: Nouveau filtre *passe-bas* pour $K = 3$

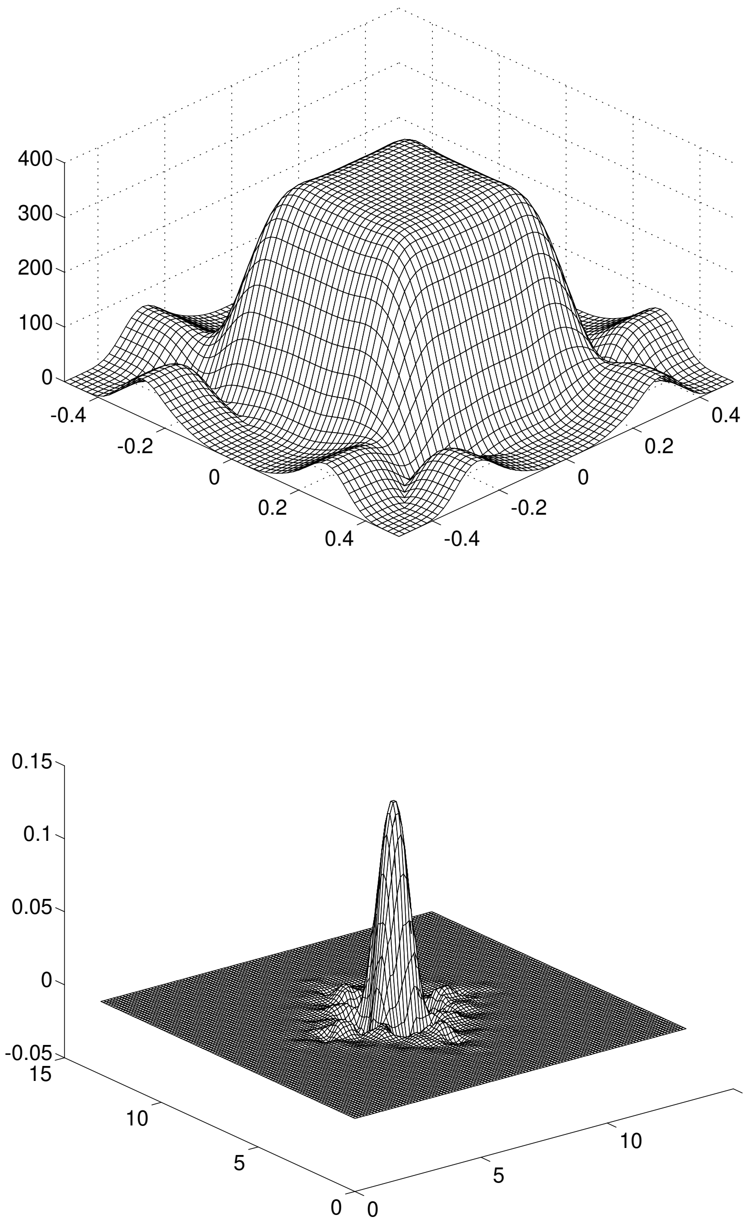

Figure 6.8: Filtre et fonction limite pour $K = 8$, $N_l = 5$ et $N_h = 5$

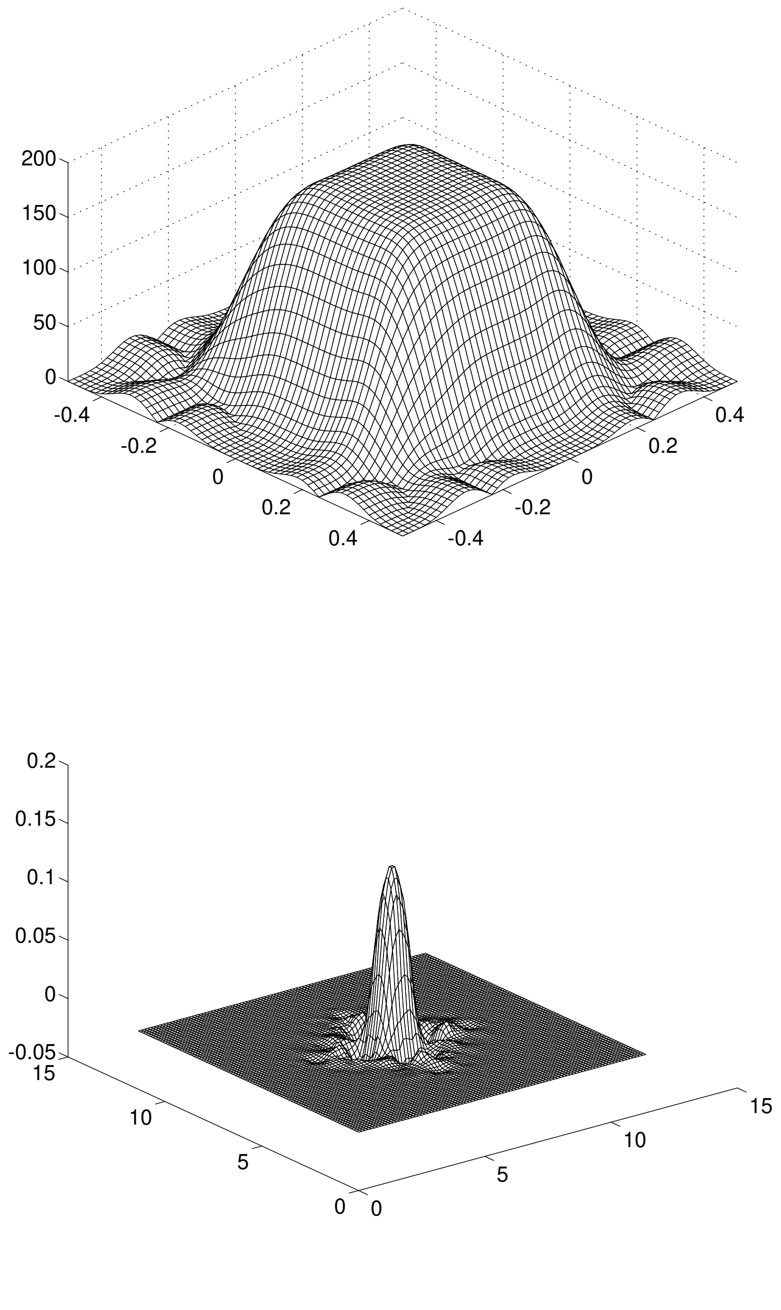

Figure 6.9: Filtre et fonction limite pour $K = 7$, $N_l = 4$ et $N_h = 4$

Chapitre 7

Le problème d'interpolation de Birkhoff

Résumé

Dans ce chapitre, nous montrons comment le calcul formel peut aider à l'étude de méthodes numériques.

Étant donnée une fonction $f : \mathbb{R} \to \mathbb{R}$, le problème d'interpolation de Birkhoff consiste à décider si la connaissance d'un nombre fini de valeurs de f où de certaines de ses dérivées en un ensemble de points fixé, suffit à déterminer un polynôme p interpolant f.

Une première partie sera consacrée à l'établissement d'un critère simple (proposé dans [G94]), permettant de ramener le problème à une étude d'hypersurfaces. Plus précisément, nous montrons que ce problème se résume à déterminer si une hypersurface admet ou non des points réels dont les coordonnées sont non nulles.

La deuxième partie décrira un certain nombre de techniques permettant, dans des cas favorables, de se ramener à l'étude de systèmes zéro-dimensionnels. Le principe de base étant de compter le nombre de points critiques réels d'une application particulière (fonction distance).

Enfin, une partie expérimentale montrera que l'on peut résoudre, en combinant ces différentes techniques, le problème de Birkhoff pour certaines classes d'exemples qui n'avaient pas pu être traitées jusqu'à ce jour.

7.1 Introduction

L'interpolation d'une fonction inconnue $f\colon \mathbb{R} \to \mathbb{R}$ par un polynôme, en connaissant les valeurs de f et de certaines de ses dérivées en quelques points de $\mathbb{R}$, est un problème fondamental en analyse numérique et en théorie de l'approximation.

Les deux problèmes d'interpolation les plus connus sont l'interpolation de Lagrange et de Hermite.

Dans le premier cas, si on suppose connues les valeurs de f en n points $x_0 < \ldots < x_n$ de $\mathbb{R}$, la formule de Lagrange montre qu'il existe un polynôme de degré inférieur ou égal à n interpolant f aux points x_i.

Le problème d'interpolation de Hermite généralise celui de Lagrange en ce sens que des informations concernant les dérivées de f peuvent être inclues. Si $x_1 < \ldots < x_n$ sont des points de $\mathbb{R}$ fixés, et $\nu_1, \ldots, \nu_n$ des entiers positifs, le problème d'interpolation de Hermite est résolu lorsque l'on peut trouver un polynôme P (unique) de degré inférieur ou égal à $N = \nu_1 + \ldots + \nu_n - 1$ tel que pour tout $k \in \{1, \ldots, n\}$ et $j \in \{0, \ldots, \nu_{k-1}\}$ on ait :

$$f^{(j)}(x_k) = P^{(j)}(x_k).$$

Le problème d'interpolation de Birkhoff est quant à lui encore plus général en ce sens qu'aucune contrainte a priori n'est imposée, tant sur le nombre ou l'ordre des dérivées que sur le nombre de points considérés.

Les conditions imposées pour l'interpolation peuvent être décrites sous forme matricielle par le biais de *matrices d'incidence*

Definition 7.1.1 *Soient n et r deux entiers tels que $n \geq 1$ et $r \geq 0$. La matrice*

$$\mathcal{E} = \begin{pmatrix} e_{1,0} & \ldots & e_{1,r} \\ \vdots & & \vdots \\ e_{n,0} & \ldots & e_{n,r} \end{pmatrix}$$

est une matrice d'incidence si $e_{i,j} \in \{0,1\}$ pour tout i et tout j.
Pour une matrice d'incidence $\mathcal{E}$ donnée, $|\mathcal{E}|$ sera égal au nombre de 1 dans $\mathcal{E}$:

$$|\mathcal{E}| = \sum_{i,j} e_{i,j}$$

Lorsque $|\mathcal{E}|$ est égal au nombre de colonnes de $\mathcal{E}$, la matrice d'incidence $\mathcal{E}$ sera dite normale.

Si $(\mathcal{X}, \mathcal{E})$ est un ensemble de réels tels que $x_1 < \ldots < x_n$, $\mathcal{E}$ une matrice d'incidence, et $\mathcal{F} = (f_{i,j})_{i=1\ldots n}^{j=0\ldots r}$ une matrice de nombres réels, nous dirons qu'un polynôme P, de degré inférieur ou égal à r, interpole $\mathcal{F}$ en $(\mathcal{X}, \mathcal{E})$ si, pour tout $i = 1 \ldots n$ et tout $j = 1 \ldots r$:

$$e_{i,j} = 1 \Rightarrow P^{(j)}(x_i) = f_{i,j}$$

Étant donnée une matrice d'incidence $\mathcal{E}$, le problème d'interpolation de Birkhoff consiste à décider si, pour tout ensemble de réels $\mathcal{X}$ tels que $x_1 < \ldots < x_n$ et tout matrice $\mathcal{F}$, il existe un unique polynôme P de degré inférieur ou égal à r interpolant $\mathcal{F}$ en $(\mathcal{X}, \mathcal{E})$.

Definition 7.1.2 *Une matrice d'incidence $\mathcal{E}$ à n lignes et $r+1$ colonnes est une $\mathcal{P}$-matrice si, pour tout ensemble de réels (nœuds) $\mathcal{X} = \{x_1, \ldots, x_n\}$ $(x_1 < \ldots < x_n)$, et toute matrice $\mathcal{F}$ de nombres réels à n lignes et $r+1$ colonnes, il existe un unique polynôme de degré au plus r interpolant $\mathcal{F}$ en $(\mathcal{X}, \mathcal{E})$.*

Par exemple, la $\mathcal{P}$-matrice correspondant au problème d'interpolation de Lagrange est :

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

Nous nous proposons de **déterminer l'ensemble $\mathcal{P}$-matrices , pour n et r fixés**.

7.2 Caractérisation des $\mathcal{P}$-matrices

L'objet de cette partie est de donner un critère algébrique pour déterminer si une matrice d'incidence donnée est une $\mathcal{P}$-matrice ou non.

A tout matrice d'incidence $\mathcal{E}$ à n lignes et $r+1$ colonnes on associe, pour $\mathcal{F}$ et $\mathcal{X}$ fixés une matrice $\mathcal{M}^0_{\mathcal{E}}(\mathcal{X}, \mathcal{E})$ à $r+1$ colonnes et $|\mathcal{E}|$, lignes de la façon suivante :

- En ordonnant l'ensemble $\{e_{i,j}\}_{i=0,\ldots,n-1}^{j=0,\ldots,r}$ selon l'ordre lexicographique croissant sur (i,j), on construit une liste ordonnée $L_{\mathcal{E}} = \{e_{i_k,j_k}\}_{k=0,\ldots,|\mathcal{E}|}$, contenant tous les termes non nuls de $\mathcal{E}$ (on peut remarquer que $\sharp L_{\mathcal{E}} = |\mathcal{E}|$).
- Si $P = \sum_{i=0}^{r} a_i T^i$ interpole $\mathcal{F}$ en $(\mathcal{X}, \mathcal{E})$, par construction de $L_{\mathcal{E}}$ et, pour tout indice k,
$$P^{(j_k)}(x_{i_k}) = f_{i_k,j_k},$$
ce qui peut se traduire par :
$$f_{i_k,j_k} = \sum_{j=j_k}^{r} a_j k! \binom{k+j}{k} x^j_{i_k}.$$
- Enfin, on pose, pour tout k, $\mathcal{M}^0_{\mathcal{E}} = (m^0_{k,j}(\mathcal{X}, \mathcal{E}))_{j=0,\ldots,r}^{k=0,\ldots,|\mathcal{E}|}$ avec $m^0_{k,j} = k!\binom{k+j}{k} x^j_{i_k}$ pour $i \geq i_k$ et $m^0_{k,j} = 0$ pour $i < i_k$.

Déterminer l'existence d'un polynôme P interpolant $\mathcal{F}$ en $(\mathcal{X}, \mathcal{E})$ revient alors exactement à décider si le système linéaire :

$$\mathcal{M}_{\mathcal{E}}^0 \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_r \end{pmatrix} = \begin{pmatrix} f_{i_1,j_1} \\ \vdots \\ f_{i_{|\mathcal{E}|},j_{|\mathcal{E}|}} \end{pmatrix}$$

admet une unique solution.

On peut généraliser ceci à la caractérisation des $\mathcal{P}$-matrices : en supposant que $x_1, \ldots, x_n$ sont cette fois des variables, on définit de la même façon $\mathcal{M}_{\mathcal{E}}$, qui est alors une matrice dont les coefficients sont cette fois des monômes de $\mathbb{Z}[x_1, \ldots, x_n]$. Décider si $\mathcal{M}_{\mathcal{E}}$ est une $\mathcal{P}$-matrice revient alors à décider si le système linéaire :

$$\mathcal{M}_{\mathcal{E}} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_r \end{pmatrix} = \begin{pmatrix} f_{i_1,j_1} \\ \vdots \\ f_{i_{|\mathcal{E}|},j_{|\mathcal{E}|}} \end{pmatrix}$$

admet une unique solution pour toute matrice $\mathcal{F}$.

La proposition suivante permet de réduire le nombre de matrices d'incidence à étudier pour un nombre de nœuds (n) et un degré (r) fixés, et donne un critère pour déterminer si une matrice d'incidence est ou non une $\mathcal{P}$-matrice :

Proposition 7.2.1 *Soit $\mathcal{E}$ une matrice d'incidence à n lignes et $r+1$ colonnes.*

- *Pour que $\mathcal{E}$ soit une $\mathcal{P}$-matrice, il faut que $\mathcal{E}$ soit normale ($|\mathcal{E}| = r+1$).*
- *Une matrice d'incidence $\mathcal{E}$ est une $\mathcal{P}$-matrice si et seulement si :*
  - *elle est normale*
  - *le déterminant $\mathcal{D}_{\mathcal{E}}$ de $\mathcal{M}_{\mathcal{E}}$ ne s'annule pas sur les ensembles de $\mathbb{R}^n$ de la forme $\{x_1, \ldots, x_n \ , \ x_1 < \ldots < x_n\}$.*

Preuve : Pour que $\mathcal{E}$ soit une $\mathcal{P}$- matrice, il faut que le système

$$\mathcal{M}_{\mathcal{E}} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_r \end{pmatrix} = \begin{pmatrix} f_{i_1,j_1} \\ \vdots \\ f_{i_{|\mathcal{E}|},j_{|\mathcal{E}|}} \end{pmatrix}$$

admette une unique solution pour toute matrice $\mathcal{F}$ et tout ensemble $\mathcal{X} = \{x_1, \ldots, x_n\}$ avec $x_1 < \ldots < x_n$. Par conséquent , $\mathcal{M}_{\mathcal{E}}$ doit être carrée, autrement dit $|\mathcal{E}| = r+1$. De plus, $\mathcal{E}$ sera une $\mathcal{P}$-matrice si et seulement si $\mathcal{D}_{\mathcal{E}}$ ne s'annule pas sur les ensembles $\mathcal{X} = \{x_1, \ldots, x_n\} \in \mathbb{R}^n$, avec $x_1 < \ldots < x_n$. $\square$

7.3 Simplification du problème

D'après ce que nous venons de voir, nous pouvons déterminer si une matrice d'incidence $\mathcal{E}$ est (ou non) une $\mathcal{P}$-matrice en décidant si l'ensemble semi-algébrique de $\mathbb{R}^n$

$$D_{\mathcal{E}}(x_1, \ldots, x_n) \ et \ x_1 < \ldots < x_n$$

est vide (ou pas).

Il s'agit là d'un problème de décision pour la théorie existentielle des réels, délicat à résoudre en pratique, que nous allons simplifier de deux façons :

- En diminuant le degré de l'hypersurface $D_{\mathcal{E}}$
- En modifiant l'expression de $D_{\mathcal{E}}$ de sorte à simplifier l'expression

$$D_{\mathcal{E}}(x_1, \ldots, x_n) \ et \ x_1 < \ldots < x_n$$

Par construction, la matrice $\mathcal{M}_{\mathcal{E}}$ peut s'exprimer :

$$\mathcal{M}_{\mathcal{E}} = \begin{pmatrix} B_0 \\ B_1 \\ \vdots \\ B_r \end{pmatrix}$$

avec :

$$B_k = \left.\overbrace{\begin{pmatrix} 0 & \ldots & 0 & k!\binom{k}{k} & k!\binom{k+1}{k} x_{i(1,k)} & \ldots & k!\binom{r}{k} x_{i(1,k)}^{r-k} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \ldots & 0 & k!\binom{k}{k} & k!\binom{k+1}{k} x_{i(e_k,k)} & \ldots & k!\binom{r}{k} x_{i(e_k,k)}^{r-k} \end{pmatrix}}^{r+1}\right\} e_k \quad (7.1)$$

en notant e_k le nombre de 1 dans la k^e colonne de $\mathcal{E}$ et $i(u,k)$ l'élément de $\{1, \ldots, n\}$ tel que le u^e 1 de la k^e colonne de $\mathcal{E}$ ait pour indice $(i(u,k), k)$ de $\mathcal{E}$. Ainsi, en notant $L_{i,j}$ $(1 \leq i < j \leq n)$ le nombre d'indices k (colonnes de $\mathcal{E}$) tels que

$$e_{i,k} = 1 \ et \ e_{j,k} = 1$$

on peut remarquer que $(x_i - x_j)^{L_{i,j}}$ divise $\mathcal{D}_{\mathcal{E}}$.

Definition 7.3.1 *Soit $\mathcal{E}$ une matrice d'incidence normale. On définit le $\mathcal{P}$-indicateur de $\mathcal{E}$ ($\in \mathbb{Z}[x_1, \ldots, x_n]$) par :*

$$\widetilde{\mathcal{D}}_{\mathcal{E}} = \frac{\mathcal{D}_{\mathcal{E}}}{\prod_{1 \leq i < j \leq n} (x_j - x_i)^{L_{i,j}}}$$

Ainsi, décider si une matrice d'incidence $\mathcal{E}$ est une $\mathcal{P}$-matrice ou non, revient à déterminer si il existe ou non $\{x_1, \ldots, x_n\} \subseteq \mathbb{R}$ tels que:

$$x_1 < \ldots < x_n \qquad \text{et} \qquad \mathcal{D}_{\mathcal{E}}(x_1, \ldots, x_n) = 0$$

Nous nous proposons maintenant de réduire ce problème à un problème plus algébrique en éliminant les inégalités dans cette expression.

Proposition 7.3.1 *Soit $\mathcal{E}$ une matrice d'incidence normale à n lignes et $r+1$ colonnes. Si $t_1, \ldots, t_{n-1}$ sont de nouvelles variables (indépendantes de $x_1, \ldots, x_n$), le polynôme:*

$$\mathcal{H}_{\mathcal{E}}(t_1, \ldots, t_{n-1}, x_1) = \widetilde{\mathcal{D}}_{\mathcal{E}}(x_1, x_1 + t_1^2, x_1 + t_1^2 + t_2^2, \ldots, x_1 + \sum_{i=1}^{n-1} t_i^2)$$

appartient à $\mathbb{Z}[t_1, \ldots, t_{n-1}]$.

Preuve : Si le degré de $\mathcal{H}_{\mathcal{E}}$ en x_1 est $d > 0$, nous pouvons exprimer $\mathcal{H}_{\mathcal{E}}$ par :

$$\mathcal{H}_{\mathcal{E}}(t_1, \ldots, t_{n-1}, x_1) = a_d(t_1, \ldots, t_{n-1})x_1^d + \ldots + a_1(t_1, \ldots, t_{n-1})x_1 + a_0(t_1, \ldots, t_{n-1})$$

où les $a_i(t_1, \ldots, t_{n-1})$ sont des polynômes à coefficients entiers et où a_d n'est pas identiquement nul.
Dans ce cas, il existe $(\beta_1, \ldots, \beta_{n-1}) \in \mathbb{R}^n$ tel que $a_d(\beta_1, \ldots, \beta_{n-1}) \neq 0$.
Si nous supposons que d est strictement positif, il existe $\alpha \in \mathbb{C}$ tel que

$$\mathcal{H}_{\mathcal{E}}(\beta_1, \ldots, \beta_{n-1}, \alpha) = 0,$$

et $\gamma \in \mathbb{C}$ tel que

$$\mathcal{H}_{\mathcal{E}}(\beta_1, \ldots, \beta_{n-1}, \alpha + \gamma) \neq 0.$$

En adaptant les énoncés au cas complexe, il existe alors un unique polynôme $Q \in \mathbb{C}[T]$ tel que Q interpole $\mathcal{F}$ en $(\mathcal{X}, \mathcal{E})$ pour

$$\mathcal{X} = \{\alpha + \gamma, \ldots, \alpha + \gamma + \sum_{i=1}^{n-1} \beta_i^2\}.$$

Ceci montre, en particulier, que $P(T) = Q(T + \gamma)$ est l'unique polynôme interpolant $\mathcal{F}$ en $(\mathcal{X}', \mathcal{E})$ pour

$$\mathcal{X}' = \{\alpha, \ldots, \alpha + \sum_{i=1}^{n-1} \beta_i^2\}$$

et que $\mathcal{H}_{\mathcal{E}}(\beta_1, \ldots, \beta_{n-1}, \alpha) \neq 0$, ce qui contredit les hypothèses. Ainsi, $d = 0$ et

$$\mathcal{H}_{\mathcal{E}} = a_0(t_1, \ldots, t_{n-1})$$

□

Avant de revenir au critère que nous voulons déterminer, il est encore une propriété de $\mathcal{H}_{\mathcal{E}}$ qui permettra de simplifier le problème :

Proposition 7.3.2 *Soit $\mathcal{E}$ une matrice d'incidence normale à n lignes et $r+1$ colonnes. Le polynôme $\mathcal{H}_{\mathcal{E}}$ est homogène et de degré borné par $2nr$.*

Preuve : Soit λ une nouvelle variable, indépendante de $t_1, \dots, t_n$. Posons :

$$\mathcal{H}_{\mathcal{E}}(\lambda t_1, \dots, \lambda t_{n-1}) =$$

$$= \widetilde{\mathcal{D}}_{\mathcal{E}}(0, \lambda^2 t_1^2, \lambda^2(t_1^2 + t_2^2), \dots, \lambda^2 \sum_{i=1}^{n-1} t_i^2) = \sum_{j=0}^{r} a_j(t_1, \dots, t_{n-1})\lambda^j = R(t_1, \dots, t_n, \lambda)$$

Si $\mathcal{H}_{\mathcal{E}}$ est constant, le résultat est démontré. Dans le cas contraire, nous allons montrer qu'au plus un des a_j n'est pas identiquement nul. Supposons que a_j et a_i ne soient pas identiquement nuls. Dans ce cas, il existe $\underline{\alpha} \in \mathbb{R}^{n-1}$ tel que

$$\mathcal{H}_{\mathcal{E}}(\underline{\alpha}) \cdot a_i(\underline{\alpha}) \cdot a_k(\underline{\alpha}) \neq 0$$

Le polynôme $R(\underline{\alpha}, \lambda)$ admet, dans ce cas, une racine non nulle (dans $\mathbb{C}$), que nous noterons λ_0.
Comme $\mathcal{H}_{\mathcal{E}}(\underline{\alpha}) \neq 0$, pour $\mathcal{F}$ fixée, il existe un unique polynôme P (à coefficients dans $\mathbb{C}$) interpolant $\mathcal{F}$ en $(\mathcal{X}_1, \mathcal{E})$, avec :

$$\mathcal{X}_1 = \{0, t_1^2, t_1^2 + t_2^2, \dots, \sum_{i=1}^{n-1} t_i^2\}$$

Posons

$$\mathcal{X}_2 = \{0, \lambda_0^2 t_1^2, \lambda_0^2(t_1^2 + t_2^2), \dots, \lambda_0^2 \sum_{i=1}^{n-1} t_i^2\}$$

Le polynôme $P(x/\lambda_0^2)$ est alors l'unique polynôme de $\mathbb{C}[x]$ interpolant $\mathcal{F}$ en $(\mathcal{X}_2, \mathcal{E})$. Ceci implique en particulier $R(\underline{\alpha}, \lambda_0) \neq 0$, ce qui est impossible.

Comme il existe au plus un a_i non identiquement nul, il existe un entier positif m tel que

$$\mathcal{H}_{\mathcal{E}}(\lambda t_1, \dots, \lambda t_n) = \lambda^m R(t_1, \dots, t_n, 1)$$

et, en posant $\lambda = 1$, on peut alors remarquer que $\mathcal{H}_{\mathcal{E}} = R(t_1, \dots, t_n, 1)$, ce qui montre que $\mathcal{H}_{\mathcal{E}}$ est homogène.

Par construction (voir relation (7.1)), le degré de $D_{\mathcal{E}}$ est clairement majoré par

$$\sum_{k=0}^{r-1} (r-k)e_k$$

Comme $\sum_{k=0}^{r-1} e_k = n$, cette dernière quantité est majorée par nr, et donc $\mathcal{H}_{\mathcal{E}}$ est de degré inférieur à $2nr$. $\square$

Corollaire 7.3.1 *Si $\mathcal{E}$ est une matrice d'incidence normale, $\mathcal{E}$ ne sera pas une $\mathcal{P}$-matrice si et seulement si $\mathcal{H}_{\mathcal{E}}(t_1,\ldots,t_{n-1}) = 0$ admet une solution réelle $(1, t_2, \ldots, t_{n-1})$ avec $t_2 \ldots t_{n-1} \neq 0$.*

Preuve : Montrons que $\mathcal{E}$ n'est pas une $\mathcal{P}$-matrice si et seulement si $\mathcal{H}_{\mathcal{E}}$ admet une solution réelle $(t_1, \ldots, t_{n-1})$ avec $t_1 \ldots t_{n-1} \neq 0$.

Si $\mathcal{E}$ n'est pas une $\mathcal{P}$-matrice, il existe $\mathcal{X} = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}$ tel que $x_1 < \ldots < x_n$ et $\widetilde{\mathcal{D}}_{\mathcal{E}}(x_1, \ldots, x_n) = 0$. En posant $t_i = (x_{i+1} - x_i)^{1/2}$, $(t_1, \ldots, t_{n-1})$ est une solution réelle de $\mathcal{H}_{\mathcal{E}} = 0$ avec $t_i \neq 0$. Inversement, si $(t_1, \ldots, t_{n-1})$ est une solution réelle de $\mathcal{H}_{\mathcal{E}} = 0$ sans coordonnée nulle, en posant $x_1 = 0$ et $x_{i+1} = x_i + t_i^2$, on peut voir que $\widetilde{\mathcal{D}}_{\mathcal{E}}(x_1, \ldots, x_n) = 0$ et donc que $\mathcal{E}$ n'est pas une $\mathcal{P}$-matrice.

Il suffit maintenant d'utiliser le fait que $\mathcal{H}_{\mathcal{E}}$ est homogène pour conclure. □

Le dernier corollaire nous donne alors un critère permettant de ramener notre problème (pour n et r fixés) à l'étude d'une hypersurface :

Corollaire 7.3.2 *Soit $\mathcal{E}$ est une matrice d'incidence normale, et T une variable indépendante de $t_2, \ldots, t_{n-1}$. La matrice $\mathcal{E}$ ne sera pas une $\mathcal{P}$-matrice si et seulement si $\mathcal{H}_{\mathcal{E}}^2 + (Tt_2 \ldots t_{n-1} - 1)^2 = 0$ admet une solution réelle.*

Preuve : L'hypersurface

$$\mathcal{H}_{\mathcal{E}}^2 + (Tt_2 \ldots t_{n-1} - 1)^2 = 0$$

admettra des points réels si et seulement si

$$Tt_2 \ldots t_{n-1} - 1 = 0$$

et

$$\mathcal{H}_{\mathcal{E}} = 0$$

admettent des zéros communs. Ceci est alors équivalent à dire que $\mathcal{H}_{\mathcal{E}} = 0$ admet une solution réelle $(1, t_2, \ldots, t_{n-1})$ avec $t_2 \ldots t_{n-1} \neq 0$. □

7.4 Stratégies de résolution

Nous décrivons, dans cette partie, comment utiliser les divers algorithmes présents dans cette thèse pour résoudre (au moins partiellement) le problème énoncé, pour n et r fixés. La plage d'exemples intéressants est faible puisque dans [G94], L. Gonzalez-Vega montre que l'utilisation d'outils simples (comme les suites de Sturm-Habicht) permettent de résoudre bon nombre de cas lorsque $n \leq 3$. Il a montré en particulier que le problème pouvait être résolu, en pratique, pour des valeurs de r allant jusqu'à $r = 5$. Par ailleurs, l'algorithme de la C.A.D (voir [Col75]) permet de résoudre certains cas où $n \geq 3$, pour de petites valeurs de r (disons $r \leq 4$). Nous nous attacherons donc à étudier le cas $r = 5$, $n = 4$, car le nombre important de matrices d'incidence normales possibles pour n et r fixés (égal à $\binom{n(r+1)}{r+1}$) ne nous permet pas ne serait-ce que de générer les hypersurfaces à étudier.

Décider si une hypersurface admet des points réels ou non est pour l'heure un problème ouvert, du point de vue pratique. Nous ne prétendons pas dans cette partie fournir un algorithme universel, mais plutôt des solutions acceptables dans des situations géométriques dites génériques, en espérant pouvoir traiter un maximum de cas pour le problème qui nous intéresse.

Le but est d'éviter un traitement récursif (éliminations successives de variables) à l'image de celui réalisé par la C.A.D. pour plusieurs raisons :

- Éviter la croissance des coefficients et des degrés (résultants)
- Éviter le calcul d'un nombre important d'informations redondantes ou inutiles pour ce problème.

Plusieurs stratégies sont suggérées dans [HRS93] faisant intervenir :

- La projection selon un axe de coordonnée dans la cas d'une hypersurface lisse et bornée.
- La fonction distance à l'origine dans le cas d'un hypersurface lisse.

Nous ne développerons ici que la seconde solution, la condition *bornée* n'étant pas vérifiable facilement, la deuxième hypothèse étant quant à elle vérifiée dans les cas génériques.

Proposition 7.4.1 *Soient f un polynôme de $Q[X_1, \ldots X_n]$ et $o \in \mathbb{R}^n$ tel que $f(o) \neq 0$. On note :*

$$V_{\mathbf{R}}(f) = \{x \in \mathbb{R}^n \mid f(x) = 0\}$$

$$\mathcal{C}^d_{\mathbf{R}}(f) = \{x \in V_{\mathbf{R}}(f) \mid \overrightarrow{ox} // \overrightarrow{grad}(f)\}$$

L'ensemble $V_{\mathbf{R}}(f)$ est vide si et seulement si $\mathcal{C}^d_{\mathbf{R}}(f)$ est vide.

Preuve : Comme $\mathcal{C}^d_{\mathbf{R}}(f) \subset V_{\mathbf{R}}(f)$, Il est clair que si $V_{\mathbf{R}}(f)$ est vide, $\mathcal{C}^d_{\mathbf{R}}(f)$ l'est aussi.

Supposons maintenant que $V_{\mathbf{R}}(f)$ soit non vide, et notons r la distance de o à $V_{\mathbf{R}}(f)$. Dans ce cas, il existe un réel R tel que l'intersection de la boule fermée $\bar{B}^n(o, R)$ de centre o et de rayon R avec $V_{\mathbf{R}}(f)$ soit non vide. Considérons alors la fonction *distance* à o :

$$\begin{array}{rccc} d: & \mathbb{R}^n & \longrightarrow & \mathbb{R} \\ & x & \longmapsto & ||\overrightarrow{ox}|| \end{array}$$

Comme d est continue et comme $\bar{B}^n(o,R) \cap V_{\mathbb{R}}(f)$ est compact, on peut trouver un $x_0 \in V_{\mathbb{R}}(f)$ tel que

$$d(x_0) = inf(d(x)\ ,\ x \in \bar{B}^n(o,R) \cap V_{\mathbb{R}}(f).$$

Le point x_0 sera à distance minimale de o et par conséquent, $x_0 \in \mathcal{C}^d_{\mathbb{R}}(f)$. □

Notons :

$$V(f) = \{x \in \mathbb{C}^n \mid f(x) = 0\}$$

et

$$\mathcal{C}_d(f) = \{x \in V(f) \mid \overrightarrow{ox}//\overrightarrow{grad}(f)\}.$$

L'ensemble $\mathcal{C}_d(f)$ se décrit facilement comme étant l'ensemble des zéros d'un idéal $\mathcal{I}_d \subset \mathbb{Q}[X_1,\ldots,X_n]$, en exprimant le produit vectoriel de $\overrightarrow{oX}$ par $\overrightarrow{grad}(f)$ et en annulant toutes ses coordonnées. La proposition 7.4.1 montre alors que le problème de décision $V_{\mathbb{R}}(f) = \emptyset$ est exactement équivalent à $V(\mathcal{I}_d) \cap \mathbb{R}^n = \emptyset$.

L'important est de remarquer que , sauf cas *dégénéré*, l'idéal $\mathcal{I}_d$ est alors zéro-dimensionnel. En effet, sauf si :

- Il existe une infinité de points de $V(f)$ équidistants de o
- L'ensemble $V(f)$ admet une infinité de points singuliers (ces points vérifient tous : $\overrightarrow{ox}//\overrightarrow{grad}(f)$),

l'ensemble $\mathcal{C}_d(f)$ est fini.

En pratique, en choisissant un point o de coordonnées non triviales, seul le cas des hypersurfaces admettant une infinité de points singuliers est préocupant. Ce cas n'est pas courant mais mérite une étude spéciale.

Beaucoup de stratégies basées sur des déformations existent dans la litérature, mais elles nécessitent, en pratique, l'utilisation d'une arithmétique complexe pour le traitement des infinitésimaux. Nous proposons ici une solution évitant cet avatar. Rappelons le résultat de base sur lequel est basé notre méthode :

Proposition 7.4.2 *Soit f un polynôme de $\mathbb{Q}[X_1,\ldots,X_n]$ et ϵ un infinitésimal (strictement) positif. Alors si l'ensemble :*

$$\{x \in \mathbb{R}[\epsilon]^n\ ,\ f-\epsilon = 0\} \cup \{x \in \mathbb{R}[\epsilon]^n\ ,\ f+\epsilon = 0\}$$

est vide, $V_{\mathbb{R}}(f)$ l'est aussi.

L'idée est de remplacer ϵ *un infinitésimal ...* par ϵ *suffisement petit ...* dans l'énoncé précédent, et de fournir un algorithme permettant de calculer une borne pour ϵ.

Soit $f \in \mathbb{Q}[X_1, \ldots, X_n]$. On considère l'ensemble $\mathcal{G}_{\mathbf{R}} = \{x \in \mathbb{R}^n \mid \overrightarrow{grad}(f) = 0\}$. Cet ensemble est un fermé de Zariski et admet par conséquent un nombre fini de composantes connexes. Si x et y sont deux points d'une telle composante, on peut toujours trouver un chemin, dans $\mathcal{G}_{\mathbf{R}}$, différentiable par morceaux reliant x à y. La fonction polynomiale f est alors constante sur ce chemin, puisque $\overrightarrow{grad}(f) = 0$. La fonction f est constante sur chaque composante connexe de $\mathcal{G}_{\mathbf{R}}$ et n'admet donc qu'un nombre fini de valeurs critiques. En particulier, sauf pour un nombre fini de valeurs de t, $f_t = f - t$ est lisse. En prenant alors $t_0 > 0$ plus petit (en valeur absolue) que toute valeur critique non nulle de f ($t_0 > 0$ quelconque dans le cas où f_t est lisse pour toute valeur non nulle de t), nous obtenons alors les relations suivantes :

- Pour tout $0 < |t|$, $|t| < |t_0|$, f_t est lisse
- Si $0 < t < t_0$ et $0 < t' < t_0$ (resp. $0 > t > -t_0$ et $0 > t' > -t_0$), $V_{\mathbf{R}}(f_t) = \emptyset$ si et seulement si $V_{\mathbf{R}}(f_{t'}) = \emptyset$.

Notons $\mathcal{I}_T$ l'idéal de $\mathbb{R}[X_1, \ldots, X_n, T]$ engendré par :

$$\begin{cases} f - T \\ \frac{\partial}{\partial X_1} f \\ \vdots \\ \frac{\partial}{\partial X_n} f \end{cases}$$

Si f_t est lisse pour tout $t \in \mathbb{R}$, ce système n'admet pas de solutions et $t_0 > 0$ peut être choisi quelconque.
Dans le cas contraire, comme f_t est lisse, sauf pour un nombre fini de valeurs de t, l'idéal $\mathbb{Q}[T] \cap I_T$ admet un générateur P_T de degré strictement supérieur à 1, pouvant être obtenu en calculant une base de Groebner pour l'ordre lexicographique avec $T < X_i$, $i = 1 \ldots n$. En exprimant $P_T = T^k Q_T$, avec $Q_T(0) \neq 0$, deux cas sont à envisager :

- Si $Q_T \in \mathbb{Q}$, f_t est lisse pour tout $t \neq 0$. Dans ce cas on peut choisir $t_0 > 0$ quelconque.
- Si $Q_T \notin \mathbb{Q}$, en utilisant la borne de Landau-Mignotte (par exemple) pour majorer le module des racines de $Q_T(1/T)$, nous obtenons alors la valeur de t_0 recherchée.

7.5 La résolution en pratique

Evidement, les stratégies de résolution proposées n'ont réellement de sens que pour des problèmes où le nombre de variables est supéreur à 2. Dans le cas contraire,

L. Gonzalez-Vega propose dans [G94] une méthode efficace basée sur l'utilisation de suites de Sturm-Habicht couplées avec l'algorithmes *Inégalités simultanées* et résout au moyen de la C.A.D. le cas où $r = n = 4$, pour lequel les degrés des polynômes restent relativement faibles. Compte tenu du nombre important de matrices d'incidence possibles pour n et r fixés :

$$\binom{n(r+1)}{r+1}$$

Nous nous attachons à tenter le cas $n = 5, r = 4$, impliquant le traitement de 53130 hypersurfaces en 3 variables de degré pouvant atteindre 40.

Pour des raisons évidentes de croissance de degrés, nous ne tentons pas brutalement le problème consistant à regarder directement si les hypersurfaces :

$$\mathcal{H}_{\mathcal{E}}^2 + (TX_1 \ldots X_n - 1)^2$$

sont vides ou pas. Dans un premier temps, nous nous contentons d'étudier les ensembles de la forme

$$\mathcal{H}_{\mathcal{E}}(X_1, \ldots, X_n) = 0\ ,\ X_1 \neq 0\ \ldots\ X_n \neq 0,$$

selon la stratégie décrite par la figure 7.1.

L'étude de ces ensembles de points nous permet en particulier d'éliminer d'emblée tous les cas (nombreux) où $\mathcal{H}_{\mathcal{E}}$ s'exprime sous forme de sommes de carrés de monômes - une au moins des coordonnées doit être nulle pour qu'un point soit sur l'hypersurface - de sorte qu'il ne reste plus qu'environ 2000 cas à traiter. En pratique, un certain nombre de *coupes* ont été effectuées avant de lancer le processus (recherche de solutions où toutes les coordonnées sont égales).

Notons enfin qu'il est très simple, à partir d'une représentation univariée rationnelle , de décider si il existe des solutions sans aucune coordonnée nulle pour un système zéro-dimensionnel donné. En effet si $f, g, g_1, \ldots, g_n$ est une représentation univariée rationnelle , il nous suffit de vérifier qu'il existe une racine réelle α de f pour laquelle les expressions $g_1(\alpha), \ldots, g_n(\alpha)$ sont non nulles. Ceci pourrait s'effectuer par l'algorithme *Inégalités simultanées* mais une méthode moins coûteuse à été en pratique utilisée. En effet, en posant :

- $f_0 = f/gcd(f, f')$ (ceci fournit au passage une représentation univariée rationnelle du radical de l'idéal étudié)

- $f_i = f_{i-1}/gcd(f_{i-1}, g_i)$, $i = 1, \ldots, n$,

on peut voir que le nombre de zéros réels du système n'ayant aucune coordonnée nulle est exactement égal au nombre de zéros réels de f_n.

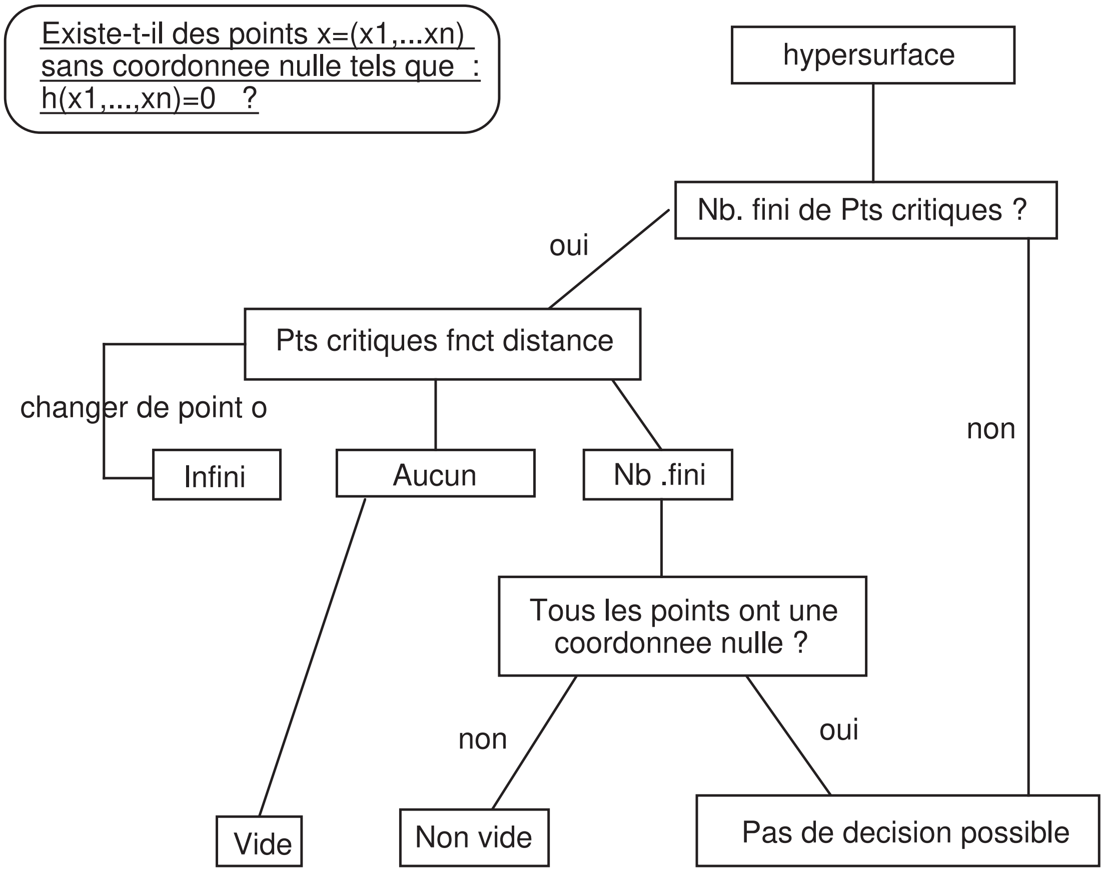

Figure 7.1: Problèmes de décision

Nous avons ensuite, sur les hypersurfaces pour lesquelles aucune décision n'a été possible, appliqué la stratégie de déformation décrite à la fin de la partie précédente, et on ne note à ce moment plus que 24 hypersurfaces pour lesquelles aucune décision n'à été possible.

Toutes ces hypersurfaces ont une infinité de points critiques et ne sont pas vides modulo une déformation infinitésimale. Pour être complets sur cette expérience, nous pouvons montrer qu'aucune d'entre elle n'est vide en fixant arbitrairement une où plusieurs de leurs variables, mais ceci ne figure plus dans le procédé de résolution automatique pour lequel des outils plus élaborés seront nécessaires.

Enfin, il est bon de préciser que, dans la plupart des cas, les temps de calculs n'ont jamais excédé quelques secondes. Ainsi, dans l'hypothèse où un algorithme général verrait le jour, cette stratégie pourrait être un test préliminaire efficace.

Perspectives - Conclusion

Dans cette thèse, nous avons établi ou optimisé un certain nombre de méthodes pour l'étude des zéros réels des systèmes polynomiaux. Couplée à des logiciels performants (Gb , librairie PoSSo), la boite à outils *RealSolving*, contenant une implantation optimisée des divers algorithmes, est devenu un outil de calcul efficace, en ce qui concerne les systèmes polynomiaux zéro-dimensionnels.

Comme l'atteste la figure 7.2, nous ne pouvions pas imaginer, en commençant ce travail, pouvoir un jour traiter des exemples aussi conséquents que ceux utilisés pour illustrer cette thèse. Il semble que certains de nos algorithmes puissent toutefois encore être optimisés. Ceci sera le sujet de la première partie de cette conclusion, dans laquelle nous présentons quelques méthodes, en cours de test à l'heure actuelle, mais pour lesquelles les stratégies à adopter ne sont pas encore claires.

La deuxième partie de cette conclusion montre comment nous envisageons, dans le futur, l'extension de notre panoplie d'algorithmes au traitement de problèmes plus complexes de géométrie réelle. C'est cette fois c'est d'un point de vue pratique que l'étude n'a pas été menée à terme, les algorithmes invoqués nécessitant le développement d'outils informatiques sophistiqués.

Techniques de calcul - Algorithmique

Durant tout le temps consacré à ce travail, une de nos préoccupations principales a été d'optimiser les techniques de calcul employées, de sorte à ne serait-ce que rendre possible le traitement automatique de certains problèmes.

Le domaine d'application de nos méthodes s'est agrandi de façon conséquente et les facteurs limitants de nos algorithmes ont eux aussi évolué. Il semble que les problèmes ne soient plus uniquement d'ordre arithmétique, puisque les temps de calcul observés ne dépassent que très rarement la demi-heure (temps C.P.U.), mais résident plutôt dans l'occupation mémoire des programmes.

Ce problème peut être (partiellement) résolu par un travail algorithmique. Trois voies ont été étudiées :

- utilisation de techniques modulaires dans le cas des systèmes à coefficients entiers,
- décomposition des problèmes,
- distribution des calculs / calcul parallèle.

Calcul modulaire

Le calcul modulaire est utilisé dans un certain nombre de nos algorithmes (prédiction de l'élément séparant dans l'algorithme RUR, calcul de pgcd). Nous nous attardons ici sur le calcul de la RUR dans le cas d'un idéal radical.

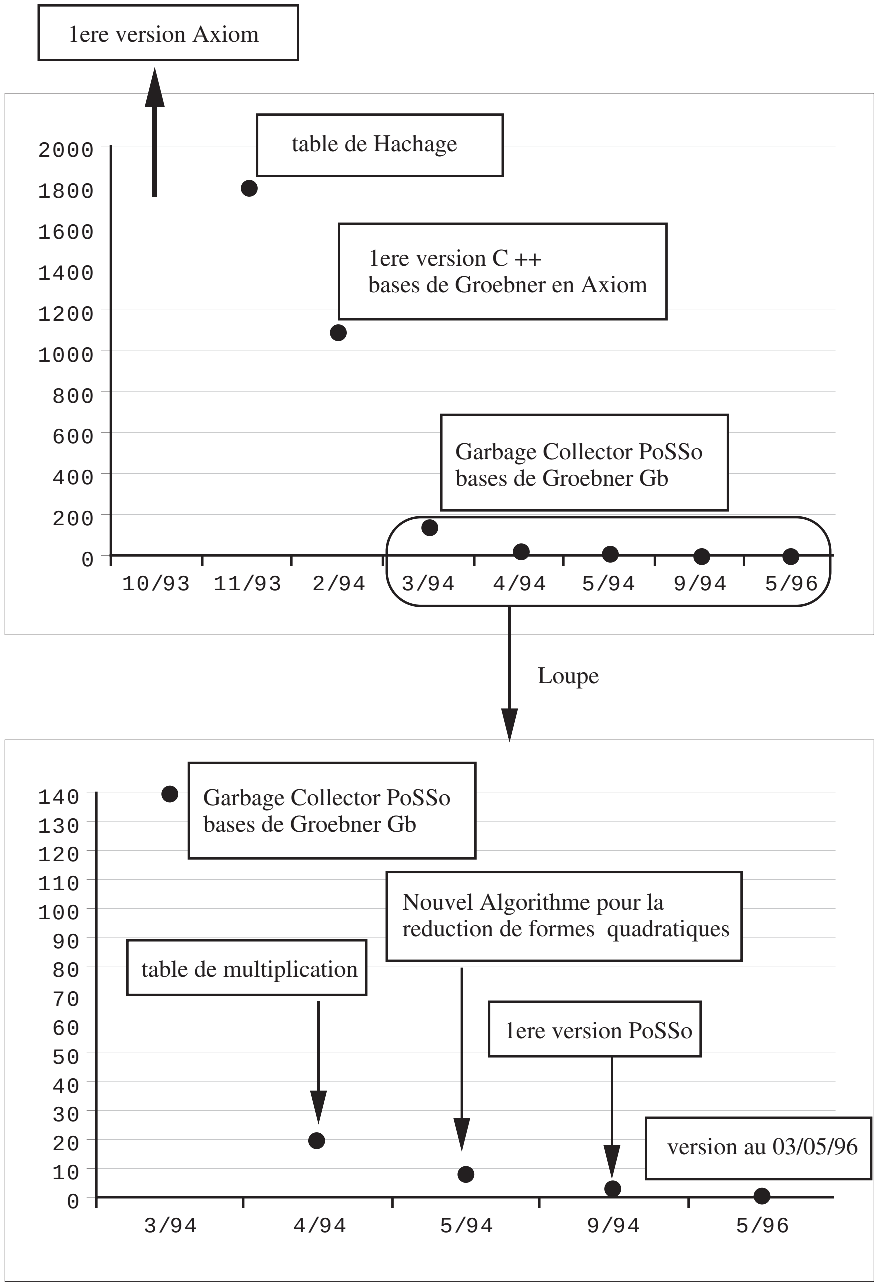

Figure 7.2: Évolution des temps de calcul (en sec.) sur un exemple simple

Nous avons pu remarquer, lors de l'étude pratique de l'algorithme RUR, que les tailles des entiers apparaissant dans le résultat pouvaient être inférieures à celle de l'entrée. Comme dans la plupart des algorithmes utilisant une arithmétique exacte, nous notons également, en pratique, une certaine croissance des coefficients, même si elle demeure efficacement contrôlée.

Dans le cas d'un idéal radical, il existe une manière simple de calculer la RUR pour un élément t séparant $V(\mathcal{I})$ donné.

En supposant que $\mathcal{I} \subset \mathbb{Q}[X_1, \ldots, X_n]$ est zéro-dimensionnel, nous notons $\mathcal{G}$ une base de Groebner de $\mathcal{I}$, $\mathcal{A} = \mathbb{Q}[X_1, \ldots, X_n]/\mathcal{I}$ le $\mathbb{Q}$-espace vectoriel de dimension D correspondant, et $\mathcal{B} = \{\omega_1 = 1, \ldots, \omega_D\}$ une base de monômes de $\mathcal{A}$.

- On construit la matrice $M(t)$ dont les colonnes sont constituées des puissances t^i , $i = 0, \ldots D-1$ dans $\mathcal{A}$ et exprimées dans la base $\mathcal{B}$

- La résolution du système linéaire $M(t)X = t^D$ nous donne alors les coefficients du premier polynôme de la RUR ($f(t,T)$). La connaissance de la matrice $M(t)$ nous permet d'évaluer rapidement tout polynôme en t.

- La résolution des systèmes $M(t) = \overrightarrow{f'(t,t)X_i}$ nous donne alors, pour tout $i = 1, \ldots, n$, les coefficients des divers polynômes $g_i(t,T)$.

Le problème de l'élément séparant est moins difficile à résoudre dans ce cas particulier, puisqu'il suffit que $M(t)$ soit inversible pour être assuré que t sépare $V(\mathcal{I})$. En utilisant la prédiction modulaire décrite dans cette thèse, nous pouvons estimer ce problème comme étant mineur.

Ainsi, une simple inversion de matrice (si $\mathcal{I}$ est radical, les puissances t^i , $i = 0, \ldots, D-1$ forment une base de $\mathcal{A}$, $M(t)$ est donc inversible), permet de calculer tous les polynômes de la représentation univariée rationnelle de $\mathcal{I}$. Malheureusement, cette inversion s'avère être en pratique extrêmement coûteuse. A titre indicatif elle représente plus de 90 pour cent du temps de calcul nécessaire à l'algorithme FGLM pour trouver une base lexicographique.

Il est toutefois intéressant de noter que ce calcul permet un gain considérable en termes d'occupation mémoire, puisqu'il ne nécessite pas le calcul de la totalité de la table de multiplication (simplement le bord).

Jean-Charles Faugère à eu le premier l'idée d'utiliser une arithmétique multi-modulaire pour le calcul de FGLM, et les premiers résultats expérimentaux sont excellents. Il utilise, entre autres, une représentation de la base lexicographique coïncidant, dans le cas d'un idéal radical, avec la représentation univariée rationnelle , ce qui lui permet en particulier de limiter le nombre d'entiers premiers à utiliser.

Nous avons déterminé, dans la partie *Résolution des systèmes zéro-dimensionnels*, la notion d'entier premier de bonne réduction pour la représentation univariée rationnelle . Nous avons exploité ces résultats pour élaborer une méthode expérimentale de type multi-modulaire, dans le cas d'idéaux radicaux, dont le principe est le suivant :

- Détermination de l'élément séparant : Si $\mathcal{I}$ est radical, ce que nous avons appelé *prédiction d'un élément séparant* peut être modifié en *calcul d'un élément séparant*. En effet, si $\mathcal{I}$ est radical, les puissances t^i , $i = 0, \ldots, D-1$ forment une base de $\mathcal{A}$, et en particulier le polynôme $f(t,T)$ est sans facteur carré. Modulo p, la partie sans facteurs carrés du polynôme caractéristique de l'élément séparant choisi est alors nécessairement égal à D si p est de bonne réduction. Le nombre d'entiers premiers de mauvaise réduction, ainsi que le nombre de polynômes potentiellement séparants à considérer étant fini, ceci nous donne un moyen de déterminer simplement un élément t séparant $V(\mathcal{I})$ par une méthode purement modulaire et sans test a posteriori.

- Calculs modulaires : Pour tout p de bonne réduction, $M(t)$ est inversible modulo p. Les systèmes décrits plus haut admettent donc tous une unique solution sauf pour un nombre fini d'entiers premiers (facilement détectables). L'idée est alors d'effectuer ces calculs modulaires pour un nombre suffisant d'entiers $p_1, \ldots, p_k$ et d'utiliser le théorème chinois pour reconstituer les solutions dans $\mathbb{Z}[T]$. Par nombre suffisant, nous entendons un nombre k tel que $\prod_{i=1}^{k} p_i > 2e_{max}^2$, où e_{max} est la valeur absolue du plus grand entier apparaissant dans les divers résultats.

  Il existe de nombreuses bornes théoriques permettant de majorer e_{max}, mais elles s'avèrent en général extrêmement larges. On peut comprendre aisément qu'un borne trop large à des effets catastrophiques sur le comportement en pratique d'un algorithme multi-modulaire.

  A l'image de certains algorithmes performants pour les calculs de pgcd (voir [DST93]), l'idée est d'effectuer périodiquement la remontée chinoise et de stopper le processus lorsque le résultat ne varie plus. Il est alors très probable que celui-ci soit exact.

- Vérification : Bien entendu, on ne peut se contenter d'un résultat *probablement bon*, aussi, il nous faut fournir un test de vérification, pour , au besoin, poursuivre le calcul.

  C'est sur ce point que les recherches sont effectuées actuellement. On peut toutefois indiquer une méthode un peu naïve (implantée pour l'occasion dans la boite à outils RealSolving). En supposant que la matrice $M(t)$ (à coefficients dans $\mathbb{Q}$) ait été calculée, il est suffisant de vérifier que les résultats obtenus sont solution des systèmes $M(t)X = \overrightarrow{f'X.i}$.

Les expériences effectuées sur cette méthode simple montrent déjà de réels progrès en ce qui concerne l'occupation mémoire mais également en termes de temps de calcul.

Par exemple, pour le système $e14$, l'occupation mémoire chute d'un facteur 5 et le temps de calcul d'un facteur 4, la construction de la matrice $M(t)$ (à coefficients rationnels) et la vérification du résultat représentant plus de 90 pour cent du temps de calcul.

Décomposition des problèmes

Une autre manière de diminuer l'occupation mémoire engendrée par nos algorithmes est de casser le problème.

Un algorithme, dû à Daniel Lazard (voir [Laz92]) permet, dans certains cas, de décomposer un système d'équations en un ensemble de systèmes triangulaires. Quelques expériences récemment menées par J.C. Faugère montrent que ces ensembles pourront être calculés très rapidement à partir d'une base de Groebner (pour un ordre admissible quelconque).

En introduisant quelques calculs de pgcd supplémentaires, ils semble par ailleurs possible de donner directement l'idéal radical associé à chacune des composantes produites.

Ces petits systèmes ont le bon goût d'être des bases lexicographiques, par conséquent directement exploitables par nos algorithmes.

Nous avons testé la combinaison de ces différentes techniques sur un problème bien connu des spécialistes : le système *cyclic 7*. Ce système n'était pas directement exploitable à priori (aucune stratégie modulaire n'avait été implantée à ce moment) par nos algorithmes, puisque le nombre de racines complexes (et par conséquent la dimension de l'espace vectoriel dans lequel nous travaillons) est 924. Le calcul des ensembles triangulaires, nous à donné plusieurs composantes dont la plus importante était de dimension 384. L'application de nos méthodes sur chacune des composantes à rendu possible le comptage des racines réelles.

Distribution des calculs/calcul parallèle

Le logiciel Gb est conçu pour fonctionner sur un réseau de stations de travail. Un certain nombre d'outils ont donc été développés par Jean-Charles Faugère pour permettre la distribution des calculs. Les bases de Groebner étant actuellement un pré-requis indispensable pour nos algorithmes, dans le souci de construire des outils homogènes nous nous sommes adaptés à ce mode de fonctionnement, en particulier en interfaçant Gb et notre boite à outils RealSolving.

Nous nous sommes rapidement rendu compte que nous pouvions tirer profit de ce genre de techniques, d'une part pour distribuer nos propres calculs, et en particulier lancer en parallèle plusieurs stratégies de calcul, d'autre part pour parallèliser les algorithmes eux-mêmes.

A titre d'exemple, nous avons élaboré une version de l'algorithme *inégalités simultanées* pouvant fonctionner en parallèle sur plusieurs stations de travail. Dans la version de démonstration de notre boite à outils, une station est réservée à la *machinerie* (indépendante de l'algorithme de comptage), les autres à l'exécution des *boites noires* (Hermite ou Sturm-Habicht selon le nombre de variables).

En pratique, le temps utilisé pour les communications étant négligeable, on arrive ainsi à diviser le temps de calcul par le nombre de machines à disposition.

Plusieurs de nos algorithmes ont une structure *naturellement parallèle* : la table de multiplication (il s'agit d'un arbre), l'algorithme RUR, ... , et en particulier les algorithmes faisant appel à des techniques multi-modulaires.

Géométrie réelle

A l'image de ce que nous avons réalisé pour l'application *interpolation de Birkhoff*, les algorithmes pour l'étude des systèmes zéro-dimensionnels pourront être utilisés pour des problèmes plus généraux (études d'hypersurfaces, élimination des quantificateurs). La méthode en cours d'étude actuellement fait référence à [BPR94].

On suppose que K est un corps ordonné de clôture réelle R et de clôture algébrique C. Soit P un polynôme de $K[X_1, \ldots, X_n]$ dont l'ensemble des zéros dans R^n (resp. C^n) est noté $V_R(P)$ (resp. $V(p)$). En posant, en supposant que X_{n+1} soit une nouvelle variable :

$$Q = P^2 + (X_1^2 + \ldots + X_n^2 + X_{n+1}^2 - \Omega^2)^2,$$

avec Ω infiniment grand positif et , on montre que $V_R(Q) \subset (R\langle 1/\Omega\rangle)^{n+1}$ est contenu dans la boule ouverte de centre 0 et de rayon $\Omega + 1$. Dans [BPR94], il est montré de plus que l'extension de chaque composante connexe de $V_R(P)$ dans $(R\langle 1/\Omega\rangle)^n$ contient la projection sur $(R\langle 1/\Omega\rangle)^n$ d'une cellule de $V_R(Q)$.

En notant $d_1, \ldots, d_n$ les degrés de Q en chacune des variables et d le degré total de Q, on pose alors :

$$Q_1 = (1-\epsilon)Q + \epsilon(X_1^{2(d_1+1)} + \ldots + X_n^{2(d_n+1)} + X_{n+1}^6 - (n+1)(\Omega+1)^{2(d+1)})$$

où ϵ est cette fois un terme infiniment petit.
Il est alors montré , dans [BPR94], les propriétés suivantes :

- L'ensemble algébrique $V_R(Q_1)$ est lisse et borné.
- Les polynômes
$$Q_1, \frac{\partial}{\partial X_2}Q_1, \ldots, \frac{\partial}{\partial X_{n+1}}Q_1$$
forment une base de Groebner pour l'ordre du degré lexicographique.

- L'ensemble K' des points critiques (dans $C\langle\epsilon, \Omega\rangle$) de la projection par rapport à l'axe de coordonnées X_1 est fini (ensemble des zéros définis par les équations du point précédent)

- Pour toute cellule $\mathcal{C}$ de $V_R(Q)$, il existe un point $p \in K'$ tel que $eval_\epsilon(p) \in \mathcal{C}$ (appliquer $eval_\epsilon$ revient à faire tendre ϵ vers 0).

A priori, connaître une RUR de l'idéal engendré par les polynômes

$$Q_1, \frac{\partial}{\partial X_2}Q_1, \ldots, \frac{\partial}{\partial X_{n+1}}Q_1,$$

permet de déterminer complètement, et de façon automatique, l'ensemble K' et donc de pouvoir fournir un point par composante connexe de $V_R(P)$. Il est à noter qu'aucun calcul de base de Groebner n'est nécessaire.

Pour l'heure, les problèmes sont d'ordre calculatoire :

- La détermination de K' nécessite le calcul d'une RUR d'un système polynomial que nous pouvons considérer, d'un point de vue pratique, à coefficients dans $K[\epsilon, \Omega][X_1, \ldots, X_{n+1}]$. Une représentation univariée rationnelle sera alors un ensemble de polynômes de $K[\epsilon, 1/\epsilon, \Omega][T]$.

- Plusieurs élévations au carré ont été réalisées, augmentant ainsi de façon très importante la dimension de l'espace vectoriel dans lequel nous réalisons les calculs.

Quelques éléments de solutions ont été établis :

- On peut découper en deux le problème, en posant $\Omega = 1$: Un premier calcul brutal permet alors de trouver un point par composante connexe de l'intersection de $V_R(P)$ et de la boule de centre 0 et de rayon 1. En compactifiant ensuite $V_R(P)$ relativement à un point o n'appartenant pas à l'hypersurface, nous pouvons traiter les autres composantes connexes, en prenant soin de retirer, si besoin est, le point o de l'ensemble des solutions.

- Une astuce se calcul consiste à remplacer ϵ par $1/\epsilon$ dans l'expression de Q_1, ce qui nous donne

  $$\epsilon Q_1 = (\epsilon - 1)Q + (X_1^{2(d_1+1)} + \ldots + X_n^{2(d_n+1)} + X_{n+1}^6 - (n+1)(\Omega+1)^{2(d+1)})$$

  En étudiant ϵQ_1, on peut alors remarquer qu'aucune puissance de $1/\epsilon$ n'apparaîtra en cours de calculs.

- Dans l'expression de Q_1, les termes de la forme $2(d_i+1)$ peuvent être remplacés par des entiers pairs strictement supérieurs à d_i, réduisant ainsi la dimension de l'espace vectoriel de travail.

En effectuant cette série de modifications, les calculs nécessaires pour déterminer une représentation univariée rationnelle sont désormais effectués en utilisant l'anneau de coefficients $K[\epsilon]$. Quelques essais ont été effectués en utilisant l'interpolation de Lagrange de sorte à réutiliser les implantations existantes, mais la solution n'est pas satisfaisante. Une arithmétique appropriée pour les infinitésimaux permettra sûrement de mener à bien cette méthode.

Enfin, la mise au point de cet algorithme servira de base (voir [BPR94]) à une méthode générale d'élimination des quantificateurs.

Épilogue

Si il nous fallait faire un bilan des progrès effectués, il nous serait bien difficile de les affecter à un quelconque domaine :

- Algèbre/Géométrie : Il est certain, par exemple, que la formalisation de ce que nous avons nommé la représentation univariée rationnelle nous à permis de mieux comprendre les objets que nous manipulions. La mise au point d'une méthode systématique de calcul nécessite une bonne compréhension générale des outils standard, mais surtout impose la résolution de tous les cas particuliers inhérents au problème.

- Complexité : C'est généralement l'étape qui précède l'étude algorithmique. Ce n'est qu'après une analyse détaillée de la complexité d'un problème que nous décidons d'accorder du temps pour une étude algorithmique, voire une implantation. Par exemple, l'algorithme de Hermite a demandé un lourd investissement. La première implantation viable, en pratique, a nécessité plusieurs milliers de lignes de code C++, alors que le programme naïf s'écrit, dans un langage spécialisé (Maple,Axiom,...), à l'aide de quelques instructions. Ce sont les résultats de complexité théorique, montrant que la méthode est intrinsèquement bonne, qui nous ont incité à persévérer.

- Algorithmique : Il est clair que ce travail à été d'une importance capitale pour préparer l'implantation de nos algorithmes et nous assurer de leur bon comportement en pratique. Deux algorithmes de même complexité théorique n'ont pas le même comportement en pratique et bien souvent, seule l'expérience permet de faire un choix.

- Informatique : les considérations informatiques n'ont pas été abordées dans cette thèse (une courte description en est toutefois donnée en annexe). Ceci ne préjuge en aucun cas de l'importance et la difficulté de ce travail. Pour implanter de façon efficace nos différents algorithmes, il a fallu utiliser un nombre conséquent de techniques (tables de hachage - représentation des données) ou d'outils (gestion de mémoire - arithmétiques efficaces - langage c++) purement informatiques, et surtout les adapter aux contraintes imposées par le calcul exact.

En ce qui concerne cette thèse, les résultats les plus significatifs ont été obtenus en intégrant à notre travail l'ensemble de ces différents points de vue.

Annexes

Annexe A

La *Test-suite*

Nous donnons ici la liste détaillée des exemples nous ayant servi pour mesurer les performances de nos algorithmes. Ils sont tous issus de la *PoSSo-test-suite* comprenant un panel d'environ 14 systèmes provenant aussi bien de problèmes d'école, de robotique ou encore de chimie.

Nous avons lancé notre boite à outils sur la plupart de ces exemples et nous avons retenus pour les tests 9 systèmes, parmi les plus représentatifs (nombre de racines, croissance des coefficients) en prenant soin d'éliminer ceux pour lesquels les mesures ne sot pas significatives (temps de calculs au dessous de la seconde par exemple).

Nous y avons adjoint 6 exemples issus de problèmes de robotique.

- e1 : caprasse

```
vars:=[x,y,z,t];
D:=HDMP(vars,INT);
F1:D:=y**2*z+2*x*y*t-2*x-z;
F2:D:=-x**3*z+4*x*y**2*z+4*x**2*y*t+2*y**3*t+4*x**2-
      10*y**2+4*x*z-10*y*t+2;
F3:D:=2*y*z*t+x*t**2-x-2*z;
F4:D:=-x*z**3+4*y*z**2*t+4*x*z*t**2+2*y*t**3+4*x*z+4*z**2-
      10*y*t-10*t**2+2;
```

- e2 : cassou

```
vars:=[b,c,d,e];
D:=HDMP(vars,INT);
F1:D:=15*b**4*c*d**2+6*b**4*c**3+21*b**4*c**2*d-144*b**2*c-
      8*b**2*c**2*e-28*b**2*c*d*e-648*b**2*d+36*b**2*d**2*e+
      9*b**4*d**3-120;
```

```
F2:D:=30*c**3*b**4*d-32*d*e**2*c-720*d*b**2*c-24*c**3*b**2*e-
      432*c**2*b**2+576*e*c-576*d*e+16*c*b**2*d**2*e+
      16*d**2*e**2+16*e**2*c**2+9*c**4*b**4+5184+
      39*d**2*b**4*c**2+18*d**3*b**4*c-432*d**2*b**2+
      24*d**3*b**2*e-16*c**2*b**2*d*e-240*c;

F3:D:=216*d*b**2*c-162*d**2*b**2-81*c**2*b**2+5184+
      1008*e*c-1008*d*e+15*c**2*b**2*d*e-15*c**3*b**2*e-
      80*d*e**2*c+40*d**2*e**2+40*e**2*c**2;

F4:D:=261+4*d*b**2*c-3*d**2*b**2-4*c**2*b**2+22*e*c-22*d*e;
```

- e3 : sendra

```
vars:=[x,y];
D:=HDMP(vars,INT);
F1:D:=-270*x**4*y**3-314*x*y**4-689*x*y**3+1428;
F2:D:=36*x**7+417*x**6*y-422*x**5*y**2-270*x**4*y**3+
      1428*x**3*y**4-1475*x**2*y**5+510*x*y**6-200*x**6-
      174*x**5*y-966*x**4*y**2+529*x**3*y**3+269*x**2*y**4+
      49*x*y**5-267*y**6+529*x**4*y+1303*x**2*y**3-314*x*y**4+
      262*y**5+36*x**4-788*x**2*y**2-689*x*y**3+177*y**4;
```

- e4 : katsura-5

  Les exemples e4 et e5 sont dus à Katsura. Ils s'agit de systèmes d'équations issues de problèmes de physique (magnétisme), dont le système générique est (Katsura-n) :

$$u_m - \sum_{i=-N}^{N} u_i u_{m-i}; m = -N+1..N+1$$
$$1 - \sum_{i=-N}^{N} u_i$$
$$u_{-m} = u_m; m = 1..2N-1$$
$$u_m = 0; m = N+1..2N-1$$

- e5 : katsura-7
- e6 : cyclic-5

  Les exemples e6, e16 et e17 sont les systèmes cycliques utilisés par bon nombre de développeurs pour tester les algorithmes. Le système générique (cyclic-n) est donné par :

$$\sum_{i=1}^{n}\prod_{j=0}^{k} x_{i+j}; k=0..n-2$$
$$\prod_{i=0}^{n} x_i - 1$$

- e7 : reimer-5

  Le système d'équations génériques (reimer-n) est donné par :

$$-1/2+\sum_{i=1}^{n}(-1)^{i+1}x_i^k; k=2..n+1$$

- e8 discret-3

$$\begin{gathered}
12600y-13440-420y^2-280yz-210yt-168yu-140yv-120ys-105ya \\
18900z-20160-840yz-630z^2-504zt-420zu-360zv-315zs-280za \\
12600t-13440-630yt-504zt-420t^2-360tu-315tv-280ts-252ta \\
103950u-110880-5544yu-4620zu-3960tu-3465u^2-3080uv-2772us-2520ua \\
83160v-88704-4620yv-3960zv-3465tv-3080uv-2772v^2-2520vs-2310va \\
900900s-960960-51480ys-45045zs-40040ts-36036us-32760vs-30030s^2-27720sa \\
772200a-823680-45045ya-40040za-36036ta-32760ua-30030va-27720sa-25740a^2 \\
675675b-720720-40040by-36036bz-32760bt-30030bu-27720bv-25740bs-24024ba
\end{gathered}$$

- e9 planN3I

  Les systèmes planXX et spatXX sont décrits dans []. Il s'agit de modélisations de plate-formes de Stewart.

- e10 spatV2A

- e11 spat2A2I

- e12 spatAV2I

- e13 rsspat66

- e14 : r24c1 Ce système est celui construit au chapitre *Robots parallèles.* Il correspond à un robot parallèle ayant 24 position réelles.

- e15 : Bellido Ce système à été proposé par A. Bellido pour la *test-suite* du projet PoSSo. Il s'agit là encore d'un problème de robotique.

$$z_1^2 + z_2^2 + z_3^2 - 12z_1 - 68$$
$$z_4^2 + z_5^2 + z_6^2 - 12z_5 - 68$$
$$z_7^2 + z_8^2 + z_9^2 - 24z_8 - 12z_9 + 100$$
$$z_1z_4 + z_2z_5 + z_3z_6 - 6z_1 - 6z_5 - 52$$
$$z_1z_7 + z_2z_8 + z_3z_9 - 6z_1 - 12z_8 - 6z_9 + 64$$
$$z_4z_7 + z_5z_8 + z_6z_9 - 6z_5 - 12z_8 - 6z_9 + 32$$
$$2z_2 + 2z_3 - z_4 - z_5 - 2z_6 - z_7 - z_9 + 18$$
$$z_1 + z_2 + 2z_3 + 2z_4 + 2z_6 - 2z_7 + z_8 - z_9 - 38$$
$$z_1 + z_3 - 2z_4 + z_5 - z_6 + 2z_7 - 2z_8 + 8$$

Annexe B

La boite à outils RealSolving

Cette annexe est destinée à décrire la boite à outil RealSolving. Nous ne faisons ici un rapide survol de ses spécificités. Pour plus d'information (en particulier concernant l'implantation), une documentation complète est fournie avec le logiciel.

B.1 Présentation

La boite à outils RealSolving est dédiée à l'étude des systèmes zéro-dimensionnels. En quelques chiffres, ce travail représente l'écriture d'un code C++ d'environ 40 000 lignes, portable sur les machines les plus variées :

- Stations Sun/sparc sous Solaris 1 ou 2
- IBM RS6000 sous Aix 3.2
- IBM PC sous Linux 1.2
- Stations Dec/alpha sous OSF/1 2
- Stations Hewlett Packard sous HP-UX

et interfacé avec les logiciels les plus performants du moment (Gb, librairie PoSSo, PoSSo-Server, Axiom (via Gb)) pour le calcul des bases de Groebner.

Le code à entièrement été développé durant le projet PoSSo (1992-1995) et sa rapidité/portabilité à été directement conditionnée par les outils performants développés pendant la durée du projet par diverses équipes européennes (gestion de mémoire , arithmétique).

Il est à noter que l'équipe de Santander (Université de Cantabria) à contribué à l'écriture des fonctionnalités de RealSolving pour les polynômes en une variable et pour la partie Inégalités simultanées.
La boite à outils peut fonctionner selon 3 modes :

- Comme partie du PoSSo-Server[1]
- Comme librairie C++ (sous ensemble de la librairie PoSSo).
- A partir de Gb, l'appel aux fonctions spécifiques de RealSolving se faisant après d'addition de serveurs compatibles avec la structure à architecture distribuée de Gb.

La stratégie adoptée pour l'écriture de ce code à été de réutiliser au maximum les outils existants. Ceci a été facilité par la participation au projet PoSSo, durant lequel bon nombre d'outils nouveaux performants ont étés élaborés (arithmétique, gestion de mémoire, bases de Groebner). RealSolving a été naturellement incorporé à Gb pour deux raisons principales :

- Le haut niveau de performances atteint par ce logiciel pour le calcul de base de Groebner, sa portabilité et la simplicité d'utilisation des outils développés par J.C. Faugère concernant les protocoles de communication.
- Le logiciel Gb existe depuis plusieurs années et nous a permis de de mettre au point notre code, sans attendre et en bénéficient d'un bon repère quant aux niveau de performances à atteindre.

La compatibilité avec ces différentes applications n'a entraîné aucune redondance dans l'écriture du code et permet une utilisation de la boite à outils adaptée aux besoins de l'utilisateur :

Le fait que RealSolving puisse être utilisé comme partie d'une librairie C++ dédiée aux systèmes polynomiaux, permet aux programmeurs d'en inclure les fonctionnalités directement dans leurs propres programmes.

Rendre la boite à outils compatible avec deux systèmes interactifs de calcul (PoSSo-Server et Gb) est un confort pour les personnes désirant appliquer nos algorithmes. D'un coté, le PoSSo-Serveur permet d'effectuer des calculs assez généraux (systèmes à coefficients autres que rationnels ou entiers) tirant ainsi partie des implantations efficaces et souples d'utilisation de coefficients génériques.

Les serveurs RealSolving compatibles avec Gb permettent de faire simplement des calculs que nous qualifierons *de l'extrême* sur des systèmes à coefficients entiers. D'une part, le code de RealSolving à été optimisé pour les calculs sur ce type de systèmes (structures de données et gestion de mémoire appropriées), d'autre part la structure à architecture distribuée de Gb permet de lancer en parallèle différentes stratégies de calcul, sur des machines d'architectures disparates. Notons que l'interface développée pour la communication avec Gb nous à permis entre autres de mettre au point des algorithmes parallèles pouvant être lancés sur un réseau de stations de travail connectées simplement par Internet (donc nul besoin de se munir

[1] Interface utilisateur de la librairie PoSSo

d'une machine parallèle) ou encore de distribuer les différentes parties de nos calculs sur plusieurs machines (ceci à eu pour avantage rendre tout simplement possibles des calculs effectués sur de très gros systèmes en diminuant l'occupation mémoire ou du moins en la répartissant sur plusieurs machines).

La figure B.1 décrit les possibilités offertes par RealSolving en fonction du type de résultat recherché.

B.2 Outils

Cette partie décrit les caractéristiques techniques de la boite à outils. Sans rentrer dans les détails, nous donnons ici quelques précisions sur le code.

B.2.1 Gestion de mémoire

L'un des problèmes fondamentaux posés par les algorithmes de calcul formel, d'un point de vue implantation, est la gestion de mémoire. Le problème majeur est la variation au cours du temps des tailles des objets utilisés. Pour prendre un exemple simple, regardons l'effet produit par le calcul suivant sur une machine dont la taille d'un mot élémentaire est $2^3 2$ (on suppose qu'un entier positif est représenté par son expression en base 2):

```
[1]>i:=2^32-1;
[2]>i:=i*2;
```

Après l'étape `[1]`, la représentation binaire de i est exactement de taille 32, soit un mot mémoire, alors qu'à la deuxième étape celle-ci est de longueur 33 et par conséquent le codage de i nécessite deux mots mémoire. Entre temps il aura fallu par conséquent augmenter la place réservée pour i.

A priori les données sont stockées séquentiellement en mémoire, et la réallocation, pour un objet donné, se traduit souvent, par une nouvelle attribution de mémoire. La conséquence immédiate de ce genre de manipulation est de fragmenter la mémoire réservée à une application. On peut envisager une méthode naïve consistant à essayer, autant que possible, ré-allouer un espace libéré, lors de la création d'un nouvel objet. Deux objections à cela :

- Le coût de cette opération devient vite prohibitif compte tenu du nombre d'objets crées lors d'un calcul. Par exemple, l'utilisation des allocateurs et destructeurs standards peut entraîner rapidement un coût supérieur à celui des opérations arithmétiques.

- Dans bon nombre d'algorithmes, la taille des objets augmente graduellement au cours des calculs, si bien que les chances de trouver un espace libre de bonne

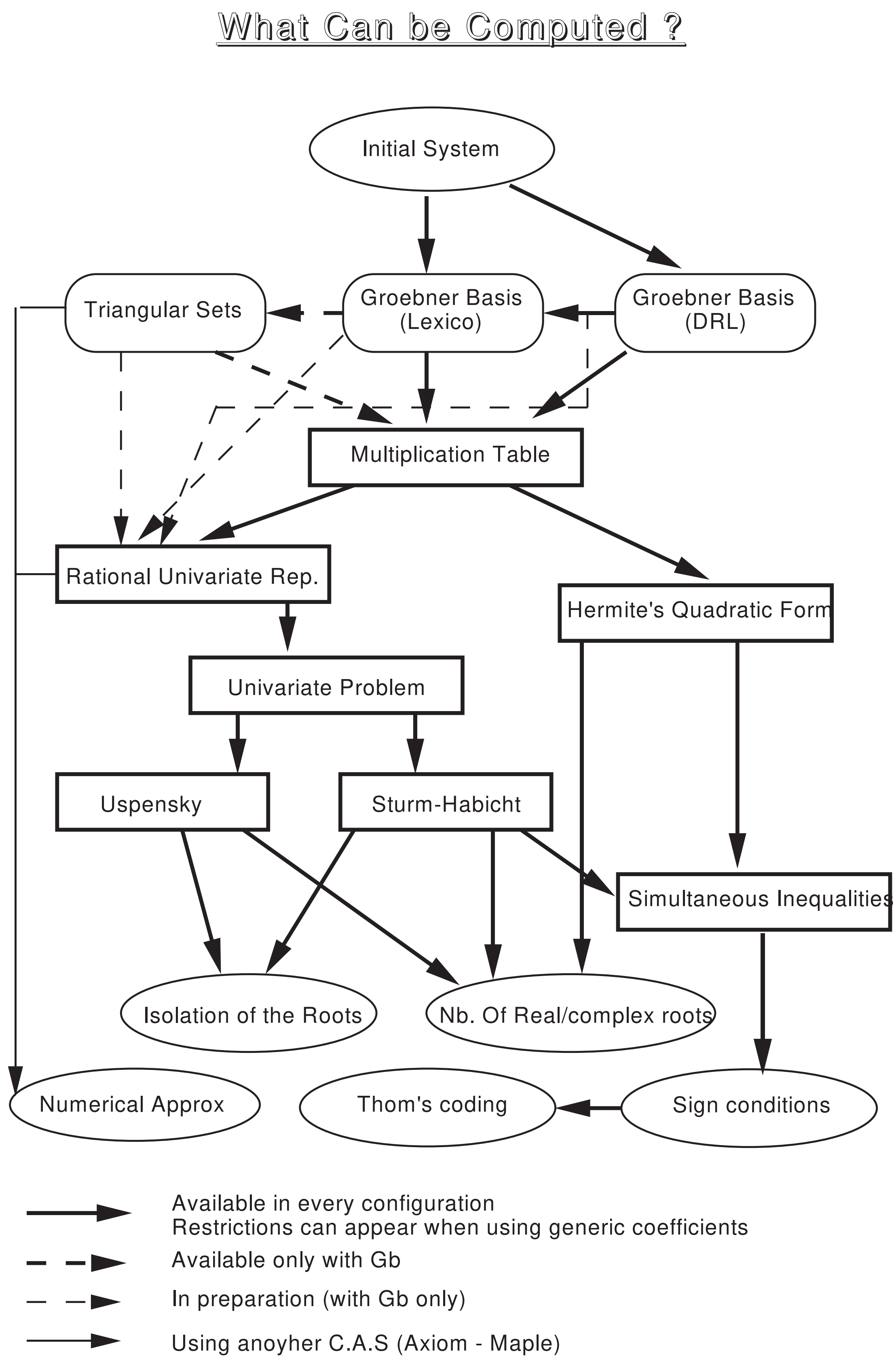

Figure B.1: Quelle stratégie pour quel résultat ?

taille, à un instant donné, est faible. On peut par exemple considérer l'effet de ce type de méthode pour un calcul de déterminant par la méthode de Gauss.

Le principe de base des *garbage collector* (traduction : ramasse - miettes) utilisés par défaut dans l'implantation de la boite à outils *RealSolving* est standard : Lorsque l'occupation mémoire atteint un certain niveau, les objets utilisés (non détruits) dans le programme sont détectés, puis déplacés de sorte à reconstruire un bloc compact, et retrouver une occupation mémoire optimale.

En particulier, l'utilisation du *garbage collector PoSSo-Cmm* permet de régler la fréquence de déclenchement de cette opération en fonction des besoins.

Toutefois, certaines parties du code ont pu être adaptés de sorte à optimiser les opérations d'allocation ou de *collecte* des éléments non utilisés. En particulier, le *garbage collector PoSSo-Cmm* permet de personnaliser la gestion de certaines zones (*heap*) de mémoire en fonction des besoins. Pour prendre un exemple concret, étudions le cas d'un algorithme utilisant une arithmétique modulaire : la prédiction modulo p d'un élément séparant (voir Algorithme RUR). Dans cette partie du programme, une forme quadratique doit être réduite dans $\mathbb{Z}/p\mathbb{Z}$. Nous savons qu'aucune croissance de coefficients ne sera à déplorer en cours de calcul et que par conséquent, si des opérations destructives sont utilisées,l'occupation mémoire ne devrait varier que très peu. Il serait donc incongru de consommer du temps en collectant les objets intervenant dans cette partie du code. La stratégie adoptées consiste justement à ne pas solliciter le *garbage collector*, en allouant les données dans une zone spéciale temporaire qui n'est pas soumise à l'action du Garbage collector, d'effectuer le calcul, de recopier le résultat dans la zone de mémoire principale du programme puis de détruire les données accumulées dans la zone temporaire (stratégie de type *mark and copy*). (Il est à noter que cette stratégie est en cours de test et ne sera présente que dans la prochaine version de *RealSolving*.)

Le code *RealSolving* peut fonctionner sans aucun gestionnaire de mémoire, ce qui est fortement déconseillé, et est compatible avec deux *garbage collector* différents. différents :

- *PoSSo-Cmm* (voir [AF94]) : utilisé par défaut pour ses performances et ses possibilités (plusieurs types de gestion de mémoire sont utilisables simultanément).

- *Boehm* (voir [Bo88]) : par sécurité, *PoSSo-Cmm* n'ayant pas été porté à ce jour sur toutes les plateformes que nous utilisons - en particulier les stations Alpha ou Hp -, *RealSolving* peut être compilé pour fonctionner avec cet outil. Il faut noter que ses performances sont moins bonnes (en termes d'occupation mémoire du programme) que celles de *PoSSo-Cmm*.

B.2.2 Arithmétique

Initialement, RealSolving était dédié exclusivement aux systèmes à coefficients entiers (les plus couramment étudiés). A l'image de l'ensemble de la librairie PoSSo, l'implantation finale du code est devenue indépendante de l'arithmétique choisie. Pour sa part, RealSolving possède une classe de coefficients particulière (`PL_RSCoef`) rendant transparent tout changement d'arithmétique.

L'utilisateur peut ainsi modifier ou ajouter lui-même de nouvelles arithmétiques sans avoir besoin de modifier le code propre de la librairie, en respectant simplement quelques conventions (signature des fonctions).

Selon l'option de choisie pour la compilation de la librairie, les coefficients sont déterminés soit à la compilation du code, soit de manière générique.

Dans le premier cas, les binaires fournis sont prévus pour fonctionner exclusivement avec cette arithmétique et pas une autre.

Dans le second cas, l'anneau des coefficients peut être changé à tout moment, après le lancement du programme.

Techniquement, c'est en utilisant des classes et fonctions virtuelles (classes *vide* - du moins ne contenant que des signatures de fonctions et déclaration diverses) que le code est compilé. Ce n'est qu'au *linkage* que des liens réels sont effectués (ce qui nécessite une initialisation par défaut de l'anneau de coefficients).
Cette stratégie offre deux avantages :

- Une écriture de code générique *à la Axiom*, ou l'on ne précise que la classe générique des coefficients (anneau-anneau ordonné -corps).

- L'élaboration de nouvelles implantation de coefficients, indépendamment du code de base. En particulier, dans une version initiale de l'algorithme de calcul de pgcd modulaire nous avons réutilisé intégralement le code générique et simplement implanté une nouvelle arithmétique, pour les calculs modulaires.

Toutefois, lorsque l'on compile la librairie de sorte à utiliser les entiers longs développés dans PoSSo (dérivé de la librairie gnu-gmp), un certain nombre d'optimisations rendent le code nettement plus efficace. En particulier des classes très spéciales ont été conçues pour le *garbage collector* dans l'implantation de RealSolving.

D'autres implantations génériques sont présentes dans la librairie PoSSo. Par exemple, les monômes (et l'ordre induit pour le calcul de bases de Groebner) sont définis de manière générique.
Exemple d'utilisation de la librairie C++ :

.
.
.

```
.

// ***************************************
// PARTIE RESERVEE A LA DETERMINATION DES
// COEFFICIENTS
// ***************************************

#ifdef PL_USE_GENERIC_COEFF

// on choisi un anneau de coefficients
// dans le cas ou la librairie a ete
// compilee pour fonctionner avec les
// coefficients generiques

    setCoeffRing(createBCoeffRing());

#endif

.
.
.

// *****************************************
// PARTIE INDEPENDANTE DE TOUTE CONFIGURATION
// On suppose que ``gr'' contient une base de
// Groebner
// *****************************************

//calcul de la table de multiplication

  PL_MultTable *hmtab=new PL_MultTable(gr);

// calcul de la representation univariee
// rationnelle

  PL_RSList<PL_UniPol> lp=RUR(hmtab);
.
.
.
```

B.2.3 Architecture client/serveur

Dans la version compatible avec Gb, RealSolving fonctionne sous un mode de type client/serveur, le programme principal de Gb (Gb - central servant de client).

Cette stratégie apporte un progrès à bien des égards puisque cela permet d'une part de fragmenter les calculs (offrant en cela une solution à la consommation de mémoire pour les cas extrêmes), mais également d'effectuer en parallèle certains calculs (soit pour lancer plusieurs stratégies en même temps, soit pour construire des algorithmes parallèles).

Le principal problème à résoudre est celui des communications entre programmes, surtout lorsque ceux-ci s'exécutent sur des machines de types différents.

Nous avons, à cet effet, réutilisé le protocole implanté dans Gb par Jean-Charles Faugère, pour son efficacité et sa portabilité. Les données sont transmises via le réseau (Internet).

Nous avons convenu de transmettre l'information au format binaire par souci d'efficacité : les temps de transmission doivent être nettement inférieurs aux temps de calcul si l'on veut prétendre faire du calcul parallèle et le volume d'informations transmis minimal. La représentation ascii d'un entier e occupe $log_{10}(e)/4$ mots machine sur une machine 32 bits alors que sa représentation binaire en occupe $log_2(e)/32$.

Toutefois, ce type de transmission (binaire) pose un certain nombre de problèmes, pour l'échange des données classiques (entiers machine-flottants) entre machines de type différents. Il est évident qu'il faut prêter attention aux longueurs de mots (sur une station alpha les mots sont de 64 bits alors qu'ils ne sont que de 32 bits sur une station sun), mais des ennuis beaucoup plus subtils apparaissent dans les données standard dont le codage dépasse 8 bits, même dans le cas de deux machines 32 bits. Par exemple, si on numérote de 1 à 4 les octets définissant le codage d'un entier machine sur une station Sun/Sparc 20, le même entier sera codé par les octets $[4, 3, 2, 1]$ sur des machines de type `BIG_ENDIAN` ou $[3, 4, 1, 2]$ sur des machines de type `PDP_ENDIAN`. Ainsi, il faut au besoin réorganiser chacun des entiers envoyés.

Une description détaillée du protocole est présente dans la documentation de RealSolving et, à titre de démonstration, un serveur spécial est fourni dans la distribution, appliquant des algorithmes parallèle pour le calcul multi-modulaire de pgcd de deux polynômes (naturellement parallèle) ou encore le codage *à la Thom* des racines d'un polynôme en une variable, utilisant 3 machines d'architectures éventuellement distinctes.

B.3 Utilisation

Cette partie est extraite du manuel de l'utilisateur livré avec RealSolving. Cette partie décrit brièvement comment effectuer les opérations de base selon les trois modes principaux de fonctionnement.

B.3.1 RealSolving servers for Gb

Before going on be sure that you have the right version of Gb an its documentation (greater than 3.1091).

If not and if you have a version of the PoSSo-Library greater than 106, add "lib/RealSolving/Servers" in the list of packages to be compiled in the PoSSo-Library, and compile the library with the options

```
gbserver=yes coeff=simple poly=simple
```

This will build all the PoSSo - library and the needed binary files for Gb will be put into the directory :

```
installed/bin/ServeurRS
installed/bin/ServeurUPINT
...
```

Another way to install these binaries is to get a reduced version of the PoSSo-Library or directly the binaries. The access to these items is restricted, also send before a e-mail to :

```
rouillie@posso.univ-rennes1.fr
```

or consult the mosaic page :

```
http://medicis.polytechnique.fr/~rouillie/rs/rs.html
```

Some operations of RealSolving are not officially available trough Gb, but all are running using servers prepared for demonstrations (see the commented example).

What to do first

We suppose that Gb is running and waiting for instructions. The first thing to do is to check if the RealSolving servers are present. Supposing Gb is running, you have to declare the servers by typing :

```
install("RealUni","ServerUPINT",false)
```

for the univariate server.

```
install("Real","ServerRS",false)
```

for the multivariate server.

There is a simple way to run some servers on remote machines : Create a shell script containing for example the lines :

```
#!/bin/sh
echo "Je lance ServerRS sur jean"
rsh  jean.polytechnique.fr /usr/local/rs/ServerRS $*
```

and name it `MysServer`. Then, give to Gb the intruction :

```
install("RealUni","MyServer",false)
```

to use "MyServer" for univariate problems.

Summary of the available commands

Only few Gb instructions are necessary for running the RealSolving toolbox. Their action depends on the type of data used or on the servers that are installed. The field `verbosity` is an integer (that can be ommited) that denotes the level of verbosity you want (from 0 to 2).

- `RSInitTable(<groebner basis>,<verbosity>);`

  Computes the multiplication table associated to the Groebner basis.

- `RSGSL(<multiplication table>, <verbosity>)`

  Compute the Rational Univariate representation of the system.

- `RSNbRealRoots(<multiplication table>,<verbosity>)`

  Computes the number of Real Roots using Hermite's algorithm.

  `RSNbRealRoots(<univariate polynomial>,<verbosity>)`

  Computes the number of Real Roots using a Sturm-Habicht sequence.

  You can change the server in order to use Uspensky algorithm by typing :

  ```
  install("RealUni","ServerUPINTU",false)
  ```

  In this case, if the verbosity level is equal to 2 you will see on the screen the isolation intervals.

- `RSNbComplexRoots(<multiplication table>,<verbosity>)`

  Computes the number of complex Roots using Hermite's algorithm.

  `RSNbComplexRoots(<univariate polynomial>,<verbosity>)`

  Computes the number of Real Roots sequence.

Commented Gb-session

```
--
--
-- Installation of the RealSolving Servers
--
--

install("RealUni","ServerUPINT",false)
install("Real","ServerRS",false)

--
--
-- Initial system
--
--

l:List(HDMP([x1,x2,x3,y1,y2,y3,z1,z2,z3],INT)):=[1*x1**2+ _
1*y1**2+1*z1**2-1,1*x1**2+1*y1**2+1*z1**2-2*x1-1,1*x2**2 + _
1*y2**2+1*z2**2-2*x2,1*x2**2+1*y2**2+1*z2**2-2*x2-2*y2 , _
1*x3**2+1*y3**2+1*z3**2-2*x3-2*y3-2,1*x1**2-2*x1*x2+1*x2**2 + _
2*x1*x3-2*x2*x3+1*x3**2+1*y1**2-2*y1*y2+1*y2**2 + _
2*y1*y3-2*y2*y3+1*y3**2+1*z1**2-2*z1*z2+1*z2**2 + _
2*z1*z3-2*z2*z3+1*z3**2-2*y1+2*y2-2*y3-3,1*x1**2-1*x1*x2-1*x1*x3 _
+ 1*x2*x3 + 1*y1**2-1*y1*y2-1*y1*y3 + 1*y2*y3 + _
1*z1**2-1*z1*z2-1*z1*z3 + 1*z2*z3-1,1*x1**2-2*x1*x3 + 1*x3**2 + _
1*y1**2-2*y1*y3 + 1*y3**2 + 1*z1**2-2*z1*z3 + 1*z3**2-3 , _
1*x1**2-2*x1*x2 + 1*x2**2 + 1*y1**2-2*y1*y2 + 1*y2**2 + _
1*z1**2-2*z1*z2 + 1*z2**2-1];
--
--
-- Compute the groebner basis wrt the DRL ordering
--
--
g:=sugar(l);
--
--
-- Multiplication table associated to g
--
--
T:=RSInitTable(g,1);
```

```
--
--
-- Number of Real Roots  (Hermite's method)
--
--
RSNbRealRoots(T)
--
--
-- Rational Univariate Representation
--
--
rur:=RSGSL(T));
--
--
-- Number of Real Roots of the first polynomial of rur
--
--
n0:=RSNbRealRoots(first(rur),0)

--
-- Number of Real Roots of the first polynomial of rur
--
--
c0:=RSNbComplexRoots(q0,0)

--
-- Computation of the lexicographic Groebner basis
-- (FGML Algorithm
--
gl:=totolex(g);
I:=pretendgbasis(gl);
--
--
-- Triangular sets
--
--
ttl:=triangSets(tl:=lextri(gl));
gl1:=sugar(ttl.1);
gl2:=sugar(ttl.2);
gl3:=sugar(ttl.3);
gl4:=sugar(ttl.4);
```

```
--
--
-- Hermite's algorithm on the first component
--
--
T1:=RSInitTable(gl1,0);
n1:=RSNbRealRoots(T1)
c1:=RSNbComplexRoots(T1)
--
--
-- RUR+Sturm-Habicht on the second component
--
--
T2:=RSInitTable(gl2,0);
q2:=first(RSGSL(T2));
n2:=RSNbRealRoots(q2,0)
c2:=RSNbComplexRoots(q2)
--
--
-- Change the Univariate server for a demonstration of
-- Uspensky algorithm
--
install("RealUni","ServerUPINTU.lsrun",false)
--
--
-- Isolation and computation of the roots
-- (RUR + Uspensky) on the 4th component
--
T4:=RSInitTable(gl4,0);
q4:=first(RSGSL(T4));
n4:=RSNbRealRoots(q4,2)
c4:=RSNbComplexRoots(q4,0)
```

B.3.2 RealSolving inside the PoSSo-Server

Before going on, be sure that you have a recent version of the PoSSo- library (106 or later). In such case, you have read the general documentation concerning the PoSSo-Server and you are ready to go on.

Summary of the available commands

In the following description some notations are used:

```
-[id]          is the numerical identifier of a remote object.
-[id(XX)]      is the numerical identifier of an object
               of remote type XX.
-[Directives] is the a directive list as explained in
               Directive specifications
```

The RealSolving's remote types supported are :

```
PL_UniPol                      (UP)
PL_RSList<PL_UniPol>           (LUP)
PL_MultTable                   (MT)
PL_CondSign                    (SC)
```

Description of the available RealSolving Operations

```
UP_Open                        --->    [id(UP)]
```

Open an univariate polynomial, returns its address

```
UP_Put                 [id(UP)] [UP]
```

Put the univariate polynomial UP in the address id(UP)

```
UP_Get                 [id(UP)]  ------> [UP]
```

Get the univariate polynomial that is in the address id(UP)

```
UP_Copy                [id(UP1)] [id(UP2)]
```

Copy an univariate polynomial from id(UP1) to id(UP2)

```
UP_CountRoots          [id(UP)]  -------> { [int1], [int2] }
```

Count the number of real (int1) and complex (int2) roots of an univariate polynomial

```
UP_SI                  [id(UP)] [id(LUP)] [id(SC)]
```

Given an univariate polynomial UP and a list of univariate polynomials LUP store in SC the list of signs of the polynomials of LUP at the zeros of UP.

```
UP_RAN                 [id(UP)] [id(SC)]
```

Given an univariate polynomial, find the list of signs of the derivatives of the polynomial at the roots of the polynomial (Thom coding of the polynomial)

`LUP_Open                        ------>  [id(LUP)]`

Open an address for a list of univariate polynomials

`LUP_Put                [id(LUP)] [LUP]`

Put a list of polynomials LUP in the address id(LUP)

`LUP_Get                [id(LUP)]  ------>  [LUP]`

Get the list of polynomials stored in id(LUP)

`LUP_GetFirst           [id(UP)] [id(LUP)]`

Store in id(UP) the first polynomial of id(LUP)

`LUP_Copy               [id(LUP1)] [id(LUP2)]`

Copy a list of univariate polynomials from id(LUP1) to id(LUP2)

`LUP_SturmHabicht       [id(UP)] [id(UP)] [id(LUP)]`

Compute in LUP the Sturm-Habicht sequence relative to the two polynomials

`SC_Open                         ------>  [id(SC)]`

Open an address for a sign condition

`SC_Get                 [id(SC)] -----> SC`

Output a sign condition list

`RST_CountRoots         [id(MT))]  -->  { [int1], [int2] }`

Count the number of real (int1) and complex (int2) roots of a multiplication table MT

`RST_SI                 [id(MT)] [id(PL)] [id(SC)]`

Given a multiplication table MT and a list of polynomials PL store in SC the list of signs of the polynomials of PL at the roots of MT, each sign list having multiplicity equal to the number of roots with that signature.

`RST_UniPol             int [id(MT)] [id(LUP)]`

Given a multiplication table MT and an integer int, find the minimal univariate polynomial in the int-th variable

`RST_GSL                [id(MT)] [id(LUP)]`

Given a multiplication table for a zero-dimensional Groebner Reductor, find an univariate representation of the associated system.

`RST_Open               [id(GR)]  ------>   [id(MT)]`

Open a multiplication table for a given zero-dimensional Groebner reductor

Commented PoSSo-session

```
Nop
% Comments are delimited by the symbol %
%


%
%put the ordering directives into (0) (DRL-ordering)

@
PR_Open
{ {0},
  {4,
        {C, {1, 1, 1, 1}},
        {V, {1, 1, 1, 1}},
        {N, {x, y, z, t}}
  }
}

% declaration of register (1) : List of polynomials
% with respect to (0)

@
PL_Open (0)

% put the system into (1)
%

@
PL_Put (1)
{{1t2,-1t,2z2,2y2,2x2}
,{2tz,2zy,2yx,-1z}
,{2ty,1z2,2zx,-1y}
,{1t,2z,2y,2x,-1}
}

% declaration of register (2) : Groebner Reductor
% with respect to (0)

@
```

```
GR_Open (0)

% Compute the Groebner basis of the system
% (the result is stored into register (2)

@
GR_ComputeBasis (2) (1)  % compute the Groebner Basis
@


% declaration of register (3) : Multiplication table
% associated to (2)

RST_Open (2) % open RST (3)

% declaration of register (4) : List of univariate
% polynomials

@
LUP_Open % open LUP (4)

% Compute the rational univariate representation from (3)
% and put the result in (4)

@
RST_GSL (3) (4) % compute GSL
@


% declaration of register (5) : univariate
% polynomial

UP_Open % open UP (5)

% compute the minimal polynomial of the first variable
% (modulus the ideal) and store the result in (5)

@
RST_UniPol 0 (3) (5)

% (6) register for sign conditions
```

```
%

@
SC_Open

% thom's coding of the roots of (5)
%

@
UP_RAN (5) (6)

% put (6) on the screen
%

@
SC_Get (6)

% compute the number of roots (real and complex) of (5)
%

@
UP_CountRoots (5)

%
% first polynom of (4) in (5)

@
LUP_GetFirst (4) (5)

% compute the number of roots (real and complex) of (5)
%

@
UP_CountRoots (5)

% compute the number of real and complex roots of the system
%

@
RST_CountRoots (3)
```

```
% declaration of register (7) : List of polynomials
% with respect to (0)

@
PL_Open (0)

%put some constrains in the list (7)
%

@
PL_Put (7)
{
{1x}
,{1z}
,{1y}
,{1t}
}

% run the Simultaneous inequalities algorithm
% with respect to (3) and (7) and put the result in
% (6)

@
RST_SI (3) (7) (6)

% put (6) on the screen
%

@
SC_Get (6)
```

B.3.3 RealSolving as a C++-Library

Before going on, be sure that you have a recent version of the PoSSo- library (106 or later). In such case, you have read the general documentation concerning the PoSSo-Library and you are ready to go on.

We suppose, at this stage, that the user is able to compute a Groebner basis using the PoSSo-Library (also these parts of the code will not be described).

For more information about the packages and classes used in the examples proposed in next section, consult the RealSolving C++ reference guide or more generally the PoSSo documentation.

We suppose that a file named si.posso contains extra polynomials for dealing with simultaneous inequalities.

In the following example, we suppose that a Groebner basis (stored in the variable `PL_GroebnerReductor * gr`) has been allready computed.

Notice that the code (excepted the initialisations part that is not described) works independently of the chosen PoSSo-Library configuration.

```
// +++++++++++++++++++++++++++++++++++++++++++++++++++
// Multiplication table
// MultTable.H must be included
// +++++++++++++++++++++++++++++++++++++++++++++++++++

// declare the variable that will contain the
// multiplication table
  PL_MultTable *hmtab;

// Compute the multiplication table associated to the
// Groebner Reductor
  hmtab=new PL_MultTable(gr);


// +++++++++++++++++++++++++++++++++++++++++++++++++++
// Hermite's method
// Hermite.H must be included
// +++++++++++++++++++++++++++++++++++++++++++++++++++

// Initilize and Compute Hermite's quadratic form
  PL_QuadForm hqf(HermiteQF(*hmtab));

// simplify the coefficients of hqf
  hqf.quadraticSimplify();

// reduce Hermite's quadratic form
  PL_Vector rqf(hqf.reduce());

// Compute the number of real roots of the system
  int sig = signatureRQF(rqf);

// Compute the number of complex roots of the system
```

```
  int ran = rankRQF(rqf);

// ++++++++++++++++++++++++++++++++++++++++++++++++++
// Simultaneous Inequalities (Multivariate case)
// SIM.H must be included
// ++++++++++++++++++++++++++++++++++++++++++++++++++

//    read the polynomials to be studied by the Simultaneous
//    Inequalities algorithm

  inle.open("si.data", ios::in);
  PL_PolyList IN;
  inle >> IN;
  inle.close();

// compute the sign conditions
  PL_SIMout *resan=&SIproc(hmtab,IN);

// result on the screen
  cout << *resan << "\n";

// ++++++++++++++++++++++++++++++++++++++++++++++++++
// Rational Univariate Representation
// Chow.H must be included
// ++++++++++++++++++++++++++++++++++++++++++++++++++

  PL_RSList<PL_UniPol> lpp(RUR(hmtab));


// ++++++++++++++++++++++++++++++++++++++++++++++++++
// Sturm-Habicht Sequence (Number of Real Roots)
// StHa.H must be included
// ++++++++++++++++++++++++++++++++++++++++++++++++++

// Get the first polynomial of the R.U.R
   p=lpp.head();

// compute the number of Real roots by the
// Sturm-Habicht algorithm

  int norr= NumberOfRealRoots(p);
```

```
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Simultaneous Inequalities (Univariate case)
// SI.H must be included
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++
// We suppose that Lpol contains the list of
// univariate to study
PL_SIout res = SIproc(P,Lpol);

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Thom's coding
// SI.H must be included
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++

PL_SIout res = RAN(P);

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Sturm Isolation (NOT WITH GENERIC COEFFS)
// Isolation.H must be included
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++

  PL_RSList<PL_RootUniPol> lro;

  lro=IsolateSturm(p);

// put the intervals on the screen

   cout << lro << "\n";

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Uspensky Isolation (NOT WITH GENERIC COEFFS)
// Isolation.H must be included
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++

       lro=IsolateUsp(p);
```

B.3.4 Informations displayed on the screen

By default, RealSolving displays some informations on the screen. They correspond to the different parts of the algorithms. In order to be able to control the execution, we give here some explanations about these informations for the principal commands.

Multiplication table

The multiplication table is the commun prerequisite needed by most of the algorithms in the multivariate case. What you will get on the screen looks like :

```
 Number of Variables : 9
 Test if the Ideal is zero-dimensional
 Compute the quotient basis
 Dimension of the associated Field : 24
 Initialize the table xj*wi
 Initialize the table wi*wj
 Computation of the table xj*wi
 Number of reductions to compute : 139
***********************************************************
***********************************************
 Construction of the multiplication matrix
 Computation of the table wi*wj
***********************************************************
***********************************************************
**********************************
        (Multiplication Table) User Time : 1,190000 sec


                RSS        : 2334720
```

Informations about the system :

```
 Number of Variables : 9
 Test if the Ideal is zero-dimensional
```

Computation of a monomial basis of the algebra (w1,...,wD) :

```
 Compute the quotient basis
 Dimension of the associated Field : 24
```

Initialisation of the Hashing table : compute of all the products of one monomial of the basis by one variable, remove duplicates and store in memory (border of the basis) :

```
 Initialize the table xj*wi
```

Initialisation of the Hashing table : compute of all the products of two monomials of the basis, remove duplicates and store in memory :

```
 Initialize the table wi*wj
```

Computation of reduced expression (border of the basis). When a star appears on the screen, this means that one expression (reduction of a monomial with respect to the basis) has been computed.

```
 Computation of the table xj*wi
 Number of reductions to compute : 139
***********************************************************
***********************************************
```

Computation of reduced expression (multiplication table). When a star appears on the screen, this means that one expression (reduction of a monomial with respect to the basis) has been computed.

```
 Construction of the multiplication matrix
 Computation of the table wi*wj
***********************************************************
***********************************************************
************************************
```

C.P.U. time information

```
        (Multiplication Table) User Time : 1,190000 sec
```

Used memory information (in bytes)

```
               RSS        : 2334720
```

Hermite's algorithm

Hermite's algorithm takes as input a multiplication table, construct a quadratic form an reduce it in order to get its signature (number of real roots) or its rank (number of complex roots). When the quadratic form is of dimension D, the index of each reduction step (`0 1 2 3 4 5 6 ... D`) appears on the screen when the program is at this stage :

```
 Construction of Hermite's quadratic form
 ----------------------------------------
(Construction of Hermite Quadratic Form) User Time : 0,100000 sec
```

```
                RSS        : 2359296

 Fraction-Free quadratic form reduction
 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
(Real Root Counting) User Time : 0,150000 sec


                RSS        : 2457600
```

Rationnal Univariate Representation

The Rational Univariate representation is computed in three steps :

- A modular prediction of a separating element (e.g. finding an injective function)
- The computation (given a separating element)
- The verification of the result : the prime used for the prediction can be bad - in such case, the algorithm changes the prime number and do the whole process again (in general this never appear). However, since the number of primes used is finite, the process ends after a finite number of loops.

```
 Construction of Hermite's quadratic form (modular)
 --------------------------------------------------

 Reduction of Hermite's quadratic form (modular)
 -----------------------------------------------

 Quadratic form reduction
 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
 Number of distincts Roots (modular): 22


 Modular search for separating element
 -------------------------------------
 Degree of the squarefree part : 1
 variable 1 is not separating ... try with another one
 Degree of the squarefree part : 7
 variable 2 is not separating ... try with another one
```

```
 Degree of the squarefree part : 10
 variable 3 is not separating ... try with another one
 Degree of the squarefree part : 10
 variable 4 is not separating ... try with another one
 Degree of the squarefree part : 1
 variable 5 is not separating ... try with another one
 Degree of the squarefree part : 11
 variable 6 is not separating ... try with another one
 Degree of the squarefree part : 16
 variable 7 is not separating ... try with another one
 Degree of the squarefree part : 13
 variable 8 is not separating ... try with another one
 Degree of the squarefree part : 17
 variable 9 is not separating ... try with another one
 No variable is separating ... Try non trivial vectors
 Degree of the squarefree part : 21
 not separating ... try with another one
 Degree of the squarefree part : 22
 I Have found a potential separating element

 Computation of the Rational Univariate Representation
 -----------------------------------------------------
Non Trivial Separating element !
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

 Remove - Multiplicities :
 -----------------------

 Check Rational Univariate Representation
 ----------------------------------------
        (Computation of RUR) User Time : 4,900000 sec


               RSS        : 3186688
```

In this example, the separating polynomial is not trivial (e.g. is not a variable). In such case, the algorithm search an element by incrementing `i` in the expression : X1+i*X2+i**2*X3 ... .

Simultaneous Inequalities

The Simutaneous Inequalities algorithm is used either for computing sign conditions in the multivariate and univariate case and Thom's coding in the univariate case.

```
 Number of distinct real roots : 4
 Using modular Linear Algebra ...
Case 4
{{0}}
{4}
Case 4
{{0,0}}
{4}
Case 6
{{0,0,+},{0,0,-}}
{2,2}
Case 6
{{0,0,+,+},{0,0,-,-}}
{2,2}
Case 4
{{0,0,+,+,0},{0,0,-,-,0}}
{2,2}
Case 6
{{0,0,-,-,0,+},{0,0,+,+,0,-}}
{2,2}
Case 6
{{0,0,-,-,0,+,+},{0,0,+,+,0,-,+},{0,0,-,-,0,+,-},{0,0,+,+,0,-,-}}
{1,1,1,1}
Case 4
{{0,0,-,-,0,+,+,0},{0,0,+,+,0,-,+,0},{0,0,-,-,0,+,-,0},{0,0,+,+,0
,-,-,0}}
{1,1,1,1}
Case 6
{{0,0,-,-,0,+,+,0,+},{0,0,+,+,0,-,+,0,+},{0,0,-,-,0,+,-,0,-},{0,0
,+,+,0,-,-,0,-}}
{1,1,1,1}
```

Each time one polynomial is added to the processus, the programm gives the state of the current sign conditions. for example :

```
Case 6
```

The method for adding a polynomial depends on the sign of it on the studied variety. There are 7 diffenrent possibilities

```
{{0,0,+},{0,0,-}}
```

Represents the sign conditions realized by the 3 first polynomials in the list.

```
{2,2}
```

Represents the number of roots for each sign condition.

The final output is the following

```
{
{<sign condition>, <number of corresponding roots>},
      .
      .
{<sign condition>, <number of corresponding roots>},
}
```

For our example :

```
{
 { {0,0,-,-,0,+,+,0,+} , 1 },
 { {0,0,+,+,0,-,+,0,+} , 1 },
 { {0,0,-,-,0,+,-,0,-} , 1 },
 { {0,0,+,+,0,-,-,0,-} , 1 }
}
```

Sturm-Habicht Sequences

The Sturm-Habicht algorithm looks like the Sturm or the Euclidean algorithm (computation of a remainder sequence). Each time one remainder is computed, a plus appears on the screen. Some extra information are provided for debugging, but however, the algorithm ends when a symbol like

```
         [<number>]
```

Example :

```
+++2+++++++++++++[7]
        (Real Roots Counting) User Time : 0,980000 sec


                RSS        : 1867776
```

Uspensky method

Uspensky method is based on Descarte's rule. It isolates the roots of a given square-free univarate polynomial in a fixed interval with positive bounds. A dichotomy is made constructing intervals that contains one or no root.

```
Number of Primes used (1): 1
        (SquareFree Part) User Time : 0,200000 sec


                RSS        : 1241088
 Bound : 31
[+]***!!**!!!!*************************
[-]!!!!*************************
```

computation of the squarefree part :

```
Number of Primes used (1): 1
        (SquareFree Part) User Time : 0,200000 sec
```

In the case of polynomials with integers coefficients, a multi-modular algorithm is used for computing the square-free part and the program gives informations about this computation.
bound on the module of the roots :

```
 Bound : 31
```

Isolation of the positive roots :

```
[+]***!!**!!!!*************************
```

Each time a root has been isolated, a `!` appears on the screen. The `*` means that an empty interval has been detected.
Isolation of the negative roots :

```
[-]!!!!*************************
```

It may append that some symetries can be detected. In the particular case of a polynomial with only odd powers, the isolation intervals of the negative roots will be deduced without any computation :

```
Number of Primes used (1): 1
        (SquareFree Part) User Time : 0,090000 sec
 Bound : 17
[+]!!**!!!*****
```

Bibliographie

[ABRW94] M. E. Alonso, E. Becker, M.-F. Roy. T. Wörmann: *Zero's, Multiplicities and Idempotents for Zero Dimensional Systems*, to appear in MEGA 94.

[Alp] Alpi: a system for experimenting with Buchberger algorithm.

[AF94] G. Attardi et T. Flagella : *A Customisable Memory Management Framework*. USENIX C++ Conference Prooceedings 1994.

[AMNR92] Alonso M.E., Mora T., Niesi G.F.,Raimondo M.: Local Parametrizations of Space Curves at Singular Points. In Computer Graphics and Mathematics, Springer-Verlag 1992.

[AS88] W. Auzinger and H. J. Stetter: *An Elimination Algorithm for the Computation of all Zeros of a System of Multivariate Polynomial Equations*. Int. Series in Numerical Mathematics **86**, 11–30, Birkhäuser (1988).

[B68] E.H. Bareiss: *Sylvester's identity and multistep integer-preserving gaussian elimination*. Math. Comp., 22:565- 578, 1968.

[Bo88] H. Boehm and M. Weiser : *Garbage Collection in an Uncooperative Environment*. Software Practice and Experience, September 1988, pp. 807-820.

[BWe93] T. Becker and V. Weispfenning : *Groebner basis, a computational approach to comutative algebra*. Graduate texts in mathematics. Springer-Verlag, 1993.

[BCR87] J. Bochnak, M. Coste and M. F. Roy: *Géometrie algébrique réelle*. Ergebnisse der Mathematik. Berlin: Springer-Verlag (1987).

[BCL82] B. Buchberger, G.E. Collins and R.Loos *Computer Algebra Symbolic and Algebraic Computation*, second edition Springer-Verlag. (1982)

[BCRS94] E. Becker, J.P. Cardinal, M.F. Roy and Z. Szafraniec: *Multivariate Bezoutians, Kronecker Symbol and Eisenbud-Levine formula*. To appear in the proceedings of MEGA-94 Conference to be published in the series Progress in Mathematics of Birkhausser (1994).

[BKR96] M. Ben-Or, D. Kozen, J. Reif: *The complexity of elementary algebra and geometry.* J. of Computation and Systems Sciences, 32:251–264, (1986).

[BPR94] S. Basu, R. Pollack, M.-F. Roy : *A new algorithm to find a point in every cell defined by a family of polynomials.* , Quantifier Elimination and Cylrindical Algebraic Decomposition, Texts and Mongraphs in Symbolic Computation, B. Caviness and J. Johnson, Eds. Springer-Verlag Wien, New York1994 to appear.

[Buch85] B. Buchberger : *Groebner basis : An algorithmic method in polynomial ideal theory.* Recent trends in multidimensional systems theory, Reider ed. Bose, (1985).

[BMMT93] E. Becker, M. G. Marinari, T. Mora and C. Traverso: it The shape of the Shape Lemma. Proceedings of ISSAC–94, 129–133, ACM Press (1993).

[BW92] E.Becker, T.Wörmann: *On the Trace Formula for Quadratic Forms.* Proc. RAGSQUAD (1992), to appear in Contemp. Math. (1993)

[BW93] E.Becker, T.Wörmann: *Radical Computations of Zero-dimensional Ideals and Real Root Counting.* to appear in Mathematics and Computers in Simulation.

[Ca88] J.F. Canny: *Some algebraic and geometric computations in PSPACE.* In Proc. Twentieth ACM Symp. on Theory of Computing, 460-467,(1988).

[Car93] J.-P. Cardinal :*Dualité et algorithmes itératifs pour la résolution des systèmes polynômiaux.* Doctoral Thesis. Université de Rennes I (1993).

[CDS] E. Cattani, A. Dickenstein and B.Sturmfels: *Computing Multidimensional Residues.* To appear in the book *Algorithms in Algebraic Geometry and Applications.*

[CGH88] L. Caniglia, A. Galligo, and J. Heintz : *Some new effectivity bounds in computational geometry.* Proceedings of AAECC-6, volume 357 of Lect. Notes in Comp.Sci. Pages 131-152 Springer-Verlag.

[Col75] G. Collins: *Quantifier elimination for real closed fields by cylindric algebraic decomposition.* In Second GI Conference on Automata Theory and Formal Languages. Lecture Notes in Computer Science, vol. 33, pp. 134-183, Springer-Verlag, Berlin (1975).

[CA76] G. Collins and A. Akritas : *Polynomial Real Root Isolation Using Descartes' Rule of Signs.* SYMSAC 1976, p. 272 - 275.

[CR88] M. Coste and M.F. Roy: *Thom's lemma, the coding of real algebraic numbers and the topology of semialgebraic sets*. Journal of Symbolic Computation 5, 121-129 (1988).

[Dal95] J. Dalbec *Geometry and combinatorics of chow forms, thesis*. Cornell University (1995).

[Dit95] Ditrit, O., Petitot, M., Walter, E. *Guaranteed numerical solution of sets of non-linear equations by Hansen's algorithm, implementation and applications.* Proceedings of the PoSSo Workshop on Software - p. 19-34 (1995).

[DST93] J. Davenport, Y.Siret, and E. Tournier. *Calcul formel.* Masson 1993. 2^{e} édition révisée.

[DT92] J. Davenport and B. Trager :*Scratchpad's view of algebra (1) : Basic commutative algebra.* Technical report, Nag Technical Reports, 1992.

[FGLM94] J.C. Faugère, P. Gianni, D. Lazard and T. Mora: *Efficient Computation of Zero-dimensional Gröbner Basis by Change of Ordering.* J. Symbolic Computation 1994.

[Fau94] J.C. Faugère : *Résolution des systèmes d'équations polynômiales*, thèse (1994).

[Fau94c] Faugère, J.C. and Lazard, D. *The combinatorial classes of parallel manipulators* Mechanism and Machine Theory (1994).

[GH91a] M. Giusti and J. Heintz: *La determination des points isoles et de la dimension d'une variete algebrique peut se faire en temps polynomial.* To appear in the Proc. of the International Meeting on Computational Commutative Algebra, 1991.

[GH91b] M. Giusti and J. Heintz: *Algorithmes - disons rapides - pour la décomposition d'une variété algébrique en composantes irréductibles et équidimensionnelles*, Proc. Effective Methods in Algebraic Geometry, MEGA '90 (T. Mora - C. Traverso ed.), Progress in Mathematics **94**, Birkhäuser (1991), 169-193.

[GHMP95] M. Giusti, G-Heintz-Morais-Pardo : *When Polynomial Equation Systems can be "solved" fast?*, Actes de AAECC-11, Lecture Notes in Computer Science 948, Springer Verlag, 1995.

[GM89] P. Gianni and T. Mora: *Algebraic solution of polynomial equations using Gröbner bases.* Proceedings AAECC-5. Lectures Notes in Computer Science **359**, 247-257, Springer-Verlag (1989).

[G94] L. González-Vega.: Using Computer Algebra to solve the Birkhoff Interpolation Problem.

[GLRR89] L. González-Vega, H. Lombardi, T. Recio and M.-F. Roy: *Sturm-Habicht Sequence.* ISSAC-89 Proceedings (Portland), 136–146, ACM-Press (1989).

[GLRR90] L.Gonzalez, H. Lombardi, T.Recio, M.-F. Roy: *Spécialisation de la suite de Sturm et sous-résultants I* Informatique théorique et applications 24 561-588 (1990).

[GLRR94] L.Gonzalez, H. Lombardi, T.Recio, M.-F. Roy: *Spécialisation de la suite de Sturm* . Informatique théorique et applications 28 1-24 (1994) .

[GMT88] P. Gianni, G. Miller and B. Trager: *Decomposition of Algebras.* Lecture Notes in Computer Science 356, 300–308, Springer-Verlag (1988).

[Gro49] W. Groebner : *Modern algebraische Geometrie : die idealtheoritischen Grundlagen.* Springer - Verlag, Wien : Innsbruck, 1949

[GV92] L.González-Vega : *The computation of the radical for a zero dimensional ideal in a polynomial ring through the determination of the trace for its quotient algebra.* Preprint, 1992.

[GRRT93] L.González-Vega, F. Rouillier, M.F Roy, G. Trujillo : *Two algorithms computing the trace quadratic form and their implementation in Axiom.* Preprint,Departemento de mathematicas, estadistica Y computation, Universidad de Cantabria, 1993.

[GTZ88] P. Gianni, B. Trager and G.B. Zacharias : *Groebner basis and primary decomposition of polynomial ideals.* J. Symb. Comp. 6:149-167, 1988.

[GVT95] L.Gonzalez-Vega and G. Trujillo: *Using symmetric functions to describe the solution set of a zero-dimensional ideal*, Proceedings AAECC-11. Lectures Notes in Computer Science **948**, 232-247, Springer-Verlag (1995).

[H48] W. Habicht: *Eine Verallgemeinerung des Sturmschen Wurzelzähl verfahrens.* Comm. Math. Helvetici 21, 99-116 (1948).

[HRS93] J. Heintz, M.F. Roy, P. Solerno : *On the Theoritical and Practical Complexity of the Existential Theory of Reals*, The Computer Journal, Vol 36, Number 5, pages 427-431, 1993.

[Her53] C. Hermite: *Remarques sur le théorème de Sturm.* C. R. Acad. Sci. Paris 36, 52-54 (1853).

[IPC93] C. Innocenti and V. Parenti-Castelli : *Direct kinematics in analytical form of a general geometry 5-4 fully parallel manipulator.* Computational kinematics, J. Angels, P. Kovacs and G. Hommel (eds), Kluwer, 1993, pages 141-152.

[J93] F. Junker: *Uber symmetrische Funktionen von mehreren Reihen von Veränderlichen*, Mathematische Annalen, **43** (1893), 225-270.

[JS92] R.D. Jenks and Robert S. Sutor :*Axiom, the scientific camputation system.* Springer - Verlag, 1992

[JSW87] R.D. Jenks, R.S. Sutor and S. Watt : *Scratchpad II: an abstract datatype system for mathematical computations.* In R. Janssen, editor, Trend in computer algebra, Proc Int. Symp. pages 12-37, bad Neuenahr, May 1987. Springer. Lecture Notes in Computer Science 296.

[KV95] J. Kovacevic and M. Vetterli : *Nonseparable Two- and Three-Dimensional Wavelets*, IEEE Transactions on Signal Processing, May 1995 Volume 43, P. 1269-1272.

[Knu81] D.E. Knuth :*The art of computer programming*, volume 2.Addison Wesley, 2 edition, 1981.

[Knu82] D.E. Knuth :*Fundamental algorithms. The art of computer programming*, volume 1. Addison Wesley, 2 edition, 1982.

[Lak91] Y. N. Lakshman and D. Lazard: *On the Complexity of Zero–dimensional Algebraic Systems.* Effective Methods in Algebraic Geometry. Progress in Mathematics **94**, 217–225, Birkhauser (1991).

[Laz92] D. Lazard : *Solving Zero - dimensional algebraic systems.* J. Symb. Comp. 13:117-132, 1992.

[Laz92b] D. Lazard :*Stewart platforms and Groebner basis.* In V. Parenti-Castelli and Lenaric J. editors, Proceedings of the third international workshop on advances in robot kinematics, pages 136-142, Ferrare, Italie, 1992.

[LCO] D. Cox, J. Little, D. O'Shea : *Ideals, varieties, and algorithms an introduction to computational algebraic geometry and commutative algebra.* Undergraduate texts in mathematics, Springer-Verlag New York-Berlin-Paris 1992.

[LJ85] M. Lejeune- Jalabert. *Effectivité des calculs polynômiaux.* Cours de DEA , Institut Fourier, 84-85.

[Loo82] R. Loos: *Generalized polynomial remainder sequences* In Computer algebra, symbolic and algebraic computation. Springer Verlag, Berlin (1982).

[Mac79] I. G. Macdonald: *Symmetric functions and Hall polynomials.* Oxford University Press (1979).

[Mer90] J. P. Merlet: *Les robots parallèles* Hermès - Paris (1990).

[Mig89] M. Mignotte : *Mathematiques pour le calcul formel.* Presses Universitaires de France. 1989.

[Moel93] M. Moeller: *Systems of algebraic equations solved by means of endomorphisms.* In pro. of AAECC-10. Volume 673 of Lect. Notes in Comp. Scien. pages 43-56, 1993.

[Mou] B. Mourrain : *The 40 generic positions of a parallel robot.* In M. Bronstein, editor, Proceedings of ISSAC'93, ACM press, pages 173-182, Kiev (Ukraine), July 1993.

[Nie93] M. Niermann : *Computing Groebner basis of arbitrary ideals by change of term orders.* Preprint 1993.

[Ped91] P.Pedersen *Counting Real Zeros, thesis.* Courant institute, New York University, (1991).

[PR93] R. Pollack, M..-F. Roy: *On the number of cells defined by a set of polynomials*, C. R. Acad. Sci. Paris, 316:573–577, (1993).

[PRS93] P.Pedersen, M.-F.Roy, A.Szpirglas: *Counting Real Zeros in the Multivariate Case*, Computational Algebraic Geometry, Frédéric Eyssette, André Galligo (editors), 203-223 (1993), Birkhäuser.

[Re92] J. Renegar: *On the computational complexity and geometry of the first-order theory of the reals*, parts I, II and III. Journal of Symbolic Computation, 13(3):255–352, 1992.

[Rio] R. Rioboo : *doctoral thésis.* Université de Paris VI.

[Ron90] L. Ronyai: *Computing the Structure of Finite Algebras. Journal of Symbolic Computation*, 9, 3, 127–145 (1990).

[R95a] F. Rouillier : *Formules de Bareiss et réduction de formes quadratiques.*, C. R. Acad. Sci. Paris, 320: 1273-78, (1995).

[R95b] F. Rouillier : *Real Root counting for some Robotics Problems.*, Computational Kinematics 95, Kluwer Academic Publishers, Solid Mechanics and its Applications, 40: 73-82. (1995).

[R95c] F. Rouillier : *PoSSo-RealSolving (zero dimensional systems of polynomials)*, Proceedings of the PoSSo-workshop. Paris 1995.

[RRS94] F. Rouillier, M.F.Roy, A. Szpirglas: *Multivariate symmetric functions and polynomial system solving.* Preprint. 1994.

[RT95] F. Rouillier and G. Trujillo : *PoSSo-RealSolving (Simultaneous Inequalities*, Proceedings of the PoSSo-workshop. Paris 1995.

[Roy1] M.F. Roy : *Basic algorithms in real algebraic geometry: from Sturm theorem to the existential theory of reals*, Lectures on Real Geometry in memoriam of Mario Raimondo,de Gruyter Expositions in Mathematics

[RS90] M.-F. Roy, A. Szpirglas: *Complexity of computations with real algebraic numbers*, J. of Symbolic Comput., 10:39–51, (1990).

[Str86] B. Stroustrup :*The C++ programming language.* Addison-Wesley. 1986.

[Stu35] C. Sturm: *Mémoire sur la résolution des équations numériques.* Inst. France Sc.Math. Phys.6 (1835).

[Syl53] J. J. Sylvester: *On a theory of syzygetic relations of two rational integral functions, comprising an application to the theory of Sturm's function.* Trans. Roy. Soc. London (1853).

[TD89] C. Traverso and L. Donati : *Experimenting the Groebner basis algorithm with the Alpi system.* In ACM Press, editor. Proc. of the 1989 International Symposium on Symbolic and algebraic computation. ISSAC'89, 1989.

[TM94] C. Traverso and T. Mora :*Linear Groebner methods and "natural" representation of algebraic numbers.* Preprint. 1994

[Tra88] C. Traverso : *Groebner Trace algorithm.* In Issac'88, Roma, volume 8, pages 125-138. Lect. Notes in Comp. Sci., 1988.

[Us48] J. Uspensky : *Theory of Equations.* McGraw-Hill, New York, (1948)

[Vi36] M. Vincent : *Sur la Résolution des Equations Numériques.* Journal de mathématiques pures et appliquées 1, p 341-372 (1836).

[Yoko92] K. Yokoyama, M. Noro and T. Takeshima: *Solutions of Systems of Algebraic Equations and Linear Maps on Residue Class Rings.* Journal of Symbolic Computation **14**, 399–417 (1992).