Jean-Charles Faugère

Calcul efficace des bases de Gröbner et Applications.

SPIN Springer's internal project number, if known

Linz Series – Monograph
(Editorials B. Buchberger)

February 9, 2007

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Dedicace Springer Verlag

Préface

This is the preface. The preface does not appear in the table of contents.

Paris, Faugère Jean-Charles

Feb 2007

Contents

List of Figures

List of Tables

Part I

Bases de Gröbner: algorithmes efficaces

1. Calcul Formel et Calcul Scientifique

Le problème abordé dans ce livre est le problème fondamental de la résolution par des méthodes du Calcul Formel des systèmes d'équations polynomiales. Le Calcul Formel peut être défini comme étant la manipulation par l'ordinateur d'expressions mathématiques. On peut estimer que la résolution des systèmes polynomiaux pourrait devenir l'une des principales application du Calcul Formel ((Davenport J.H. and Siret Y. and Tournier E., 1993)). L'un des objets mathématiques fondamentaux qui sera utilisé dans cet ouvrage est *la notion de base de Gröbner*. C'est Bruno Buchberger qui introduisit la définition de base de Gröbner et qui proposa un algorithme (l'algorithme de Buchberger) pour calculer explicitement cet objet (Buchberger B., 1965; Buchberger B., 1970; Buchberger B., 1976; Buchberger, 1987). Le principal objectif de ce livre est d'une part la description de plusieurs algorithmes permettant de rendre le calcul des bases de Gröbner beaucoup plus efficace en pratique et d'autre part de montrer des applications (Cryptologie, Robotique, Théorie du signal, ...) où ces techniques ont été utilisées. Ces applications montrent aussi que la classe de problèmes qu'on pouvait traiter par les méthodes connues du Calcul Formel n'était pas suffisante et que donc l'amélioration de l'efficacité des calculs (en particulier des bases de Gröbner) est une condition nécessaire pour être en mesure de répondre aux questions posées dans ces disciplines. On verra aussi à travers ces applications qu'il est souvent nécessaire d'adapter la mise en équations du problème pour tenir compte des spécificités algébriques de ces méthodes. Ces algorithmes sont compétitifs en termes de vitesse et de qualité du résultat par rapport aux meilleurs algorithmes numériques; on peut donc résolument inscrire toutes ces méthodes dans le cadre du Calcul Scientifique.

Le problème fondamental est de chercher les solutions d'un système d'*équations algébriques*:

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ \cdots \\ f_m(x_1, \ldots, x_n) = 0 \end{cases} \tag{1.1}$$

où les polynômes $f_1, \ldots, f_m$ sont des polynômes en les variables $x_1, \ldots, x_n$ et à coefficients dans un corps $\mathbb{K}$ dans lequel on sait calculer (par exemple $\mathbb{Q}$ ou $\mathbb{F}_p$). Lorsque $\mathbb{K} = \mathbb{F}_p$ les méthodes proposées sont souvent sans équivalent.

1.1 Que veut dire résoudre un système algébrique ?

On sait que pour résoudre une équation algébrique en une variable, (le cas $m = n = 1$ dans le système (1.1))

$$f(x) = 0 \text{ où } f \in \mathbb{K}[x]$$

il est illusoire (et même nuisible) de chercher des formes closes des solutions; en effet, dès que le degré de l'équation est supérieur à 5 ($\deg(f) \geq 5$) il n'existe pas, en général, de formules exprimant les racines de f par radicaux (voir par exemple (Stewart, 1989)). La question se pose alors de ce que veut dire résoudre une équation et donc a fortiori un système d'équations algébriques ?

Dans la suite de cette introduction, nous avons cherché à présenter de manière volontairement simplifiée les algorithmes et les objets mathématiques qui vont être mis en oeuvre pour répondre à cette question. La plupart des algorithmes présentés sont disponibles dans plusieurs systèmes de Calcul Formel; dans le livre on utilise le logiciel généraliste de Calcul Formel Maple (Char B. and Geddes K. and Gonnet G. and Leong B. and Monagan M. and Watt S., 1991) couplé à des bibliothèques spécialisées FGb (pour le calcul des bases de Gröbner) et RS (logiciel de F. Rouillier pour tout ce qui touche aux racines réelles). Ces librairies sont disponibles sur le Web ou dans les versions récentes de Maple permettant ainsi de résoudre immédiatement des problèmes concrets et de vérifier la plupart des calculs décrit dans le livre.

Considérons le système algébrique suivant (voir la définition générale de l'exemple Cyclic n en 2.8.3 page 43)

$$C_3 \begin{cases} x_1 + x_2 + x_3 = 0 \\ x_1x_2 + x_2x_3 + x_3x_1 = 0 \\ x_1x_2x_3 = 1 \end{cases} \tag{1.2}$$

Il est clair que ce système admet comme solutions complexes:

$$\left\{ x_1 = e^{\frac{2}{3}\imath\pi\,(k+2\,j)}, x_2 = e^{\frac{2}{3}\imath\pi\,(k+j)}, x_3 = e^{\frac{2}{3}\imath\pi\,k} \right\} \quad \text{avec } k = 0, 1, 2 \; j = 1, 2 \tag{1.3}$$

En premier lieu et contrairement à un système d'*équations linéaires*, les systèmes algébriques admettent en général *plusieurs* solutions (6 dans le cas présent). De plus, même si les coefficients du système initial sont des entiers ou des rationnels (ce qui est le cas ici) les solutions ne sont pas en général des rationnels mais des nombres complexes (plus généralement les solutions sont des éléments de la clôture algébrique du corps de départ). De même que l'expression à l'aide de radicaux des solutions d'une équations algébrique de degré > 5 n'est en général pas possible, il faut noter que l'expression des

solutions d'un système algébrique ne peut se faire explicitement. L'exemple Cyclic 3 est donc atypique en ce sens. Dans la pratique on devra se contenter d'une expressions *formelle* ou *approchée* des solutions; toutefois on s'attachera à garantir ou certifier tous les résultats.

Si on revient au cas d'un polynôme $f(x) = 0$ on dira que résoudre dans $\mathbb{R}$ une telle équation c'est être capable, pour un $\varepsilon > 0$ fixé, de retourner une union d'intervalles $\cup_{i=1}^{r}[l_i, r_i]$ telle que:

1. r est le nombre de racines réelles (éventuellement avec multiplicités)
2. $[l_i, r_i]$ contient une et une seule racine réelle
3. l_i et r_i sont des rationnels
4. $0 \leq r_i - l_i \leq \varepsilon$.

Résoudre une équation univariée dans un corps fini $\mathbb{F}_p$ est conceptuellement plus simple puisqu'il suffit de faire le p.g.c.d. de f et de $x^p - x$ puis de factoriser le polynôme résultant pour trouver toutes les solutions dans le corps $\mathbb{F}_p$; si p est petit on peut même recherche de manière exhaustive toutes les solutions dans $\mathbb{F}_p$. L'étude et la description de ces algorithmes n'est pas l'objet du présent volume (voir toutefois l'annexe 21 pour le cas réel).

Pour expliciter la notion de résolution formelle dans le cas d'un système polynomial on revient à l'exemple (1.2); dans ce cas une façon d'exprimer symboliquement toutes les solutions est de d'écrire:

$$\text{Solutions Cyclic 3} = Sol_1 \cup Sol_2 \cup Sol_3 \tag{1.4}$$

où

$$Sol_1 \begin{cases} x_2^2 + x_2 + 1 = 0 \\ x_3 = 1 \\ x_1 = -x_2 - 1 \end{cases} \quad Sol_2 \begin{cases} x_3^2 + x_3 + 1 = 0 \\ x_2 = -x_3 - 1 \\ x_1 = 1 \end{cases} \quad Sol_3 \begin{cases} x_3^2 + x_3 + 1 = 0 \\ x_2 = 1 \\ x_1 = -x_3 - 1 \end{cases}$$

En d'autres termes, on *décompose* l'ensemble des solutions comme une *union* de trois sous-ensembles; chaque sous-ensemble de solutions est représenté formellement par une liste d'équations dont l'une d'entre elles est un polynôme en une variable. On verra par la suite que chacun des systèmes Sol_i est aussi une base de Gröbner (pour l'ordre lexicographique). Ainsi pour retrouver, par exemple, la valeur numérique des solutions de Sol_1 il suffit de résoudre l'équation univariée $x_2^2 + x_2 + 1 = 0$ et de reporter dans les deux autres équations les valeurs trouvées. De façon intuitive, la solution formelle d'un système algébrique consiste donc à *réécrire* le système initial en un autre système équivalent plus simple, ou en une liste de systèmes algébriques plus simples dont la réunion des solutions est l'ensemble des solutions du système de départ. Plus exactement on cherche à se ramener au cas simple suivant:

$$\begin{cases} P_n(x_n) = 0, \\ x_{n-1} = P_{n-1}(x_n) \\ \cdots \\ x_1 = P_1(x_n) \end{cases}$$

On verra par la suite qu'une base de Gröbner pour l'ordre lexicographique d'un système ayant autant d'équations que d'inconnues et le plus souvent de cette forme. Parfois il est impossible de se ramener à cette forme et on calculera alors une *RUR (Représentation Rationnelle Univariée (Rouillier, 1999))*:

$$\begin{cases} P_n(T) = 0, \\ x_n = \frac{N_{n-1}(T)}{D_{n-1}(T)} \\ \cdots \\ x_1 = \frac{N_1(T)}{D_1(T)} \end{cases}$$

Cette représentation nécessite l'adjonction d'une nouvelle variable T. On donnera dans les sections 2.2.1 et 2.2.2 des définitions plus précises de ces notions de décomposition. Les formes de représentation formelle des solutions que nous calculerons seront les *bases de Gröbner,*les *ensembles triangulaires* et la forme *RUR.*

1.1.1 Un exemple provenant d'une application.

La classe des problèmes qui peuvent être abordé par les méthodes algébriques est plus vaste qu'il n'y parait: ainsi bon nombre de problèmes faisant intervenir les fonctions trigonométriques, des racines carrés, des fractions rationnelles, ... peuvent se ramener à une formulation algébrique après changement de variables (par exemple $c = cos(\alpha), s = sin(\alpha)$ et en ajoutant l'équation $c^2 + s^2 - 1 = 0$). Souvent le problème à résoudre est présenté non pas comme un problème algébrique mais comme un problème *d'optimisation globale* $F(x)$ avec contraintes non linéaires $G(x)$:

$$\min \{F(x) \mid x = (x_1, \ldots, x_n) \in \mathbb{C}^n \text{ et } G(x) = 0\}$$

On demandait, par exemple, de maximiser la fonction de Breguet (voir (J.C., 2001))

Endurance(S, L) donnant l'endurance d'un avion en fonction de la surface S et de la longueur L de l'aile. La fonction d'endurance Endurance(S, L) est donné par la formule suivante (voir l'annexe 20.6 p. 319 pour la définition complète):

$$\left(\frac{1}{\sqrt{\begin{array}{l} Kr\ (k_f + k_e + k_t + k_c u)\ \sqrt{S} L^{3/2} + \frac{\sqrt{2}\mu_{\mathrm{mot}}\, g^{3/2} C_{x0}\, \sqrt{S}}{\nu_{\mathrm{prop}}\, \sqrt{\rho} {C_z}^{3/2} \sqrt{m_{\mathrm{init}}}} + \\ \frac{\sqrt{2}\mu_{\mathrm{mot}}\, g^{3/2} Kr\, C_{x0}\, S L^{3/2}}{\nu_{\mathrm{prop}}\, \sqrt{\rho} {C_z}^{3/2} \sqrt{m_{\mathrm{init}}}} + \frac{ms\, S}{m_{\mathrm{init}}} + \frac{\sqrt{2}\mu_{\mathrm{mot}}\, g^{3/2} \sqrt{C_z} Kr\, S \sqrt{L}}{\nu_{\mathrm{prop}}\, \sqrt{\rho} \sqrt{m_{\mathrm{init}}} \pi} + \\ \frac{\sqrt{2}\mu_{\mathrm{mot}}\, g^{3/2} \sqrt{C_z} \sqrt{S}}{\nu_{\mathrm{prop}}\, \sqrt{\rho} \sqrt{m_{\mathrm{init}}} \pi\, L} + k_f + k_e + k_t + k_c u \end{array}}} - 1 \right) \times \sqrt{S} \left(1/2\, \frac{C_{x0}\, C_{sp}\, g^{3/2} \sqrt{m_{\mathrm{init}}} \sqrt{2}}{\nu_{\mathrm{prop}}\, \sqrt{\rho} {C_z}^{3/2}} + 1/2\, \frac{\sqrt{C_z} C_{sp}\, g^{3/2} \sqrt{m_{\mathrm{init}}} \sqrt{2}}{\pi\, \nu_{\mathrm{prop}}\, \sqrt{\rho} L} \right)^{-1}$$

La première transformation à faire consiste à introduire les nouvelles variables s et l et les équations $S - s^2 = 0$, $L - l^2 = 0$ de façon à se ramener à une expression algébrique du problème. Dans un deuxième temps pour résoudre le problème d'optimisation on se ramène au système algébrique:

$$\begin{cases} \frac{\partial \mathrm{Endurance}(S,L)}{\partial S} = 0 \\ \frac{\partial \mathrm{Endurance}(S,L)}{\partial L} = 0 \end{cases}$$

On calcule la base de Gröbner (moins de 1 seconde) pour un ordre d'élimination: la forme de cette base de Gröbner est

$$\begin{cases} P(l) = l^{51} + \frac{1040042212527722361792468285452351668713877376 17}{126056540094480407802405345058366940423 96} l^{50} + \cdots \\ s = H(l) \end{cases}$$

Le système admet donc 51 racines complexes et 11 racines réelles:

$$[[-485.2286842, -485.2286842], [-6.665573441, -6.665573440], \ldots] .$$

Pour chacune de ces valeurs 11 il est facile de calculer l'endurance; on sélectionne ensuite la plus grande ce qui donne le vrai maximum global Endurance(S, L) =245.9661472. À noter qu'il existe un autre extremum (donc local_ qui donne une valeur quasi identique à l'endurance maximale:

$$\begin{array}{l} L = 44.42986929,\ S = 132.8760537 \longrightarrow \mathrm{Endurance}(S, L) = 245.9625354 \\ L = 44.53253798,\ S = 131.7333942 \longrightarrow \mathrm{Endurance}(S, L) = 245.9661472. \end{array}$$

Remarque: on pourrait aussi introduire une nouvelle variable E et une équation $E - \mathrm{Endurance}(S, L) = 0$ puis calculer une base de Gröbner donnant directement un polynôme en E (toujours de degré 51). La plus grande racine réelle de ce polynôme donne directement le maximum global.

1.2 Bases de Gröbner efficace

1.2.1 Le besoin d'efficacité

Pour traiter des applications rendre les calculs de bases de Gröbner plus efficace de plusieurs ordres de grandeur est une nécessité. En effet, considérons une application comme la Cryptologie et la taille minimale des systèmes polynomiaux qu'on devra résoudre: l'évaluation de la robustesse de beaucoup de cryptosystèmes peut se ramener à l'évaluation de la difficulté de résoudre un système d'équations booléennes (c'est à dire des équations polynomiales en les variables $x_1, \ldots, x_n$ à coefficients dans le corps fini $\mathbb{F}_2$). Comme il est toujours possible de procéder à une recherche exhaustive des solutions d'un tel système on connaît une borne inférieure sur le nombre de variables:

$$\text{Complexité recherche exhaustive} \approx 2^n \ > 2^{80}$$

(on admet généralement que 2^{80} opérations est un nombre d'opérations qu'aucun ordinateur ne sera capable d'effectué d'ici 20 ans). Ceci implique donc que le nombre de variables doit être $n > 80$. Or cette taille de système est complètement hors de portée des algorithmes et des implantations classiques des calculs des bases de Gröbner ! Prenons l'exemple du cryptosystème HFE (traité en détail dans le chapitre 8) dont la clé publique est constituée de n polynômes quadratiques dense à coefficients dans $\mathbb{F}_2$; dans le tableau suivant on mesure la valeur maximale de n que l'on peut traiter en calculant une base de Gröbner de la clé publique lorsqu'on limite le temps de calcul à 10 minutes puis à 2 heures sur un PC Pentium 4:

Algorithme	Algorithme de Buchberger					
temps \ n	Maple	slimGb	Macaulay 2	Singular	F_4	F_5
après 10m	12	17	18	19	22	35
après 2h	14	19	20	21	28	45

Table 1.1. Valeur maximale de n atteinte par le calcul d'une base de Gröbner

Ce table nous permet de constater:

1. Pour un même un algorithme (Buchberger) la qualité est l'implantation est cruciale pour la vitesse d'exécution: il suffit de comparer l'implantation en Maple (langage interprété ne disposant pas de type informatique pour les polynômes multivariés) et les implantations en C (Singular, Macaulay2).
2. Augmenter le temps de calculs de 10 minutes à 2 heures augmente seulement la valeur maximale de n de 2 (ne dépend pas des implantations).

3. Pour les implantations de l'algorithme de Buchberger le calcul d'une base de Gröbner semble moins bon que la recherche exhaustive. De plus il semble hasardeux de vouloir extrapoler la complexité (polynomiale ou exponentielle) à partir des données expérimentales.
4. Une taille réaliste d'un point de vue Cryptologie ($n = 80$) semble complètement hors de portée de l'algorithmique classique. En revanche pour les algorithmes comme F_4 et F_5 présentés dans les chapitres 4 et 5 on observe un comportement bien meilleur lors du passage à l'échelle. Des exemples de taille $n \geq 80$ peuvent être traité avec ces nouveaux algorithmes.

Plus généralement l'ensemble des applications présentées dans les parties II et III illustrent le fait que sans des algorithmes plus beaucoup plus efficaces il aurait été impossible de faire aucun calcul significatif dans ces domaines d'application. Les applications de la partie III montrent aussi que le calcul de base de Gröbner n'est qu'une première étape dans la chaîne de résolution et qu'il est donc nécessaire de disposer d'outils complémentaires pour répondre aux questions posées (en particulier des outils pour extraire les racines réelles).

1.2.2 Amélioration des algorithmes

Afin d'améliorer les performances on pourrait optimiser l'algorithme de Buchberger dans plusieurs directions:

1. optimisation des choix de calcul des paires critiques (stratégie de calcul voir par exemple (Giovini A. and Mora T. and Niesi G. and Robbiano L. and Traverso C., 1991))
2. utilisation d'un calcul modulo un petit nombre premier pour éviter les paires critiques inutiles (Traverso C., 1998).
3. optimisation de l'implantation: implantation dans un langage de bas niveau (C, assembleur) et créer des types adaptés (structure pour représenter efficacement les polynômes).
4. parallélisation des calculs (utilisation de clusters) (Attardi G. and Traverso C., 1994; Faugère, 1994).

Toutefois toutes ces méthodes ont vite atteints leur limite:

1. le choix des paires critiques est complètement heuristique (même pour les meilleures stratégies comme la stratégie du sucre) et dépendent fortement de la nature des exemples.
2. si l'utilisation d'un calcul modulo p donne de bons résultats lorsque que le coût de l'arithmétique est élevé (c'est le cas par exemple dans $\mathbb{Q}$) il faut néanmoins observer qu'on n'évite pas un calcul coûteux a posteriori si on veut un résultat certifié; de plus cette méthode ne s'applique pas lorsqu'on cherche les solutions dans un corps fini (le cas extrême étant le cas du corps $\mathbb{F}_2$).

3. les optimisations dans un langage de bas niveau permettent de gagner effectivement un facteur important mais les meilleures implantations en langage C (Singular, Macaulay2, Gb, ...) ne semblent plus progresser significativement (autrement que d'un facteur constant).
4. Les tentatives de parallélisation de l'algorithme de Buchberger sont paradoxales puisqu'on arrivent à la conclusion que la meilleure stratégie est de suivre pas à pas le déroulement de l'algorithme de Buchberger séquentiel

1.2.3 Nouveaux algorithmes

Afin d'améliorer radicalement l'algorithme de Buchberger nous avons été amené à faire trois modifications majeures:

1. Utiliser un algorithme de calcul de base de Gröbner général uniquement pour l'ordre le moins coûteux (ordre du degré DRL): introduction d'algorithmes pour calculer une base de Gröbner pour n'importe quel ordre à partir d'une base de Gröbner donné. On nomme ces algorithmes (FGLM, LLL du chapitre 3) algorithmes de changement d'ordre. La structure de ces algorithmes est radicalement différente de l'algorithme de Buchberger et utilise intensivement l'algèbre linéaire. Ainsi lorsque l'idéal est de dimension 0 l'algorithme FGLM comment par calculer le polynôme minimal d'une certaine matrice (matrice de multiplication par une variable).
2. Unification de l'algorithme de Buchberger et la méthode de Macaulay dans l'algorithme F_4 (chapitre 4) permettant d'utiliser pleinement des outils d'algèbre linéaire (inversion par blocs d'une matrice par exemple) tout en gardant le test d'arrêt de l'algorithme de Buchberger (notion de paire critique). Par opposition avec les matrices de Macaulay l'algorithme F_4 permet aussi de construire des matrices beaucoup plus petites; de plus les matrices issues de F_4 ont une structure triangulaires par blocs; elles sont toutefois plus denses que les matrices de Macaulay.
3. Remplacer les critères de Buchberger par un nouveau critère permettant d'éviter *toutes* les réductions à zéro (paires critiques inutiles) lorsque le système initial est non régulier (ou semi-régulier dans le cas surdéterminé). La solution retenue dans l'algorithme F_5 (chapitre 5) implique aussi un changement profond puisqu'il est nécessaire de calculer, au moins partiellement, *incrémentalement* les bases de Gröbner. En comparaison avec la méthode de Macaulay les matrices générées par l'algorithme F_5 sont de tailles optimales puisque le rang de ces matrices est égale au nombre de lignes (pour des systèmes réguliers).

Ces nouveaux algorithmes permettent effectivement de gagner plusieurs ordres de grandeurs (voir par exemple la table 1.2.1). L'Implémentation de ces algorithmes, en particulier pour F_4, n'est aisée et demande une refonte complète de l'implantation par rapport à l'implantation de l'algorithme de

Buchberger. Par exemple l'implémentation dans un langage de bas niveau représente environ 242000 lignes de code; les tailles des matrices générées par les algorithmes sont souvent gigantesques: un gestionnaire mémoire dédié a ainsi été crée et un procédé de compression des matrices a aussi été ajouté (par exemple la résolution du challenge 1 de HFE par l'algorithme F_5 engendre plusieurs matrices ayant 1.6 millions de colonnes).

1.3 Schéma global de résolution

Comme dans l'exemple de la formule de Breguet et dans bon nombre de problèmes physiques on ne cherche que les solutions réelles (ou réelles positives): dans ce cas on commence par appliquer les techniques de la première partie de ce livre (calcul de bases de Gröbner puis calcul d'élimination) puis éventuellement décomposition en ensembles triangulaires (voir l'annexe 20); dans un deuxième temps on cherche les solutions réelles (ou réelles positives) et pour cela on calcule une forme RUR du système (voir 2.9.4) puis on applique des méthodes d'isolation des racines. On arrive ainsi au schéma global (voir figure 1.1) de résolution (applicable au moins dans le cas zéro dimensionnel, c'est à dire quand le *nombre de solutions est fini*):

Dans la suite on désigne par "Zéro-Dim Solve" l'ensemble de ces outils et ce schéma de résolution.

1.4 Mise en équations.

De façon générale le problème de la *mise en équation* d'un problème physique ou mathématique est une opération cruciale et difficile à laquelle il convient d'apporter un soin particulier: choix des bonnes variables, utilisation des symétries du problème, élimination des solutions parasites ou n'ayant pas de sens physique. Le simple de choisir les bonnes variables permet parfois de rendre facile des calculs qui étaient très difficiles voir impossibles et ce sans changer d'algorithme de calcul ou d'implantation !Le problème de la mise en équations est abordé dans ce livre au travers des applications très différentes dans les parties II (applications ayant comme origines communes la Cryptologie) et III (autres applications). Il faut encore insister sur le fait que pour la plupart des applications de la partie III le calcul des bases de Gröbner ne constitue nullement une fin en soit: il faut ensuite extraire les racines réelle ou tracer des courbes pour traiter effectivement ces applications. Un autre cas extrême est le cas de l'algorithme "standard" de chiffrement AES: il est possible de mettre en équation ce problème et d'ordonner les variables pour que le système soit directement une base de Gröbner (voir (J.Buchmann *et al.* , 2006)). De cette façon on exprime simplement les solutions dans la clôture algébrique de $\mathbb{F}_{128}$; pour calculer les vrais solutions (dans $\mathbb{F}_{128}$) il faut effectuer un changement d'ordre qui rend le calcul exponentiel !

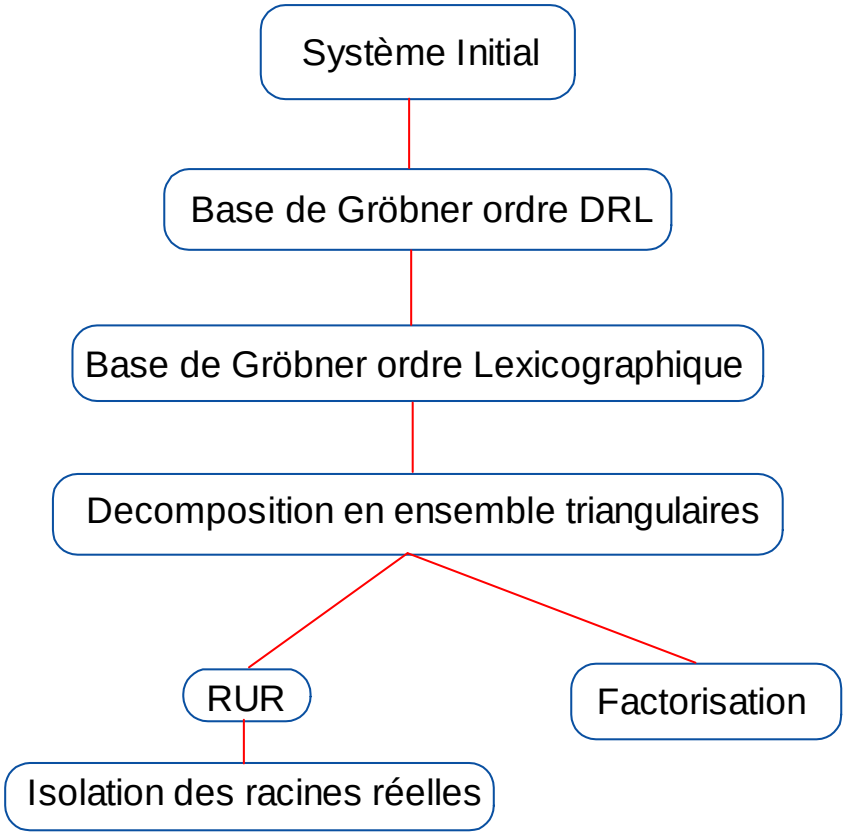


Fig. 1.1. Stratégie générale de résolution des systèmes zéro dimensionnels.

dans une moindre mesure les applications suivantes sont utilisées comme exemple dans le livre mais ne font pas l'objet d'un traitement complet:

Problème	Corps/ Solutions/ CPU	Chap.	Spécificité	Utilise
HFE	$\mathbb{F}_2$ $\mathbb{F}_2$ Jours	Chap 8 Page 155	Système sur-déterminés Équations de corps $x^2 = x$	Gröbner
C*	$\mathbb{F}_{2^5}$ $\mathbb{F}_{2^5}$ Jours	Chap 18.2 Page 295	Solutions dans $\mathbb{F}_{32}$ Spécialisation.	Gröbner Chgt d'ordre
IP	$\mathbb{F}_q$ $\mathbb{F}_q$ Jours	Chap 9 Page 167	Utilisation de la structure Système sur-déterminés	Gröbner
2R-	$\mathbb{F}_q$ $\mathbb{F}_q$ Jours	Chap 10 Page 181	Utilisation idéal des dérivées partielles.	Gröbner Quotient $I : f$
Codes Correcteurs	$\mathbb{F}_q$ $\mathbb{F}_q$ 10^{-5} sec	Chap 13 Page 231	Utilisation des symétries Mise en équations.	Générations d'un programme C pour calculer la Gb.
Jacobienne	$\mathbb{F}_q$ $q \approx 2^{50}$ NA 10^{-6} sec	Chap 11 Page 195	Générations de formules optimisées	FGLM Quotient $I : f$ thm structure Convolution rapide
Biologie	$\mathbb{Q}$ $\mathbb{C}$ Heure	Chap 12 Page 215	Utilisation des symétries (S_n)	Gröbner
Robotique	$\mathbb{Q}$ $\mathbb{R}$ Seconde	Chap 14 Page 233	Mise en équations	Utilisations des équations linéaires
Ridges	$\mathbb{Q}$ $\mathbb{R}$ Minute	Chap 15 Page 235	Système sur-déterminés Élimination dans $\mathbb{Q}[u, v]$	Changement d'ordre Radical Décompositions
Ondelettes	$\mathbb{Q}$ $\mathbb{C}^\dagger$ Minute	Chap 16 Page 237	Changement de vars $\dim(I) > 0$	Forme Normale Gröbner Ens. Triangulaires
Filtres Hyper-fréquences	$\mathbb{Q}$ $\mathbb{R}$ Jours	Chap 17 Page 271	Rendre le système très sur-déterminé.	Zéro-Dim Solve

Table 1.2. Classification des applications

Problème	Corps/ Solutions/ CPU	Chap.	Spécificité	Utilise
Katsura	$\mathbb{Q}$ $\mathbb{R}$ Jours	Chap 2.9.4,3.1.10 Page 48,62	Solutions dans $[0, 1]$	Zéro-Dim Solve
Flurry	$\mathbb{F}_{2^5}$ $\mathbb{F}_{2^5}$ Jours	Chap 3.1.10 Page 63	Choix de l'ordre	Gröbner FGLM
NTRU	$\mathbb{F}_2$ $\mathbb{F}_2$ NA	Chap 6 Page 123	Évaluation de l'ajout de nouvelles équations	Résultats de complexité

Table 1.3. Classification des autres applications

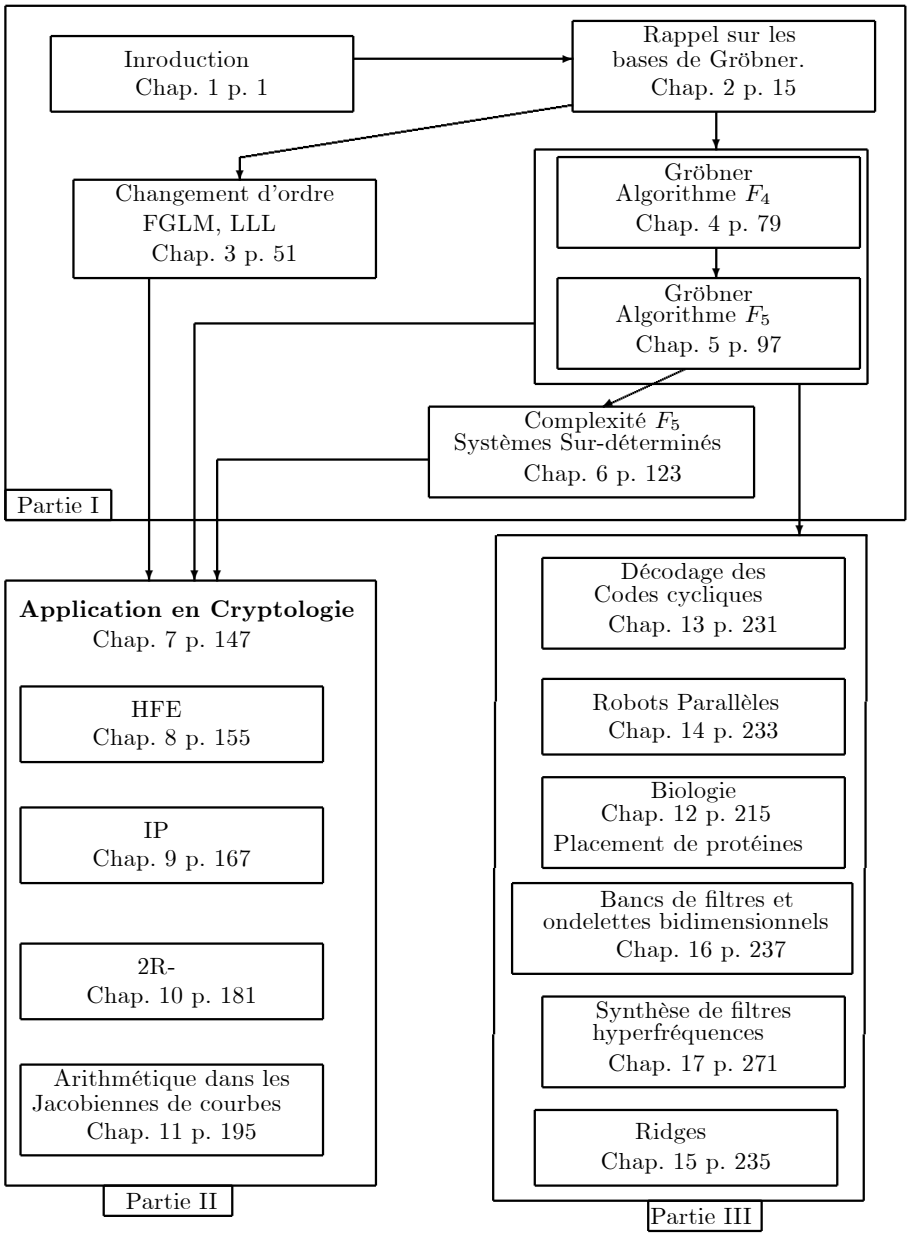


Fig. 1.2. Plan de lecture du livre

2. Bases de Gröbner et algorithme de Buchberger

2.1 Introduction

Afin de rendre plus intuitive l'introduction des quelques définitions nécessaires à la définition de base de Gröbner nous avons recourt à l'analogie avec la résolution des systèmes d'équations linéaires. L'objet mathématique associé à la résolution des systèmes linéaires est le concept d'espace vectoriel. Pour la résolution des systèmes polynomiaux l'objet mathématique sera la notion d'idéal. Plus exactement on définit un système d'équations polynomiales $f_1 = 0, \ldots, f_m = 0$ comme étant une liste de polynômes en plusieurs variables avec des coefficients dans un corps $\mathbb{K}$ dans l'anneau $\mathbb{K}[x_1, \ldots, x_n]$. À un tel système on associe I, l'idéal engendré par $f_1, \ldots, f_m$; c'est le plus petit idéal contenant ces polynômes et c'est aussi l'ensemble des $\sum_{k=1}^{m} g_k \cdot f_k$ où g_k sont dans $\mathbb{K}[x_1, \ldots, x_n]$. Comme P_k s'annule exactement aux points où tous les polynômes de I s'annulent, il est équivalent de d'étudier le système d'équations ou l'idéal I.

Pour un ensemble d'équations linéaires on peut calculer un système triangulaire équivalent en "éliminant" tous les termes de têtes de chaque équation par application de l'algorithme de Gauß. On peut appliquer une méthode similaire pour les polynômes. Pour cela il est nécessaire de définir ce qu'est le terme de tête d'un polynôme, ou en d'autres mots il faut se donner un *ordre* sur les monômes. La définition d'un ordre compatible avec la multiplication est donné en 2.3.1. Dans ce contexte une base de Gröbner (voir la définition 2.4.1) sera un autre système de générateurs de l'idéal I ayant une structure triangulaire (lorsque l'ordre admissible est l'ordre lexicographique).

Il est toujours délicat d'attribuer le mérite d'une découverte dans le domaine scientifique. Ceci est particulièrement vrai pour les bases de Gröbner ! On se bornera à Macaulay(Macaulay, 1927) qui a introduit la notion d'ordre sur les monômes. En 1964, W. Gröbner donna comme sujet de thèse le problème de trouver explicitement une base d'un idéal zéro dimensionnel sans mentionner les travaux de Macaulay; c'est alors que son étudiant, Bruno Buchberger, introduisit la notion fondamentale de paire critique(Buchberger B., 1965), un algorithme (l'algorithme de Buchberger), des critères pour éliminer des paires (les critères de Buchberger) et qui donna le nom base de Gröbner (Buchberger B., 1970; Buchberger B., 1976; Buchberger, 1987) à ce nouvel objet.

Il faut aussi mentionner un analogue des bases de Gröbner dans le domaine plus général des séries, les "bases standards" introduites par Hironaka en 1964 ainsi qu'un algorithme de division (voir le livre (Eisenbud, 1995) pour d'autres détails historiques).

2.2 Idéaux. Variétés

2.2.1 Idéaux. Théorèmes de Hilbert.

Soit $\mathbb{K}$ un corps (ou un anneau euclidien). Parfois on se placera dans $\mathbb{L}$ un corps contenant $\mathbb{K}$. On note $\overline{\mathbb{K}}$ la clôture algébrique de $\mathbb{K}$ (ainsi par exemple si $\mathbb{K} = \mathbb{Q}$ alors la clôture algébrique est $\overline{\mathbb{Q}} \subset \mathbb{C}$). La notation $\mathbb{K}[x_1, \ldots, x_n]$ désigne l'ensemble des polynômes à plusieurs variables. Un polynôme p de $\mathbb{K}[x_1, \ldots, x_n]$ est une somme de monômes:

$$p = \sum_{\alpha \in \mathbf{N}^n} c_\alpha x^\alpha$$

dont presque tous les coefficients sont nuls; on utilise la notation $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$.

Nous supposons connu du lecteur les notions d'idéaux et on pourra consulter (Cox *et al.* , 1992; Van der Waerden B.L., 1991) comme ouvrage de référence. Si F est un sous ensemble fini de $\mathbb{K}[x_1, \ldots, x_n]$, alors $\mathrm{Id}(F) = \langle F \rangle$ désigne l'idéal engendré par F.

Si

$$S \begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ \cdots \\ f_m(x_1, \ldots, x_n) = 0 \end{cases}$$

est un système d'équations algébriques (on écrit aussi un tel système en omettant les égalités à zéro $S = (f_1, \ldots, f_m)$), on lui associe I l'idéal $\mathrm{Id}(f_1, \ldots, f_m) = \langle f_1, \ldots, f_m \rangle$ engendré par les équations de départ. Résoudre formellement un tel système c'est trouver un système de générateurs de l'idéal I *"plus simple"* que les équations de départ.

Theorem 2.2.1. *(Hilbert) Pour tout idéal I de $\mathbb{K}[x_1, \ldots, x_n]$, il existe un système fini de générateur $(g_1, \ldots, g_k)$ de polynômes tel que $I = \mathrm{Id}(g_1, \ldots, g_k)$.*

Theorem 2.2.2. *(Noetherianité) Si $(I_i)_{i \in \mathbb{N}}$ une famille d'idéaux tels que*

$$I_1 \subset I_2 \subset I_3 \subset \cdots$$

alors il existe un $N \in \mathbb{N}$ tel que

$$I_N = I_{N+1} = I_{N+2} = \cdots$$

Lorsqu'on cherche à *résoudre* un système algébrique on veut trouver les solutions du système algébrique; l'objet mathématique est alors la variété algébrique. Comme l'ensemble des solutions dépend de l'ensemble dans lequel on cherche les solutions:

Definition 2.2.1. *Soit $\mathbb{L}$ un corps contenant $\mathbb{K}$, alors la variété algébrique dans $\mathbb{L}$ associée à un idéal I est*

$$V_{\mathbb{L}}(I) = \{(a_1, \ldots, a_n) \in \mathbb{L}^n \text{ tel que } f(a_1, \ldots, a_n) = 0, \ \forall f \in I\}$$

(remarquer que cette définition dépend du corps dans lequel on se place).

Par exemple si $\mathbb{K} = \mathbb{Q}$ alors on peut chercher les solutions complexes du système $V_{\mathbb{C}}(I)$ ou uniquement les solutions réelles $V_{\mathbb{R}}(I) = V_{\mathbb{C}}(I) \cap \mathbb{R}^n$.

Note 2.2.1. Si $S = (f_1, \ldots, f_m)$ est un système d'équations algébriques à résoudre, on peut donc lui associer deux objets mathématiques:

- l'idéal $I = \mathrm{Id}(f_1, \ldots, f_m)$.
- la variété algébrique correspondante $V_{\mathbb{L}}(I)$ qui est l'ensemble des zéros.

Il faut observer que I contient plus d'informations que $V_{\mathbb{L}}(I)$: par exemple considérons le système en une variable et une équation:

$$x_1^2 = 0$$

l'idéal associé est $I_1 = \mathrm{Id}(x_1^2)$ et la variété associée à I_1 est $V_{\mathbb{C}}(I_1) = \{0\}$. Donc les idéaux I_1 et $I_2 = \mathrm{Id}(x_1)$ ont la même variété algébrique associée $\{0\}$. De façon intuitive, dans la variété on perd la notion de *multiplicité*.

Réciproquement à un ensemble de points on peut associer un idéal:

Definition 2.2.2. *Soit W un ensemble de $\mathbb{K}^n$. Alors*

$$I(W) = \{f \in \mathbb{K}[x_1, \ldots, x_n] \mid f(a_1, \ldots, a_n) = 0, \ \forall (a_1, \ldots, a_n) \in W\}$$

est un idéal.

Nous avons dit qu'un idéal contient a priori des informations sur les multiplicités; il est toutefois possible d'éliminer les multiplicités en calculant l'idéal radical:

Definition 2.2.3. *Si I est un idéal alors*

$$\sqrt{I} = \left\{f \in \mathbb{K}[x_1, \ldots, x_n] \mid \exists k \in \mathbf{N} \text{ tel que } f^k \in I\right\}$$

Le théorème suivant dit du Nullstellensatz, établit la correspondance entre idéal et variété:

Theorem 2.2.3. *(Hilbert Nullstellensatz) Si I est un idéal de $\mathbb{K}[x_1, \ldots, x_n]$ alors*

$$I(V_{\overline{\mathbb{K}}}(I)) = \sqrt{I}$$

Si V est une variété algébrique sur $\mathbb{K}$ alors

$$V_{\mathbb{K}}(I(V)) = V$$

En d'autres termes en passant de l'idéal à la variété algébrique on perd essentiellement l'information sur les multiplicités des racines. Une autre interprétation du théorème est qu'on peut traduire tout énoncé sur les variétés algébriques en un énoncé sur les idéaux radicaux (à condition que $\mathbb{K}$ soit algébriquement clos).

2.2.2 Décompositions d'idéaux. Idéal Premier.

Definition 2.2.4. *Un idéal I de $\mathbb{K}[x_1, \ldots, x_n]$ est* premier *si $f \cdot g \in I$ implique $f \in I$ ou $g \in I$.*

Definition 2.2.5. *Une variété V est dite* irréductible *si dans toute expression de V comme union de deux variétés $V_1 \cup V_2$ on a $V = V_1$ ou $V = V_2$.*

Proposition 2.2.1. *Soit V une variété algébrique. V est irréductible si et seulement si $I(V)$ est premier.*

Theorem 2.2.4. *Si I est un idéal alors il existe $P_1, \ldots, P_k$ des idéaux premiers tels que*

$$\sqrt{I} = \cap_{i=1}^{k} P_i$$

Si V est une variété algébrique, il existe $V_1, \ldots, V_k$ des variétés algébriques irréductibles telles que

$$V = \cup_{i=1}^{k} V_i$$

Il existe aussi un théorème de décomposition sur les idéaux (la décomposition primaire) qui est plus difficile à énoncer. La présentation des solutions de 1.2 page 2 sous la forme 1.4 est une décomposition en variétés irréductibles.

2.3 Réduction. Ordres monomiaux.

2.3.1 Ordres admissibles

$T(x_1, \ldots, x_n) = T$ est l'ensemble des termes que l'on peut former avec les variables $(x_1, \ldots, x_n)$. Par suite T est isomorphe a $\mathbb{N}^n$. Si $t = x_1^{\alpha_1} \cdots x_n^{\alpha_n} \in T$, on note $\deg(t)$ le *degré total* de t:

$$\deg(t) = \sum_{i=1}^{n} \alpha_i$$

Definition 2.3.1. *Soit $<$ un ordre total sur les exposants (donc dans T); on dit que $<$ est* admissible *si l'une des conditions équivalentes est vérifiée:*

(i) $0 \leq \alpha$ pour tout $\alpha \in T$
(ii) $\alpha < \beta$ implique $\alpha + \gamma < \beta + \gamma$ pour tout $\gamma \in T$
(iii) n'y a pas de suite infinie strictement décroissante

On définit ensuite quelques ordres admissibles qui seront les plus utilisés. L'ordre qui induit le plus de structure sur la base de Gröbner est l' ordre **lexicographique:**

$$x^\alpha = x^{(\alpha_1,\ldots,\alpha_n)} <_{\text{Lex}} x^\beta = x^{(\beta_1,\ldots,\beta_n)} \text{ s'il existe } i \text{ tel que } \begin{cases} \alpha_j = \beta_j \text{ pour } j < i \\ \alpha_i < \beta_i \end{cases}$$

Ordre du degré lexicographique:

$$x^\alpha = x^{(\alpha_1,\ldots,\alpha_n)} <_{\text{Deg}} x^\beta = x^{(\beta_1,\ldots,\beta_n)} \text{ si } \begin{cases} \alpha_1 + \cdots + \alpha_n < \beta_1 + \cdots + \beta_n \\ \text{ou} \\ \alpha_1 + \cdots + \alpha_n = \beta_1 + \cdots + \beta_n \\ \text{et} \quad \begin{cases} \alpha_j = \beta_j \text{ pour } j < i \\ \alpha_i < \beta_i \end{cases} \end{cases}$$

L'ordre qui sera le plus efficace en pratique (voir l'article (D. & M., 1987) pour une justification) pour le calcul des bases de Gröbner sera l'**ordre du degré lexicographique inverse (DRL)** (Macaulay, 1927) Refaire

$$x^\alpha = x^{(\alpha_1,\ldots,\alpha_n)} <_{\text{DRL}} x^\beta = x^{(\beta_1,\ldots,\beta_n)} \text{ si } \begin{cases} \alpha_1 + \cdots + \alpha_n < \beta_1 + \cdots + \beta_n \\ \text{ou} \\ \alpha_1 + \cdots + \alpha_n = \beta_1 + \cdots + \beta_n \\ \text{et} \quad \begin{cases} \alpha_j = \beta_j \text{ pour } j > i \\ \alpha_i > \beta_i \end{cases} \end{cases}$$

Remarquer que l'inégalité est inversée $\alpha_i > \beta_i$: c'est donc l'ordre obtenu en filtrant par le degré puis en inversant l'ordre des variables et en prenant

l'ordre lexicographique *opposé*. Pour le calcul des bases de Gröbner, l'ordre DRL est l'ordre le plus efficace en pratique ((D. & M., 1987)) dans la majorité des exemples (cependant sur certains exemples il peut être plus avantageux de calculer directement pour un ordre lexicographique comme pour certains exemples de robotique chap. 12).

Ordre par blocs

On peut aussi fabriquer des ordres en découpant l'ensemble des variables $X = [x_1, \ldots, x_n]$ en $X_1 \cup X_1$ avec $X_1 = [x_1, \ldots, x_i]$ et $X_2 = [x_{i+1}, \ldots, x_n]$; si on se donne deux ordres admissibles $<_1$ sur $\mathbb{N}^i$ et $<_2$ sur $\mathbb{N}^{n-i}$:

$$x^\alpha = x^{(\alpha_1,\ldots,\alpha_n)} <_{X_1,X_2} x^\beta = x^{(\beta_1,\ldots,\beta_n)} \text{ si } \begin{cases} x^{(\alpha_1,\ldots,\alpha_i)} <_1 x^{(\beta_1,\ldots,\beta_i)} \\ \text{ou} \\ \begin{cases} (\alpha_1,\ldots,\alpha_i) = (\beta_1,\ldots,\beta_i) \\ \text{et } \ x^{(\alpha_{i+1},\ldots,\alpha_n)} <_2 x^{(\beta_{i+1},\ldots,\beta_n)} \end{cases} \end{cases}$$

On peut aussi définir des ordres pondérés (voir un exemple, l'ordre $C_{a,b}$, dans le chapitre 11.3.1): par si $w = (w_1, \ldots, w_n)$ est vecteur d'entiers positifs alors on définit un degré total pondéré pour un terme $x^\alpha = x^{(\alpha_1,\ldots,\alpha_n)}$ par $\deg_w(x^\alpha) = \alpha_1 w_1 + \cdots + \alpha_n w_n$.

Definition 2.3.2. *On définit le p.p.c.m. (lcm en anglais) de deux termes par la formule:*

$$\operatorname{lcm}(x^\alpha, x^\beta) = x_1^{\max(\alpha_1,\beta_1)} \cdots x_n^{\max(\alpha_n,\beta_n)}$$

Par extension, si $<$ est un ordre admissible on définit $\operatorname{lcm}(f, g) = \operatorname{lcm}(\mathrm{LT}_<(f), \mathrm{LT}_<(g))$ *où f et g sont des polynômes de $\mathbb{K}[x_1, \ldots, x_n]$.*

2.3.2 Terme de tête

Soit $f \in \mathbb{K}[x_1, \ldots, x_n]$, $f \neq 0$, tel que $f = \sum c(\alpha_1, \ldots, \alpha_n) x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ (où $c(\alpha_1, \ldots, \alpha_n)$ sont des éléments de $\mathbb{K}$). On peut définir l'ensemble $M(f)$ des *monômes de f* comme

$$M(f) = \{c(\alpha_1, \ldots, \alpha_n) x_1^{\alpha_1} \cdots x_n^{\alpha_n} \mid c(\alpha_1, \ldots, \alpha_n) \neq 0\}$$

L'ensemble $T(f)$ des *termes de f* est:

$$T(f) = \{x_1^{\alpha_1} \cdots x_n^{\alpha_n} \mid c(\alpha_1, \ldots, \alpha_n) \neq 0\}$$

On définit $T_<(F)$ comme étant cet ensemble trié pour l'ordre admissible $<$:

$$T_<(F) = \operatorname{Sort}(\{T(f) \mid f \in F\}, <)$$

Le *degré total* de $f \neq 0$ est défini par $\deg(f) = \max\{\deg(t) \mid t \in T(f)\}$.

On peut maintenant définir le *terme de tête* $\mathrm{LT}_<(f)$, le *monôme de tête* $\mathrm{LM}_<(f)$, et le *coefficient de tête* $\mathrm{LC}_<(f)$ de f par rapport à $<$ de la façon suivante: $\mathrm{LT}_<(f) = max(T(f))$, $\mathrm{LM}_<(f) = \max(M(f))$, et $\mathrm{LC}_<(f)$ comme étant le coefficient de $\mathrm{LM}_<(f)$.

Si F est un sous ensemble de $\mathbb{K}[x_1, \ldots, x_n]$ on peut étendre ces définitions: $\mathrm{LM}_<(F) = \{\mathrm{LM}_<(f) \mid f \in F\}$, $\mathrm{LT}_<(F) = \{\mathrm{LT}_<(f) \mid f \in F\}$ et $T(F) = \bigcup_{f \in F} T(f)$.

Les ordres lexicographique et DRL vont induire des propriétés de structure différentes pour le calcul d'une base de Gröbner:

Proposition 2.3.1. *Soient f un polynôme de $\mathbb{K}[x_1, \ldots, x_n]$ et $<$ l'ordre lexicographique tel que $x_1 > \cdots > x_n$, alors si $\mathrm{LT}_{<_{Lex}}(f) \in \mathbb{K}[x_i, \ldots, x_n]$ on a $f \in \mathbb{K}[x_i, \ldots, x_n]$. En particulier si $\mathrm{LT}_{<_{Lex}}(f) \in \mathbb{K}[x_n]$ alors f est un polynôme en une variable (en x_n).*

Proposition 2.3.2. *Soit f un polynôme de $\mathbb{K}[x_1, \ldots, x_n]$et $<$ l'ordre DRL tel que $x_1 > \cdots > x_n$. On a la propriété suivante*

$$\textit{si } \mathrm{LT}(f) = x_n^m \textit{ pour un entier } m > 0 \textit{ alors } x_n^m \textit{ divise } f.$$
$$\textit{pour tout } k \textit{ si } \mathrm{LT}(f) = 0 \mod\{x_k, \ldots, x_n\} \textit{ alors } f = 0 \mod\{x_k, \ldots, x_n\}$$

2.3.3 Réduction d'un polynôme

Soient $f, g, p \in \mathbb{K}[x_1, \ldots, x_n]$ tels que $p \neq 0$, et soit P un sous ensemble fini de $\mathbb{K}[x_1, \ldots, x_n]$. Alors on dit que:

- *f se réduit en g modulo p* (notation $f \xrightarrow{p} g$), s'il existe $t \in T(f)$, et il existe $s \in T$ tels que $s\,\mathrm{LT}(p) = t$ et $g = f - \frac{a}{\mathrm{LC}(p)} * s * p$ où a est le coefficient de t dans p.
- *f se réduit en g modulo P* (notation $f \xrightarrow{P} g$), si $f \xrightarrow{p} g$ pour un certain $p \in P$.
- *f est réductible modulo p* s'il existe $g \in \mathbb{K}[x_1, \ldots, x_n]$ tel que $f \xrightarrow{g} g$.
- *f est réductible modulo P* s'il existe $g \in \mathbb{K}[x_1, \ldots, x_n]$ tel que $f \xrightarrow{P} g$.
- *f est top réductible modulo P* s'il existe $g \in \mathbb{K}[x_1, \ldots, x_n]$ tel que $f \xrightarrow{P} g$ et $\mathrm{LT}(g) < \mathrm{LT}(f)$.
- $f \xrightarrow{*} g$ est la clôture réflexive transitive de $\longrightarrow$. Dans ce cas on note encore $\textsc{Reduction}(f, F) = g$
- Le *S-polynôme* de f et g est défini par

$$\mathrm{Spol}(f, g) = \mathrm{LC}(g)\frac{\mathrm{lcm}(\mathrm{LT}(f), \mathrm{LT}(g))}{\mathrm{LT}(f)} f - \mathrm{LC}(f)\frac{\mathrm{lcm}(\mathrm{LT}(f), \mathrm{LT}(g))}{\mathrm{LT}(g)} g$$

Algorithme 1 RÉDUCTION

Input: $\begin{cases} p \text{ un polynôme} \\ F = (f_1, \ldots, f_m) \text{ une liste de polynômes} \\ < \text{ ordre admissible} \end{cases}$
Output: *un polynôme réduit.*
while $p \neq 0$ ***et*** p *est top réductible modulo* F ***do***
  $\exists f \in F$ *tel que* $\mathrm{LT}(f)$ *divise* $\mathrm{LT}(p)$
  $p := p - \frac{\mathrm{LM}(p)}{\mathrm{LM}(f)} f$
return p

Proposition 2.3.3. *L'algorithme Réduction1 se termine.*

Proof. Supposons qu'on ait une boucle infinie, c'est à dire si on peut fabriquer une suite infinie (p_n) telle que $p_0 = p$ et

$$p_{n+1} := p_n - \frac{\mathrm{LM}(p_n)}{\mathrm{LM}(g_n)} g_n \text{ où } \; g_n \in F$$

on en déduit que $\mathrm{LT}(p_{n+1}) < \mathrm{LT}(p_n)$ ce qui contredit le fait que $<$ est admissible.

Remarquer que la réduction d'un polynôme par un ensemble de polynômes n'est pas unique et que le résultat dépend a priori de la *stratégie* employée c'est à dire de l'ordre des polynômes dans F.

Proposition 2.3.4. *Si* $r =$RÉDUCTION(p, F) *alors* $r - p \in \mathrm{Id}(F)$

Proof. Évident car (voir proposition 2.3.3) il existe une suite *finie* $p_0 = p$ et

$$p_{n+1} := p_n - \frac{\mathrm{LM}(p_n)}{\mathrm{LM}(g_n)} g_n \;\; g_n \in F$$

telle que $p_k = r$. On voit que:

$$p_n := p + \sum_{i=0}^{n-1} \frac{\mathrm{LM}(p_i)}{\mathrm{LM}(g_i)} g_i$$

et $p_n - p$ est dans l'idéal $\mathrm{Id}(F)$.

Corollary 2.3.1. *Si* $r =$RÉDUCTION(p, F) *alors il existe deux suites finies* $(g_n)_{n=0,\ldots,k}$ *et* $(m_n)_{n=0,\ldots,k}$ *des monômes telles que* $g_n \in F$ *et* $r - p = \sum_{i=1}^{k} m_i g_i$ *avec* $\mathrm{LT}(p) = \mathrm{LT}(m_1 g_1) > \mathrm{LT}(m_2 g_2) > \cdots > \mathrm{LT}(m_k g_k)$

Proposition 2.3.5. *Soit la fonction* $\varphi : \mathbb{K}[x_1, \ldots, x_n] \longrightarrow \mathbb{K}[x_1, \ldots, x_n]$, *définie par* $\varphi(p) =$RÉDUCTION(f, F_0). *Alors* φ *est linéaire et* $\mathrm{Ker}(\varphi) \subset \mathrm{Id}(F_0)$ *(l'inclusion peut être stricte).*

Proposition 2.3.6. *Dans de la cas particulier où les f_i sont des monômes,* $Ker\varphi = \mathrm{Id}(F_0)$

Note 2.3.1. Voici un exemple de réduction dont le résultat change selon l'ordre des calculs: $\varphi(X^2 + X, [X^2 + 1, X + 2]) = -3$ et $\varphi(X^2 + X, [X + 2, X^2 + 1]) = 2$.

Proposition 2.3.7. *Si* $p =$Réduction(f, F) *alors* $\mathrm{LT}(p) \notin \mathrm{Id}(\mathrm{LT}(F))$

2.3.4 Réduction totale d'un polynôme

La fonction Réduction se contente de simplifier (réduire) le terme de tête d'un polynôme. La fonction ReductionTotale permet de réduire tous les termes d'un polynôme (et donc d'obtenir une expression plus canonique):

Algorithme 2 ReductionTotale

Input: $\begin{cases} p \text{ un polynôme} \\ F = (f_1, \ldots, f_m) \text{ une liste de polynômes} \\ < \text{ ordre admissible} \end{cases}$
Output: *un polynôme totalement réduit.*
$p_0 := 0$
while $p \neq 0$ ***do***
  $p :=$Réduction(p, F)
  $p_0 := p_0 + \mathrm{LM}(p)$
  $p := p - \mathrm{LM}(p)$
return p 0

Proposition 2.3.8. ReductionTotale *se termine.*

Proposition 2.3.9. *Si* $p =$ReductionTotale(f, F) *alors* $T(p) \cap \mathrm{Id}(\mathrm{LT}(F)) = \emptyset$

Proposition 2.3.10. *Si* $r =$ReductionTotale(p, F) *alors* $r - p \in \mathrm{Id}(F)$

Proof. Évident.

2.4 Bases de Gröbner

2.4.1 Définition d'une base de Gröbner

Definition 2.4.1. *Soit I un idéal. Un sous ensemble fini $(g_1, \ldots, g_k)$ de $\mathbb{K}[x_1, \ldots, x_n]$ est dit une base de Gröbner de I pour l'ordre admissible $<$ si pour tout $t \in \mathrm{LT}(I)$ il existe $1 \leq i \leq k$ tel que $\mathrm{LT}(g_i)$ divise t.*

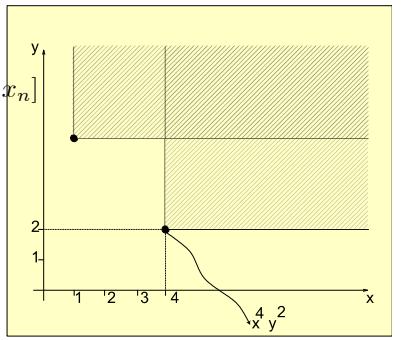


Fig. 2.1. Escalier d'un ideal.

Il est commode de reporter sur un dessin (voir figure 2.1) les points $\mathrm{LT}(f)$ pour $f \in I$; par exemple s'il existe un polynôme f de l'idéal I tel que $\mathrm{LM}(f) = x^4y^2$ on reporte sur le dessin le point de coordonnées $(4, 2)$. Le dessin met en évidence la structure d'escalier (les points extrêmes sont justement les éléments de la base de Gröbner):

Theorem 2.4.1. *On fixe un ordre admissible $<$. Tout idéal I possède une base de Gröbner G.*

On pourrait démontrer ce théorème par une preuve non constructive (basée sur lemme de Dickson) mais nous allons donner un *algorithme* permettant de calculer une base de Gröbner à partir d'un système de générateurs.

Theorem 2.4.2. *(Buchberger) Soit $G = (g_1, \ldots, g_k)$ et $<$ fixés. Les conditions suivantes sont équivalentes:*

(i) G est une base de Gröbner de $\mathrm{Id}(g_1, \ldots, g_k)$ pour $<$.
(ii) $\textsc{Réduction}(p, G) = 0$ si et seulement si $p \in \mathrm{Id}(G)$

Proof. Si G est une base de Gröbner et supposons qu'il existe $p \in I$ tel que $r = \textsc{Réduction}(p, G) \neq 0$. Alors d'après la proposition 2.3.4 $r - p \in \mathrm{Id}(G)$ et donc $r \in \mathrm{Id}(G)$. Par définition d'une base de Gröbner il existe $g \in G$ tel que $\mathrm{LT}(g)$ divise $\mathrm{LT}(r)$. Ceci contredit la proposition 2.3.7

Theorem 2.4.3. *(Buchberger) Soit $G = (g_1, \ldots, g_k)$ une base de Gröbner de $\mathrm{Id}(g_1, \ldots, g_k)$ pour $<$ fixé. Alors $\textsc{ReductionTotale}(p, G)$ est unique quelque soit la stratégie de réduction.*

Proof. Supposons que $r_1 = \textsc{ReductionTotale}(p, G)$ et $r_2 = \textsc{ReductionTotale}(p, \sigma G)$ (c'est à dire si on a modifié l'ordre des éléments de G) avec $r_1 \neq r_2$. D'après la proposition 2.3.10 il existe $(g_1, g_2) \in \mathrm{Id}(G)^2$ tels que $r_1 = p + g_1$ et $r_2 = p + g_2$ et donc $r_1 - r_2 = g_1 - g_2 \in \mathrm{Id}(G)$. Donc il existe $g \in G$ tel que $\mathrm{LT}(g) | \mathrm{LT}(r_1 - r_2)$ par suite $\mathrm{LT}(g)$ divise un élément de $T(r_1) \cup T(r_2)$ ce qui contredit la proposition 2.3.9.

Definition 2.4.2. *$G = (g_1, \ldots, g_k)$ une base de Gröbner d'un idéal I pour un ordre admissible $<$, pour tout polynôme f, on note $\textsc{NormalForm}(f, G, <)$ le résultat de la fonction $\textsc{ReductionTotale}(f, G)$. On notera aussi $\mathrm{NF}(f, G, <)$ la forme normale.*

2.4.2 Algorithme de Buchberger

Algorithme 3 *de Buchberger*

Input: $\begin{cases} F = (f_1, \ldots, f_m) \text{ une liste de polynômes} \\ < \text{un ordre admissible} \end{cases}$
Output: G *un sous ensemble (fini) de* $\mathbb{K}[x_1, \ldots, x_n]$.
$G := F$
$P := \{(f_i, f_j) \mid 1 \leq i < j \leq n\}$
while $P \neq \emptyset$ ***do***
 Sélectionner une paire (f, g) *de* P
 $P := P \backslash \{(f, g)\}$
 $f_{n+1} := \mathrm{Spol}(f, g)$
 $f_{n+1} :=$RÉDUCTION(f_{n+1}, G)
 if $f_{n+1} \neq 0$ ***then***
 $n := n + 1$
 $P := P \cup \{(f_i, f_n) \mid \ 1 \leq i < n\}$
 $G := G \cup \{f_n\}$
return G

Theorem 2.4.4. *L'algorithme de Buchberger se termine.*

Proof. La preuve de la terminaison de l'algorithme de Buchberger se fait en remarquant que:

- Le seul moment où $\mathrm{Id}(G)$ augmente est lorsque $G := G \cup \{f_n\}$ et donc $\mathrm{Id}(G \cup \{f_n\}) \supsetneq \mathrm{Id}(G)$
- a fortiori $\mathrm{Id}(\mathrm{LT}(G) \cup \{\mathrm{LT}(f_n)\}) \supset \mathrm{Id}(\mathrm{LT}(G))$
- f_n n'est pas réductible par G et donc $\mathrm{LT}(f_n)$ n'est pas dans $\mathrm{Id}(\mathrm{LT}(G))$ d'où

$$\mathrm{Id}(\mathrm{LT}(G) \cup \{\mathrm{LT}(f_n)\}) \supsetneq \mathrm{Id}(\mathrm{LT}(G))$$

- on fabrique une suite d'idéaux strictement croissante qui doit donc stationner à un moment.

pour achever la preuve de la terminaison il reste à montrer que l'algorithme de réduction Réduction termine aussi: lorsque on effectue une réduction du terme de tête $f \xrightarrow{h} g$ on a $\mathrm{LT}(g) < \mathrm{LM}(f)$ d'où une chaîne strictement décroissante de termes.

Note 2.4.1. Cette preuve de l'algorithme de Buchberger est non constructive et donc on ne peut pas en tirer une borne de complexité.

Theorem 2.4.5. *L'algorithme de Buchberger calcule une base de Gröbner de l'idéal engendré par* $(f_1, \ldots, f_m)$.

Pour faire la preuve de ce théorème nous avons besoin d'un théorème caractérisant les bases de Gröbner qui fait l'objet du paragraphe suivant.

2.4.3 Caractérisation d'une base de Gröbner

Definition 2.4.3. *Soient $P = [p_1, \ldots, p_k]$ un sous ensemble fini de $\mathbb{K}[x_1, \ldots, x_n]$, $0 \neq f \in \mathbb{K}[x_1, \ldots, x_n]$, et $t \in T$. Si*

$$f = \sum_{i=1}^{k} m_i p_i$$

*alors on dit que c'est une $t-$*représentation *de f par rapport à P si $t \geq \mathrm{LT}(m_1 p_1) \geq \mathrm{LT}(m_2 p_2) \geq \cdots$. On note $f = \mathcal{O}_P(t)$ cette propriété.*

Definition 2.4.4. *Soient P un sous ensemble fini de $\mathbb{K}[x_1, \ldots, x_n]$, $0 \neq f \in \mathbb{K}[x_1, \ldots, x_n]$, et $t \in T$. On note $f = o_P(t)$ s'il existe $t' < t$ tel que $f = \mathcal{O}_P(t')$.*

Proposition 2.4.1. *Si f, g sont des polynômes, t un terme, P un sous ensemble fini de polynômes alors*

$$\begin{array}{llll} f = \mathcal{O}_P(t) & g = \mathcal{O}_P(t) & \textit{implique} & f + g = \mathcal{O}_P(t) \\ f = o_P(t) & g = o_P(t) & \textit{implique} & f + g = o_P(t) \\ f = \mathcal{O}_P(t) & u \in T & \textit{implique} & uf = \mathcal{O}_P(ut) \\ f = o_P(t) & u \in T & \textit{implique} & uf = o_P(ut) \end{array}$$

Proposition 2.4.2. *Si* $\textsc{Réduction}(p, P) = 0$ *alors* $p = \mathcal{O}_P(\mathrm{LT}(p))$

Proof. Il suffit d'utiliser le corollaire 2.3.1.

Theorem 2.4.6. *G est une base de Gröbner si et seulement si $\forall 0 \neq f \in \mathrm{Id}(G)$, $f = \mathcal{O}_G(\mathrm{LT}(f))$.*

Proof.
- Si on suppose que G est une base de Gröbner (constitué de polynômes unitaires), alors $\forall f \neq 0 \in Id(G)$, $\textsc{Réduction}(f, G) = 0$ donc $f = \mathcal{O}_G(\mathrm{LT}(f))$ d'après la proposition 2.4.2.
- Réciproquement, soit $f \in \mathrm{Id}(G)$, $f = \sum_{i=1}^{k} t_i g_i$ avec $\mathrm{LT}(f) \geq \mathrm{LT}(t_1 g_1)$. L'inégalité stricte est impossible donc $\mathrm{LT}(f) = \mathrm{LT}(t_1 g_1) = \cdots = \mathrm{LT}(t_{i_1} g_{i_1})$. Donc f est top réductible par g_1 donc par G et G est une base de Gröbner.

Le théorème suivant est plus puissant et sera utilisé à la fois pour la preuve de l'algorithme de Buchberger avec critères et celle de l'algorithme F_4 avec critères (théorème 4.3.2):

Theorem 2.4.7. *Soit G un ensemble fini de polynômes ne contenant pas zéro. On suppose que pour tout g_1, g_2 dans G, $\mathrm{Spol}(g_1, g_2) = 0$ ou $\mathrm{Spol}(g_1, g_2) = o_G(\mathrm{lcm}(\mathrm{LT}(g_1), \mathrm{LT}(g_2)))$. Alors G est une base de Gröbner.*

Proof. Soit f un élément non nul de l'idéal fixé. On peut écrire $f = \sum_{i=1}^{k} m_i g_i$ avec la convention $\mathrm{LT}(m_1 g_1) \geq \mathrm{LT}(m_2 g_2) \geq \cdots$. Parmi toutes ces représentations on peut choisir celle qui rende minimale $\mathrm{LT}(m_1 g_1)$ et, en cas d'égalité, celle qui minimise r le nombre de $\mathrm{LT}(m_1 g_1) = \cdots = \mathrm{LT}(m_r g_r) > \mathrm{LT}(m_{r+1} g_{r+1})$. On veut utiliser le théorème précédant et montrer que $t = \mathrm{LT}(m_1 g_1) \leq \mathrm{LT}(f)$. Supposons le contraire, c'est à dire $t > \mathrm{LT}(f)$. Ceci implique $r \geq 2$. On peut écrire f sous la forme:

$$f = m_1 g_1 - \frac{\mathrm{LC}(m_1)}{\mathrm{LC}(m_2)} m_2 g_2 + \left[1 + \frac{\mathrm{LC}(m_1)}{\mathrm{LC}(m_2)}\right] m_2 g_2 + m_3 g_3 + \cdots + m_k g_k$$

On pose $m'_1 = m_1$ et $m'_2 = \frac{\mathrm{LC}(m_1)}{\mathrm{LC}(m_2)} m_2$. Et donc $t = \mathrm{LT}(m'_1 g_1) = \mathrm{LT}(m'_2 g_2)$, par conséquent $\mathrm{lcm}(\mathrm{LT}(g_1), \mathrm{LT}(g_2))$ divise t, c'est à dire:

$$\begin{array}{l} m'_1 g_1 - m'_2 g_2 \; \frac{\mathrm{LC}(m_1) t}{\mathrm{lcm}(\mathrm{LT}(g_1), \mathrm{LT}(g_2))} spol(g_1, g_2) \\ \qquad\qquad \frac{t}{\mathrm{lcm}(\mathrm{LT}(g_1), \mathrm{LT}(g_2))} o_G(\mathrm{lcm}(\mathrm{LT}(g_1), \mathrm{LT}(g_2))) \\ \qquad\qquad o_G(t) \end{array}$$

d'où

$$f = \alpha m_2 g_2 + m_3 g_3 + \cdots + m_k g_k + o_G(t)$$

avec $\alpha = 1 + \frac{\mathrm{LC}(m_1)}{\mathrm{LC}(m_2)}$.

Si $\alpha \neq 0$ on obtient une écriture avec $(r-1)$ termes de têtes au lieu de r et si $\alpha = 0$ on obtient une écriture avec un nombre de termes inférieur ou égal à $(r-2)$ termes. Dans les deux cas ceci contredit l'hypothèse de minimalité.

Corollary 2.4.1. *(Buchberger) Soit G un sous ensemble fini de polynômes. G est une base de Gröbner si et seulement si $Spol(f,g) \xrightarrow{*} 0$ pour tout $(f,g) \in G^2 \; f \neq g$.*

Le corollaire suivant trouvera une application surprenante pour optimiser le calcul d'une base de Gröbner:

Corollary 2.4.2. *Soit G un sous ensemble fini de polynômes. G est une base de Gröbner si et seulement si $Reduction(Spol(f,g), G) = 0$ pour tout $(f,g) \in G^2 \; f \neq g$.*

Proof. Soient $(f,g) \in G^2 \; f \neq g$. On pose $t = \mathrm{LT}(Spol(f,g)) < \mathrm{lcm}(\mathrm{LT}(f), \mathrm{LT}(g))$ Si $Reduction(Spol(f,g), G) = 0$ alors d'après le théorème (2.4.2) $Spol(f,g) = \mathcal{O}_G(\mathrm{LT}(Spol(f,g))) = \mathcal{O}_G(t) = o_G(\mathrm{lcm}(\mathrm{LT}(f), \mathrm{LT}(g)))$.

Note 2.4.2. Un moyen d'appliquer ce théorème est d'appliquer l'algorithme de Buchberger (6) sur la liste de polynômes et de vérifier que f_{n+1} est toujours nul après réduction.

On peut maintenant terminer la preuve du théorème de Buchberger:

Proof. **du théorème 2.4.5**
On note G_n la base G à l'étape n. $G_0 = F$ et $G_k = G$ la base finale. On a $G_0 \subset G_1 \subset \cdots \subset G_k$. Pour tout $1 \leq i < j \leq k$ on a:
ou Réduction$(Spol(f_i, f_j), G_n) = 0$ donc $Spol(f_i, f_j) \xrightarrow{*} 0$
ou Réduction$(Spol(f_i, f_j), G_n) = f_{n+1} \neq 0$ et Réduction$(Spol(f_i, f_j), G_{n+1}) = 0$ donc $Spol(f_i, f_j) \xrightarrow{*} 0$. La preuve est terminée grâce au corollaire 2.4.1.

2.4.4 Base de Gröbner réduite

On va montrer qu'une base de Gröbner pour un ordre fixé est "pratiquement" unique.

Definition 2.4.5. *Une base de Gröbner G est minimale si pour tout $g \in G$*

(i) $\mathrm{LC}(g) = 1$
(ii) $\mathrm{LT}(g) \notin \mathrm{Id}(\mathrm{LT}(G \backslash \{g\}))$

L'algorithme suivant permet de rendre minimale une base de Gröbner:

Algorithme 4 minGBasis

Input: *$F = (f_1, \ldots, f_m)$ une base de Gröbner*
Output: *une base de Gröbner minimale.*
if *$F = \emptyset$* ***then***
  return *$\emptyset$*
p le plus grand élément de F pour $<$
$G := minGBasis(F \backslash \{p\})$
if *p n'est pas top réductible modulo G* ***then***
  $G := G \cup \left\{ \frac{p}{\mathrm{LC}(p)} \right\}$
return *G*

Proposition 2.4.3. *Si G est une base de Gröbner pour $<$ et G_1, G_2 des bases de Gröbner minimales de G pour $<$. Alors*

(i) $\mathrm{LT}(G_1) = \mathrm{LT}(G_2)$
(ii) *cardinal G_1 = cardinal G_2*

Proof. Si on pouvait trouver $g \in G_1$ tel que $\mathrm{LT}(g) \notin \mathrm{LT}(G_2)$ alors $g \in Id(G_1) = Id(G) = Id(G_2)$ donc il existe $g_2 \in G_2$ tel que $\mathrm{LT}(g_2)$ divise $\mathrm{LT}(g)$. De même il existe $g_1 \in G_1$ tel que $\mathrm{LT}(g_1)$ divise $\mathrm{LT}(g_2)$. Si on avait $g = g_1$ alors $\mathrm{LT}(g_2) = \mathrm{LT}(g) = \mathrm{LT}(g_1)$ ce qui est absurde car $\mathrm{LT}(g) \notin \mathrm{LT}(G_2)$. Donc $g \neq g_1$. Mais alors on a trouvé $g_1 \in G \backslash \{g\}$ tel que $\mathrm{LT}(g_1)$ divise $\mathrm{LT}(g)$ ce qui est contraire à la minimalité de G_1.

Pour rendre une base de Gröbner vraiment unique il faut des hypothèses plus fortes que pour une base minimale:

Definition 2.4.6. *Une base de Gröbner G est réduite si pour tout $g \in G$*

(i) $\mathrm{LC}(g) = 1$
(ii) $T(g) \cap \mathrm{Id}(\mathrm{LT}(G \backslash \{g\})) = \emptyset$

L'algorithme suivant permet de calculer une base réduite:

Algorithme 5 BaseReduite

Input: $F = (f_1, \ldots, f_m)$ *une base de Gröbner*
Output: *une base de Gröbner réduite.*
$G = minGBasis(F) = [g_1, \ldots, g_k]$
return [ReductionTotale$(g_i, G \backslash \{g_i\})$ $i = 1, \ldots, k$]

On peut facilement modifier l'algorithme de Buchberger pour qu'il retourne une base de Gröbner *réduite*:

Algorithme 6 *de Buchberger*

Input: $\begin{cases} F = (f_1, \ldots, f_m) \text{ une liste de polynômes} \\ < \text{ un ordre admissible} \end{cases}$
Output: G *un sous ensemble (fini) de* $\mathbb{K}[x_1, \ldots, x_n]$.
$G := F$
$P := \{(f_i, f_j) \mid 1 \leq i < j \leq n\}$
while $P \neq \emptyset$ ***do***
Sélectionner une paire (f, g) *de* P
$P := P \backslash \{(f, g)\}$
$f_{n+1} := \mathrm{Spol}(f, g)$
$f_{n+1} :=$ReductionTotale(f_{n+1}, G)
if $f_{n+1} \neq 0$ ***then***
$n := n + 1$
$P := P \cup \{(f_i, f_n) \mid \ 1 \leq i < n\}$
$G := G \cup \{f_n\}$
return BaseReduite(G)

L'avantage de calculer une base de Gröbner complètement réduite est que le résultat est alors unique une fois l'ordre fixé:

Theorem 2.4.8. *Soient I un idéal et $<$ un ordre admissible fixé. Alors il existe une unique base de Gröbner complètement réduite de l'idéal I.*

Proof. Soient G_1 et G_2 deux base de Gröbner réduites d'un même idéal I pour un ordre $<$ fixé. On suppose que $G_1 \neq G_2$. D'après la proposition 2.4.3 $HT(G_1) = \mathrm{LT}(G_2)$. Donc on peut trouver $g_1 \in G_1$ et $g_2 \in G_2$ tels que $g_1 \neq g_2$ et $\mathrm{LM}(g_1) = \mathrm{LT}(g_1) = \mathrm{LT}(g_2) = \mathrm{LM}(g_2)$. Comme $g_1 - g_2 \in I$, $\mathrm{ReductionTotale}(g_1 - g_2, G_1) = 0$. D'où il existe $g \in G_1$ tel que $\mathrm{LT}(g)$ divise $\mathrm{LT}(g_1 - g_2)$ donc $\mathrm{LT}(g)$ divise $T(g_1 - \mathrm{LT}(g_1)) \cup T(g_2 - \mathrm{LT}(g_2))$. Absurde.

2.5 Critères de Buchberger

L'algorithme de Buchberger tel que présenté est très inefficace. Il est nécessaire de lui apporter diverses améliorations. Dans la prochaine section nous verrons des méthodes plus éloigné de l'algorithme de Buchberger pour améliorer encore plus la vitesse des calculs.

Dans l'algorithme de Buchberger (de la section6) on passe 99% du temps de calcul à réduire des paires critiques à zéro, d'où une perte de temps importante. Les critères de Buchberger sont très utiles pour éliminer les réductions à zéro pendant le calcul. Ils présentent l'avantage d'être locaux (c'est à dire qu'il faut juste examiner les termes de têtes) et ont l'inconvénient de n'être pas parfaits: après application 90% du temps de calcul est encore passé à réduire vers zéro.

2.5.1 Premier critère de Buchberger

Le premier critère est très facile à énoncer et à implanter. Toutefois les preuves de ces critères ne sont pas à négliger.

Theorem 2.5.1. *Soient g_1 et g_2 deux polynômes* disjoints, *c'est à dire tels que* $\operatorname{lcm}(\operatorname{LT}(g_1), \operatorname{LT}(g_2)) = \operatorname{LM}(g_1)\operatorname{LM}(g_2)$ *alors* $\operatorname{Spol}(g_1, g_2) \xrightarrow{*} 0$. *On dit aussi que les termes de têtes sont étrangers.*

Corollary 2.5.1. *On peut retirer les paires critiques disjointes pendant l'algorithme.*

Corollary 2.5.2. *L'algorithme de Buchberger est une généralisation de l'algorithme de Gauss pour les systèmes linéaires.*

Proof. du théorème 2.5.1 On peut supposer g_1 et g_2 unitaires.

On a $\operatorname{lcm}(\operatorname{LT}(g_1), \operatorname{LT}(g_2)) = \operatorname{LT}(g_1)\operatorname{LT}(g_2)$ et donc

$$\operatorname{Spol}(g_1, g_2) = \operatorname{LT}(g_2)g_1 - \operatorname{LT}(g_1)g_2 = -r_2 g_1 + r_1 g_2$$

où $r_i = g_i - \operatorname{LT}(g_i)$. Comme $\operatorname{LT}(r_i) < \operatorname{LT}(g_i)$ on a:

$$\operatorname{Spol}(g_1, g_2) = \mathcal{O}_{g_1,g_2}(\max(r_1 \operatorname{LT}(g_1), r_2 \operatorname{LT}(g_2))) = o_{g_1,g_2}(\operatorname{LT}(g_1)\operatorname{LT}(g_2))$$

$$= o_{g_1,g_2}(\operatorname{lcm}(\operatorname{LT}(g_1), \operatorname{LT}(g_2)))$$

2.5.2 Deuxième critère de Buchberger

Le deuxième critère est plus compliqué à mettre en oeuvre et à démontrer. Il dit que, sous certaines conditions, lorsqu'on a traité les paires (f, g) et (g, h) il n'est pas nécessaire de calculer (f, h).

Proposition 2.5.1. *On pose* $X(f,g) = \frac{\mathrm{Spol}(f,g)}{\mathrm{lcm}(\mathrm{LT}(f),\mathrm{LT}(g))}$ *on a*

$$X(g_1,g_2) + X(g_2,g_3) = X(g_1,g_3)$$

Proof. La propriété est évidente car $\mathrm{Spol}(f,g) = \frac{\mathrm{lcm}(\mathrm{LT}(f),\mathrm{LT}(g))}{\mathrm{LT}(f)} f - \frac{\mathrm{lcm}(\mathrm{LT}(f),\mathrm{LT}(g))}{\mathrm{LT}(g)} g$ et donc $X(f,g) = \frac{f}{\mathrm{LT}(f)} - \frac{g}{\mathrm{LT}(g)}$

On peut maintenant décrire le deuxième critère de Buchberger:

Theorem 2.5.2. *Soit G une base de Gröbner. Soient g_1, g_2, p des polynômes tels que*

(i) $\mathrm{LT}(p) \mid \mathrm{lcm}(\mathrm{LT}(g_1), \mathrm{LT}(g_2))$
(ii) $\mathrm{Spol}(g_i, p) = o_G(\mathrm{lcm}(\mathrm{LT}(g_i), \mathrm{LT}(p)))$ *pour* $i = 1, 2$.

Alors $\mathrm{Spol}(g_1, g_2) = o_G(\mathrm{lcm}(\mathrm{LT}(g_1), \mathrm{LT}(g_2)))$

Proof. L'hypothèse (i) entraîne que $\mathrm{lcm}(\mathrm{LT}(g_i), \mathrm{LT}(p))$ divise $\mathrm{lcm}(\mathrm{LT}(g_1), \mathrm{LT}(g_2))$ $(i = 1, 2)$ et donc

$$\begin{aligned}
\mathrm{Spol}(g_1,g_2) &= \mathrm{lcm}(\mathrm{LT}(g_1),\mathrm{LT}(g_2)) X(g_1,g_2) \\
&= \mathrm{lcm}(\mathrm{LT}(g_1),\mathrm{LT}(g_2)) \left(X(g_1,p) + X(p,g_2)\right) \\
&= \tfrac{\mathrm{lcm}(\mathrm{LT}(g_1),\mathrm{LT}(g_2))}{\mathrm{lcm}(\mathrm{LT}(g_1),\mathrm{LT}(p))} \mathrm{Spol}(g_1,p) + \tfrac{\mathrm{lcm}(\mathrm{LT}(g_1),\mathrm{LT}(g_2))}{\mathrm{lcm}(\mathrm{LT}(g_2),\mathrm{LT}(p))} \mathrm{Spol}(p,g_2) \\
&= \tfrac{\mathrm{lcm}(\mathrm{LT}(g_1),\mathrm{LT}(g_2))}{\mathrm{lcm}(\mathrm{LT}(g_1),\mathrm{LT}(p))} o_G(\mathrm{lcm}(\mathrm{LT}(g_1),\mathrm{LT}(p))) \\
&\qquad + \tfrac{\mathrm{lcm}(\mathrm{LT}(g_1),\mathrm{LT}(g_2))}{\mathrm{lcm}(\mathrm{LT}(g_2),\mathrm{LT}(p))} o_G(\mathrm{lcm}(\mathrm{LT}(p),\mathrm{LT}(g_2))) \\
&= o_G(\mathrm{lcm}(\mathrm{LT}(g_1),\mathrm{LT}(g_2)))
\end{aligned}$$

2.5.3 Nouvelle version de l'algorithme de Buchberger avec critères

L'algorithme suivant tient compte des critères de Buchberger pour éliminer des paires critiques; la procédure UPDATE qui permet d'éliminer d'une liste de paires critiques les paires inutiles par application des deux critères de Buchberger est décrite plus bas.

Algorithme 7 *de Buchberger avec critères*

Input: $\begin{cases} F = (f_1, \ldots, f_m) \textit{ une liste de polynômes} \\ < \textit{un ordre admissible} \end{cases}$
Output: G *un sous ensemble (fini) de* $\mathbb{K}[x_1, \ldots, x_n]$.
$G := F$
$P := \emptyset$
while $F \neq \emptyset$ ***do***
Sélectionner f *dans* F
$F := F \backslash \{f\}$
$(G, P) :=$UPDATE(f, G, P)
while $P \neq \emptyset$ ***do***
Sélectionner une paire (f, g) *de* P
$P := P \backslash \{(f, g)\}$
$f_{n+1} := \mathrm{Spol}(f, g)$
$f_{n+1} :=$REDUCTIONTOTALE(f_{n+1}, G)
if $f_{n+1} \neq 0$ ***then***
$n := n + 1$
$(G, P) :=$UPDATE(f_n, G, P)
$G := G \cup \{f_n\}$
return BASEREDUITE(G)

Il nous faut maintenant décrire la fonction de mise à jours des paires critiques et de la base de Gröbner en construction; on commence par deux tests simples permettant de vérifier si une paire critique (f, g) vérifie le premier ou le deuxième critère de Buchberger:

Algorithme 8 *Critère 2 de Buchberger* CRIT2

Input: $\begin{cases} (f, g) \in \mathbb{K}[x_1, \ldots, x_n]^2 \\ P \textit{ liste de paires critiques} \end{cases}$
if $\exists (h, g) \in P$ *tel que* $\mathrm{lcm}(\mathrm{LT}(h), \mathrm{LT}(g)) | \mathrm{lcm}(\mathrm{LT}(g), \mathrm{LT}(f))$ ***then***
return true
else
return false

Algorithme 9 *Critère 1 de Buchberger* CRIT1

Input: $\begin{cases} (f, g) \in \mathbb{K}[x_1, \ldots, x_n]^2 \\ P \textit{ liste de paires critiques} \end{cases}$
if $\mathrm{LT}(f)$ *et* $\mathrm{LT}(g)$ *étrangers* ***then***
return true
else
return false

La fonction suivante permet d'éliminer d'une liste de paires critiques les paires vérifiant le critère 1 ou 2; la procédure est un peu complexe car il faut prendre soin de ne pas éliminer trop vite une paire par application du 1er critère alors que cette paire pourrait être utilisée une paire selon le 2ème

critère. Dans la procédure suivant on met également à jour la base de Gröbner en construction en éliminant les polynômes dont le terme de tête est réductible par le terme de tête du nouveau polynôme:

Algorithme 10 *Update paires et base* UPDATE

Input: $\begin{cases} f \in \mathbb{K}[x_1, \ldots, x_n] \\ G_{old} \text{ liste de polynômes} \\ P_{old} \text{ liste de paires critiques} \end{cases}$
Output: G_{new} *et* P_{new}
$P_2 := \emptyset$ $P_3 := \emptyset$
$P_1 := \{(f, g) \mid g \in G_{old}\}$
while $P_1 \neq \emptyset$ ***do***
sélection et retirer (f, g_1) *dans* P_1
if CRIT1(f, g_1) ***or not*** CRIT2$(f, g_1, P_1 \cup P_2)$ ***then***
$P_2 := P_2 \cup \{(f, g_1)\}$
while $P_2 \neq \emptyset$ ***do***
sélection et retirer (f, g) *dans* P_2
if not CRIT1(f, g) ***then***
$P_3 := P_3 \cup \{(f, g)\}$
$P_{new} := P_3$
$G_{new} := \{\, g \mid \mathrm{LT}(g)$ *n'est pas top-reductible par* $\mathrm{LT}(f)\}$

2.6 Stratégies: polynômes homogènes, sucre

2.6.1 *d* base de Gröbner

Definition 2.6.1. *Un polynôme f est homogène si pour tout $t \in T(f)$ on a* $\deg(t) = deg(f)$.

Supposons que tous les polynômes de F sont homogènes. Pour tout $(f, g) \in F^2$, les polynômes $h = \mathrm{Spol}(f, g) = m\, f - m^{'}\, g$ et $f \to f' = f - m\, p$, où $\mathrm{LT}(m\, p) \in T(f)$, sont encore homogènes.

Definition 2.6.2. *On note G_d est le résultat de l'algorithme de Buchberger tronqué au degré d (c'est à dire qu'on élimine toutes les paires critiques dont le degré total (voir la définition 4.3.2) est $> d$); on appelle G_d une d-base de Gröbner de I.*

Le théorème suivant ((Lazard D., 1983), (Becker T. and Weispfenning V., 1993) p. 471) permet de donner une structure à cette liste de polynômes lorsqu'ils sont homogènes:

Theorem 2.6.1. *Pour des polynômes homogènes $f_1, \ldots, f_l$, G_d est une base de Gröbner "jusqu'au degré d" c'est à dire:*

- $\xrightarrow{*}$ *ne dépend pas de l'ordre des calculs pour les polynômes* f *tels que* $\deg(f) \leq d$.
- $\forall p \in I$ *s.t.* $\deg(p) \leq d \implies p \xrightarrow{*} 0$
- $\mathrm{Spol}(f,g) \xrightarrow{*} 0$ *pour* (f,g) *dans* G_d^2 *tels que* $\deg(\mathrm{lcm}(\mathrm{LT}(f),\mathrm{LT}(g))) \leq d$
  De plus, il existe un D_0 *tel que pour tout* $d \geq D_0$, $G_d = G_{D_0}$ *est une base de Gröbner de* I. *On note* ∞ *ce nombre* D_0.

2.6.2 Stratégie normale. Polynômes homogènes.

Dans l'algorithme de Buchberger il existe de nombreux choix qui n'influent pas sur le résultat final mais qui ont une incidence très grande sur le temps de calcul, on appelle ces choix des *stratégies*:

Stratégie sur le choix des paires critiques. La stratégie la plus naturelle pour sélectionner les paires critiques est de considérer prioritairement les paires de bas degré:

Algorithme 11 Sélection *(Stratégie Normale)*

Input: $P \neq 0$ *une liste de paires critiques*
Output: *sélection d'une paire dans* P
$P = [(f_i, g_i),\ i = 1, \cdots, k]$
$t_0 = \min_<\{\mathrm{lcm}(\mathrm{LT}(f_i), \mathrm{LT}(g_i)),\ i = 1, \cdots, k\}$
$\exists i_0$ *tel que* $\mathrm{lcm}(\mathrm{LT}(f_{i_0}), \mathrm{LT}(g_{i_0})) = t_0$
return (f_{i_0}, g_{i_0})

Stratégie pour la réduction d'un polynôme..

Lors de la réduction de f par la famille G, si $\exists(g_1, g_2) \in G^2$ tels que $\mathrm{LT}(g_1) | \mathrm{LT}(f)$ et $\mathrm{LT}(g_2) | \mathrm{LT}(f)$, alors on choisit le polynôme le plus ancien (par ordre de création (Giovini A. and Mora T. and Niesi G. and Robbiano L. and Traverso C., 1991)) dans G pour réduire f. Avec cette stratégie et pour des polynômes homogènes, on obtient le diagramme suivant (voir figure 2.2) donnant le degré maximal du monôme $\mathrm{lcm}(\mathrm{LT}(f), \mathrm{LT}(g))$ en fonction de l'étape de l'algorithme.

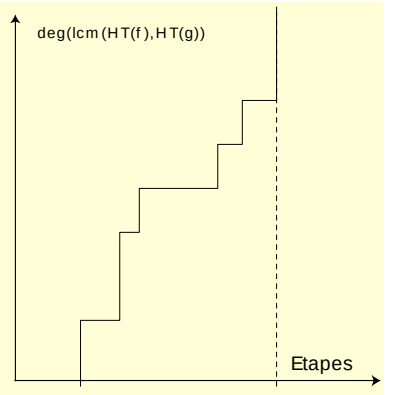


Fig. 2.2. comportement en homogene

Par contre, si on applique cette même stratégie à des polynômes affines, on observe des chutes de degré (voir la figure 2.3). Ceci induit souvent un comportement chaotique de l'algorithme et, par exemple, une croissance exagérée des coefficients.

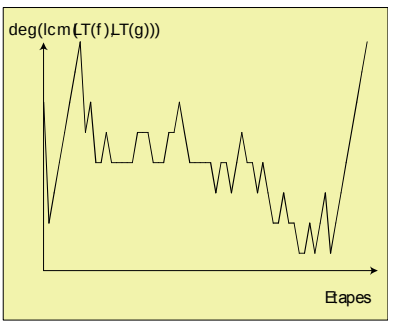


Fig. 2.3. comportement en affine

Pour remédier à ce problème, on ramène le cas des polynômes affines au cas des polynômes homogènes par homogénéisation :

Definition 2.6.3. *On définit l'opération d'homogénéisation des polynômes:*

$$\begin{aligned}\mathbb{K}[x_1, \cdots, x_n] &\to \mathbb{K}[x_1, \cdots, x_n, h] \\ f &\mapsto f^h = h^{\deg(f)} \cdot f(\tfrac{x_1}{h}, \cdots, \tfrac{x_n}{h})\end{aligned}$$

On note f_i^H la partie homogène de plus haut degré de f_i:

$$f_i^H = f^h(x_1, \cdots, x_n, 0)$$

Réciproquement on peut déshomogénéiser:

$$\begin{aligned}\mathbb{K}[x_1, \cdots, x_n, h] &\to \mathbb{K}[x_1, \cdots, x_n, h] \\ f &\mapsto f_h = f(x_1, \cdots, x_n, 1)\end{aligned}$$

Theorem 2.6.2. *Soit $F \subset \mathbb{K}[x_1, \cdots, x_n]$ un ensemble fini, on considère F^h les polynômes homogènes obtenus à partir de F par homogénéisation. On peut ensuite calculer une base de Gröbner, G, de F^h en utilisant la stratégie normale. Enfin, par déshomogénéisation, on obtient une liste de polynômes $G' = G_h$. Alors G' est une base de Gröbner (non réduite) de F.*

Résumons la stratégie sur un diagramme:

$$\begin{array}{ccc} & \text{homogénéisation} & \\ F & \longrightarrow & F^h \\ \updownarrow\ ? & & \downarrow \\ G_h & \longleftarrow & G := \texttt{Groebner}(F^h) \\ & \text{déshomogénéisation} & \end{array}$$

Note 2.6.1. La base de Gröbner ainsi obtenue n'est pas réduite. On peut dire que cette différence $G_h \backslash G$ constitue des solutions parasites (ou encore "solutions à l'infini") correspondant à $h = 0$.

2.6.3 Utilisation d'un calcul modulo p

Si on travaille sur des polynômes à coefficients dans $\mathbf{Z}$ ou $\mathbf{Q}$, pour diminuer le coût élevé des calculs arithmétiques on peut utiliser un calcul auxiliaire en effectuant l'arithmétique dans $\mathbf{Z}/p\mathbf{Z}$ où p est un nombre premier petit (souvent $p < 2^{32}$):

Soit F le système dont on veut calculer une base de Gröbner. L'idée est d'appliquer l'algorithme de Buchberger en parallèle à la famille F et à sa réduction modulo p, mais au lieu de faire la réduction de toutes les paires critiques issues de F on réduit uniquement les paires dont la réduction modulo p donne un élément non nul.

Le problème est que la famille G obtenue à la fin de l'algorithme n'est pas nécessairement une base de Gröbner. Pour vérifier que G est une base de Gröbner on applique l'algorithme de Buchberger à la famille $G \cup F$: si l'on obtient bien G c'est que c'était bien une base de Gröbner (et dans ce cas le calcul sera très souvent rapide).

2.7 Opérations sur les idéaux

Ces algorithmes et définitions seront utilisés dans les chapitre **??** et 11.

2.7.1 Somme et produit d'idéaux

Les opérations d'addition et de multiplication d'idéaux sont triviales:

Definition 2.7.1. *Soient I et J des idéaux dans $\mathbb{K}[x_1, \ldots, x_n]$. Alors:*
la somme de I et J, noté $I + J$, est définie par:

$$\mathcal{I} + \mathcal{J} = \{f + g : f \in I \text{ et } g \in J\}.$$

le produite de I et J, noté $I \cdot J$, est définie par:

$$\mathcal{I} \cdot \mathcal{J} = \{f \cdot g : f \in I \text{ et } g \in J\}.$$

Il est clair que $I + J$ et $I \cdot J$ sont des idéaux.

2.7.2 Quotient de deux idéaux

Definition 2.7.2. *Si I, J sont des idéaux alors on définit $I : J$*

$$I : J = \{f \in \mathbb{K}[x_1, \ldots, x_n] \mid f \cdot g \in I \quad \text{pour tous les } g \in J\}$$

On montre facilement que $I : J$ est un idéal.

Proposition 2.7.1. *Si I, J sont des idéaux de $\mathbb{K}[x_1, \ldots, x_n]$ alors*

$$\overline{V(I) - V(J)} \subset V(I : J)$$

Si de plus $\mathbb{K}$ est algébriquement clos et I est radical il y a égalité:

$$\overline{V(I) - V(J)} = V(I : J)$$

Pour calculer $I : J$ nous commençons par le cas particulier où J est principal c'est à dire pour un idéal I et un polynôme f de $\mathbb{K}[x_1, \ldots, x_n]$. On se ramène ensuite à calculer l'intersection de deux idéaux. Pour cela nous avons besoin d'un algorithme d'élimination des variables.

2.7.3 Élimination

Definition 2.7.3. *Soit $\mathbb{K}[x_1, \ldots, x_n, y_1, \ldots, y_s]$, un ordre $<$ est un ordre d'élimination par rapport à $y_1, \ldots, y_s$ si*

$$\forall f \in \mathbb{K}[x_1, \ldots, x_n, y_1, \ldots, y_s],\ \mathrm{LT}(f) \in f \in \mathbb{K}[y_1, \ldots, y_s] \text{ implique } f \in \mathbb{K}[y_1, \ldots, y_s]$$

Un exemple d'ordre d'élimination est l'ordre par bloc $<_{DRL,DRL}$ où le premier groupe de variables est $[x_1, \ldots, x_n]$.

Theorem 2.7.1. *Si I est un idéal de $\mathbb{K}[x_1, \ldots, x_n, y_1, \ldots, y_s]$, $<$ un ordre d'élimination par rapport à $y_1, \ldots, y_s$ et G une base de Gröbner de I pour cet ordre alors $G \cap \mathbb{K}[y_1, \ldots, y_s]$ est une base de Gröbner de $I \cap \mathbb{K}[y_1, \ldots, y_s]$.*

Note 2.7.1. Si on considère $<$ un ordre éliminant x_n alors une base de Gröbner d'un idéal zéro dimensionnel pour cet ordre $<$ contient un polynôme univarié en x_n.

2.7.4 Algorithme pour l'intersection d'idéaux

Definition 2.7.4. *Si I et J sont des idéaux alors $I \cap J$ est l'idéal:*

$$I \cap J = \{f \in \mathbb{K}[x_1, \ldots, x_n] \mid f \in I \quad et \quad f \in J\}$$

Theorem 2.7.2. *Soient I, J des idéaux de $\mathbb{K}[x_1, \ldots, x_n]$, et t une nouvelle variable. Alors:*

$$\mathcal{I} \cap \mathcal{J} = (t \cdot \mathcal{I} + (1-t) \cdot \mathcal{J}) \cap \mathbb{K}[x_1, \ldots, x_n],$$

où $t \cdot I = \{t \cdot h : h \in I\}$, et $(1-t) \cdot J = \{(1-t) \cdot h : h \in J\}$ sont dans $\mathbb{K}[t, x_1, \ldots, x_n]$.

Ce résultat combiné avec le théorème d'élimination (théorème 2.9.1), on obtient une méthode pour calculer l'intersection d'idéaux:

Algorithme 12 INTERSECT *(Intersection d'idéaux)*

Input: $I = \mathrm{Id}(f_1, \ldots, f_s)$ *et* $J = \mathrm{Id}(g_1, \ldots, g_m)$
Sortie: G *une base de Gröbner de l'idéal $I \cap J$ pour l'ordre $<$.*
Soit t est une nouvelle variable telle que $t \notin X = [x_1, \ldots, x_n]$ et $<_{t,X}$ un ordre d'élimination tel que pour tout i, $x_i < t$.
$G' :=$GRÖBNER$([tf_1, \ldots, tf_s, (1-t)g_1, \ldots, (1-t)g_m], <_{t,X})$.
$G := [g \in G' \mid t \nmid \mathrm{LT}(g)]$
return G

2.7.5 Algorithme pour le calcul du quotient de deux idéaux

La proposition suivante est à la base de l'algorithme du calcul d'un quotient (voir chapitre **??** pour une application):

Proposition 2.7.2. *Soit I, et $\{I_k\}_{1 \leq k \leq r}$ des idéaux dans $\mathbb{K}[x_1, \ldots, x_n]$. Alors:*

i) $(\cap_{k=1}^r \mathcal{I}_k) : \mathcal{I} = \bigcap_{k=1}^r (\mathcal{I}_k : \mathcal{I})$
ii) $\mathcal{I} : (\sum_{k=1}^r \mathcal{I}_k) = \bigcap_{k=1}^r (\mathcal{I} : \mathcal{I}_k)$

Un cas particulier de ii) est:

$$\mathcal{I} : \langle f_1, \ldots, f_r \rangle = \bigcap_{k=1}^r (\mathcal{I} : f_k).$$

Pour calculer le quotient il suffit de remarquer ((Cox *et al.* , 1996)):

Theorem 2.7.3. *Soit I un idéal de $\mathbb{K}[x_1, \ldots, x_n]$, et $f \in \mathbb{K}[x_1, \ldots, x_n]$. Si $\langle g_1, \ldots, g_s \rangle = I \cap \langle f \rangle$, alors*

$$\left\langle \frac{g_1}{f}, \ldots, \frac{g_s}{f} \right\rangle = \mathcal{I} : f.$$

L'algorithme suivant permet de calculer $I : (f)$ et sera utilisé dans le chapitre **??** (on verra dans ce chapitre une optimisation pour le cas où $f = x_n$).

Algorithme 13 QUOTIENT1 *(Quotient d'idéaux)*

Input: $I = \mathrm{Id}(f_1, \ldots, f_m)$ *et* f *un polynôme*
Sortie: G *une base de Gröbner de l'idéal* $I : (f)$ *pour l'ordre* $<$.
Soit t *est une nouvelle variable telle que* $t \notin [x_1, \ldots, x_n]$
et $<'$ *un ordre d'élimination tel que pour tout* i, $x_i < t$.
$G' :=$GRÖBNER$([tf_1, \ldots, tf_m, (1-t)f], <')$.
$G := \left[\frac{g}{f} \mid g \in G' \text{ et } t \nmid \mathrm{LT}(g)\right]$
return G

Maintenant nous pouvons calculer $I : J$ en appliquant plusieurs fois l'algorithme QUOTIENT1 (algorithme 13) et la procédure INTERSECTION décrite plus haut:

Algorithme 14 QUOTIENT *(Quotient d'idéaux)*

Input: $I = \mathrm{Id}(f_1, \ldots, f_s)$ *et* $J = \mathrm{Id}(g_1, \ldots, g_m)$
Sortie: G *une base de Gröbner de l'idéal* $I : J$ *pour l'ordre* $<$.
$G \leftarrow$QUOTIENT$(I, g_1, <)$.
for i ***from*** 2 ***to*** m ***do***
  $G :=$INTERSECTION$(G,$QUOTIENT1$(I, g_i, <), <)$. // $G := G \cap (I : (g_i))$
return G

2.8 Que peut on lire sur une base de Gröbner pour un ordre quelconque ?

2.8.1 Nombre fini de solutions

Si $(f_1, \ldots, f_m)$ est un système d'équations, on lui associe I l'idéal $\mathrm{Id}(f_1, \ldots, f_m)$ et on calcule une base de Gröbner réduite G de I pour un ordre admissible $<$ fixé (mais pas nécessairement l'ordre lexicographique).

On peut détecter immédiatement si le système admet des solutions:

Proposition 2.8.1. *Le système admet des solutions dans la clôture algébrique de* $\mathbb{K}$ *si et seulement si* G *n'est pas* $\{1\}$.

Proof. En effet si $1 \in G$ alors tout polynôme de $\mathbb{K}[x_1, \ldots, x_n]$ se réduit à 0 par 1 donc $I = \mathbb{K}[x_1, \ldots, x_n]$ et $V(I) = \emptyset$.

Réciproquement si $V = \emptyset$ alors $I(V) = \mathbb{K}[x_1, \ldots, x_n]$ donc en particulier $1 \in I$ et comme G est une base de Gröbner il existe $g \in G$ tel que $\mathrm{LT}(g)$ divise 1 donc $g = 1$ et comme G est réduite $G = \{1\}$.

il est très facile de détecter si l'ensemble des solutions est fini.

Lemma 2.8.1. *On suppose que $\mathbb{L}$ est algébriquement clos. On note π_i : $\begin{pmatrix} \mathbb{L}^n & \longrightarrow & \mathbb{L} \\ (x_1,\ldots,x_n) & \mapsto & x_i \end{pmatrix}$ est la ième projection et $\pi_i(V_{\mathbb{L}}(I))$ est fini alors il existe un polynôme univarié $p \in \mathbb{K}[X]$ tel que $p(x_i) \in I$.*

Proof. Supposons que $\pi_i(V_{\mathbb{L}}(I)) = \{u_1,\ldots,u_r\}$. On forme le polynôme

$$Q(x_1,\ldots,x_n) = (x_i - u_1)\cdots(x_i - u_r)$$

alors $\forall (a_1,\ldots,a_n) \in V_{\mathbb{L}}(I)$, $Q(a_1,\ldots,a_n) = 0$, c'est à dire $Q \in I(V_{\mathbb{L}}(I))$ et d'après le Nullstellensatz il existe m tel que $Q^m \in I$. On pose $P = Q^m$.

Le corollaire suivant est immédiat d'après la définition d'une base de Gröbner et permet de lire directement la propriété sur n'importe quelle base de Gröbner:

Corollary 2.8.1. *Si $\pi_i(V_{\mathbb{L}}(I))$ est fini et G une base de Gröbner de I alors il existe $p \in G$ tel que $\mathrm{LT}(p) = x_i^k$ pour un certain k.*

La réciproque est fausse en général sauf si on suppose que l'escalier est fini (c'est à dire sur le dessin que les points qui se trouvent sous l'escalier sont en nombre fini). La preuve du lemme suivant est constructive et permet de construire un polynôme univarié:

Theorem 2.8.1. *Si pour tout $i \in \{1,\ldots,n\}$, $\mathrm{LT}(p_i) = x_i^{k_i}$ où $p_i \in G$ alors $V_{\mathbb{L}}(I)$ est fini. Autrement dit, le système admet un nombre fini de solutions.*

Proof. Lorsque l'escalier est fini, l'espace vectoriel $\mathbb{K}[x_1,\ldots,x_n]/I$ est un espace vectoriel de dimension fini N et les monômes sous l'escalier forme une base de cet espace. De plus la fonction Réduction projette les polynômes sur $\mathbb{K}[x_1,\ldots,x_n]/I$:

$$\varphi\begin{pmatrix} \mathbb{K}[x_1,\ldots,x_n] & \longrightarrow & \mathbb{K}[x_1,\ldots,x_n]/I \\ p & \mapsto & \textsc{ReductionTotale}(p,I) \end{pmatrix}$$

φ est une fonction linéaire dont le noyau est précisément I. Le cas où $G = \{1\}$ ayant déjà été traité on peut supposer que $1 \notin I$ et donc $\mathbb{K}[x_1,\ldots,x_n]/I \neq \{0\}$. A l'aide de cette fonction φ (on peut la représenter aussi sous forme de matrice), on calcule les fonctions de multiplication par une variable

$$\psi_i\begin{pmatrix} \mathbb{K}[x_1,\ldots,x_n]/I & \longrightarrow & \mathbb{K}[x_1,\ldots,x_n]/I \\ p & \mapsto & \varphi(p\cdot x_i) \end{pmatrix}$$

(encore une fois on peut représenter ψ_i par une matrice $N \times N$. Soit M_i le polynôme minimal de ψ_i. On le note $M_i(X) = \sum_j a_{i,j}X^j$ (de degré $\leq N$).

Par définition $\sum_j a_{i,j}\psi_i^j = 0$ et comme $1 \in \mathbb{K}[x_1, \ldots, x_n]/I$, on peut évaluer cette égalité de fonction au point 1 (en termes de matrices on multiplie à droite par le vecteur colonne $(1, \ldots, 0)^T$).

$$\begin{array}{l}
\sum_j a_{i,j}\psi_i^j(1) = 0_{\mathbb{K}} \\
a_{i,0} + a_{i,1}\psi_i(1) + a_{i,2}\psi_i^2(1) + \cdots = 0_{\mathbb{K}} \\
a_{i,0} + a_{i,1}\varphi(x_i) + a_{i,2}\varphi(x_i\varphi(x_i)) + \cdots = 0_{\mathbb{K}} \\
\sum_j a_{i,j}\varphi(x_i^j) = 0_{\mathbb{K}} \\
\varphi(\sum_j a_{i,j}x_i^j) = \varphi(M_i(x_i)) = 0_{\mathbb{K}}
\end{array}$$

et donc $M_i(x_i)$ est dans l'idéal I et les racines de $\pi_i(V)$ sont incluses dans celles de M_i. Puisqu'en chaque variable on peut trouver un polynôme univarié on conclut que l'idéal est de dimension zéro ((Gianni P. and Mora T., 1987)) .

Theorem 2.8.2. *Soit D le nombre de monômes sous l'escalier. Si D est fini, D est le nombre de racines dans la clôture algébrique comptée avec multiplicité.*

Si le nombre de solutions est infini ?

2.8.2 Fonction et série de Hilbert. Dimension. Degré.

la fonction de Hilbert d'un idéal I regroupe des propriétés combinatoires et géométriques associées à cet idéal. Cette fonction est une donnée intrinsèque de l'idéal, et ne dépend pas en particulier du système de générateurs choisi. Nous rappelons la définition et les propriétés essentielles de la fonction de Hilbert (pour plus de détails voir (Cox *et al.* , 1996)).

Pour $d \in \mathbb{N}$ l'ensemble $\mathbb{K}[x_1, \ldots, x_n]_d = \{f \in \mathbb{K}[x_1, \ldots, x_n] \mid \deg(f) = d\}$ est un $\mathbb{K}$ espace vectoriel de dimension $\binom{n+d-1}{d}$. Si I est un idéal, alors $I_d = I \cap \mathbb{K}[x_1, \ldots, x_n]_d$ est aussi un $\mathbb{K}$ espace vectoriel.

Definition 2.8.1. *La fonction de Hilbert d'un idéal homogène* $I = \mathrm{Id}\,(f_1, \ldots, f_m)$ *en degré d est définie par*

$$\mathrm{HF}_I(d) = \mathrm{HF}(d) = \dim(\mathbb{K}[x_1, \ldots, x_n]/I)_d = \dim(\mathbb{K}[x_1, \ldots, x_n]_d) - \dim(I_d)$$

Theorem 2.8.3. *(Hilbert) À partir d'un certain degré* d_0 *il existe un polynôme P tel que*

$$\mathrm{HF}_I(d) = P(d) \text{ pour } d \geq d_0$$

d_0 *est appelé la régularité de Hilbert, ou indice de régularité; on le note noté* $H(I)$.

Le degré de P est la dimension de l'idéal et noté $\dim(I)$.

Definition 2.8.2. *La série de Hilbert est la série génératrice de* HF_I*:*

$$\mathrm{HS}_I(z) = \sum_{d \geq 0} \mathrm{HF}_I(d)\, z^d$$

d'après le théorème 2.8.3 c'est une fraction rationnelle, qui peut s'écrire

$$\frac{N(z)}{(1-z)^d} \text{ avec } N(1) \neq 0$$

où d *est la* dimension *de* I *et* $N(1)$ *est le* degré de l'idéal I *(noté aussi* $\deg(I)$*).*

Note 2.8.1. Lorsqu'on voudra parler de la dimension d'un $\mathbb{K}$ espace vectoriel E on utilisera la notation $\dim_{\mathbb{K}}(E)$ afin de distinguer la dimension d'un idéal $\dim(I)$.

2.8.3 Calcul de la dimension d'un idéal

À partir de la fonction de Hilbert d'un idéal I on peut calculer la *dimension* et le *degré* de l'idéal. Intuitivement la dimension d'un idéal premier ou d'une variété algébrique est le nombre de "paramètres libres". Pour un idéal I quelconque, la dimension de I est le maximum des dimensions dans une décomposition en idéaux premiers (théorème 2.2.4). De façon pratique:

Proposition 2.8.2. *Si* I *est un idéal et* G *une base de Gröbner, alors* $\dim(I) = \dim(\mathrm{LT}(G))$.

Comme de plus la dimension d'un idéal est égale a la dimension de l'idéal radical on peut se contenter de calculer $\dim(\mathrm{flat}(\mathrm{LT}(G)))$ où $\mathrm{flat}(x_1^{\alpha_1} \cdots x_n^{\alpha_n}) = x_1^{\beta_1} \cdots x_n^{\beta_n}$ où $\beta_i = 1$ si $\alpha_i \neq 0$ et $\beta_i = 0$ si $\alpha_i = 0$.

L'exemple suivant est un exemple classique dans la résolution des équations algébriques:

$$\text{(Cyclic } n\text{) } C_n \begin{cases} C_{n,0} = 0 \\ \cdots \\ C_{n,k} = 0 \\ \cdots \\ C_{n,n-2} = 0 \\ C_{n,n-1} = n \end{cases} \quad \text{avec } C_{n,k} = \sum_{i=0}^{n-1} \prod_{j=0}^{k} x_{(i+j) \bmod n}$$

Exemple: Cyclic 4 Soit G_4 la base de Gröbner de C_4 pour l'ordre DRL sur $[a, b, c, d]$.

$$\mathrm{LT}(G) = [c^2d^4, c^3d^2, bd^4, bcd^2, bc^2, b^2, a]$$

on calcule le radical:

$$\mathrm{Id}(\sqrt{\mathrm{LT}(G)}) = \mathrm{Id}(cd, cd, bd, bcd, bc, b, a) = \mathrm{Id}(cd, a, b) = \mathrm{Id}(d, a, b) \cap \mathrm{Id}(c, a, b)$$

et le système est de dimension 1.

2.8.4 Suites régulières. Degré de régularité

Considérons un système algébrique $F = [f_1, \ldots, f_m]$ auquel on applique un algorithme de calcul de base de Gröbner comme F_4 (voir chap. 4) qui utilise l'algèbre linéaire pour effectuer les calculs: dire qu'une matrice généré par cet algorithme n'est pas de rang plein est équivalent à dire que les lignes de cette matrice ne sont pas indépendantes. Comme chaque ligne de la matrice est un produit $t \times f$ où t est un terme et $f \in F$, la dépendance linéaire s'exprime sous la forme $\sum_{f \in F, t \in T} \lambda_{t,f}\ t\, f = 0$ ou en regroupant les termes:

$$\sum_{i=1}^{m} g_i f_i = 0 \tag{2.1}$$

où les g_i sont des polynômes de $\mathbb{K}[x_1, \ldots, x_n]$. On dit encore que $(g_1, \ldots, g_m)$ est une syzygie. On peut aussi écrire la relation (2.1) sous la forme:

$$g_1 f_1 = 0 \text{ modulo } \mathrm{Id}(f_2, \ldots, f_m) \tag{2.2}$$

autrement dit on a un *diviseur de zéro* (*si* $g_1 \neq 0$).

Pour un système linéaire on dit que le système est non singulier s'il n'existe pas de combinaison linéaire non nulle

$$\sum_{i=1}^{m} \lambda_i \, f_i = 0 \text{ avec } \lambda_i \in \mathbb{K} \tag{2.3}$$

mais pour les systèmes algébriques il n'est pas possible d'imposer la non existence de relations (2.1) non nulles: en effet il existe toujours des relations

$$f_i \, f_j - f_j \, f_i = 0 \tag{2.4}$$

qui sont des syzigies triviales; ainsi il est naturel de dire définir qu'un système est régulier s'il n'existe pas de relation de type (2.1) hormis les relations triviales (voir aussi la définition 2.8.3 du chapitre 2):

Definition 2.8.3. *Définition géométrique: le système* $(f_1, \ldots, f_m)$ *de polynômes* homogènes *est régulier si pour tout* $i \in \{1, \ldots, m\}$*, la dimension de* $\langle f_1, \ldots, f_i \rangle$ *est* $n - i$*. On dit encore que la suite* $(f_1, \ldots, f_m)$ *est régulière.*

Définition algébrique: le système $(f_1, \ldots, f_m)$ *de polynômes* homogènes *est régulier si pour tout* $i = 1, \ldots, m$ *et* g *tel que*

$$g \cdot f_i \in \langle f_1, \ldots, f_{i-1} \rangle$$

alors g *est aussi dans* $\langle f_1, \ldots, f_{i-1} \rangle$*.*

Le système $(f_1, \ldots, f_m)$ *de polynômes (pas nécessairement homogènes) est régulier si le système* $(f_1^h, \ldots, f_m^h)$ *l'est (* f_i^h *est la partie homogène de plus haut degré de* f_i*).*

Note 2.8.2. Une autre façon de caractériser les suites régulières est qu'il n'existe pas de relations algébriques non nulles de la forme:

$$\sum_i g_i \cdot f_i = 0 \text{ avec } g_i \in \mathbb{K}[x_1, \ldots, x_n]$$

hormis les relations induites par les relations triviales $f_i f_j = f_j f_i$.

Note 2.8.3. En utilisant la définition géométrique, il est facile de voir qu'il n'existe pas de système régulier lorsque $m > n$; la définition 6.2.3 est adaptée au cas des systèmes sur-déterminés (la semi-régularité) au chapitre 6.

Les suites régulières sont des objets mathématiques ayant de bonnes propriétés: par exemple on connaît leur série de Hilbert, la dimension de la variété algébrique associée qui est nécessairement $n - m$. Les suites régulières sont également caractérisées par l'absence de diviseurs de zéro : une suite est régulière si le i-ème polynôme f n'est pas diviseur de zéro dans $\mathbb{K}[x_1, \ldots, x_n]/\langle f_1, \ldots, f_m \rangle$. Les propriétés suivantes permettent de caractériser les suites régulières ((Cox *et al.* , 1998; Lang, 2002; Fröberg, 1997)) et leur comportement par rapport à un calcul de base de Gröbner est parfaitement connu:

Theorem 2.8.4. *1.* $(f_1, \ldots, f_m)$ *est régulier si et seulement si sa série de Hilbert (voir définition 2.8.2) est égale à:*

$$\mathrm{HS}_{\langle f_1,\ldots,f_m\rangle}(z) = \sum_{d\geq 0} \mathrm{HF}_{\langle f_1,\ldots,f_m\rangle}(d)\, z^d = \frac{\prod_{i=1}^{m}(1-z^{d_i})}{(1-z)^n}$$

2. après un change linéaire générique des variables le degré des polynômes d'une base de Gröbner pour un ordre DRL est borné par le l'index de régularité:

$$\textit{Borne de Macaulay: } \sum_{i=1}^{m}(d_i - 1) + 1$$

Proof. Pour la preuve de (i) voir (Lang, 2002, Theorem 6.6 p. 436); ou (Fröberg, 1997, p. 137). La preuve de l'assertion (ii) se trouve dans (Lazard D., 1983; Giusti, 1994).

Note 2.8.4. L'algorithme F_5 5 permettra de faire un calcul de base de Gröbner sans calcul inutile lorsque le système algébrique de départ est régulier.. Les suites régulières seront au coeur du chapitre 6 sur la complexité de l'algorithme F_5 et la complexité des systèmes sur-déterminés.

2.9 Quelle est la forme d'une base de Gröbner ?

Une autre façon de formuler la question serait: "pourquoi est-ce utile de calculer une base de Gröbner pour résoudre un système ?". En effet, pour le moment, il n'est pas facile de voir en quoi calculer une base de Gröbner aide à résoudre un système d'équations algébriques en pratique. C'est la raison pour laquelle nous donnons une description intuitive de la forme d'une base de Gröbner selon l'ordre admissible $<$.

2.9.1 Propriété d'élimination

Une autre propriété fondamentale des bases lexicographiques est qu'on peut *éliminer* des variables:

Theorem 2.9.1 (Elimination Theorem). *Soit* I *un idéal de* $\mathbb{K}[x_1, \ldots, x_n]$, *et* $l \in \{1, \ldots, n\}$. *Si* G *est une base de Gröbner pour l'ordre lexicographique ou un ordre par blocs de* I, *alors* $G \cap \mathbb{K}[x_{l+1}, \ldots, x_n]$ *est une base de Gröbner de* $I \cap \mathbb{K}[x_{l+1}, \ldots, x_n]$.

Si on applique le théorème 2.9.1 avec $l = n$ on obtient que $G \cap \mathbb{K}[x_n]$ est une base de Gröbner de $I_n = I \cap \mathbb{K}[x_n]$ dans $\mathbb{K}[x_n]$; dans ce I_n est un idéal principal donc engendré par un polynôme $P_n(x_n)$ éventuellement nul. Dans le paragraphe suivant on donne la forme typique d'une base de Gröbner d'un idéal zéro dimensionnel.

2.9.2 Ordre lexicographique

Une base de Gröbner pour un ordre lexicographique possède une structure forte. Par exemple, une base de Gröbner d'un système zéro-dimensionnel a toujours la forme suivante:

$$\begin{cases} f(x_n) = 0 \\ f_2(x_{n-1}, x_n) = 0 \\ \vdots \\ f_{k_2}(x_{n-1}, x_n) = 0 \\ f_{k_2+1}(x_{n-2}, x_{n-1}, x_n) = 0 \\ \vdots \\ f_{k_{n-1}+1}(x_1, \ldots, x_n) = 0 \\ \vdots \\ f_{k_n}(x_1, \ldots, x_n) = 0 \end{cases}.$$

2.9.3 Ordre lexicographique. Shape Position

Lorsque les conditions suivantes sont vérifiées

- $n = m$ le nombre d'équations est égal au nombre de variables.
- l'idéal I est de dimension 0.
- l'ordre $<$ est l'ordre lexicographique avec $x_1 > \cdots > x_n$.
- le système est en Shape Position ou encore en position *générique* (en pratique un système tiré aléatoirement est en Shape Position)

alors la forme de G, base de Gröbner pour l'ordre lexicographique, est (Becker *et al.* , 1994):

$$\begin{cases} P_n(x_n) = 0, \\ x_{n-1} = P_{n-1}(x_n) \\ \cdots \\ x_1 = P_1(x_n) \end{cases} \tag{2.5}$$

où tous les P_i sont des polynômes **univariés**. A partir de cette représentation des solutions il est maintenant "facile" (au moins en théorie) de résoudre numériquement le système: soient $z_1, \ldots, z_d$ toutes les racines complexes de P_n (on peut se restreindre aux racines réelles si besoin est) alors l'ensemble des solutions est:

$$\{(x_n = z_i, x_{n-1} = P_{n-1}(z_i), \ldots, x_1 = P_1(z_i)) \quad i = 1, \ldots, d\}$$

2.9.4 RUR

Pour des raisons de stabilité numérique ou pour limiter la taille des coefficients des polynômes P_i on préfère exprimer les solutions sous forme *rationnelle* (voir la RUR l'annexe et (Rouillier, 1999, F. Rouillier)):

$$\begin{cases} P_n(x_n) = 0, \\ x_{n-1} = \frac{N_{n-1}(x_n)}{D_{n-1}(x_n)} \\ \cdots \\ x_1 = \frac{N_1(x_n)}{D_1(x_n)} \end{cases}$$

Le plus souvent les dénominateurs sont identiques: $D_i(x_n) = \frac{\partial P_n}{\partial x_n}$. Le principe pour calculer des approximations des racines est le même.

Lorsque le système algébrique possède un nombre fini de solution mais n'est pas en position générique il est nécessaire de se ramener au cas univarié; c'est la forme RUR (Rouillier, 1999) qui permet d'établir une bijection entre les racines (resp. les racines réelles) du système initial et les racines du système suivant:

$$\begin{cases} P_n(T) = 0, \\ x_n = \frac{N_{n-1}(T)}{D_{n-1}(T)} \\ \cdots \\ x_1 = \frac{N_1(T)}{D_1(T)} \end{cases}$$

À noter qu'une variable supplémentaire T a été ajoutée; de plus la forme RUR conserve les *multiplicités* des racines.

Le problème Katsura n (Katsura K., 1986) est un exemple classique dans les systèmes polynomiaux. Cet exemple provient de la physique (électromagnétisme); L'exemple est un exemple provenant de la physique: le problème de Katsura ; il s'agit de calculer des probabilités en se ramenant à un système algébrique quadratique en les variables $x_0, \ldots, x_{N-1}$:

$$\text{Katsura}_N \quad \begin{cases} \sum_{l=-N}^{N} x_{|l|} \cdot x_{|m-l|} = x_m, \;\; m = 0, \ldots, (N-1) \\ \sum_{l=-N}^{N} x_{|l|} = 1 \end{cases}$$

par exemple il s'agit de calculer toutes les solutions réelles du système (cas $N = 4$):

$$\text{Katsura}_4 \begin{cases} -x_2 + 2\,x_5\,x_4 + 2\,x_4\,x_3 + 2\,x_3\,x_2 + 2\,x_2\,x_1, \\ -x_3 + 2\,x_5\,x_3 + 2\,x_4\,x_2 + 2\,x_3\,x_1 + x_2^2, \\ -x_1 + 2\,x_5^2 + 2\,x_4^2 + 2\,x_3^2 + 2\,x_2^2 + x_1^2, \\ -x_4 + 2\,x_5\,x_2 + 2\,x_4\,x_1 + 2\,x_3\,x_2, \\ -1 + 2\,x_5 + 2\,x_4 + 2\,x_3 + 2\,x_2 + x_1 \end{cases}$$

avec les conditions supplémentaires $0 \leq x_i \leq 1$. On calcule une représentation rationnelle des solutions en calculant une base de Gröbner pour l'ordre lexicographique (dans le cas présent c'est aussi une Rational Univariate Representation). On trouve:

$$P_5 = {x_5}^{16} - \tfrac{8}{7}\,{x_5}^{15} + \tfrac{16265}{36036}\,{x_5}^{14} - \tfrac{9923}{255528}\,{x_5}^{13} - \tfrac{3568601}{146162016}\,{x_5}^{12} + \tfrac{13056217}{1554632352}\,{x_5}^{11} - \tfrac{4655285}{6514649856}\,{x_5}^{10} - \tfrac{46509877}{273615293952}\,{x_5}^{9} + \tfrac{14897785}{312703193088}\,{x_5}^{8} - \tfrac{622165}{273615293952}\,{x_5}^{7} - \tfrac{5967119}{8755689406464}\,{x_5}^{6} + \tfrac{279035}{2918563135488}\,{x_5}^{5} + \tfrac{6425}{2694058278912}\,{x_5}^{4} - \tfrac{15481}{17511378812928}\,{x_5}^{3} + \tfrac{11}{1819364032512}\,{x_5}^{2} + \tfrac{109}{46697010167808}\,x_5$$

et les dénominateurs sont tous identiques $D_i = \frac{1}{16}\frac{\partial P_5}{\partial x_5}$ (on rend le dénominateur unitaire pour améliorer la stabilité numérique). Les numérateurs sont:

$$N_1 = \tfrac{487}{924}\,{x_5}^{15} - \tfrac{42059}{72072}\,{x_5}^{14} + \tfrac{152749}{681408}\,{x_5}^{13} - \tfrac{2761925}{134918784}\,{x_5}^{12} - \tfrac{2736517027}{273615293952}\,{x_5}^{11} + \tfrac{264168611}{78175798272}\,{x_5}^{10} - \tfrac{71124337}{243213594624}\,{x_5}^{9} - \tfrac{38620969}{729640783872}\,{x_5}^{8} + \tfrac{259225391}{17511378812928}\,{x_5}^{7} - \tfrac{234523}{312703193088}\,{x_5}^{6} - \tfrac{775601}{5003251089408}\,{x_5}^{5} + \tfrac{140879}{6671001452544}\,{x_5}^{4} + \tfrac{67751}{280182061006848}\,{x_5}^{3} - \tfrac{35335}{280182061006848}\,{x_5}^{2} + \tfrac{25}{16980730970112}\,x_5 + \tfrac{109}{747152162684928}$$

$$N_2 = 1/33\,{x5}^{15} - \tfrac{11909}{288288}\,{x5}^{14} + \tfrac{24359}{1070784}\,{x5}^{13} - \tfrac{19254679}{3507888384}\,{x5}^{12} - \tfrac{3415463}{34201911744}\,{x5}^{11} + \tfrac{422017319}{1094461175808}\,{x5}^{10} - \tfrac{1127725}{14031553536}\,{x5}^{9} + \tfrac{7141895}{8755689406464}\,{x5}^{8} + \tfrac{12105659}{5837126270976}\,{x5}^{7} - \tfrac{3212897}{11674252541952}\,{x5}^{6} - \tfrac{178501}{17511378812928}\,{x5}^{5} + \tfrac{66775}{15565670055936}\,{x5}^{4} - \tfrac{5881}{93394020335616}\,{x5}^{3} - \tfrac{11065}{560364122013696}\,{x5}^{2} + \tfrac{149}{560364122013696}\,x5$$

On peut maintenant calculer toutes les racines (dans le problème il faut ne garder que les racines réelles entre 0 et 1):

$x_5 = [0, -.2213900152, -.1306019370, -.1305929660, -.1068488804,$

$-.550947041910^{-1}, \quad .725861843210^{-1} - .1813736388\,i,$

$.725861843210^{-1} + .1813736388\,i, .885750369610^{-1},$

$.1204723010, .1396485709 - .1368283732\,i,$

$.1396485709 + .1368283732\,i, .2449333158,$

$.2734590803, .3021430681, .3333333333]$

et on substitue ces valeurs de $x_5[i]$ dans N_i et D_i; pour $x_5[1] = 0$ on trouve:

$[x_5 = 0, x_1 = 1.0, x_2 = 0, x_3 = 0, x_4 = 0]$

pour $x_5[2] = -.2213900152$ on trouve:

$[x_5 = -0.2213900152, x_1 = 0.5371015101, x_2 = 0.09809297652,$

$x_3 = 0.1402685276, x_4 = 0.2144777578]$

2.9.5 Base pour un ordre du degré

Bien que la forme d'une base de Gröbner pour un ordre DRL ne soit pas simple à décrire ces bases ont la propriété de contenir tous les polynômes de plus bas degré d'un idéal I. Plus exactement on peut calculer une base de I_d lorsque d est le degré minimal d'un élément non nul de I:

Theorem 2.9.2. *Soit $I \subset \mathbb{K}[x_1, \ldots, x_n]$, $d = \min\{\deg(f) \mid 0 \neq f \in I\}$, et G une base de Gröbner pour l'ordre DRL de I. Alors:*

$$\mathrm{Vect}_{\mathbb{K}}(g \in G \mid \deg(g) = d) = \mathrm{Vect}_{\mathbb{K}}\left(g \in I \mid \deg(g) = d\right).$$

Autrement dit, les éléments de plus bas degré de G forment une base (comme espace vectoriel) de I_d.

Note 2.9.1. Voir une application de théorème dans l'article (Ars, G. and Faugère, J.-C, 2005).

3. Algorithmes pour l'élimination

Pour améliorer l'efficacité du calcul d'une base de Gröbner on peut travailler dans deux directions:

1. améliorer le calcul direct d'une base de Gröbner et remplacer ou optimiser l'algorithme de Buchberger: c'est l'objet des chapitres 4 et 5.
2. effectuer le calcul d'une base de Gröbner en plusieurs étapes: en effet on remarque le calcul d'une base pour un ordre lexicographique (ou un ordre d'élimination) est en général le plus utile pour exprimer les solutions (voir 2.9.2 et la forme Shape Position (2.5)) alors que le calcul d'une base de Gröbner pour un ordre du degré est plus rapide (voir (D. & M., 1987)) mais donne moins d'informations à la lecture immédiate de la base.

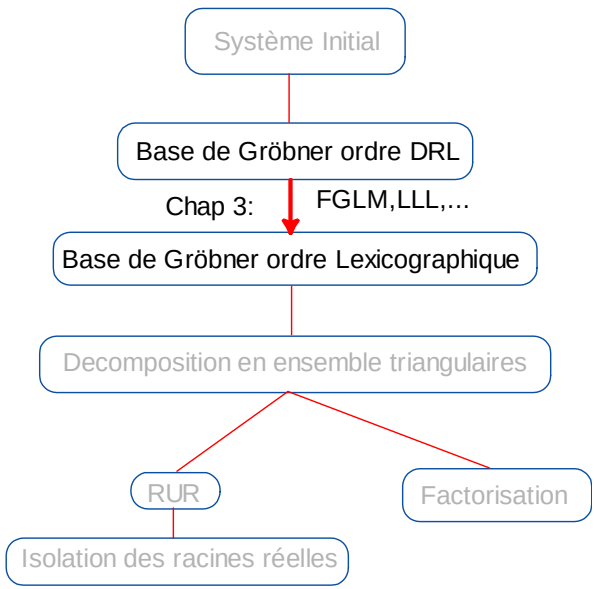


Fig. 3.1. Algorithmes de changement d'ordre dans le processus global de résolution.

En fait les algorithmes de ce chapitre (FGLM et LLL) permettent de transformer efficacement une base de Gröbner calculée pour un premier ordre admissible en une base de Gröbner pour un deuxième ordre admissible. On parle dans tous les cas d'algorithme de changement d'ordre. La figure 3.1 représente l'utilisation typique de ces algorithmes lorsqu'on veut résoudre un système algébrique.

3.1 Algorithme FGLM

L'algorithme FGLM (le nom est la réunion des initiales des auteurs de l'algorithme) a été publié dans (Faugère, J.C., Gianni, P., Lazard, D. and Mora T., 1993).

3.1.1 Introduction.

L'algorithme FGLM permet de ramener à un calcul d'algèbre linéaire l'opération de changer l'ordre d'une base de Gröbner d'un idéal zéro dimensionnel. Un avantage théorique de cet algorithme est qu'on peut dériver une estimation très précise de la complexité de cet algorithme; de plus cela permet d'améliorer la complexité du calcul d'une base de Gröbner pour n'importe quel ordre: par exemple en utilisant les résultats de (Caniglia L. and Galligo A. and Heintz J., 1988; Caniglia L. and Galligo A. and Heintz J., 1991) on peut montrer que pour l'ordre DRL, la complexité du calcul d'une base de Gröbner d'un système algébrique d'un système de n équations en n variables est borné par $C_1\, d^{C_2\, n^2}$ (où C_1, C_2 sont des constantes). Avec l'hypothèse supplémentaire que le système admet un nombre fini de solutions à l'infini cette complexité est majorée par $C_1\, d^{C_2\, n}$ (voir (Lazard D., 1983)). En pratique le calcul des bases de Gröbner est souvent plus rapide mais on constate que c'est souvent avec l'ordre DRL que le calcul est le plus rapide. Pour l'ordre lexicographique la combinaison de l'algorithme FGLM et de ces résultats permettent de garder la même complexité: par exemple ceci permet d'améliorer la borne $C_1\, d^{C_2\, n^3}$ ((Caniglia L. and Galligo A. and Heintz J., 1988)) en $C_1\, d^{C_2\, n^2}$. Le chapitre 6 contient d'autres résultats de complexité dans le cas où le système initial est régulier ou semi-régulier.

Pour appliquer l'algorithme FGLM il suffit d'une base de Gröbner calculée pour un ordre admissible; à partir de cette base on calcule des matrices de multiplications par les variables (voir la section 3.1.3 15 et l'algorithme 15). À noter que ces matrices peuvent être utilisées pour calculer numériquement les racines d'un système algébrique ((Auzinger and Stetter H., 1998; Möller H.M., 1993)): les valeurs propres des matrices de multiplications donnent les (projections) des solutions du système. Symboliquement le plus petit polynôme de la base de Gröbner pour l'ordre lexicographique est aussi le polynôme minimal de la matrice de multiplication par la plus petite variable.

Lorsque la base de l'ordre lexicographique n'est pas en Shape Position il est souvent préférable de calculer directement la RUR (Rouillier, 1999) de l'idéal; l'algorithme utilise également les matrices de multiplication mais calcule des polynômes caractéristiques d'une combinaison linéaire des matrices de multiplication.

L'algorithme FGLM est implanté dans tous les systèmes de Calcul Formel (Mathematica, Maple, Magma, Singular, ...) et l'expérience a montré l'efficacité de cet algorithme par rapport à un calcul direct; on illustre l'efficacité de cet algorithme dans une application en Cryptologie (voir 3.1.10). Dans le chapitre 11 on verra aussi une utilisation de l'algorithme FGLM pour prouver des formules d'arithmétique dans une jacobienne d'une courbe super elliptique (dans ce cas on effectuera un changement d'ordre en partant d'un ordre lexicographique vers un ordre du degré).

3.1.2 Escalier. Frontière d'un idéal.

On considère uniquement un idéal I zéro dimensionnel (voir le théorème 2.8.1 pour un moyen algorithmique de vérifier qu'un idéal est zéro dimensionnel). Pour simplifier l'écriture des algorithmes on écarte également le cas trivial où $I = \mathbb{K}[x_1, \ldots, x_n]$.

Definition 3.1.1. *Étant donné un idéal de dimension zéro I dans $\mathbb{K}[x_1, \ldots, x_n]$ et $(G, <)$ une base de Gröbner de I, on considère l'ensemble*

$$\mathcal{E}(G) = \{t \in T \mid t \text{ n'est pas réductible par } G\} \text{ trié pour l'ordre } < .$$

On dit que $\mathcal{E}(G)$, l'escalier de I, est une base canonique par rapport à G du $\mathbb{K}$-espace vectoriel $\mathbb{K}[x_1, \ldots, x_n]/I$. On note $\deg(I)$ la dimension du $\mathbb{K}$-espace vectoriel $\mathbb{K}[x_1, \ldots, x_n]/I$: c'est donc le degré de l'idéal I (voir définition 2.8.2 p. 42) et le cardinal de la base canonique $\mathcal{E}(G)$. On note

$$\mathcal{E}(G) = \left\{e_1 = 1 < e_2 < \cdots < e_{\deg(I)}\right\}$$

L'escalier $\mathcal{E}(G)$ est clos pour la division:

Proposition 3.1.1. ??*Si $1 \neq e \in \mathcal{E}(G)$ alors pour tout i tel que x_i divise e on a $\frac{e}{x_i} \in \mathcal{E}(G)$.*

On cherche une estimation du nombre d'éléments dans G en fonction de $\deg(I)$; pour cela on introduit le bord de l'escalier:

Definition 3.1.2. *Soit $\mathcal{E}(G)$ la base canonique de $\mathbb{K}[x_1, \ldots, x_n]$ $/I$, alors*

$$\mathcal{F}(G) = \{x_i e \mid e \in \mathcal{E}(G),\ 1 \leq i \leq n \text{ et } x_i\, e \notin \mathcal{E}(G)\}$$

est la frontière de G.

On peut caractériser les éléments de $\mathcal{F}(G)$:

Proposition 3.1.2. *Si I est un idéal de dimension zéro, $(G, <)$ une base de Gröbner réduite par rapport à un ordre admissible $<$, alors pour tout élément $t \in \mathcal{F}(G)$ alors*

1. *ou il existe $g \in G$ tel que $t = \mathrm{LT}(g)$*
2. *ou il existe j et $t' \in \mathcal{F}(G)$ tel que $t = x_j t'$.*

Proof. On considère l'ensemble $A = \{1 \leq j \leq n \text{ tel que } x_j \text{ divise } t \text{ et} \frac{t}{x_j} \notin \mathcal{E}(G)\}$.

1. Si A est vide. Comme $t \notin \mathcal{E}(G)$ il existe $g \in G$ tel que $\mathrm{LT}(g)$ divise t: soit $u = \frac{t}{\mathrm{LT}(g)}$. S'il existait x_j divisant u alors en posant $v = \frac{u}{x_j}$ on aurait $\frac{t}{x_j} = \mathrm{LT}(g)\, v \notin \mathcal{E}(G)$ et donc $j \in A$ ce qui est absurde. Donc $u = 1$ et $\mathrm{LT}(g) = t$.
2. Si A n'est pas vide il existe j tel que x_j divise t et $t' = \frac{t}{x_j} \notin \mathcal{E}(G)$. Comme $t \in \mathcal{F}(G) : x_j\, t' = t = x_i\, e$ avec $b \in \mathcal{E}(G)$. L'égalité $i = j$ est impossible car cela impliquerait $t' = e \in \mathcal{E}(G)$ Donc $i \neq j$ et x_j divise e; de plus e/x_j (prop. **??**) est dans $\mathcal{E}(G)$; ainsi $t' = x_i \cdot (e/x_j) \in \mathcal{F}(G)$.

Corollary 3.1.1. *Le nombre de générateurs d'une base de Gröbner d'un idéal zéro dimensionnel I; alors est inférieur à $n \deg(I)$.*

Proof. Si $g \in G$, alors $\mathrm{LT}(g) \notin \mathcal{E}(G)$ et pour tout k tel que x_k divise $\mathrm{LT}(g)$ on a $\frac{\mathrm{LT}(g)}{x_k} \in \mathcal{E}(G)$; donc $\mathrm{LT}(g) \in \mathcal{F}(G)$. La borne découle du fait $\mathrm{LT}(G) \subseteq \mathcal{F}(G)$.

Note 3.1.1. On verra une meilleure borne (théorème 6.6.1) lorsque le système est générique (plus exactement en position de Noether simultanée) et que $\deg(I)$ est égale à la borne de Bezout.

3.1.3 Construction des matrices de multiplication

Dans la suite on travaille dans l'espace vectoriel $\mathbb{K}[x_1, \ldots, x_n]/I$ et on considère la base canonique associée à G:

$$\mathcal{E}(G) = \left\{e_1 = 1 < e_2 < \cdots < e_{\deg(I)}\right\}.$$

Pour trouver l'expression d'un polynôme f dans cette base on doit calculer $\mathrm{NF}(f, G)$ en utilisant la procédure 2: cependant il est difficile d'obtenir une borne de complexité précise en théorie ou en pratique. C'est la raison pour laquelle on va ramener ce calcul un produit matrice vecteur dont il sera facile de borner la complexité.

Dans la suite on suppose qu'on connaît une fonction linéaire

$$\varphi : \mathbb{K}[x_1, \ldots, x_n] \longrightarrow \mathbb{K}[x_1, \ldots, x_n]/I$$

qui soit une forme normale, c'est à dire qui vérifie les propriétés:

$\varphi(p) = 0$ si et seulement si $p \in I$
$\varphi(p \cdot q) = \varphi(p \cdot \varphi(q)) = \varphi(\varphi(p) \cdot \varphi(q))$

Un moyen pour se donner une telle forme normale est de calculer une base de Gröbner G de I pour l'ordre admissible $<$ et ensuite de prendre $\varphi(p) = \mathrm{NF}(p, G, <)$. Une solution alternative est d'utiliser les formes normales généralisées (voir (Mourrain & Trébuchet, 2005)). Dans la suite on note $\varphi(p) = \mathrm{NF}(p)$ cette fonction de forme normale.

Pour optimiser le calcul on va exploiter la structure de $\mathcal{F}(G)$ donnée par la proposition 3.1.2: on va calculer uniquement des formes normales de la forme $\varphi(x_i \cdot p)$ où p est déjà réduit. On considère donc les applications linéaires de multiplication par une variable :

$$\phi_i : f \to \varphi(x_i f)$$

Definition 3.1.3. *On définit les matrices de multiplication par une variable: pour tout* $1 \leq i \leq n$ *on définit la matrice* $M^{(k)}$ *de taille* $\deg(I) \times \deg(I)$ *telle que:*

$$M^{(k)}_{i,j} = \text{ le coefficient } e_i \text{ dans } x_k\, e_j$$

Afin de décrire l'algorithme permettant de calculer les matrices de multiplication par une variable, on utilise les notations: $\delta_{i,j}$ est le symbole de Kronecker et vaut 1 si $i = j$ et 0 sinon; si M est une matrice alors $\mathrm{Col}(M, j)$ désigne la $j$ème colonne de M.

Algorithme 15 *Matrices de multiplications*

Input: *G une base de Gröbner réduite pour l'ordre $<$*
$\mathcal{E}(G) = \{e_1 = 1 < e_2 < \cdots < e_{\deg(I)}\}$ base canonique pour G.
$N := []$ // une table de polynômes indexée par T
// et vérifiant pour tout $t \in T : N[t] = \mathrm{NF}(t, G, <)$
for *i* ***from*** *1* ***to*** *$\deg(I)$* ***do***
  $N[e_i] := e_i$
  for each *k tel que $e_i = x_k\, e_j$* ***do***
    $M_{l,j}^{(k)} := \delta_{l,i}$ pour tout $l \in \{1, \ldots, n\}$
$F := [x_j\, e_i$ pour $j = 1, \ldots, n$, $i = 1, \ldots, \deg(I)]$
trier F pour $<$, éliminer les doublons et les éléments de $\mathcal{E}(G)$:
for *t* ***in*** *F* ***do***
  if *t est un multiple strict d'un terme de tête de G*
    $t = x_j\, t'$ avec $t' < t$
    On a déjà calculé $N[t'] = \sum_{i=1}^{s} \mu_i e_i$ avec $\mu_i \in \mathbb{K}$ et $e_s < t'$
    $N[t] = \sum_{i=1}^{s} \mu_i \,\mathrm{Col}(M^{(j)}, i) = \sum_{i=1}^{\deg(I)} \lambda_i e_i$
    for each *k tel que $t = x_k\, e_l$ pour un certain l* ***do***
      $M_{i,j}^{(k)} := \lambda_i$ pour tout $i \in \{1, \ldots, n\}$
  else
    il existe $g = t + \sum_{i=1}^{\deg(I)} \lambda_i e_i$ et $\lambda_i \in \mathbb{K} \in G$ tel que $t = \mathrm{LT}(g)$
    $N[t] := -\sum_{i=1}^{\deg(I)} \lambda_i\, e_i$
    for each *k tel que $t = x_k\, e_j$ pour un certain j* ***do***
      $M_{i,j}^{(k)} := -\lambda_i$ pour tout $i \in \{1, \ldots, n\}$
return *$M^{(k)}$ // matrice de multiplication par x_k*

Theorem 3.1.1. *L'algorithme 15 calcule les matrices $M^{(k)}$ et la complexité arithmétique est bornée par $O(n \deg(I)^3)$.*

Proof. Pour montrer la correction de l'algorithme il suffit de montrer que dans l'expression $N[t] = \sum_{i=1}^{s} \mu_i \,\mathrm{Col}(M^{(j)}, i)$ la colonne i de la matrice $M^{(j)}$ a déjà été calculée auparavant. Comme $N[t'] = \sum_{i=1}^{s} \mu_i e_i$ avec $e_s < t'$ on a $\mathrm{NF}(t) = \mathrm{NF}(x_j\, t') = \sum_{i=1}^{s} \mu_i\ \mathrm{NF}(x_j\, e_i)$, comme l'ordre est admissible $x_j e_i < x_j e_s < x_j\, t' = t$. Donc tous les termes $x_j\, e_i$ ont été traité et la colonne i de la matrice $M^{(j)}$ a donc été remplie.

Dans l'algorithme il est clair que F est la frontière $\mathcal{F}(G)$ et donc le nombre d'itération dans la boucle principale est bornée par $n\, D\,(I)$; de plus le calcul de $\sum_{i=1}^{s} \mu_i \,\mathrm{Col}(M^{(j)}, i)$ nécessite au plus $s \deg(I) \leq \deg(I)^2$ opérations.

3.1.4 Description de FGLM

3.1.5 Algorithme FGLM

Dans l'algorithme suivant si S est une liste alors:

- $\#S$ désigne le nombre d'éléments de S

- first(S) est le premier élément de la liste ou $\emptyset$ si S est vide.

Algorithme 16 *FGLM*

Input: $<_2$ *un ordre admissible et* NF *une forme normale.*
Output: *base de Gröbner réduite de l'idéal* I *pour* $<_2$
où $I = \{f \in \mathbb{K}[x_1, \ldots, x_n] \mid \mathrm{NF}(f) = 0\}$
$L := [\,]$ // *liste des prochains termes à étudier*
$S := [\,]$ // *l'escalier pour le nouvel ordre* $<_2$
$V := []$ // $V = \mathrm{NF}(S)$
$G := [\,]$, $t := 1$
infinite loop
  $v := \mathrm{NF}(t)$ *et* $s := \#S$ *le nombre d'éléments de* S.
  if $v \in \mathrm{Vect}_{\mathbb{K}}(V)$ ***then***
    on peut trouver (λ_i) *t.q.* $v = \sum_{i=1}^{s} \lambda_i \cdot V_i$
    $G := G \cup \left[v - \sum_{i=1}^{s} \lambda_i \cdot S_i \right]$
  else
    $S := S \cup [t]$ *et* $V := V \cup [v]$
    $L := \mathrm{Sort}(L \cup [x_i\, t \mid i = 1, \ldots, n], <_2)$
    Éliminer de L *les doublons et les multiples de* $\mathrm{LT}(G)$
  if $L = \emptyset$ *then*
    return G
  $t := \mathrm{first}(L)$ *et supprime* t *de* L.

Le théorème 3.1.2 prouve que cet algorithme se termine et retourne le bon résultat.

3.1.6 Version matricielle de FGLM

Afin de rendre explicite la détection de la dépendance linéaire des vecteurs du $\mathbb{K}$-espace vectoriel $\mathbb{K}[x_1, \ldots, x_n]/I$ on introduit la matrice de passage P entre l'ancienne base $\mathcal{E}(G) = \left\{e_1 = 1 < e_2 < \cdots < e_{\deg(I)}\right\}$ et la nouvelle base S en cours de construction. Si on note $S = \left[\varepsilon_1, \ldots, \varepsilon_{\deg(I)}\right]$ cet base alors à tout moment de l'algorithme

$$S = P \cdot \mathcal{E}(G)$$

Initialement $S = [e_1]$, et on construit incrémentalement des vecteurs $v = \varphi(x_k\, w) = M^{(k)} \cdot w$ où v, w sont des vecteurs exprimés dans la base $\mathcal{E}(G)$:

$$v = \sum_{i=1}^{\deg(I)} v_i\, e_i$$

Pour tester l'indépendance linéaire il suffit de calculer:

$$\lambda = P \cdot v = \sum_{i=1}^{\deg(I)} \lambda_i \, e_i$$

1. si $\lambda_{s+1} = \cdots = \lambda_{\deg(I)} = 0$ où s est le nombre d'éléments de S alors le vecteur v appartient au $\mathbb{K}$-espace vectoriel engendré par S.
2. s'il existe $k > s$ tel que $\lambda_k \neq 0$ alors $\varepsilon_{s+1} = \lambda$ est un vecteur linéairement indépendant On calcule une nouvelle matrice P' tel que:

$$P' \cdot v =^T [0, \ldots, 0, 1, 0, \ldots, 0] = \varepsilon_{s+1} \tag{3.1}$$

La procédure UPDATE met à jour la matrice P pour que l'équation (3.1) soit vérifiée:

Algorithme 17 UPDATE *Mise à jour matrice de passage P*

Input: $s \in \mathbb{N}$, *le vecteur* λ, *la matrice* P
Output*: la matrice* P *mise à jour*
$k := \min\{j > s \text{ tel que } \lambda_j \neq 0 \}$
for j ***from*** 1 ***to*** $\deg(I)$ ***do***
  $\alpha := \frac{P_{j,k}}{P_{k,k}}, P_{j,k} := P_{s+1,j}, P_{s+1,j} := \alpha$
  if $\alpha \neq 0$ ***then***
    for i ***from*** 1 ***to*** $\deg(I)$ *tel que* $i \neq s+1$ ***do***
      $P_{i,j} := P_{i,j} - \alpha \, \lambda_i$
return P

On peut donc maintenant décrire explicitement l'algorithme FGLM:

Algorithme 18 *Version matricielle de FGLM*

Input: $<$ *un ordre,* $M^{(k)}$ *les matrices de multiplications,* φ *forme normale*
Output: *base de Gröbner réduite pour l'ordre* $<$ *de* $\ker(\varphi)$.
$S := [1]$ *// l'escalier pour le nouvel ordre* $<$.
$V := [e_1]$ *//* $V = \mathrm{NF}(S)$
$L := [(i,1),\ i = 1,\ldots,(n-1)]$ *// liste de paires* (k,l) *correspondant à* $x_i \cdot S_l$
$G := [\,]$, $t := (n,1)$
$P := I_{\deg(I)}$ *matrice de passage entre la nouvelle base* S *et* $\mathcal{E}(G)$
infinite loop
 $s := \#S$ *le nombre d'éléments de* S.
 $t = (k,l)$: *on calcule* $v = M^{(k)} \cdot V_l$ *puis* $\lambda = P \cdot v$
 if $\lambda_{s+1} = \cdots = \lambda_{\deg(I)} = 0$ ***then***
 $G := G \cup \left[x_k S_l - \sum_{i=1}^{s} \lambda_i \cdot S_i\right]$
 else
 $P :=$UPDATE(s, λ, P)
 $S := S \cup [x_k S_l]$ *et* $V := V \cup [v]$
 $L := \mathrm{Sort}(L \cup [(i,s) \mid i = 1,\ldots,n], <)$
 Éliminer de L *les doublons et les multiples de* $\mathrm{LT}(G)$
 if $L = \emptyset$ *then*
 return G
 $t := \mathrm{first}(L)$ *et supprime* t *de* L.

3.1.7 Exemple pas à pas

On se place dans $\mathbb{Q}[x_1, x_2]$ et soit $G_{<_{\mathrm{DRL}}} = \left[x_1^2 - 3\,x_2 - x_1 + 1, x_2^2 - 2\,x_1 + x_2 - 1\right]$ qui est une base de Gröbner pour l'ordre DRL avec $x_2 > x_1$ et on chercher à calculer la base pour l'ordre lexicographique avec $x_2 > x_1$.

$$\mathcal{E}(G) = \{e_1 = 1, e_2 = x_1, e_3 = x_2, e_4 = x_1\,x_2\}$$

On obtient les matrices de multiplication par x_1 et par x_2:

$$M^{(1)} = \begin{array}{c|cccc} & x_1\,e_1 & x_1\,e_2 & x_1\,e_3 & x_1\,e_4 \\ e_1 & 0 & -1 & 0 & 3 \\ e_2 & 1 & 1 & 0 & 6 \\ e_3 & 0 & 3 & 0 & -4 \\ e_4 & 0 & 0 & 1 & 1 \end{array}$$

$$,M^{(2)} = \begin{array}{c|cccc} & x_2\,e_1 & x_2\,e_2 & x_2\,e_3 & x_2\,e_4 \\ e_1 & 0 & 0 & 1 & -2 \\ e_2 & 0 & 0 & 2 & 3 \\ e_3 & 1 & 0 & -1 & 6 \\ e_4 & 0 & 1 & 0 & -1 \end{array}$$

On commence par $L := [(2,1)], S := [1], V := [e_1], G := [\,]\ , t := (1,1)$ correspondant au monôme $1 \cdot S_1 = 1, P := I_4$

Étape 1: On calcule $v := M^{(1)} \cdot V_1 = M^{(1)} \cdot 1 = e_2$ puis $\lambda = P \cdot v = e_2$ comme $\lambda_2 \neq 0$, $S := [1, x_1]$, $V := [e_1, e_2]$ et la matrice P reste inchangée, puis $L := [(1,2),(2,1),(2,2)]$.

Étape 2: $t = (1,2)$. On calcule $v := M^{(1)} \cdot V_2 = M^{(1)} \cdot e_2 =^T [-1,1,3,0]$ puis $\lambda = P \cdot v =^T [-1,1,3,0]$ comme $\lambda_3 \neq 0$, $S := [1, x_1, x_1^2]$, $V := [e_1, e_2, ^T[-1,1,3,0]]$, puis $P := \begin{bmatrix} 1 & 0 & 1/3 & 0 \\ 0 & 1 & -1/3 & 0 \\ 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $L := [(1,3),(2,1),(2,2),(2,3)]$.

Étape 3: $t = (1,3)$. On calcule $v := M^{(1)} \cdot V_3 = M^{(1)} \cdot^T [-1,1,3,0] =^T [-1,0,3,3]$ puis $\lambda = P \cdot v =^T [0,-1,1,3]$ comme $\lambda_4 \neq 0$, $S := [1, x_1, x_1^2, x_1^3]$, $V := [e_1, e_2, V_3, ^T[-1,0,3,3]]$, puis $P := \begin{bmatrix} 1 & 0 & 1/3 & 0 \\ 0 & 1 & -1/3 & 1/3 \\ 0 & 0 & 1/3 & -1/3 \\ 0 & 0 & 0 & 1/3 \end{bmatrix}$, $L := [(1,4),(2,1),(2,2),(2,3),(2,4)]$.

Étape 4: $t = (1,4)$. On calcule $v := M^{(1)} \cdot V_4 = M^{(1)} \cdot^T [-1,0,3,3] =^T [9,17,-12,6]$ puis $\lambda = P \cdot v =^T [5,23,-6,2]$ comme $\lambda_5 = 0$, $G := [x_1^4 - 2\,x_1^3 + 6\,x_1^2 - 23\,x_1 - 5]$, puis $L := [(2,1),(2,2),(2,3),(2,4)]$.

Étape 5: $t = (x_2, 1)$. On calcule $v := M^{(2)} \cdot V_1 = M^{(2)} \cdot e_1 = e_3$ puis $\lambda = P \cdot e_3 =^T \left[\frac{1}{3}, \frac{-1}{3}, \frac{1}{3}, 0\right]$ comme $\lambda_5 = 0$, $G := [x_1^4 - 2\,x_1^3 + 6\,x_1^2 - 23\,x_1 - 5, x_2 - \frac{1}{3}x_1^2 + \frac{1}{3}x_1 - \frac{1}{3}]$, puis en éliminant les multiples de $\mathrm{LT}(G_2) = x_2$, on obtient $L := []$ et l'algorithme FGLM se termine.

3.1.8 Preuve de l'algorithme

Theorem 3.1.2. *Les algorithmes 16 et 18 se terminent et calculent une base de Gröbner Le nombre d'opérations dans $\mathbb{K}$ de l'algorithme 18 est borné par $O(n \deg(I)^3)$.*

Proof. On note I le noyau de la forme normale; I est un idéal et G' la base de Gröbner réduite de I pour l'ordre $<$: $G' = [g'_1, g'_2, \ldots]$ avec $\mathrm{LT}_<(g_1) < \mathrm{LT}_<(g_2) < \cdots$. De plus si $G = [g_1, g_2, \ldots]$ est résultat de l'algorithme S_i désigne la valeur de S au moment ou on ajoute g_i dans G.

Il est clair que les éléments de S sont linéairement indépendants modulo I. De plus tous les éléments de G sont dans l'idéal I: en effet lorsqu'on rajoute le polynôme $p = x_k S_l - \sum_{i=1}^{\#S} \lambda_i \cdot S_i$ on a par construction $\varphi(p) = 0$ et donc $p \in I$ par définition d'une forme normale. On montre par récurrence la propriété suivante:

$H(i)$: $\quad g_i = g'_i$, S_i est la base canonique de $[g_1, \ldots, g_i]$ pour l'ordre $<$.

Si la propriété est vraie pour $H(j), j < i$ alors on note S' la base canonique de $[g_1, \ldots, g_i]$ pour l'ordre $<$ privé de S_{i-1}: pour tout $t \in S'$, t n'est pas

un multiple de $\mathrm{LT}_<([g_1, \ldots, g_i - 1])$; si $t = x_{i_1} x_{i_2} \cdots x_{i_r}$ alors on montre successivement que $t_j = x_{i_1} x_{i_2} \cdots x_{i_j}$ est dans linéairement indépendant et donc qu'on rajoute $t_j \cdot x_k$ dans la liste L et donc en particulier $t_{j+1} \in L$. Par conséquent $S' = S_i$. Soit maintenant $t = \mathrm{LT}(g_i')$; il existe k tel que x_k divise t; comme G' est réduite $t/x_k \in S'$ donc $t/x_k \in L$ et $t \in L$.Si $g_i' = t + \sum \lambda_i t_i$ alors $\varphi(t) = -\sum \lambda_i \varphi(t_i)$ et donc on trouve la relation de dépendance linéaire et on rajoute g_i' dans G. Ceci prouve $g_i = g_i'$ et donc $H(i)$.

Il est clair que l'algorithme 17 se termine et que sa complexité arithmétique est bornée par $O(n \deg(I)^2)$, il suffit donc de prouver la terminaison de l'algorithme principal FGLM. Comme on est dans le cas zéro dimensionnel il y a un nombre fini de termes linéairement indépendants: dans l'algorithme on augmente L seulement lorsqu'on détecte un vecteur linéairement indépendant; par conséquent la taille de L et le nombre d'itérations de l'algorithme est borné par $n \deg(I)$. Les autres opérations arithmétiques sont des produits $M \cdot v$ dont la complexité est bornée par $O(n \deg(I)^2)$.

3.1.9 Complexité de FGLM.

Dans cette section on résume les résultats de complexité obtenus dans ce chapitre. Les résultats sont complètement différents si les opérations arithmétiques se font en temps constant (corps fini de petite taille) ou si on prend en compte la croissance des coefficients. Dans ce dernier il est toujours préférable de calculer une forme RUR ou RR qui permet de limiter la croissance des coefficients dans les calculs et dans le résultat (voir l'annexe 21 et (Rouillier, 1999)). On se réfère à l'article (Faugère, J.C., Gianni, P., Lazard, D. and Mora T., 1993) pour une estimation de la complexité booléenne de l'algorithme FGLM.

Corollary 3.1.2. *Soit I un idéal zéro dimensionnel et $(G_1, <_1)$, une base de Gröbner réduite par rapport à un ordre admissible $<_1$. Étant donné un ordre admissible $<_2$, il est possible de calculer la base de Gröbner du même idéal I par rapport à l'ordre $<_2$ en $O(n \deg(I)^3)$ opérations arithmétiques.*

Proof. En utilisant l'algorithme 15 on calcule en $O(n \deg(I)^3)$ opérations les matrices de multiplications. On peut ensuite appliquer l'algorithme 18 pour effectuer le changement d'ordre.

En combinant ce résultat avec des résultats de complexité pour le calcul d'une base de Gröbner pour un ordre DRL on obtient le résultat général:

Theorem 3.1.3. *Soit I un idéal de dimension zéro engendré par des polynômes en n variables de degré au plus d. Si les générateurs de I ont un nombre fini de solutions à l'infini alors on peut calculer la base de Gröbner de I pour n'importe quel ordre en temps polynomial en d^n. Si l'ensemble des zéros à l'infini est infini, alors la complexité est polynomiale en d^{n^2}.*

Proof. Si le nombre de solutions à l'infini est fini alors (Lazard D., 1983) montre que le calcul de la base de Gröbner par l'algorithme de Buchberger se fait en temps polynomial en d^n; le même résultat est obtenu avec les algorithmes F_4 ou F_5 (voir le chapitre 6). On applique ensuite l'algorithme FGLM pour calculer la base pour n'importe quel ordre. Le résultat résulte de la borne $O(n \deg(I)^3)$ (du corollaire 3.1.2) et de la borne de Bezout $\deg(I) \leq d^n$. Si on prend en compte la croissance des coefficients voir la preuve dans (Faugère, J.C., Gianni, P., Lazard, D. and Mora T., 1993)).

Note 3.1.2. Pour des résultats plus récent sur la complexité du calcul d'une base de Gröbner en dimension 0 on pourra consulter la thèse (Hashemi, 2006).

3.1.10 Benchmarks FGLM

L'algorithme FGLM est implanté dans tous les systèmes de Calcul Formel: Mathematica, Maple, Magma, Singular, ... Il existe aussi une implantation très efficace dans le logiciel FGb.

Benchmarks classiques. Les tests suivant ont été effectués en 2006 afin de comparer l'efficacité du calcul d'une base de Gröbner pour l'ordre lexicographique. Un test complet devrait inclure plusieurs centaines d'exemples. C'est donc une étude en miniature qui figure dans la table 3.2.

Dans la table suivante on reporte les temps de calculs pour l'exemple Katsura_N (voir la définition et un exemple 2.9.4):

- `F4/FGb` le temps de calcul de la base de Gröbner modulo p pour l'ordre DRL avec l'algorithme F_4 implanté dans FGb.
- `F4/Magma` le temps de calcul de la base de Gröbner modulo p pour l'ordre DRL avec l'algorithme F_4 implanté dans Magma.
- `F5` le temps de calcul de la base de Gröbner modulo p pour l'ordre DRL avec l'algorithme F_5 implanté dans FGb.
- `Buch/FGb` le temps de calcul de la base de Gröbner modulo p pour l'ordre DRL avec l'algorithme de Buchberger implanté dans Gb.
- `Lex Magma` le temps de calcul de la base de Gröbner modulo p pour l'ordre lexicographique avec Magma (F_4+FGLM).
- `Slim Gb lex` le temps de calcul de la base de Gröbner modulo p pour l'ordre lexicographique avec Singular et l'algorithme SlimGb.
- `FGLM FGb` le temps de calcul de la base de Gröbner modulo p pour le changement d'ordre vers l'ordre lexicographique avec FGb (algorithme FGLM uniquement).
- INT `Lex Magma` le temps de calcul de la base de Gröbner sur $\mathbb{Q}$ pour l'ordre lexicographique avec Magma (F_4+FGLM).
- INT `RR` le temps de calcul de la base de Gröbner sur $\mathbb{Q}$ pour l'ordre lexicographique (forme RR) avec FGb (F_4+FGLM).

	Calcul Modulo p							Calcul sur Q	
N	F4/FGb	F4/Magma	F5	Buchb/Gb	LexMagma	SlimGble x	FGLM/FGb	INT LexMagma	INT RR
6	0,0			0,3		0,2		7,9	0,4
7	0,0		0,0	2,4	0,5	> 65 0	0,0	308, 2	2,4
8	0,3	0,2	0,1	20, 1	1,4		0,1	12928, 6	11,0
9	1,8	1,1	0,4	186, 9	7,9		0,4		156,8
10	14, 6	8,3	2,2		58, 6		3,0		2424,0
11	106, 6	52, 4	14, 3		437, 7		21, 8		38785,0
12	881, 2	416, 2	99, 5		3482, 3		160, 8		112h
13	6319, 7	3037, 9	616, 2				1180, 5		1 328h
14			5943, 3				9141,5		

Fig. 3.2. Changement d'ordre pour le problème Katsura.

On reporte ensuite sur les figures 3.3 et 3.4 suivantes le gain entre l'implantation Magma et l'implantation FGb: par exemple pour Katsura 8 sur $\mathbb{Q}$ on observe un gain de $\frac{38735}{11.0} \approx 1175$.

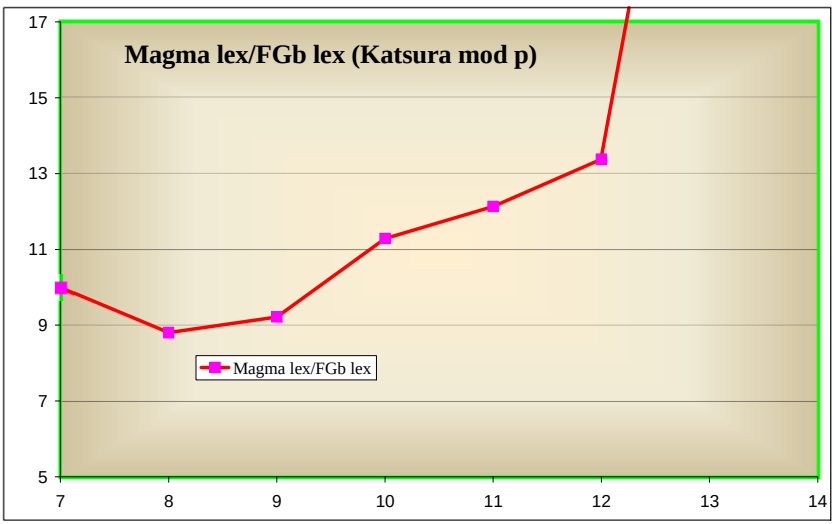


Fig. 3.3. Rapport: Magma/FGb calculer de la base Lexicographique modulo p.

Applications en Cryptologie: Flurry. 3.1.10L'idée des articles (J.Buchmann *et al.* , 2005; J.Buchmann *et al.* , 2006) est pour un crypto-système donné de donner une mise en équation algébrique du problème et un ordre admissible (au sens des bases de Gröbner) sur les variables de telle sorte qu'on obtient sans aucun calcul une base de Gröbner (par application du critère des termes de têtes étrangers ou critère numéro 1 de Buchberger). L'idée des articles (J.Buchmann *et al.* , 2005; J.Buchmann *et al.* , 2006) est pour un crypto-système donné de donner une mise en équation algébrique du problème et un ordre admissible (au sens des bases de Gröbner) sur les variables

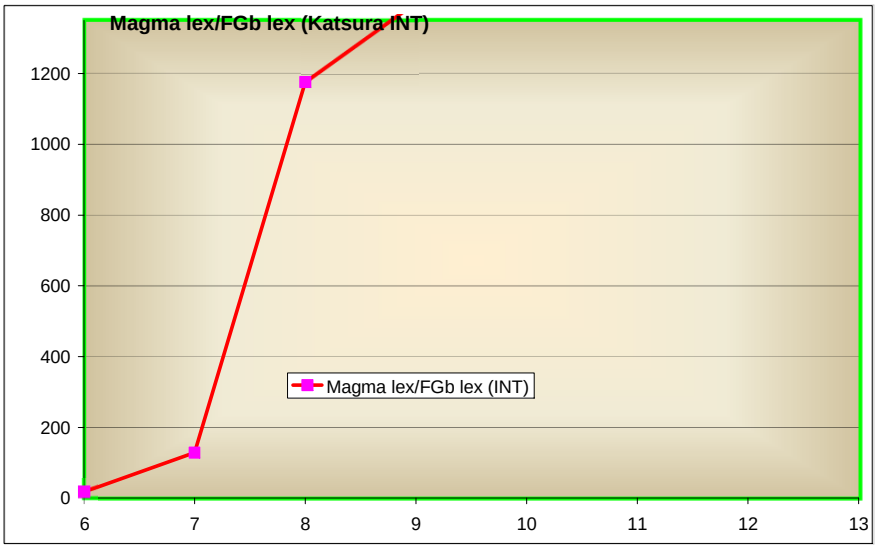


Fig. 3.4. Rapport: Magma/FGb calculer de la base Lexicographique $\mathbb{Q}$.

de telle sorte qu'on obtient sans aucun calcul une base de Gröbner (par application du critère numéro 1 de Buchberger 2.5.1).

On donne un exemple de système qui est déjà une base de Gröbner pour le shéma de Feistel **Flurry,** dont on donne une description sommaire (J.Buchmann *et al.* , 2005).

$\mathbb{K}$ est le corps fini à 2^k éléments et ω un générateur de $\mathbb{K}$ sur le corps $\mathbb{F}_2$: $\mathbb{K} = \mathbb{F}_2(w)$.

R est l'anneau des polynômes à coefficients dans $\mathbb{K}$.

La famille de Feistel **Flurry**(k, m, r, f, D) est définie par:

- k la taille du corps fini.
- m la taille d'un demi-bloc, dans la suite on note aussi $t = 2\,m$ la taille de bloc.
- r le nombre de tours.
- f la fonction `Sbox`. En pratique $f(x) = f_p(x) = x^p$ ou $f(x) = f_{\text{inv}} = x^{k-2}$ c'est-à-dire l'inverse dans K.
- D une matrice $m \times m$ à coefficients dans $\mathbb{K}$ (en pratique une matrice MDS).

On note un demi-état gauche (resp. droit) $L = [l_1, \ldots, l_m]$ (resp. $R = [r_1, \ldots, r_m]$) et une clé est de la forme $K = [k_1, \ldots, k_m]$. On définit alors la fonction $\rho : K^m \times K^m \times K^m \to K^m \times K^m$ par

$$\rho(L, R, K) = (R, D.^T[f(r_1 + k_1), \ldots, f(r_m + k_m)])$$

Partant d'une clé initiale secrète de taille $t = 2\,m$ $[K_0, K_1]$ on calcule les clés qui vont servir dans les autres tours:

$$K_i = D.{}^T K_{i-1} + {}^T K_{i-2} + v_i \; i = 2, 3, \ldots, r+1$$

où v_i est la constante:

$$v_i = {}^T\left[(w+1)^i, (w+1)^{i+1}, \ldots, (w+1)^{i+m-1}\right]$$

Partant d'un message initial de taille t $[L_0, R_0]$ on l'encode en itérant l'appel à la fonction ρ:

$$(L_i, R_i) = \rho(L_{i-1}, R_{i-1}, K_{i-1}) \; i = 1, 2, \ldots, r-1$$
$$(L_r, R_r) = \rho(L_{r-1}, R_{r-1}, K_{r-1}) + (K_r, K_{r+1})$$

L'exemple suivant correspond aux paramètres $(2, 4, f_3, D_4)$: dans les équations suivantes $x_{i,r}$ représente le message initial après r tours de chiffrement et $K_{i,r}$ est la clé obtenue au tour r. On suppose qu'on connaît $[x_{1,0}, x_{2,0}, x_{3,0}, x_{4,0}]$ le message initial en clair et le $[x_{1,4}, x_{2,4}, x_{3,4}, x_{4,4}]$ le message chiffré; on cherche $[K_{1,0}, K_{2,0}, K_{3,0}, K_{4,0}]$ la clé secrète (dans l'écriture suivante les polynômes sont triés pour l'ordre DRL afin de faire ressortir la structure):

Message clair:

$$x_{4,0} - 10333,$$
$$x_{3,0} + 20618,$$
$$x_{2,0} + 13990,$$
$$x_{1,0} + 23979,$$

Message chiffré:

$$x_{4,4} + 28672,$$
$$x_{3,4} + 7942,$$
$$x_{2,4} - 30797,$$
$$x_{1,4} + 18239,$$

Cadencement de clé:

$$K_{1,1} - 21840\,K_{2,3} + 21840\,K_{1,3} - 21840\,K_{1,0} - 15,$$
$$K_{2,1} - 21841\,K_{2,3} - 21840\,K_{1,3} - 21840\,K_{2,0} + 54,$$
$$K_{1,2} - K_{2,1} - 2\,K_{1,1} - K_{1,0} - 9,$$
$$K_{2,2} - K_{2,1} - K_{1,1} - K_{2,0} - 27,$$
$$K_{1,4} - K_{1,2} - K_{2,3} - 2\,K_{1,3} - 81,$$
$$K_{2,4} - K_{2,2} - K_{2,3} - K_{1,3} - 243,$$

Premier tour:

$$
\begin{aligned}
&x_{1,1} - x_{3,0},\\
&x_{2,1} - x_{4,0},\\
&K_{2,0}^3 + 3\,K_{2,0}^2\,x_{4,0} + 3\,K_{2,0}\,x_{4,0}^2 + x_{4,0}^3 - 2\,x_{4,1} + x_{3,1} + 2\,x_{2,0} - x_{1,0},\\
&K_{1,0}^3 + 3\,K_{1,0}^2\,x_{3,0} + 3\,K_{1,0}\,x_{3,0}^2 + x_{3,0}^3 + x_{4,1} - x_{3,1} - x_{2,0} + x_{1,0},
\end{aligned}
$$

Deuxième tour:

$$
\begin{aligned}
&x_{1,2} - x_{3,1},\\
&x_{2,2} - x_{4,1},\\
&x_{3,1}^3 + 3\,x_{3,1}^2\,K_{1,1} + 3\,x_{3,1}\,K_{1,1}^2 + K_{1,1}^3 + x_{4,2} - x_{3,2} - x_{2,1} + x_{1,1},\\
&x_{4,1}^3 + 3\,x_{4,1}^2\,K_{2,1} + 3\,x_{4,1}\,K_{2,1}^2 + K_{2,1}^3 - 2\,x_{4,2} + x_{3,2} + 2\,x_{2,1} - x_{1,1},
\end{aligned}
$$

Troisième tour:

$$
\begin{aligned}
&x_{1,3} - x_{3,2},\\
&x_{2,3} - x_{4,2},\\
&x_{3,2}^3 + 3\,x_{3,2}^2\,K_{1,2} + 3\,x_{3,2}\,K_{1,2}^2 + K_{1,2}^3 - x_{2,2} + x_{1,2} + x_{4,3} - x_{3,3},\\
&x_{4,2}^3 + 3\,x_{4,2}^2\,K_{2,2} + 3\,x_{4,2}\,K_{2,2}^2 + K_{2,2}^3 + 2\,x_{2,2} - x_{1,2} - 2\,x_{4,3} + x_{3,3},
\end{aligned}
$$

Quatrième tour:

$$
\begin{aligned}
&x_{4,3} - x_{2,4},\\
&x_{3,3} - x_{1,4},\\
&K_{1,3}^3 + 3\,K_{1,3}^2\,x_{3,3} + 3\,K_{1,3}\,x_{3,3}^2 + x_{3,3}^3 - x_{2,3} + x_{1,3} + x_{4,4} - x_{3,4},\\
&K_{2,3}^3 + 3\,K_{2,3}^2\,x_{4,3} + 3\,K_{2,3}\,x_{4,3}^2 + x_{4,3}^3 + 2\,x_{2,3} - x_{1,3} - 2\,x_{4,4} + x_{3,4}
\end{aligned}
$$

Le tableau suivant 3.1.10 montre qu'avec un bon algorithme de changement d'ordre la partie dominante est le calcul la base de Gröbner pour un ordre DRL. (Le tableau contient également une comparaison avec un autre algorithme de changement d'ordre dont le nom est le Gröbner Walk: la complexité théorique est difficile à établir avec précision). En effet le temps de résolution est le la somme du temps pour faire le calcul de la base de Gröbner pour un ordre du degré (colonnes de droites) et le temps nécessaire pour effectuer un changement d'ordre (colonnes du milieu).

Flurry	Dim Quotient	Magma FGLM	FGb FGLM	Magma Walk	FGb F_5	FGb Hilbert
$m=1, r=4, x^5$	625	8.9s	0.6 s	×	60.3 s	67.5 s
$m=1, r=5, x^3$	243	0.96s	0.07s	1.3s	6.6 s	7.3s
$m=1, r=6, x^3$	729	22.2s	1.5s	21.2s	×	×
$m=1, r=10, x^{-1}$	221	57.1s	0.23s	×	×	×
$m=1, r=12, x^{-1}$	596	×	2.7s	×	6.3 s	7.2s
$m=3, r=4, x^{-1}$	583	×	7.7s	×	×	×

Table 3.1. Comparaison FGLM sur le cryptosysteme Flurry

3.2 LLL pour calculer des bases de Gröbner

Travail réalisé en collaboration avec A. Basiri (Basiri & Faugère, 2003b; Basiri, 2003).

3.2.1 Introduction

. On a vu qu'une base de Gröbner permettait de calculer les polynômes les plus petits pour la relation de divisibilité (voir la section 2.4.1 page 24) . D'autre part l'algorithme LLL (Lenstra *et al.* , 1982) est un algorithme bien connu pour calculer des vecteurs courts dans un réseau (mais pas nécessairement les *plus petits*). Il est donc naturel de comparer les deux algorithmes. Il est possible de modifier LLL (voir (Von Zur Gathen, 1984; Paulus, 1998; v.-z. Gathen & Gerhard, 1999) et la description de l'algorithme 21) en remplaçant les entiers par des polynômes en une variable: la taille d'un élément n'est plus sa norme euclidienne mais le degré du polynôme. À noter que cette version polynomiale de LLL se distingue de la version sur les entiers par le fait que l'algorithme devient *exact:* le résultat de l'algorithme est donc réellement le vecteur le plus court. Un avantage éventuel de LLL est que la complexité de cet algorithme dépend fortement de la taille du résultat. On peut donc espérer un bon comportement de l'algorithme LLL lorsque le résultat attendu est petit. L'algorithme LLL est donc utilisé comme l'autre algorithme de ce chapitre, FGLM, pour calculer une base de Gröbner pour un ordre admissible quelconque à partir d'une base de Gröbner calculée pour un autre ordre. L'algorithme présenté ici ((Basiri & Faugère, 2003a)) se restreint au cas particulier de deux variables mais est valide aussi en dimension positive. Pour cela il a été nécessaire de généraliser la version polynomiale de l'algorithme LLL existante (Von Zur Gathen, 1984; Paulus, 1998; v.-z. Gathen & Gerhard, 1999, exercice 16.12 page 474) pour remplacer l'ordre du degré par n'importe quel ordre admissible et pour remplacer les polynômes en une variable par des polynômes en deux variables. La complexité théorique dans le cas le pire de cet algorithme n'est pas meilleure que l'algorithme de FGLM, $O(n \deg(I)^3)$, mais en pratique si le degré des polynômes en une des deux variables est petit l'algorithme devient plus rapide que FGLM. On peut aussi donner une borne de complexité $O(D_Y^3 D_X^2)$ qui dépend de la taille du résultat (D_X est le maximum des degrés en X du résultat et D_Y, le degré en Y, dépend de l'entrée et de la sortie). On peut aussi citer une application en Cryptologie de cette correspondance entre l'algorithme FGLM et LLL: l'idée sera de remplacer dans l'implantation de l'arithmétique dans la jacobienne de courbes générales (courbes $C_{a,b}$) (Basiri *et al.* , 2002b); la méthode proposée

par (Galbraith *et al.* , 2002; Harasawa & Suzuki, 2000) l'utilisation de LLL par FGLM et donc d'obtenir des formules très efficaces pour certaines courbes (dont le genre est petit). Cette application est décrite dans le chapitre 11. Plus généralement l'algorithme LLL original est très utilisé en Cryptologie (voir (Joux & Stern, 1998; Nguyen & Stern, 2000; Nguyen & Stern, 2001)).

Le plan de cette section est le suivant: en 3.2.3 on présente la version modifiée de LLL dont la preuve de correction et de terminaison est donnée en 3.2.5. On analyse la complexité théorique en 3.2.6 et le comportement en pratique est donné en 3.2.7 sous forme de comparaison avec l'implantation de FGLM dans Maple.

On garde le nom LLL pour cet algorithme. Dans le reste du chapitre LLL désigne la version modifiée de l'algorithme LLL original (Lenstra *et al.* , 1982).

3.2.2 Définitions. Réseaux.

Dans le reste du chapitre $\mathbb{K}[X,Y]$ est l'anneau des polynômes en deux variables à coefficients dans $\mathbb{K}$ et $I = \mathrm{Id}(f_1,\ldots,f_m)$ un idéal de $\mathbb{K}[X,Y]$ engendré par $f_1,\ldots,f_m$. Un ordre admissible $<$ est fixé (voir 2.3.1 pour les définitions de base sur les base de Gröbner).

Polynômes dans $\mathbb{K}[X,Y]$ et $\mathbb{K}[X][Y]$. Si f est un polynôme de $\mathbb{K}[X,Y]$ alors $\deg_Y(f)$ (respectivement $\deg_X(f)$) est le degré en Y (resp. X) de f. Tout polynôme $f \in \mathbb{K}[X,Y]$ peut se décomposer en

$$f(X,Y) = \sum_{j=1}^{D} f_j(X)\, Y^{j-1} \in \mathbb{K}[X][Y]$$

où $D = \deg_Y(f)$ et $f_j(X) = \mathrm{coeff}_Y(f, j-1)$ désigne le coefficient de Y^{j-1} dans $f(X,Y)$. Dans l'anneau des polynômes $\mathbb{K}[X][Y]$ un monôme est donc

$$c(X)\, Y^k$$

où $c(X)$ est un polynôme de $\mathbb{K}[X]$; on note $\mathcal{M}(Y)$ (respectivement $\mathcal{M}_d(Y)$ l'ensemble des monômes dans $\mathbb{K}[X][Y]$ (resp. l'ensemble des monômes de degré d dans $\mathbb{K}[X][Y]$). En notant $\mathcal{M}^{(D)}(Y) = \mathcal{M}_0(Y) \times \cdots \times \mathcal{M}_{D-1}(Y)$ on a alors l'injection naturelle pour tout entier $D > 0$:

$$\pi : \begin{pmatrix} \mathcal{M}^{(D)}(Y) \longrightarrow \mathbb{K}[X,Y] \\ \boldsymbol{v} = [v_1(X),\ldots,v_D(X)] \longmapsto v = \sum_{j=1}^{D} v_j(X) \end{pmatrix}.$$

Réciproquement on a:

$$\pi_1 : \begin{pmatrix} \mathbb{K}[X,Y] \longrightarrow \mathcal{M}^{(D)}(Y) \\ v(X,Y) \longmapsto \boldsymbol{v} = \left[\mathrm{coeff}_Y(v,0),\ldots,\mathrm{coeff}_Y(v,D-1)\, Y^{D-1}\right] \end{pmatrix}.$$

Si $\boldsymbol{v}$ est un vecteur de $\mathcal{M}_0(Y) \times \cdots \times \mathcal{M}_{D-1}(Y)$ on définit $\|\boldsymbol{v}\| = \mathrm{LT}_<(\pi(\boldsymbol{v}))$. Pour alléger les notations, on écrit $\mathrm{LT}_<(\boldsymbol{v})$ à la place de $\mathrm{LT}_<(\pi(\boldsymbol{v}))$. Pour un tel vecteur $\boldsymbol{v}$ on note v_j sa j ème composante et $\boldsymbol{v} = [v_1, \ldots, v_D]$. C'est donc une généralisation de la norme définie dans (Von Zur Gathen, 1984) où $\|\boldsymbol{v}\|$ etait définie par la formule $\|\boldsymbol{v}\|_\infty = \max\{\deg(v_i) \mid 1 \leq i \leq D\}$ pour $\boldsymbol{v} = (v_1, \ldots, v_D) \in \mathbb{K}[X]^D$.

Definition 3.2.1. *Soit $G = (g_1, \ldots, g_m)$ une base de Gröbner réduite de I pour un ordre admissible $<$, on note r_i, le degré de $\mathrm{LT}(g_i)$ par rapport à Y. On peut toujours supposer que les indices $(1, \ldots, m)$ sont triés de sorte que $r_i \leq r_{i+1}$. Pour tout entier $D \geq \deg_Y(G)$, on définit l'ensemble des polynômes:*

$$B_D(G) = \{Y^{j_i} g_i \mid 1 \leq i \leq m,\ \ 0 \leq j_i \leq r_{i+1} - r_i - 1\} \subset \mathcal{M}(Y)$$

où $r_{m+1} := D$. On note $M_D(G)$, le $\mathbb{K}[X]$ sous-module de $\mathbb{K}[X, Y]$ engendré par $B_D(G)$; on dit que $M_D(G)$ est le D-ème $\mathbb{K}[X]$ module associé à l'idéal I par rapport à $<$. Dans ce cas $B_D(G)$ est nommée la base associée à l'idéal G en degré D en tant que $\mathbb{K}[X]$ module.

L'algorithme suivant permet d'extraire une base en tant que $\mathbb{K}[X]$ module d'une base de Gröbner:

Algorithme 19 ModuleBasis

Input: *$G = (g_1, \ldots, g_m)$ base de Gröbner réduite pour $<$ D, un entier positif*
Output: *$B_D(G)$*
$r_i := \deg_Y(g_i)$ pour $i = 1, \ldots, m$ et $r_{m+1} := D$
Permuter les indices $(1, \ldots, m)$ pour que $r_i < r_{i+1}$
$k := 0$
for *i* ***from*** *1* ***to*** *m* ***do***
  for *j* ***from*** *0* ***to*** *$r_{i+1} - r_i - 1$* ***do***
    $k := k + 1$
    $\boldsymbol{b_k} := \pi_1(\mathrm{LT}_<(Y^j g_i) + \mathrm{NF}(Y^j g_i - \mathrm{LT}_<(Y^j g_i), G, <))$
return *$(\boldsymbol{b_1}, \ldots, \boldsymbol{b_k})$*

Réseaux. Minimum successifs.. On donne maintenant les définitions élementaires concernant les réseaux.

Definition 3.2.2. *Soient $\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}$ des vecteurs dans $\mathcal{M}^{(D)}(Y)$ où l et D sont des entiers positifs et $l \leq D$. Le* réseau *(lattice en anglais) $L \subset \mathcal{M}^{(D)}(Y)$ engendré par $\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}$ est défini par*

$$L = \sum_{i=1}^{l} \mathbb{K}[X]\boldsymbol{b_i} = \left\{ \sum_{i=1}^{l} \lambda_i(X)\boldsymbol{b_i} \mid \lambda_i(X) \in \mathbb{K}[X],\ \ 1 \leq i \leq l \right\}.$$

Ainsi si $L \subset \mathcal{M}^{(D)}(Y)$ est un réseau, alors $\pi(L)$ est le $\mathbb{K}[X]$-sous module $M(L)$ de $K[X,Y]$ défini par:

$$M(L) = \left\{ v = \sum_{j=1}^{D_Y} v_j \mid \boldsymbol{v} = (v_1, \ldots, v_{D_Y}) \in L \right\}.$$

Definition 3.2.3. *Soit $b_1, \ldots, b_l$ une base du $K[X]$-sous module $M(L)$ de $\mathbb{K}[X,Y]$ et soit $\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}$ la base correspondante du réseau L. On note $B = (b_{i,j}(X,Y))_{i,j}$ la matrice de taille $l \times D$ où $b_{i,j}(X,Y)$ est la j ème composante de $\boldsymbol{b_i} = [b_{i,1}(X,Y), \ldots, b_{i,D}(X,Y)]$. On peut alors définir le déterminant $\det(M(L))$ de $M(L)$ comme étant le maximum, pour l'ordre admissible $<$, des déterminants des sous-matrices $l \times l$ de B ($l \leq D$). On définit ensuite le déterminant $\det(L)$ de L comme étant le déterminant $\det(M(L))$ de $M(L)$. La valeur de $\det(L)$ ne dépend pas du choix de la base de L.*

Definition 3.2.4. *le* défaut d'orhtogonalité $\mathrm{OD}(\boldsymbol{b_1}, \ldots, \boldsymbol{b_l})$ *de la base $\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}$ du réseau L par rapport à. $<$, est défini comme étant*

$$\mathrm{OD}(\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}) = \mathrm{LT}_<(\boldsymbol{b_1}) \cdot \mathrm{LT}_<(\boldsymbol{b_2}) \cdot \cdots \cdot \mathrm{LT}_<(\boldsymbol{b_l}) - \mathrm{LT}_<(\det(L)).$$

Definition 3.2.5. *On dit que la base $\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}$ du réseau L est réduite si*

$$\mathrm{OD}(\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}) = 0.$$

Pour $1 \leq i \leq l$, le i-ème *minimum successif* de $M(L)$ par rapport à $<$ est un élément minimum m_i de $M(L)$, tel que m_i n'appartient pas au $\mathbb{K}[X]$ sous module de $M(L)$, engendré par $m_1, \ldots, m_{i-1}$. On remarque que m_i est indépendant du choix de $m_1, \ldots, m_{i-1}$. Voir (Mahler, 1941). Contrairement à la définition sur $\mathbb{Z}$ les éléments d'une base réduite sont aussi minimaux:

Proposition 3.2.1. *Soit $\boldsymbol{b_1}, \ldots$ une base réduite du réseau $L \subset \mathcal{M}^{(D)}(Y)$ avec $l \leq D$ telle que $\mathrm{LT}_<(\boldsymbol{b_i}) \leq \mathrm{LT}_<(\boldsymbol{b_{i+1}})$ pour $1 \leq i < l$. Alors $\boldsymbol{b_i}$ est un i-ème minimum successif de $M(L)$ par rapport à $<$ pour $1 \leq i \leq l$.*

Proof. Voir (Lenstra, 1985).

Proposition 3.2.2. *Soit $<$ un ordre admissible et $\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}$ une base du réseau $L \subset \mathcal{M}^{(D)}(Y)$ avec $l \leq D$. Si on peut réordonner les indices de $\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}$ pour que:*

(i) $\mathrm{LT}_<(\boldsymbol{b_1}) \leq \mathrm{LT}_<(\boldsymbol{b_2}) \leq \cdots \leq \mathrm{LT}_<(\boldsymbol{b_l})$
(ii) $b_{i,j} \leq b_{i,i}$ *pour* $1 \leq j \leq D$ *avec une inégalité stricte si* $j < i$, *pour* $1 \leq i \leq l$

alors la base $\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}$ est réduite.

Proof. (Voir (Von Zur Gathen, 1984; Paulus, 1998)) Par exemple si $l = D$ soit $B = (b_{i,j})_{i,j}$ la matrice associée à $\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}$, alors:

$$\det(B) = \sum_{\sigma \in S_l} \operatorname{sgn}(\sigma) \prod_{j=1}^{l} \left(b_{\sigma(j),j}\right)$$

pour chaque terme $\prod_{j=1}^{l} b_{\sigma(j),j}$ est inférieur à $\prod_{j=1}^{l} b_{j,j}$ pour l'ordre admissible; il est facile de vérifier que la condition (ii) implque que l'inégalité est stricte si σ n'est pas l'identité. Donc $\det(B) = \mathrm{LT}_<(b_1) \cdots \mathrm{LT}_<(b_l)$ et donc $\mathrm{OD}(\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}) = 0$.

Theorem 3.2.1. *Soient $G = (g_1, \ldots, g_m)$ une base de Gröbner réduite de l'idéal I pour l'ordre admissible $<$ et D un nombre entier positif supérieur à $\deg_Y(G) + 1$ et I_D l'ensemble de tous les polynômes de I dont le degré en Y est inférieur à D, alors $M_D(G) = I_D$.*

Proof. L'inclusion $M_D(G) \subseteq I_D$ est évidente. Si on avait $M_D(G) \neq I_D$, soit h le plus petit polynôme (par rapport à $<$) dans I_D qui n'appartient pas à $M_D(G)$. Soit $\mathrm{LT}(h) = X^s Y^r$ et $\mathrm{LT}(g_i) = X^{s_i} Y^{r_i}$ pour $1 \leq i \leq m$. En renumérotant G, on peut toujours supposer $r_i \leq r_{i+1}$. Puisque G est réduite nous avons $r_i < r_{i+1}$ et $s_i > s_{i+1}$ pour $1 \leq i \leq m-1$ (en effet si $r_i = r_{i+1}$ pour un $1 \leq i \leq m-1$ alors $\mathrm{LT}(g_i) \mid \mathrm{LT}(g_{i+1})$ ou $\mathrm{LT}(g_{i+1}) \mid \mathrm{LT}(g_i)$, et si $s_i \leq s_{i+1}$ alors $\mathrm{LT}(g_i) \mid \mathrm{LT}(g_{i+1})$ car $r_i < r_{i+1}$). Soit

$$i_0 = \max \{ i \leq m \mid \mathrm{LT}(g_i) \mid \mathrm{LT}(h) \}.$$

On veut montrer que $i_0 = m$ ou $r_{i_0} \leq r < r_{i_0+1}$. Si ce n'était pas vrai alors: $i_0 < m$ et $r_{i_0} < r_{i_0+1} \leq r$; mais comme $s_{i_0+1} < s_{i_0} \leq s$, nous aurions donc $\mathrm{LT}(g_{i_0+1}) \mid \mathrm{LT}(h)$. Absurde.

On pose $b = Y^{r-r_{i_0}} g_{i_0}$. Si $r_{i_0} \leq r < r_{i_0+1}$ alors $0 \leq r - r_{i_0} \leq r_{i_0+1} - r_{i_0} - 1$ et donc $b \in B_D(G)$. Si $i_0 = m$ alors $0 \leq r - r_{i_0} = r - r_m \leq D - r_m$ (car $h \in I_{D_Y}$) et la encore $b \in B_D(G)$. On définit:

$$h' = h - \frac{\mathrm{LM}(h)}{\mathrm{LM}(b)} b = h - \frac{\mathrm{LC}(h) X^s Y^r}{\mathrm{LC}(b) X^{s_{i_0}} Y^r} b = h - \frac{\mathrm{LC}(h)}{\mathrm{LC}(b)} X^{s-s_{i_0}} b$$

et donc $h' \in M_D(G)$, $h' \in I_D$ et $\mathrm{LT}(h') < \mathrm{LT}(h)$, donc $h' < h$. Si on avait $h' = 0$ ou $h' \in M_D(G)$ alors on aurait $h \in M_D(G)$. Donc a trouvé $h' < h$ vérifiant $h' \in I_D \backslash M_D(G)$. Contradiction.

3.2.3 Description de l'algorithme LLL pour le changement d'ordre d'une base de Gröbner

On présente maintenant la version de l'algorithme LLL qui permet de calculer une base de Gröbner pour un autre ordre. La procédure ModuleBasis permettant d'extraire une base en tant que module à partir d'une base de Gröbner est décrite plus bas.

Algorithme 20 LLL

Input : $(\boldsymbol{g_1}, \ldots, \boldsymbol{g_l})$ *où chaque* $\boldsymbol{g_i}$ *est un vecteur dans* $\mathcal{M}^{(D)}(Y)$
$<$ *est un ordre admissible*
Output : *une base réduite de* $\sum_{i=1}^{l} \mathbb{K}[X]\boldsymbol{g_i}$
Trier $\mathrm{LT}_<(\boldsymbol{g_1}) \leq \mathrm{LT}_<(\boldsymbol{g_2}) \leq \cdots \leq \mathrm{LT}_<(\boldsymbol{g_l})$
$k := 1$
while $k < l$ ***do***
  $t := \|\boldsymbol{g_k}\| = \mathrm{LT}_<(\boldsymbol{g_k})$
  for i ***from*** 1 ***to*** $k-1$ ***do***
    $\boldsymbol{g_k} := \boldsymbol{g_k} - \mu_i\, \boldsymbol{g_i}$ *où* $\mu_i := g_{k,i}$ quo $g_{i,i}$
  if $\mathrm{LT}_<(\boldsymbol{g_k}) = \|\boldsymbol{g_k}\| < t$ ***then*** // on a trouvé un plus petit vecteur
    $m := \min\{j \mid (j = k)\ ou\ (1 \leq j < k\ et\ \mathrm{LT}_<(\boldsymbol{g_j}) > \mathrm{LT}_<(\boldsymbol{g_k}))\}$
    $(\boldsymbol{g_1}, \ldots, \boldsymbol{g_l}) := (\boldsymbol{g_1}, \ldots, \boldsymbol{g_{m-1}}, \boldsymbol{g_k}, \boldsymbol{g_{m+1}} \ldots, \boldsymbol{g_l})$
    $k := m$
  else
    $m := \min\{j \mid k \leq j \leq D\ et\ \mathrm{LT}_<(g_{k,j}) = t)\}$
    for i ***from*** 1 ***to*** l ***do*** Swap$(g_{i,k}, g_{i,m})$
    $k := k + 1$
return $(\boldsymbol{g_1}, \ldots, \boldsymbol{g_l})$

Lemma 3.2.1. *Pendant l'execution de LLL pour tout* $j \in \{1, \ldots, D-1\}$ *il existe* $k_j \in \{0, \ldots, D-1\}$ *tel que:* $g_{i,j} \in \mathcal{M}_{k_j}(Y)$ *pour tout* $1 \leq i \leq l$.

Proof. On montre la propriété par récurrence: lorsqu'on calcule $\mu_i := g_{k,i}$ quo $g_{i,i}$ il pourrait sembler qu'on calcule un quotient dans $\mathbb{K}[X, Y]$ mais comme $g_{k,i} = h_{k,i}(X)\, Y^{k_i}$ et $g_{i,i} = h_{i,i}(X)\, Y^{k_i}$ pour un certain k_i il est clair que $\mu_i = q(X)$ où $q(X)$ est le quotient, dans $\mathbb{K}[X]$, de $h_{k,i}(X)$ par $h_{i,i}(X)$. Par conséquent, après l'opération $\boldsymbol{g_k} := \boldsymbol{g_k} - \mu_i\, \boldsymbol{g_i}$ on a encore $g_{i,j} \in \mathcal{M}_{k_j}(Y)$ pour tout $1 \leq i \leq l$.

En combinant l'algorithme LLL avec l'algorithme ModuleBasis il est facile d'obtenir un algorithme de changement d'ordre (on verra dans la preuve de l'algorithme comment choisir D_Y):

Algorithme 21 *Changement d'ordre avec LLL*

Input : G_{old} *une base de Gröbner réduite pour l'ordre* $<_{old}$
$<_{new}$ *ordre admissible et* D_Y *un entier positif.*
Output : G_{new} *une base de Gröbner non réduite pour* $<_{new}$
$(\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}) :=$ ModuleBasis$(G_{old}, <_{old}, D_Y)$
$(\boldsymbol{g_1}, \ldots, \boldsymbol{g_l}) :=$ LLL$((\boldsymbol{b_1}, \ldots, \boldsymbol{b_l}), <_{new}, D_Y)$
return $(\pi(\boldsymbol{g_1}), \ldots, \pi(\boldsymbol{g_l}))$

3.2.4 Exemple détaillé pas à pas.

On reprend le même exemple que la section 3.1.7: $G = \left[f_1 = x^2 - 3\,y - x + 1,\ f_2 = y^2 - \right.$ est une base de Gröbner pour l'ordre DRL avec $y > x$ dans $\mathbb{Q}[x, y]$. On choisit l'ordre lexicographique $>_{\text{lex}}$.

On construit la matrice $G = (\boldsymbol{g_1}, \boldsymbol{g_2}, \boldsymbol{g_3})$ correspondant à $[f_1, y\,f_1, f_2]$:

$l := 3$, $k := 1$:

$$G := \begin{bmatrix} x^2 - x + 1 & -3\,y & 0 \\ -6\,x - 3 & y\left(x^2 - x + 4\right) & 0 \\ -2\,x - 1 & y & y^2 \end{bmatrix}$$

alors $t := \|\boldsymbol{g_1}\| = \text{LT} >_{\text{lex}} (\boldsymbol{g_1}) = y$ et donc $m := \min\{j \mid 1 \le j \le 3 \text{ et } \text{LT}_<(g_{1,j}) = y$ 2: donc échange les colonnes $k = 1$ et $m = 2$:

$k := 2$:

$$G := \begin{bmatrix} -3\,y & x^2 - x + 1 & 0 \\ y\left(x^2 - x + 4\right) & -6\,x - 3 & 0 \\ y & -2\,x - 1 & y^2 \end{bmatrix}$$

$\boldsymbol{g_2} := \boldsymbol{g_2} - \left(\frac{-x^2+x-4}{3}\right) \cdot \boldsymbol{g_1} = \left[0, x^4 - 2\,x^3 + 6\,x^2 - 23\,x - 5, 0\right]$ et comme $\|\boldsymbol{g_2}\| = x^4 <_{\text{lex}} y$ on a fabriqué un plus petit vecteur; on échange $\boldsymbol{g_1}$ et $\boldsymbol{g_2}$.

$k := 1$:

$$G := \begin{bmatrix} 0 & x^4 - 2\,x^3 + 6\,x^2 - 23\,x - 5 & 0 \\ -3\,y & x^2 - x + 1 & 0 \\ y & -2\,x - 1 & y^2 \end{bmatrix}$$

$l := 2$ donc on échange les colonnes1 et 2.

$$G := \begin{bmatrix} x^4 - 2\,x^3 + 6\,x^2 - 23\,x - 5 & 0 & 0 \\ x^2 - x + 1 & -3\,y & 0 \\ -2\,x - 1 & y & y^2 \end{bmatrix}$$

et la base de Gröbner pour l'ordre lexicographique est $G := [x^4 - 2\,x^3 + 6\,x^2 - 23\,x - 5, \left(x^2 - x + 1\right) - 3\,y]$.

3.2.5 Preuve de l'algorithme

Theorem 3.2.2. *Soient G_{old} une base de Gröbner réduite pour l'ordre $<_{old}$ et $<_{new}$ ordre admissible. On fixe $D_Y > 0$ un entier tel que*

$$D_Y \ge \max\{\deg_Y(G_{old}), \deg_Y(G_{new})\}$$

Par exemple si $<_{old}=<_{lex}$ *et* $<_{new}$ *est un ordre du degré on peut prendre* $D_Y = 2\deg(G_{old}) - 1$ *(voir les bornes dans (Buchberger, 1983; Lazard, 1983)). Si* $<_{old}$ *est un ordre du degré et* $<_{new}=<_{lex}$ *alors on peut prendre* $D_Y = \deg_Y(G_{old})$. *Dans ces conditions l'algorithme 21 calcule une base de Gröbner* G_{new} *pour l'ordre* $<_{new}$ *dans* $\mathbb{K}[X,Y]$ *telle que* $\mathrm{Id}(G_{old}) = \mathrm{Id}(G_{new})$.

Proof. Terminaison: On nomme partie II la portion de l'algorithme 20 contenant l'instruction $k := k + 1$.Il y a un nombre fini de passages dans la partie II car on augmente la valeur de k et $k \leq l$; le nombre de passages est également borné dans la partie ou on detecte un plus petit vecteur car dans ce cas le produit

$$\Delta = \|\boldsymbol{g_1}\| \cdots \|\boldsymbol{g_k}\| \, \|\boldsymbol{g_{k+1}}\| \cdots \|\boldsymbol{g_l}\|$$

décroit strictement et Δ est laissé invariant dans la partie II.

Correction: Il est clair que $(\boldsymbol{b_1}, \ldots, \boldsymbol{b_l})$ et $(\boldsymbol{g_1}, \ldots, \boldsymbol{g_l})$ engendrent le même $\mathbb{K}[X]$ sous module M de $\mathbb{K}[X,Y]$. D'après le théorème 3.2.1, $M = I_{D_Y}$. Mais d'aprẽs la proposition 3.2.2, $(\boldsymbol{g_1}, \ldots, \boldsymbol{g_l})$ est une base réduite du réseau L engendré par $(\boldsymbol{b_1}, \ldots, \boldsymbol{b_l})$: en effet les propriété suivantes sont vérifiées:

(i) $\|\boldsymbol{g_i}\| \leq \|\boldsymbol{g_j}\|$ pour $1 \leq i < j \leq k$
(ii) $\|\boldsymbol{g_k}\| \leq \|\boldsymbol{b_j}\|$ pour $k < j \leq l$
(iii) $g_{i,j} < g_{i,i} > g_{i,l}$ pour $1 \leq j < i \leq k$ et $i < k \leq D_Y$.

Ainsi d'après la proposition 3.2.2, $\boldsymbol{g_i}$ est le i-ème minimum successif de M et $\|\boldsymbol{g_i}\| < \|\boldsymbol{g_{i+1}}\|$ (en effet si on avait $\|\boldsymbol{g_i}\| = \|\boldsymbol{g_{i+1}}\|$ alors $\boldsymbol{g} = \boldsymbol{g_{i+1}} - \lambda\, \boldsymbol{g_i} \in M$ et $\|\boldsymbol{g}\| < \|\boldsymbol{g_{i+1}}\|$ pour un certain $\lambda \in \mathbb{K}$; ce implique que $\boldsymbol{g}$ est linéairement dépendant de $\boldsymbol{g_1}, \ldots, \boldsymbol{g_i}$, et donc que $\boldsymbol{g_{i+1}} = \boldsymbol{g} + \lambda\, \boldsymbol{g_i}$ l'est aussi ce qui est en contradiction avec le choix de $\boldsymbol{g_{i+1}}$). Soit maintenant f un polynôme dans $I_{D_Y} = M$, alors il existe $\lambda_1, \ldots, \lambda_l \in \mathbb{K}[X]$ tels que

$$\pi_1(f) = \sum_{j=1}^{l} \lambda_j \boldsymbol{g_j}$$

avec les conditions $1 \leq i < j \leq l$, $\mathrm{LT}(\lambda_i \boldsymbol{g_i}) \neq \mathrm{LT}(\lambda_j\, \boldsymbol{g_j})$ car sinon il existerait t_i, t_j s.t. $\mathrm{LT}(\lambda_i \boldsymbol{g_i}) = X^{t_i}\,\mathrm{LT}(\boldsymbol{g_i})$ and $\mathrm{LT}(\lambda_j \boldsymbol{g_j}) = X^{t_j}\,\mathrm{LT}(\boldsymbol{g_j})$, et comme $\mathrm{LT}(\boldsymbol{g_i}) < \mathrm{LT}(\boldsymbol{g_j})$ implique $t_i > t_j$ (si $t_i < t_j$ alors $X^{t_i}\,\mathrm{LT}(\boldsymbol{g_i}) < X^{t_j}\,\mathrm{LT}(\boldsymbol{g_j})$, et si $t_i = t_j$ alors $\mathrm{LT}(\boldsymbol{g_i}) = \mathrm{LT}(\boldsymbol{g_j})$) ainsi $\boldsymbol{g} := X^{t_i - t_j}\, \boldsymbol{g_i} - \boldsymbol{g_j} \in M$ et $\mathrm{LT}(\boldsymbol{g}) < \mathrm{LT}(\boldsymbol{g_j})$ ce qui implique que $\boldsymbol{g}$ est linéairement dépendant de $a_1, \ldots, a_{j-1}$, donc $\boldsymbol{g_i} = X^{t_i - t_j}\, \boldsymbol{g_i} - \boldsymbol{g}$ est lié avec $\boldsymbol{g_1}, \ldots, \boldsymbol{g_{j-1}}$ ce qui est en contradiction avec le choix de $\boldsymbol{g_{i+1}}$. Par conséquent il y a un unique $1 \leq j \leq l$ tel que $\mathrm{LT}(f) = \mathrm{LT}(\lambda_j \boldsymbol{g_j})$, donc $\mathrm{LT}(\boldsymbol{g_j}) | \mathrm{LT}(f)$ ce qui montre que $(\boldsymbol{g_1}, \ldots, \boldsymbol{g_l})$ est une base de Gröbner de I par rapport à $<_{\mathrm{new}}$.

3.2.6 Borne de complexité de LLL

On peut estimer la complexité de l'algorithme 21 en termes de nombre d'opérations arithmétiques dans $\mathbb{K}$. By an arithmetical operation in K, we mean addition, subtraction, multiplication or division of two elements of K. On a besoin des paramètres suivants de mesure de la complexité:

- $d_Y = \deg_Y(G_{\text{old}})$
- $D_Y = \max\{d_Y, \deg_Y(G_{\text{new}})\}$
- $l = D_Y - r_1$ où $r_1 = \deg_Y(\text{LT}(g_1))$
- $d_X = \deg_X(G_{\text{old}})$
- $D_X = \deg_X(G_{\text{new}})$

On estime d'abord le nombre d'itérations de l'algorithme LLL dans le pire cas:

Theorem 3.2.3. *Le nombre d'itérations de l'algorithme 20 est borné par* $S = D_Y^2.D_X$ *pour l'ordre* $<_{Lex}$ *ou l'ordre* $<_{DRL}$.

Proof. À chaque itération de la boucle principale soit k est incrémenté de 1 soit $\Delta = \|\boldsymbol{g_1}\| \cdots \|\boldsymbol{g_k}\| \, \|\boldsymbol{g_{k+1}}\| \cdots \|\boldsymbol{g_l}\|$ décroit strictement (pour $<_{new}$), c'est à dire en posant

$$\boldsymbol{g} := \boldsymbol{g_k} - \mu_i \, \boldsymbol{g_i} \text{ où } \mu_i := g_{k,i} \text{ quo } g_{i,i},$$

on a $\|\boldsymbol{g_1}\| \cdots \|\boldsymbol{g_{k-1}}\| \, \|\boldsymbol{g}\| \cdots \|\boldsymbol{g_l}\| <_{\text{new}} \|\boldsymbol{g_1}\| \cdots \|\boldsymbol{g_{k-1}}\| \, \|\boldsymbol{g_k}\| \cdots \|\boldsymbol{g_l}\|$. Ainsi

$$\deg_X(\text{LT}(\boldsymbol{g})) \leq \deg_X(\text{LT}(\boldsymbol{g_k})) + D_X$$

et

$$\deg_Y(\text{LT}(\boldsymbol{g})) \leq D_Y.$$

Dans le cas de l'ordre lexicographique $<_{\text{Lex}}$on a

$$\begin{aligned}
X^{n_{1,0}} &<_{Lex} \cdots <_{Lex} X^{n_{1,1}} \\
X^{n_{2,0}}Y &<_{Lex} \cdots <_{Lex} X^{n_{2,1}}Y \\
X^{n_{3,0}}Y^2 &<_{Lex} \cdots <_{Lex} X^{n_{3,1}}Y^2 \\
&\vdots \\
X^{n_{ld_Y,0}}Y^{ld_Y-1} &<_{Lex} \cdots <_{Lex} X^{n_{ld_Y,1}}Y^{ld_Y-1} \\
X^{n_{ld_Y+1,0}}Y^{ld_Y} &<_{Lex} \cdots <_{Lex} X^{n_{ld_Y+1,1}}Y^{ld_Y}
\end{aligned}$$

et donc le nombre de passages dans la boucle est borné par:

$$S = \sum_{i=1}^{ld_Y+1} n_{i,1} - n_{i,0}$$

mais $\deg_X(\text{LT}(\boldsymbol{g}))) \leq \deg_X(\text{LT}(\boldsymbol{g_k})) + D_X$ implique

$$n_{i,1} \leq n_{i+1,0} + D_X \quad \text{pour } i = 1, \ldots, ld_Y$$

et donc

$$S \leq n_{ld_Y+1,1} + ld_Y D_X - n_{1,0}.$$

le nombre d'itérations est borné par $S = l.d_Y.D_X$ pour l'ordre lexicographique. Dans le cas de l'ordre DRL on

$$\begin{array}{c} X <_{\text{DRL}} Y \\ \vdots \\ X^{ld_X-1}Y^m <_{\text{DRL}} X^{ld_X-2}Y^{m+1} \cdots <_{\text{DRL}} X^{ld_X-D_Y-1}Y^{m+D_Y} \\ X^{ld_X}Y^m <_{\text{DRL}} X^{ld_X-1}Y^{m+1} \cdots <_{\text{DRL}} X^{ld_X-D_Y}Y^{m+D_Y}. \end{array}$$

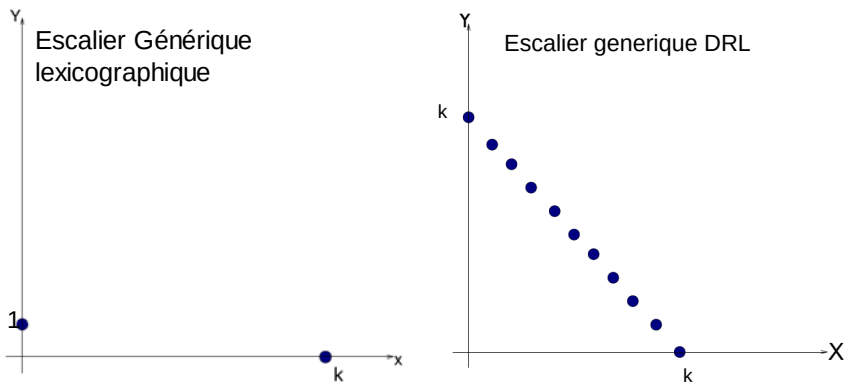
et dans ce cas la borne est $S = l.d_X.D_Y$.
Dans les deux cas on obtient la borne:

$$S = D_Y^2 D_X.$$

On à chaque passage dans l'algorithme on a au plus $O(l.D_X)$ opérations dans $\mathbb{K}$, on en déduit:

Theorem 3.2.4. *Pour l'ordre $<_{Lex}$ ou l'ordre $<_{DRL}$, l'algorithm 21 nécessite moins de $O(D_Y^3.D_X^2)$ opérations arithmétiques dans $\mathbb{K}$.*

Le théorème permet d'estimer la complexité de l'algorithme LLL; cependant les paramètres de mesure ne sont pas les mêmes que dans l'algorithme FGLM. Si I est un idéal zéro dimensionnel on note $\deg(I)$ la dimension du $\mathbb{K}$ espace vectoriel $\mathbb{K}[X,Y]/I$. On a vu que la complexité de FGLM était (théorème 3.1.2) bornée par $O(\deg(I)^3)$. Une façon grossière d'exprimer la complexité donné par le théorème 3.2.4 en fonction de $\deg(I)$ est de majorer: $D_X \leq N_b$ et $D_Y \leq N_b$; on trouve ainsi une borne en $O(\deg(I)^5)$. Pour estimer de manière plus réaliste la quantité $D_Y^3.D_X^2$ on considère un idéal dont la base lexicographique est en Shape Position (voir équations (2.5)): $[Y - Q(X), P(X)]$ où P (resp. Q) est un polynôme de degré $\deg(I)$ (resp. $\deg(I) - 1$). Dans ce cas les escaliers pour l'ordre lexicographique et l'ordre DRL sont représentés sur les figures suivantes:



Par conséquent on peut calculer les bornes des degrés en X et Y:

$$\begin{array}{cc} \text{Ordre Lexicographique} & \text{Ordre DRL} \\ \deg(I) & \text{avec } \deg(I) = \frac{k(k+1)}{2} \\ d_Y = 1, d_X = \deg(I) & d_X \approx d_Y \approx \frac{-1+\sqrt{1+8\ \deg(I)}}{2} \\ D_Y \approx \frac{-1+\sqrt{1+8\ \deg(I)}}{2}, D_X = \deg(I) & \\ D_Y^3.D_X^2 = O(\deg(I)^{\frac{7}{2}}) & \end{array}$$

Ainsi, pour presque tout système, l'algorithme 21 nécessite au plus $O(\deg(I)^{3.5})$ opérations arithmetiques pour calculer une base G_{Lexico} dans $\mathbb{K}[X,Y]$ telle que $\text{Id}(G_{\text{DRL}}) = \text{Id}(G_{\text{Lexico}})$ (la même borne est valable si on calcule de l'ordre lexicographique vers l'ordre DRL). Lorsque D_Y est petit l'algorithme devient meilleur que FGLM.

3.2.7 Comparaison avec le FGLM en Maple

L'algorithme 21 a été implanté dans *Maple (release 5 ou 10 selon les tests)*. Dans les tables suivantes on reporte les temps des calculs (tests effectués sur un Pentium 3 à 800 Mhz) modulo le nombre premier $p = 65521$. Il est nécessaire d'insister que ces tests n'ont d'autre but que de montrer la validité de la méthode et de donner un ordre de grandeur de la *complexité pratique* de l'algorithme LLL. Ainsi cette implantation en Maple ne peut être comparée avec une implantation dans un langage de bas niveau (langage C comme dans Magma ou FGb). Il a aussi été nécesssaire de se restreindre à des exemples de petite ou moyenne taille. Pour toutes ces raisons on a ajouté dans les tables le nombre d'opérations de multiplication dans $\mathbb{F}_{65521}$: ce nombre ne dépend pas de la qualité de l'implantation et est donc intrinsèque. Pour chaque exemple on donne les valeurs des paramètres $\deg(I)$, l, d_X, D_X, d_Y et D_Y.

Pour ramener les exemples ayant n variables au cas de 2 variables on commence par *éliminer* $n-2$ variables. Ceci est réalisé facilement grâce à un calcul de base de Gröbner poiur un ordre d'élimination. La deuxième partie de ces tables contient des exemples "*aléatoires*": plus exactement rand-*d*-*k* est l'idéal $\text{Id}(y - Q(x), P(x))$ où Q et P sont des polynômes univariés générés aléatoirement en degré d et de degré maximal k en y.

De ces tables on peut conclure que l'implantation non optimisée de LLL est toujours plus rapide que l'implantation par défaut de l'algorithme FGLM dans *Maple;* cette version est cependant beaucoup plus lente que l'implanation de FGLM dans FGb (comparer les exemples Katsura 7 et 8 avec la section 3.1.10). Il est aussi clair que le meilleur cas est obtenu lorsque D_Y est très petit: alors la complexité est simplement $O(N_b^2)$ (c'est le cas par exemple pour les exemples rand-50-17, rand-100-34 ,rand-150-51, rand-200-67 et rand-300-101 où D_Y vaut 2).

DRL → Lex	Nb of × in LLL	deg(I)	FGLM (sec)	LLL (sec)	l	d_X	D_X	D_Y
benchmarkD1	575	48	15.4	.15	3	25	48	2
Cyclic5	37	55	2.1	.25	9	12	15	8
UteshevBikker	19,721	36	43.6	1.5	9	8	36	8
Fabrice24	22,799	40	138.3	1.6	9	9	40	8
dessin2	28,821	42	145.2	1.8	9	9	42	8
dessin1	45,357	46	305.7	2.4	10	10	46	9
benchmarki1	188,997	66	579.4	6.4	12	11	66	11
cyclic6	50,938	126	421.8	3.15	13	24	48	12
katsura7	2,053,674	128	>12000	40.2	16	16	128	15
katsura8	45,992,680	256	>12000	356.8	23	23	241	22
rand-50-17	1,632	50	10.5	.1	3	34	50	2
rand-100-34	6,666	100	141.4	.3	3	67	100	2
rand-150-51	14,798	150	834.8	.5	3	101	150	2
rand-200-67	30,822	200	4261	.8	3	134	200	2
rand-300-101	69,300	300	>4000	1.4	3	201	300	2
rand-50-1	52,758	50	364.5	2.6	10	10	50	9
rand-100-1	792,333	100	>5000	17.5	14	14	100	13
rand-150-1	4,196,783	150	>20000	62.6	17	17	150	16
rand-200-1	17,950,766	200	>20000	157.2	20	20	200	19
rand-300-1	106,495,461	300	>20000	766.7	25	25	300	24

Table 3.2. Comparison FGLM/LLL (de DRL vers Lex) modulo p.

Lex → DRL	Nb of × in LLL	N_b	FGLM (sec)	LLL (sec)	l	D_X	d_Y	D_Y
benchmarkD1	1,200	48	8.6	.2	3	48	1	2
cyclic5	307	55	2.4	.5	9	15	7	8
UteshevBikker	46,111	36	186.5	7.3	9	36	1	8
Fabrice24	56,550	40	245.4	8.8	9	40	1	8
dessin2	62,588	42	291.0	9.3	9	42	1	8
dessin1	94,325	46	522.1	13.4	10	46	1	9
benchmarki1	285,970	66	2,847.7	35.3	12	66	1	11
cyclic6	11,070	126	75.1	7.1	13	48	6	12
katsura7	1,941,523	128	>12000	192.2	16	128	2	15
katsura8	23,187,086	256	>12000	1927.	23	241	16	22
rand-50-17	4,896	50	15.1	.35	3	50	1	2
rand-100-34	19,760	100	216.7	.9	3	100	1	2
rand-150-51	44,394	150	1486.1	1.7	3	150	1	2
rand-200-67	79,596	200	8090.	2.8	3	200	1	2
rand-300-101	178,794	300	> 8000	5.4	3	300	1	2
rand-50-1	110,774	50	339.3	15.1	10	50	1	9
rand-100-1	905,447	100	>6000	100.6	14	100	1	13
rand-150-1	3,052,423	150	>50000	310.5	17	150	1	16
rand-200-1	7,646,326	200	>50000	711.9	20	200	1	19
rand-300-1	27,156,881	300	>50000	2647.5	25	300	1	24

Table 3.3. Comparison FGLM/LLL (de Lex vers DRL) modulo p.

4. Algorithme F_4

4.1 Introduction

Si on se contente d'appliquer les algorithmes (Buchberger B., 1965; Buchberger B., 1970; Buchberger B., 1985) déjà décrit dans le chapitre 2 sur une liste de problèmes typiques provenant d'applications diverses (voir notamment les chapitres 8,9,10,18,12) on constate que même les meilleures implantations sont incapables de calculer les bases de Gröbner pour des calculs de taille moyenne. Dans ce chapitre, on va décrire un autre algorithme (dont le nom est F_4) pour calculer efficacement des bases de Gröbner et principalement des bases pour un ordre du degré (DRL). Il sera donc nécessaire en pratique d'appliquer un autre algorithme d'élimination ou de changement d'ordre sur le résultat fournit par F_4 (une autre alternative étant de calculer une forme RUR(Rouillier, 1999)). On verra cependant des exemples où le calcul pour un ordre d'élimination est aussi très efficace (voir problèmes de robotique du chapitre 12). En simplifiant, on pourrait suggérer deux améliorations de l'algorithme de Buchberger: comme 90% du temps est passé à réduire des paires critiques vers zéro il serait utile de posséder des critères plus puissant que les critères de Buchberger (Buchberger B., 1979) pour éliminer *toutes* les paires critiques inutiles (de façon théorique il existe un tel critère mais son application en pratique est très coûteuse). Cet aspect fondamental est l'objet du chapitre suivant (chap.5). La seconde amélioration concerne les stratégies: durant un calcul de base de Gröbner on est confronté à plusieurs choix: choisir une paire critique dans une liste de paires critiques, choisir un réducteur dans une liste. On sait que quelque soit les choix le résultat final de l'algorithme ne change pas (théorème de Buchberger), en revanche les calculs peuvent devenir impossible en cas de mauvais choix. Même si diverses stratégies ont été proposées ((Giovini A. and Mora T. and Niesi G. and Robbiano L. and Traverso C., 1991) ou même (Gerdt V.P., 1995)), les heuristiques sur lesquelles elles reposent ne sont pas entièrement expliquées et sont plus ou moins efficaces sur divers exemples; il est difficile d'en désigner une qui soit optimale dans tous les cas. L'objectif de ce chapitre est de présenter un algorithme plus puissant de réduction. Dans ce but on va chercher à réduire *simultanément* plusieurs polynômes par une liste de polynômes en utilisant des techniques d'algèbre linéaire (la réduction de Gauß principalement).

L'algorithme F_4 de base est présenté dans la section **??**. Cette section est divisée en plusieurs parties: premièrement nous établissons le lien entre algèbre linéaire (matrices) et l'algèbre des polynômes et la méthode de Macaulay(Macaulay, 1916) et lien entre et le calcul des bases de Gröbner est explicité. Ensuite nous présentons dans la sous-section 4.3.1 une version simplifiée de l'algorithme qui n'utilise que partiellement le résultat de la réduction de Gauß. Une version améliorée de l'algorithme incluant les critères de Buchberger est donnée dans 4.3.2. Nous terminons cette section dans 4.3.4 en donnant des indications sur le choix d'une bonne stratégie de sélection des paquets de paires critique. Cet algorithme a été implanté dans un logiciel écrit par l'auteur nommé FGb (Fast Gb) maintenant accessible via le logiciel généraliste Maple et plus récemment dans d'autres logiciels (voir 4.4.1) dont en particulier Maple(Char B. and Geddes K. and Gonnet G. and Leong B. and Monagan M. and Watt S., 1991), Magma(Cannon J., 1998). Dans la section **??** nous présentons quelques données expérimentales réalisées par A. Steel incluant une comparaison de son implantation de l'algorithme F_4 dans le logiciel Magma(Cannon J., 1998) avec les meilleurs programmes de calcul des bases de Gröbner utilisant l'algorithme de Buchberger.

Le nom de cet algorithme est simplement l'algorithme numéro 4 (ne pas confondre avec le corps $\mathbb{F}_4$!). Dans le reste du document, F_4 désigne cet algorithme.

4.2 Bases de Gröbner et Macaulay

4.2.1 Représentation matricielle des polynômes.

Definition 4.2.1. *Si $F = [f_1, \ldots, f_m]$ est un vecteur de m polynômes et $<$ un ordre admissible, $\mathrm{T}_<(F) = [t_1, \ldots, t_l]$ les termes du support de F triés pour l'ordre $<$ (voir la définition dans la section 2.3.2 p. 20). Alors une représentation matricielle $M_{T_<(F)}(F)$ de F est une matrice:*

$$M_{T_<(F)}(F) = \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \left| \begin{array}{ccc} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{array} \right. \end{array} \begin{array}{l} f_1 \\ f_2 \\ f_3 \end{array}$$

dont le coefficient d'indice (i, j) est le coefficient du terme t_j dans f_i . De plus, $M_{T_<(F)}(F)$ vérifie l'équation:

$$F = M_{T_<(F)}(F) \cdot T_<(F)$$

Pour alléger les notations on note $M(F) = M_{T_<(F)}(F)$.

Definition 4.2.2. *Réciproquement si M est une matrice de taille $l \times m$ à coefficients dans $\mathbb{K}$ et $X = [t_1, \ldots, t_m]$ un vecteur de termes alors la représentation polynomiale de M par rapport à X est le vecteur de l polynômes déterminé par l'équation:*

$$F = M \cdot X$$

À partir de polynômes on peut toujours construire une matrice, généralisation naturelle de la matrice de Sylvester, qui consiste à multiplier tous les polynômes par tous les termes possibles:

Definition 4.2.3. *Matrice de Macaulay ((Macaulay, 1916)). Soient $F = [f_1, \ldots, f_m]$ un vecteur de m polynômes et d un entier positif alors la matrice de Macaulay en degré d de F, notée $\mathcal{M}_d^{\text{acaulay}}(F)$, est la représentation matricielle de*

$$F^{(d)} = [t_j \cdot f_i \mid 1 \leq i \leq m \quad et \ t_j \in T \quad avec \ \deg(t_j) \leq d - \deg(f_i)]$$

$$\mathcal{M}_d^{\text{acaulay}}(F) = M(F^{(d)}) = \begin{array}{r} \\ t_1\, f_1 \\ t_2\, f_2 \\ \cdots \end{array} \begin{array}{c} \begin{array}{ccc} m_1 & m_2 & m_3 \end{array} \\ \left|\begin{array}{ccc} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{array}\right| \end{array}$$

L'opération de base pour les matrices est le calcul d'une forme échelon; ce sera l'opération principale et bien souvent la plus coûteuse.

Definition 4.2.4. *Si $M(F)$ est la représentation matricielle d'un vecteur de polynômes F on note $\widetilde{M(F)}$ le résultat d'une élimination de Gauß de la matrice $M(F)$ (sans faire de pivot sur les colonnes).*

Pour simplifier l'exposition de l'algorithme on définit également l'élimination de Gauß d'un vecteur de polynômes:

Definition 4.2.5. *Soit F un sous ensemble fini de $\mathbb{K}[x_1, \ldots, x_n]$ et $<$ un ordre admissible. À partir de F on construit sa représentation matricielle, $M(F)$, puis on calcule sa forme échelon, $\widetilde{M(F)}$, enfin on considère la représentation polynomiale, notée $\tilde{F}$ de cette dernière matrice; on dira que $\tilde{F}$ est la forme échelonnée de F (ou élimination de Gauß) par rapport à $<$.*

4.2.2 Méthode de Macaulay

L'utilisation de l'algèbre linéaire pour résoudre un système algébrique remonte à Macaulay (voir (Macaulay, 1916)); Macaulay généralise la matrice de Sylvester **??**(J., 1853) (c'est la matrice utilisé pour le résultant de deux polynômes univariés) aux polynômes à plusieurs variables.

Le lien entre le calcul d'une base de d-Gröbner (voir définition 2.6.2 p.33) et l'algèbre linéaire découle du fait qu'on peut calculer une base de Gröbner à partir de la matrice de Macaulay suivant le théorème de D. Lazard:

Theorem 4.2.1. *(Lazard (Lazard D., 1983; Lazard D., 1981)) Si* $F = \{f_1, \ldots, f_m\}$ *est un ensemble de polynômes homogènes alors la représentation polynomiale de* $\widetilde{\mathcal{M}_d^{\text{acaulay}}}(F)$ *est une d-base de Gröbner (non réduite) de* F*.*

Par conséquent si d est *assez grand* alors la matrice de Macaulay correspondante contient la base de Gröbner complète; en particulier:

Theorem 4.2.2. *(borne de Macaulay). Soit* $F = \{f_1, \ldots, f_m\}$ *un ensemble de polynômes homogènes réguliers. On définit:*

$$D = 1 + \sum_{i=1}^{m} (deg(f_i) - 1)$$

alors $\widetilde{\mathcal{M}_D^{\text{acaulay}}}(F)$ *est une base de Gröbner (non réduite) de* F*.*

Note 4.2.1. D'un point de vue pratique l'utilisation de la matrice de Macaulay est désastreuse; en effet les lignes de cette matrice ne sont pas indépendantes et donc appliquer une élimination de Gauß est inutilement coûteux. Le but de ce chapitre et du chapitre suivant (chap. 5) est de construire des matrices définissant le même espace vectoriel mais dont la taille sera beaucoup plus petite. De plus dans la méthode de Macaulay il faut une borne D sur le degré maximal des calculs ou une hypothèse sur le système initial. Le présent chapitre permet de réduire considérablement les tailles de ces matrices; le chapitre suivant (chap. 5) construit des matrices dont la taille est optimale dans le sens où les matrices seront de rang pleins. Toutefois il est important de noter que des matrices de plus petites tailles n'impliquent pas *automatiquement* des calculs plus rapides.

Note 4.2.2. En cryptographie, l'algorithme XL (Courtois *et al.* , 2000) a été créé pour chercher *une* solution d'un système algébrique dans un corps fini. En résumé, le principe est de construire la matrice de Macaulay sans un certain degré puis de trianguler la matrice pour faire apparaitre un polyôme unvarié dont il est facile d'extraire une solution dans le corps fini. Cet algorithme peut toujours être simulé par un calcul de base de Gröbner (voir (Ars *et al.* , 2004; G., 2005)).

4.3 L'algorithme F_4

4.3.1 Description de l'algorithme F_4

On sait que durant l'exécution de l'algorithme de Buchberger, on dispose de plusieurs degré de liberté:

- choisir une paire critique dans une liste de paires critique.
- choisir un réducteur dans une liste de réducteurs lorsqu'on réduit un polynôme par une liste de polynômes.

Bruno Buchberger (Buchberger B., 1965) a prouvé que ces choix ne sont pas importants en regard de la preuve de l'algorithme, en revanche ces choix influent de manière *cruciale* sur le temps de calcul. De plus les meilleures stratégies ((Giovini A. and Mora T. and Niesi G. and Robbiano L. and Traverso C., 1991)) se basent uniquement sur le terme de têtes des polynômes pour faire un choix. Si on prend le cas extrême où tous les polynômes ont le même terme de tête, toutes les paires critiques sont égales et il n'est pas possible de faire un choix. Nous résolvons ce problème d'une manière simple et un peu surprenante: *on ne fait pas de choix.* Plus exactement au lieu de choisir *une paire critique* à chaque étape, on considère un *sous-ensemble* de paires critiques que l'on va traiter *simultanément.* À partir de ce choix on construit une matrice la plus creuse possible mais contenant toutes les réductions *a priori.* Ensuite il faut appliquer une réduction de Gauß sur cette matrice. Par conséquent, on reporte les choix nécessaires (choix de pivot par exemple) dans la seconde phase de l'algorithme qui est la phase d'algèbre linéaire de l'algorithme.

Nous modifions légèrement la définition usuelle de paire critique:

Definition 4.3.1. *Une paire critique de deux polynômes* (f_i, f_j) *est un élément de* $T^2 \times \mathbb{K}[x_1, \ldots, x_n] \times T \times \mathbb{K}[x_1, \ldots, x_n]$,

$$\mathrm{Pair}(f_i, f_j) := (\mathrm{lcm}_{ij}, t_i, f_i, t_j, f_j)$$

tel que

$$\mathrm{lcm}(\mathrm{Pair}(f_i, f_j)) = \mathrm{lcm}_{ij} = \mathrm{LT}(t_i f_i) = \mathrm{LT}(t_j f_j) = \mathrm{lcm}(\mathrm{LT}(f_i), \mathrm{LT}(f_j))$$

On définit les deux opérateurs de projections:

Definition 4.3.2. *On dira que le degré d'une paire critique* $p_{i,j} = \mathrm{Pair}(f_i, f_j)$, $\deg(p_{i,j})$, *est* $\deg(lcm_{i,j})$. *De plus on définit les opérateurs:*

$$\mathrm{Left}(p_{i,j}) := t_i \cdot f_i \quad et \quad \mathrm{Right}(p_{i,j}) := t_j \cdot f_j$$

Nous avons maintenant les outils pour présenter une version simplifiée de l'algorithme. Toutes les matrices apparaissant dans les algorithmes suivants sont la représentation matricielle d'une liste de polynômes tel que décrit dans la définition 4.2.1.

Algorithme 22 *Algorithme F_4 (version simplifiée)*

Input: $\begin{cases} F \text{ un sous-ensemble fini de } \mathbb{K}[x_1,\ldots,x_n] \\ \mathcal{S}el \text{ une fonction } List(Pairs) \to List(Pairs) \\ \quad \text{tel que } \mathcal{S}el(l) \neq \emptyset \text{ si } l \neq \emptyset \end{cases}$
Output: *un sous ensemble fini de* $\mathbb{K}[x_1,\ldots,x_n]$.
$G := F$, $\tilde{F_0^+} := F$, $d := 0$ *et* $P := \left\{\mathrm{Pair}(f,g) \mid (f,g) \in G^2 \text{ avec } f \neq g\right\}$
while $P \neq \emptyset$ ***do***
 $d := d+1$
 $P_d := \mathcal{S}el(P)$
 $P := P \backslash P_d$
 $L_d := \mathrm{Left}(P_d) \cup \mathrm{Right}(P_d)$
 $\tilde{F_d^+}$:=REDUCTION(L_d, G)
 for $h \in \tilde{F_d^+}$ ***do***
  $P := P \cup \{\mathrm{Pair}(h,g) \mid g \in G\}$
  $G := G \cup \{h\}$
return G

Nous devons maintenant étendre la définition de la réduction d'un polynôme modulo un sous-ensemble de $\mathbb{K}[x_1,\ldots,x_n]$, à la réduction d'un sous ensemble de $\mathbb{K}[x_1,\ldots,x_n]$ modulo un autre sous-ensemble de $\mathbb{K}[x_1,\ldots,x_n]$:

Algorithme 23 REDUCTION

Input: L, G *sous-ensembles finis de* $\mathbb{K}[x_1,\ldots,x_n]$
Output: *un sous-ensemble fini de* $\mathbb{K}[x_1,\ldots,x_n]$ *(éventuellement vide).*
F :=SYMBOLICPREPROCESSING(L,G)
$\widetilde{F}$:= *Réduction de Gauß de* F *par rapport à* $<$
$\tilde{F^+} := \left\{f \in \tilde{F} \mid \mathrm{LT}(f) \notin \mathrm{LT}(F)\right\}$ // partie "utile" de $\widetilde{F}$
return $\tilde{F^+}$

Nous décrivons maintenant la fonction principale de l'algorithme, c'est à dire la construction de F et donc de la matrice $M(F)$. Elle ajoute, en fonction des données L et G, tous les polynômes nécessaires pour que la réduction de Gauß contienne les réductions modulo G de l'ensemble L. Dans la mesure ou aucune opération arithmétique n'est utilisée, c'est vraiment un pré-traitement *symbolique*.

Algorithme 24 SYMBOLICPREPROCESSING

Input: *L, G sous-ensembles finis de $\mathbb{K}[x_1, \ldots, x_n]$*
Output: *un sous-ensemble fini de $\mathbb{K}[x_1, \ldots, x_n]$*
$F := L$
$Done := \mathrm{LT}(F)$
while $\mathrm{T}(F) \neq Done$ ***do***
    choisir m un élément de $\mathrm{T}(F) \backslash Done$
    $Done := Done \cup \{m\}$
    if *m top reductible modulo G* ***then***
        il existe $g \in G$ et un $m' \in T$ tel que $m = m' \cdot \mathrm{LT}(g)$
        $F := F \cup \{m' \cdot g\}$
return F

Note 4.3.1. La procédure SymbolicPreprocessing est très efficace puisque sa complexité est proportionnelle en la taille de sa sortie si $\#G$ est plus petite que le taille finale de $T(F)$ ce qui est souvent le cas en pratique. Comme l'ordre monomial n'intervient pas on peut aussi envisager une implantation en parallèle.

Nous renvoyons à ((Faugère J.C., 1999)) pour des preuves complètes de terminaison et de correction de l'algorithme et nous nous contentons d'en esquisser les grandes lignes:

Lemma 4.3.1. *Pour tout polynôme $p \in L$, on a $p \xrightarrow{G \cup \tilde{F^+}} 0$*

Theorem 4.3.1. *L'algorithme F_4 calcule une base de Gröbner G dans $\mathbb{K}[x_1, \ldots, x_n]$ telle que $F \subseteq G$ et $Id(G) = Id(F)$.*

Proof. Terminaison: Par l'absurde. Supposons que la boucle **while** ne se termine pas. On en déduit qu'il existe une suite croissante (d_i) d'entiers naturels tels que $\tilde{F^+_{d_i}} \neq \emptyset$ pour tout i. Fixons $q_i \in \tilde{F^+_{d_i}}$ (et donc q_i peut être n'importe quel élément dans $\tilde{F^+_{d_i}}$). On définit U_i comme étant l'idéal $U_{i-1} + \mathrm{Id}(\mathrm{LT}(q_i))$ pour $i > 1$ et $U_0 = \mathrm{Id}(0)$. La ligne

$$\tilde{F^+} := \left\{ f \in \tilde{F} \mid \mathrm{LT}(f) \notin \mathrm{LT}(F) \right\}$$

implique que $U_{i-1} \subsetneq U_i$. Ceci contredit le fait que $\mathbb{K}[x_1, \ldots, x_n]$ soit noethérien.

validité: On a $G = \cup_{d \geq 0} \tilde{F^+_d}$. On montre que les quantités suivantes sont des invariants de la boucle **while**: G est un sous-ensemble fini de $\mathbb{K}[x_1, \ldots, x_n]$ tel que $F \subset G \subset \mathrm{Id}(F)$, et $spol(g_1, g_2) \xrightarrow{*} 0$ pour tout $(g_1, g_2) \in G^2$ tel que $(g_1, g_2) \notin P$. Pour cette dernière assertion, si $(g_1, g_2) \notin P$, ceci implique que $Pair(g_1, g_2) = (lcm_{1,2}, t_1, g_1, t_2, g_2)$ a été sélectionné lors d'une précédente étape (disons à l'étape d) par la fonction de sélection $\mathcal{S}el$. Par conséquent $t_1 \cdot g_1$ et $t_2 \cdot g_2$ sont dans L_d, donc $spol(g_1, g_2) = t_1 \cdot g_1 - t_2 \cdot g_2 \xrightarrow{*} 0$ d'après le lemme 4.3.1.

Note 4.3.2. Si $\#\mathcal{S}el(l) = 1$ pour tout $l \neq \emptyset$ alors l'algorithme F_4 est exactement l'algorithme de Buchberger. Dans ce cas la fonction $\mathcal{S}el$ correspond à la stratégie de sélection des paires critiques de l'algorithme de Buchberger.

Note 4.3.3. Dans la preuve de terminaison on peut se demander pourquoi on ne considère qu'un seul élément de $\tilde{F_d^+}$ et pas $\tilde{F_d^+}$ en entier. En considérons l'exemple suivant:

pour l'ordre lexicographique tel que $x > y > z$, soient $F = [f_1 = xy^2 + 1, f_2 = xz^2 + 1, f_3 = y^3 + y^2]$ et $\mathcal{S}el$ = la fonction identité. On trouve successivement $P_1 = \{Pair(f_1, f_2), Pair(f_2, f_3), Pair(f_1, f_3)\}$ puis $\tilde{F_1^+} = \{y^2 - z^2, y + 1\}$ et donc que $Id(\mathrm{LT}(\tilde{F_1^+})) = \{y\}$. Par suite, et contrairement à l'algorithme de Buchberger, il n'est pas vrai qu'après chaque opération

$$G' := G \cup \{h\},$$

on ait $Id(\mathrm{LT}(G')) \supsetneq Id(\mathrm{LT}(G))$.

4.3.2 Ajout des critères de Buchberger dans F_4

Afin d'obtenir une version vraiment efficace de l'algorithme il est indispensable de lui adjoindre des critères pour éviter des calculs inutiles. Dans ce chapitre on intègre a F_4 les critères de Buchberger; une autre possibilité est d'utiliser les critères F_5 du chapitre5. Dans la description suivante on utilise l'implantation standard de ces critères ((Gebauer & Möller, 1988)):

Algorithme 25 *Critères de Buchberger*

(G_{new}, P_{new}) :=UPDATE(G_{old}, P_{old}, h)

Input: $\begin{cases} \textit{un sous-ensemble fini } G_{old} \textit{ de } \mathbb{K}[x_1, \ldots, x_n] \\ \textit{un ensemble fini } P_{old} \textit{ de paires critiques de } \mathbb{K}[x_1, \ldots, x_n] \\ 0 \neq h \in \mathbb{K}[x_1, \ldots, x_n] \end{cases}$

Output: *un sous-ensemble fini de* $\mathbb{K}[x_1, \ldots, x_n]$ *et une liste de paires critiques correspondant à la mise à jour de la liste des paires critiques (voir 10 page 33).*

4.3.3 Version optimisée de l'algorithme F_4

Dans la version initiale de l'algorithme nous avons utilisé seulement *certaines* lignes de la matrice mise sous forme échelonnée (les lignes correspondant à F_d^+). Dans la nouvelle version de l'algorithme on va garder ces lignes inutilisées en essayant de remplacer certains produits $m \cdot f$ apparaissant dans les lignes de la matrice F par un produit équivalent $m' \cdot f'$ avec $m \geq m'$ (en pratique on ne conserve que des produits de la forme $x_k \cdot f'$. Une nouvelle variable $\mathcal{F}$ est introduite dans l'algorithme: $\mathcal{F}$ est un vecteur contenant toutes les formes échelons qui ont été calculées:

Algorithme 26 *Algorithme F_4 (version améliorée)*

Input: $\begin{cases} F \text{ un sous-ensemble fini de } \mathbb{K}[x_1, \ldots, x_n] \\ \mathcal{S}el \text{ une fonction } \mathrm{List}(Pairs) \to \mathrm{List}(Pairs) \\ \quad \text{tel que } \mathcal{S}el(l) \neq \emptyset \text{ si } l \neq \emptyset \end{cases}$
Output: *un sous ensemble fini de* $\mathbb{K}[x_1, \ldots, x_n]$.
$G := \emptyset$ *et* $P := \emptyset$ *et* $d := 0$
while $F \neq \emptyset$ ***do***
 $f := \mathrm{first}(F)$
 $F := F \backslash \{f\}$
 $(G, P) :=$UPDATE(G, P, f)
while $P \neq \emptyset$ ***do***
 $d := d + 1$
 $P_d := \mathcal{S}el(P)$
 $P := P \backslash P_d$
 $L_d := \mathrm{Left}(P_d) \cup \mathrm{Right}(P_d)$
 $(\tilde{F_d^+}, F_d) :=$REDUCTION$(L_d, G, (F_i)_{d=1,\ldots,(d-1)})$
 for $h \in \tilde{F_d^+}$ ***do***
  $P := P \cup \{\mathrm{Pair}(h, g) \mid g \in G\}$
  $(G, P) :=$UPDATE(G, P, h)
return G

La nouvelle fonction de réduction est identique à la précédente à l'exception d'un nouvel argument en entrée; en sortie on retourne d'une part les nouveaux éléments de la base de Gröbner mais aussi la matrice complète mise sous forme échelon:

Algorithme 27 REDUCTION

Input: $\begin{cases} L, G \text{ sous-ensemble finis de } \mathbb{K}[x_1, \ldots, x_n] \\ \mathcal{F} = (F_k)_{k=1,\ldots,(d-1)}, \text{ où } F_k \\ \quad \text{est un sous ensemble fini de } \mathbb{K}[x_1, \ldots, x_n] \end{cases}$
Output: *deux sous-ensembles fini de* $\mathbb{K}[x_1, \ldots, x_n]$.
$F :=$SYMBOLICPREPROCESSING$(L, G, \mathcal{F})$
$\tilde{F} :=$ *Réduction de Gauß de* F *par rapport à* $<$
$\tilde{F^+} := \left\{ f \in \tilde{F} \mid \mathrm{LT}(f) \notin \mathrm{LT}(F) \right\}$
return $(\mathcal{F}, \tilde{F^+})$

La variante principale avec la version simplifiée de l'algorithme se trouve dans la fonction SYMBOLICPREPROCESSING: c'est ici qu'on opère la substitution d'un produit $m \cdot f$ par un "meilleur" produit $m' \cdot f'$:

Algorithme 28 SYMBOLICPREPROCESSING

Input: $\begin{cases} L, G \text{ sous-ensemble finis de } \mathbb{K}[x_1, \ldots, x_n] \\ \mathcal{F} = (F_k)_{k=1,\ldots,(d-1)}, \text{ où } F_k \\ \quad \text{est un sous ensemble fini de } \mathbb{K}[x_1, \ldots, x_n] \end{cases}$
Output: *un sous-ensemble fini de* $\mathbb{K}[x_1, \ldots, x_n]$
$F := L$
$Done := \mathrm{LT}(F)$
while $T(F) \neq Done$ ***do***
  choisir m *un élément de* $T(F) \backslash Done$
  $Done := Done \cup \{m\}$
  if m *top reductible modulo* G ***then***
    il existe $g \in G$ *et un* $m' \in T$ *tel que* $m = m' \cdot \mathrm{LT}(g)$
    $F := F \cup \{\textsc{Simplify}(m', g, \mathcal{F})\}$
return F

La fonction Simplify cherche à remplacer le produit $m \cdot f$ par un produit $(u\,t) \cdot f'$ où (t, f') est une étiquette de ligne dont on a déjà calculé la forme échelon et $u\,t$ divise le terme m'; si on trouve effectivement un meilleur produit on ré-appelle récursivement la fonction Simplify (voir l'exemple dans la section 4.3.5 pour un exemple d'appel récursif):

Algorithme 29 Simplify

Input: $\begin{cases} t \in T \text{ un terme} \\ f \in \mathbb{K}[x_1, \ldots, x_n] \text{ un polynôme} \\ \mathcal{F} = (F_k)_{k=1,\ldots,(d-1)}, \text{ où } F_k \\ \quad \text{est un sous ensemble fini de } \mathbb{K}[x_1, \ldots, x_n] \end{cases}$
Output: *un élément de la forme* $m' \cdot f'$
for $u \in$ *liste des diviseurs de* t ***do***
 if $\exists j$ $(1 \leq j < d)$ *tel que* $(u \cdot f) \in F_j$ ***then***
  $\tilde{F_j}$ *est la forme échelonnée de* F_j *par rapport à* $<$
  il existe un et un seul $p \in \tilde{F_j}^+$ *tel que* $\mathrm{LT}(p) = \mathrm{LT}(u \cdot f)$
  if $u \neq t$ ***then***
    return $\textsc{Simplify}(\frac{t}{u}, p, \mathcal{F})$
  else
    return $1 \cdot p$
return $t \cdot f$

Note 4.3.4. L'expérience montre que l'effet de Simplify est de retourner, dans 95% des cas, un produit $x_i \cdot p$ où x_i est une variable (et même le plus souvent le produit $x_n \cdot p$ par la dernière variable). Cette technique est un peu similaire à l'algorithme FGLM (voir le chapitre 3.1 page 52) où on utilise des matrices de multiplication pour calculer des formes normales.

Encore une fois on se réfère à l'article original ((Faugère J.C., 1999)) pour la preuve complète de l'algorithme modifié; le théorème suivant permet de se ramener au théorème (2.4.7 p. 26) caractérisant les bases de Gröbner:

Theorem 4.3.2. *Soit F un sous-ensemble fini de $\mathbb{K}[x_1,\dots,x_n]$, $\mathcal{F}=(F_k)_{k=1,\dots,(d-1)}$, où F_k est un sous-ensemble fini de $\mathbb{K}[x_1,\dots,x_n]$, $Pair(g_1,g_2)=(lcm_{1,2},t_1,g_1,t_2,g_2)$ avec $lcm_{1,2},t_1,t_2\in T$ tel que les conditions suivantes soient vérifiées:*

(i) $(\tilde{F}_k)_+\subset G$ *pour* $k=1,\dots,(d-1)$
(ii) $f_i=\textsc{Simplify}(t_i,g_i,\mathcal{F})$ *pour* $i=1,2$.
(iii) $\operatorname{spol}(f_1,f_2)=o_F(\operatorname{lcm}(\operatorname{LT}(f_1),\operatorname{LT}(f_2)))$ *(voir section 2.4.4 page 26).*

Alors $\operatorname{spol}(g_1,g_2)=o_F(lcm_{1,2})$.

Proof. Soit (t_i',g_i') le résultat de $\textsc{Simplify}(t_i,g_i,\mathcal{F})$. D'après le lemme **??** on a $\operatorname{LT}(t_1'g_1')=\operatorname{LT}(t_1g_1)=lcm_{1,2}=\operatorname{LT}(t_2g_2)=\operatorname{LT}(t_2'g_2')$ de telle sorte que (en supposant tous les polynômes unitaires):

$$\begin{aligned}\operatorname{spol}(g_1',g_2') &= t_1'g_1'-t_2'g_2'\\ &= (t_1'g_1'-t_1g_1)+(t_1g_1-t_2g_2)+(t_2g_2-t_2'g_2')\\ &= r+\operatorname{spol}(f_1,f_2)+r'\end{aligned}$$

avec $r,r'\in \operatorname{Id}(\tilde{\mathcal{F}^+}\cup\mathcal{F})\subset Id(G)$ tel que $\max(\operatorname{LT}(r),\operatorname{LT}(r'))<lcm_{1,2}$. Ainsi $\operatorname{spol}(g_1,g_2)=\mathcal{O}_F(t')$ pour $t'=\max(\operatorname{LT}(r),\operatorname{LT}(r'),t)<lcm_{1,2}$. Le résultat découle du théorème 2.4.7.

4.3.4 Fonction de sélection

Le choix d'une bonne fonction $\mathcal{S}el$, c'est à dire une stratégie de calcul, est très important pour les performances de l'algorithme.

Calculer une base de Gröbner pour un ordre du degré est souvent une partie très difficile dans la chaîne des calculs aboutissant à la résolution complète d'un système algébrique. Une explication est que l'entrée de l'algorithme est simplement un sous-ensemble de $\mathbb{K}[x_1,\dots,x_n]$ *sans* structure mathématique. On peut se donner une certaine structure dès le début de l'algorithme en utilisant le concept de d-bases de Gröbner (voir 2.6.2 p. 2.6.2). C'est à dire que la fonction de sélection choisit *toutes* les paires critiques de degré minimale:

Algorithme 30 $\mathcal{S}el$

Input: *P une liste de paires critiques*
Output*: une liste de paires critiques.*
$d:=\min\{\deg(\operatorname{lcm}(p))\mid p\in P\}$
$\tilde{F}:=$ *Réduction de Gauß de F par rapport à $<$*
$P_d:=\{p\in P\mid \deg(\operatorname{lcm}(p))=d\}$
return P_d

On appelle cette stratégies la *stratégie normale pour l'algorithme F_4*. Ainsi si les polynômes de départ son homogènes, on obtient, en degré d, une d base de Gröbner; $\mathcal{S}el$ sélectionne donc à l'étape suivante toutes les paires qui sont nécessaires pour calculer une base de Gröbner jusqu'en degré $d+1$.

Une variante serait de changer $\deg(\mathrm{lcm}(p))$ par $\deg_{Sugar}(\mathrm{lcm}(p))$ où $\deg_{Sugar}$ représente le "sucre" (voir l'article (Giovini A. and Mora T. and Niesi G. and Robbiano L. and Traverso C., 1991)).

4.3.5 Exemple détaillé étape par étape

L'exemple suivant est donné à titre d'illustration; il est malheureusement impossible de montrer le gain en efficacité sur un exemple de petite taille. On considère le problème cyclic 4. On prend comme ordre admissible l'ordre degree reverse lexicographical ordering (DRL) et la stratégie normale (voir aussi section 4.3.4)

$$F = \begin{bmatrix} f_4 = abcd - 1, f_3 = abc + abd + acd + bcd, \\ f_2 = ab + bc + ad + cd, f_1 = a + b + c + d \end{bmatrix}$$

Au départ $G = \{f_4\}$ et $P_1 = \{\mathrm{Pair}(f_3, f_4)\}$ de telle sorte que $L_1 = \{(1, f_3), (b, f_4)\}$.

On rentre dans la fonction SymbolicPreprocessing$(L_1, G, \emptyset)$; $F_1 = L_1$, $Done = \mathrm{LT}(F_1) = \{ab\}$ et $T(F_1) = \{ad, ab, b^2, bc, bd, cd\}$, on choisit un élément dans $T(F_1) \backslash Done$, par exemple ad, mais ad est top réductible par G; ainsi $Done = \{ab, ad\}$, $F_1 = F_1 \cup \{df_4\}$ et $T(F_1) = T(F_1) \cup \{d^2\}$.

Comme les autres éléments de $T(F_1)$ ne sont pas top réductible par G, SymbolicPreprocessing retourne

$$F_1 = [f_3, bf_4, df_4].$$

La représentation matricielle de F est:

$$A_1 = M(F_1) = \begin{bmatrix} 0\,0\,0\,1\,1\,1\,1\,0 \\ 1\,0\,1\,1\,0\,1\,0\,0 \\ \\ 1\,1\,1\,0\,1\,0\,0\,0 \end{bmatrix}$$

et la réduction de Gauß de A_1 est:

$$\tilde{A}_1 = \begin{bmatrix} 0\,0\,0\,1 & 1 & 1 & 1 & 0 \\ 1\,0\,1\,0 & -1 & 0 & -1 & 0 \\ \\ 0\,1\,0\,0 & 2 & 0 & 1 & 0 \end{bmatrix}$$

par conséquent $\tilde{F_1} = [f_5 = ad + bd + cd + d^2, f_6 = ab + bc - bd - d^2, f_7 = b^2 + 2\,bd + d^2]$ et comme $ab, ad \in \mathrm{LT}(F_1)$ on a $\tilde{F_1}_+ = [f_7]$ et maintenant $G = \{f_4, f_7\}$.

Lors de la prochaine étape on doit considérer $P_2 = \{\text{Pair}(f_2, f_4)\}$, ainsi $L_2 = \{(1, f_2), (bc, f_4)\}$ et $\mathcal{F} = \{F_1\}$.

Dans SymbolicPreprocessing on cherche à simplifier les produits $1 \cdot f_2$ et $bc \cdot f_4$ avec $\mathcal{F}$. On voit que $bf_4 \in F_1$ et que f_6 est l'unique polynôme dans $\tilde{F}_1$ tel que $\text{LT}(f_6) = \text{LT}(bf_4) = ab$, ainsi $\text{Simplify}(bc, f_4, \mathcal{F}) = c \cdot f_6$. Maintenant $F_2 = [f_2, cf_6]$ et $T(F_2) = \{abc, bc^2, abd, acd, bcd, cd^2\}$. On choisit abd qui est réductible par bdf_4 mais encore une fois on peut remplacer ce produit par $b \cdot f_5$. Après quelques itérations de l'algorithme on trouve que

$$F_2 = [cf_5, df_7, bf_5, f_2, cf_6]$$

$$A_2 = M(F_2) = \begin{bmatrix} 0\,0\,0\,0\,1 & 1 & 1\,0 & 1 & 0\,0 \\ 0\,0\,0\,1\,0 & 0 & 0\,2 & 0 & 1\,0 \\ 0\,0\,1\,1\,0 & 1 & 0\,1 & 0 & 0\,0 \\ 1\,0\,1\,0\,1 & 1 & 0\,0 & 0 & 0\,0 \\ 1\,1\,0\,0\,0 & -1 & 0\,0 & -1 & 0\,0 \end{bmatrix}$$

$$\tilde{A}_2 = \widetilde{M(F_2)} = \begin{bmatrix} 0\,0\,0\,0\,1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0\,0\,0\,1\,0 & 0 & 0 & 2 & 0 & 1 & 0 \\ 0\,0\,1\,0\,0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 1\,0\,0\,0\,0 & -1 & -1 & 1 & -1 & 1 & 0 \\ 0\,1\,0\,0\,0 & 0 & 1 & -1 & 0 & -1 & 0 \end{bmatrix}$$

$\tilde{F}_2 = [f_9 = acd + bcd + c^2d + cd^2,\ f_{10} = b^2d + 2\,bd^2 + d^3,\ f_{11} = abd + bcd - bd^2 - d^3,\ f_{12} = abc - bcd - c^2d + bd^2 - cd^2 + d^3,\ f_{13} = bc^2 + c^2d - bd^2 - d^3]$ et

$$G = \{f_4, f_7, f_{13}\}.$$

Lors de la prochaine étape on a

$$L_3 = \{(1, f_1), (bcd, f_4), (c^2, f_7), (b, f_{13})\}$$

et on appelle récursivement la fonction Simplify: $\text{Simplify}(bcd, f_4) = \text{Simplify}(cd, f_6) = \text{Simplify}(d, f_{12}) = (d, f_{12})$. On obtient

$$F_3 = [f_1, df_{12}, c^2 f_7, bf_{13}].$$

On remarque que $c^2 f_7$ ne peut être simplifié. Après quelques étapes dans SymbolicPreprocessing on trouve

$$F_3 = [f_1, df_{12}, c^2 f_7, bf_{13}, df_{13}, df_{10}]$$

puis $\tilde{F}_3 = [f_{15} = c^2b^2 - c^2d^2 + 2\,bd^3 + 2\,d^4,\ f_{16} = abcd - 1,\ f_{17} = -bcd^2 - c^2d^2 + bd^3 - cd^3 + d^4 + 1,\ f_{18} = c^2bd + c^2d^2 - bd^3 - d^4,\ f_{19} = b^2d^2 + 2\,bd^3 + d^4]$.

Un rapide calcul montre que le rang de F_3 est seulement 5. Cela signifie qu'il y a une réduction à zéro qui n'a pas été évitée.

4.3.6 Optimisation de l'algèbre linéaire

L'étape de mise sous forme échelon des matrices est l'étape la plus coûteuse en temps d'exécution et en espace mémoire. Dans son implantation dans le système de Calcul Formel *Magma* A. Steel a utilisé de nombreux algorithmes et astuces d'implantation (en particulier l'utilisation de structure adapté à la manipulation de matrices *creuses*). On se contente de décrire brièvement une technique de compression des matrices et le travail réalisé par Sylvain Lachartre pendant sa thèse ((Lachartre, 2006)).

Compression des matrices. Lorsque les matrices sont grosses il est nécessaire d'adopter un shéma de stockage des données plus compliqué afin de réduire l'occupation mémoire: si on considère une matrice $5.10^4 \times 5.10^4$ avec 10% d'éléments non nuls (c'est exactement le cas pour le benchmark Cyclic 9 par exemple); même si on alloue un seul octet pour chaque coefficient (ce qui est plutoˆt optimiste si on considère que les coefficients ont souvent plusieurs centaines de chiffres) il faut $250 * 10^6$ octets pour stocker la matrice. Dans notre implantation on évite de dupliquer les coefficients (en effet la plupart des lignes sont des multiplication d'un même polynôme f par plusieurs monômes); ainsi on a juste a considérer les positions des éléments non nuls des éléments de la matrice: c'est une succession de 1 et 0 qu'il faut compresser (c'est donc une bitmap). Nous avons testé plusieurs méthodes:

(i) Pas de compression: inefficace à la fois d'un point de vue temps de calcul et d'un point de vue mémoire.

(ii) Compression bitmap: si on note par

$$j_1, j_2, j_3, \cdots$$

les positions des éléments non nuls dans une ligne de la matrice, alors $\sum_k 2^{j_k - 1}$ est la forme bitmap. Cette méthode est efficace mais ne compresse par beaucoup la matrice (et seulement d'un facteur constant).

(iii) Une autre technique consiste a considérer les différences entre les positions non nulles:

$$\boxed{j_1}\boxed{j_2 - j_1}\boxed{j_3 - j_2}\cdots$$

quand les la différence $j_k - j_{k-1}$ est petite (< 128), et ceci arrive souvent, on peut la stocker sur un octet. Cette méthode est plus efficace en terme d'occupation mémoire que la précédente et seulement un peu plus lente (10% plus lente).

(iv) On peut aussi appliquer une technique à la gzip sur la représentation précédente: par on peut identifier des *patterns* de r blocs de 1 consécutifs (c'est à dire $j_i = j_{i+1} - 1 = j_{i+2} - 2 = \cdots$. Cette méthode est plus complexe à mettre en oeuvre mais elle est beaucoup plus efficace.

Utilisation du hardware pour optimiser l'implantation.. Sylvain Lachartre montre ((Lachartre, 2006)) qu'on peut gagner un facteur 100 avec une bonne implantation en machine: en effet le principal se situe au niveau de la bande passante pour accéder à la mémoire. Une solution est de rester le plus possible dans le cache de plus bas niveau puisque les accès au cache de niveau 1 se font en un temps comparable avec une opération basique (un `XOR` par exemple). Ainsi il est souvent nécessaire de revoir les algorithmes et de travail sur des blocs de matrices plutôt que sur des lignes ou des colonnes par exemple. Le cas particulier du corps F_2 présente la particularité que les opérations arithmétiques élémentaires se font très rapidement (relativement aux accès mémoire): ainsi sur un Pentium 4 il existe une opération permettant de faire des `XOR` 128 bits en une seule opération.

4.4 Implantations de F_4.Benchmarks

La présente section contient des tests effectués pour comparer l'efficacité de l'algorithme de Buchberger et l'algorithme F; pour d'autres tests incluant F_4 on pourra aussi consulter la section 5.6.3 p. 119. Même si l'algorithme F_4 n'est pas *toujours* plus efficace que l'algorithme de Buchberger (et il est sans doute illusoire de penser qu'un tel algorithme puisse exister) les tests suivants et surtout les exemples provenant des applications réelles (voir notamment chap. 8,9,10,18,12) on montré qu'un gain d'au moins un ordre de grandeur pouvait être obtenu.

4.4.1 Quelques implantations de F_4

Depuis la parution de l'article et son implantation dans le logiciel FGb d'autres implantations ont été réalisées:

- Implantation très efficace dans le logiciel Magma(Cannon J., 1998). L'implantation est générale (corps fini, entiers, ...) et contient même des extensions dans le cas non commutatif et le cas euclidien.
- Dans Maple (Char B. and Geddes K. and Gonnet G. and Leong B. and Monagan M. and Watt S., 1991) sous forme de plugin à partir de la version 9 et intégré dans la distribution à partir de la version 11 (en fait c'est FGb par inclusion par directe du code C au moyen d'une librairie dynamique).
- Sugita, Mitsunari, Watanabe (Sugita *et al.* , 2006) ont réalisé une implantation efficace dans le cas des corps fini. De plus c'est une version dont le code peut être téléchargé à l'adresse suivante:
  http://www.ipa.go.jp/security/fy16/development/crypt_stream/ipa_smw.tar.gz
- Implantation par A. Joux dans le cas du corps $\mathbb{F}_2$.
- Implantation par O. Billet (Billet, 2005) pour les corps finis.
- ...

4.4.2 Les benchmarks d'A. Steel

Les benchmarks sont toujours sujet à caution car il est difficile d'être juge et parti ce qui est le cas lorsqu'on est l'auteur de l'algorithme et de l'implantation. C'est la raison pour laquelle je reproduit intégralement les temps de calculs de la page Web d'Allan Steel (voir (Steel, 2004)) contenant une comparaison de son implantation de l'algorithme F_4 avec d'autres implantations efficaces de l'algorithme de de Buchberger (les temps pour les exemples provenant de la Cryptologie ne sont pas inclus ici).

Machine et version des logiciels. Les tests ont été effectué sur un AMD Athlon XP 2800+ PC (2.087GHz), ayant des tailles de cache L1 64K, D 64K, L2 512K, 1.5GB de mémoire RAM; le système d'exploitation était Redhat Linux 9.0 (kernel version 2.4.20-8). Les logiciels utilisés:

- Magma V2.11-8 (released in September 2004), qui utilise l'algorithme F_4 (nouvelle version).
- Magma V2.11-1 (released in May 2004), qui utilise l'algorithme F_4.
- Magma V2.10, qui implante l'algorithme de Buchberger.

- Singular version 2-0-5 (April 28, 2004) (algorithme de Buchberger).
- Macaulay 2 version 0.9.2 (May 13, 2004) (algorithme de Buchberger).

Tous les calculs de bases de Gröbner sont effectués pour l'ordre DRL (graded reverse lexicographical). Les temps sont en secondes sauf précision contraire. La mention $> T$ signifie que le calcul a été stoppé au bout du temps T (et il est difficile d'avoir une estimation du temps restant); $\gg T$ signifie que le calcul a été stoppé au bout du temps T (sachant que le temps restant était très long).

Calcul modulo un petit nombre premier (A. Steel). Les calculs se font dans le corps $\mathbb{F}_{32003}$ (limitation de Macaulay 2 et de Singular):

	Algorithme F_4		Algorithme de Buchberge		
Logiciel	Magma 2.11-8	Magma 2.11-1	Magma 2.10	Singular 2-0-5	Ma
Katsura 8	0.3	0.6	9.3	2.4	11.8
Katsura 9	1.7	5.8	100.9	22.5	123
Katsura 10	13.1	51.8	1259.3	186.2	134
Cyclic 7 (Hom)	0.4	0.5	13.7	3.7	12.3
Cyclic 8 (Hom)	10.4	24.7	628.4	132.8	643
Cyclic 9	1453.6	5177.9	$\gg$3 days	38298.8	?
Cyclic 10	17.6 days*				

Table 4.1. Comparaison F4 modulo p

* For Cyclic 10 modulo p, the computation was done on a 750MHz Sunfire v880. Only one sequential processor was used, and 30.7GB of memory was needed.

The ideal is zero-dimensional, and the quotient of the polynomial ring by the ideal has dimension 34940 as a vector space. The largest step took 6.4 days and involved partially triangularizing a 544684 by 582742 matrix with density 1.3%.

Calcul à coefficients rationnels (A. Steel). Tous les calculs sont à coefficients dans $\mathbb{Q}$:

	Algorithme F_4		Algorithme de Buchberger		
Logiciel	Magma 2.11-8	Magma 2.11-1	Magma 2.10	Singular 2-0-5	Macaula
Katsura 8	2.2	3.0	26.7	29.6	550.5
Katsura 9	12.2	28.2	291.2	800.0	6884.0
Katsura 10	130.7	266.3	4603.9	22972.2	
Cyclic 7 (Hom)	2.2	2.2	51.1	5174.6	1649.5
Cyclic 8 (Hom)	83.7	134.8	4320.8	$\gg$3 days	>2 days
Cyclic 9	4.6 days*	7.5 days*			

Table 4.2. Comparaison F4 sur les rationnels

* For Cyclic 9 over the rationals, the computation was done on a 750MHz Sunfire v880 and 11GB of memory was needed.

4.5 Conclusion

En résumé on pourrait dire qu'on a reporté les degrés de liberté dans l'algorithme de Buchberger (choix des paires, ...) vers des choix de stratégies pour résoudre efficacement des problèmes d'algèbre linéaire classique. Le gain attendu est qu'on construit une matrice A représentant l'ensemble des calculs en un certain degré: on peut commencer par la trianguläriser par "n'importe quel bout" ou appliquer une stratégie par blocs ou encore distribuer les calculs. Pour les calculs sur les entiers on a aussi l'immense avantage de pouvoir appliquer les p-adique ou multi-modulaire qui permette de s'affranchir des croissances parasites des coefficients (et aussi d'augmenter la localité des accès mémoire). Outre une implantation de l'algorithme qui n'est pas aisée, le point le plus négatif est que la matrice A est souvent gigantesque et qu'il n'est pas facile de maîtriser la gestion en mémoire. Un bon algorithme de compression des lignes des matrices permet, cependant, de diviser par un facteur 10 l'occupation mémoire. Des tests expérimentaux y compris dans divers logiciels ont montré un gain d'au moins un ordre de grandeur par rapport à l'algorithme de Buchberger sur la plupart des systèmes.

5. Critère F_5 et algorithme F_5.

5.1 Introduction

Une première méthode pour améliorer l'efficacité de l'algorithme de Buchberger (Buchberger B., 1965; Buchberger B., 1979; Buchberger B., 1985) a déjà été décrite dans le chapitre 4; le but de ce chapitre est de décrire un nouveau critère pour remplacer les deux critères de Buchberger et éliminer *complètement* toutes les réductions inutiles lors d'un calcul de base de Gröbner (ou de manière équivalente générer des matrices de rang plein pour un algorithme matriciel de type F_4). En effet même si les critères de Buchberger sont très efficaces puisqu'ils permettent d'éviter la majorité des calculs inutiles, pour un pourcentage élevé d'exemples 90% des paires critiques se réduisent à zéro après application des critères de Buchberger. Il faut préciser aussi qu'il n'est pas toujours possible d'éliminer *toutes* les réductions inutiles puisque ce n'est déjà pas vrai pour un système linéaire d'équations singulier (c'est à dire dont le rang n'est pas maximal). Il faut observer qu'on cherche à obtenir ce résultat sans sacrifier l'efficacité des algorithmes existants; ainsi des techniques calculant en même temps que la base de Gröbner le premier module des syzygies (Mora, T. and Möller, H.M. and Traverso, C., 1992) sont à exclure. On verra (section 5.6.3) que l'algorithme F_5 (Faugère J.C., 2002) permet, en pratique, de gagner un ordre de grandeur par rapport à l'algorithme F_4 sur une large classe d'exemples; ceci est également illustré dans plusieurs applications (voir par exemple les chapitres **??**,**??**,18). On se reporte à l'article (Faugère J.C., 2002) pour une comparaison avec des méthodes connexes:

- les critères de Buchberger (Buchberger B., 1979; Buchberger B., 1985) et l'implantation de ces critères (Gebauer & Moller, 1986).
- les "staggered linear bases" (Gebauer & Moller, 1986)
- le calcul simultané de la base de Gröbner et du module des syzygies (Mora, T. and Möller, H.M. and Traverso, C., 1992).
- Le concept des bases involutives (V.P. Gerdt and Yu.A.Blinkov, 1998) peut être vu comme un moyen d'interdire certaines réductions et donc d'éviter des calculs. L'article (J. & R., 2002) présente un analogue du deuxième critère de Buchberger pour le calcul des bases involutives.

Notons toutefois qu'aucune de ces méthodes ne permet de répondre à l'ensemble des spécifications: élimination totale des calculs inutiles et efficacité; et selon les auteurs de l'article (Mora, T. and Möller, H.M. and Traverso, C., 1992): *"many useless pairs are discovered, but it involves a lot of extra computation, so the execution time is increased"*.

La stratégie de l'algorithme F_5 est de calculer incrémentalement degré par degré et équation par équation les d-base de Gröbner des systèmes (f_m), (f_{m-1}, f_m), $\ldots$, $(f_1, \ldots, f_m)$. On montre (voir corollaire 5.4.2) que si le système initial est régulier alors il n'y a pas de réduction à zéro. De plus, en pratique, pour la plupart des systèmes (voir les résultats de la section 5.6.1 et un exemple réel dans le chapitre **??**) il n'y a pas de réduction à zéro. Même si la complexité du pire cas (doublement exponentiel) n'est pas amélioré on constate expérimentalement que pour une large classe de systèmes l'algorithme F_5 est plus rapide que toutes les autres implantation (en particulier l'algorithme F_4) : la section 5.6.3 inclut des tests incluant les algorithmes de Buchberger F_4, F_5. L'idée de l'algorithme est d'abord détaillée pas à pas sur un exemple (5.2) puis une version matricielle (F_5 matriciel) est ensuite décrite: cette version présente le double avantage d'être, d'une part, facilement implantable et déjà très efficace (c'est cette version qui a été utilisé pour "casser" le challenge HFE 1 du chap **??**) et, d'autre part, on peut analyser assez finement la complexité de cet algorithme (chap.).

5.2 L'idée préliminaire à l'algorithme F_5.

Afin d'introduire l'idée on considère un système algébrique de degré 2 en 3 variables x, y, z dépendant d'un paramètre b qui prendra la valeur 0 ou 1:

$$\mathcal{S}_b \begin{cases} f_3 = x^2 + 18\,xy + 19\,y^2 + 8\,xz + 5\,yz + 7\,z^2 \\ f_2 = 3\,x^2 + (7+b)\,x\,y + 22\,x\,z + 11\,yz + 22\,z^2 + 8\,y^2 \\ f_1 = 6\,x^2 + 12\,xy + 4\,y^2 + 14\,xz + 9\,yz + 7\,z^2 \end{cases}$$

On veut calculer la base de Gröbner de f_1, f_2, f_3 modulo 23 pour un ordre du degré tel que $x > y > z$. Si on applique l'algorithme de Buchberger (avec les critères de Buchberger) il y a 5 paires critiques utiles et 5 paires inutiles (qui se réduisent à 0). Dans un premier temps on suppose $b = 0$. Pour calculer la base de Gröbner on procède degré par degré. En degré 2 on construit directement la représentation matricielle (chap 4 section 4.2.1) de $[f_1, f_2, f_3]$:

$$A_2 = \begin{array}{c|cccccc} & x^2 & x\,y & y^2 & x\,z & y\,z & z^2 \\ f_3 & 1 & 18 & 19 & 8 & 5 & 7 \\ f_2 & 3 & 7 & 8 & 22 & 11 & 22 \\ f_1 & 6 & 12 & 4 & 14 & 9 & 7 \end{array}$$

ce qui donne après triangulation de la matrice A_2 (contrairement au chapitre 4 on calcule des triangulations de matrice et pas la forme échelonnée

complète, la raison en sera explicité plus bas):

$$B_2 = \begin{array}{c} \\ f_3 \\ f_2 \\ f_1 \end{array} \begin{array}{|cccccc|} x^2 & x\,y & y^2 & x\,z & y\,z & z^2 \\ 1 & 18 & 19 & 8 & 5 & 7 \\ & 1 & 3 & 2 & 4 & -1 \\ & & 1 & -11 & -3 & -5 \end{array}$$

(afin de mieux faire ressortir la structure des matrices on remplace un 0 par un espace). Donc on a construit deux "nouveaux" polynômes dans l'idéal $f_4 = xy + 4\,yz + 2\,xz + 3\,y^2 - z^2$ et $f_5 = y^2 - 11\,xz - 3\,yz - 5\,z^2$.
En degré 3 on pourrait construire la représentation matricielle de

$$[x\,f_1, y\,f_1, z\,f_1, x\,f_2, y\,f_2, z\,f_2, x\,f_3, y\,f_3, z\,f_3]$$

et obtenir la matrice suivante:

$$A_3 = \begin{array}{c} \\ zf_3 \\ yf_3 \\ xf_3 \\ zf_2 \\ yf_2 \\ xf_2 \\ zf_1 \\ yf_1 \\ xf_1 \end{array} \begin{array}{|cccccc|} x^3 & x^2y & xy^2 & y^3 & x^2z & \dots \\ 0 & 0 & 0 & 0 & 1 & \dots \\ 0 & 1 & 18 & 19 & 0 & \dots \\ 1 & 18 & 19 & 0 & 8 & \dots \\ 0 & 0 & 0 & 0 & 3 & \dots \\ 0 & 3 & 7 & 8 & 0 & \dots \\ 3 & 7 & 8 & 0 & 22 & \dots \\ 0 & 0 & 0 & 0 & 6 & \dots \\ 0 & 6 & 12 & 4 & 0 & \dots \\ 6 & 12 & 4 & 0 & 14 & \dots \end{array}$$

Pour trianguler la matrice on peut réduire les lignes xf_2 et xf_1 par la ligne xf_3. Mais ce serait une perte de temps puisque ceci a déjà été réalisé à l'étape précédente: par exemple $f_4 = -f_2 + 3f_3$, et donc $xf_4 = -xf_2 + 3xf_3$. C'est une idée importante de l'algorithme de Buchberger: on réutilise le plus possible ce qui a déjà été calculé en degré plus petit. Il est clair qu'on ne doit pas mettre f_1 et f_4 *simultanément* dans la même matrice car ces deux polynômes ne sont pas linéairement indépendants. Par conséquent on remplace, dans A_3, f_2 (resp. f_1) par f_4 (resp. f_5) et on calcule la représentation matricielle de $[x\,f_5, y\,f_5, z\,f_5, x\,f_4, y\,f_4, z\,f_4, x\,f_3, y\,f_3, z\,f_3]$:

$$A_3' = \begin{array}{c} \\ zf_3 \\ yf_3 \\ xf_3 \\ zf_4 \\ yf_4 \\ xf_4 \\ zf_5 \\ yf_5 \\ xf_5 \end{array} \begin{array}{|ccccccccccc|} x^3 & x^2y & xy^2 & y^3 & x^2z & xyz & y^2z & xz^2 & yz^2 & z^3 \\ & & & & 1 & 18 & 19 & 8 & 5 & 7 \\ & 1 & 18 & 19 & 0 & 8 & 5 & 0 & 7 & 0 \\ 1 & 18 & 19 & 0 & 8 & 5 & 0 & 7 & 0 & 0 \\ & & & & & 1 & 3 & 2 & 4 & 22 \\ & & 1 & 3 & 0 & 2 & 4 & 0 & 22 & 0 \\ & 1 & 3 & 0 & 2 & 4 & 0 & 22 & 0 & 0 \\ & & & & & & 1 & 12 & 20 & 18 \\ & & & 1 & 0 & 12 & 20 & 0 & 18 & 0 \\ & & 1 & 0 & 12 & 20 & 0 & 18 & 0 & 0 \end{array}$$

Après triangulation:

$$\widetilde{A_3'} = \begin{array}{c|cccccccccc} & x^3 & x^2y & xy^2 & y^3 & x^2z & xyz & y^2z & xz^2 & yz^2 & z^3 \\ \hline xf_3 & 1 & 18 & 19 & 0 & 8 & 5 & 0 & 7 & 0 & 0 \\ yf_3 & & 1 & 18 & 19 & 0 & 8 & 5 & 0 & 7 & 0 \\ yf_2 & & & 1 & 3 & 0 & 2 & 4 & 0 & 22 & 0 \\ \mathbf{xf_2} & & & & \mathbf{1} & 0 & 0 & 8 & 1 & 18 & 15 \\ zf_3 & & & & & 1 & 18 & 19 & 8 & 5 & 7 \\ zf_2 & & & & & & 1 & 3 & 2 & 4 & 22 \\ zf_1 & & & & & & & 1 & 12 & 20 & 18 \\ \mathbf{yf_1} & & & & & & & & \mathbf{1} & 11 & 13 \\ \mathbf{xf_1} & & & & & & & & & \mathbf{1} & 18 \end{array}$$

Donc on construit 3 nouveaux polynômes (ligne dont l'index est en fonte grasse) comme par exemple $f_6 = y^3 + 8\,y^2z + xz^2 + 18\,yz^2 + 15\,z^3$; il faut noter que ce polynôme provient, en fait, de la ligne étiquetée par $x\,f_4$ elle même équivalente à $x\,f_2$; plus exactement on sait qu'on peut trouver un polynôme $g_{6,3}$ de degré 1 tel que

$$f_6 = (\alpha_6\,x + \cdots)\;f_2 + g_{6,3}\,f_3 \text{ avec } \alpha_6 \in \mathbb{K}$$

et de manière similaire il existe $(g_{8,2}, g_{8,3}, g_{7,2}, g_{7,3}) \in \mathbb{K}[x_1, \ldots, x_n]^4$ tels que:

$$f_7 = \left(\alpha_7 \boxed{\text{y}} + \cdots\right)\;f_1 + g_{7,2}\,f_2 + g_{7,3}\,f_3 \text{ avec } \alpha_7 \in \mathbb{K} \tag{5.1}$$

$$f_8 = \left(\alpha_8 \boxed{\text{x}} + \cdots\right)\;f_1 + g_{8,2}\,f_2 + g_{8,3}\,f_3 \text{ avec } \alpha_8 \in \mathbb{K} \tag{5.2}$$

Pour la suite de l'algorithme on peut noter également que si on remplace f_8 par une combinaison linéaire faisant intervenir les lignes f_7, f_8, f_6 alors l'écriture (5.2) est préservée: si $f_8' = \beta_8\,f_8 + \beta_7\,f_7 + \beta_6\,f_6$ (avec $\beta_8 \neq 0$) alors

$$f_8' = \left(\alpha_8' \boxed{\text{x}} + \cdots\right)\;f_1 + g_{8,2}'\,f_2 + g_{8,3}'\,f_3 \text{ avec } \alpha_8' \in \mathbb{K} \text{ et } \left(g_{8,2}', g_{8,3}'\right) \in \mathbb{K}[x_1, \ldots, x_n]^2$$

En revanche ceci n'est pas vrai si on modifie la ligne f_7 par $f_7' = \gamma_8\,f_8 + \gamma_7\,f_7 + \gamma_6\,f_6$ la structure (5.1) n'est pas conservée:

$$f_7' = \left(\alpha_7' \boxed{\text{x}} + \cdots\right)\;f_1 + g_{7,2}'\,f_2 + g_{7,3}'\,f_3 \text{ avec } \alpha_7' \in \mathbb{K} \text{ et } \left(g_{7,2}', g_{7,3}'\right) \in \mathbb{K}[x_1, \ldots, x_n]^2$$

Dans l'algorithme on pourra donc réduire la ligne f_8 par f_7 mais pas l'inverse; autrement dit on pourra réduire la ligne f_8 d'étiquette $x\,f_1$ par une ligne dont l'étiquette est strictement plus petite. Ceci explique pourquoi on se contente d'une forme triangulaire et pas d'une forme échelonnée.

En degré 4 un nouveau phénomène apparaît: la représentation matricielle de:

$$\left[x^2 f_i, x\,y f_i, y^2 f_i, x\,z f_i, y\,z f_i, z^2 f_i,\ i = 1, 2, 3\right]$$

n'est pas de rang plein ! (ceci correspond à 3 paires critiques inutiles dans l'algorithme de Buchberger). La raison en est que partant de la relation triviale $f_2 f_3 - f_3 f_2 = 0$ on peut la réécrire:

$$\begin{aligned} &3\,x^2\,f_3 + (7+b)\,xy\,f_3 + 8\,y^2\,f_3 + 22\,xz\,f_3 + 11\,yz\,f_3 + 22\,z^2\,f_3 \\ &\quad - \boxed{x^2\,f_2} - 18\,xy\,f_2 - 19\,y^2\,f_2 - 8\,xz\,f_2 - 5\,yz\,f_2 - 7\,z^2\,f_2 = 0 \end{aligned}$$

Cette écriture permet d'expliciter la dépendance linéaire entre les lignes et même, plus précisément, quelle ligne on peut retirer de la matrice: ici on enlève $x^2 f_2$ de A_4. En utilisant la deuxième relation triviale $f_1 f_3 - f_3 f_1 = 0$ on peut retirer de la même façon la ligne $x^2 f_1$ de la matrice A_4. Comme il existe une dernière relation triviale $f_1 f_2 = f_2 f_1$ on sait qu'il existe une autre ligne inutile dans la matrice A_4 mais comme on a déjà retiré $x^2 f_1$ quelle autre ligne peu on retirer ? peut on retirer $x\,y\,f_1$? Pour cela il faut combiner les relations triviales entre elles (et dans ce calcul on reprend les polynômes originaux de $\mathcal{S}_b$ avec $b \in \{0,1\}$):

$$\begin{aligned} 0 &= (f_2 f_1 - f_1 f_2) - 3(f_3 f_1 - f_1 f_3) \\ 0 &= (f_2 - 3f_3) f_1 - f_1 f_2 + 3 f_1 f_3 \\ 0 &= f_4 f_1 - f_1 f_2 + 3 f_1 f_3 \\ 0 &= \Big((1-b)\boxed{x\,y} + 4\,yz + 2\,xz + 3\,y^2 - z^2\Big)\, f_1 \\ &\quad -(6x^2 + \cdots)\, f_2 + 3(6x^2 + \cdots)\, f_3 \end{aligned}$$

On en déduit qu'on peut retirer de la matrice A_4 la ligne $x\,y\,f_1$lorsque que $b \neq 0$ et la ligne $y\,z\,f_1$ lorsque $b = 0$. Par conséquent des calculs sont nécessaire pour savoir quelle est la ligne à éliminer (car cela dépend de la *valeur* de b). Plus généralement, si on combine les trois relations triviales $f_i f_j = f_j f_i$ on obtient:

$$u(f_2 f_1 - f_1 f_2) + v(f_3 f_1 - f_1 f_3) + w(f_2 f_3 - f_3 f_2) = 0$$

où u, v, w sont des polynômes quelconques. En regroupant les termes:

$$(uf_2 + vf_3) f_1 - u f_1 f_2 - v f_1 f_3 + w f_2 f_3 - w f_3 f_2 = 0$$

Ainsi toutes les relations triviales $h f_1$ sont telles que h est dans l'idéal engendré par f_2 et f_3. Si on calcule une base de Gröbner G_{prev} de l'idéal engendré (f_2, f_3) on élimine les lignes $m\,f_1$ où m est un monôme divisible par le terme de tête d'un élément de G_{prev}. Plus généralement, si on a calculé une base de Gröbner de $(f_2, \ldots, f_m)$, lors du calcul de la base de Gröbner de $(f_1) + G_{\text{prev}}$ on va construire des matrices dont les étiquettes des lignes sont de la forme $m\,f_1$ avec m un monôme irréductible par rapport à G_{prev}.

Pour terminer l'exemple (on suppose à nouveau $b = 0$) et afin d'utiliser les calculs en degré 2 et 3 on applique les règles de réécriture (dans cet ordre):

$$x\,f_2 \to f_6,\ \ f_2 \to f_4$$
$$x\,f_1 \to f_8,\, y\,f_1 \to f_7$$
$$f_1 \to f_5$$

Maintenant les lignes de la matrice A_4 sont donc

$$\left[y f_7, z f_8, z f_7, z^2 f_5, y f_6, y^2 f_4, z f_6, y\,z f_4, z^2 f_4, x^2 f_3, x\,y f_3, y^2 f_3, x\,z f_3, y\,z f_3, z^2 f_3\right]$$

et la matrice est

$$A_4 = \begin{bmatrix}
1 & 18 & 19 & 0 & 0 & 8 & 5 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 0 \\
 & 1 & 18 & 19 & 0 & 0 & 8 & 5 & 0 & 0 & 7 & 0 & 0 & 0 & 0 \\
 & & 1 & 18 & 19 & 0 & 0 & 8 & 5 & 0 & 0 & 7 & 0 & 0 & 0 \\
 & & & 1 & 3 & 0 & 0 & 2 & 4 & 0 & 0 & 22 & 0 & 0 & 0 \\
 & & & & 1 & 0 & 0 & 0 & 8 & 0 & 1 & 18 & 0 & 15 & 0 \\
 & & & & & 1 & 18 & 19 & 0 & 8 & 5 & 0 & 7 & 0 & 0 \\
 & & & & & & 1 & 18 & 19 & 0 & 8 & 5 & 0 & 7 & 0 \\
 & & & & & & & 1 & 3 & 0 & 2 & 4 & 0 & 22 & 0 \\
 & & & & & & & & 1 & 0 & 0 & 8 & 1 & 18 & 15 \\
 & & & & & & & & & 1 & 18 & 19 & 8 & 5 & 7 \\
 & & & & & & & & & & 1 & 11 & 0 & 13 & 0 \\
 & & & & & & & & & & & 1 & 12 & 20 & 18 \\
 & & & & & & & & & & & & 1 & 11 & 13 \\
 & & & & & & & & & & & & & 1 & 18 \\
 & & & & & & & & & & 1 & 3 & 2 & 4 & 22
\end{bmatrix}$$

Ainsi A_4 est pratiquement triangulaire si on excepte un bloc 5×5 :

$$A_4' = \begin{array}{c|ccccc}
 & xyz^2 & y^2z^2 & xz^3 & yz^3 & z^4 \\
\hline
z^2 f_4 & 1 & 3 & 2 & 4 & 22 \\
z^2 f_5 & & 1 & 12 & 20 & 18 \\
z f_7 & & & 1 & 11 & 13 \\
z f_8 & & & & 1 & 18 \\
y f_7 & 1 & 11 & & 13 &
\end{array} \tag{5.3}$$

La réduction de la matrice donne le nouveau polynôme $f_9 = z^4$ et comme il comme correspond à la ligne d'étiquette $y\,f_7 \approx y^2\,f_1$ on sait qu'on peut trouver une écriture de f_9 de la forme:

$$f_9 = \left(\alpha_9\, y^2 + \cdots\right)\, f_1 + (\cdots)\, f_2 + (\cdots)\, f_3.$$

Dans cet exemple le calcul se fait sans aucune réduction à zéro. Il en ressort également que dans l'algorithme il est nécessaire de conserver l'étiquette originelle de chaque ligne pour chaque polynôme calculé: se sera la "signature" (voir section **??**). Par exemple la signature de f_6 sera $x f_2$. La description des règles de simplification fait l'objet de la section **??**.

5.3 Algorithme F_5 matriciel

Le but de cette section est la description de la version matricielle de l'algorithme F_5 qui sera adaptée à une analyse de complexité et efficace pour des systèmes denses (comme par exemple 8). Une différence fondamentale avec la version originale de l'algorithme F_5 ((Faugère J.C., 2002)) est que, d'une part le degré maximal atteint par les calculs doit être donné en entré de l'algorithme (c'est le paramètre D dans la description de l'algorithme 31); d'autre part on ne tient pas compte du caractère éventuellement creux des polynômes. L'algorithme qui en résulte est particulièrement simple à décrire et à implanter.

5.3.1 Représentation matricielle avec étiquette

Comme cela a été illustré dans la section 5.2, il nécessaire d'adjoindre à chaque polynôme, et donc à chaque ligne de matrice, une *étiquette;* pour cette raison il nous faut étendre la définition de la représentation matricielle 4.2.1:

Definition 5.3.1. *Si $F = [f_1, \ldots, f_m]$ est un vecteur de m polynômes, $e = [e_1, \ldots, e_m]$ un vecteur de m étiquettes et $<$ un ordre admissible, $\mathrm{T}_<(F) = [t_1, \ldots, t_l]$ les termes du support de F triés pour l'ordre $<$. Alors une représentation matricielle $M_{e,T_<(F)}(F)$ de F est une matrice dont les lignes sont indexées par $[e_1, \ldots, e_m]$:*

$$M_{e,T_<(F)}(F) = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & \cdots \end{array} \\ \left|\begin{array}{ccc} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{array}\right. \end{array} \begin{array}{l} \\ f_1 \\ f_2 \\ f_3 \end{array}$$

et dont le coefficient d'indice (e_i, j) est le coefficient du terme t_j dans f_i . Dans la notation précédente on ajoute la colonne de droite simplement pour indiquer l'origine des coefficients de la matrice; cette colonne est donc optionnelle dans la notation. De plus, $M_{e,T_<(F)}(F)$ vérifie l'équation:

$$F = M_{e,T_<(F)}(F) \cdot T_<(F)$$

Pour alléger les notations on note $M_e(F)$ ou $M(F)$ la matrice $M_{e,T_<(F)}(F)$.

Réciproquement si M_e est une matrice étiquettée par $e = [e_1, \ldots, e_m]$, de taille $m \times l$, à coefficients dans $\mathbb{K}$, et $X = [t_1, \ldots, t_l]$ un vecteur de termes alors la représentation polynomiale étiquetée de M_e par rapport à X est le vecteur de m polynômes déterminé par l'équation $F = M \cdot X$ et d'étiquettes e.

Par exemple la matrice $A_4^{'}$ (équation 5.3) d'étiquettes $\left[z^2 f_2, z^2 f_1, yz\, f_1, xz\, f_1, y^2\, f_1\right]$:

$$A_4^{'} = \begin{array}{c|ccccc|c} & xyz^2 & y^2z^2 & xz^3 & yz^3 & z^4 & \\ z^2f_2 & 1 & 3 & 2 & 4 & 22 & z^2f_4 \\ z^2f_1 & & 1 & 12 & 20 & 18 & z^2f_5 \\ yz\,f_1 & & & 1 & 11 & 13 & zf_7 \\ xz\,f_1 & & & & 1 & 18 & zf_8 \\ y^2\,f_1 & 1 & 11 & & 13 & 0 & yf_7 \end{array}$$

devient après triangulation (on utilise la même notation $\widetilde{\ }$ mais on autorise uniquement une réduction de ligne par une ligne dont l'étiquette est strictement plus petite):

$$\widetilde{A_4^{'}} = \begin{array}{c|ccccc|c} & xyz^2 & y^2z^2 & xz^3 & yz^3 & z^4 & \\ z^2f_2 & 1 & 3 & 2 & 4 & 22 & z^2f_4 \\ z^2f_1 & & 1 & 12 & 20 & 18 & z^2f_5 \\ yz\,f_1 & & & 1 & 11 & 13 & zf_7 \\ xz\,f_1 & & & & 1 & 18 & zf_8 \\ y^2\,f_1 & & & & & 1 & \end{array}$$

et la dernière ligne de cette matrice $\widetilde{A_4^{'}}$ correspond au polynôme z^4 d'étiquette $y^2\,f_1$. Dans la description de l'algorithme il sera commode d'employer une notation simple pour décrire la façon de rajouter une ligne $f \in \mathbb{K}[x_1, \ldots, x_n]$ d'étiquette ε à une matrice étiquetée $M_e(F)$:

$$M_{e+[\varepsilon]}(F + [f]) = \begin{array}{c|c|c} & M_e(F) & \\ \varepsilon & \cdots & f \end{array}$$

Par exemple

$$xz\,f_1 \left| \begin{array}{c} \widetilde{A_4^{'}} \\ \cdots \end{array} \right| z^4$$ désigne la matrice:

$$\begin{array}{c|ccccc|c} & xyz^2 & y^2z^2 & xz^3 & yz^3 & z^4 & \\ z^2f_2 & 1 & 3 & 2 & 4 & 22 & z^2f_4 \\ z^2f_1 & & 1 & 12 & 20 & 18 & z^2f_5 \\ yz\,f_1 & & & 1 & 11 & 13 & zf_7 \\ xz\,f_1 & & & & 1 & 18 & zf_8 \\ xz\,f_1 & & & & 1 & 18 & zf_8 \\ y^2\,f_1 & & & & & 1 & \end{array}$$

(remarquer qu'on trie les lignes et les colonnes).

5.3.2 Algorithme F_5 matriciel

On va décrire maintenant l'algorithme F_5 matriciel. Afin d'unifier le cas général et le cas particulier $\mathbb{F}_2$ on utilise la notation suivante: $\delta_{\mathbb{K},\mathbb{F}_2}$, le symbole de Kornecker, est égal à 1 si $\mathbb{K} = \mathbb{F}_2$ et 0 sinon. En effet dans le cas où cherche à calculer une base de Gröbner sur $\mathbb{F}_2$d'un système algébrique

$[f_1, \ldots, f_m]$ on doit ajouter les équations de corps $x_i^2 - x_i$. Par conséquent, il faut tenir compte de nouvelles relations triviales

$$f^2 = f$$

provenant de l'action du morphisme de Frobenius (voir le chapitre 6 et la section **??** pour la justification du nouveau critère). À noter que c'est cette version de F_5 qui a été utilisée pour résoudre le challenge 1 de HFE (voir les chapitres **??** et 18.1).

Avec ces notations on peut décrire très simplement l'algorithme F_5 matriciel:

Algorithme 31 *Agorithme F_5 matriciel.*

Input: $\begin{cases} \text{le corps } \mathbb{K} \\ F = [f_1, \ldots, f_m] \text{ polynômes de degrés total } d_1 \leq \cdots \leq d_m, \\ \text{un entier } D \end{cases}$

Output: *une D-base de Gröbner de F pour un ordre admissible $<$.*

$M^{(*)}([]) := \emptyset$, $\widetilde{M^{(*)}}([]) := \emptyset$

for d ***from*** d_1 ***to*** D ***do*** // Boucle sur le degré

  for i ***from*** 1 ***to*** m ***do*** // Boucle sur les équations

    // Construire la nouvelle matrice $M^{(d)}([f_1, \ldots, f_i])$:

    if $d = d_i$ ***then***

      $M^{(d)}([f_1, \ldots, f_i]) := {}_{f_i}\left|\begin{matrix}\widetilde{M^{(d)}}([f_1, \ldots, f_{i-1}]) \\ \cdots\end{matrix}\right|_{f_i}$

    else

      $M^{(d)}([f_1, \ldots, f_i]) := \widetilde{M^{(d)}}([f_1, \ldots, f_{i-1}])$

      // $J_{Critères}$ *est un l'*idéal monomial:

      $J_{Critères} := \mathrm{Id}\left(\mathrm{LT}\left(\widetilde{M^{(d-d_i)}}([f_1, \ldots, f_{i-1+\delta_{\mathbb{K},\mathbb{F}_2}}])\right)\right)$

      for *toute ligne f d'étiquette $t\, f_i$ dans* $\widetilde{M^{(d-1)}}([f_1, \ldots, f_i])$ ***do***

        Soit k le plus grand entier tel que x_k divise t

        for j ***from*** $k + \delta_{\mathbb{K},\mathbb{F}_2}$ ***to*** n ***do***

          if $t\, x_j \notin J_{Critères}$ ***then***

            // Si $\mathbb{K} = \mathbb{F}_2$ *alors* dans la matrice suivante on

            // remplace dans le produit $x_j f$ les carrés x_i^2 par x_i.

            $M^{(d)}([f_1, \ldots, f_i]) := {}_{tx_jf_i}\left|\begin{matrix}\widetilde{M^{(d)}}([f_1, \ldots, f_i]) \\ \cdots\end{matrix}\right|_{x_j f}$

    Calculer $\widetilde{M^{(d)}}([f_1, \ldots, f_i])$ *la réduction de Gauß (en respectant l'ordre des étiquettes).*

return *la représentation polynomiale de* $\widetilde{M^{(D)}}([f_1, \ldots, f_m])$

5.4 Algorithme F_5 version simplifiée

Soit I' un idéal de $\mathbb{K}[x_1, \ldots, x_n]$; on suppose qu'on a déjà calculé une base de Gröbner G'de I' pour un ordre fixé $<$ et qu'on veut calculer une base de Gröbner de l'idéal $(f_1) + I'$pour $f_1 \in \mathbb{K}[x_1, \ldots, x_n]$. Les différences entre l'algorithme de cette section avec l'algorithme F_5 matriciel (31) de la section précédente sont:

1. il n'est pas nécessaire de donner le degré maximal D des calculs.
2. c'est une version creuse de l'algorithme matriciel (on ne considère pas systématiquement le produit par toutes les variables).

5.4.1 Signature d'un polynôme

On veut tout d'abord définir une signature unique et canonique pour tous les éléments de $T(I) \subset T$ les termes de tête de tous les polynômes de l'idéal I.

Definition 5.4.1. *Pour tout p dans l'idéal I on définit $w(p)$ comme étant $min_<\{g_1 \in \mathbb{K}[x_1, \ldots, x_n] \,|\, (g_1 f_1 - p) \in I'\} = min_<((I' : (f_1))/I')$. On définit $v_1(p)$ comme étant le terme de tête de $w(p)$:*

$$v_1 : \begin{pmatrix} I \longrightarrow & T(I) \\ p \longmapsto & v_1(p) = \mathrm{LT}_<(w(p)) \end{pmatrix}.$$

Proposition 5.4.1. *Si p_1 et p_2 sont deux polynômes de I tels que $\mathrm{LT}(p_1) \neq \mathrm{LT}(p_2)$ alors $v_1(p_1) \neq v_1(p_2)$.*

Dans l'algorithme F_5, $v_1(p)$ sera la *signature* du polynôme p: elle est unique et ne dépend pas des calculs. Le nouveau critère utilise la signature pour éliminer des paires critiques. Par rapport à la version matricielle de F_5 la signature correspond à l'étiquette associée à chaque ligne de la matrice. Informatiquement il faut inclure la signature dans la représentation interne des polynômes; mathématiquement on doit "agrandir" l'anneau des polynômes:

Definition 5.4.2. *On définit un nouvel anneau de polynômes $\mathcal{R} = T \times \mathbb{K}[x_1, \ldots, x_n]$. Pour tout $r \in \mathcal{R}$,si $r = (t, f)$ alors on définit la projection $\pi(r) = f \in \mathbb{K}[x_1, \ldots, x_n]$ et signature de r comme étant $\mathcal{S}(r) = t \in T$.*

On verra que durant l'execution de l'algorithme F_5 on aura toujours $\mathcal{S}(r) = v_1(\pi(r))$.

Definition 5.4.3. *(admissiblilité) On dit que $r \in \mathcal{R}$ est admissible s'il existe $g_1 \in \mathbb{K}[x_1, \ldots, x_n]$ tel que $g_1 f_1 - \pi(r) \in I'$ et $\mathrm{LT}(g_1) = \mathcal{S}(r)$.*

Note 5.4.1. Avec les notations de l'exemple 5.2, p. 98 on a: $I' = \mathrm{Id}(f_3)$ et on considère $I = (f_2) + I'$:

$$r_2 = (1, 3\,x^2 + 7x\,y + 22\,x\,z + 11\,yz + 22\,z^2 + 8\,y^2)$$
$$r_6 = (x, y^3 + 8\,y^2 z + xz^2 + \cdots) = (x + \cdots) f_2 + (\cdots) f_3$$

Note 5.4.2. Pour tout $r \in R$ tel que $\pi(r) \in I'$ on a $\mathcal{S}(r) = 0$.

Definition 5.4.4. *Soient* $0 \neq \lambda \in \mathbb{K}$, $v \in T$, *et* $r = (u, p) \in \mathcal{R}$ *on définit* $\lambda r = (u, \lambda p)$ *et* $vr = (uv, vp)$.

Note 5.4.3. On doit aussi étendre la définition des opérateurs usuels sur $\mathcal{R}$:

pour $r \in \mathcal{R}$, $\mathrm{LT}(r) = \mathrm{LT}(\pi(r))$ et $\mathrm{LC}(r) = \mathrm{LC}(\pi(r))$.
pour $r \in \mathcal{R}$, et $G \subset \mathbb{K}[x_1, \ldots, x_n]$, $\mathrm{NF}(r, G, <) = (\mathcal{S}(r), \mathrm{NF}(\pi(r), G, <))$.
pour $r, r' \in \mathcal{R}^2$, $\mathrm{lcm}(r, r') = \mathrm{lcm}(\mathrm{LT}(\pi(r)), \mathrm{LT}(\pi(r')))$
R est une base de Gröbner si $\pi(R)$ est une base de Gröbner.

5.4.2 Réductibilté, paire critique, Spolynôme au sens de F_5

Dans le reste de ce chapitre on restreint les réduction valides:

Definition 5.4.5. *(normalized)*
On dit que $r \in \mathcal{R}$ *est normalisé si* $\mathcal{S}(r)$ *est* 0 *ou n'est pas top-réductible par* I'. *On dit que* $(u, r) \in T \times \mathcal{R}$ *est normaliśe si* $u\,r$ *est normaliśe. On dit encore qu'une paire critique orientée* $(r, r') \in \mathcal{R}^2$ *est normaliśee si les deux conditions sont vraies:*

(i) $u'\mathcal{S}(r') \prec u\mathcal{S}(r)$
(ii) (u, r) *et* (u', r') *sont normalisées où* $u = \frac{lcm(r,r')}{\mathrm{LT}(r)}$, $u' = \frac{lcm(r,r')}{\mathrm{LT}(r')}$.

Note 5.4.4. Si G' est une base de Gröbner de I' l'implantation de la définition précédente est immédiate: r est normalisé si et seulement si $\mathcal{S}(r)$ n'est pas reductible par G'.

Note 5.4.5. (suite de l'exemple) $r_6 = (x, y^3 + 8\,y^2z + xz^2 + 18\,yz^2 + 15\,z^3)$
$x \times r_6$ n'est pas normalisé car $x\,\mathcal{S}(r_6) = x^2$ et $x^2 \in \mathrm{LT}(f_3)$
$y \times r_6$ est normalisé car $y\,\mathcal{S}(r_6) = xy$ et $xy \notin \mathrm{LT}(f_3)$

Definition 5.4.6. *(S-polynôme)*
Soit (r, r') *une paire critique normalisée alors*

$$\mathrm{Spol}(r, r') = \left(\frac{\mathrm{lcm}(r, r')}{\mathrm{LT}(r)}\mathcal{S}(r), \mathrm{Spol}(\pi(r), \pi(r'))\right) \in \mathcal{R}$$

Definition 5.4.7. *(strong reduction)*
Soient $r, r', r'' \in \mathcal{R}$ *et* R *un sous ensemble de* $\mathcal{R}$. *Alors on dit que*

(i) r *est fortement réductible par* r', *si* $\mathrm{LT}(r')$ *divise* $\mathrm{LT}(r)$ *et*

$$\frac{\mathrm{LM}(r)}{\mathrm{LM}(r')}\mathcal{S}(r') \leq \mathcal{S}(r)$$

Dans ce cas la réduction r *par* r' *est:*

$$r \xrightarrow{r'} (\mathcal{S}(r), \pi(r) - \frac{\mathrm{LM}(r)}{\mathrm{LM}(r')}\pi(r')) \in \mathcal{R}$$

(iii) est fortement réductible par R s'il existe $r' \in R$ tel que r est fortement réductible par r'. Notation

$$r \xrightarrow{R} r''$$

$\xrightarrow{R^*}$ *est la cloture réflexive transitive de* $\xrightarrow{R}$.

Proposition 5.4.2. *Soient $r, r' \in \mathcal{R}$ et R un sous ensemble de $\mathcal{R}$. Alors*

(i) si la paire (r, r') est normalisée $\mathrm{Spol}(r, r')$ *est admissible.*
(ii) pour $\lambda \in \mathbb{K}$ et $t \in T$, $(\lambda t)\, r$ est admissible.
(iii) si $r \xrightarrow{R^*} r'$ alors il existe $t \leq \mathcal{S}(r')$ tel que $(t, \pi(r'))$ est admissible.

5.4.3 Nouveau critère F_5

Le théorème principal de ce chapitre est aussi une nouvelle caractérisation des bases de Gröbner: c'est une version plus forte du théorème 2.4.1:

Theorem 5.4.1. *(critère F_5) Soit G' une base de Gröbner d'un idéal I' et f_1 un polynôme. Soit R un sous ensemble fini de $\mathcal{R}$ tel que:*

(i) tous les éléments de $r \in R$ sont admissibles et unitaires.
(ii) $\{r \in R \mid \mathcal{S}(r) = 0\} = G'$ et $(1, f_1) \in R$.
(iii) pour toute paire critique normalisée $(r, r') \in R$, on a $\mathrm{Spol}(r, r') \xrightarrow{R^*} 0$.

Alors R est une base de Gröbner (non réduite) de $I = (f_1) + I'$.

Proof. **New**
We may suppose that $f_1 \notin I'$ (if $f_1 \in I'$ the theorem is obvious). Let n_R (resp. $n_{G'}$) be the number of elements of R (resp. G') et $R = [r_1, \ldots, r_{n_R}]$. Let g_i be $\pi(r_i)$. By hypothesis, r_i is admissible, hence $\pi(r_i) = g_i = w_i f_1 + h_i$ where $\mathrm{LT}(w_i) = \mathcal{S}(r_i)$ et $h_i = \sum_{r \in R,\ \mathcal{S}(r)=0} \mu_{r,i} \pi(r) \in I'$. Let f be an element of $I \backslash I'$. Consider the set of all representations of f:

$$\mathcal{V}(f) = \{\mathbf{s} = (s_1, \ldots, s_{n_R}) \in \mathbb{K}[x_1, \ldots, x_n]^{n_R} \mid \sum_{i=1}^{n_R} s_i g_i = f\}$$

In order to define a total ordering on $\mathcal{V}$ we introduce the notation:

$$\begin{array}{ll} & \mathcal{S}(\mathbf{s}) = \max_i \mathcal{S}(s_i r_i) \\ \text{for } \mathbf{s} \in \mathcal{V} & \mathrm{LT}(\mathbf{s}) = \max\{\mathrm{LT}(s_i g_i) \mid i \text{ such that } \mathcal{S}(s_i r_i) = \mathbf{s}\} \\ & \psi(\mathbf{s}) = (\mathcal{S}(\mathbf{s}), \#\{i \mid \mathcal{S}(s_i r_i) = \mathcal{S}(\mathbf{s})\}, \mathrm{LT}(\mathbf{s}), \#\{i \mid \mathrm{LT}(s_i g_i) = \mathrm{LT}(\mathbf{s})\}) \end{array}$$

We define $\mathbf{s} <_1 \mathbf{s}'$ iff $\psi(\mathbf{s}) <_{lex} \psi(\mathbf{s}')$. We take $\mathbf{s} = min_{<_1} \mathcal{V}$. Hence $f = \sum_{i=1}^{n_R} s_i g_i$, $w = \mathcal{S}(\mathbf{s})$, $t = \mathrm{LT}(\mathbf{s})$, $\mathcal{J} = \{i \mid \mathcal{S}(s_i r_i) = \mathcal{S}(\mathbf{s})\}$, $\mathcal{I} = \{i \in \mathcal{J} \mid \mathrm{LT}(s_i g_i) = \mathrm{LT}(\mathbf{s})\}$.

First we suppose, for a contradiction, that there exists $i \in \mathcal{J}$ such that (s_i, r_i) is not normalized (condition i). Saying that (s_i, r_i) is not normalized is equivalent to say that $s_i w_i$ is top reducible by $G' \subset \pi(R)$:

$$s_i w_i = u + \sum_{r \in R,\ \mathcal{S}(r)=0} \lambda_r \pi(r)$$

with $\mathrm{LT}(u) < \mathrm{LT}(s_i w_i)$ et $\mathrm{LT}(\lambda_r \pi(r)) \leq \mathrm{LT}(s_i w_i) = \mathrm{LT}(s_i)\mathcal{S}(r_i)$. Then

$$\begin{aligned} f &= \textstyle\sum_{j=1}^{n_R} s_j g_j = \sum_{j \neq i} s_j g_j + s_i w_i f_1 + s_i h_i \\ &= \textstyle\sum_{j \neq i} s_j g_j + u f_1 + \sum_{r \in R\ \mathcal{S}(r)=0} f_1 \lambda_r \pi(r) + s_i h_i \\ &= \textstyle\sum_{j \neq i} s_j g_j + u f_1 + \sum_{r \in R,\ \mathcal{S}(r)=0} (s_i \mu_{r,i} + f_1 \lambda_r)\pi(r) \end{aligned}$$

This expression is $<_1 \mathbf{s}$ et there is a contradiction. Therefore for all $i \in \mathcal{J}$ the (s_i, r_i) are normalized.
Suppose, for a contradiction, that $t > \mathrm{LT}(f)$: necessarily $r \geq 2$. If $\#\mathcal{J}$ would be > 1 (condition ii) we could write f as follow:

$$\begin{aligned} f &= (\textstyle\sum_{j \in \mathcal{J}} s_j w_j) f_1 + \sum_{j \notin \mathcal{J}} s_j g_j + \sum_{j \in \mathcal{J}} s_j h_j \\ &= (\textstyle\sum_{j \in \mathcal{J}} s_j w_j) f_1 + \sum_{j \notin \mathcal{J}} s_j g_j + \sum_{r \in R,\ \mathcal{S}(r)=0} \left(\sum_{j \in \mathcal{J}} \mu_{r,j} \right) \pi(r) \end{aligned}$$

so we could find a smaller expression of f. Consequently $\#\mathcal{J} = 1$ and let $k \in \mathcal{J}$ et $l \in \mathcal{I} \backslash \{k\}$. By construction we have $\mathcal{S}(s_l r_l) \prec \mathcal{S}(s_k r_k)$ (condition iii). We write f as follow:

$$f = s_k g_k - \frac{\mathrm{LC}(s_k)}{\mathrm{LC}(s_l)} s_l g_l + \left(1 + \frac{\mathrm{LC}(s_k)}{\mathrm{LC}(s_l)}\right) s_l g_l + \sum_{i \neq k,l} s_i g_i$$

Let $m_k = \mathrm{LM}(s_k)$ et $m_l = \mathrm{LC}(s_k)\ \mathrm{LT}(s_l)$ and $s'_i = s_i - \mathrm{LM}(s_i)$. Hence $t = \mathrm{LT}(m_k g_k) = \mathrm{LT}(m_l g_l)$, et consequently $lcm(g_k, g_l)$ divides t, that is to say:
$m_k g_k - m_l g_l = \mathrm{LC}(s_k) \frac{t}{lcm(g_k, g_l)}\ \mathrm{Spol}(g_k, g_l)$ Since (s_k, g_k) et (s_l, g_l) are normalized we deduce that (g_k, g_l) is normalized, so that there exists some $t' < lcm(g_k, g_l)$ such that:

$$\mathrm{Spol}(g_k, g_l) = \sum_{i=1}^{n_R} v_i g_i \text{ where } \mathrm{LT}(v_i g_i) \leq t' \text{ and } \frac{lcm(g_k, g_l)}{\mathrm{LT}(g_k)} \mathcal{S}(r_k) \geq \mathcal{S}(v_i r_i)$$

introducing new variables $v'_i = HC(s_k) \frac{t}{lcm(g_k, g_l)}\ v_i$ we have:

$$m_k g_k - m_l g_l = \sum_{i=1}^{n_R} v'_i g_i \text{ where } \mathrm{LT}(v'_i g_i) \leq \frac{t\, t'}{lcm(g_k, g_l)} < t \text{ and } \frac{t}{\mathrm{LT}(g_k)} \mathcal{S}(r_k) \geq \mathcal{S}(v'_i r_i)$$

Hence

$$f = \sum_{i=1}^{n_R} v'_i g_i + s'_k g_k + (\alpha s_l + \beta s'_l) g_l + \sum_{i \neq k,l} s_i g_i$$

where $\beta = \frac{\mathrm{LC}(s_k)}{\mathrm{LC}(s_l)}$, $\alpha = 1 + \beta \in \mathbb{K}$. Since $\mathrm{LT}(s'_i) < \mathrm{LT}(s_i)$ et $HT(s_k)\,\mathcal{S}(r_k) \geq \mathcal{S}(v'_i r_i)$ this is a new expression of f which is $<_1 \mathbf{s}$. This is a contradiction and we must have $t \leq \mathrm{LT}(f)$. So we can reduce f by an element of $\pi(R)$ et $\pi(R)$ is a Gröbner basis.

Proof. We may suppose that $f_1 \notin I'$ (if $f_1 \in I'$ the theorem is obvious since $\pi(R)$ contains a Gröbner basis of $I = I'$). By hypothesis $r_1 = (1, f_1) \in R$. We want to show that for all $f \in I$ there exists $r \in R$ such that $\mathrm{LT}(r)$ divides $\mathrm{LT}(f)$. We define:

$$\mathcal{V} = \{(w, h) \in \mathcal{R} \times I' \mid \forall r \in R \ \ \mathrm{LT}(r) \text{ ne divise pas } \mathrm{LT}(wf_1 + h)\}$$

We suppose, for a contradiction, that $\mathcal{V} \neq \emptyset$ et we take the minimum (lexicographical order) $(w, h) \in \mathcal{V}$. Let f be $wf_1 + h \in I$. We have to consider three cases:

- $\mathrm{LT}(wf_1) = \mathrm{LT}(f)$ then f is top reducible by the leading term of $r_1 = (1, f_1)$ so that $(w, h) \notin \mathcal{V}$.
- $\mathrm{LT}(wf_1) < \mathrm{LT}(f)$ then $\mathrm{LT}(f) = \mathrm{LT}(h) \in \mathrm{LT}(I')$ is top reducible by some element $g \in G'$. By hypothesis, there exists $r \in R$ such that $\pi(r) = \pi(g)$ so that $(w, h) \notin \mathcal{V}$.
- $\mathrm{LT}(wf_1) > \mathrm{LT}(f)$ so that $w \neq 0$; we first proof that w is not top reducible by G'. Suppose this is false: the normalForm w' of w w.r.t. G' is such that $w = w' + u_1 g_{j_1} + u_2 g_{j_2} + \cdots$ et $\mathrm{LT}(w') < \mathrm{LT}(w)$, et $\mathrm{LT}(w) = \mathrm{LT}(u_1 g_{j_1}) > \mathrm{LT}(u_2 g_{j_2}) > \cdots$

  $$f = wf_1 + h = w'f_1 + (h + u_1 f_1 g_{j_1} + u_2 f_2 g_{j_2} + \cdots) = w'f_1 + h'$$

  and so we can construct $(w', h') \in \mathcal{V}$ with $w' < w$. Contradiction. Hence w is not top reducible by G'. Next $f - wf_1 = h \xrightarrow{R^*} 0$ et we can write

  $$f - wf_1 = u_1 g_{j_1} + u_2 g_{j_2} + \cdots \text{ où } \mathrm{LT}(f - wf_1) = \mathrm{LT}(u_1 g_{j_1}) > \mathrm{LT}(u_2 g_{j_2}) > \cdots$$

  and by assumption $\mathrm{LT}(f - wf_1) = \mathrm{LT}(wf_1) = \mathrm{LT}(u_1 g_{j_1})$. There exists $r \in R$ such that $\pi(r) = g_{j_1}$. Let τ be $\mathrm{lcm}(\mathrm{LT}(f_1), \mathrm{LT}(r))$, obviously $\mathrm{LT}(f_1)$ et $\mathrm{LT}(r)$ divide $\mathrm{LT}(w\,f_1)$ so that τ divides $\mathrm{LT}(wf_1)$. Let $t = \frac{\mathrm{LT}(wf_1)}{\tau}$, $\alpha = \frac{\tau}{\mathrm{LT}(f_1)}$ and $\beta = \frac{\tau}{\mathrm{LT}(r)}$ et we have

  $$\mathrm{Spol}(f_1, \pi(r)) = \alpha f_1 - \beta r = \frac{1}{\mathrm{LC}(w)\,t}\,(\mathrm{LM}(w) f_1 + u_1 r)$$

  Since $\mathrm{LT}(w)$ is not top reducible by G' then $\alpha = \frac{\mathrm{LT}(w)}{t}$ is not top reducible by G'; since $\pi(r) \in G'$ we have $\mathcal{S}(r) = 0$ hence (r_1, r) is normalized so that $\mathrm{spol}(r_1, r) \xrightarrow{R^*} 0$:

$$\pi(\mathrm{Spol}(r_1, r)) = v_1\pi(r_{k_1}) + v_2\pi(r_{k_2}) + \cdots$$
$$where \begin{cases} \mathrm{LT}(\mathrm{Spol}(r_1, r)) = \mathrm{LT}(v_1 r_{k_1}) > \mathrm{LT}(v_2 r_{k_2}) > \cdots \\ \text{et } \mathcal{S}(v_i r_{k_i}) \leq \alpha \end{cases}$$

thus we have rewritten f as follow:

$$\begin{aligned} f &= \mathrm{LC}(w)\, t\, (v_1\pi(r_{k_1}) + v_2\pi(r_{k_2}) + \cdots + v_l\pi(r_{k_l})) + (w - \mathrm{LM}(w))f_1 + u_2 g_{j_2} + \cdots \\ &= v_1'\pi(r_{k_1}) + v_2'\pi(r_{k_2}) + \cdots + v_l'\pi(r_{k_l}) + (w - \mathrm{LM}(w))f_1 + u_2 g_{j_2} + \cdots \end{aligned}$$

where $\mathrm{LT}(wf_1) = t\tau > \mathrm{LT}(v_1' r_{k_1}) > \mathrm{LT}(v_2' r_{k_2}) > \cdots$ et $\mathcal{S}(v_i' r_{k_i}) < \mathrm{LT}(w)$ when $i < l$ et $\mathcal{S}(v_l' r_{k_l}) = \mathrm{LT}(w)$. Let r' be $v_l' r_{k_l}$. Since $f \in \mathcal{V}$, $\mathrm{LT}(f) \neq \mathrm{LT}(r')$; if $\mathrm{LT}(r') < \mathrm{LT}(f)$, then we found $f' = f - r' \in \mathcal{V}$ et f' is smaller. Thus if $\mathrm{LT}(r') > \mathrm{LT}(f)$.
By hypothesis none of the $\mathrm{LT}(v_i' r_{k_i})$ is equal to $\mathrm{LT}(f)$ hence

$$f' = f - v_1'\pi(r_{k_1}) - v_2'\pi(r_{k_2}) - \cdots = (w - \mathrm{LM}(w))f_1 + u_2 g_{j_2} + \cdots$$

Note 5.4.6. Dans le théorème si on restreint la condition (iii) aux paires critiques de degré $\leq d$ alors on obtient une $d-$base de Gröbner.

5.4.4 Réduction au sens de F_5

Lorsqu'on réduit un des polynômes on doit garder tous les polynômes admissibles et on doit modifier la définition usuelle de réduction; ainsi dans la fonction TopReduction(r, r') si r est réductible par r': il existe $t \in T$ tel que $\mathrm{LM}(r') \cdot t = \mathrm{LM}(r)$ et si r, r' sont unitaires et admissibles on peut trouver $(w, w') \in \mathbb{K}[x_1, \ldots, x_n]^2$ et $(h, h') \in I'^2$ tels que

$$\pi(r) = wf_1 + h,\ \pi(r') = w'f_1 + h' \text{ et } \mathcal{S}(r) = \mathrm{LT}(w),\ \mathcal{S}(r') = \mathrm{LT}(w').$$

Par conséquent

$$\pi(\tilde{r}) = \pi(r) - t\pi(r') = (w - tw')\, f_1 + (h - th')$$

et une condition suffisante pour que $\mathrm{LT}(w - tw') = \mathrm{LT}(w)$ est que $t\,\mathrm{LT}(w') < \mathrm{LT}(w)$ c'est à dire $t \cdot \mathcal{S}(r') < \mathcal{S}(r)$; dans ce cas $(\mathcal{S}(r), \pi(r) - t\pi(r'))$ est admissible.

Sinon, $t\,\mathrm{LT}(w') < \mathrm{LT}(w)$, et on doit construire un *nouveau polynôme* $(t\mathcal{S}(r'), \pi(r) - t\pi(r'))$ qui est admissible (si on pense l'algorithme en termes d'algèbre linéaire cela signifie qu'on doit échanger deux lignes dans la matric). Donc il est nécessaire de retourner plusieurs polynômes dans la fonction TopReduction et c'est donc une différence essentielle avec l'algorithme de Buchberger. De plus on doit s'assurer de ne pas générer deux polynômes avec la même signature: on maintient donc une liste de termes $\mathcal{A}$ contenant toutes les signatures déjà crées: un réducteur potentiel r' sera considéré seulement si $\mathcal{S}(r')$ n'est pas déjà dans $\mathcal{A}$.

Algorithme 32 TOPREDUCTION

Input: $r \in \mathcal{R},\ R \subset \mathcal{R},\ \mathcal{A} \subset T$
Output: (r'', S) *où* $\begin{cases} r'' = \frac{r}{\mathrm{LC}(r)} \text{ si } r \text{ est réductible et } \emptyset \text{ sinon.} \\ S \text{ est un ensemble de } 0, 1 \text{ ou } 2 \text{ polynômes} \end{cases}$
if $\pi(r) = 0$ ***then***
  return $(\emptyset, \emptyset)$ // reduction à zéro !
for $r' \in R$ ***do***
  if $(t = \frac{\mathrm{LT}(r')}{\mathrm{LT}(r)} \in T)$ *et* $((t, r')$ *est normalisé) et* $((t\mathcal{S}(r') \notin \mathcal{A})$ ***then***
    if $\mathcal{S}(tr') \prec \mathcal{S}(r)$ ***then***
      return $(\emptyset, \{(\mathcal{S}(r), \pi(r) - t\,\pi(r'))\})$
    else
      $r' = (t\,\mathcal{S}(r'), t\,\pi(r') - \pi(r)) \in \mathcal{R}$
      $\mathcal{A} = \mathcal{A} \cup \mathcal{S}(r')$
      return $(\emptyset, \{r', r\})$
return $(\frac{1}{\mathrm{LC}(r)}\, r, \emptyset)$

Lemma 5.4.1. *Si* $\{r\} \cup R$ *est constitué de polynômes admissibles alors tous les polynômes retournés par* TOPREDUCTION *sont admissbles.*

Proof. Ceci découle de la discussion au début de ce paragraphe.

On peut maintenant décrire la fonction REDUCTION qui est une fonction globale au sens de la réduction dans l'algorithme F_4.

Algorithme 33 REDUCTION

Input: F, R *listes de polynômes normalisés dans* $\mathcal{R}$
Output: *une liste de polynômes.*
$F =$ *éliminer dans* F *les doublons (pour la signature)*
$\mathcal{A} = \mathcal{S}(F)$, $R' = \emptyset$
while $F \neq \emptyset$ ***do***
  Prendre et retirer dans F *l'élément* r *avec la plus petit signature.*
  $(r', F') =$TOPREDUCTION$(r, R \cup R', \mathcal{A})$
  $R' = R' \cup \{r'\}$ *et* $F = F \cup F'$
return R'

5.4.5 Version simplifiée de l'algorithme F_5

On présente maintenant une version simplifiée de l'algorithme. On essaye d'être aussi proche que possible de la présentation de l'algorithme de Buchberger. Soit I' un idéal. On suppose qu'on a déjà calcule une base de Gröbner G' et on veut calculer la base de Gröbner de $(f_1) + I'$. On décrit maintenant l'algorithme utilisant le nouveau critère du théorème 5.4.1. L'implantation du test de normalisation et de calcul du S-polynôme est immédiat à partir

des définitions 5.4.5 et 5.4.6 (voir aussi la remarque 5.4.4). L'algorithme fait appel à la fonction REDUCTION (algorithme 33) qui retourne une liste de polynômes réduits (comme dans l'algorithme F_4) . On verra dans l'exemple 5.5 qu'on peut remplacer cette fonction par une fonction de réduction totale.

Algorithme 34 *(F_5 version de base)*

Input: *f_1 un polynôme et G' une base de Gröbner*
Output: *une base de Gröbner de $(f_1) + G'$*
$R = \{(0, g') \mid g' \in G'\} \cup \{(1, f_1)\}$
$P = [(r, r') \mid r' \in G'$ *et* (r, r') *est normalisée*$]$
while $P \neq \emptyset$ ***do***
  Soit d_0 le degré minimal de P
  $P_{d_0} = \{p \in P \mid \deg(p) = d_0\}$
  $R_{d_0} =$REDUCTION$(\mathrm{Spol}(P_{d_0}), R)$
  for $r \in R_{d_0}$ ***do***
    $P = P \cup [(r, r') \mid r' \in R$ *et* (r, r') *normalisée*$]$
    $R = R \cup \{r\}$
return R

Note 5.4.7. Pour un polynôme homogène f_1, une variante de l'algorithme est de ne garder dans la liste des paires critiques P que les paires de degré $\leq d$; dans ce cas on obtient une $d-$base de Gröbner. On note $d-$Gröbner F_5 cette variante de l'algorithme F_5.

5.4.6 Preuve de l'algorithme F_5

On suppose que le polynôme f_1 est homogène. Soit $\tilde{R}$ l'ensemble de tous les polynômes qui sont générés pendant l'algorithme. On donne la preuve de l'algorithme.

Proposition 5.4.3. *Pour tout $r \in \tilde{R}$, r est admissible et normalisé.*

Proof. Au départ $(1, f_1)$ est admissible. Les nouveaux polynômes sont crées dans:

(i) dans le calcul du S-polynôme: $\mathrm{Spol}(P_d)$ et par définition la paire est normalisée.
(ii) dans TOPREDUCTION : on utilise alors le lemme 5.4.1.

Theorem 5.4.2. *Pour tout d, le résultat de $d-$Gröbner F_5 est une base de Gröbner jusqu'au degré d.*

Proof. Pour appliquer le théorème 5.4.1 on considère une paire critique normalisée $(r, r') \in R$ et d son degré. Soit r'' le résultat de la réduction de $\mathrm{Spol}(r, r')$ par R_{d-1} et $\tau = \mathrm{lcm}(\mathrm{LT}(r), \mathrm{LT}(r'))$.En notant $u = \frac{\tau}{\mathrm{LT}(r)}$ on a:

$$\mathcal{S}(r'') = u\,\mathcal{S}(r)$$
$$\text{et } \mathrm{LT}(\pi(r'')) < \mathrm{lcm}(\mathrm{LT}(r), \mathrm{LT}(r')) = u\,\mathrm{LT}(r)$$

et donc

$$\mathrm{Spol}(r, r') \xrightarrow{R_{d-1}} r'' \xrightarrow{R_{d-1}\cup\{r\}} 0$$

Comme $r'' \in R_d$ on en déduit $\mathrm{Spol}(r, r') \xrightarrow{R_d} 0$. D'après le lemme 5.4.3 on peut appliquer le théorème 5.4.1 et on en déduit que R_d est une base de Gröbner de l'idéal engendré par $(f_1) + I$ jusqu'au degré d.

Theorem 5.4.3. *Si on suppose que le polynôme f_1 est homogène et qu'il n'y a pas de réduction à zéro pendant le calcul. On note R_d le résultat de* Reduction *dans l'algorithmeF_5 34 et R la valeur de R dans l'algorithme avant* Reduction. *Alors $Id(\mathrm{LT}(R)) \neq Id(\mathrm{LT}(R \cup R_d))$.*

Corollary 5.4.1. *L'algorithme F_5 se termine.*

Proof. Soit u le maximum des signatures des élements de R_d: $u = \max_<\{\mathcal{S}(r) \mid r \in R_d\}$; il existe donc $r \in R_d$ tel que $\mathcal{S}(r) = u$. Par l'absurde, supposons qu'il existe $r' \in G_1 \cup R_d \backslash \{r\}$ tel que $u = \frac{\mathrm{LT}(r)}{\mathrm{LT}(r')} \in T$. On doit distinguer deux cas:

1. Si $u\,\mathcal{S}(r')$ n'est pas top réductible par G' alors
   a) si $u\,\mathcal{S}(r') > \mathcal{S}(r)$ alors la paire critique $(r', r) = (u, r', 1, r)$ a été introduite dans la liste P et comme il n'y a pas de réduction à zéro alors $u\,r' \in R_d$. C'est une contradiction car r avait la signature maximale.
   b) si $u\,\mathcal{S}(r') < \mathcal{S}(r)$ alors r peut être réduite par r'. Contradiction.
2. Si $u\,\mathcal{S}(r')$ est top réductible par G'. Alors comme r' est admissible $\pi(r') = s'_1\,f_1 + f'$ avec $f' \in \mathrm{Id}(G')$ et $\mathrm{LT}(s'_1) = \mathcal{S}(r')$ et $u\,s'_1 = v + v'$ où v est totalement réduit par G_2 et $\mathrm{LT}(v) < \mathrm{LT}(u\,s'_1)$. On a donc:

$$\begin{aligned} u\pi(r') &= us'_1 f_1 + u\,f' \\ &= v f_1 + u\,f' + f_1\,v' \end{aligned}$$

   Soit $\mathcal{T} = \{\mathrm{LT}(t\,f_1) > \mathrm{LT}(r) \mid t \in T(v)\}$. Si $\mathcal{T}$ n'est pas vide alors pour tout $t \in \mathcal{T}$, (t, f_1) est normalisé et donc a été placé dans la liste des paires critiques. Au cours du preocessus ce réduction on trouve un polynôme r'' tel que $\mathcal{S}(r'') = \mathrm{LT}(v)$ et $\mathrm{LT}(r'') = \mathrm{LT}(r)$. Contradiction. Si $\mathcal{T} = \emptyset$, alors $u\,f' + f_1\,v' = \sum_{g\in G'} \mu_g\,g$ où $\mathrm{LT}(\mu_g\,g) \leq \mathrm{LT}(r)$. Ainsi $\mathrm{LT}(r) = \mathrm{LT}(\mu_g g)$ pour un certain $g \in G_2$ ou $\mathrm{LT}(r) = \mathrm{LT}(v\,f_1)$. Donc on peut réduire r par $G' \cup \{f_1\}$.

Definition 5.4.8. *Let PSyz be the module generated by the principal syzygies. For a generic (random) polynomial system $(f_1, \ldots, f_m)$, $Syz = PSyz$. We can extend the admissible ordering $<$ to E^m with the following definition:*

$$\sum_{k=i}^{m} g_k F_k \prec \sum_{k=j}^{m} h_k F_k \text{ iff} \begin{cases} i > j \text{ and } h_j \neq 0 \\ \text{or} \\ i = j \text{ and } \mathrm{LT}(g_i) < \mathrm{LT}(h_i) \end{cases}$$

Theorem 5.4.4. *If the algorithm finds a reduction to zero, $r_{i_k} \to 0$ then there exists $\mathbf{s} \in Syz \backslash PSyz$ with $\mathrm{LT}(\mathbf{s}) = \mathcal{S}(r_{i_k})$.*

Proof. We may suppose wlog that $\mathcal{S}(r_{i_k}) = t\mathbf{F}_1$ for some $t \in T$. Now for all $\mathbf{s} \in \mathrm{PSyz}$ with $\mathrm{index}(\mathbf{s}) = 1$ we have
$\mathbf{s} = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \lambda_{i,j} \mathbf{s}_{i,j}$
$= \sum_{i=1}^{m} \sum_{j=i+1}^{m} \lambda_{i,j} f_j \mathbf{F}_i - \sum_{i=1}^{m} \sum_{j=i+1}^{m} \lambda_{i,j} f_i \mathbf{F}_j$ Consequently $\mathrm{LT}(\mathbf{s}) =$
$= \sum_{j=2}^{m} \lambda_{1,j} f_j \mathbf{F}_1 + \sum_{i=2}^{m} (\cdots) \mathbf{F}_i$
$\mathrm{LT}(\sum_{j=2}^{m} \lambda_{1,j} f_j) \mathbf{F}_1$, that is to say $\mathrm{LT}(\mathbf{s}) \in \mathrm{LT}(Id(f_2, \ldots, f_m))$. Hence if $r_{i_k} = 0$, then $\mathcal{S}(r_{i_k}) = \mathrm{LT}(\mathbf{s})$ for some $s \in Syz$. Since r_{i_k} is normalized $\mathrm{LT}(s) \notin \mathrm{LT}(Id(f_2, \ldots, f_m))$, hence $\mathbf{s} \notin PSyz$.

Corollary 5.4.2. *If the input system is a regular sequence there is no reduction to zero.*

5.5 F_5: exemple pas à pas

On calcule maintenant la base de Gröbner pour un exemple tiré de l'article (Mora, T. and Möller, H.M. and Traverso, C., 1992). L'ordre utilisé est l'ordre DRL tel que $x > y > z > t$ et les coefficients sont les nombres rationnels ($\mathbb{Q}$):

$$\begin{cases} f_1 = yz^3 - x^2t^2 \\ f_2 = xz^2 - y^2t \\ f_3 = x^2y - z^2t \end{cases}$$

L'algorithme F_5 calcule successivement les bases de Gröbner des idéaux $\mathrm{Id}(f_3)$, $\mathrm{Id}(f_2, f_3)$ et $\mathrm{Id}(f_1, f_2, f_3) = \langle f_1 \rangle + \mathrm{Id}(f_2, f_3)$. Comme le dernier calcul est le plus significatif on décrit uniquement ce dernier cas. Les polynômes de la base de Gröbner $G' = \pi(R')$ de $\mathrm{Id}(f_2, f_3)$:

$R' = [r_3, r_2, r_4, r_5]$ où $r_3 = (0, f_3)$, $r_2 = (0, f_2)$, $r_4 = (0, x\,y^3\,t - z^4\,t)$, $r_5 = (0, z^6\,t - y^5\,t^2)$.

On note $\varphi' = \mathrm{NF}(., [r_3, r_2, r_4, r_5])$ la forme normale par rapport à $G\prime$.
$r_1 = (1, f_1)$
$R = R' \cup \{r_1\} = [r_3, r_2, r_4, r_5, r_1]$
Il y a quatre paires critiques: $p_7 = (x\,y\,z^3, x, r_1, y\,z, r_2)$, $p_8 = (x^2\,y\,z^3, x^2, r_1, z^3, r_3)$, $p_9 = (y\,z^6\,t, z^3\,t, r_1, y, r_5)$, $p_{10} = (x\,y^3\,z^3\,t, x\,y^2\,t, r_1, z^3, r_4)$.

Les signatures de $\mathcal{S}(p_7), \ldots, \mathcal{S}(p_{10})$ sont resp. x, x^2, z^3, $x\,y^2$ sont toutes invariants par φ' donc les paires sont normalisées.
$P = [p_7, p_8, p_9, p_{10}]$
$\boxed{d_0 = 5}$, on calcule $\mathrm{Spol}(P_5)$ avec $P_5 = [p_7]$ et $P = [p_8, p_9, p_{10}]$
$r_6 = (x,\ y^3\,z\,t - x^3\,t^2)$ et $F := [r_6]$
On ajoute la règle de récriture $x \to r_6$
Il n'y a aucune réduction possible de r_6 par G' donc le résultat retourné est $R_5 = [r_6]$

$R = [r_3, r_2, r_4, r_5, r_1, r_6]$
On met à jour la liste de paires critiques: $p_{11} = (y^3\,z^3\,t, z^2, r_6, y^2\,t,\ r_1)$, $p_{12} = (y^3\,z^6\,t, z^5, r_6, y^3, r_5)$, $p_{13} = (x\,y^3\,z\,t, x, r_6, z\,t, r_4)$, $p_{14} = (x^2\,y^3\,z\,t, x^2, r_6, y^2\,z\,t, r_3)$, $p_{15} = (x\,y^3\,z^2\,t, x\,z, r_6, y^3\,t\ , r_2)$. On vérifie que $\mathcal{S}(z^2\,r_6) = xz^2$ et $\mathcal{S}(z^5\,r_6) = x\,z^5$ sont réductible par φ' donc les paires p_{11} et p_{12} sont *éliminées* par le critère de F_5. Donc $P = [p_8, p_9, p_{10}, p_{13}, p_{14}, p_{15}]$.

$\boxed{d = 6}$, on calcule Spol(P_6) avec $P_6 = [p_8, p_{13}]$ et $P = [p_9, p_{10}, p_{14}, p_{15}]$
Comme on a la règle $x^2\,r_1 \longrightarrow x\,r_6$ on élimine la paire critique p_8.
En revanche on garde l'autre paire p_{13} puisque Rewritten?$(x, r_6) = false$ et Rewritten?$(z, r_4) = false$; par suite $r_7 = (x^2, z^5\,t - x^4\,t^2)$.
On ajoute la règle $x^2 \longrightarrow r_7$ et il n'y a pas de réduction possible de r_7 par R donc on retourne $R_6 = [r_7]$ et mainenant $R = [r_3, r_2, r_4, r_5, r_1, r_6, r_7]$.
Parmi toutes les paires critiques possibles on vérifie que (r_7, r_1), (r_7, r_6), (r_7, r_3) et (r_7, r_4) sont *éliminées* par le critère de F_5.
Les nouvelles paires normaliśes sont $p_{16} = (z^6\,t, z, r_7, 1, r_5)$ et $p_{17} = (x\,z^5\,t, x, r_7, z^3\,t, r_2)$.

$\boxed{d = 7}$, on calcule Spol(P_7) avec $P_7 = [p_{15}, p_{16}, p_{17}, p_{14}]$ et $P = [p_9, p_{10}]$.
Comme on a la règle $x\,z\,r_6 \longrightarrow z\,r_7$ on élimine la paire critique p_{15}.
p_{16} est normalisée et on calcule $r_8 = (x^2\,z\,, y^5\,t^2 - x^4\,z\,t^2)$ puis on ajoute la règle $x^2\,z \to r_8$.
p_{17} est normalisée et on calcule $r_9 = (x^3\,\mathbf{F}_1, -x^5\,t^2 + y^2\,z^3\,t^2)$ puis on ajoute la règle $x^3 \to r_9$.
Comme on a la règle $x^2\,r_6 \longrightarrow r_9$ on ne garde pas p_{14}. Il y a donc deux paires à traiter: $F = [r_8, r_9]$
Les éléments de F ne sont pas top reductible par R (dans la fonction REDUCTION) selon la description de l'algorithme 34 mais il est possible de *réduire totalement* r_9 par $y\,t^2 \times r_1$: alors $r_9 = (x^3\,, -x^5\,t^2 + x^2\,y\,t^4)$ et le résultat final est $r_9 = -\varphi'(r_9) = (x^3, x^5\,t^2 - z^2\,t^5)$
Le résultat de REDUCTION est donc: $R_7 = [r_9, r_8]$. Maintenant $R = [r_3, r_2, r_4, r_5, r_1, r_6, r_7$
The critical pairs (r_9, r_1), (r_9, r_6), (r_9, r_7), (r_9, r_2), (r_9, r_3), (r_9, r_4), (r_9, r_5), (r_8, r_1), (r_8, r_6), (r_8, r_7), (r_8, r_9), (r_8, r_2) and (r_8, r_5) are not valid.
Les paires critiques sont $p_{18} = (x\,y^5\,t^2, x, r_8, y^2\,t, r_4)$ et $p_{19} = (x^2\,y^5\,t^2, x^2, r_8, y^4\,t^2, r_3)$.

$\boxed{d = 8}$, on calcule Spol(P_8) avec $P_8 = [p_9, p_{10}, p_{18}]$ et $P = [p_{19}]$.
p_9 est normalisée et $r_{10} = (z^3\,t\,, y^6\,t^2 - x^2\,z^3\,t^3)$ est calculée: on ajoute la règle $z^3\,t \to r_{10}$.
Comme on a la règle $x\,y^2\,t, r_1 \longrightarrow y^2\,t\,r_6$ on ne garde pas p_{10}
Comme on a la règle $x, r_8 \longrightarrow z\,r_9$ on ne garde pas p_{18}
Maintenant $r_{10} = \varphi'(r_{10}) = (z^3\,t\,, y^6\,t^2 - x\,y^2\,z\,t^4)$ est totalement réduit et le résultat du calcul est $R_8 = [r_{10}]$, $R = [r_3, r_2, r_4, r_5, r_1, r_6, r_7, r_8, r_9, r_{10}]$.
Toutes les nouvelles paires de la forme (r_{10}, r_i) avec $i = 1, \ldots, 8$ sont *éliminées* par le critère de F_5.

$\boxed{d = 9}$, on calcule Spol(P_9) avec $P_9 = [p_{19}]$ et $P = \emptyset$.
Comme on a la règle $x^2\,r_8 \longrightarrow x\,z\,r_9$ on ne garde pas la paire p_{19} et il n'y aucun calcul à effectuer.
$F = \emptyset$ et $R_9 = \emptyset$

L'algorithme se termine et retourne $G = \pi(R)$.

On remarque que le calcul n'a généré aucune paire critique inutile. Avec l'algorithme de Buchberger il y a 7 paires inutiles et 5 utiles.

5.6 Résultats expérimentaux

5.6.1 Nombre de paires inutiles

This is interesting to compare the number of useless critical pairs in practice for the various algorithms because this number does not depend on the implementation (at least for the F_5 algorithm). The first line of the following tabular (figure 5.6.1) contains all the examples of (Gebauer & Moller, 1986) et (Mora, T. and Möller, H.M. and Traverso, C., 1992) the other are well known. Note that reductions to zero are unavoidable for `Trinks7` (7 equations, 6 variables). The table brought the `Eco n` to our attention since the number of useless pairs is not zero: we found that the system can be straightforwardly rewritten by factorizing the original equations. By reformulating these problem we obtain an equivalent system `Eco n fact` with no reduction to zero ! The conclusion is that for a lot of practical examples there is no reduction to zero.

Example	(Buchberger B., 1985)	(Gebauer & Moller, 1986)	(Mora, T. and Möller, H.
Raksanyi	1	?	
Hairer1	10	?	
Rose	22	19	
Trinks6	17	8	
Trinks7	12	11	
Katsura3	1	?	
Katsura4	18	10	
Katsura5	50	28	
Katsura10	3936	?	
Binary10	2147	?	
Noon8	7886	?	
Eco 6	61	?	
Eco 6 fact	63	?	
Eco 8 fact	315	?	

Table 5.1. Nombre de paires critiques inutiles

5.6.2 Première implantation

A first implementation of the F_5 has been made in the Maple computer algebra system et then translated in Gb (C++) and FGb (C). From a traditional

implementation of the Buchberger algorithm it is very easy to implement the new algorithm: the only data type to modify is to add to the property list of each polynomial r an integer (the index k of r) et a power product t ($\mathcal{S}(r) = r\mathbf{F}_k$). Hence the extra memory cost is very small. The behavior of the algorithm is very good: it is at least one order of magnitude faster than the fastest known algorithm/implementation (F_4) et two order of magnitude faster than one of the fastest programs (Singular 2.0 (Greuel G.-M. and Pfister G. and Schoenemann H., 2002)). In tabular 5.6.2 we give the timings for the well known cyclic n problem: a Gröbner basis of Cyclic 10 was computed for the first time.

Cyclic	7	8	9	10
F_4	1.26	36.0	4949.1	
F_4 inc	1.4	171.3		
F_5	1.0	27.9		
F_5'	0.4	7.2	1002.3	57600
F_5''	0.8	3.95	676.2	

Table 5.2. (2002): Comparison of F_4 et F_5 for the Cyclic n problem modulo p (Inspiron PIII 1Ghz): CPU Time in seconds.

In table 5.6.2 "F_4 inc" is the F_4 algorithm applied incrementally. "F_5'" et "F_5''" are different version of the F_5 algorithm that will be described in a future paper. We report now detailed CPU timings for the Katsura n problem modulo a small prime p (there is no useless pairs for this example).

The algorithm F_5 is not always faster than F_4: for cyclic n the basic version of the F_5 algorithm is just a little faster than F_4; the maximal efficiency of the F_5 algorithm is expected when the number of equation is less or equal than the number of variables. On the contrary bad performance is expected when the system is overconstrained: for instance compute a Gröbner basis for a total degree et then rerun the F_5 algorithm on the result.

n	Singular	Gb	F_4	F_5	Singular
	2-0-0				1-2-3
7	1.6	2.2	0.4	**0.15**	3.1
8	13.6	22.25	2.8	**0.8**	36.4
9	135.3	252.5	23.1	**4.1**	411.2
10	1140.2	2907.1	220.2	**25.5**	4311.8
11	11671	34903	2097	**174.2**	58174.6
12			25161	**1460.7**	
13			240667	**10748**	

Table 5.3. Katsura n PII 400 Mhz (CPU time in seconds)

In the following tables 5.6.2 we compute the speedup: for instance `O Sing/Gb` is the CPU time for the old version of Singular (1-2-3) divided by the CPU of Gb on the same example.

n	F_4/F_5	Sing/Gb	Gb/F_4	O Sing/Gb	**Sing**/F_5
7	2.7	0.7	5.2	1.4	**10.6**
8	3.3	0.6	8.0	1.6	**16.4**
9	5.6	0.5	10.9	1.6	**33.1**
10	8.6	0.4	13.2	1.5	**44.8**
11	12.0	0.3	16.6	1.7	**67.0**
12	17.2				
13	22.4				

Table 5.4. Katsura n PII 400 Mhz (Speedup)

5.6.3 Benchmarks

	F4/FGb	F4/Mag	SlimGb	Buch/Sin	F5	Buch/Gb	Trebuchet	Lex Magma	SlimGb lex	FGLM FGb	INT Involutive	INT Lex Magma	INT RR	Speedup INT
6	0,01		0,23	0,06		0,26	0,03		0,21			7,88	0,44	33
7	0,04		3,58	0,41	0,02	2,42	0,12	0,5	> 650	0,03	2,15	308,15	2,4	80
8	0,3	0,2	36,71	3,26	0,09	20,07	0,65	1,409		0,07	27,48	12928,55	10,99	157
9	1,82	1,13	409,85	24,39	0,44	186,9	4,15	7,929		0,42	337,52		156,79	373,3
10	14,64	8,25		199,17	2,2		28,7	58,59		2,99	4790,55		2424	810,7
11	106,55	52,38			14,31		218,69	437,65		21,76			38785	1782,4
12	881,24	416,24			99,5		2017,23	3482,329		160,78			112,19	2512
13	6319,71	3037,86			616,2		17550,7			1180,5			1328,1	4050
14					5943,3					9141,5				
15					39657									

Fig. 5.1. Comparaison experimentale de l'algorithme F_5 sur l'exemple Katsura.

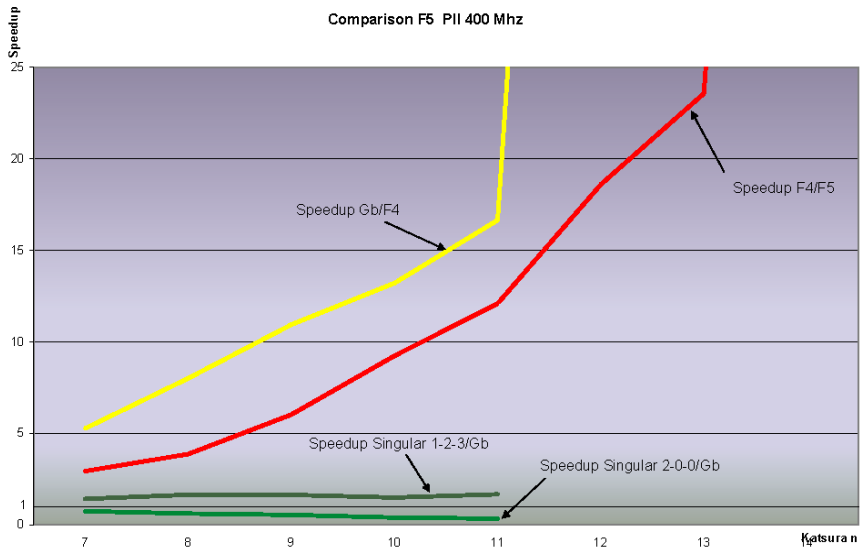
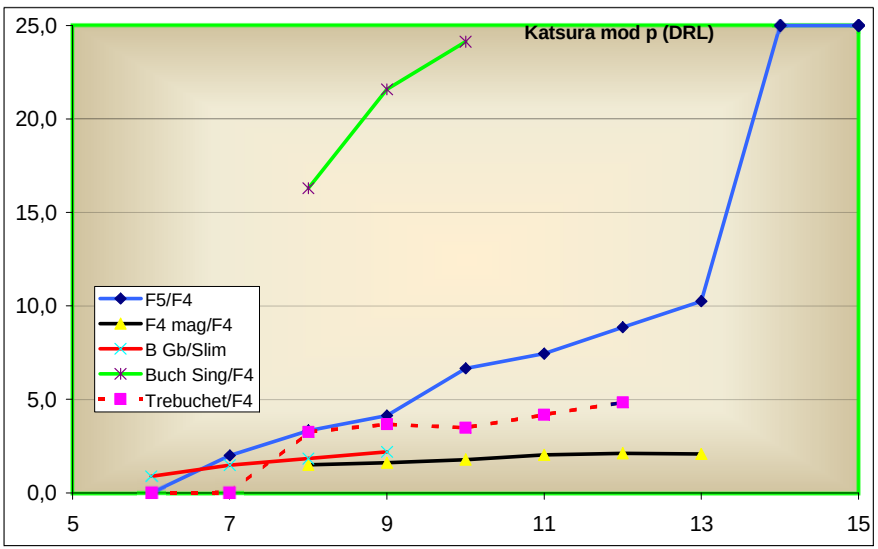


Fig. 5.2. Comparaison experimentale de l'algorithme F_5 sur l'exemple Katsura sous forme de speedup.



Comparaison experimentale sur l'exemple Katsura sous forme de speedup entre l'algorithme F_5 et plusieurs autres méthodes.

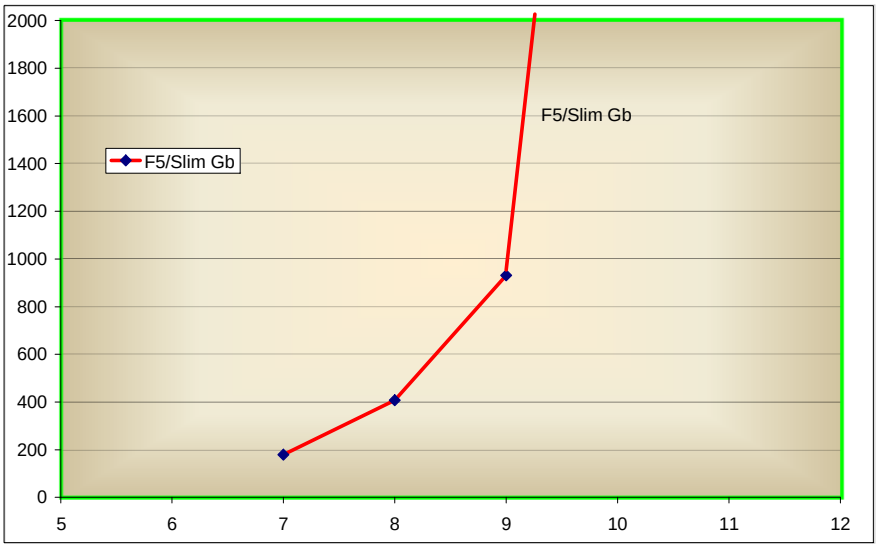


Fig. 5.3. Comparaison experimentale de l'algorithme F_5 sur l'exemple Katsura avec l'algorithme SlimGb.

6. Complexité. Systèmes sur-déterminés

Ce chapitre résume les travaux en collaboration avec M. Bardet et B. Salvy (voir la thèse de Magali Bardet (Bardet, 2004) et les articles (Bardet *et al.* , 2003a; Bardet *et al.* , 2005; Bardet *et al.* , 2004)).

6.1 Introduction et motivation.

Les études de complexité des bases de Gröbner sont nombreuses: par exemple, le pire cas est bien connu depuis l'exemple de Mayr-Meyer et on sait que dans ce cas la complexité du calcul est doublement exponentiel. Heureusement, en pratique le comportement des bases de Gröbner n'est jamais aussi catastrophique. Le but de chapitre est donner des résultats de complexité non pas dans le pire cas mais dans le cas "générique" (on va donner une définition précise de semi-régularité pour expliciter la notion de générique).

On trouve des systèmes surdeterminés dans bon nombre d'applications: par exemple dans les codes correcteurs (décodage des codes cycliques voir le chap. 13), robotique (chap. **??**), conception de filtres (chap **??**) et la cryptographie. La sécurité de beaucoup de primitives cryptographiques repose sur la difficulté de la résolution des systèmes algébriques. Dans le contexte de la *cryptographie publique multivariée* (voir les chapitres **??**, **??**, 10) les clés publiques sont directement données sous forme de polynômes en plusieurs variables. Plus généralement on peut ramener tout cryptosystème à un système algébrique et en évaluer la robustesse par le biais de l'analyse de complexité du système algébrique: cette demarche porte le nom de Cryptanalyse Algébrique. Dans la plupart des applications dans les corps finis on recherche les solutions non pas dans une extension algébrique du corps de base mais dans le corps lui même: ainsi sur $\mathbb{F}_2$ si on cherche à résoudre un système algébrique $[f_1, \ldots, f_m]$ ayant $m \geq n$ équations et n variables on doit lui adjoindre des équations de corps $x_i^2 - x_i = x_i\,(x_i - 1) = 0$ (pour $i = 1, \ldots, n$). On a donc en fait un système avec $m + n \geq 2\,n$ équations, donc un système surdéterminé.

Dans le contexte de la cryptographie publique multivariée il est très important de pouvoir distinguer un système "aléatoire" avec un système algébrique provenant d'une instance particulière: on verra que par exemple la clé publique de HFE (chap. **??**) semble constitué de polynômes aléatoires

(par exemple en examinant la densité des équations); pourtant le calcul de la base de Gröbner est beaucoup plus facile que pour un système générique (voir (Dubois *et al.* , 2006) pour un autre type de distingeur). Un moyen simple est de reporter sur un graphique le paramètre clé permettant l'analyse de complexité, à savoir le degré maximal atteint par le calcul de la base de Gröbner; c'est ce qui a été réalisé sur le dessin de la figure 6.1: la courbe rouge représente le degré maximal observé expérimentallement pour le calcul d'une base de Gröbner d'un système aléatoire de n équations quadratiques (denses) en n variables sur le corps $\mathbb{F}_2$ (par conséquent il faut ajouter encore n équations de corps de degré 2); les autres courbes proviennent du système HFE de taille n (et le paramètre d est le degré du polynôme secret voir le chapitre 8):

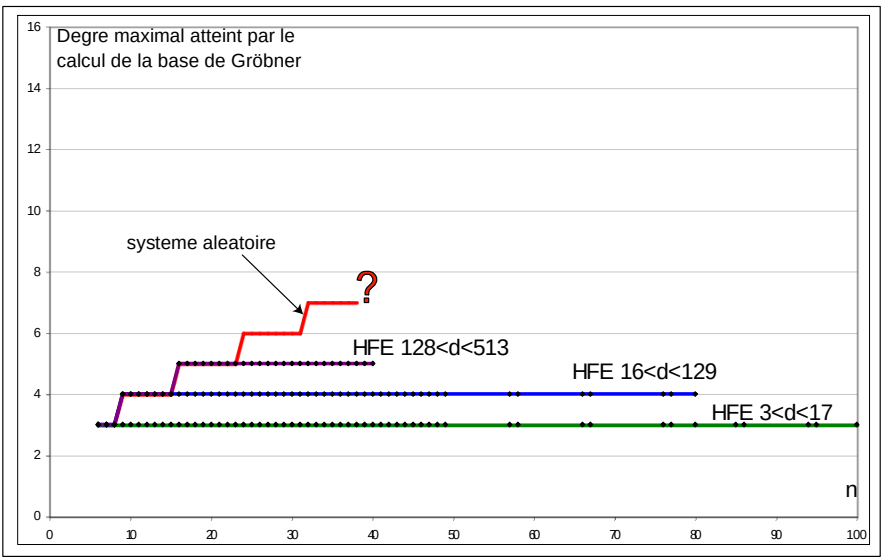


Fig. 6.1. Complexité: degré maximal atteint.

Cependant, on remarque que, très vite, les calculs sont trop difficiles pour les systèmes aléatoires et il n'est pas possible d'obtenir les valeurs pour la courbe rouge même pour des valeurs petites ($n = 40$). Lorsqu'on aura cette courbe il sera facile de distinguer un système aléatoire d'un système particulier: on peut stopper le calcul de la base de Gröbner dès qu'on observe une chute de degré strictement inférieure à la valeur prédite par la courbe théorique (en rouge). De plus on voudrait déterminer la pente de la courbe rouge. Le présent chapitre répond à toutes ces questions: par exemple la pente est de $\frac{1}{11.14}$ (voir le tableau 6.5.2) et toutes les points de la courbe rouge peuvent être calculés (voir la figure 6.5).

Pendant longtemps les cryptographes n'ont pas utilisé les bases de Gröbner car les performances étaients désastreuses dans les systèmes de Calcul Formel généralistes. L'algorithme utilisé était l'algorithme XL (Courtois *et al.* , 2000). La complixité de l'algorithme n'est pas l'objet du présent chapitre, mais comme XL peut être simulé par un calcul de base de Gröbner (Ars *et al.* , 2004; G., 2005) les bornes pour les bases de Gröbner sont aussi vraies pour XL.

Dans la suite la constante $\omega < 2.39$ désigne l'exposant de la complexité des matrices de multiplication.

6.2 Suites régulières et semi-régulières

D'après la définition géométrique de suite régulière (définition 2.8.3), dès que $m > n$, f_m devient un diviseur de zéro. Ainsi, si on calcule une base de Grobner de l'idéal en utilisant une stratégie Normale (algorithme 11), à partir d'un certain degré on aura des des réductions à zéro. Pour étendre la notion de suites régulières à des suites surdéterminées, il est donc nécessaire de modifier la définition de suite régulière: ainsi la définition 6.2.3 impose que f_i ne soit pas un diviseur de zéro lorsque $\deg(f_i)$ est inférieur au degré de régularité de l'idéal (définition 6.2.2).

6.2.1 Généricité

Les suites régulières représentent bien le comportement d'une suite dont les coefficients ont été tirés au hasard. Plus précisément, on peut montrer que, lorsque le corps des coefficients est infini, "presque toute" suite est une suite régulière, où plus rigoureusement que "être une suite régulière1' est une propriété générique au sens de la définition suivante :

Definition 6.2.1. *((Bardet, 2004)) Considérons l'ensemble $E(n, m, d_1, ..., d_m)$ des suites $f_1, \ldots, f_m$ de m polynômes de $\mathbb{K}[x_1, \ldots, x_n]$ en n variables, de degrés $d_1, ..., d_m$. Lorsque le corps $\mathbb{K}$ est infini, on dit qu'une propriété des suites est générique si l'ensemble des suites vérifiant cette propriété est un ouvert non vide de Zariski, c'est à dire si elle est vérifiée par toutes les suites de E, sauf un ensemble algébrique de codimension au moins un.*

Montrer que la propriété de semi-régularité est générique est une conjecture difficile de Fröberg (Fröberg, 1985) démontrée seulement dans le cas $m = n + 1$ (la difficulté est de montrer que c'est un ensemble non vide).

6.2.2 Suites régulières et semi-régulières. Degré de régularité.

Afin d'unifier le cas général et le cas particulier $\mathbb{F}_2$ on utilise la même notation que dans le chapitre 5: $\delta_{\mathbb{K},\mathbb{F}_2}$ est le symbole de Kornecker qui est égal à 1 si $\mathbb{K} = \mathbb{F}_2$ et 0 sinon.

Si $\mathbb{K}$ est le corps $\mathbb{F}_2$ et si on cherche les solutions du système algébrique $(f_1, \ldots, f_m)$ il faut rajouter à l'idéal I engendré par $(f_1, \ldots, f_m)$ les équations de corps $x_i^2 - x_i$. Il faut alors remarquer que dans l'anneau quotient $\mathbb{F}_2[\overline{x_1}, \ldots, \overline{x_n}] = \mathbb{F}_2[x_1, \ldots, x_n]/\langle x_1^2 - x_1, \ldots, x_n^2 - x_n\rangle$, tout polynôme f de l'idéal $\langle f_1, \ldots, f_m\rangle$ est solution de l'équation triviale

$$f^2 = f$$

(dit autrement f est laissé invariant par le morphisme Frobenius $p \to p^2$).

On note $R_{\mathbb{K}}$ l'anneau de polynômes

$$R_{\mathbb{K}} = \mathbb{K}[x_1, \ldots, x_n] \text{ si } \mathbb{K} \neq \mathbb{F}_2$$

et

$$R_{\mathbb{F}_2} = \mathbb{F}_2[x_1, \ldots, x_n]/\langle x_1^2, \ldots, x_n^2\rangle \text{ si } \mathbb{K} = \mathbb{F}_2$$

(il s'agit des polynômes *sans carré).* Ainsi si $M_d(n)$ désigne le nombre de termes en n variables en degré d dans $R_{\mathbb{K}}$ il est facile de prouver que $M_d(n) = \binom{n+d-1}{d}$ et $M_d(n) = \binom{n}{d}$ si $\mathbb{K} = \mathbb{F}_2$. Par conséquent:

$$\sum_{d=0}^{\infty} M_d(n) z^d = \left(\frac{1 - \delta_{\mathbb{K},\mathbb{F}_2}\, z^2}{1 - z}\right)^n \tag{6.1}$$

Comme il n'existe pas de suite régulière lorsque le nombre de polynômes est plus grand que le nombre de variables, on doit modifier la définition usuelle de régularité en limitant le degré des non diviseurs de zéro (Bardet, 2004; Bardet *et al.* , 2004):

Definition 6.2.2. *Le degré de régularité d'un idéal homogène $I = \langle f_1, \ldots, f_m\rangle$ dans l'anneau $R_{\mathbb{K}}$ est*

$$d_{reg} = \min\{d \geq 0 \mid \dim_{\mathbb{K}}(\{f \in I, \quad \deg(f) = d\}) = M_d(n)\}$$

Cette définition implique en particulier qu'un calcul de base de Gröbner pour un ordre du degré ne dépasse jamais le degré d_{reg}.

Definition 6.2.3. *Une suite de polynômes homogènes $(f_1, \ldots, f_m)$ est semi-régulière si pour tout $i = 1, \ldots, m$ et g tel que*

$$g \cdot f_i \in \langle f_1, \ldots, f_{i-1}\rangle \text{ et } \deg(g \cdot f_i) < d_{reg}$$

alors g est aussi dans $\langle f_1, \ldots, f_{i-1+\delta_{\mathbb{K},\mathbb{F}_2}}\rangle$.

Une suite de polynômes affines $(f_1, \ldots, f_m)$ est semi-régulière si $(f_1^H, \ldots, f_m^H)$ est semi-régulière ,où f_i^H est la partie homogène de plus haut degré.

Note 6.2.1. La notion de généricité est également plus problèmatique dans $\mathbb{F}_2$ est: l'ensemble des suites de polynômes étant en effet fini, on peut seulement estimer la probabilité qu'une suite de polynômes soit semi-régulière. Pour des systèmes à coefficients dans $\mathbb{F}_2$, l'ensemble E de la définition 6.2.1 est de dimension zéro, et donc cette définition n'a plus de sens. Nous conjecturons cependant qu'une suite "tirée au hasard" sera semi-régulière sur $\mathbb{F}_2$, dans le sens ou la proportion de suites semi-régulières tend vers 1 quand n tend vers l'infini. Expérimentallement cette conjecture est parfaitement vérifiée pour les petites valeurs de n.

6.2.3 Définition alternative de semi-régularité

Dans (Pardue & Richert, 2003) une définition légérement différente de semi-régularité est donnée:

Definition 6.2.4. *(Pardue & Richert, 2003) Une suite de polynômes* $(f_1, \ldots, f_m)$ *de* $\mathbb{K}[x_1, \ldots, x_n]^m$ *est dites semi-regulière si pour tout* $i = 1, \ldots, m$*, la matrice de multiplication:*

$$(\mathbb{K}[x_1, \ldots, x_n]/\langle f_1, \ldots, f_{i-1}\rangle)_{d-\deg(f_i)} \xrightarrow{f_i} (\mathbb{K}[x_1, \ldots, x_n]/\langle f_1, \ldots, f_{i-1}\rangle)_d$$

est de rang plein pour tout d*.*

La défintion de semi-régularité 6.2.3 est plus générale que la définition de semi-régularité au sens de la définition 6.2.4: en effet cette dernière définition implique que toute sous-suite $f_1, \ldots, f_i$ est encore semi-régulièrece qui n'est pas vrai avec notre définition (il suffit de considérer l'exemple $(f_1 = x_1^2, f_2 = x_1x_2, f_3 = x_2^2)$). Par suite la propriéte 6.4.1 du théorème 6.4.1 n'est plus vraie pour les suites semi-regulières au sens de Pardue-Richert, mais les résultats de complixité s'appliquent.

6.3 Complexité de l'algorithme F_5

Dans ce chapitre on donne deux types de résultat concernant la complexité de l'algorithme F_5: d'une part des résultats généraux sur le degré atteint par l'algorithme F_5 pour les suites régulières ou semi-régulières: ceci donne un moyen explicite de calculer d_{reg}. C'est l'objet de la présente section. Dans la section 6.6 on donne des résultats beaucoup plus précis permettant d'estimer le nombre d'opérations arithmétiques lorsque le système algébrique vérifie des hypothèse supplémentaires (position de Noether simultané).

6.3.1 Degré maximal atteint par l'algorithme F_5

Le but de section est de prédire le degré maximal atteint par l'algorithme F_5 et d'en déduire une estimation de complexité.

Ceci est résumé dans le théorème suivant:

Theorem 6.3.1. *Pour une suite semi-régulière $(f_1, \ldots, f_m)$, il n'y a pas de réduction à 0 dans l'algorithme F_5 en degré inférieur à son degré de régularité d_{reg}; de plus d_{reg} est le degré en z du premier coefficient négatif de la série:*

$$\prod_{i=1}^{m} \left(\frac{1 - (1 - \delta_{\mathbb{K},\mathbb{F}_2})\, z^{d_i}}{1 + \delta_{\mathbb{K},\mathbb{F}_2} z^{d_i}} \right) \left(\frac{1 - \delta_{\mathbb{K},\mathbb{F}_2}\, z^2}{1 - z} \right)^n$$

où d_i est le degré total de f_i. Par conséquent, le nombre total d'opérations arithmétiques dans $\mathbb{K}$ nécessaire à F_5 (voir algorithme 31) est borné par

$$Cste \cdot M_{d_{reg}}(n)^{\omega}.$$

Preuve: On découpe la preuve en deux parties: d'abord le calcul explicite grace une récurrence des tailles des matrices puis le calcule des séries génératrices permettant le calcul explicite du degré de régularité d_{reg}.

6.3.2 Récurrence sur la taille des matrices. Degré maximal

L'idée est on qu'on suit pas à pas l'algorithme F_5 et on va calculer la taille des matrices dans les algorithmes 31 et **??** qui sont de la forme suivante en degré d:

$$M^{(d)}([f_1, \ldots, f_i]) = \begin{array}{c|c|c} & \text{termes de degré } d \text{ en } x_1, \ldots, x_n \\ t_1\, f_1 & \cdots & \cdots \\ t_3\, f_2 & \cdots & \cdots \\ t_3\, f_3 & \cdots & \cdots \end{array}$$

où les t_i sont des termes de degré $d - \deg(f_i)$. Comme ces matrices sont de rang maximal, le degré où l'algorithme F_5 termine est précisemment le degré d où le nombre de lignes dans la matrice A_d est *supérieur* au nombre de colonnes. Estimer le nombre de colonnes est facile car c'est exactement le nombre de termes en $x_1, \ldots, x_n$ en degré d; pour estimer le nombre de lignes il suffit d'observer comment on construit les matrices dans F_5 (voir algorithme 31):

$$M^{(d)}([f_1, \ldots, f_i]) := \begin{array}{c} \\ t x_j f_i \end{array} \left| \begin{array}{c} \widetilde{M^{(d)}}([f_1, \ldots, f_i]) \\ \cdots \end{array} \right|_{x_j f}$$

Le critère F_5 implique que $t x_j f_i$ figure dans cette matrice si $t x_j \notin \mathrm{Id}(\mathrm{LT}(G_{j-1+\delta_{\mathbb{K},\mathbb{F}_2}}))$, où G_j est la base de Gröbner de $[f_1, \ldots, f_j]$. Par conséquent si on note $U_{d,i}(n)$ le nombre de lignes de la matrice $M^{(d)}([f_1, \ldots, f_i])$ on a la relation de récurrence pour $d \geq 2$:

$$U_{d,i}(n) = i \cdot \underbrace{M_{d-d_i}(n)}_{\substack{\text{nombre de termes} \\ \text{de degré } d - d_i}} - \underbrace{\sum_{j=1}^{i-1+\delta_{\mathbb{K},\mathbb{F}_2}} U_{d-d_i,j}(n)}_{\text{critères } F_5} \tag{6.2}$$

On note $h_{d,i}(n)$ la différence entre le nombre de lignes et le nombre des colonnes de $M^{(d)}([f_1, \ldots, f_i])$:

$$h_{d,i}(n) = M_d(n) - U_{d,i}(n)$$

La condition d'arret est donc équivalente à trouver le plus petit d tel que $h_{d,m}(n) < 0$. Cependant comme $h_{d,m}(n)$ est un polynôme en n il est plus simple de fixer d et de calculer N_d la plus grande racine réelle des polynômes $h_{d,i}(n)$. Par exemple dans le cas d'équations quadratiques, $m = n$ sur $\mathbb{F}_2$: en utilisant la relation de récurrence (6.2) on peut calculer explicitement:

$U_{0,i}(n) = 0$
$U_{1,i}(n) = 0$
$U_{2,i}(n) = i \binom{n}{0} - 0 = i$
$U_{3,i}(n) = i \binom{n}{1} - \sum_{j=1}^{i} U_{1,j}(n) = i\,n$
$U_{4,i}(n) = i \binom{n}{2} - \sum_{j=1}^{i} U_{2,j}(n) = i \frac{n\,(n-1)}{2} - \sum_{j=1}^{i} j = \frac{i\left(n^2-n-i-1\right)}{2}$

Puis:

$$\begin{aligned} h_{3,n}(n) &= M_3(n) - U_{3,n}(n) \\ &= \binom{n}{3} - n^2 \\ &= \frac{n\left(n^2-9\,n+2\right)}{6} \end{aligned}$$

Il suffit ensuite de calculer la plus grande racine réelle de ce polynôme (utiliser l'algorithme d'isolation des racines réelles 21.2):

$$h_{3,n}(n) = n \left(n - 9/2 - 1/2\sqrt{73}\right) \left(n - 9/2 + 1/2\sqrt{73}\right)$$

la plus grande racine est dabs ce cas: $9/2 + 1/2\sqrt{73} \approx 8.772$ et donc $N_3 = \mathbf{9}$. En poursuivant les calculs on trouve:

d	3	4	5	6	7	8	9
N_d	9	16	24	32	41	49	58

Table 6.1. Degré maximal sur F_2

Pour lire ce tableau il faut partir de la ligne inférieure:

(i)Lorsque le nombre de variables n est inférieur à $N_3 = 9$ alors le degré maximal atteint par l'algorithme F_5 est 3; par conséquent la taille de la plus grande matrice est $n^3 \times n^3$ et donc une complexité arithmétique globale en $O(n^9)$.

(ii) Pour $N_3 = 9 \leq n < N_4 = 16$ le degré maximal est 4 et donc une complexité de $O(n^{12})$.
(iii) Pour $N_4 = 16 \leq n < N_5 = 24$ le degré maximal est 5 et donc une complexité de $O(n^{15})$.

De cette façon on construit la courbe théorique (rouge) de la figure 6.5.

6.3.3 Série génératrice.

Pour obtenir un calcul plus explicite du degré maximal on va calculer la série génératrice:

$$H_m(z) = \sum_{d=0}^{\infty} h_{d,m}(n) z^d$$

En soustrayant l' équation (**??**) on trouve:

$$U_{d,m}(n) - U_{d,m-1}(n) = M_{d-d_m}(n) - U_{d-d_m,m-1+\delta_{\mathbb{K},\mathbb{F}_2}}(n)$$

Comme $h_{d,m}(n) = M_d(n) - U_{d,m}(n)$ on a immédiatement

$$h_{d,m}(n) - h_{d,m-1}(n) = -h_{d-d_m,m-1+\delta_{\mathbb{K},\mathbb{F}_2}}(n) \tag{6.3}$$

d'où en passant aux séries (on note δ pour $\delta_{\mathbb{K},\mathbb{F}_2}$dans cette preuve):

$$\begin{aligned} H_m(z) &= \sum_{d=0}^{\infty} h_{d,m-1}(n) z^d - z^{d_m} \sum_{d=0}^{\infty} h_{d,m-1+\delta}(n) z^d \\ &= \sum_{d=0}^{\infty} h_{d,m-1}(n) z^d - \delta\, z^{d_m} \sum_{d=0}^{\infty} h_{d,m}(n) z^d - (1-\delta)\, z^{d_m} \sum_{d=0}^{\infty} h_{d,m-1}(n) z^d \end{aligned}$$

et donc en groupant les termes:

$$\left(1 + \delta\, z^{d_m}\right) H_m(z) = \left(1 - (1-\delta)\, z^{d_m}\right) \sum_{d=0}^{\infty} h_{d,m-1}(n) z^d$$

et par suite:

$$\begin{aligned} H_m(z) &= \frac{1-(1-\delta)\, z^{d_m}}{1+\delta\, z^{d_m}} \sum_{d=0}^{\infty} h_{d,m-1}(n) z^d \\ &= \cdots \\ &= \prod_{i=1}^{m} \left(\frac{1-(1-\delta)\, z^{d_i}}{1+\delta\, z^{d_i}}\right) \sum_{d=0}^{\infty} h_{d,0}(n) z^d \\ &= \prod_{i=1}^{m} \left(\frac{1-(1-\delta)\, z^{d_i}}{1+\delta\, z^{d_i}}\right) \sum_{d=0}^{\infty} M_d(n) z^d \\ &= \prod_{i=1}^{m} \left(\frac{1-(1-\delta)\, z^{d_i}}{1+\delta\, z^{d_i}}\right) \left(\frac{1-\delta\, z^2}{1-z}\right)^n \end{aligned}$$

Pour calculer d_{reg} il suffit donc de développer cette série et de calculer le premier indice dont le coefficient est < 0. Ceci termine la preuve du théorème.

Corollary 6.3.1. *Pour des équations quadratiques les séries sont:*
$H_m(z) = \frac{(1-z^2)^m}{(1-z)^n}$ *pour un corps quelconque* $\mathbb{K}$
$H_m(z) = \frac{(1+z)^n}{(1+z^2)^m}$ *pour le corps* $\mathbb{F}_2$

6.3.4 Exemple: NTRU

Exemple d'application des formule: NTRU.

Le problème fondamental dans crypto-système NTRU (Hoffstein *et al.* , 1998) est le suivant: étant donné un polynôme $H(X)$ de degré n à coefficients dans $\mathbb{Z}_{2^q}$ on cherche $N(X), D(X)$ des polynômes de degré n à coefficients dans $\{0,1\}$ tels que:

$$H(X) = \frac{N(X)}{D(X)} \bmod (X^n - 1) \bmod q.$$

Il est clair que ce problème revient à résoudre un système *linéaire sous-determiné*:

$$\begin{cases} l_1(x_1, \ldots, x_n) = y_1 \\ \cdots \\ l_n(x_1, \ldots, x_n) = y_n \end{cases}$$

où l_i une équation linéaire à coefficients dans l'anneau 2^k. On cherche une solution $(x_1, \ldots, x_n, y_1, \ldots, y_n)$ dans $\{0,1\}^{2\,n}$. En utilisant les deux premiers bits des vecteurs de Witt, Smart Vercauteren et Silverman (Smart *et al.* , 2005) montrent que la recherche de la solution se ramène à la résolution de n équations quadratiques n variables dans $\mathbb{F}_2$. Dans (Bourgeois, 2006) l'auteur propose d'utiliser les 3ème et 4ème bit des vecteurs de Witt pour améliorer l'attaque précédente: on obtient ainsi n équations supplémentaires de degré 4 (resp. 8) pour le 3ème bit (resp. 4ème bit). Peut on évaluer le gain en complexité pour des valeurs crypto réalistes ($n = 251$ ou $n = 503$) ?

Il suffit d'appliquer la méthode du théorème 6.3.1 (avec $\delta_{\mathbb{K},\mathbb{F}_2} = 1$): on calcule les séries suivantes et on cherche le premier coefficient négatif:

1 er bit: $(1+z)^n\,(1+z^2)^{-n} = 1 + n\,z + \frac{n\,(n-3)}{2} z^2 + \cdots$
2 ème bit: $(1+z)^n\,(1+z^2)^{-n}\,(1+z^4)^{-n}$
3 ème bit: $(1+z)^n\,(1+z^2)^{-n}\,(1+z^4)^{-n}\,(1+z^8)^{-n}$

Dans le tableau suivant on reporte la valeur de d_{reg} ainsi obtenue:

n	1 bit	2bit	3bit
251	29	28	28
503	53	52	52

Le gain que l'on peut esperer *a priori* est donc faible.

6.4 Caractérisation des suites semi-régulières.

Les propriétés des suites semi-régulières sont résumées dans le théorème 6.4.1; on aurra besoin de la notation:

Definition 6.4.1. *Pour une série* $\sum_{i=0}^{\infty} a_i z^i$, *la notation* $\left[\sum_{i=0}^{\infty} a_i z^i\right]^+$ *désigne la série* $\sum_{i=0}^{\infty} b_i z^i$ *avec* $b_i = \begin{cases} a_i & si \;\; a_j > 0,\ \forall 0 \leq j \leq i \\ 0 & sinon \end{cases}$

Proposition 6.4.1. *Soit* $(f_1, \ldots, f_m)$ *une suite de* m *polynômes en* n *variables,* $d_i = \deg(f_i)$ *. Alors:*

(i) La suite $(f_1, \ldots, f_m) \subset R_{\mathbb{K}}$ *est semi-régulière si et seulement si la série de Hilbert de la suite* $(f_1^H, \ldots, f_m^H)$ *est donné par :*

$$[H_m(z)]^+$$

où $H_m(z) = \prod_{i=1}^{m} \left(\frac{1-(1-\delta_{\mathbb{K},\mathbb{F}_2})\, z^{d_i}}{1+\delta_{\mathbb{K},\mathbb{F}_2} z^{d_i}}\right) \left(\frac{1-\delta_{\mathbb{K},\mathbb{F}_2}\, z^2}{1-z}\right)^n$.

(ii) Pour un corps $\mathbb{K}$ *quelconque et* $m \leq n$, *la suite* $(f_1, \ldots, f_m)$ *est régulière si et seulement si elle est semi-régulière: dans ce cas les deux notions sont les mêmes.*

(iii) Le degré de régularité de l'idéal engendré par $(f_1, \ldots, f_m)$ *est l'indice du premier coefficient négatif de la série* $H_m(z)$.

Proof. ((Bardet *et al.* , 2005; Bardet, 2004)) On démontre (i) pour des polynômes homogènes. Considérons la suite exacte:

$$\begin{aligned} 0 &\to \left(k[x_1, \ldots, x_n]/\langle f_1, \ldots, f_{i-1+\delta_{\mathbb{K},\mathbb{F}_2}}\rangle\right)_{d-d_i} \\ &\xrightarrow{f_i} \left(k[x_1, \ldots, x_n]/\langle f_1, \ldots, f_{i-1+\delta_{\mathbb{K},\mathbb{F}_2}}\rangle\right)_d \\ &\to \left(k[x_1, \ldots, x_n]/\langle f_1, \ldots, f_i\rangle\right)_d \\ &\to 0 \end{aligned}$$

alors tant que $d < d_{\text{reg}}$ la fonction de Hilbert correspondante vérifie (Cox *et al.* , 1998):

$$\text{HF}_{\langle f_1,\ldots,f_{i-1+\delta_{\mathbb{K},\mathbb{F}_2}}\rangle}(d-d_i) - \text{HF}_{\langle f_1,\ldots,f_{i-1}\rangle}(d) + \text{HF}_{\langle f_1,\ldots,f_i\rangle}(d) = 0 \quad (6.4)$$

pour tout $d < d_{\text{reg}}$. De plus, $\text{HF}_{\langle f_1,\ldots,f_i\rangle}(d) = 0$ pour tout i,d; comme $\text{HF}_{\langle 0\rangle}(d) = M_d(n)$ on retrouve la même relation de récurrence que l'équation (6.3). Par conséquent $\text{HS}_{\langle f_1,\ldots,f_m\rangle}(z) = H_m$.

Réciproquement, considérons la suite exacte

$$\begin{aligned} 0 &\to K_{d-d_i} \\ &\left(k[x_1, \ldots, x_n]/\langle f_1, \ldots, f_{i-1+\delta_{\mathbb{K},\mathbb{F}_2}}\rangle\right)_{d-d_i} \\ &\xrightarrow{f_i} \left(k[x_1, \ldots, x_n]/\langle f_1, \ldots, f_{i-1+\delta_{\mathbb{K},\mathbb{F}_2}}\rangle\right)_d \\ &\to \left(k[x_1, \ldots, x_n]/\langle f_1, \ldots, f_i\rangle\right)_d \\ &\to 0 \end{aligned}$$

où K est le noyau de la fonction de multiplication par f_i, alors pour tout $d < d_{\text{reg}}$ le noyau $K_{d-d_i} = \{0\}$, et donc d'après la définition 6.2.3 la suite est semi-régulière.

La propriété (ii) découle de (i) et du théorème 2.8.4 page 45. La propriété (*iii*) est une conséquence de la définition 6.2.2.

6.5 Développements asymptotiques de d_{reg}

Dans cete section nous présentons une analyse asymptotique du degré de régularité d'un idéal de dimension zéro défini par une suite semi-régulière sur un corps $\mathbb{K}$ ou sur $\mathbb{F}_2$.

6.5.1 Méthode du col.

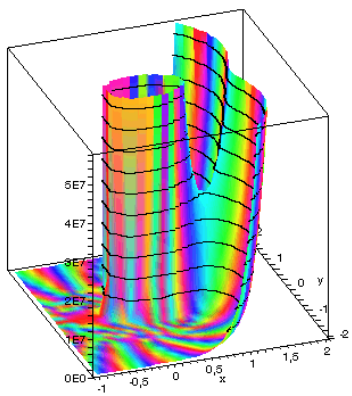


Fig 6.5.1: un point col simple.

Des méthodes de points cols et de points cols coalescents sont utilisées (pour le détail des preuves on se reporte à (Bardet, 2004)) pour calculer un développement asymptotique du degré de régularité d_{reg} lorsque le nombre de variables n tend vers l'infini. Nous donnons de nombreux résultats explicites de ces développements, dont les premiers termes fournissent une approximation quasi exacte du degré de régularité et ceci même pour des petites valeurs de n. On cherche dans la série suivante le premier indice d pour lequel le coefficient de degré d est négatif.

$$H_m(z) = \prod_{i=1}^{m} \left(\frac{1-(1-\delta)\, z^{d_i}}{1+\delta z^{d_i}} \right) \left(\frac{1-\delta\, z^2}{1-z} \right)^n$$

Pour cela la méthode se résume à:

- écrire le d-th coefficient de la série en utilisant la formule de Cauchy:

$$\mathcal{I}_n(d) = s_{d,m}(n) = \frac{1}{2\imath\pi} \oint H_m(z) \frac{dz}{z^{d+1}} = \frac{1}{2\imath\pi} \oint e^{n\, f(z)} dz \tag{6.5}$$

  où le chemin d'intégration est un lacet simple, entourant l'origine et aucune autre pôle de $H_m(z)$.

- calculer le terme dominant de $\mathcal{I}_n(d)$ en fonction de d et n lorsque $n \to \infty$, d étant considéré comme un paramètre.
- trouver la valeur de d qui annule ce terme dominant: ceci nous donne le premier terme dans le developpement asymptotique de d_{reg} .

En itérant ce processus on peut calculer les termes suivants dans le developpemnt de d_{reg}.

Le développement asymptotique de $\mathcal{I}_n(d)$ est calculé en utilisant la méthode du col ou des points cols coalescents. L'idée de ces méthodes est de faire passer le chemin d'intégration par les points cols (les zéros de $f'(z)$ voir la figure 6.5.1) de la fonction à intégrer. On montre alors que la contribution des parties du chemin qui ne sont pas voisines des cols est asymptotiquement négligeable, et qu'au voisinage de ces cols la fonction à intégrer peut être approchée par une fonction gaussienne (pour la méthode des cols) ou une fonction d'Airy (pour la méthode des points cols coalescents). Nous renvoyons le lecteur à des ouvrages comme (Chester *et al.* , 1957) pour plus de détails.

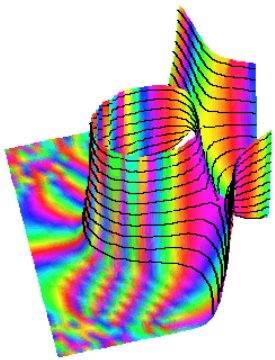


Fig 6.5.1: Points cols coalescents.

6.5.2 Classification

On considère une suite semi-régulière constituée d'équations $(f_1, \ldots, f_m)$. Le tableau suivant résume le résultat de plusieurs théorèmes donne le développement asymptotique de d_{reg} lorsque $n \to \infty$ en fonction de la valeur du rapport entre le nombre d'équations et le nombre de variables $\frac{m}{n}$.

Légende des symboles utilisés dans le tableau:

k est une constante (qui ne dépend pas de n).

d_i est le degré total de f_i.

$H_k(X)$ est le k ème polynôme d'Hermite; $h_{k,1}$ est le plus grand zéro de H_k (tous les zéros de $H_k(X)$ sont réels).

$a_1 \approx -2.3381$ est le plus grand zéro de la fonction d'Airy (solution de $\frac{\partial^2 y}{\partial z^2} - z\, y = 0$).

$\Phi(z) = \frac{z}{n}\frac{\partial}{\partial z} \log\left((1-z)^n \prod_{i=1}^{m}(1-z^{d_i})^{-1}\right) = \frac{z}{1-z} - \frac{1}{n}\sum_{i=1}^{m}\frac{d_i z^{d_i}}{1-z^{d_i}}$ et z_0 est la racine de $\Phi'(z)$ qui minimise $\Phi(z_0) > 0$.

m	Degré	d_{reg}
$m < n$	$\mathbb{K}, d_i = 2$	$m + 1$ (Borne de Macaulay)
$n + 1$	$\mathbb{K}$	$\sum_{i=1}^{n+1} \frac{d_i - 1}{2}$ (A. Szanto)
$n + k$	$\mathbb{K}, d_i = 2$	$\frac{m}{2} - h_{k,1}\sqrt{\frac{m}{2}} + o(1)$
$n + k$	$\mathbb{K}$	$\sum_{i=1}^{n+k} \frac{d_i - 1}{2} - h_{k,1}\sqrt{\sum_{i=1}^{n+k} \frac{d_i^2 - 1}{6}} + o(1)$
$2\,n$	$\mathbb{K}, d_i = 2$	$\frac{n}{11.6569} + 1.04\,n^{\frac{1}{3}} - 1.47 + 1.71\,n^{-\frac{1}{3}} + O\left(n^{-\frac{2}{3}}\right)$
$k\ n$	$\mathbb{K}, d_i = 2$	$(k - \frac{1}{2} - \sqrt{k(k-1)})n + \frac{-a_1}{2(k(k-1))^{\frac{1}{6}}} n^{\frac{1}{3}} + O(1)$
$k\ n$	$\mathbb{K}$	$\Phi(z_0)\,n - a_1\left(-\frac{1}{2}\Phi''(z_0)z_0^2\right)^{\frac{1}{3}} + O\left(n^{\frac{1}{3}}\right)$
n	$\mathbb{F}_2, d_i = 2$	$\frac{n}{11.1360} + 1.0034\,n^{\frac{1}{3}} - 1.58 + O(n^{-\frac{1}{3}})$
$k\ n$	$\mathbb{F}_2, d_i = 2$	$\left(-k + \frac{1}{2} + \frac{1}{2}\sqrt{2k(k-5) - 1 + 2(k+2)\sqrt{k(k+2)}}\right) n$

Table 6.2. Généralisations de la borne de Macaulay

Note 6.5.1. Dans le cas où $m = n + 1$ on a $h_{1,1} = 0$ et donc $d_{\text{reg}} = \frac{m}{2} + o(1)$ ce qui est en accord avec le résultat d'A. Szanto (Szanto, 2004).

Note 6.5.2. Afin d'illustrer la précision de ces développement asymptotique on trace sur même courbe: les points obtenus lors de la preuve du théorème 6.3.1 dans le cas de $m = n$ équations quadratiques sur $\mathbb{F}_2$, les courbes obtenus en partant du tableau en considérant deux ou trois termes.

Le dessin montre que les courbes ne sont pas discernables et donc que les développements asymptotiques sont suffisants avec de petites valeurs de n.

6.6 Amélioration des bornes de complexité de F_5

Même si les résultats obtenus jusqu'à maintenant sont optimaux pour l'estimation du paramètre de complexité d_{reg} l'analyse de la complexité arithmétique reste sommaire: une fois connu d_{reg} on estime de la plus grande matrice apparaissant dans le calcul: c'est une matrice $M_{d_{\text{reg}}}(n) \times M_{d_{\text{reg}}}(n)$ et donc le coût total est estimé à $M_{d_{\text{reg}}}(n)^3$. S'il est vrai que pour un système semi-régulier les matrices sont de tailles croissantes on va voir, et l'expérience le confirme, que ce n'est pas la dernière matrice qui est la plus couteuse à traiter: cette dernière matrice est en effet quasi-triangulaire et il y a peu de travail à effectuer pour la rendre triangulaire.

On considère des polynômes *homogènes* $(f_1, \ldots, f_m)$ dans $\mathbb{K}[x_1, \ldots, x_n]$, d_i est le degré total de f_i et on se limite explicetement à l'ordre $<_{\text{DRL}}$. Dans cette étude $m \leq n$.

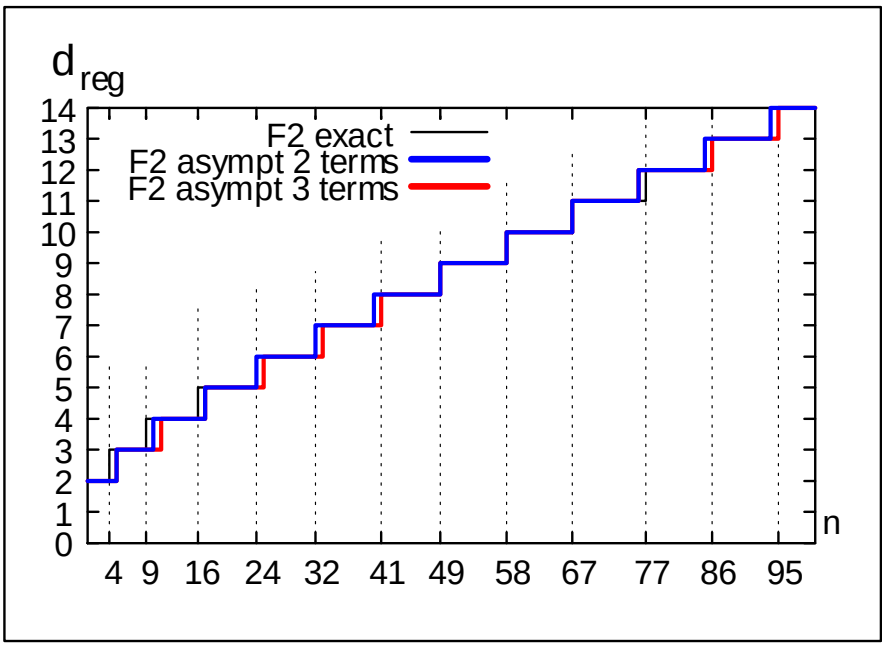


Fig. 6.2. précision des développements asymptotiques.

6.6.1 Position de Noether

Definition 6.6.1. *Les variables* $(x_1, \ldots, x_m)$ *sont en position de Noether par rapport au système* $(f_1, \ldots, f_m)$ *si dans* $\mathbb{K}[x_1, \ldots, x_n]/\langle f_1, \ldots, f_m\rangle$, $\overline{x_i}$ *est un entier algébrique sur* $\mathbb{K}[x_{m+1}, \ldots, x_n]$ *et de plus* $\mathbb{K}[x_{m+1}, \ldots, x_n] \cap \langle f_1, \ldots, f_m\rangle = \langle 0\rangle$.

Géométriquement cela signifie que le système est de dimension $n - m$ et que dans la cloture algébrique de $\mathbb{K}$ le nombre de solution (avec multiplicité) reste invariant quelque soit la spécialisation des variables $(x_{m+1}, \ldots, x_n)$.

La proposition suivante donne une caractérisation commode:

Proposition 6.6.1. *((Bermejo & Gimenez, 2001), lemme 4.1). Les variables* $(x_1, \ldots, x_m)$ *sont en position de Noether par rapport au système* $(f_1, \ldots, f_m)$ *si et seulement si pour tout* $1 \leq j \leq m$ *il existe* $n_j \in \mathbb{N}$ *tel que* $x_j^{n_j} \in \mathrm{LT}_{<_{DRL}}(\langle f_1, \ldots, f_m\rangle)$ *c'est à dire* $x_j \in \sqrt{\mathrm{LT}_{<_{DRL}}(\langle f_1, \ldots, f_m\rangle)}$.

6.6.2 Position de Noether simultanée.

F_5 étant incrémental il est nécessaire d'avoir une propriété plus forte:

Definition 6.6.2. *(SNP) Les variables* $(x_1, \ldots, x_m)$ *sont en position de Noether simultané par rapport au système* $(f_1, \ldots, f_m)$ *si* $(x_1, \ldots, x_i)$ *est en position de Noether par rapport au système* $(f_1, \ldots, f_m)$ *pour tout* $i \in \{1, \ldots, m\}$.

En utilisant la proposition 6.6.1 on obtient la définition suivante:

Definition 6.6.3. *(SNP). Soient des polynômes* homogènes $(f_1, \ldots, f_m)$ *dans* $\mathbb{K}[x_1, \ldots, x_n]$, *et l'ordre DRL* $<_{DRL}$. *Si* G_i *est une base de Gröbner de* $(f_1, \ldots, f_i)$ *pour l'ordre* $<_{DRL}$ *pour* $1 \leq i \leq m \leq n$. *On dit que* $(f_1, \ldots, f_m)$ *est en position de Noether simultanée si*

$$\text{pour tout } i \in \{1, \ldots, m\} \text{ on a } x_i \in \sqrt{LT(G_i)} \text{ et } x_i \notin \sqrt{LT(G_{i-1})}.$$

Note 6.6.1. Encore une fois c'est une propriété qui est vraie pour des polynômes aléatoires. On pourra aussi voir la thèse (Hashemi, 2006) et une définition de position de Noether forte.

Proposition 6.6.2. *Si* $(f_1, \ldots, f_m)$ *est en SNP alors* $(f_1, \ldots, f_m)$ *est une suite régulière.*

Lemma 6.6.1. *Si* $(f_1, \ldots, f_m)$ *est en SNP alors pour tout* $1 \leq i \leq m$,

$$(f_1, \ldots, f_i, x_{i+1}, \ldots, x_n)$$

est une suite régulière.

Proof. D'après la définition 6.6.3, $\sqrt{LT(G_i)} \subset \{x_1, \ldots, x_i\}$ et donc

$$(f_1, \ldots, f_i, x_{i+1}, \ldots, x_n)$$

est un idéal zéro-dimensionel.

6.6.3 Structure d'une base DRL.

Dans la suite on suppose que G_i est une base de Gröbner de $(f_1, \ldots, f_i)$ pour l'ordre $<_{\text{DRL}}$ calculé par l'algorithme F_5. Par conséquent on pour chaque $g \in G$ on a aussi la signature $\mathcal{S}(g) = s_g = (i_g, t_g) \in \mathbb{N} \times T$. On rappelle que ceci implique (voir chapitre 5) l'existence d'une écriture:

$$g = (t_g + \cdots)\, f_{i_g} + (\cdots)\, f_{i_g+1} + \cdots$$

Le lemme suivant donne la structure des termes de têtes d'une base de Gröbner pour un ordre DRL:

Lemma 6.6.2. *Pour tout polynôme* g *apparaissant dans le calcul de* G_i *dont la signature est* $s_g = (i, t)$ *avec* $t \in \mathbb{K}[x_1, \ldots, x_i]$ *on a* $\text{LT}(g_i) \in \mathbb{K}[x_1, \ldots, x_i]$.

Proof. Soit k le plus grand j tel $x_j|\,\mathrm{LT}(g)$;supposons que $k > i$. Alors (d'après algorithme F_5) il existe $(g_1, \ldots, g_i)$ tels que

$$g = \sum_{j=1}^{i} g_j\, f_j \text{ avec } \mathrm{LT}(g_i) = t \in \mathbb{K}[x_1, \ldots, x_i]$$

et g_i est réduit modulo $[f_1, \ldots, f_{i-1}]$. Comme $\mathrm{LT}(g) = 0 \mod [x_k, \ldots, x_n]$ on a (voir la proposition 2.3.2) $g = 0 \mod [x_k, \ldots, x_n]$; a fortioiri $g = 0 \mod [x_{i+1}, \ldots, x_n]$. De plus t est encore le terme de tête de $g_i \mod [x_{i+1}, \ldots, x_n]$ et donc $g_i \neq 0 \mod [x_{i+1}, \ldots, x_n]$. Ainsi $(x_{i+1}, \ldots, x_n, f_1, \ldots, f_i)$ n'est pas une suite régulière ce qui est contraire au lemme 6.6.1.

Le théorème suivant est plus précis et il donne la structure des signatures des polynômes d'une base de Gröbner:

Theorem 6.6.1. *Si le système $(f_1, \ldots, f_m)$ est en SNP alors pour tout $(s_g, g) \in G_i$, $\mathrm{LT}(g) \in \mathbb{K}[x_1, \ldots, x_i]$ et s_g (la signature) est de la forme $s_g = (j, t)$ avec $j \leq i$ et t est un terme dans $\mathbb{K}[x_1, \ldots, x_{j-1}]$.*

Proof. Dans la preuve de ce théorème $\mathcal{T}_i$ est l'ensemble des termes en $x_1, \ldots, x_i$ et $\mathcal{T} = \mathcal{T}_n$. On raisonne par l'absurde et on suppose que l'ensemble

$$\mathcal{A} = \{(s_f, f) \in G_i \mid \text{tel que } 1 \leq i \leq n,\ s_f = (i, t) \text{ avec } t \in \mathcal{T} \backslash \mathcal{T}_{i-1}\}$$

tel que. On prend (s_h, h) le plus petit élément de $\mathcal{A}$ (c'est à dire avec la plus petite signature s_h). Donc $s_h = (i, s)$ avec $s \notin \mathcal{T}_{i-1}$. La seule façon de créer ce polynôme h dans F_5 est d'ajouter dans la matrice une ligne $t \cdot (h_0, s_{h_0})$ où $h_0 \in G_i$, $s_{h_0} = (i, t_0)$ et t est un terme de degré ≥ 1; ainsi $s_h = (i, t \cdot t_0)$. Comme (s_h, h) est le plus petit élément de $\mathcal{A}$ on sait que $t_0 \in \mathcal{T}_{i-1}$ et donc $\mathrm{LT}(h_0) \in \mathcal{T}_i$ (d'après le lemme 6.6.2). Comme $t \cdot t_0 = s \notin \mathcal{T}_{i-1}$, on peut trouver un indice $l \geq i$ tel que x_l divise t. Maintenant cette ligne (s_h, h) de la matrice a été réduite par une autre ligne: on peut donc trouver $k \leq i$, $g \in G_k$ et $w \in \mathcal{T}$ tels que $k \leq i$, $s_g = (k, v) < s_h$ et

$$w \cdot \mathrm{LT}(g) = \mathrm{LT}(h) = t \cdot \mathrm{LT}(h_0). \tag{6.6}$$

À cause de la minimalité de h on a $v \in \mathcal{T}_{k-1}$ et $\mathrm{LT}(g) \in \mathcal{T}_k$. Comme $\frac{t}{\gcd(t,w)} h_0$ est réductible par g et que $t \cdot h_0$ est le plus petit élément de $\mathcal{A}$ on en déduit que $\gcd(t, w) = 1$; par suite t divise $\mathrm{LT}(g) \in \mathcal{T}_k$ et donc $k = i = l$, $t \in \mathcal{T}_i$. De l'équation (6.6) on déduit immédiatement que $w \in \mathcal{T}_i$; mais si x_i divise w alors $\mathrm{LT}(g)$ divise $\frac{w}{x_i}\mathrm{LT}(g) = \frac{t}{x_i}\mathrm{LT}(h_0)$ et comme $\frac{t}{x_i} h_0$ est strictement inférieur à h dans $\mathcal{A}$ ceci est impossible. Donc, en fait, $w \in \mathcal{T}_{i-1}$ et $s_{w \cdot g} = (i, w \cdot v)$ avec $w \cdot v \in \mathcal{T}_{i-1}$. On obtient ainsi une contradiction car l'indice de la ligne h est $(i, t \cdot t_0)$ avec $x_i = x_l$ divisant t et donc a fortiori $t \cdot t_0$ qui est $<_{\mathrm{DRL}} w \cdot v \in \mathcal{T}_{i-1}$ (c'est une propriété de l'ordre DRL): l'opération élémentaire sur la ligne est donc interdite.

6.6.4 Nombre d'élements d'une base de Gröbner DRL.

Le théorème suivant donne une nouvelle borne très précise sur le nombre de polynômes dans la base de Gröbner:

Theorem 6.6.2. *Soit $(f_1, \ldots, f_m)$ un système homogène pour lequel les variables $(x_1, \ldots, x_n)$ sont en SNP. Soit G_i la base de Gröbner réduite de $(f_1, \ldots, f_i)$ pour l'ordre DRL et pour $1 \leq i \leq m$, alors le nombre de polynômes de degré d dans $G_i \setminus G_{i-1}$ est borné par $N_{d,i}$, où*

$$\sum_{d=0}^{\infty} N_{d,i} z^d = z^{d_i} \prod_{j=1}^{i-1} \frac{1 - z^{d_j}}{1 - z} \tag{6.7}$$

Proof. On fait la preuve par récurrence sur i. Si $i = 1$ alors par défintion de la position de Noether, la base est réduite à un seul polynôme dont le terme de tête est $x_1^{d_1}$ et donc l'équation (6.7) est correcte. Supposons la propriété vraie pour $i - 1$. Considèrons $g \in G_i$; on peut toujours l'écrire (algorithme F_5): $g = g_i f_i + \cdots + g_1 f_1$,pour des polynômes g_i. Alors d'après le lemme 6.6.2 on peut supposer que $\mathrm{LT}(g_i) \in \mathbb{K}[x_1, \ldots, x_{i-1}]$ et $\mathrm{LT}(g_i)$ réduit par rapport à G_{i-1}; or le nombre termes dans $\mathbb{K}[x_1, \ldots, x_{i-1}]$ qui ne sont pas top-réductibles par $\langle f_1, \ldots, f_{i-1} \rangle$ est exactement $N_{d,i} = \mathrm{HF}_{\langle f_1, \ldots, f_{i-1} \rangle}(d - d_i)$ dans $\mathbb{K}[x_1, \ldots, x_{i-1}]$ (on peut imaginer qu'on substitue $x_i = \cdots = x_n = 0$). En appliquant le théorème 2.8.4 avec $m = i - 1$ on a:

$$\sum_{d \geq 0} \mathrm{HF}_{\langle f_1, \ldots, f_{i-1} \rangle}(d)\, z^d = \sum_{d=0}^{\infty} N_{d+d_i,i} z^d = \frac{\prod_{j=1}^{i-1} (1 - z^{d_j})}{(1 - z)^{i-1}}$$

ce qui prouve le théorème.

Corollary 6.6.1. *Pour des équations quadratiques $\sum_{d=0}^{\infty} N_{d,i} z^d = z^2 (1 + z)^{i-1}$ et donc $N_{d,i} = \binom{i-1}{d-2}$.*

Si de plus $m = n$, le nombre total d'élements dans la base de Gröbner est majoré par $\sum_{i=1}^{n} \sum_{d=2}^{i+1} \binom{i-1}{d-2} = \sum_{i=1}^{n} 2^{i-1} = 2^n - 1$.

Note 6.6.2. En fait, en pratique, la borne du théorème 6.6.2 est exacte. On peut aussi comparer le résultat de ce théorème avec la borne du corollaire 3.1.1: on trouvait que le nombre d'éléments étaient bornés par $\leq n\, D(I) = n\, 2^n$. Dans le cas d'un SNP la borne 6.6.1 est meilleure.

6.6.5 Complexité arithmétique de F_5

Theorem 6.6.3. *Le nombre total d'opérations arithmétique utilisé par l'algorithme F_5 pour calculer une base de Gröbner basis d'un système homogène $(f_1, \ldots, f_m)$ pour lequel les variables $(x_1, \ldots, x_n)$ sont en SNP est borné par:*

$$N_{F_5} = \sum_{i=1}^{m} \sum_{d=d_i}^{D} N_{d,i} \binom{i+d-1}{d} \binom{n+d-1}{d}, \tag{6.8}$$

où $D = 1 + \sum_{j=1}^{m} (d_j - 1)$ *et* $N_{d,i}$ *est donné dans le théorème 6.6.2.*

Lorsque $m = n$ *et* $\deg(f_i) = d_i = 2$ *la formule se simplifie en*

$$N_{F_5} = \sum_{d=0}^{n-1} \binom{2d+2}{d} \binom{n+d+1}{d+2} \binom{n+d+2}{2d+3} - \binom{n+1}{2} \tag{6.9}$$

Proof. Les opérations arithmétiques proviennent des réductions durant l'élimination de Gauß. Pour tout $1 \leq i \leq m$ et tout $d_i \leq d \leq D$, d'après les théorèmes **??** et 6.6.2 il y au plus $N_{d,i}$ polynômes dans $M^{(d)}([f_1, \ldots, f_i])$ qui n'était pas dans $M^{(d)}([f_1, \ldots, f_{i-1}])$ et on doit les réduire par $\widetilde{M^{(d)}}([f_1, \ldots, f_{i-1}])$. D'après le lemme 6.6.2 les termes de têtes du résultat sont dans ading term of the result is in $\mathbb{K}[x_1, \ldots, x_i]_d$, ce qui limite à $\binom{i+d-1}{d}$ le nombre de lignes impliquées dans la réduction chacune de ces lignes ayant au plus $\binom{n+d-1}{d}$ éléments non nuls. On obitent ainsi la formule (6.8) . Pour la formule (6.9) voir dans (Bardet, 2004).

Lorsqu'on applique l'algorithme F_5 on se trouve face à deux stratégies: en degré d utiliser autant que possible les calculs effectués en degré $< d$ ou générer des matrices très creuses en utilisant uniquement des produits des équations initiales: dans le premier cas on a tendance à générer des matrices assez denses mais qui ont une structure triangulaires par blocs; dans le deuxième cas on génère une sous matrice de la matrice de Macaulay. La question naturelle est de déterminer quelle est la meilleure stratégie ? On peut faire le calcul explicitement pour des valeurs de n et comparer la formule (6.9) et la formule obtenue dans le théorème 6.3.1: par exemple pour $n = 30$ équations quadratiques sur $\mathbb{Q}$ on trouve

$N_{F_5} = 2^{122.6}$ par l'équation (6.9)

$2^{56.7\,\omega}$ avec le théorème 6.3.1

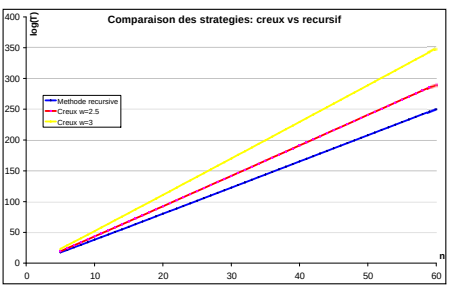


Fig. 6.3. comparaison de deux stratégies.

par conséquent la borne est bien meilleure pour la valeur réaliste de $\omega = 3$. Pour d'autres valeurs de n et de $\omega = 3$ ou 2.5 on reporte sur un dessin les différentes valeurs des bornes obtenues pour l'algorithme F_5. Même si le dessin montre que la nouvelle borne est bien meilleure que la borne du théorème 6.3.1, il est cependant

délicat de conclure en faveur d'une méthode ou d'une autre puisqu'on compare des bornes supérieures.

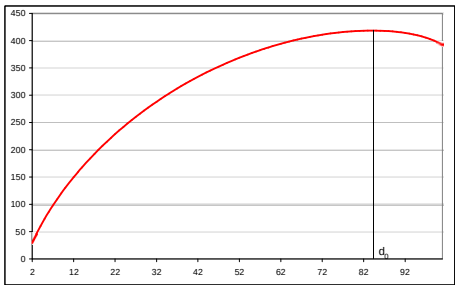


Fig. 6.4. Coût relatif des algorithmes.

La borne (6.8) permet aussi de répondre à une autre question: quelle est l'étape la plus coûteuse ? Plus exactement si on considère un système quadratique SNP ayant n équations et n variables (6.9) la dernière étape, en degré $d_{\text{reg}} = n + 1$, nécessite de générer la matrice dont la taille est la plus grande: cette étape est elle la plus coûteuse ? Afin de déterminer son maximum, on trace maintenant la fonction

$$F : d \longmapsto F(d) = \binom{2d+2}{d}\binom{n+d+1}{d+2}\binom{n+d+2}{2d+3}.$$

On étudie la fonction décroissante $f(d) = \frac{F(d+1)}{F(d)} - 1 = \frac{(2\,d+3)(n+d+2)(n+d+3)(n-d-1)}{(d+3)^2(d+1)(2\,d+5)} - 1$. On cherche la racine d_0 de f dans l'intervalle $\frac{n}{2} < d_0 < n$; lorsque $d > d_0$ on $F(d) < F(d_0)$. Asymptotiquement on cherche d_0 sous la forme $d_0 \approx \lambda n$ et on trouve:

$$f(\lambda n) = \frac{-2\lambda(2\,\lambda^3 + \lambda^2 - \lambda - 1)n^4 + \cdots}{\cdots}$$

Par conséquent le coefficient de n^4 dans le numérateur doit être nul: ainsi on trouve $d_0 \approx 0.83n$. Maintenant $N_{F_5} = \sum_{d=0}^{m-1} F(d) - \binom{n+1}{2}$ est majoré par $n\,F(0.83\,n)$ et en utilisant la formule:

$$\log\left(\binom{an+b}{cn+d}\right) \approx (a\log(a) - c\log(c) - (a-c)\,\log(a-c))n$$

on trouve $\log(F(d_0)) \approx F_0\,n$ avec $F_0 \approx 4.3$. On obtient donc le résultat suivant:

Theorem 6.6.4. *Le nombre total d'opérations arithmétique utilisé par l'algorithme F_5 pour calculer une base de Gröbner basis d'un système quadratique homogène $(f_1, \ldots, f_m)$ pour lequel les variables $(x_1, \ldots, x_n)$ sont en SNP est borné par:*

(i)pour l'algorithme F_5 matricielle:

$$N_{F_5} \approx 2^{4.3\,n+o(n)}$$

(iii) par la méthode d'élimination de Gauß de la matrice de Macaulay:

$$N_{Macaulay} \approx \frac{1}{4\pi^{\frac{3}{2}}\sqrt{n}} 2^{6\,n}$$

Proof. Pour la matrice de Macaulay en degré $d_{\rm reg} = n+1$ il y a $M_{d_{\rm reg}}(n) = \binom{n+d_{\rm reg}-1}{d_{\rm reg}} = \binom{2\,n}{n+1}$ colonnes et $n\,M_{d_{\rm reg}-2}(n) = n\binom{2\,n-2}{n-1}$ lignes. La complexité de l'élimination de Gauß est donc bornée par $n\,M_{d_{\rm reg}-2}(n)\,M_{d_{\rm reg}}(n)^2$. La fonction `asympt` de Maple donne le résultat.

6.7 Conclusion

Dans ce chapitre on a étendu la définition de suite semi-régulière pour les systèmes sur-determinés: on conjecture que presque tout système est une suite semi-régulière (en caractéristique 0 c'est une conséquence de la conjecture de (Fröberg, 1985)). Pour ces systèmes on donne des équivalents asymptotiques très précis du degré de régularité. Par exemple dans le cas de $2\,n$ équations en n variables on améliore la borne de Macaulay d'un facteur 11. De plus on répond à la question posée dans l'introduction et on peut étendre la courbe théorique (figs 6.1 et 6.5):

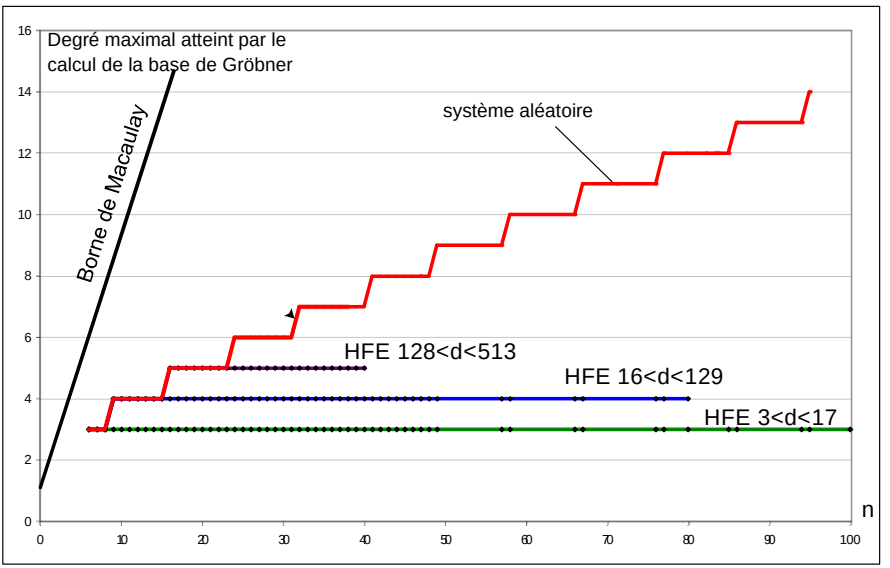


Fig. 6.5. Complexité: courbe théorique et pratique.

De plus le calcul d'une base de Gröbner d'un système semi-régulier ayant $m = \alpha n$ équations et n variables se fait en temps simplement exponentiel; par exemple un système ayant aléatoire avec 80 équations quadratiques est impossible à résoudre par les techniques Gröbner. Les systèmes algébriques constituent donc une source interessante de problème difficile qui peuvent être utilisé pour concevoir de nouveaux crypto-systèmes: par exemple les formules données dans ce chapitre ont été utilisé pour régler les paramètres du cryptosystème de chiffrement à flot QUAD ((Berbain *et al.* , 2006)).

Pour des suites réguliès vérifiant une propriété plus forte (être en position de Noether simultanée) on donne des estimations très précises de la complexité arithmétique de F_5 en utilisant la structure d'une base DRL calculée par cet algorithme: on améliore ainsi l'exposant de la borne de complexité d'un facteur $\frac{3}{2}$.

Part II

Applications à la Cryptologie

7. Applications en Cryptologie

7.1 Cryptanalyse algébrique

La cryptologie, est un domaine d'application priviligié pour les méthodes algébriques.

Beaucoup de cryptosystèmes et donc de problèmes de cryptanalyse possèdent une forte structure algébrique que celle-ci soit explicite (par exemple la clé publique d'une instance de HFE est un système polynomial quadratique multivarié sur le corps fini à 2 éléments) ou implicite. Bien entendu tout problème peut s'écrire sous forme d'un système polynomial booléen, une forte structure algébrique peut rendre ce système facile (disons plutôt moins difficile) à résoudre.

Beaucoup de cryptosystèmes et donc de problèmes de cryptanalyse possèdent une forte structure algébrique. La première étape de la cryptanalyse algébrique consiste donc à mettre en équation le problème. C'est parfois une étape très difficile: c'est le cas en particulier du système à chiffrement AES (standard de chiffrement retenu par l'administration américaine). Diverses mises en équation du problème (ou d'une version simplifiée du cryptosystème) sont ainsi en cours d'étude. Certaines mises en équation font intervenir plusieurs équations en plusieurs milliers de variables; d'autres mises en équation font intervenir beaucoup moins de variables, mais sont de degré plus élevé et/ou sur un corps de plus grande taille.

Une fois la mise en équation effectuée, la résolution se fait par un calcul de base de Gröbner. En première approximation, il suffit juste de changer l'arithmétique sur les coefficients dans les algorithmes généraux comme F_5. Cependant, pour le cas très particulier de $GF(2)$, il a été nécessaire de refondre complètement l'algorithme et surtout l'implantation afin de tenir compte de l'action du Frobenius ($f^2 = f$) qui rajoute des relations triviales; de plus, une nouvelle bibliothèque d'algèbre linéaire spécifique à $GF(2)$ a été créée; cette version, que nous nommons $F_5/2$, est plus rapide d'un facteur environ 400 que la meilleure implantation de F_4.

Ainsi calculer des bases de Gröbner de systèmes polynômiaux à coefficient dans $GF(2002)$ constitue sans conteste une nouvelle technique de cryptanalyse. Le premier succès de cette technique date de 2002 avec la résolution avec le logiciel FGb du challenge public de HFE. Une autre application, est l'application de ces techniques aux générateurs pseudo-aléatoires produits

par des registres filtrés. Cette application est extrêmement importante, car ces générateurs sont effectivement utilisés en cryptographie lorsque l'on veut atteindre des débits de chiffrement élevés.

Le premier problème fondamental auquel on est confronté dans les problèmes issus de la cryptographie est de reconnaître rapidement si un système algébrique est "aléatoire" ou non. Pour cela, nous avons été amenés à établir de nouvelles bornes de complexité théorique très précises pour les systèmes algébriques aléatoires dans un corps fini et les systèmes surdéterminés. Ceci est un travail en collaboration avec Bruno Salvy (projet Algo). Nous donnons en particulier l'équivalent de la borne de Macaulay (voir plus haut). Cette borne est quasi exacte dès que $n \geq 3$ et permet de calculer exactement les tailles des matrices qui apparaissent dans l'algorithme F_5. Ces bornes théoriques sont particulièrement utiles pour faire la distinction en pratique entre un système aléatoire (difficile) et un système provenant d'un cryptosystème particulier (comme HFE).

Lorsque les ressources sont limitées ou bruitées (téléphone GSM) et qu'on désire un débit élevé, on utilise des techniques de chiffrement différentes: le chiffrement par flot (nommé encore chiffrement à la volée). Dans ce contexte, les registres filtrés (LFSR) sont des systèmes performants pour générer des séquences aléatoires de très grande période à partir de clefs courtes (128 bits); ils sont de ce fait très utilisés dans les communications militaires. L'utilisation des techniques algébriques et en particulier de l'algorithme $F_5/2$ semble très concluante puisque des exemples réalistes de longueur 128 bits peuvent être résolus dès à présent (ces exemples sont largement hors de portée des autres techniques). Un contrat avec la DGA (Celar/Rennes) vise à étudier en *vrai grandeur* les registres attaquables par les bases de Gröbner en fonction des divers paramètres du registre (nature de la fonction filtrante, position des pattes, taille du registre, etc.). Ce travail fait l'objet d'une collaboration avec A. Canteaut (attaque par corrélation) du projet CODES. Une application directe de l'étude théorique de complexité sur les systèmes aléatoires permet d'énoncer immédiatement: pour un registre filtré de longueur n, étant donnés $n \log^2 n$ échantillons, on peut retrouver en temps sous-exponentiel l'état initial (secret) du registre. Un travail en commun (bourse CIFRE) avec la société Thalès débute également en 2003 sur ce sujet.

HFE (Hidden Fields Equations) est un cryptosystème à clef publique n'utilisant pas la théorie des nombres (comme RSA), mais des opérations sur les polynômes à coefficients dans un corps fini. Ce cryptosystème a été proposé par Jacques Patarin à Eurocrypt 96 en améliorant les idées de Matsumoto et Imai. Ce cryptosystème semble très prometteur car il peut servir à générer des signatures très courtes: 128, 100 ou même 80 bits. L'idée de HFE est de prendre un polynôme univarié secret à coefficients dans $GF(2^n)$ puis d'exprimer ce polynôme sur $GF(2)$. On obtient ainsi un système algébrique en n variables (la clef publique).

Retrouver le message original connaissant la clef secrète est "facile" puisque cela revient à résoudre un problème univarié. En revanche, avec seulement la clef publique, cela devient un problème très difficile puisqu'il s'agit de résoudre un système algébrique (complexité exponentielle dans le cas des systèmes aléatoires). Ce cryptosystème a été étudié par plusieurs cryptographes (Shamir, Courtois, etc.).

Les avancés algorithmiques (F_5) permettent maintenant de :

- casser assez facilement (deux jours de CPU) le Challenge proposé par J. Patarin (il s'agit d'un exemple réaliste en taille 80 bits). Cette taille de système (80 équations denses de degré 2) était totalement encore hors de portée il y a quelques mois ;
- mener une étude de complexité expérimentale pour les systèmes HFE. Pour les valeurs admissibles des paramètres la complexité du calcul d'une base de Gröbner est seulement $O(n^{10})$.

L'importance cryptographique croissante prise par les jacobiennes de courbes algébriques sur les corps finis, rend de plus en plus fréquent d'avoir affaire avec des idéaux dans les corps de fonctions des courbes ou de leurs jacobiennes. Calculer dans de tels corps de fonctions peut se faire en manipulant des idéaux d'algèbres de polynômes. Ceci permet donc de généraliser la loi de groupe bien connue sur les courbes elliptiques à des courbes de degré supérieur (cubiques ou plus). En utilisant des techniques de bases de Gröbner (algorithme FGLM en particulier) on peut maintenant des *formules explicites* pour la loi de groupes. Ces formules sont très efficaces (100 multiplications seulement sur le corps de base) et sont facilement implémentables en hardware.

7.1.1 Autre intro

Last years a new kind of cryptanalysis has made its entrance in cryptography: the so-called algebraic cryptanalysis. A fundamental issue of this cryptanalysis consists in finding zeroes of algebraic systems. Gröbner bases, which are a fundamental tool of commutative algebra, constitute the most elegant and efficient way for solving this problem. They provide an algorithmic solution for solving several problems related to algebraic systems (some of them can be found in (Adams & Loustaunau, 1994)). We present here a new application of Gröbner bases.

7.2 Problème mathématiquement difficile

Pour construire un système de chiffrement à clé publique l'idée (référence ????) est d'utiliser un problème "mathématiquement difficile dans un sens et facile dans l'autre": le problème du logarithme discret est un exemple bien

connu: dans ce cryptosystème de Diffie-Hellman (Diffie & Hellman, 1976). on se donne $(G, \times)$, un groupe cyclique fini, et un générateur g de G. Pour tout entier $m \in \mathbb{N}$ il est très facile, en général, de calculer $h = g^m$ l'opération inverse:

"étant donnés g et h dans G retrouver un entier m tel que $h = g^m$".

est un problème difficile (on verra des exemples de groupes dans le chapitre 11). Un autre exemple bien connu est celui de la factorisation des entiers: calculer $c = a \times b$ est très facile, en revanche retrouver a et b à partir de l'entier c est difficile.

7.2.1 Cryptographie multivariée

Pour concevoir un cryptosystème dans le cadre de la "cryptographie multivariée" (principalement l'oeuvre de Jacques Patarin (Matsumoto & Imai, 1988; Patarin, 1996b; Patarin & Goubin, 1997a) ???) il faut trois ingrédients:

1. un problème difficile construit à partir d'un problème portant sur des polynomes en plusieurs variables; la clé publique sera un système algébrique $\mathcal{S}$ qui devra ressemble à un système aléatoire.
2. une ‹trappe secrète› ou une façon de construire $\mathcal{S}$ à partir d'une clé secrète qui permette de résoudre facilement le problème 1 assez efficacement.
3. une façon très efficace de chiffrer ou de signer un message à partir de $\mathcal{S}$

Prenons l'exemple du cryptosystème $2R$ (Goubin & Patarin, 1997; Patarin & Goubin, 1997a): la clé publique est obtenue par composition de deux systèmes quadratiques facile à inverser: la clé publique est constituée de polynômes de degré 4. Pour chiffrer un message il suffit d'evaluer les polynômes de la clé; le problème difficile associé est la résolution d'un système algébrique. Comme la façon de construire la clé publique est connue (c'est le principe de Kerchoffs) on sait que cette clé est obtenue par composition de polynômes: par conséquent pour résoudre le problème 1 on peut dans un premier temps essayer de retrouver les polynômes utilisés dans la composition. Ainsi la robustesse de ce cryptosystème repose en fait sur un *deuxième* problème difficile qui est le problème de décomposition fonctionelle.

7.2.2 Problèmes difficiles

On liste quelques problèmes difficiles qui sont utilisés en Cryptographie multivariée:

Polynomial System Solving (PoSSo)
Input: des polynômes $f_1, \ldots, f_m$ dans $\mathbb{K}[x_1, \ldots, x_n]$.
Trouver un point $a = (a_1, \ldots, a_n) \in \mathbb{K}^n$ tel que
$$f_1(a_1, \ldots, a_n) = \cdots = f_m(a_1, \ldots, a_n) = 0.$$

Le problème PoSSo de résoudre un système d'équations algébriques dans $\mathbb{F}_2$ est NP-complet (Fraenkel & Yesha, 1979; Fraenkel & Yesha, 1980). Cependant souvent un problème, bien que NP-complet, est en pratique très facile à résoudre: c'est le cas par exemple du problème 3-SAT. Bien qu'il existe des instances difficiles du problème (le problème est NP-complet) pour toute instance tirée aléatoirement on peut résoudre le problème en temps polynômial.

Le problème PoSSo de résoudre un système algébrique semble lui être un problème génériquement difficile: si on admet que les suites semi-régulières sont génériques alors on sait (voir chapitre 6) que le degré d'un système algébrique de n équations quadratiques en n variables est de la forme $\alpha\, n$ où α est une constante; ceci implique que tout calcul utilisant un calcul de base de Gröbner se fait en temps simplement exponentiel. En pratique on observe également ce comportement exponentiel: ainsi un système de 80 e, i.e. que lorsque n +oo, la proportion de suites semi-régulières en n variables et m équations de degrés dl,. .. ,dm tend vers 1(le nombre d'équations m pouvant dépendre de n). Cela signifie que "presque toute suite" est semi-régulière. Or, nous avons vu Chapitre 4 que, pour des suites semi-régulières, on peut trouver un équivalent très précis du degré de régularité Dr„ du système. En admettant notre conjecture de généricité des suites semi-régulières, les bornes pour ces suites donnent des estimations de com- plexité en moyenne, et peuvent être utilisées comme bornes de complexité pour des systèmes issus de problèmes cryptographiques.

[FY80, FY791. Pour ce qui est du calcul d'une base de Grobner, nous avons vu chapitre 1 que le cas le pire est simplement exponentiel : le degré de régularité (voir Définition 3.5.1, c'est une borne sur le degré maximal d'un polynôme apparaissant dans un calcul de base de Grobner) est Dr„ < n.

Functional Decomposition Problem (FDP)
Input: des polynômes $h_1, \ldots, h_u$ dans $\mathbb{K}[x_1, \ldots, x_n]$.
Trouver: – s'il en existe – des polynômes $f_1, \ldots, f_u$, et $g_1, \ldots, g_n$, tels que:
$$(h_1, \ldots, h_u) = (f_1(g_1, \ldots, g_n), \ldots, f_u(g_1, \ldots, g_n)).$$

Isomorphism of Polynomials (IP)
Input: des polynômes $\boldsymbol{a} = (a_1, \ldots, a_u)$, et $\boldsymbol{b} = (b_1, \ldots, b_u)$ dans $\mathbb{K}[x_1, \ldots, x_n]^u$.
Trouver: – s'il en existe – des matrices $(S, U) \in GL_n(\mathbb{F}_q)^2$ telles que:
$$\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}S)U.$$
où $\boldsymbol{x}$ est le vecteur $[x_1, \ldots, x_n]$.

Le problème PoSSo est utilisé dans les cryptosystèmes C^*,HFE (Matsumoto & Imai, 1988; Patarin, 1996b): le problème facile est l'évaluation d'une liste de polynômes en un point de $\mathbb{K}^n$:

Polynomial System Evaluation (PSE)
Input: des polynômes $f_1, \ldots, f_m$ dans $\mathbb{K}[x_1, \ldots, x_n]$ et $a = (a_1, \ldots, a_n) \in \mathbb{K}^n$
Calculer: $b = (b_1, \ldots, b_m) \in \mathbb{K}^m$ où $b_1 = f_1(a_1, \ldots, a_n), \ldots, b_m = f_m(a_1, \ldots, a_n)$

7.2.3 Design

Pas uniquement pour la cryptanalyse

7.3 Autres travaux

Les travaux suivants ne sont pas abordé directement dans ce livre et on fait l'objet de la thèse de G. Ars.

7.3.1 Immunité Algébrique

Gwénolé Ars a généralisé les notions d'immunité algébrique en définissant l'immunité de flot (respectivement de bloc) d'une fonction de $\mathbb{F}_q^n$ vers $\mathbb{F}_q^n$. Il a montré que le calcul d'une base de Gröbner du système composé des équations d'une telle fonction et des équations du corps $\mathbb{F}_q$ fournit, pour des ordres monomiaux bien choisis, son immunité de flot ou de bloc. Pour des fonctions utilisées dans des algorithmes de chiffrement à flot tel que E0 (Bluetooth) ou le Summation Generator, G.Ars a obtenu ainsi une nouvelle technique permettant de déterminer efficacement, pour les différentes valeurs du nombre de sorties consécutives considérées, une base des relations de plus bas degré dans lesquelles les variables de mémoire ont été éliminées.

7.3.2 Registres LFSR

G. Ars et J.C. Faugère ont étudié en 2002 et 2003 la cryptanalyse, à l'aide de techniques de bases de Gröbner, des algorithmes de chiffrement à flot constitués d'un registre filtré non linéairement, et expérimenté diverses stratégie de résolution du système d'équations en la clé fourni par la donnée de N sorties. En appliquant un calcul de base de Gröbner à chaque équation de sortie isolément (ou à des "paquets d'équations" résultant d'un très petit nombre de sorties) avant de résoudre le système modifié ainsi précalculé, ils ont constaté des baisses de degré spectaculaires conduisant à des attaques efficaces. La thèse contient également une amélioration et des variantes des attaques algébriques rapides proposée par N. Courtois en 2003 (en collaboration avec F. Armknecht).

7.3.3 Etudes sur AES

Une étude expérimentale de variantes simplifiées de l'AES a été entreprise par G. Ars et J.C. Faugère en utilisant une modélisation algébrique et des outils algébriques fondés sur les bases de Gröbner: chaque tour de l'AES est modélisé par la succession d'un système de n équations quadratiques en $2n$ variables (n bits d'entrée et n bits de sortie), d'une transformation linéaire bijective, et d'une addition d'une fonction linéaire bijective de la clé distincte à chaque tour. Les équations quadratiques sont aléatoires, mais comportent des constantes calculées de façon garantir l'existence d'une chaîne de valeurs intermédiaires reliant un bloc d'entrée et de sortie fixés à l'avance. Ces résultats laissent entrevoir des perspectives étonnantes qu'il conviendra de vérifier.

7.3.4 Lien entre l'algorithme XL et les bases de Gröbner

Le développement de la "cryptographie multivariée" a conduit à la mise au point, par des cryptologues, d'algorithmes de résolution de systèmes multivariés comme XL, proposé en 2000 par N. Courtois, A. Klimov, J. Patarin et A. Shamir. Il est démontré que, sur un même exemple, l'algorithme XL est toujours moins efficace qu'un calcul de base de Gröbner. Une partie de ces résultats est repris dans l'article Asiacrypt'2004 intitulé «Comparison between XL and Gröbner Basis Algorithms », écrit par G. Ars, J.C. Faugère, H. Imai, M. Kawazoe et M. Sugita). Ces résultats permettent de clarifier l'efficacité des divers algorithmes en cryptographie.

8. HFE

Le chapitre suivant contient une partie des résultats de l'article (Faugère & Joux, 2003) en collaboration avec A. Joux. Parmi les résultats récents on peut citer (Dubois *et al.* , 2006; Joux *et al.* , 2006)

8.1 Introduction

The security of many public key cryptosystems relies on the intractability of some well known mathematical problem (integer factorization, ...). Since solving system of algebraic equations is a difficult problem (NP-complete), it is a good candidate for the design of new public key encryption and signature schemes. HFE (Hidden Fields Equations) is a public key cryptosystem using (multivariate) polynomial operations over finite fields. It has been proposed by Jacques Patarin (Patarin, 1996b) following the ideas of Matsumoto and Imai (Matsumoto & Imai, 1988). It has long been regarded as a very promising cryptosystem because it can be used to produce signatures as short as 128, 100 and even 80 bits.

In (Kipnis & Shamir, 1999) a polynomial time attack on HFE was presented; this method is based on "relinearization" techniques. In (Courtois, 2001b) the complexity of this attack was estimated to be at last $n^{\log^2 d}$ where d is the degree of the (secret) univariate polynomial of HFE. The same attack was improved by Courtois(Courtois, 2001b) to obtain a theoretical complexity of $n^{3\log_2 d+\mathcal{O}(1)}$.

In this paper we present a new and efficient attack of this cryptosystem based on fast algorithms for computing Gröbner basis (Buchberger B., 1965; Buchberger B., 1970; Buchberger B., 1979). The public key of HFE is a list of algebraic equations, and since Gröbner bases is a well known and efficient method for solving polynomial system of equations, our attack is very simple: we compute a Gröbner basis of the public key of HFE. Of course the efficiency of this attack depends strongly on the choice of the algorithm for computing the Gröbner basis. With the best implementation of the Buchberger algorithm (Buchberger B., 1965) only toys examples can be solved ($\approx$ 20 bits). On the other hand, by using the new and efficient F_5 ((Faugère J.C., 2002)) algorithm for computing Gröbner we were able to break the first

HFE challenge (80 bits) in only two days of CPU time on a single processor (Alpha). The goal of this paper is to present a methodology to study experimentally a cryptosystem like HFE with algebraic tools. We made a series of computer simulations on real size HFE problems (up to 160 bits) so that we can establish precisely the complexity of the Gröbner attack: for all practical values of d (less than 512) the complexity is at most $\mathcal{O}(n^{10})$.

In (Courtois *et al.* , 2000), another algorithm (XL) was proposed for solving algebraic systems over finite fields. It is clearly an interesting point to compare the XL and the Gröbner approaches; but the evaluation of the theoretical complexity of such algorithms is difficult. Moreover, as many other algorithms (LLL, the simplex algorithm for solving linear programs, ...), Gröbner bases algorithms behave much better in practice than in the worst case, so considering just the worst-case bounds may lead to underestimate their practical utility. A typical example is precisely the computation of Gröbner bases over $\mathbb{F}_2$, whose asymptotic worst-case time bound is exponential, while its running time is bounded by a low-degree polynomial for HFE. No benchmarks or implementation of the algorithm XL are available so the comparison with XL is out of the scope of this paper and is the subject of another paper.

8.2 Description of HFE

We refer to (Patarin, 1996b) for a complete description of HFE and we describe "the basic HFE" (HFE without variations). We denote by $\mathbb{F}_2$ (resp. $\mathbb{F}_{2^n}$), the finite field of cardinality (2) (resp. 2^n) and characteristic 2. Let

$$f(x) = \sum \beta_{i,j} x^{2^{\theta_{i,j}}+2^{\varphi_{i,j}}} + \sum_k \alpha_k x^{2^{\epsilon_k}} + \mu$$

be a polynomial in x over $\mathbb{F}_{2^n}$ of degree d, for integers $\theta_{i,j}, \varphi_{i,j}, \epsilon_k \geq 0$. In the rest of this paper d is always the degree of the univariate polynomial f.

Since $\mathbb{F}_{2^n}$ is isomorphic to $\mathbb{F}_2[z]/(g(z))$ where $g(z) \in \mathbb{F}_2[z]$ is irreducible of degree n, elements of $\mathbb{F}_{2^n}$ may be represented as n-tuples over $\mathbb{F}_2$, and f may be represented as a polynomial in n variables $x_1, \ldots, x_n$ over $\mathbb{F}_2$:

$$f(x_1, \ldots, x_n) = (q_1(x_1, \ldots, x_n), \cdots, q_n(x_1, \ldots, x_n))$$

with $q_i(x_1, \ldots, x_n) \in \mathbb{F}_2[x_1, \ldots, x_n]$ for $i = 1, 2, \ldots, n$. The q_i are polynomials of *total degree* 2 due to the choice of f and the fact that $x \mapsto x^2$ is a linear function of $\mathbb{F}_{2^n} \longrightarrow \mathbb{F}_{2^n}$.

Let S and T be two $n \times n$ non singular matrices then we can compose S, f and T:

$$S(f(TX)) = (p_1(x_1, \ldots, x_n), \cdots, p_n(x_1, \ldots, x_n))$$

where $\mathbb{F}_2^n$ is regarded as an n-dimensional vector space over $\mathbb{F}_2$ and X is the vector $(x_1, \ldots, x_n)$. Obviously p_i are again quadratic polynomials.

We can now describe the HFE (Hidden Field Equations) public key encryption scheme:

Secret key. The function f , two affine bijections S and T as above.

Public key. Some way of representing $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$. Polynomials p_i for $i = 1, 2, \ldots, n$ as above, computed using the secret key f , S, T.

Encryption. To encrypt the n-tuple $x = (x_1, \ldots, x_n) \in (\mathbb{F}_2)^n$(representing the message), compute the ciphertext

$$y = (p_1(x_1, \ldots, x_n), \cdots, p_n(x_1, \ldots, x_n))$$

Decryption. To decrypt the ciphertext y, first find all solutions z to the univariate equation $f(z) = T^{-1}y$, next compute $S^{-1}z$.

When the polynomial f is a monomial the HFE reduces to cryptosystem of Matsumoto and Imai (Matsumoto & Imai, 1988) broken by Patarin in (Patarin, 1995a). In the following we consider only *random quadratic* polynomial ($\beta_{i,j}$, α_k, μ random in $\mathbb{F}_{2^n}$).

Find the roots of a polynomial of degree d with coefficients in $\mathbb{F}_{2^n}$ can be done (see (von zur Gathen & Gerhard, 1999) for instance) in $\mathcal{O}(\mathbf{M}(d)\log(d))$ operations in $\mathbb{F}_{2^n}$ where $\mathbf{M}(d)$ is the cost of polynomial multiplication. We report the time to find *one* solution of univariate polynomial with NTL(Shoup, 2003) (PC PIII 1000 Mhz):

(n, d)	(80,129)	(80,257)	(80,513)	(128,129)	(128,257)	(128,513)
NTL CPU time (sec)	0.6 s	2.5 s	6.4 s	1.2 s	3. s	9.05 s

From these experimental results we conclude that, in practice, we cannot take arbitrarily big value for the degree of the univariate polynomial (say $d \leq 512$). The recommended values (Patarin, 1996b; Patarin, 1996c) for n are $n \geq 32$ and $n = 80$, $d = 96$ for the first HFE Challenge.

8.3 Complexity of Gröbner bases

A crucial point in the cryptanalysis of HFE is the ability to distinguish a "random" (or generic) algebraic system from an algebraic system coming from HFE. We will establish in section 8.4.1 that this can be done by computing Gröbner bases and comparing the maximal degree occurring in these computations.

First let us recall the asymptotic behavior of the maximal degree occurring in the computation is (see eq **??**):

$$d = \max \text{ total degree} \approx \frac{n}{11.114\ldots}$$

From this result we know that computing Gröbner bases of random systems is simply exponential; consequently, in practice, it is impossible to solve a system of n equations of degree 2 in n variables when n is big (say $n \geq 80$). From a practical point of view it is even more important to have *exact values* (see (Bardet *et al.* , 2003b) and chapter 6) for D and N_D when n is small:

n	15	16	$\cdots$	23	24	$\cdots$	80
degree	4	5	$\cdots$	5	6	$\cdots$	12
nb of rows	1379	8840	$\cdots$	40480	223124	$\cdots$	2^{46}

Table 8.1. Maximal degree occurring in Gröbner for random systems.

For instance when $n = 80$, we read the maximal degree in the table 8.3: $D = 12$ and the size of the matrix is 2^{46} so the total cost is bigger than $2^{46\omega} \geq 2^{92}$. In (Bardet *et al.* , 2003b) we give explicit expressions for N_D in function of D and n; the two following formulas are useful for HFE:

Proposition 8.3.1. *Let S be a system of n random equations in n variables. During the computation of S with F_5 the size of matrix at degree D is:*

Number of rows	*in degree 4*	*in degree 5*
in the matrix	$1/2\, n\left(1+n^2\right)$	$1/6\, n\,(n-1)\,(n-3)\,(n+1)$

8.4 Experimental results

The results of this section are all coming from experiments: we are running Gröbner basis computations for real size HFE problems; then we analyse the results in the light of theoretical results obtained in section **??**.

Let HFE(d, n) be the algebraic system corresponding to the basic HFE problem with $f(x)$ a random (quadratic) polynomial of degree d and random coefficients in the field $\mathbb{F}_{2^n}$.

To generate the system of equations HFE(d, n) we have used two programs: one written by JF Michon using NTL(Shoup, 2003) the other one being written in C by D. Augot. For instance when $d = 16$ and $n = 12$ it takes 1 min 25 sec (PIII 1000 Mhz) to generate the algebraic system and the size of the output file is 13 Mbytes.

From the Gröbner point of view we note that the two affine transform (see section 8.2) S and T are useless: the effect T (resp. S) is equivalent to a

random change of coordinates (resp. to replace the generators of the ideal I by linear combinations). Hence the ideal (the hilbert function) remains unchanged. Of course, without S, T, HFE could be attacked by *other* methods.

As reported in Annex 18.1 the first HFE Challenge was broken using F_5 and only 4Gbytes of RAM. It must be emphasized that this computation is far beyond the capacity of all the other implementations and algorithms for computing Gröbner basis as is made clear by the following table (except a new implementation of the F_4 algorithm in Magma on a super computer with 32 Gbytes of RAM).

8.4.1 HFE algebraic systems are not random

We have collected a lot of experimental data by running thousand of HFE systems for various $d \leq 1024$ and $n \leq 160$. In the following graph, the maximal degree occurring in the Gröbner basis computation of an algebraic system coming from HFE (resp. from a random system as described in table 8.3 section **??**) is plotted:

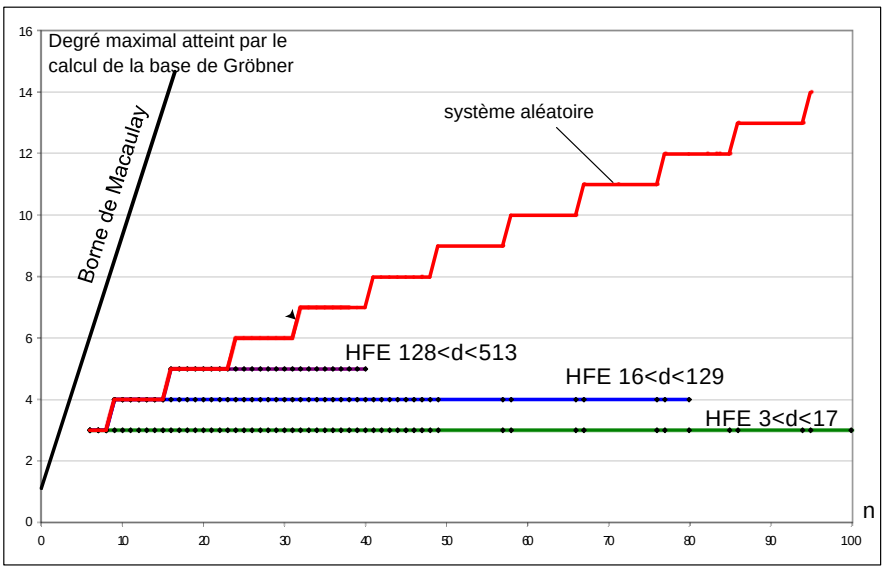


Fig. 8.1. Complexité: courbe théorique et pratique.

The red curve is the maximal degree of a random algebraic system (or more precisely a semi-regular sequence) as explained in chapter 6 and reported in 6.7. As is made clear by this graph, HFE algebraic system are not equivalent to random system from the Gröbner basis point of view.

Note 8.4.1. A common pitfall is to compare an HFE algebraic system and random system for too small values of n. For instance, if we want to *experimentally prove* that the maximal degree occurring in the computation of HFE$(129, n)$ is always less than 5. We read in table 8.3 (section **??**) that we must take $n \geq 24$: in fact when $n < 24$ for all random systems in n variables the computation stops at degree 5. Hence, when $n < 24$ it is impossible to distinguish HFE$(129, n)$ from a random system.

Note 8.4.2. For the first HFE challenge, the difference with a random system can be detected after 6 hours of computation.

From the graph and (Kipnis & Shamir, 1999; Courtois, 2001b) it is natural to conjecture:

Proposition 8.4.1. *The basic HFE problem corresponding a secret polynomial of degree d with coefficients in the field $\mathbb{F}_{2^n}$ can be solved in $\mathcal{O}(n^{\omega D})$ where $D < \log_2(d)$ is the maximal degree occurring in the computation. For practical value of $d < 513$ we have $D \leq 5$. More precisely $D \leq 4$ (resp. $D \leq 3$) when $D \leq 128$ (resp. $D \leq 16$).*

8.4.2 Experimental complexity

We know from proposition 8.4.1 that the complexity for computing a Gröbner basis of HFE(d, n) is polynomial in n (say $\mathcal{O}(n^{k_d})$) when d is fixed but we want to find precisely k_d for practical value of $d < 513$. Consequently we analyse some simulation results. From a practical point of view the complexity could be the running-time but it is a noisy measurement: it depends strongly on the architecture (32 or 64 bits), the size of the various level of cache, the load of the computer, ... Hence we give *also* the total number of arithmetic operations: since the most consuming part is linear algebra over $\mathbb{F}_2$ we give the number of 64 bits xor operations (XOR). This number is the same for all computers and depends only on the linear algebra that we have implemented (in our case standard Gaussian elimination).

First we want to establish that there are only three "class of complexity" when $d < 513$:

C_1 when $4 < d < 17$ all the HFE(d, n) are in roughly equivalent.
C_2 when $16 < d < 129$ all the HFE(d, n) are in roughly equivalent.
C_3 when $128 < d < 513$ all the HFE(d, n) are in roughly equivalent.

For all admissible values of d inside a class C_j we compare $HFE(d, n)$ with a "reference degree": 12 (class C_1), 17 (C_2), 129 (C_3). For instance for the second class C_2 we compare HFE(d, n) with HFE$(17, n)$:

n	28	30	32	40	50	60	64	70	80
XOR(96)/XOR(17)	9.4	9.3	9.1	8.3	7.5	6.9	6.7	6.4	6.1
CPU(96)/CPU(17)	6.5	6.4	6.2	5.7	5.0	4.7	4.9	5.1	4.4

Table 8.2. Comparison: HFE$(96, n)$/HFE$(17, n)$

n	40	50	55	60	64	70	80
XOR(17)/XOR(12)	195.6	307.3	373.6	444.9	508.3	603.2	789.7
CPU(17)/CPU(12)	131.4	262.0	374.3	487.6	701.3	932.4	1505.4

Table 8.3. Comparison: HFE$(17, n)$/HFE$(12, n)$

n	19	22	24	28	30	31	32
XOR(129)/XOR(17)	29.3	50.4	61.6	81.9	93.4	96.9	99.4
CPU(129)/CPU(17)	49.5	100.2	145.5	246.5	329.7	360.7	397.3

Table 8.4. Comparison: HFE$(129, n)$/HFE$(17, n)$

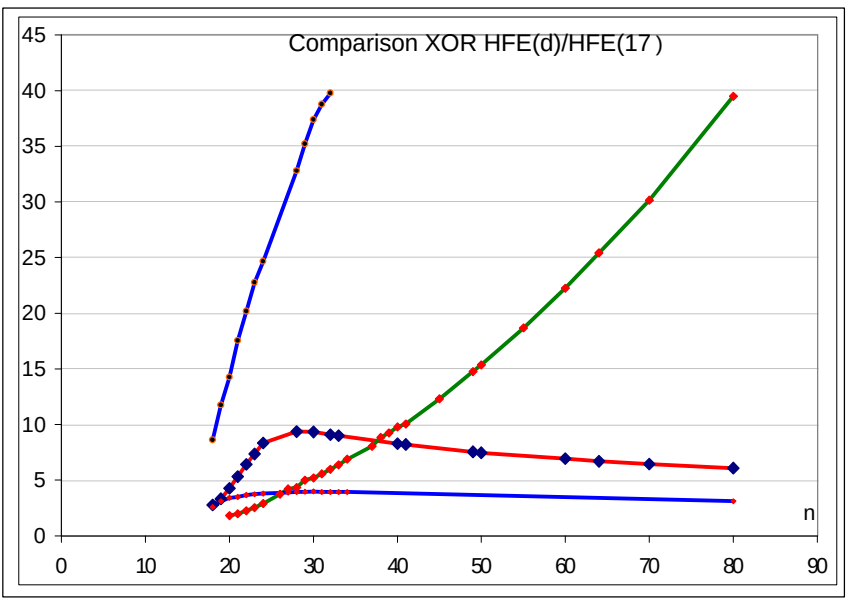


Fig. 8.2. Small dots correspond to a computer simulation.

From this table we conclude that HFE$(96, n)$ is only 6 times more difficult than HFE$(17, n)$ so that the exponents k_{96} and k_{17} are the same. On the other the hand the following results clearly indicates that $k_{12} < k_{17} < k_{129}$:

To sum up:

$$\text{HFE(129,n)} \gg \text{HFE(96,n)} \approx \text{HFE(17,n)} \gg \text{HFE(12,n)}$$

Next we want to find k_{12}, k_{17}, k_{129}. We have used several methods to find the exponent. We begin by a theoretical analysis made on the following hypothesis: suppose that the HFE(d, n) behave like a random system except that the maximal degree occurring in the computation D' is much less than D (given by the bound of section **??**). If this is true we have to solve a linear system whose size is $r \times c$ where r is given by proposition 8.3.1 and c the number of columns is simply the number of monomials in degree D':

$$c = \sum_{i=1}^{D'} \binom{n}{i}.$$

Since the rows of the matrix have the shape $m\, f_i$ and f_i is a polynomial of degree 2, the number of non zero elements in the $r \times c$ matrix is at most NZ$= r\, \frac{n\,(n+1)}{2}$. By applying sparse linear algebra technique (Wiedemann's algorithm, ...), we can find the solution in $\mathcal{O}(r\,\mathrm{NZ}) = \mathcal{O}(r^2 n^2)$ operations. From proposition 8.4.1 we know that $D' \leq 4$ when $d \leq 128$ and in that case $r \approx \frac{n^3}{2}$ (from proposition 8.3.1). Consequently the total cost is $\mathcal{O}(n^8)$. In the same way we found $\mathcal{O}(n^{10})$ (resp. $\mathcal{O}(n^6)$) when $129 \leq d \leq 512$ (resp $d < 17$).

It must be emphasized that the previous computation is not a complexity proof since we cannot check the hypothesis. We must confirmed this results experimentally by doing *real simulations*.

Suppose that the complexity is $f(n) = Cn^k$ then we draw the curve $\log(f(n)) = \log(C) + k\log(n)$ to find the slope of line. We can also try to find a good polynomial approximation (using least squares method) of the curve $f(n)$. For instance, when HFE$(d = 12, n)$ we have collected a set of data $10 < n < 160$ and found:

$$\begin{aligned}
&\tfrac{f(n+1)}{f(n)} \approx 1 + \tfrac{6.129055587}{n}\\
&log(f(n)) \approx 6.086191079\, log(n) - 4.324402957\\
&f(n) \approx -194.0797\, n^3 + 11.1747\, n^4 - .3354\, n^5 + .02212\, n^6\\
&f(n) \approx 224.4411\, n^3 - 15.4212\, n^4 + .2696\, n^5 + .01623\, n^6 + 2.0810^{-5}\, n^7
\end{aligned}$$

Hence it is clear that $k_{12} = 6$. We report in the following tables the result of our simulations. The running-times are given for HP workstation with an alpha EV68 processor at 1000 Mhz. (C_1 and C_2 are constants):

All results presented in the above tables (and similar simulations for other values of d) confirm the validity and the accuracy of the previous estimation.

Theorem 8.4.1. *The complexity of the Gröbner basis compute HFE(d, n) is:*

n	93	94	96	98	100	120	140	160
XOR	$2^{33.6}$	$2^{33.7}$	$2^{33.8}$	$2^{34.0}$	$2^{34.2}$	$2^{35.8}$	$2^{37.1}$	$2^{38.3}$
$C_1 n^6$	$2^{33.6}$	$2^{33.7}$	$2^{33.8}$	$2^{34.0}$	$2^{34.2}$	$2^{35.8}$	$2^{37.1}$	$2^{38.3}$
XOR/$C_1 n^6$	.998	1.000	1.000	1.002	.999	1.001	1.000	1.000
CPU (sec)	$2^{6.2}$	$2^{6.2}$	$2^{6.4}$	$2^{6.6}$	$2^{6.7}$	$2^{8.4}$	$2^{9.8}$	$2^{10.9}$
$C_2 n^6$	$2^{6.1}$	$2^{6.2}$	$2^{6.4}$	$2^{6.6}$	$2^{6.8}$	$2^{8.4}$	$2^{9.8}$	$2^{10.9}$
CPU/$C_2 n^6$	1.047	.996	.984	.992	.944	1.033	.995	1.000

Table 8.5. Comparison between running-times for HFE$(12, n)$ and theoretical $O(n^6)$

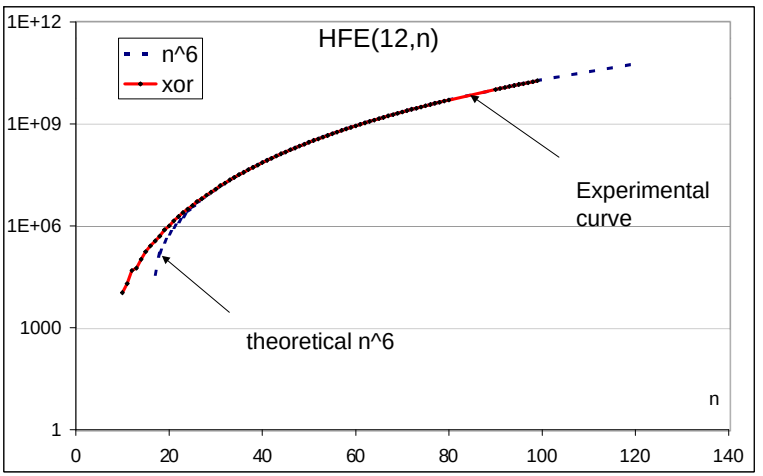


Fig. 8.3. Comparison between running-times for HFE$(12, n)$ and theoretical $\mathcal{O}(n^6)$.

8.5 Patarin original attack revisited

It is interesting to compare the Gröbner bases method with the original attack of Patarin (Patarin, 1995a). We consider the "toy example" ((Matsumoto & Imai, 1988) page 420): the secret key is $f(x) = x^3$, $n = 8$ that is to say the field is $\mathbb{F}_{2^8}$ and the public key is:

n	41	45	50	55	60	64	70	80
XOR	$2^{34.0}$	$2^{35.1}$	$2^{36.4}$	$2^{37.5}$	$2^{38.5}$	$2^{39.3}$	$2^{40.3}$	$2^{41.9}$
$C_1 n^8$	$2^{34.0}$	$2^{35.1}$	$2^{36.4}$	$2^{37.5}$	$2^{38.5}$	$2^{39.3}$	$2^{40.3}$	$2^{41.9}$
XOR/$C_1 n^8$)	1.014	1.003	.999	.999	.997	1.002	1.000	1.000
CPU (sec)	$2^{6.8}$	$2^{7.8}$	$2^{9.2}$	$2^{10.4}$	$2^{11.4}$	$2^{12.5}$	$2^{13.6}$	$2^{15.4}$
$C_2 n^8$	$2^{6.5}$	$2^{7.7}$	$2^{9.1}$	$2^{10.4}$	$2^{11.5}$	$2^{12.4}$	$2^{13.6}$	$2^{15.4}$
CPU/$C_2 n^8$	1.159	1.092	1.026	.991	.927	1.045	.995	1.000

Table 8.6. Comparison between running-times for HFE$(17, n)$ and theoretical $O(n^8)$

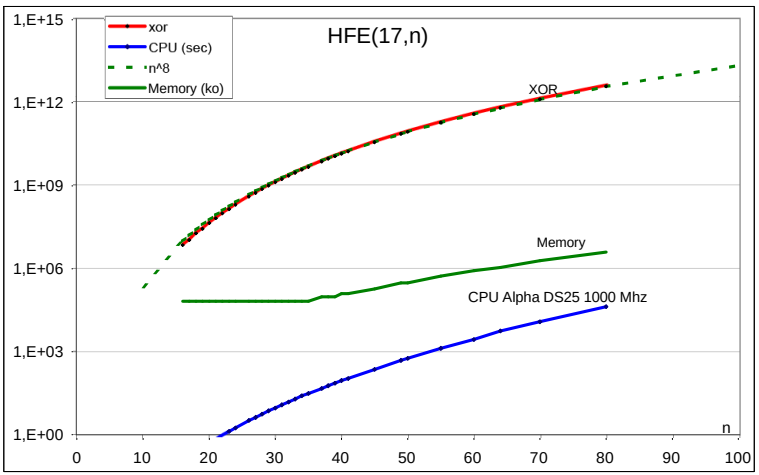


Fig. 8.4. Comparison between running-times for HFE$(17, n)$ and theoretical $\mathcal{O}(n^8)$.

degree of $f(x)$	$d \leq 16$	$17 \leq d \leq 128$	$129 \leq d \leq 512$
Gröbner complexity	$\mathcal{O}(n^6)$	$\mathcal{O}(n^8)$	$\mathcal{O}(n^{10})$

Table 8.7. Comparison between running-times for HFE$(17, n)$ and theoretical $O(n^8)$

$$
\begin{aligned}
&[x_0+x_1+x_3+x_7+x_0\,x_1+x_0\,x_2+x_0\,x_4+x_0\,x_5+x_0\,x_6+x_0\,x_7+x_1\,x_4+x_1\,x_6+x_1\,x_7+x_2\,x_6\\
&\quad +x_3\,x_4+x_3\,x_5+x_3\,x_7+x_4\,x_5+x_5\,x_6+x_5\,x_7+y_0,\\
&x_1+x_2+x_4+x_6+x_0\,x_3+x_0\,x_6+x_0\,x_7+x_1\,x_3+x_1\,x_4+x_1\,x_6+x_2\,x_5+x_2\,x_7+\\
&x_3\,x_4+x_3\,x_7+x_4\,x_6+x_4\,x_7\\
&\quad +x_6\,x_7+y_1,\\
&1+x_0+x_1+x_2+x_3+x_5+x_6+x_0\,x_1+x_0\,x_2+x_0\,x_5+x_1\,x_2+x_1\,x_4+x_1\,x_6+\\
&x_1\,x_7+x_2\,x_6+x_2\,x_7+x_3\,x_5\\
&+x_3\,x_6+x_3\,x_7+x_4\,x_5+x_5\,x_6+x_5\,x_7+y_2,\\
&x_0+x_2+x_3+x_7+x_0\,x_3+x_0\,x_5+x_1\,x_4+x_1\,x_5+x_1\,x_6+x_1\,x_7+x_2\,x_3+x_2\,x_4+\\
&x_2\,x_7+x_3\,x_4+x_3\,x_7+y_3,\\
&1+x_0+x_1+x_2+x_6+x_7+x_0\,x_2+x_0\,x_4+x_0\,x_5+x_1\,x_3+x_1\,x_7+x_2\,x_6+x_3\,x_4+\\
&x_3\,x_5+x_3\,x_6+x_4\,x_5+x_4\,x_6\\
&+x_4\,x_7+x_5\,x_6+x_5\,x_7+x_6\,x_7+y_4,\\
&x_4+x_6+x_0\,x_1+x_0\,x_2+x_0\,x_3+x_1\,x_2+x_1\,x_3+x_1\,x_5+x_2\,x_6+x_3\,x_4+x_3\,x_7+y_5,
\end{aligned}
$$

8.5.1 Patarin's attack

The idea of Patarin (see (Patarin, 1995b) page 12) is to find a low degree relation between x and $y = f(x)$. In our case we have, for instance, $y^5 = (x^3)^5 = x^{15} = x$. This imply that we can find equations of degree 2 (since $5 = 1 + 4$) in y and 1 in x:

$$a_0 + \sum_{i=0}^{7} b_i x_i + \sum_{0 \le i < j \le 7} c_{i,j} y_i y_j + \sum_{i=0}^{7} d_i y_i = 0$$

This give three independent equations:

$$\begin{aligned}
x_0 + x_1 + x_3 + \; & x_4 + x_7 = y_0\, y_2 + y_0\, y_3 + y_2\, y_4 + y_3\, y_4 + y_0\, y_7 \\
& + y_4\, y_7 + y_2 + y_4 + y_5 + y_6 + y_7 \\
x_2 + x_3 = \; & y_2\, y_3 + y_2\, y_4 + y_3\, y_4 + y_2\, y_5 + y_3\, y_5 + y_2\, y_6 + y_3\, y_6 \\
& + y_3\, y_7 + y_4\, y_7 + y_5\, y_7 + y_6\, y_7 + y_0 + y_2 + y_3 + y_5 + y_6 + y_7 \\
x_1 + x_6 + x_7 = \; & y_0\, y_3 + y_0\, y_4 + y_3\, y_4 + y_0\, y_5 + y_4\, y_5 + y_0\, y_6 + y_4\, y_6 \\
& + y_2 + y_3 + y_4 + y_7 + 1
\end{aligned}$$

As a result, from these equations, we can eliminate three variables x_0, x_1 and x_2 in the 8 public equations. We obtain 5 equations of degree 2. We can now find the solution by doing an exhaustive search for 3 variables (x_3, x_4 and x_5).

8.5.2 Generic Gröbner bases in precomputation phase

Proposition **??** and 17.4.1 tell us that a computation of the Gröbner basis (for an appropriate ordering) of the public equations give us the relations among the y_i of *lowest degree*. The Gröbner bases G contains 178 polynomials; among them 24 are of total degree two and linear in x_i (we will denote G' this subset of G). G' contains the three previous linear equations found in (Patarin, 1995b) but also many others; for instance one such equation is:

$$x_3 + x_2 + (x_1 + x_6 + x_7)\,(y_4 + y_3 + y_6 + y_5 + 1) + y_0 + y_4 + 1 = 0$$

Now to find the solution it is enough to substitute the values of y_i (can be done in $\mathcal{O}(n^3)$ operations) and then to solve a linear system (again $\mathcal{O}(n^3)$ operations). So the most costly operation is the computation of G': since G' contains equations of degree 2 this can be done (by linear algebra techniques) in $\mathcal{O}(n^7)$ operations. Note that this step has to be done only once (one precomputation phase for each new public key); hence the complexity is $\mathcal{O}(n^7) + K\mathcal{O}(n^3)$ where K is the number of messages to decipher. This must be compared with the complexity $\mathcal{O}(n^6)$ found in section 8.4.2 (theorem 8.4.1). We conclude that Gröbner bases is useful to find *automatically* all the low degree relations; the drawback is that we cannot find a bound similar to (Patarin, 1995b) for the number of independent equations.

8.6 Conclusion

We have presented a very efficient attack on the basic HFE cryptosystem based on Gröbner bases computation. It is not only a theoretical attack with a good complexity but also a very practical method since our implementation was able to break the first HFE challenge (80 bits). However, several modified versions of HFE have been proposed (Patarin, 1996b; Patarin *et al.* , 1998a). These perturbations (for instance one can simply remove some equations of the public key) are applied to the basic HFE and are expected to make attacks harder. Hence, the SFLASHv2 is the modified version of HFE that has been recommended by European project NESSIE. It is an open issue to evaluate the practical robustness of these modified versions of HFE by using the techniques presented in this chapter.

9. Cryptanalyse IP (Isomorphisme de Polynômes)

Ce travail a été réalisé en collaboration avec Ludovic Perret((Faugère, J.-C. and Perret L., 2005; Faugère & Perret, 2006b; Faugère & Perret, 2006c; Faugère, 2006)).

9.1 Introduction

Multivariate cryptography – which can be roughly defined as the cryptography using polynomials in several variables – offers a relatively wide spectrum of problems that can be used in public-key cryptography. The Isomorphism of Polynomials (IP) lies in this family (Patarin, 1996b). Briefly, this problem consists in recovering a particular transformation between two sets of multivariate polynomials permitting to obtain one set from the other. It originally corresponds to the problem of recovering the secret key of a C^* scheme (Matsumoto & Imai, 1988). Besides, the security of several other schemes is directly based on the practical difficulty of IP, namely the authentication/signature schemes proposed by J. Patarin at Eurocrypt'96 (Patarin, 1996b), and the traitor tracing scheme described by O. Billet and H. Gilbert at Asiacrypt'03 (Billet & Gilbert, 2003). We also mention that IP is in a certain manner related to the security of Sflash (Courtois *et al.* , 2004) – the signature scheme recommended by the European consortium Nessie for low-cost smart cards (Nessie, 2004) – and can be alternatively viewed as the problem of detecting affine equivalence between S-Boxes (Biryukov *et al.* , 2003). All in all, one can consider the hardness of IP as one of the major issues in multivariate cryptography. The goal of this paper is to provide new insights on the theoretical and practical complexity of IP and some of its relevant variants.

9.1.1 Previous Work

To the best of our knowledge, the most significant results concerning IP are presented in (Patarin *et al.* , 1998a), where an upper bound on the theoretical complexity of IP is given. Nevertheless, we point out that the proof provided is actually not complete. Anyway, the upper bound presented in that paper

is original and general. It is indeed based on a group theoretic approach of IP and actually dedicated to "IP-like" problems. A new algorithm for solving IP, called "To and Fro", is also described in (Patarin *et al.* , 1998a). This algorithm is however devoted to special instances of IP, namely the ones corresponding to a public key of C* (Matsumoto & Imai, 1988). Thus, it can not be used for solving generic instances of IP. This is not the case for the algorithm presented here. Besides, we present in Section 9.4 experimental results demonstrating that our algorithm outperforms the "To and Fro" method. Finally, we would like to mention a result due to W. Geiselmann, R. Steinwandt, and T. Beth (Geiselmann *et al.* , 2001). In the context of C*, they showed how to easily recover the affine parts of a solution of IP. A similar property also holds in the context of HFE (Felke, 2005). Such a kind of result does not exist for generic instances of IP. Nevertheless, it means that in the cryptographic context we can focus our attention on the linear variant of IP, called 2PLE here.

9.1.2 Organization of the Paper and Main Results

The paper is organized as follows. We begin in Section 10.2.1 by introducing our notation and defining essential tools of our algorithm, namely varieties and Gröbner bases. A recent algorithm (i.e. F_5 (Faugère J.C., 2002)) for computing these bases is also succinctly described. Finally, we define more formally the Isomorphism of Polynomials (IP) and two of its variants, namely the Isomorphism of Polynomials with one Secret (IP1S) (Patarin, 1996b), and the linear variant of IP that we name 2PLE. In Section 9.3.1, we show that these problems are actually particular instances of a more general problem that we call Polynomial Equivalence (PE). This problem provides a formal definition of an "IP-like" problem. Using classical results of group theory, we conclude this section by providing an upper bound on the theoretical hardness of PE. A new algorithm for solving 2PLE is presented in Section 9.4. The idea is to generate a suitable polynomial system of equations whose zeroes correspond to a solution of IP. In order to construct this system, we also provide some specific properties of 2PLE. From a practical point of view, we used the most recent (and efficient) Gröbner basis algorithm, namely F_5 (Faugère J.C., 2002), for solving this system. It is difficult to obtain a complexity bound really reflecting the practical behavior of the F_5 algorithm. We therefore carried out experimental results illustrating the practical efficiency of our approach. We have indeed broken several challenges proposed in literature (Patarin, 1996b; Patarin, 1996c; Billet & Gilbert, 2003). For instance, we solved a challenge proposed by O. Billet and H. Gilbert at Asiacrypt'03 (Billet & Gilbert, 2003) in less than one second.

9.2 Preliminaries

The notation used throughout this paper is the following. We denote by $\mathbb{F}_q$ the finite field with $q = p^r$ elements (p a prime, and $r \geq 1$), and by $\mathcal{M}_{n,u}(\mathbb{F}_q)$ the set of $n \times u$ matrices whose components are in $\mathbb{F}_q$. As usual, $GL_n(\mathbb{F}_q)$ represents the set of invertible matrices of $\mathcal{M}_{n,n}(\mathbb{F}_q)$, and $AGL_n(\mathbb{F}_q)$ denotes the cartesian product $GL_n(\mathbb{F}_q) \times \mathbb{F}_q^n$. Finally, let $\boldsymbol{x} = (x_1, \ldots, x_n)$, and $\mathbb{F}_q[\boldsymbol{x}] = \mathbb{F}_q[x_1, \ldots, x_n]$, be the polynomial ring in the n indeterminates $x_1, \ldots, x_n$ over $\mathbb{F}_q$. By convention, a boldfaced letter will always refer to a row vector.

9.2.1 Isomorphism of Polynomials and Related Problems

Before defining formally IP, we briefly come back here to the origin of this problem. To do so, we describe the encryption scheme called C* (Matsumoto & Imai, 1988). The public key of this system is a set of multivariate quadratic polynomials $\boldsymbol{b} = (b_1(\boldsymbol{x}), \ldots, b_n(\boldsymbol{x})) \in \mathbb{F}_q[\boldsymbol{x}]^n$. These polynomials are obtained by applying two bijective affine transformations $(S, \mathbf{V})$ and $(U, \mathbf{V})$ of $AGL_n(\mathbb{F}_q)$ to a particular set of polynomials $\boldsymbol{a} = (a_1(\boldsymbol{x}), \ldots, a_n(\boldsymbol{x})) \in \mathbb{F}_q[\boldsymbol{x}]^n$. That is:

$$(b_1(\boldsymbol{x}), \ldots, b_n(\boldsymbol{x})) = (a_1(\boldsymbol{x}S + \mathbf{T}), \ldots, a_n(\boldsymbol{x}S + \mathbf{T}))U + \mathbf{V},$$

denoted $\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}S + \mathbf{T})U + \mathbf{V}$ in the sequel.
To encrypt, we simply evaluate a message $\boldsymbol{m} \in \mathbb{F}_q^n$ on $\boldsymbol{b}$, i.e. $(b_1(\boldsymbol{m}), \ldots, b_n(\boldsymbol{m}))$. To recover the correct plaintext, the legitimate recipient uses the bijectivity of the affine transformations combined with the particular structure of the polynomials of $\boldsymbol{a}$. How these polynomials are constructed is not relevant here. But, due to particular constraints, the polynomials of $\boldsymbol{a}$ are always considered as a public data. The secret key of C* is constituted of $(S, \mathbf{T}), (U, \mathbf{V}) \in AGL_n(\mathbb{F}_q)$.

The first approach for attacking this scheme consists in trying to retrieve the message corresponding to a ciphertext $\boldsymbol{c} \in \mathbb{F}_q^n$, i.e. finding a zero of $\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{c}$. This corresponds to solving a particular instance of the so-called MQ problem, which is NP-Hard in general (Courtois, 2001a; Garey & Johnson, 1979). We emphasize that such a kind of result uniquely guarantees the worst-case hardness and does not provide any information concerning the average-case difficulty. For instance, J.-C. Faugère and A. Joux proposed a polynomial-time algorithm for solving instances of MQ corresponding to the public key of HFE (Faugère & Joux, 2003), which is an extension of C*.
Another approach for breaking C* consists in attempting to recover the affine transformations hiding the structure of $\underline{a}$. That is, extracting the secret key from the public key. This problem, introduced by J. Patarin at Eurocrypt'96 (Patarin, 1996b), is defined as follows:
Isomorphism of Polynomials (IP)
Input: $\boldsymbol{a} = (a_1, \ldots, a_u)$, and $\boldsymbol{b} = (b_1, \ldots, b_u)$ in $\mathbb{F}_q[\boldsymbol{x}]^u$.

Question: Find – if any – $(S, \mathbf{V}) \in AGL_n(\mathbb{F}_q)$ and $(U, \mathbf{V}) \in AGL_u(\mathbb{F}_q)$, s. t.:

$$\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}S + \mathbf{V})U + \mathbf{V}.$$

More precisely, it is usually the linear variant of IP which is considered in practice (Patarin, 1996b; Billet & Gilbert, 2003). That is, when the vectors $\mathbf{T}$ and $\mathbf{V}$ are both equal to the null vector. This problem, that we call 2PLE is the following:
Input: $\boldsymbol{a} = (a_1, \ldots, a_u)$, and $\boldsymbol{b} = (b_1, \ldots, b_u)$ in $\mathbb{F}_q[\boldsymbol{x}]^u$.
Question: Find – if any – $(S, U) \in GL_n(\mathbb{F}_q) \times GL_u(\mathbb{F}_q)$, such that:

$$\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}S)U.$$

However, it is without solving any of the two problems mentioned above that J. Patarin proposed a full cryptanalysis of C* (Patarin, 1995a). This attack uses the very particular structure of the polynomials of $\underline{a}$. This result thus does not then affect at all the practical hardness of IP. The security estimate provided for this problem (Patarin, 1996c) is based on the complexity of the "To and Fro" (TF) algorithm (Patarin *et al.* , 1998a; Patarin *et al.* , 1998b), which is $q^{n/2}$ for quadratic polynomials, and q^n otherwise.

In the rest of this paper, $(\boldsymbol{a} = (a_1, \ldots, a_u), \boldsymbol{b} = (b_1, \ldots, b_u))$ will always denote an element of $\mathbb{F}_q[\boldsymbol{x}]^u \times \mathbb{F}_q[\boldsymbol{x}]^u$. We will always suppose that all the polynomials of $\boldsymbol{a}$ have the same maximal total degree noted D (in the practical applications, we have $2 \leq D \leq 4$). Note that, if $\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}S)U$, for some $(S, U) \in GL_n(\mathbb{F}_q) \times GL_u(\mathbb{F}_q)$, then the polynomials of $\boldsymbol{b}$ must have the same maximal total degree than the ones of $\boldsymbol{a}$, i.e. D.

9.3 A Unified Point of View

9.3.1 Polynomial Equivalence Problems and Group theory

The Isomorphism of Polynomials and 2PLE problems have actually a very similar formulation. An input of these problems is formed of two systems of multivariate polynomials and the question consists in recovering a particular transformation permitting to express one system in function of the other. All transformations have the same characteristic: inducing a group action on $\mathbb{F}_q[\boldsymbol{x}]^u$. Recall that a group $(G, \cdot)$, with identity element e, *acts* on $\mathbb{F}_q[\boldsymbol{x}]^u$ if there exists a map $\phi : G \times \mathbb{F}_q[\boldsymbol{x}]^u \rightarrow \mathbb{F}_q[\boldsymbol{x}]^u$ such that $\phi(e, \boldsymbol{p}) = \boldsymbol{p}, forall \boldsymbol{p} \in \mathbb{F}_q[\boldsymbol{x}]^u$, and:

$$\phi(g, \phi(g', \boldsymbol{p})) = \phi(g \cdot g', \boldsymbol{p}), forall g, g' \in G, and forall \boldsymbol{p} \in \mathbb{F}_q[\boldsymbol{x}]^u.$$

Note 9.3.1. In order to simplify the notations, we will write G instead of $(G, \cdot)$.

For 2PLE, one can then easily check that $GL_n(\mathbb{F}_q) \times GL_u(\mathbb{F}_q)$ acts on $\mathbb{F}_q[\boldsymbol{x}]^u$ through:

$$\begin{array}{rl} \phi_{\text{2PLE}} : GL_n(\mathbb{F}_q) \times GL_u(\mathbb{F}_q) \times \mathbb{F}_q[\boldsymbol{x}]^u \rightarrow & \mathbb{F}_q[\boldsymbol{x}]^u \\ ((S,U),\boldsymbol{a}) \mapsto & \boldsymbol{a}(\boldsymbol{x}S)U \end{array}$$

Similarly for IP, $AGL_n(\mathbb{F}_q) \times AGL_u(\mathbb{F}_q)$ acts on $\mathbb{F}_q[\boldsymbol{x}]^u$ through:

$$\phi_{\text{IP}} : \left(\begin{array}{rl} AGL_n(\mathbb{F}_q) \times AGL_u(\mathbb{F}_q) \times \mathbb{F}_q[\boldsymbol{x}]^u \rightarrow & \mathbb{F}_q[\boldsymbol{x}]^u \\ ((S,\mathbf{T}),(U,\mathbf{V}),\boldsymbol{a}) \mapsto & \boldsymbol{a}(\boldsymbol{x}S+\mathbf{T})U+\mathbf{V} \end{array} \right)$$

This observation naturally leads to the introduction of the following problem. Let $(G,\cdot)$ be a group, and $\phi : G \times \mathbb{F}_q[\boldsymbol{x}]^u \rightarrow \mathbb{F}_q[\boldsymbol{x}]^u$ be an action of G on $\mathbb{F}_q[\boldsymbol{x}]^u$.
Given $(\boldsymbol{a},\boldsymbol{b}) \in \mathbb{F}_q[\boldsymbol{x}]^u \times \mathbb{F}_q[\boldsymbol{x}]^u$, the problem we call *Polynomial Equivalence*, with respect to $(G,\cdot)$ and ϕ – and denoted by $\text{PE}(G,\phi)$ – is the one of finding (if any) $g \in G$, verifying:

$$\boldsymbol{b} = \phi(g,\boldsymbol{a}),$$

denoted $\boldsymbol{a} \equiv_{(G,\phi)} \boldsymbol{b}$ in the sequel. This formulation is very convenient since it procures a unified description of IP and 2PLE. Indeed, $\text{PE}(GL_n(\mathbb{F}_q) \times GL_u(\mathbb{F}_q), \phi_{\text{2PLE}})$=2PLE, and $\text{PE}(AGL_n(\mathbb{F}_q) \times AGL_u(\mathbb{F}_q), \phi_{\text{IP}})$=IP. More generally, PE provides a unified description of "IP-like" problems. In our mind, such a kind of problems consists in recovering a particular transformation between two sets of multivariate polynomials. For instance, the Isomorphism of Polynomials with one Secret (IP1S) – introduced at Eurocrypt'96 by J. Patarin (Patarin, 1996b) – falls into this new formalism. This problem, which can be used to design an authentication (resp. signature) scheme (Patarin, 1996b), is as follows. Given $(\boldsymbol{a},\boldsymbol{b}) \in \mathbb{F}_q[\boldsymbol{x}]^u \times \mathbb{F}_q[\boldsymbol{x}]^u$, find – if any – $(S,\mathbf{T}) \in AGL_n(\mathbb{F}_q)$, such that $\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}S+\mathbf{T})$. Using our formalism, we immediately obtain that $\text{PE}(AGL_n(\mathbb{F}_q), \phi_{\text{IP1S}}) = IP1S$, with $\phi_{\text{IP1S}} : AGL_n(\mathbb{F}_q) \times \mathbb{F}_q[\boldsymbol{x}]^u \rightarrow \mathbb{F}_q[\boldsymbol{x}]^u, ((S,\mathbf{T}),\boldsymbol{a}(\boldsymbol{x})) \mapsto \boldsymbol{a}(\boldsymbol{x}S+\mathbf{T})$. Finally, the following lemma justifies the use of the word equivalence in PE.

Lemma 9.3.1. *Let $(G,\cdot)$ be a group, and $\phi : G \times \mathbb{F}_q[\boldsymbol{x}]^u \rightarrow \mathbb{F}_q[\boldsymbol{x}]^u$ be an action of G on $\mathbb{F}_q[\boldsymbol{x}]^u$. Then, $\equiv_{(G,\phi)}$ is an equivalence relation on $\mathbb{F}_q[\boldsymbol{x}]^u$.*

9.3.2 Action de Groupe

In the Graph Isomorphism context, the introduction of group theory concepts permitted to achieve significant advances from both a theoretical and algorithmic point of view (Hoffman., 1982; Fortin, 1996). The formalism previously given permits to naturally extend these results to Polynomial Equivalence problems.

Definition 9.3.1. *Let $(G, \cdot)$ be a group. We shall call $Aut_{(G,\phi)}(\boldsymbol{a}) = \{g \in G : \phi(g, \boldsymbol{a}) = \boldsymbol{a}\}$, $Aut_{(G,\phi)}(\boldsymbol{b}) = \{g \in G : \phi(g, \boldsymbol{b}) = \boldsymbol{b}\}$, the automorphism groups of a and b w.r.t. (G, ϕ). We shall also set $S_{(G,\phi)}(\boldsymbol{a}, \boldsymbol{b}) = \{g \in G : \boldsymbol{b} = \phi(g, \boldsymbol{a})\}$.*

$Aut_{(G,\phi)}(\boldsymbol{a})$ and $.Aut_{(G,\phi)}(\boldsymbol{b})$ are also known as *stabilizer* of $\boldsymbol{a}$ (resp. $\boldsymbol{b}$) w.r.t. (G, ϕ). However, we will rather call these sets automorphism groups. This designation being indeed more usually used in the Graph Isomorphism context (Hoffman., 1982). Anyway, the results that we are going to expose are classical results of group theory concerning the stabilizers and orbits, and then given without proofs.

Proposition 9.3.1. *Let $(G, \cdot)$ be a group, and $\phi : G \times \mathbb{F}_q[x_1, \ldots, x_n]^u \longrightarrow \mathbb{F}_q[x_1, \ldots, x_n]^u$ be an action of G on $\mathbb{F}_q[x_1, \ldots, x_n]^u$. If there exists $g \in G$, such that $\boldsymbol{b} = \phi(g, \boldsymbol{a})$, then $S_{(G,\phi)}(\boldsymbol{a}, \boldsymbol{b})$ is a left (resp. right) coset – in G – of the automorphism group $Aut_{(G,\phi)}(\boldsymbol{a})$ (resp. $Aut_{(G,\phi)}(\boldsymbol{b})$). That is:*

$$\begin{cases} S_{(G,\phi)}(\boldsymbol{a}, \boldsymbol{b}) = \{g \cdot h : h \in Aut_{(G,\phi)}(\boldsymbol{a})\} = g \cdot Aut_{(G,\phi)}(\boldsymbol{a}), \\ S_{(G,\phi)}(\boldsymbol{a}, \boldsymbol{b}) = \{h \cdot g : h \in Aut_{(G,\phi)}(\boldsymbol{b})\} = Aut_{(G,\phi)}(\boldsymbol{b}) \cdot g. \end{cases}$$

Moreover, the automorphism groups $Aut_{(G,\phi)}(\boldsymbol{a})$ and $Aut_{(G,\phi)}(\boldsymbol{b})$ are conjugate, i.e. $Aut_{(G,\phi)}(\boldsymbol{b}) = g \cdot Aut_{(G,\phi)}(\boldsymbol{a}) \cdot g^{-1}$, and we have:

$$|S_{(G,\phi)}(\boldsymbol{a}, \boldsymbol{b})| = |Aut_{(G,\phi)}(\boldsymbol{b})| = |Aut_{(G,\phi)}(\boldsymbol{a})|.$$

9.3.3 A Generic Upper Bound on the Complexity of "IP-like" Problems

Using the Polynomial Equivalence problem previously defined, we give in this part a general upper bound on the theoretical complexity of "IP-like" problems.

Theorem 9.3.1. *If the polynomial hierarchy does not collapse then IP, 2PLE, and IP1S are not NP-Hard.*

Proof. (see (Faugère & Perret, 2006b))

An "IP-like" problem is then intrinsically not NP-Hard. Furthermore, we believe that our formalism is of independent interest. It indeed procures a general framework for studying "IP-like" problems. However, this is out of the scope of this chapter. We investigate now another aspect of these problems.

9.4 An Algorithm for Solving 2PLE

We study here the practical hardness of a particular Polynomial Equivalence problem, namely 2PLE. Precisely, we present a new algorithm for solving

this problem. We emphasize that – as explained in 10.1.1 – it is usually sufficient to consider this problem rather than its affine variant IP. Besides, any algorithm solving 2PLE can be transformed into an algorithm solving IP (Patarin *et al.* , 1998a; Patarin *et al.* , 1998b).

9.4.1 A First Attempt: Evaluation and Linearization

Instead of directly describing the details of our method, we present the different steps that yielded to this algorithm. Anyway, most of the intermediate results that we are going to present will be used in our final algorithm, but differently. Our earliest idea for solving 2PLE was based on the following remark. If $\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}S)U$, for $(S, U) \in GL_n(\mathbb{F}_q) \times GL_u(\mathbb{F}_q)$, then:

$$\boldsymbol{b}(\boldsymbol{p})U^{-1} = \boldsymbol{a}(\boldsymbol{p}S), \text{ for all } \boldsymbol{p} \in \mathbb{F}_q^n. \tag{9.1}$$

We hence obtain, for each $\boldsymbol{p} \in \mathbb{F}_q^n$, u non-linear equations in the $n^2 + u^2$ components of the matrices S and U^{-1}. We point out that the coefficients of U^{-1} only appear linearly in these equations. This is the advantage of considering the inverse of U rather than simply U in (9.1). The number of equations obtained is then significantly bigger than the number of unknowns. In this situation, one can simply use a linearization method (i.e. associating a new variable to each monomial) for solving the algebraic system. Unfortunately, our experiments rapidly revealed that the equations generated in this way are not all linearly independent. Besides, it also appeared that the number of unknowns is significantly bigger than the number of linearly independent equations. The use of a linearization method is then clearly no longer relevant. Let us explain this phenomenon.

Lemma 9.4.1. *Let* $\boldsymbol{y} = (y_{1,1}, \ldots, y_{1,n}, \ldots, y_{n,1}, \ldots, y_{n,n})$, *and* $\boldsymbol{z} = (z_{1,1}, \ldots, z_{1,u}, \ldots, z_{u,1}, \ldots, z_{u,u})$. *For each* $i, 1 \leq i \leq u$, *there exists a subset* $S_i \subseteq \mathbb{F}_q^n$ *and polynomials* $p_{\alpha,i} \in \mathbb{F}_q[\boldsymbol{y}, \boldsymbol{z}]$, *such that the following equality holds:*

$$(\boldsymbol{b}(\boldsymbol{x})U^{-1} - \boldsymbol{a}(\boldsymbol{x}S))_i = \sum_{\alpha \in S_i} p_{\alpha,i}(S, U^{-1})\boldsymbol{x}^{\alpha}, \tag{9.2}$$

$p_{\alpha,i}(S, U^{-1})$ *being the evaluation of* $p_{\alpha,i}$ *on* $S = \{s_{i,j}\}_{1 \leq i,j \leq n}$, $U^{-1} = \{u'_{i,j}\}_{1 \leq i,j \leq u}$.

Proof. The polynomial $(\boldsymbol{b}(\boldsymbol{x})U^{-1} - \boldsymbol{a}(\boldsymbol{x}S))_i$ can be regarded as an element of:

$$\mathbb{F}_q[s_{1,1}, \ldots, s_{1,n}, \ldots, s_{n,1}, \ldots, s_{n,n}, u'_{1,1}, \ldots, u'_{1,u}, \ldots, u'_{u,u}, \ldots, u'_{u,u}][x_1, \ldots, x_n], \tag{9.3}$$

i.e. a polynomial with unknowns $x_1, \ldots, x_n$ and whose coefficients are polynomials in the components of S and U^{-1}. In this setting, the polynomials $p_{\alpha,i}$ exactly correspond to the coefficients of the monomials (in $x_1, \ldots, x_n$) occurring in $(\boldsymbol{b}(\boldsymbol{x})U^{-1} - \boldsymbol{a}(\boldsymbol{x}S))_i$. Lastly $S_i = \{\alpha \in \mathbb{F}_q^n : p_{\alpha,i} \neq 0\}$.

The cost of generating the polynomials $p_{\alpha,i}$ is proportional to the number of monomials occurring in $(\boldsymbol{b}(\boldsymbol{x})U^{-1} - \boldsymbol{a}(\boldsymbol{x}S))_i$ viewed as a polynomial of (9.3), i.e. $O(n^{2D})$. Note also that each $p_{\alpha,i}$ is by construction the sum of a polynomial in $\mathbf{y}$, plus a linear polynomial in $\mathbf{z}$. Furthermore, the maximal total degree reached by a monomial in the variables $\mathbf{y}$ is equal to D.
From (9.2), we obtain that for all $i, 1 \leq i \leq u$:

$$(\boldsymbol{b}(\mathbf{p})U^{-1} - \boldsymbol{a}(\mathbf{p}S))_i = \sum_{\alpha \in S_i} p_{\alpha,i}(S, U^{-1}) p_1^{\alpha_1} \cdots p_n^{\alpha_n}, forall \mathbf{p} = (p_1, \ldots, p_n) \in \mathbb{F}_q^n.$$

It follows that, for all $\mathbf{p} \in \mathbb{F}_q^n$, the equations procured by (9.1) are linear combinations of the $p_{\alpha,i}(S, U^{-1})$. The number of polynomials $p_{\alpha,i}$ is limited by the number of monomials occuring in $(\boldsymbol{b}(\mathbf{p})U^{-1} - \boldsymbol{a}(\mathbf{p}S))_i$. Thus, $u \cdot C_{n+D}^D$ bounds from above the number of linearly independent equations provided by linearizing (9.1). On the other hand, the number of unknowns in the linearized system is equal to the number of monomials in the variables $\mathbf{y}$ of degree smaller than D, plus the u^2 variables corresponding to $\mathbf{z}$. Using a rough bound, the linearization method yields a linear system of at most $O(u \cdot n^D)$ linearly independent equations with $O(u \cdot n^{2D})$ unknowns.

9.4.2 The 2PLE algorithm

The linearization can thus not be employed for solving efficiently 2PLE. However, Gröbner basis procures another method for solving the algebraic system given by (9.1). From a practical point of view, this approach is quite promising. Indeed, the system obtained by evaluating $\boldsymbol{b}(\boldsymbol{x})U^{-1} = \boldsymbol{a}(\boldsymbol{x}S)$ on several vectors is overdetermined. Nevertheless, all the equations derived from $\boldsymbol{b}(\mathbf{p})U^{-1} = \boldsymbol{a}(\mathbf{p}S)$ are according to (9.2) linear combinations the polynomials $p_{\alpha,i}$. It is hence sufficient to only consider the system formed by these equations. Formally:

Proposition 9.4.1. *Let $\mathcal{I} = \langle p_{\alpha,i} : \text{ for all } i, 1 \leq i \leq u, \text{ and for all } \alpha \in S_i \rangle \subset \mathbb{F}_q[\mathbf{y}, \mathbf{z}]$ be the ideal generated by the polynomials $p_{\alpha,i}$ defined as in Lemma 9.4.1, and $V(\mathcal{I})$ be the following variety:*

$$V(\mathcal{I}) = \{\mathbf{s} \in \mathbb{F}_q^{n^2+u^2} : p_{\alpha,i}(\mathbf{s}) = 0, \text{ for all } i, 1 \leq i \leq u, \text{ and for all } \alpha \in S_i\}.$$

If $\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}S)U$, for some $(S, U) \in GL_n(\mathbb{F}_q) \times GL_u(\mathbb{F}_q)$, then:

$$(\phi_1(S), \phi_2(U^{-1})) \in V(\mathcal{I}),$$

with:

$$\phi_1 : \mathcal{M}_{n,n}(\mathbb{F}_q) \to \mathbb{F}_q^{n^2}, S = \{s_{i,j}\}_{1 \leq i,j \leq n} \mapsto (s_{1,1}, \ldots, s_{1,n}, \ldots, s_{n,1}, \ldots, s_{n,n}), \text{ and}$$
$$\phi_2 : \mathcal{M}_{u,u}(\mathbb{F}_q) \to \mathbb{F}_q^{u^2}, U^{-1} = \{u'_{i,j}\}_{1 \leq i,j \leq u} \mapsto (u'_{1,1}, \ldots, u'_{1,u}, \ldots, u'_{u,1}, \ldots, u'_{u,u}).$$

Proof. For all, $i, 1 \leq i \leq u$:

$$(\boldsymbol{b}(\boldsymbol{x})U^{-1} - \boldsymbol{a}(\boldsymbol{x}S))_i = \sum_{\alpha \in S_i} p_{\alpha,i}(S, U^{-1})\boldsymbol{x}^{\alpha} = 0.$$

Thus, $p_{\alpha,i}(S, U^{-1}) = 0, \forall i, 1 \leq i \leq u$, and $\forall \alpha \in S_i$, i.e. $(\phi_1(S), \phi_2(U^{-1})) \in V(\mathcal{I})$.

In other words, if $\boldsymbol{b} = \boldsymbol{a}(\boldsymbol{x}S)U$, for some $(S, U) \in GL_n(\mathbb{F}_q) \times GL_u(\mathbb{F}_q)$, then the variety $V(\mathcal{I})$ contains the components of the matrices S and U^{-1}. The system associated to $\mathcal{I}$ has $n^2 + u^2$ variables and is of degree D. Once again, we recall that the variables of $\mathbf{z}$ only appear linearly in this system. The number of equations of the system is equal to the number of monomials occurring in the polynomials of $\mathbf{a}$, i.e. $O(u \cdot C_{n+D}^{D})$. The system is then overdetermined.

Note 9.4.1. In order to guarantee that $V(I) \subseteq F_q^{2n}$, we must generally join the field equations to the initial system. The fields considered in our case can be relatively large, leading then to a significant increase of the system's degree. This can artificially render impracticable the computation of a Gröbner basis. Fortunately, our systems are overdetermined and it is not necessary in practice to include the field equations. In our experiments the elements of $V(I)$ were indeed – without including these equations – all the times in F_q^{2n}. It implies in particular that the hardness of 2PLE is not related to the size of the field. This is an important remark since the current security bound for 2PLE depends on this size.

The next proposition is fundamental to understand the practical behaviour of our approach. This result permits furthermore to improve the efficiency of our method.

Proposition 9.4.2. *Let d be a positive integer, and $\mathcal{I}_d \subset \mathbb{F}_q[\mathbf{y}, \mathbf{z}]$ be the ideal generated by the polynomials $p_{\alpha,i}$ of maximal total degree smaller than d. Let also $V(\mathcal{I}_d)$ be the variety associated to $\mathcal{I}_d$. If $\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}S)U$, for some $(S, U) \in GL_n(\mathbb{F}_q) \times GL_u(\mathbb{F}_q)$, then:*

$$(\phi_1(S), \phi_2(U^{-1})) \in V(\mathcal{I}_d), \text{ for all } d, 0 \leq d \leq D,$$

ϕ_1 and ϕ_2 being defined as in proposition 9.4.1.

The proof is obviously deduced from the following result:

Lemma 9.4.2. *Let $(S, U) \in GL_n(\mathbb{F}_q) \times GL_u(\mathbb{F}_q)$. We have:*

$$\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}S)U \iff \boldsymbol{b}^{(d)}(\boldsymbol{x}) = \boldsymbol{a}^{(d)}(\boldsymbol{x}S)U, for all d, 0 \leq d \leq D,$$

$\boldsymbol{b}^{(d)}$ (resp. $\boldsymbol{a}^{(d)}$) being the homogeneous components of degree d (i.e. the sum of the terms of total degree d) of the polynomials of $\boldsymbol{b}$ (resp. $\boldsymbol{a}$).

The systems associated to $\mathcal{I}_1$ and $\mathcal{I}_0$ only contain linear equations in the components of S and U^{-1}. Indeed, let $\mathbf{0_n}$ be the null vector of $\mathbb{F}_q^n$, and $A \in \mathcal{M}_{n,u}(\mathbb{F}_q)$(resp. $B \in \mathcal{M}_{n,u}(\mathbb{F}_q)$) be the matrix representation of $\boldsymbol{a}^{(1)}$ (resp. $\boldsymbol{b}^{(1)}$), i.e. $\boldsymbol{x}A = \boldsymbol{a}^{(1)}(\boldsymbol{x})$ (resp. $\boldsymbol{x}B = \boldsymbol{b}^{(1)}(\boldsymbol{x})$). According to Lemma 9.4.2:

$$\boldsymbol{b} = \boldsymbol{a}(\boldsymbol{x}S)U, for(S,U) \in GL_n(\mathbb{F}_q)\times GL_u(\mathbb{F}_q) \Longrightarrow \begin{cases} \boldsymbol{b}^{(0)}(\mathbf{0_n})U^{-1} = \boldsymbol{a}^{(0)}(\mathbf{0_n}), \\ BU^{-1} = SA. \end{cases}$$

That is, we get linear dependencies between the components S and U^{-1}. More precisely, we obtain $u(n+1)$ linear equations in the n^2+u^2 components of the matrices solution. Anyway, we can not solve 2PLE just by using these equations. On the other hand, it is not necessary to consider the system formed by all the polynomials $p_{\alpha,i}$. According to Proposition 9.4.2, we can actually restrict our attention on $\mathcal{I}_{d_0}$, with d_0 being the smaller integer rendering the system overdetermined. This d_0 can be defined in function of $\boldsymbol{a}$. Indeed, $d_0 \approx \min\{d > 1 : \boldsymbol{a}^{(d)} \neq \underline{0_u}\}$. In practice, it is usually sufficient to take $d_0 = 2$. The hardness of an instance of 2PLE is then related to d_0 rather than to the maximal total degree D of this instance. It is also an important remark since the maximal degree of an instance is taken into account in the security estimate of 2PLE given by J. Patarin (Patarin, 1996b; Patarin, 1996c). Our algorithm for solving this problem is as follows:

Input: $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{F}_q[\boldsymbol{x}]^u \times \mathbb{F}_q[\boldsymbol{x}]^u$
Let $d_0 = \min\{d > 1 : \boldsymbol{a}^{(d)} \neq \mathbf{0_u}\}$
Construct the polynomials $p_{\alpha,i}$ of max. total deg. smaller than d_0
Compute $V(\mathcal{I}_{d_0})$ using the F_5 algorithm
Find an element of $V(\mathcal{I}_{d_0})$ corresponding to a solution of 2PLE
Return this solution

The system associated to $\mathcal{I}_{d_0}$ is overdetermined by its very construction (u^2+n^2 unknowns, and $O(u \cdot C_{n+d_0}^{d_0})$ equations). The variety $V(\mathcal{I}_{d_0})$ is then very likely reduced to a solution of 2PLE (this has been indeed verified in our experiments). The complexity of this algorithm is (theoretically) dominated by the Gröbner basis computation. It is difficult to obtain a complexity bound really reflecting the practical behavior of the F_5 algorithm. We therefore carry out now experimental results illustrating the practical efficiency of our approach.

9.4.3 Experimental Results

We present in this part experimental results obtained with our algorithm. Before that, we provide the conditions of our experiments.

Generation of the instances
We have only considered instances $(\boldsymbol{a}, \boldsymbol{b})$ of 2PLE admitting a solution. We

constructed the instances in the following way:
(1) Choose the polynomials of $\boldsymbol{a}$
(2) Randomly choose $(S, U) \in GL_n(\mathbb{F}_q) \times GL_u(\mathbb{F}_q)$
(3) Return $(\boldsymbol{a}(\boldsymbol{x}), \boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}S)U)$

Precisely, we constructed the polynomials of $\boldsymbol{a}$ in two different ways. The first one simply consists in randomly choosing – w.r.t. a given maximal total degree D – the polynomials of $\boldsymbol{a}$. Precisely, each polynomial is a random linear combination of all the monomials of total degree smaller (or equal) to D. Note that we obtain in this way dense polynomials. We shall call *random instance*, an instance of 2PLE generated in this manner. In the second method, $\boldsymbol{a}$ corresponds to the public key of a C* scheme (Matsumoto & Imai, 1988). An instance of 2PLE generated in this way will be named *C* instance*.

Programming language – Workstation
The experimental results have been obtained with an Opteron bi-processors 2.4 Ghz, with 8 Gb of Ram. The systems associated to an instance of 2PLE have been generated using the Magma software(Cannon J., 1998). We used our own implementation (in language C) of F_5 for computing the Gröbner bases. However, for the sake of comparison, we sometimes used the last version of Magma (i.e. 2.12) for obtaining these bases. This version includes an implementation of the F_4 algorithm.

Table Notations
The following notations are used in the tables below:
– n, the number of variables,
– q, the size of the field,
– deg, the maximal total degree of the considered instance,
– T_{Gen}, the time needed to construct the system,
– T_{F_5}, the time of our algorithm for finding a solution of 2PLE (using the F_5 algorithm for computing the Gröbner bases,
– T, the total time of our algorithm, i.e. $T = T_{F_5} + T_{Gen}$,
– $T_{F_4/Mag}$, the time of our algorithm for recovering a solution of 2PLE, using Magma v. 2.12 for computing Gröbner bases,
– $q^{n/2}$ (resp. q^n), the security bound given in (Patarin *et al.* , 1998a; Patarin *et al.* , 1998b) for instances of $deg = 2$ (resp. $deg > 2$).

Practical Results – Random Instances
We present here the results obtained on random instances of 2PLE. We emphasize that this family of instances is the one employed in the authentication and signature schemes based on 2PLE proposed by J. Patarin at Eurocrypt'96 (Patarin, 1996b; Patarin, 1996c). He suggested to use $u = n$ in practice. Since our main motivation is to study the security of these schemes, we can restrict our attention on the case $u = n$.

Note 9.4.2. Our implementation of F_5 is faster than the Gröbner basis algorithm available in Magma 2.12. For $n = 20$, F_5 is for instance 41 times faster than Magma. To fix ideas, $u = n = 8$, and $u = n = 16$ were two challenges

n	q	deg	T_{Gen}	T_{F_5}	$T_{F_4/Mag}/T_{F_5}$	T	$q^{n/2}$
8	2^{16}	2	0.35 s.	0.14 s.	6	0.49 s.	2^{64}
10	2^{16}	2	1.66 s.	0.63 s.	10	2.29 s.	2^{80}
12	2^{16}	2	7.33 s.	2.16 s.	16	9.49 s.	2^{96}
15	2^{16}	2	48.01 s.	10.9 s.	23	58.91 s.	2^{120}
17	2^{16}	2	137.21 s.	27.95 s.	31	195.16 s.	2^{136}
20	2^{16}	2	569.14 s.	91.54 s.	41	660.68 s.	2^{160}

Table 9.1. Practical Results – Random Instances ($q = F_q$).

n	q	deg	T_{Gen}	T_{F_5}	$T_{F_4/Mag}/T_{F_5}$	T	$q^{n/2}$
10	65521	2	1.21 s.	0.44 s.	10	1.65 s.	$\approx 2^{80}$
15	65521	2	35.58 s.	8.08 s.	23	43.66 s.	$\approx 2^{120}$
20	65521	2	434.96 s.	69.96 s.	41	504.92 s.	$\approx 2^{160}$
23	65521	2	1578.6 s.	235.92 s.		1814 s.	$\approx 2^{184}$

Table 9.2. Practical Results – Random Instances ($q = 65521$).

proposed at Eurocrypt'96 (Patarin, 1996c). We obtained exactly the same results as the ones quoted in the two previous tables for random instances of $deg > 2$. On the other hand, the security estimate for these instances is at least equal to $2^{128} (n = 8)$. The maximal total degree of the systems is indeed the same as for instances of $deg = 2$, i.e. d_0 is equal to 2 independently of D. In other words, increasing the maximal total degree of a random instance will not change its practical hardness. We observe the same behavior for the size of the field, that is increasing q does not really change the hardness of a random instance. This will indeed modify only the cost of the arithmetic operations in the different steps our algorithm.

Interpretation of the results

In all these experiments, the varieties computed were reduced to one element, i.e. the components of the matrices solution of 2PLE. Furthermore, we observe in practice that the complexity of our algorithm is dominated by the time required to construct the system, and not by the Gröbner basis computation. This is surprising, but it clearly highlights that the systems considered here can be easily solved in practice. The generation of the systems being polynomial, we then conclude experimentally that our algorithm solves random instances of 2PLE in polynomial-time. This conclusion is supported by the fact that in all these experiments, the matrices generated by F_5 were of size at most equal to n^3. Experimentally, we deduce a complexity of $(n^3)^3 = n^9$ for our algorithm on random instances of 2PLE.

Practical Results – C* Instances

We now present the results obtained on C* instances $(\boldsymbol{a}, \boldsymbol{b})$ of degree D. We highlight that these instances are used in the traitor tracing scheme described in (Billet & Gilbert, 2003). In this context, we also have $u = n$. The polyno-

mials of $\boldsymbol{a}$ correspond to the public-key of a C* scheme (Matsumoto & Imai, 1988). Precisely, these polynomials are the "multivariate representation" of a univariate monomial (see (Billet & Gilbert, 2003) for details concerning the generation of this multivariate representation). The univariate monomial has the following shape: $m^{1+q^{\theta_1}+q^{\theta_2}+\cdots+q^{\theta_{D-1}}}$ with $\theta_1, \theta_2, \cdots, \theta_{D-1} \in \mathbb{N}^*$.

n	q	deg	T_{Gen}	T_{F_5}	$T_{F_4/Mag}/T_{F_5}$	T	q^n
5	2^{16}	4	0.2 s.	0.13 s.	45	0.33 s.	2^{80}
6	2^{16}	4	0.7 s.	1.03 s.	64	1.73 s.	2^{96}
7	2^{16}	4	1.5 s.	6.15 s.	90	7.65 s.	2^{112}
8	2^{16}	4	3.88 s.	54.34 s.	112	58.22 s.	2^{128}
9	2^{16}	4	5.43 s.	79.85 s.	145	85.28 s.	2^{144}
10	2^{16}	4	12.9 s.	532.33 s.	170	545.23 s.	2^{160}

Table 9.3. Practical Results – C^* instances.

Note 9.4.3. $n = 5$, and $deg = 4$ is the first challenge proposed at Asiacrypt'03 (Billet & Gilbert, 2003). Similarly to random instances, we observed that the size of the field does not really change the practical hardness of the C* instances. We can conclude that it is a general behaviour of 2PLE *instances.*

Interpretation of the results and Future work.
Our algorithm is no longer polynomial for C* instances. The systems obtained for these instances are indeed harder to solve than the random ones. We believe that it is due to the fact that the systems are here sparser. The equality $\boldsymbol{b}(\mathbf{0_n}) = \boldsymbol{a}(\mathbf{0_n})U$ does not provide any information ($\boldsymbol{b}(\mathbf{0_n}) = \boldsymbol{a}(\mathbf{0_n}) = \mathbf{0_n}$ in the C* case). It is not clear yet but it seems that C* instances with $n = 19$ (the second challenge proposed in (Billet & Gilbert, 2003)), can not be solved with our approach.
More generally, we think that $d_0 = \min\{d \geq 0 : \boldsymbol{a}^{(d)} \neq \mathbf{0_u}\}$ provides a relevant measure of the practical hardness of 2PLE instances. It seems actually that this practical difficulty increases in function of d_0. Indeed, for random instances of 2PLE, $d_0 = 0$ and our algorithm solves 2PLE efficiently. For C* instances, $d_{min} = 1$ and there is a change of complexity class. We also checked that the practical complexity increases for homogeneous instances of degree 2, i.e. $d_0 = 2$.
To summarize, for $d_0 = 0$ it is relatively clear that our algorithm solves 2PLE efficiently (likely in polynomial-time). For $d_0 \geq 1$, we conjecture that our algorithm is subexponential in n, and will depend on d_0. This anyway needs further investigations. It is an open problem to precisely determine, as a function of d_0, the asymptotic complexity of our algorithm. It could be possible that techniques presented in chapter 6 provide an answer.

10. Cryptanalyse de 2R

Ce travail a été réalisé en collaboration avec Ludovic Perret ((Faugère & Perret, 2006a)).

10.1 Introduction

More precisely, we propose a new algorithm for solving the Functional Decomposition Problem (FDP). The problem is as follows:

Functional Decomposition Problem (FDP)
Input : multivariate polynomials $h_1, \ldots, h_u$ in $\mathbb{K}[x_1, \ldots, x_n]$.
Find : – if any – non linear multivariate polynomials
$f_1, \ldots, f_u$, and $g_1, \ldots, g_n$, such that:
$(h_1, \ldots, h_u) = (f_1(g_1, \ldots, g_n), \ldots, f_u(g_1, \ldots, g_n))$.

This problem is related to security of $2R^-$ schemes (Patarin & Goubin, 1997a; Patarin & Goubin, 1997b).

10.1.1 Autres attaques.

As stated by E. Biham (Biham, 2000), "the design of this scheme (2R) is unique as it uses techniques from symmetric ciphers in designing public key cryptosystems, while still claiming security based on relation to the difficulty of decomposing compositions of multivariate ... functions". Anyway, the security of 2R schemes has been already carefully investigated (Biham, 2000; Ye *et al.* , 1999; Ye *et al.* , 2001). E. Biham proposed in (Biham, 2000) a successful cryptanalysis of 2R schemes with S-Boxes. This attack exploits the birthday paradox, but can be avoided by increasing the security parameters of 2R schemes (Patarin & Goubin, 1997b). At Crypto'99 (Ye *et al.* , 1999), D.F. Ye, Z.D. Dai, and K.Y. Lam have presented a quite efficient method for solving the Functional Decomposition Problem. The security of 2R schemes is indeed related to this problem. To thwart this last attack, L. Goubin and J. Patarin have proposed (Patarin & Goubin, 1997b) to use a general technique for repairing multivariate schemes, namely keeping secret some polynomials

of the public key. The resulting schemes are called $2R^-$ schemes. Note that V. Carlier, H. Chabanne, and E. Dottax (V. Carlier and H. Chabanne and E. Dottax, 2005) have described a method for protecting the confidentiality of block ciphers design exploiting the principle of $2R^-$ schemes. Usually, the "minus modification" leads to a real strengthening of the considered schemes. For instance, C^* is broken (Patarin, 1995a) while C^{*--} is the basis of Sflash (Courtois *et al.* , 2004), the signature scheme recommended for low-cost smart cards by the European consortium Nessie[1]. Here, we show that $2R^-$ is not more secure than 2R.

10.1.2 Description de l'attaque.

The paper is organized as follows. We begin in Section 10.2.1 by introducing our notations and defining essential tools used in this paper, namely ideals, Gröbner bases, and several operations on ideals (sum, intersection, quotient, ...). Section 10.2.2 gives a brief review of one-round, 2R and $2R^-$ schemes. We also present the Functional Decomposition problem (FDP) in a more formal manner, which is at the basis of the security of 2R and $2R^-$ schemes. An algorithm for solving this problem efficiently would allow to decompose the public key of 2R and $2R^-$ schemes into two independent quadratic systems, making thereby the principle of these cryptosystems useless. In Section 10.3, we present a general algorithm for solving FDP. Our method is inspired on the algorithm of D.F. Ye, Z.D. Dai, and K.Y. Lam (Ye *et al.* , 1999). Note that their algorithm only works for particular instances of FDP, namely when $u = n$, or $u = n - 1$. Briefly, our algorithm works as follows. Let $(h_1, \ldots, h_u) = (f_1(g_1, \ldots, g_n), \ldots, f_u(g_1, \ldots, g_n))$ be an instance of FDP. We first construct the ideal

$$\partial I_h = \left\langle \frac{\partial h_i}{\partial x_j} : 1 \leq i \leq u, 1 \leq j \leq n \right\rangle$$

generated by the partial derivatives of the h_is. We then show that for all

$$i, 1 \leq i \leq n,\ x_n^{d+1} g_i \in \partial I_h$$

, for some $d \geq 0$. In most cases, this allows to recover a basis of the vector space $L(g) = \mathrm{Vect}_{\mathbb{K}}(g_1, \ldots, g_n)$ generated by $g_1, \ldots, g_n$. This is the most difficult part of our algorithm. The f_is being indeed recovered from the knowledge of $L(g)$ by solving a linear system. The complexity of this algorithm depends on the ratio n/u. For example, our algorithm runs in $O(n^{12})$, if $n/u < 1/2$. More generally, we provide a global analysis of the theoretical complexity of our method. As a side effect, we give several insights into the theoretical behavior of the algorithm of D.F. Ye, Z.D. Dai, and K.Y. Lam. We conclude this section by providing experimental results illustrating the efficiency of our

[1] `https://www.cosic.esat.kuleuven.be/nessie/deliverables/decision-final.pdf`.

approach. We have been able to solve in few hours instances of FDP used in 2R$^-$ schemes for most of the challenges proposed in (Patarin & Goubin, 1997b).

10.2 Le crypto-système 2R$^-$

10.2.1 Notations

Throughout this paper, we denote by $\mathbb{K}[x_1, \ldots, x_n]$ the polynomial ring in the n indeterminate $x_1, \ldots, x_n$ over a finite field $\mathbb{K}$ with $q = p^r$ elements (p a prime, and $r \geq 1$). The set of polynomials $p_1, \ldots, p_s$ of $\mathbb{K}[x_1, \ldots, x_n]$ can be regarded as a mapping $\mathbb{K}^n \to \mathbb{K}^s$:

$$(v_1, \ldots, v_n) \mapsto (p_1(v_1, \ldots, v_n), \ldots, p_s(v_1, \ldots, v_n)).$$

We will call these polynomials components. We will also denote by $I = \langle p_1, \ldots, p_s \rangle$ the ideal generated by $p_1, \ldots, p_s$.

10.2.2 Description de 2R

In (Matsumoto & Imai, 1985), T. Matsumoto and H. Imai proposed one of the first examples of PKCs using compositions of multivariate polynomials. The public key of one of them, called C* ((Patarin, 1995a)), represented by "$t \circ \psi \circ s$", where t, s are two secret linear mappings over $\mathbb{F}_{2^n}$, and ψ is the multivariate representation of $\mathbb{F}_{2^n} \to \mathbb{F}_{2^n}, x \mapsto x^{1+2^\theta}$. This scheme has been broken by J. Patarin at Crypto'95 (Patarin, 1995a).
One-round schemes (Patarin & Goubin, 1997a; Patarin & Goubin, 1997b) are generalizations of C*. The public key of these schemes is indeed of the form "$t \circ \psi \circ s$", where t, s are two affine mappings over $\mathbb{K}^n$, and a $\psi : \mathbb{K}^n \to \mathbb{K}^n$ is a bijective mapping given by n multivariate polynomials of degree two. J. Patarin and L. Goubin (Patarin & Goubin, 1997a; Patarin & Goubin, 1997b) propose several constructions for ψ:

- 1. S-box functions: $(a_1, \ldots, a_n) \mapsto$

  $$(S_1(a_1 \ldots, a_{n_1}), S_2(a_{n_1+1} \ldots, a_{n_1+n_2}), \ldots, S_b(a_{n_1+n_2+\cdots+n_{d-1}+1}, \ldots, a_n)),$$

  where $n = \sum_i n_i$, and each $S_i : \mathbb{K}^{n_i} \to \mathbb{K}^{n_i}$ is quadratic.
- 2. Triangular functions:

$$(a_1, \ldots, a_n) \mapsto (a_1, a_2 + q_1(a_1), a_3 + q_2(a_1, a_2, a_3), \ldots, a_n + q_{n-1}(a_1, \ldots, a_{n-1})),$$

  where each q_i is quadratic.
- 3. Combinations of S-box and triangular functions.

They showed that all these constructions are insecure (Patarin & Goubin, 1997a; Patarin & Goubin, 1997b). To circumvent attacks, they introduce two-round schemes whose public key is the composition of two one-round schemes. The secret key of two-round schemes consists of:

- Three affine bijections $r, s, t : \mathbb{K}^n \to \mathbb{K}^n$.
- Two applications $\psi, \phi : \mathbb{K}^n \to \mathbb{K}^n$, given by n quadratic polynomials.

The public key is composed of n polynomials $p_1, \ldots, p_n$ of total degree 4 describing:

$$p = t \circ \psi \circ s \circ \phi \circ r, \mathbb{K}^n \to \mathbb{K}^n.$$

When all the polynomials are given, this scheme is called 2R scheme. If only some of them are given, it is called 2R$^-$ scheme. The public-key part of the computation is merely an application of the mapping p (for encrypting a message, or checking the validity of a signature). For the secret-key computations, we need to invert the mappings ψ and ϕ. The authors then propose to choose the mappings among the constructions $1, 2, 3$ described above and also:

- 4. C* functions: monomials over an extension of degree n over $\mathbb{K}$,
- 5. D* functions (Goubin & Patarin, 1997).

In (Patarin & Goubin, 1997a; Patarin & Goubin, 1997b), it has been proved that when ψ is chosen in the classes 2. and 4., then the resulting 2R scheme is weak. It is not clear that a similar result holds for 2R$^-$ schemes. Anyway, does composing two weak one-round schemes leads to a secure scheme ? The answer is closely related to the difficulty of the following problem:

Functional Decomposition Problem (FDP)
Input : $h = (h_1, \ldots, h_u) \in \mathbb{K}[x_1, \ldots, x_n]^u$.
Find : – if any – $f = (f_1, \ldots, f_u) \neq h \in \mathbb{K}[x_1, \ldots, x_n]^u$, and $g = (g_1, \ldots, g_n) \in \mathbb{K}[x_1, \ldots, x_n]^n$, such that:

$$(h_1(x), \ldots, h_u(x)) = (f_1(g_1(x), \ldots, g_n(x)), \ldots, f_u(g_1(x), \ldots, g_n(x))),$$

noted $h(x) = (f \circ g)(x)$ hereafter, where $x = (x_1, \ldots, x_n)$.
During the last years several results have been obtained on the univariate polynomial decomposition area (Kozen & Landau, 1989; Von zur Gathen, 1990; Von Zur Gathen, 1990). However, multivariate decomposition problem has not been studied so much. Particular instances (multi-univariate,...) of FDP have been investigated in (Von zur Gathen, 1990; Gutierrez *et al.* , 2003). In (Dickerson, 1989), M. Dickerson provided several insights into the theoretical complexity of FDP. However, this kind of results solely guarantees the difficulty of the worst-case. In the cryptographic context, D.F. Ye, Z.D.

Dai, and K.Y. Lam presented in (Ye *et al.* , 1999; Ye *et al.* , 2001) a quite efficient method for solving instances of FDP used in 2R. Note that their method only works when $u = n$, or $u = n - 1$ (Ye *et al.* , 2001). To the best of our knowledge, there exists no previously known algorithm for solving FDP when $u < n - 1$. An efficient method for solving FDP in this case would permit to decompose $2R^-$ schemes into two independent schemes given by quadratic polynomials. To break these schemes, we then would only have to solve two quadratic systems. As mentioned by J. Patarin and L. Goubin (Patarin & Goubin, 1997b), this would make the principle of two-round schemes, including $2R^-$, useless.

10.3 Un algorithme efficace pour résoudre FDP

In this part, we present a new algorithm for solving FDP. Our approach is inspired on the works of D.F. Ye, Z.D. Dai, and K.Y. Lam (Ye *et al.* , 1999; Ye *et al.* , 2001). In order to restrict our attention to homogeneous instances of FDP (Ye *et al.* , 1999) we use the homogenization of a polynomial $p \in \mathbb{K}[x_1, \ldots, x_n]$, denoted p^h, is defined by $p^h(x_0, x_1, \ldots, x_n) = x_0^{deg(p)} p(x_1/x_0, \ldots, x_n/x_0)$, where x_0 is a new variable (see also definition 2.6.3). For any mapping $f : \mathbb{K}^n \to \mathbb{K}^u$, given by the polynomials $f_1, \ldots, f_u$, we define its homogenization by $f^h = (x_0^{deg(f)}, f_1^h, \ldots, f_n^h)$. The dehomogenization of f^h is then $f = (f_1^h(1, x_1, \ldots, x_n), \ldots, f_n^h(1, x_1, \ldots, x_n))$. We have:

Lemma 10.3.1. *(Ye* et al. *, 1999) Let $f : \mathbb{K}^n \to \mathbb{K}^u$ and $g : \mathbb{K}^n \to \mathbb{K}^n$ be two mappings, then:*

$$(f \circ g)^h = f^h \circ g^h.$$

Note 10.3.1. In (Ye *et al.* , 1999), it is stated that this lemma is correct only if $deg(f) \cdot deg(g) > |\mathbb{K}|$. We no longer need this condition over $\mathbb{K}[x_1, \ldots, x_n]$.

Thus, if we can decompose $h^h = f^h \circ g^h$, then a decomposition of $h = f \circ g$ is simply obtained by dehomogenization of f^h and g^h(Ye *et al.* , 1999). Now, we assume that $f : \mathbb{K}^n \to \mathbb{K}^u$ and $g : \mathbb{K}^n \to \mathbb{K}^n$ are two homogeneous functions of degree two. Finally, let $h = f \circ g$, and $\{h_i\}_{1 \leq i \leq u}, \{f_i\}_{1 \leq i \leq u}, \{g_i\}_{1 \leq i \leq n}$ be the components of h, f, g respectively.

10.3.1 Structure de l'idéal différentiel

The aim of our algorithm is to find the vector space $L(g) = \mathrm{Vect}_{\mathbb{K}}(g_1, \ldots, g_n)$ generated by $g_1, \ldots, g_n$. More precisely, this vector space will be recovered from a DRL Gröbner basis of a suitable ideal. Note that the knowledge of $L(g)$ is sufficient for decomposing h. Indeed, any bijective linear combination A of the g_is leads to a decomposition of h since:

$$h = (f \circ A^{-1}) \circ (A \circ g).$$

Let us first assume that we know the vector space $\mathcal{L}(g)$. we try to find f_i and g_i with the following shape:

$$f_i(x_1, \ldots, x_n) = \sum_{1 \le k,\ell \le n} f_{k,\ell}^{(i)} x_k x_\ell \in \mathbb{K}[x_1, \ldots, x_n], \text{ with } f_{k,\ell}^{(i)} \in \mathbb{K} \text{ for all } i, 1 \le i \le u$$
$$g_i(x_1, \ldots, x_n) = \sum_{1 \le k,\ell \le n} g_{k,\ell}^{(i)} x_k x_\ell \in \mathbb{K}[x_1, \ldots, x_n], \text{ with } g_{k,\ell}^{(i)} \in \mathbb{K} \text{ for all } i, 1 \le i \le n.$$

Therefore, for all $i, 1 \le i \le u$:

$$h_i(x_1, \ldots, x_n) = f_i(g_1, \ldots, g_n) = \sum_{1 \le k,\ell \le n} f_{k,\ell}^{(i)} g_k g_\ell. \tag{10.1}$$

By comparing the coefficients in the right-most and left-most parts of these equalities, we obtain a linear system of $O\left(u\binom{n+2}{2}\right)$ equations in the $u\binom{n+2}{2}$ unknown coefficients of the f_is. It seems difficult to rigorously evaluate the rank of this linear system, a question that has been avoided in the previous works on FDP (Ye *et al.* , 1999; Ye *et al.* , 2001). However, it is very likely that this linear system is of full rank when the f_is are dense polynomials. For the instances of FDP used in $2R^-$ schemes, we experimentally only obtain linear systems of full rank. The difficult part is actually to determine the vector space $\mathcal{L}(g)$. For this, we observe that:

$$\frac{\partial h_i}{\partial x_j} = \sum_{1 \le k,\ell \le n} f_{k,\ell} \left(\frac{\partial g_k}{\partial x_j} g_\ell + g_k \frac{\partial g_\ell}{\partial x_j} \right). \tag{10.2}$$

The polynomials $g_1, \ldots, g_n$ being of degree two, their partial derivatives are of degree one. Hence:

$$\partial \mathcal{I}_h = \left\langle \frac{\partial h_i}{\partial x_j} : 1 \le i \le u, 1 \le j \le n \right\rangle \subseteq \langle x_k g_\ell \rangle_{1 \le k,\ell \le n} = \mathcal{V}.$$

This ideal ∂I_h usually provides enough information for recovering the polynomials $g_1, \ldots, g_n$.

Theorem 10.3.1. *Let $M(d)$ be the set of monomials of degree $d \ge 0$ in $x_1, \ldots, x_n$, and*

$$\tilde{V}_d = \mathrm{Vect}_{\mathbb{K}} \left(\left\{ m \frac{\partial h_i}{\partial x_j} : m \in M(d), 1 \le i \le u, \text{ and } 1 \le j \le n \right\} \right).$$

If $\dim(\tilde{V}_d) \ge n|M(d+1)|$, *then, for all* $i, 1 \le i \le n$:

$$x_n^{d+1} g_i \in \partial \mathcal{I}_h,$$

where $\dim(\tilde{V}_d)$ *is the dimension of* $\tilde{V}_d$ *as a vector space over* $\mathbb{K}$.

Proof. We first study the case $d = 0$. Let $\tilde{V} = \tilde{V}_0$ be the linear space generated by the partial derivatives of the h_is, i.e.:

$$\tilde{V}_0 = \tilde{V} = \mathrm{Vect}_{\mathbb{K}} \left(\left\{ \frac{\partial h_i}{\partial x_j} \right\}_{1 \le j \le n}^{1 \le i \le u} \right) \subset \partial \mathcal{I}_h.$$

According to (10.2), each element of $\tilde{V}$ can be written as a sum of $\{x_k g_\ell\}_{1 \le k, \ell \le n}$. Now let $A_{\tilde{V}} \in M_{n^2 \times n^2}(\mathbb{K})$ be the following matrix:

$$A_{\tilde{V}} = \begin{array}{c|cccccccccc} & \frac{\partial h_1}{\partial x 1} & \cdots & \frac{\partial h_1}{\partial x 1} & \cdots & \frac{\partial h_i}{\partial x_j} & \cdots & \frac{\partial h_u}{\partial x_1} & \cdots & \frac{\partial h_u}{\partial x_n} & \\ x_1\, g_1 & & & & & \cdots & & & & & \\ \vdots & & & & & \cdots & & & & & \\ x_n\, g_1 & & & & & \cdots & & & & & \\ \vdots & & & & & \cdots & & & & & \\ x_k\, g_l & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\ \vdots & & & & & \cdots & & & & & \\ x_1\, g_1 & & & & & \cdots & & & & & \\ \vdots & & & & & \cdots & & & & & \\ x_n\, g_n & & & & & \cdots & & & & & \end{array}$$

One can see at once that the $x_n g_i$s lie in $\tilde{V}$ if the number of linearly independent rows of this matrix is at least equal to its number of columns. That is, $x_n g_i \in \partial I_h$, for all $i, 1 \le i \le n$, if:

$$\dim(\tilde{V}) \ge n|M(1)| = n^2.$$

Observe that $\dim(\tilde{V})$ is upper-bounded by $u\,n$. Thus, $\dim(\tilde{V}) \ge n|M(1)| = n^2$ only holds if $u = n$. This explains why the method proposed in (Ye *et al.* , 1999; Ye *et al.* , 2001) is limited to 2R schemes. To circumvent this problem, we have to consider a vector space of higher dimension. This is the motivation for considering:

$$\tilde{V}_d = \mathrm{Vect}_{\mathbb{K}} \left(\left\{ m \frac{\partial h_i}{\partial x_j} : m \in M(d), 1 \le i \le u, \text{ and } 1 \le j \le n \right\} \right).$$

From (10.2), we deduce that each polynomial of $\tilde{V}_d$ can be written as a sum of elements of:

$$V_d = \mathrm{Vect}_{\mathbb{K}} \left(m g_k : m \in M(d+1), \text{ and } 1 \le k \le n \right).$$

We consider $A_{\tilde{V}_d}$ the following matrix:

$$A_{\tilde{V}_d} = \begin{array}{c|ccccccc} & m_{1,1}\frac{\partial h_1}{\partial x1} & \cdots & m_{i,j}\frac{\partial h_i}{\partial x_j} & \cdots & m_{u,n}\frac{\partial h_u}{\partial x_n} \\ \vdots & & & \cdots & & \\ x_n\, g_1 & & & \cdots & & \\ \vdots & & & & & \\ & & & \cdots & & \\ m'\, g_k & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & & & \cdots & & \end{array}$$

Thus, $x_n^{d+1} g_i \in \tilde{V} \subset \partial I_h$, for all $i, 1 \leq i \leq n$, if $\dim(\tilde{V}_d)$ is at least equal to the number of columns of $A_{\tilde{V}_d}$. That is, if $\dim(\tilde{V}_d) \geq n|M(d+1)|$.

Note 10.3.2. At the end of this part, we will provide an explicit value of d in function of the ratio n/u.

According to Theorem 10.3.1, the polynomials g_is are contained, up to some power of x_n, in ∂I_h. Therefore, the quotient of this ideal by a suitable power of x_n contains the polynomials $g_1, \ldots, g_n$.

Corollary 10.3.1. *Using the same notations as in Theorem 10.3.1. If* $\dim(\tilde{V}_d) \geq n|M(d+1)|$*, then:*

$$\mathcal{L}(g) \subset \langle g_1, \ldots, g_n \rangle \subseteq \partial \mathcal{I}_h : (x_n^{d+1}).$$

Proof. The proof of this corollary is obviously deduced from Theorem 10.3.1, and very definition of the quotient. □

Thus each element of $L(g)$ is included in $\partial I_h : (x_n^{d+1})$. Let then G be a (reduced) DRL Gröbner basis of this ideal. It is then natural to consider the set $B_g = \mathrm{Vect}_{\mathbb{K}}(g \in G : deg(g) = 2)$, since according to Theorem 2.9.2:

$$\mathcal{L}(g) = B_g, \text{if } \#B_g = n, \text{ and } \min(\deg(g) : g \in G) = 2.$$

If these conditions are not fulfilled, then one can not recover efficiently $L(g)$ from B_g. Observe that the condition $\#B_g = n$ implies that there exists a unique decomposition (up to bijective linear combinations). To get away with these problems, we can apply several heuristics such as computing $\partial I_h : (x_1^{d+1}), \ldots, \partial I_h : (x_{n-1}^{d+1})$. In practice, it has been always sufficient to compute $\partial I_h : (x_n^d)$, for a suitable d (i.e. $\dim(\tilde{V}_d) \geq n|M(d+1)|$).

10.3.2 Description de l'algorithme

We describe now our algorithm for general instances of FDP, i.e. we no longer suppose here that h is given by homogeneous polynomials.

Algorithme 35 FDP

Input: $h = f \circ g : \mathbb{K}^n \to \mathbb{K}^u$, *donné par u polynômes* $h_1, \ldots, h_u \in \mathbb{K}[x_1, \ldots, x_n]$ *de degré* 4
Output : $f'_1, \ldots, f'_u, g'_1, \ldots, g'_n$, *tels que:* $(h_1, \ldots, h_u) = (f'_1(g'_1, \ldots, g'_n), \ldots, f'_u(g'_1, \ldots, g'_n))$
$h_0^h(x_0, x_1, \ldots, x_n) \leftarrow x_0^4$
$h_i^h(x_0, x_1, \ldots, x_n) \leftarrow x_0^4 h_i(x_1/x_0, \ldots, x_n/x_0)$, *pour tout* $i, 1 \leq i \leq u$
$\partial\mathcal{I}_h^h \leftarrow \left\langle \frac{\partial h_i^h}{\partial x_j} : 0 \leq i \leq u, 0 \leq j \leq n \right\rangle$
Soit d le plus petit entier tel que $\dim(\tilde{V}_d^h) \geq n|M(d+1)|$, *avec:*
$\tilde{V}_d^h = \mathrm{Vect}_{\mathbb{K}}\left(\left\{m\frac{\partial h_i^h}{\partial x_j} : m \in M(d), 0 \leq i \leq u, \text{ and } 0 \leq j \leq n\right\}\right)$.
Calculer une 2*-DRL base de Gröbner G de* $\partial\mathcal{I}_h^h : (x_n^{d+1})$
$B_{g^h} \leftarrow \{g^h \in G, deg(g^h) = 2\}$
if $(\#B_{g^h} \neq n+1)$ ***or*** $(\min(\deg(g) : g \in G) \neq 2)$ ***then***
  return `Fail`
Résoudre le système linéaire (10.1)
On obtient une base B_{f^h} *de* $\mathrm{Vect}_{\mathbb{K}}(f^h)$
return $\{g^h(1, x_1, \ldots, x_n) \in B_{g^h}\}$ *et* $\{f^h(1, x_1, \ldots, x_n) \in B_{f^h}\}$

Note 10.3.3. In practice, our algorithm never returned Fail for instances of FDP used in $2R^-$.

Theorem 10.3.2. *Let* $g_0^h(x_0, x_1, \ldots, x_n) = x_0^2$, *and* $g_i^h(x_0, x_1, \ldots, x_n) = x_0^2 g_i(x_1/x_0, \ldots, x_n/x_0)$, *for all* $i, 1 \leq i \leq n$. *Moreover, let* $M(d)$ *be the set of monomials of degree* $d \geq 0$ *in* $x_0, x_1, \ldots, x_n$, *and*

$$V_d^h = \mathrm{Vect}_{\mathbb{K}}\left(mg_k^h : m \in M(d+1), \text{ and } 0 \leq k \leq n\right),$$
$$\tilde{V}_d^h = \mathrm{Vect}_{\mathbb{K}}\left(\left\{m\frac{\partial h_i^h}{\partial x_j} : m \in M(d), 0 \leq i \leq u, \text{ and } 0 \leq j \leq n\right\}\right).$$

Algo$_{\mathrm{FDP}}$ *returns a solution of FDP (and not Fail) in:*

$$O(n^{3(d+3)}),$$

where d is the smallest integer such that $\dim(\tilde{V}_d^h) \geq (n+1)|M(d+1)|$.

Proof. Let us suppose that our algorithm returns a solution (and not Fail). According to Corollary 10.3.1, we know that for all $i, 0 \leq i \leq n, g_i^h \in \partial I_h^h : (x_n^{d+1})$. The complexity of Algo$_{\mathrm{FDP}}$ is then dominated by the cost of computing a reduced DRL Gröbner basis G of $\partial I_h^h : (x_n^{d+1})$. This step can be done as explained in Section 10.2.1. However, an alternative method can be used in this particular situation. This is due to the particular role of x_n in a DRL order. From Lemma 2.3.2, we know that if x_n^{d+1} divides the leading monomial of a polynomial, then it also divides the entire polynomial. Thus, we can restrict our attention to polynomials of a DRL Gröbner Bases G' of ∂I_h^h whose leading monomials contain x_n^{d+1}. One can see directly that:

$$(g \in G : \deg(g) = 2) = \left(\frac{g'}{x_n^{d+1}} \mid g' \in G' \text{ and } x_n^{d+1} \text{ divise } \mathrm{LT}(g', \prec_{DRL}) \right).$$

More precisely, it is sufficient to compute a reduced $(d+3)$-DRL Gröbner basis of ∂I_h^h. According to Appendix A, this can be done with the F_5 algorithm in $O(n^{3(d+3)})$. From a practical point of view, the two methods proposed for computing G are similar. But the last one is more suitable for evaluating the complexity.

10.3.3 Comparison avec les autres méthodes

In short, our method can be viewed as a generalization of the approach of D.F. Ye, Z.D. Dai, and K.Y. Lam (Ye *et al.* , 1999; Ye *et al.* , 2001). When $u = n$, it is sufficient to consider the ideal $\partial I_h^h : (x_n^1)$ for recovering $L(g)$. This is a simplified description of the method described in (Ye *et al.* , 1999; Ye *et al.* , 2001). When $u < n$, $\partial I_h^h : (x_n^1)$ no longer provides enough information for recovering $L(g)$. To overcome this difficulty, we proposed here to consider ideals of the form $\partial I_h^h : (x_n^{d+1})$. We then proved that $L(g)$ is contained in this ideal as soon as d is sufficiently large.

It is important to know the exact value of the parameter d. This value can be lower-bounded in fonction of the ratio n/u. For this, we observe that $(n+1)|M(d+1)| = (n+1)\binom{n+1+d}{d+1}$ and $\dim(\tilde{V}_d^h)$ is very likely to be equal $(u+1)(n+1)\binom{n+d}{d}$. We then obtain that d should verify:

$$d \geq \frac{n}{u} - 1.$$

For instance, if the number of equations removed (i.e. $n-u$) is smaller than $\lfloor \frac{n}{2} \rfloor$, this yields a complexity of $O(n^{12})$, and $O(n^9)$ if $u = n$. We will show now that this approximation is perfectly coherent with our experimental results.

10.3.4 Résultats expérimentaux

Generation of the instances
We have only considered instances $h = f \circ g$ of FDP admitting a solution. We constructed these instances in the following way:
– $f = t \circ \psi \circ s$ and $g = \phi \circ r$, with $r, s, t \circ \psi \circ s : \mathbb{K}^n \to \mathbb{K}^n$ are random affine bijections, and $\psi, \phi : \mathbb{K}^n \to \mathbb{K}^n$ are S-box functions contructed as explained in Section 10.2.2. We then remove $r \geq 0$ polynomials of h.

Programming language – Workstation
The experimental results have been obtained with a Xeon bi-processor 3.2 Ghz, with 6 Gb of Ram. The instances of FDP have been generated using the Maple software. We used our own implementation (in language C) of F_5 for computing truncated Gröbner bases.

Table Notations

The following notations are used in the table below:
– n, the number of variables,
– b, the number of blocks (as defined in Section 10.2.2),
– n_i, the number of variables in each block (see Section 10.2.2),
– q, the size of the field,
– r, the number of polynomials removed,
– $d_{theo} = \lceil \frac{n}{u} - 1 \rceil$, the predicted (see 10.3.2) value of d for which Algo_{FDP} returns a solution
– d_{real}, the real value of d for which Algo_{FDP} returns a solution
– T, the total time taken by our algorithm,
– $\sqrt{q^n}$, the current security bound (Patarin & Goubin, 1997b; Biham, 2000) for $2R^-$ schemes.

Practical Results

Let us now present results obtained with our algorithm.

n	b	n_i	r	q	d_{theo}	d_{real}	T	$\sqrt{q^n}$
8	4	2	0	65521	0	0	0.0 s.	
8	4	2	4	65521	1	1	0.0 s.	$\approx 2^{64}$
8	4	2	5	65521	2	2	0.3 s.	$\approx 2^{64}$
8	4	2	6	65521	3	3	1.9 s.	$\approx 2^{64}$
10	5	2	5	65521	1	1	0.2 s.	$\approx 2^{80}$
10	5	2	6	65521	2	2	3.2 s.	$\approx 2^{80}$
10	5	2	7	65521	3	3	21.4 s.	$\approx 2^{80}$
10	5	2	8	65521	4	4	180.8 s.	$\approx 2^{80}$
12	3	4	0	65521	1	1	0.1 s.	
12	3	4	5	65521	1	1	0.9 s.	$\approx 2^{96}$
12	3	4	6	65521	1	1	0.9 s.	$\approx 2^{96}$
12	3	4	7	65521	2	2	20.5 s.	$\approx 2^{96}$
12	3	4	8	65521	2	2	25.2 s.	$\approx 2^{96}$
12	3	4	9	65521	3	3	414 s.	$\approx 2^{96}$
20	5	4	0	65521	0	0	1.6 s.	
20	5	4	5	65521	1	1	55.2 s.	$\approx 2^{160}$
20	5	4	10	65521	1	1	78.9 s.	$\approx 2^{160}$
20	10	2	10	65521	1	1	78.8 s.	$\approx 2^{160}$
20	2	10	10	65521	1	1	78.7 s.	$\approx 2^{160}$
24	6	4	0	65521	0	0	4.9 s.	
24	6	4	12	65521	1	1	376.1 s.	$\approx 2^{192}$
30	15	2	15	65521	1	1	2910.5 s.	$\approx 2^{160}$
32	8	4	0	65521	0	0	31.3 s.	
32	8	4	10	65521	1	1	3287.9 s.	$\approx 2^{256}$
32	8	4	16	65521	1	1	4667.9 s.	$\approx 2^{256}$
36	18	2	15	65521	1	1	13427.4 s.	$\approx 2^{256}$

Interpretation of the results

Let us mention that $n = 16$ and $n = 32$ were two challenges proposed by the designers of 2R$^-$ schemes. First it should be observed that the parameters b and n_i of the S-box functions seem irrelevant for the complexity of our algorithm. We also tested our algorithm for instances of FDP constructed with various forms of ψ, ϕ (C^h+S-Box functions, Triangular+S-Box functions,...) and several values of q. These results are very similar to the ones obtained for S-Box functions, and thus not quoted here. The major observation is that our algorithm behaves exactly as predicted. That is, $d_{theo} = \lceil \frac{n}{u} - 1 \rceil$ is exactly equal to the d_{real} observed in practice.

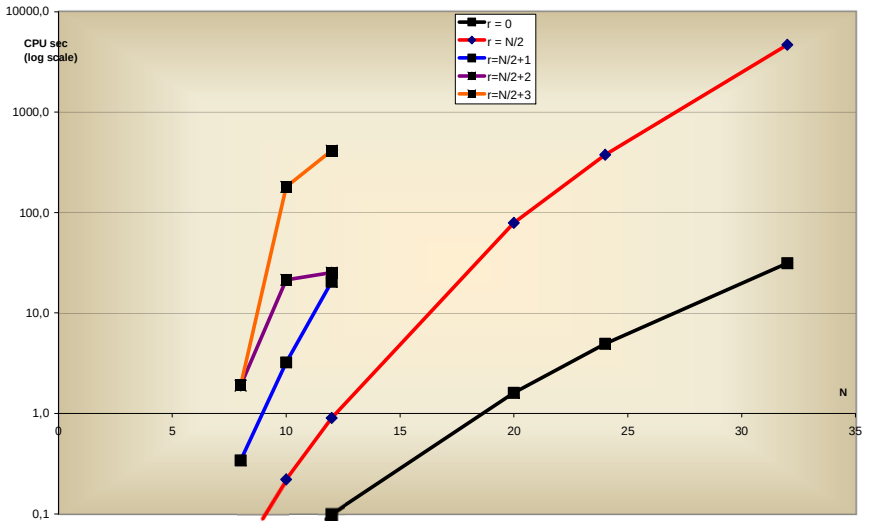


Fig. 10.1. Résultats expérimentaux pour 2R$^-$; temps CPU pour se ramener au problème 1R.

10.4 Crypto-système 1R

Dans le tableau suivant on donne le temps moyen total pour trouver *une* solution au problème $2R^-$ (ceci comprend donc la résolution de trois systèmes algébriques):

N=12 q=65521 *Temps moyen pour resoudre le probleme 2R --*

Ni/Minus	2	3	4	5	6	All
2	3489,1	7024,5	1434,5	4043,1	4349,3	4068,1
3	3957,1	684,1	829,9	1447,4	1462,7	1676,2
4	1424,9	850,5	688,1	681,9	3834,3	1495,9
6	725,8	533,2	371,5	842,1	610,4	616,6
12		561,7				

Résultats expérimentaux pour la résolution *globale* de $2\mathrm{R}^{-}$.

11. Arithmétique efficace Jacobienne d'une courbe $C_{a,b}$

Travail réalisé en collaboration avec A. Basiri, A. Enge et N. Gürel; pour tous les aspects techniques on se reportera aux articles originaux ((Basiri, 2003; the arithmetic of $C_{3,4}$ curves., 2004; Basiri *et al.* , 2004)).

Le but de ce chapitre est de montrer comment on peut générer automatiquement des formules très efficaces pour implanter l'arithmétique dans la Jacobienne d'une courbe super elliptique. En utilisant symboliquement l'algorithme FGLM du chapitre 3.1 et plusieurs techniques du Calcul Formel, la loi de groupe dans la Jacobienne d'une courbe super elliptique de genre 3 peut être calculée avec 130 multiplications, 10 carrés, 1 cube et 2 divisions. Pour plus de détails et notamment la définition exacte d'une Jacobienne on se reportera au livre (Cohen *et al.* , 2006).

11.1 Problème du logarithme discret

11.1.1 Diffie Hellman

On cherche à envoyer des données confidentielles sur un canal de communication non sûr, où l'information peut être interceptée par un espion. La cryptographie à clé publique est un moyen de répondre à ce problème. En particulier, un exemple bien connu est le cryptosystème proposé par W. Diffie et M.E. Hellman (Diffie & Hellman, 1976). Dans le cryptosystème Diffie-Hellman on se donne un groupe cyclique fini G d'ordre n, noté multiplicativement, et un générateur g de G. Pour tout entier x, en général il est très facile de calculer:

$$h = g^x$$

mais ce n'est pas le cas pour l'opération inverse:

"étant donnés g et h dans G retrouver un entier x tel que $h = g^x$".

C'est ce qu'on appelle le problème du logarithme discret. Le protocole de Diffie-Hellman peut se décrire de la façon suivante : Alice et Bob veulent communiquer un message. D'abord ils se mettent d'accord en choisissant un groupe cyclique fini G d'ordre n et un générateur g de G. De plus, chacun

d'entre eux choisit un entier (la clé privée). Notons a la clé privée d'Alice et b celle de Bob. Alice calcule $h_a = g^a$ et l'envoie à Bob. Celui-ci calcule $h_b = g^b$ et l'envoi en retour à Alice. Celle-ci calcule h_b^a et Bob calcule h_a^b. Maintenant chacun d'entre eux possède un secret commun $h_{ab} = g^{ab}$. Il est clair que si G est un groupe pour lequel résoudre le problème du logarithme discret est facile alors un espion peut facilement retrouver a et b depuis h_a et h_b et donc il peut connaître h_{ab}. Il faut donc rechercher des groupes pour lesquels résoudre le problème du logarithme discret est difficile tout en gardant une loi de groupe facile à calculer. Un exemple de famille de groupes intéressants est la famille des courbes elliptiques et des Jacobiennes des courbes hyperelliptiques (Miller, 1987; Koblitz, 1987; Koblitz, 1989). Dans ce cas le problème du logarithme discret semble difficile pour des courbes aléatoire.

11.1.2 Loi de groupe. Courbes super elliptiques.

Pour les courbes elliptiques on dispose de formules explicites et très optimisées (voir les formules 11.1). Dans le cas des courbes hyperelliptiques, il existe également des algorithmes en temps polynomial pour calculer la loi de groupe (voir par exemple la thèse de Gaudry (Gaudry, 2000)), et dans ce cas on dispose aussi de formules explicites. Cependant d'un point de vue cryptographique il pourrait s'avérer dangereux de se limiter à ces seules courbes: en effet il existe plusieurs algorithmes pour résoudre le problème du logarithme discret adapté à certaines instances de courbes elliptiques dans un corps fini; pour les courbes hyperelliptiques il existe aussi des algorithmes sous-exponentiels lorsque le genre de la courbe est suffisamment grand. Il est alors naturel de chercher à étendre la définition de ces groupes pour les courbes plus générales, comme les courbes super elliptiques et les courbes C_{ab} qui ont été introduites par Miura et Kamiya (Miura, 1998; Miura & Kamiya, 1993) :

$$\begin{aligned} \text{Courbe } C_{ab} \quad &: \quad Y^a + \sum_{i\, a+j\, b<ab} c_{i,j} X^i Y^j + X^b \\ \text{Courbe supperelliptiques} \quad &: \quad Y^a + \sum_{i \leq b} c_i X^i \end{aligned}$$

Encore une fois il existe des algorithmes sous-exponentiels lorsque le genre de la courbe est suffisamment grand (supérieur à 5, voir (Enge, 2002; Enge, 2001; Enge & Gaudry, 2002)), et d'un point de vue cryptographique on peut donc se limiter aux courbe super elliptiques et C_{ab} de genre 3 et 4 l'objectif étant de donner des formules les plus efficaces pour calculer la loi de groupe dans la Jacobienne (Basiri *et al.* , 2002a; Basiri *et al.* , 2003).

Les courbes super elliptiques ont déjà été étudiées par Galbraith, Paulus et Smart (Galbraith *et al.* , 2002) et Bauer (Bauer, 2001), et les courbes C_{ab} par Arita, Harasawa et Suzuki (Arita, 1999; Harasawa & Suzuki, 2000).

Dans ces travaux le principe est de présenter les éléments de la Jacobienne de façon à pouvoir les manier facilement en les mettant systématiquement sous forme canonique. L'idée est d'utiliser l'isomorphisme entre le groupe de la Jacobienne et le groupe des classes d'idéaux de $K[X,Y]$ modulo l'idéal engendré par la courbe C. En partant d'un élément quelconque dans une classe de diviseurs, le but est de calculer le représentant distingué associé à l'élément en question. Dans la méthode proposée par (Galbraith *et al.* , 2002; Harasawa & Suzuki, 2000) cela sera effectué en utilisant un analogue polynomial de l'algorithme LLL (Lenstra *et al.* , 1982).

11.1.3 Arithmétique: formules explicites par les bases de Gröbner

Arita (Arita, 1999) propose une autre technique pour implanter la loi de groupe en utilisant plusieurs calculs de base de Gröbner pour un ordre du degré (pondéré). Cet algorithme a une complexité majorée par $O(g^3)$ (où g est le genre de la courbe). Mais comme cet algorithme fait appel, en particulier, à l'algorithme de Buchberger il est impossible de prédire plus précisément le nombre de multiplications lorsque g prend une valeur fixe petite ($g = 3$). La méthode proposée dans (Galbraith *et al.* , 2002; Harasawa & Suzuki, 2000) fondée sur l'algorithme LLL polynomial possède une meilleure complexité théorique par rapport à la méthode Arita, $O(g^2)$ au lieu de $O(g^3)$, mais nous n'avons toujours pas d'estimation précise pour des petites valeurs de g.

C'est la raison pour laquelle nous présentons un nouvel algorithme de calcul de la loi de groupe pour les courbes C_{ab}. C'est une *méthode générale* pour obtenir des formules explicites. Le premier travail a été d'unifier les approches (Galbraith *et al.* , 2002; Harasawa & Suzuki, 2000) et (Arita, 1999) en remarquant qu'on pouvait interpréter la première méthode (resp. la deuxième méthode) comme un calcul de base de Gröbner pour l'ordre lexicographique (resp. l'ordre du degré pondéré par a et b). Ensuite, l'idée est d'effectuer les calculs "symboliquement", c'est à dire pour des paramètres non instanciés. Ceci a nécessité de donner des théorèmes de structure, en particulier un théorème décrivant la structure d'une base de Gröbner pour un ordre du degré pondéré (voir théorème 11.4.2, page 205). Plus précisément la preuve de ce théorème suit pas à pas l'algorithme FGLM (Faugère *et al.* , 1993) qui, à partir d'une base de Gröbner pour un ordre donné, trouve une base de Gröbner pour un autre ordre (voir aussi chapitre 3.1, page 52). À noter qu'il est difficile avec LLL de suivre symboliquement les calculs, c'est la raison pour laquelle nous avons remplacé LLL par FGLM (voir 3.2); ainsi nous pouvons établir des formules *explicites*. On obtient donc des formules "brutes" qu'il s'agit ensuite d'évaluer le plus rapidement possible. Notre première implantation utilisant la fonction `optimize` de *Maple* sur les formules de la section 11.4.4 (page 206) nécessitait à peu près 400 multiplications, ce qui représente un gain supérieur à 300 (resp. 100) multiplications par rapport à la méthode originale de (Arita, 1999) (resp. (Harasawa & Suzuki, 2000)). Le problème général d'optimiser l'évaluation d'une formule est un problème

difficile. Nous avons donc essayé d'optimiser les formules "à la main", par exemple en regroupant les termes et en utilisant des convolutions rapides comme la méthode de Karatsuba pour calculer le produit de deux polynômes. Avec les formules obtenues par cette méthode la loi de groupe dans la Jacobienne d'une courbe super elliptique de genre 3 peut être calculée avec 130 multiplications, 10 carrés, 1 cube et 2 divisions pour l'addition (voir les formules de la section 11.5.4, page 210). Ce qui représente un gain de trois par rapport à la version initiale.

11.2 Courbes elliptiques

On appelle courbe elliptique sur $\mathbb{R}$ toute courbe plane $\mathcal{C}$ d'équation $y^2 = x^3 + ax + b$, où le discriminant $-(4a^3 + 27b^2)$ de $x^3 + ax + b$ est non nul. On rajoute à cette courbe un point à l'infini noté O.

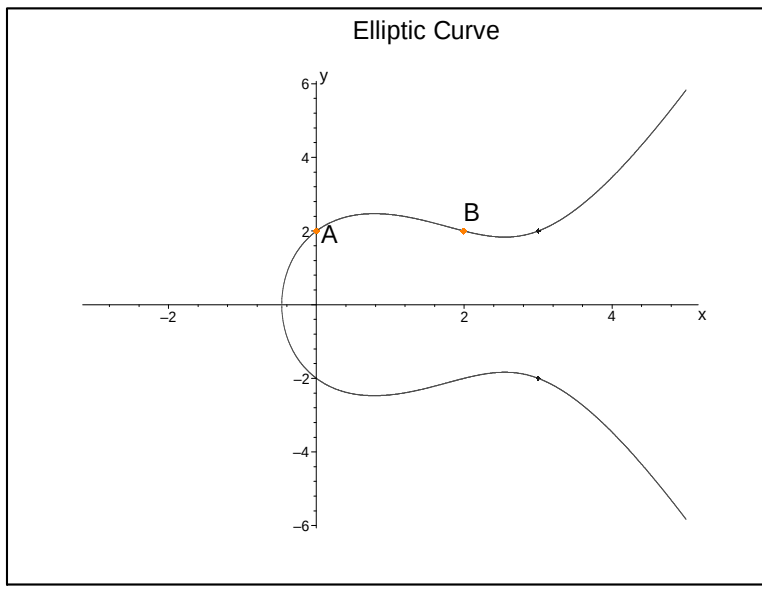
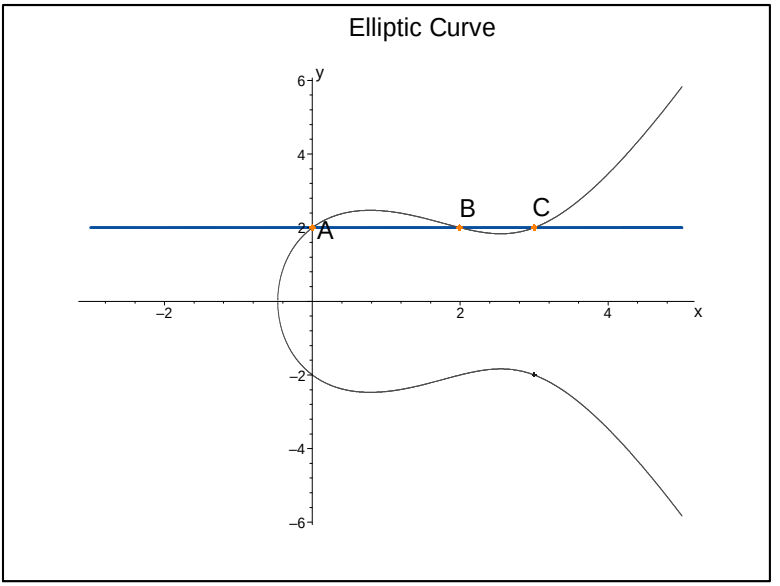


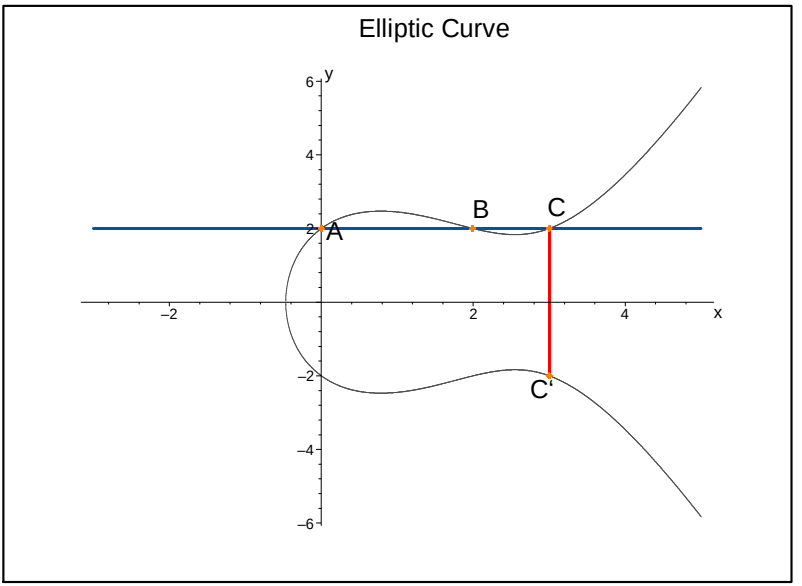
Fig.11.2 Courbe Elliptique

L'intérêt de ces courbes elliptiques est qu'on peut les munir d'une opération de groupe commutatif. Prenons A et B deux points distincts de la courbe elliptique (et différents du point à l'infini O).

On trace la droite (AB). Deux cas peuvent se produire :



1. La droite coupe la courbe en un 3ème point . Le symétrique de ce 3ème point par rapport à l'axe des abscisses est $C\prime = A + B$.
2. La droite ne coupe la courbe qu'en A et B. Ceci n'est possible que si (AB) est parallèle à l'axe des ordonnées. On définit alors $A + B = O$ (point à l'infini).



Si $A = B$, on considère la tangente à la courbe en A, et on définit $A + A$ comme ci-dessus. Enfin, on prolonge la définition de $+$ en posant $A + O =$

$O + A = A$. On peut alors prouver que l'opération + définit une loi de groupe sur la courbe elliptique. On peut d'ailleurs effectuer les calculs, pour obtenir les coordonnées de $A + B$ en fonction de celles de A et de B. Si $A = (x_1, y_2)$, $B = (x_2, y_2)$ alors les coordonnées (x_3, y_3) de $A + B$ sont données par:

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \tag{11.1}$$
$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$$

Par conséquent on dispose de formules explicites très optimisées sur les coordonnées (une division + un carré + une multiplication).

11.3 Courbes $C_{a,b}$ et super elliptiques.

11.3.1 Courbe hyperelliptique. Jacobienne.

Dans la suite k est un corps de caractéristique différente de 3 et $\overline{k}$ sa clôture algébrique (pour le cas hyperelliptique on pourra consulter (Duquesne & Lange, 2006)).

Une courbe *hyperelliptique* de genre g sur un corps k est une équation de la forme:

$$y^2 + h(x)\, y = f(x) \tag{11.2}$$

où $h \in k[X]$ est de degré au plus g et $f \in k[X]$ unitaire de degré $2\,g + 1$. (Une courbe elliptique est donc une courbe hyperelliptique de genre 1).

Contrairement aux courbes elliptiques, le groupe associé à une courbe hyperelliptique n'est en général plus formé par l'ensemble de ses points. Il s'agit de la Jacobienne qui peut être interprétée par des multi-ensembles d'au plus g points modulo une certaine relation d'équivalence. L'addition se décompose en deux étapes : la *composition* est simplement la réunion des deux multi-ensembles, la *réduction* remplace cet ensemble ayant, en général $2\,g$ points par un ensemble équivalent avec aux plus g points. Pour représenter les ensembles de points, on utilise en général la représentation polynomiale de Mumford : un polynôme $u(x)$ a comme racine les coordonnées x des points, un deuxième polynôme $v(x)$ interpole les coordonnées en y. Un point est donc donné par les équations:

$$\begin{cases} u(x) \\ y - v(x) \end{cases}$$

c'est donc aussi une base de Gröbner pour l'ordre lexicographique $y > x$ (et la base est en Shape Position 2.9.3). Dans la suite on représentera un multi-ensemble par une base de Gröbner pour l'ordre lexicographique (et

pour certains calculs intermédiaires par une base de Gröbner pour un ordré du degré). Cette représentation par une base de Gröbner permet donc de représenter un multi-ensemble dont la représentation polynomiale n'est pas de Mumford.

Une fois que l'arithmétique est spécifiée en terme d'opérations avec des polynômes, l'étape suivante consiste à transformer ces algorithmes en *formules explicites* sur les coefficients des polynômes et donc se ramener uniquement à des opérations arithmétiques dans le corps k.

11.3.2 Courbe $C_{a,b}$. Courbes super elliptiques.

Les courbes C_{ab} (Miura, 1998; Miura & Kamiya, 1993) sont définies par:

$$\text{Courbe } C_{ab}: \; Y^a + \sum_{i\, a+j\, b<ab} c_{i,j} X^i Y^j + X^b$$

avec a et b premiers entre eux. par conséquent les courbes elliptiques (resp. hyperelliptiques) ne sont que les courbes C_{23} (resp. C_{2b}) et les courbes super elliptiques ne sont que les courbe C_{ab} de la forme,

$$\text{Courbe supperelliptiques}: \; Y^a + f(X)$$

où f est un polynôme de degré b. Nous renvoyons le lecteur aux livres de Stichtenoth (Stichtenoth, 1993) et Fulton (Fulton, 1969) et la thèse (Basiri, 2003) pour la définition de la Jacobienne $\mathrm{Jac}_K(C)$ d'une courbe C. On peut démontrer en appliquant la formule de Riemann-Hurwitz ((Fulton, 1969), pages 8-36), que le genre d'une courbe $C_{a,b}$ est donné par la formule:

$$g = \frac{(a-1)\,(b-1)}{2}$$

Nous décrirons en termes d'idéaux l'algorithme d'Arita qui calcule la loi de groupe dans la Jacobienne d'une courbe C_{ab} , pour cela il nous faut définir un ordre admissible:

Definition 11.3.1. *Pour $\alpha = (\alpha_1, \alpha_2)$ et $\beta = (\beta_1, \beta_2) \in \mathbb{N}^2$ on définit l'ordre $C_{a,b}$ (ordre du degré pondéré puis lexicographique):*

$$x^\alpha <_{a,b} x^\beta \text{ si } \begin{cases} a\,\alpha_1 + b\,\alpha_2 < a\,\beta_1 + b\,\beta_2 \\ \quad ou \\ a\,\alpha_1 + b\,\alpha_2 = a\,\beta_1 + b\,\beta_2 \quad et \quad \beta_1 < \alpha_1 \end{cases}$$

c'est bien un ordre admissible.

11.4 Implantation de l'arithmétique

11.4.1 Algorithme d'Arita

Nous donnons d'abord une version synthétique de l'algorithme d'Arita réalisant l'addition dans le groupe de la Jacobienne $\mathrm{Jac}_K(C)$ puis nous donnons une version détaillée du même algorithme .

Algorithme 36 *Arita*

Input: *Deux idéaux I_1 et I_2 (points de la courbe C).*
Output: *Un idéal réduit I équivalent à l'idéal $I_1.I_2$.*
$J \leftarrow I_1.I_2$. *// Phase de composition*
$f \leftarrow$ *le polynôme minimal* $0 \neq f \in J$, *par rapport à l'ordre* C_{ab}.
$h \leftarrow$ *le polynôme minimal* $0 \neq h \in \mathrm{Id}(f, C) : J$, *par rapport à l'ordre* C_{ab}.
$I = \mathrm{RED}\,(J) \leftarrow \frac{h}{f}.J$. *// Phase de réduction*

Il est facile d'en déduire une version algorithmique en utilisant les outils du chapitre 2.7 page 36:

Algorithme 37 *Arita (détaillé)*

Input: *Deux idéaux $I_1 = (f_1, \ldots, f_a)$ et $I_2 = \mathrm{Id}(g_1, \ldots, g_a)$ (points de la courbe C*
Output: *Un idéal réduit I équivalent à l'idéal $I_1.I_2$.*
1. $G :=$GROEBNER$\big([F] \cup [f_i\, g_j \mid 1 \leq i, j \leq a], <_{a,b}\big)$.
2. $f :=$ *first*(G) *// le plus petit polynôme de G pour l'ordre C_{ab}.*
3. $G' :=$ QUOTIENT$(\mathrm{Id}(f, C), G, <_{ab})$ *// voir algorithme 14*
4. $h :=$ *first*(G') *// le plus petit polynôme de G' pour l'ordre C_{ab}*
5. $G'' :=$QUOTIENT1$([C, h] \cup G, f, <_{ab})$ *// voir algorithme 13*
return G''

11.4.2 Exemple Arita pour une courbe elliptique

Il est instructif d'appliquer l'algorithme d'Arita dans le cas particulier de la courbe elliptique utilisée dans l'introduction:

correspondant à la figure 11.2 :

$$C = y^2 - f\ (x) \text{ avec } f(x) = x\,(x-2)\,(x-3) + 4$$

Et on considère le point A:

$$I_1 = [x, y-2]$$

et le point B:

$$I_2 = [x-2, y-2]$$

La première étape (1) de composition de l'algorithme consiste à calculer le produit:

G :=GROEBNER($[F] \cup [f_i\, g_j \mid 1 \leq i, j \leq a]$) ce qui correspond aux deux points (A, B):

$$G = [y-2, x\ (x-2)]$$

par conséquent le plus petit polynôme de G est $f_{\min} = y - 2$ (étape 2) ; dans la phase de réduction on calcule ensuite (étape 3) G' := QUOTIENT(Id$(f_{\min}, C), G)$ ce qui correspond à l'intersection de la droite (A, B) avec la courbe:

$$G' = [x-3, y-2]$$

dont le plus petit polynôme est (étape 4) $h = x - 3$; on calcule ensuite une base de Gröbner de $\left[y^2 - f, h\right] \cup G$ ce qui donne:

$$[(x-3)\,(y-2)\,, (y-2)\,(y+2)\,, x\,(x-2)\,(x-3)] \tag{11.3}$$

le quotient par le polynôme $f_{\min} = y - 2$ donne immédiatement le point C' conjugué (étape 5) :

$$G'' = \text{RED}(G') = [x-3, y+2] \tag{11.4}$$

Note 11.4.1. Dans le chapitre 13 on a appliqué une autre stratégie: plutôt que de générer des formules qu'on doit évaluer par la suite on a généré un programme en C traçant les calculs des bases de Gröbner . Cette solution n'est pas efficace dans le cas présent car dans les calculs intermédiaires les bases de Gröbner sont plus grosses que dans le résultat final: il suffit de comparer les équations (11.3) et (11.4). L'avantage des formules est qu'on peut les simplifier sachant que le résultat final est "petit": par exemple, pour des courbe $C_{a,b}$ on effectue l'addition de multi-ensembles/idéaux représentant a points de la courbe; dans ce cas le résultat est aussi un multi-ensemble/idéal

représentant a points et les polynômes figurant dans le résultat sont donc de degrés inférieurs à a.

11.4.3 Structure de la base de Gröbner de l'idéal réduit

Definition 11.4.1. *Pour $s \in k[X,Y]$ et $u \in k[X]$ nous notons par $\phi_u(s)$ et $\delta_u(s)$ respectivement le reste et le quotient de la division de s par u, c'est à dire:*

$$\phi_u(s) := \mathrm{RED}(s,[u]) \text{ et } \delta_u(s) := \mathrm{Quo}(s,[u]) = \frac{s-\phi(s)}{u}$$

Maintenant nous calculons la première étape (étape 1) de l'algorithme pour le cas général où $I_1 \neq I_2$. Le théorème suivant calcule une base de Gröbner pour l'idéal en Shape Position $I_1 I_2 + \mathrm{Id}(C)$ à partir d'une base de Gröbner pour I_1 et une telle base pour l'idéal I_2 pour l'ordre lexicographique:

Theorem 11.4.1. *Soit $I_i = \mathrm{Id}(u_i(X), Y - v_i(X))$, deux idéaux de $K[X,Y]$ pour $i = 1,2$ et $C \in I_i$. Supposons que u_1 et u_2 sont premiers entre eux et prenons $s_1, s_2 \in K[X]$ tels que $s_1u_1 + s_2u_2 = 1$. Alors $[u, Y - v]$ est une base de Gröbner de l'idéal $I = I_1I_2 + \mathrm{Id}(C)$ pour l'ordre $(\prec_{Lex})$ et u divise $C(X,v)$, où $u = u_1u_2$ et $v = \phi_u(s_1u_1v_2 + s_2u_2v_1) = \phi_u(u_1s_1(v_2 - v_1) + v_1)$.*

Nous avons donc calculé symboliquement la première étape de l'algorithme 37 (page, 202). Pour la deuxième étape nous n'avons rien à calculer car le polynôme minimal de l'idéal est le polynôme $f_{\min} := u = \min_{Lex}(I)$, calculé dans le théorème 11.4.1. Maintenant nous passons à la troisième étape de l'algorithme 37 : il faut calculer $\min_{C_{ab}} \mathrm{Id}(u,C) : I$. Dans le lemme suivant on calcule un polynôme noté q qui est un des générateurs de l'idéal $\mathrm{Id}(u,C) : I$:

Lemma 11.4.1. *Si $v(X) \in K[X]$ et $C(X,Y) := Y^a + \sum_{j=0}^{a-1} C_j(X)Y^j \in K[X,Y]$ alors $Y - v$ divise $C(X,Y) - C(X,v)$. Dans ce cas on note:*

$$q := \frac{C(X,Y) - C(X,v)}{Y - v}. \tag{11.5}$$

Le lemme suivant calcule un ensemble de générateurs de l'idéal $[\mathrm{Id}(u,C) : a]$.

Lemma 11.4.2. *Soit $I = \mathrm{Id}(u(X), Y - v(X))$, $C = Y^a + \sum_{j=0}^{a-1} C_j(X)Y^j \in I$ et u divise $C(X,v)$ alors*

$$\mathrm{denom}(I)I^{-1} = [\mathrm{Id}(u,C) : a] = \mathrm{Id}(C,u,q).$$

Le lemme suivant calcule un ensemble de générateurs de l'idéal $[\mathrm{Id}(u,C) : I]$ comme un $K[X]$-module.

Lemma 11.4.3. *Soit* $I = \mathrm{Id}(u(X), Y - v(X))$, $C = Y^a + \sum_{j=0}^{a-1} C_j(X)Y^j \in I$, *u divise* $C(X, v)$ *et* $J = (\mathrm{Id}(u, C) : I)$ *alors*

$$J = \mathrm{Id}(C) + [u, uY, ..., uY^{a-2}, q]_{K[X]} \tag{11.6}$$

c'est à dire:

$$J = \left\{ \lambda C + \sum_{i=0}^{a-2} \gamma_i u Y^i + \gamma_{a-1} q \mid \lambda \in K[X, Y] \text{ et } \gamma_i \in K[X] \text{ pour } i = 0..a-1 \right\}$$

Le théorème suivant calcule la forme d'un élément minimal de l'idéal $[\mathrm{Id}(u, C) : I]$ par rapport à l'ordre C_{ab}, à noter que

Theorem 11.4.2. *Soit* $I = \mathrm{Id}(u(X), Y - v(X))$, $J_1 = [u, uY, ..., uY^{a-2}, q]_{K[X]}$ *alors*

$$\min_{C_{ab}}(J_1) = \min_{C_{ab}}(\{u\} \cup \{\phi_u(\alpha q) | \alpha \in K[X]\}).$$

Proof. pour calculer l'élément minimal explicitement on utilise l'algorithme FGLM (voir la section 3.1, page 52): l'idée est que I est un idéal donné sous pour l'ordre lexicographique et on veut calculer (explicitement) une base pour l'ordre du degré pondéré (ordre $C_{a,b}$).

Ici nous sommes à la fin de l'étape 4 de la procédure 37 (page 202). Le théorème suivant calcule un ensemble de générateurs pour l'idéal $\mathrm{RED}(I)$ où I est un idéal en Shape Position (l'étape 5 de la procédure 37). Là aussi pour calculer exactement la formule explicite de la base de Gröbner voir la section 11.4.4, page 206.

Theorem 11.4.3. *Soit* $I = \mathrm{Id}(u(X), Y - v(X))$, $C = Y^a + \sum_{j=0}^{a-1} C_j(X)Y^j \in I$, $J = \mathrm{Id}(C) + [u, uY, ..., uY^{a-2}, q]_{K[X]}$, $J_1 = [u, uY, ..., uY^{a-2}, q]_{K[X]}$ *et il existe* $w \in K[X]$ *tel que* $C(X, v) = uw$, *alors*

1. *s'il existe un* $\alpha \in K[X]$ *tel que* $\min_{C_{ab}}(J_1) = \phi(\alpha q)$, *alors*

$$\mathrm{RED}(I) = \mathrm{Id}(C, \phi_u(\alpha q), \alpha w + \delta_u(\alpha q)(Y - v)).$$

2. *si* $\min_{C_{ab}}(J_1) = \phi(q)$ *alors*

$$\mathrm{RED}(I) = \mathrm{Id}(\phi_u(q), w + \delta_u(q)(Y - v))$$

en particulier si $\min_{C_{ab}}(J) = q$ *alors*

$$\mathrm{RED}(I) = \mathrm{Id}(q, w).$$

3. *si* $\min_{C_{ab}}(J_1) = u$ *alors*

$$\mathrm{RED}(I) = I.$$

Dans la sous-section suivante nous donnons les formules plus explicites pour une courbe super elliptique du genre 3.

11.4.4 Cas super elliptique de genre 3

Dans ce paragraphe nous obtenons la forme de l'idéal réduit $\mathrm{RED}(I)$ de l'idéal $I = \mathrm{Id}(u(X), Y - v(X))$ de $K[X,Y]/\mathrm{Id}(C)$ où C est une courbe super elliptique de genre 3.

Dans cette section nous adoptons également les notations suivantes :

- f est un polynôme de degré quatre dans $k[X]$ sans racines multiples dans $\overline{k}$ et $C = Y^3 - f$.
- u, v et w sont des polynômes dans $k[X]$ dont $\deg(v) < \deg(u) \leq 6$ et $v^3 - f = wu$.

On définit $q = Y^2 + vY + v^2$ et l'idéal recherché est:

$$J = \mathrm{denom}(I)I^{-1} = \mathrm{Id}(u, C) : I = \mathrm{Id}(C, u, q) = \mathrm{Id}(C) + [u, uY, q]_{K[X]}.$$

Dans la table suivante nous donnons l'expression de $\mathrm{RED}(I)$ selon les différents degrés des polynômes u et v pour calculer l'idéal réduit:

$\deg(u)$	$\deg(v)$	$\mathrm{RED}(I)$
6	5	$\mathrm{Id}(C, \phi_u(\alpha q), \alpha w + (\delta_u(\alpha v)Y + \delta_u(\alpha v^2))(Y - v))$
6	5	$\mathrm{Id}(C, \phi_u(\alpha q), \alpha w + (Y + \delta_u(\alpha v^2))(Y - v))$
6	4	$\mathrm{Id}(C, \phi_u(\alpha q), \alpha w + (Y + \delta_u(\alpha v^2))(Y - v))$
5	4	
4	3	
6	3	$\mathrm{Id}(\phi_u(q), w + \delta_u(v^2)(Y - v))$
5	3	
4	2	
5	2	$\mathrm{Id}(w, q)$
3	≤ 1	$\mathrm{Id}(w, q)$
6	2	$\mathrm{Id}(1)$
4	≤ 1	$\mathrm{Id}(1)$
3	2	$\mathrm{Id}(u, Y - v)$
≤ 2	-	$\mathrm{Id}(u, Y - v)$

Table 3: les différents cas du problème.

11.5 Formules optimisées

Dans cette section on décrit la façons dont on a obtenu les formules explicites optimisées (voir l'article original (the arithmetic of $C_{3,4}$ curves., 2004) pour les détails) pour les deux opérations principales de composition et de réduction; on compte aussi le nombre de multiplications et d'inversions dans le corps de base.

11.5.1 Convolutions rapides

Comme brique de base dans les formules optimisées on utilise aussi des multiplications de polynômes, des produits courts de polynômes (c'est à dire qu'on ne garde que les termes de bas degré dans un produit (the arithmetic of $C_{3,4}$ curves., 2004)).

Nous avons utilisé des algorithmes de convolutions rapides comme la méthode de Karatsuba pour calculer le produit de deux polynômes (voir (Nussbaumer, 1981)); en utilisant cette méthode pour la multiplication de deux polynômes de degré deux on peut ainsi économiser 3 multiplications par rapport à la méthode naïve. On note $M(m,n)$ le nombre de multiplications dans le corps de base nécessaire pour multiplier deux polynômes de degré $m-1$ et $n-1$. Nous avons utilisé l'interpolation de polynômes pour calculer efficacement certains produits: par exemple en utilisant cette méthode pour la multiplication de deux polynômes de degré deux on peut ainsi économiser 4 multiplications par rapport à la méthode naïve. Dans cette méthode on doit calculer les divisions par 2, 3 et 5. Pour cela nous pré-calculons 1/2, 1/3 et 1/5 une seule fois, dans ce cas la division peut être traitée facilement par addition et décalage des bits. Par exemple, la procédure DIV-PAR-3 (dans un système de 32 bits) réalise la division par 3. Dans la procédure la notation $(n >> i)$ désigne le décalage à droite et est donc équivalente à $\lfloor n/2^i \rfloor$.

DIV-PAR-3
Input: Un entier n et la constante $d_3 = 1/3 \bmod p$
Output: $q := n/3 \bmod p$
$q \leftarrow (n >> 2) + (n >> 4)$
$q \leftarrow q + (q >> 4)$
$q \leftarrow q + (q >> 8)$
$q \leftarrow q + (q >> 16)$
$r \leftarrow n - q \times 3$
$q \leftarrow q + (11r >> 5)$
$r \leftarrow n - q \times 3$
return $m + r \times d_3$

La procédure DIV-PAR-3 nécessite 11 (resp. 11) décalages et 12 (resp. 13) additions. Les procédures INTER_{mn} réalisent la multiplication de deux polynômes de degré $(m-1)$ et $(n-1)$. Dans la table suivante, nous donnons le nombre de multiplications dans le corps, qui est nécessaire pour chaque appel d'une procédure d'interpolation:

Opération	Spécification	Nb de ×	Degré du résultat
$p_1 \cdot p_2$	$\deg_1 \times \deg_1$	3	1
$p_1 \cdot p_2$	$\deg_2 \times \deg_2$	5	4
$p_1 \cdot p_2$	$\deg_2 \times \deg_3$	6	6
$p_1 \cdot p_2$	$\deg_1 \times \deg_2$	3	2
$p_1 - p_2.p_3$	$\deg_5$ - $\deg_2 \times \deg_3$	3	2
$p_1.p_2 - p_3.p_4$	$\deg_2 \times \deg_5$-$\deg_1 \times \deg_6$	8	3
$p_1.p_2 - p_3.p_4$	$\deg_3 \times \deg_5$-$\deg_2 \times \deg_6$	12	5

Par exemple il faut $M(2,2) = 5$ multiplications pour réaliser la multiplication de polynômes de degré 2 (cette fonction utilise à sont tour la procédure DIV-PAR-3)

INTER$_{33}$: $(a_0 + a_1 X + a_2 X^2) \times (b_0 + b_1 X + b_2 X^2)$
$C_0 := a_0 \times b_0$;
$C_4 := a_2 \times b_2$;
$t_1 := (a_0 + a_1 + a_2) \times (b_0 + b_1 + b_2)$;
$t_{-1} := (a_0 - a_1 + a_2) \times (b_0 - b_1 + b_2)$;
$t_2 := (a_0 + 2 \times a_1 + 4 \times a_2) \times (b_0 + 2 \times b_1 + 4 \times b_2)$;
$C_1 := t_1 - 1/2 \times C_0 - 1/3 \times t_{-1} + 2 \times C_4 - 1/6 \times t_2$;
$C_2 := 1/2 \times t_{-1} - C_0 + 1/2 \times t_1 - C_4$;
$C_3 := 1/6 \times t_2 + 1/2 \times C_0 - 1/2 \times t_1 - 1/6 \times t_{-1} - 2 \times C_4$;
return $C_0 + C_1 X + C_2 X^2 + C_3 X^3 + C_4 X^4$

11.5.2 Composition — addition

Theorem 11.5.1. *Soient $I_1 = \langle u_1, Y - v_1 \rangle$ et $I_2 = \langle u_2, Y - v_2 \rangle$ tels que u_i divise $v_i^3 + v_i h - f$ et $\gcd(u_1, u_2) = 1$, alors sous réserve de généricité des restes successifs dans l'algorithme d'Euclide, on peut calculer:*

$$I_1 I_2 = \langle u, Y - d^{-1} v \rangle$$

avec 37 multiplications. .

Proof. On calcule d'abord $u = u_1 u_2$ par la multiplication à la Toom–Cook multiplication avec 5 multiplications. On doit calculer le polynôme v donné par les formules (voir le théorème 11.4.1):

$$s_1 = u_1^{-1} \bmod u_2$$
$$t = s_1(v_2 - v_1) \bmod u_2$$
$$v = v_1 + t u_1$$

on détermine s_1 de degré 2 et d tel que $s_1 u_1 \equiv d \bmod u_2$ en appliquant l'algorithme d'Euclide étendu à u_2 et $u_1 - u_2$. On peut réaliser cette opérations

lorsque $n = 2$ et r_{-1} est unitaire avec 15 multiplications seulement. Le calcul de $t_1 = s_1(v_2 - v_1)$ nécessite encore 5 multiplications et le quotient q par u_2 est obtenu avec 1 multiplication supplémentaire. Le polynôme $t = t_1 - q \cdot u_2$ de degré 2 est obtenu par interpolation sur 3 points. Enfin, $v = d \cdot v_1 + u_1 \cdot t$ est calculé en $M(1,3) + M(3,3) = 8$ multiplications.

11.5.3 Composition — doublement

Le calcul de la phase de composition dans le cas du doublement (calcul de l'idéal I_1^2) est très similaire au calcul de la composition dans le général; cependant il nécessite plus d'opérations:

Theorem 11.5.2. *Soient $I_1 = \langle u_1, Y - v_1 \rangle$ tel que $u_1 | v_1^3 + v_1 h - f$, et sous réserve de généricité des restes successifs dans l'algorithme d'Euclide,on peut calculer*

$$I_1^2 = \langle u, Y - d^{-1} v \rangle$$

avec 61 *multiplications.*

Proof. (Esquisse de la preuve) On doit calculer le polynôme v donné par les formules ((Basiri, 2003; the arithmetic of $C_{3,4}$ curves., 2004; Basiri *et al.* , 2004)):

$$s_3 = (3v_1^2 + h)^{-1} \bmod u_1$$

$$w_1 = \frac{v_1^3 + h v_1 - f}{u_1} \tag{11.7}$$

$$t = -s_3 w_1 \bmod u_1; : \tag{11.8}$$

$$v = v_1 + t u_1. \tag{11.9}$$

On commence par calculer $u = u_1^2$ et v_1^2 en 5 multiplications pour chacun. Pour obtenir w_1, on détermine d'abord les quatre coefficients de plus haut degré de $v_1^3 + v_1 h - f = v_1^2 \cdot (v_1 + h) - f$ avec 6 multiplications. Par division (exacte) par u_1 on obtient w_1 en 6 multiplications. La prochaine étape consiste à déterminer s_3 de degré 2 et d dans le corps de base tels que $s_3 \cdot (3v_1^2 + h) \equiv d \bmod u_1$. Ceci nécessite 19 multiplications.

On peut alors réduire w_1, qui est de degré 3, modulo le polynôme unitaire u_1 ce qui donne t et nécessite un total de 12 multiplications. Finalement, v est obtenu par multiplication de v_1 par d et t par u en $M(3,1) + M(3,3) = 8$ multiplications.

Note 11.5.1. On peut noter que la phase de composition a le même coût pour une courbe $C_{3,4}$ ou une courbe super elliptique..

11.5.4 Réduction

On utilise le théorème 11.4.3 et la table 3 pour calculer efficacement RED(I) où I est l'idéal obtenu lors de la phase de composition:

Theorem 11.5.3. *Soient $I = \langle u, Y - d^{-1}\tilde{v}\rangle$ un idéal tel que $u|(d^{-1}\tilde{v})^3 + d^{-1}\tilde{v}h - f$, u polynôme unitaire de degré 6, $\tilde{v}$ de degré 5, alors on peut calculer* RED(I) $= \langle u', Y - v'\rangle$ *avec* 113 *multiplications et* 2 *inversions. Dans le cas super elliptique, on peut économiser* 10 *multiplications.*

Proof. (voir (the arithmetic of $C_{3,4}$ curves., 2004) pour la preuve).

Dans la table suivante on donne le nombre d'opérations arithmétiques nécessaire pour réaliser l'addition dans la Jacobienne. Ces nombres ne sont sans doute pas minimaux car on peut toujours optimiser les formules mais ils donnent une idée précise sur la façon dont les algorithmes se comportent. Dans la table on fait la distinction entre le nombre de multiplications et le nombre d'opérations. Le nombre d'inversions est toujours 2.

Courbe	super elliptique			$C_{3,4}$		
	$\#x \times y$	$\#x^2$	#Total	$\#x \times y$	$\#x^2$	#Total
$I + J$	129	11	140	139	11	150
$I + I$	143	21	164	153	21	174

Afin d'être complet nous comparons ces formules avec celles de Flon et Oyono (voir (Flon & Oyono, 2003)) et les formules plus récentes FOR mais nécessitant une hypothèse "flex" supplémentaire sur la courbe (voir (Flon *et al.* , 2004)):

Addition	nos formules	Flon et Oyono	FOR (cas flex)
$\#x \times y$	129	144	114
$\#x^2$	11	12	16

11.6 Conclusion

En utilisant *symboliquement* l'algorithme FGLM (Algorithme 16, page 57) et l'algorithme d'Euclide, nous avons obtenu des formules explicites de la réduction qui nous donnent les coefficients des polynômes de la réduction à partir des coefficients des polynômes de l'entrée.

L'arithmétique des courbes hyperelliptiques de genre 3 est décrite dans (Pelzl *et al.* , 2003): il faut 76 opérations dans dans le corps de base et une inversion pour réaliser l'addition de deux éléments distincts, et 71 multiplications et une inversion pour le doublement d'un élément. Même si les formules pour les courbes super elliptiques et $C_{3,4}$ sont plus coûteuses, le surcoût est

inférieur à 2 montrant que l'utilisation des courbes $C_{3,4}$ en cryptographie reste une alternative raisonnable.

Le code MAGMA(CANNON J., 1998) (facilement adaptable dans n'importe quel autre langage) est disponible.

Part III

Applications en Robotique, Théorie du Signal, Théorie des Codes, Géométrie Algorithmique

12. Placement de Protéines

We are deeply indebted to Prof. Bernd Sturmfels for bringing this problem to our attention. Travail en collaboration avec M. Hering and J. Phan (Faugère *et al.* , 2000; Faugère *et al.* , 2003).

12.1 Introduction

Both the shapes and positions of proteins which are embedded in a cell membrane can influence their biological function. It is the interaction between the proteins which dictates how they become arranged, but little is known about this interaction and its exact cause is uncertain. However, for conical proteins, a likely explanation is the bending of the membrane caused by the proteins. Specifically, an embedded conical protein induces a curvature in the two dimensional membrane which influences the positions of neighboring proteins. There is an energy associated to this curvature and the proteins will tend to arrange themselves so as to minimize this energy. Recent work in (Kim *et al.* , 1998) shows that any minimum energy arrangement is a zero energy arrangement. Furthermore, if z_i is the position of the ith protein using complex coordinates, it was also shown that the energy at the ith protein is a constant multiple of $|f_i(z_1, \ldots, z_N)|^2$ where

$$f_i(z_1, \ldots, z_N) = \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{1}{(z_i - z_j)^2}.$$

Therefore the N proteins are at equilibrium if and only if $(z_1, \ldots, z_N)$ is a solution to the Membrane Inclusions Curvature Equations, or MICE:

$$f_i(z_1, \ldots, z_N) = 0, \quad i = 1, \ldots, N. \tag{12.1}$$

For brevity, we refer to the N-th system of equations as M_N.

One possible application of knowing how these proteins arrange themselves is to deduce the form of proteins by examining the shapes they form. In this case, if they arrange themselves according to our solutions it is very

likely that they are conical. Determining the shapes of proteins is still an unsolved problem in biology.

Gröbner bases are used to find the solutions of M_N for several N. In section **??**, we review the most efficient algorithms for computing Gröbner bases and their implementations. Direct application of these algorithms gives all the solutions of the problem for $N < 7$ and is described in section 12.1.1. Because the difficulty of computing Gröbner bases increases rapidly with respect to the complexity of the input equations, it is necessary to reformulate the system before most of the computations will successfully terminate. Two reformulations of M_N into equivalent systems are given in section 12.1.2. The first reformulation employs an algorithm for converting the numerators of the M_N equations into symmetric polynomials, which are then expressed in terms of the elementary symmetric functions prior to computing. The second reformulation uses a differential equation describing the minimum polynomial for the coordinates of a solution and gives directly a system already formulated using the elementary symmetric functions. Both reformulations can be used jointly to decrease the computation time. Finally, we consider a much simplified system obtained from M_N by limiting our search to those solutions which have a certain geometric regularity to them; namely, we look for solutions whose coordinates form concentric rings of regular polygons. While this last approach does not detect all solutions for a given N, it does allow many to be found.

Our main result is a complete classification of the solutions for small values for N:

Theorem 12.1.1. *There are no solutions for $N \leq 12$ except for $N = 5$ (finite number of solutions) and $N = 8$ (M_8 form a 1 dimensional variety).*

The proof of this theorem is included in sections 12.1.1 and 12.1.2. For larger values of N we have only a partial result:

Theorem 12.1.2. *There exist solutions to M_N for $N = 5, 8, 16, 21, 33, 37, 40, 56, 65, 85, 119, 133, 161, 175, 208, 225, 261$ and 280. Moreover the number of solutions for M_{16} and M_{21} is infinite.*

We explain in section 12.1.3 how we find this list of "regular solutions".

12.1.1 First experiments

First, we observe that the set of solutions to M_N is invariant under translation and multiplication by complex scalars. These considerations allow us to change coordinates so that $z_1 = 0$ and $z_2 = 1$.

Since the f_i in the system M_N are rational functions we need to transform the system into a polynomial system. In order to avoid "parasite" solutions, where $z_i = z_j$ for some $i \neq j$, we introduce a new variable u and let P_i be the numerator of each f_i in M_N. That is to say

$$P_i(z_1,\ldots,z_N)=\sum_{j\neq i}\prod_{\substack{k\neq i\\ k\neq j}}(z_i-z_k)^2=0,\quad i=1,\ldots,N, \tag{12.2}$$

$$M'_N=\begin{cases} u\prod_{i=1}^{N}\prod_{j=i+1}^{N}(z_i-z_j)=1\\ P_i(z_1,\ldots,z_N)=0\quad i=1,\ldots,N\\ z_1=0\\ z_2=1. \end{cases}$$

Proposition 12.1.1. *There is no solution for $N\leq 4$ and $N=6$. The only solution for M_5 is a regular pentagon.*

Proof. For $N\leq 5$ it takes less than 0.1 second to compute a lexicographic Gröbner basis with FGb on a PC Pentium II 300 Mhz. For $N<5$ the Gröbner basis is $\{1\}$. For $N=5$ we can factorize the univariate polynomial and find a decomposition into irreducible varieties: $V=V_1\cup V_2\cup V_3\cup V_4\cup V_5\cup V_6$ and

$$I(V_1)=[z_3-{z_5}^3+{z_5}^2-z_5, z_4+{z_5}^2-z_5, {z_5}^4-{z_5}^3+{z_5}^2-z_5+1].$$

For any polynomial p in $x_1,\ldots,x_N$ and any permutation σ, set $\sigma.p=p(x_{\sigma(1)},\ldots,x_{\sigma(N)})$ and $\sigma.I(V)=\{\sigma.p:\ \forall p\in I(V)\}$. It is easy to check that

$$\begin{array}{l}(z_4,z_5).I(V_1)=I(V_6)\\ (z_3,z_5).I(V_1)=I(V_3)\\ (z_3,z_4).I(V_1)=I(V_2)\\ (z_3,z_4,z_5).I(V_1)=I(V_5)\\ (z_3,z_5,z_4).I(V_1)=I(V_4).\end{array}$$

Now we have

$${z_5}^4-{z_5}^3+{z_5}^2-z_5+1=\frac{z_5^5+1}{z_5+1},$$

so that $z_5=e^{\frac{\alpha\imath\pi}{5}}$ and we see that the only solution is the regular pentagon.

The case $N=6$ is a little more difficult: the degree of the polynomial $u\Pi_{i=1}^{N}\Pi_{j=i+1}^{N}(z_i-z_j)=1$ is $1+\frac{N(N-1)}{2}=16$ and so big that it does not help the Gröbner basis computation. In that case we can replace this condition by $uz_3z_4z_5z_6=1$ and it takes only 13.6 seconds to find $\{1\}$ with Fgb.

In conclusion, the straightforward approach solves the problem for small N but leads to several problems:

- intermediate computations contain the same solution several times (because solutions are invariant under permutations of the variables $z_3, \ldots, z_N$), The symmetric group S_{N-2} is permuting the variables $x3, ..., xN$. so the degrees of the intermediate varieties are big.
- it is not easy to remove the parasite solutions $z_i = z_j$.

We have stopped the computation for $N = 7$ after 2000 seconds.

12.1.2 Using the symmetry

It is clear from (12.1) that if $(z_1, \ldots, z_N) \in \mathbf{C}^N$ is a solution of M_N then $(z_{i_1}, \ldots, z_{i_N})$ is also a solution of M_N for every possible permutation of $(i_1, \ldots, i_N)$ of $(1, \ldots, n)$. Hence it is enough to compute the polynomial

$$f(X) = (X - z_1) \cdots (X - z_N) = X^N - e_1 X^{N-1} + \cdots (-1)^N e_N,$$

where the $e_i = e_i(z_1, \ldots, z_N)$ are the elementary symmetric functions in $z_1, \ldots, z_N$. In this paper we will say that f is a solution to M_N.

In general solving efficiently a polynomial system with symmetries is an open issue especially when the group acting is a proper subgroup of the symmetric group. In our problem the solutions are invariant under the symmetric group but unfortunately f_i is not a symmetric polynomial in $(z_1, \ldots, z_n)$ but only in $\{z_j \mid j \neq i\}$. If we exchange the role of z_j and z_k then f_i remain unchanged while f_j becomes f_k and reciprocally

$$z_j \longleftrightarrow z_k \qquad f_i = f_i \text{ for } i \neq j, k \qquad f_j \longleftrightarrow f_k.$$

nilCoxeter algebra. Let e_r be the rth elementary symmetric function in N variables. For $\lambda = (\lambda_1, \ldots, \lambda_r)$ let

$$m_\lambda = \sum z_{i_1}^{\lambda_1} \cdots z_{i_r}^{\lambda_r} \tag{12.3}$$

denote the monomial symmetric functions, where the sum ranges over all monomials whose exponent vector is equal to a permutation of λ. Solving M_N is equivalent to finding a polynomial

$$\begin{cases} f = X^N - e_1 X^{N-1} + e_2 X^{N-2} - \cdots + (-1)^N e_N \\ f \text{ is squarefree,} \end{cases} \tag{12.4}$$

whose roots are a solution to M_N. For any polynomial p in $z_1, \ldots, z_N$, set

$$\partial_i(p) = \frac{p(z_1, z_2, \ldots, z_N) - p(z_i, z_2, \ldots, z_{i-1}, z_1, z_{i+1}, \ldots, z_N)}{z_1 - z_i}. \tag{12.5}$$

Let I_1 be the ideal generated by $P_1, \ldots, P_N$. We define by induction

$$I_k = I_{k-1} : (\prod_{i_1<i_2} (z_{i_1} - z_{i_2})) \tag{12.6}$$

and $I = I_\infty$. Note that $P_i = (1, i).P_1$ for $1 \leq i \leq N$ and P_1 is symmetric in $z_2, \ldots, z_N$.

Theorem 12.1.3. *Define for* $1 \leq i_1 < \cdots < i_{k+1} \leq N$

$$P_{i_1,\ldots,i_k,i_{k+1}} = \frac{P_{i_1,\ldots,i_k} - P_{i_1,\ldots,i_{k-1},i_{k+1}}}{z_{i_k} - z_{i_{k+1}}},$$

so that $P_{i_1,\ldots,i_k} \in I_k$ *and* $P_{i_1,\ldots,i_k}$ *is symmetric in* $z_{i_1}, \ldots, z_{i_k}$ *and in the complementary set of variables. Hence*

$$H_k = \sum_{1\leq i_1<\cdots<i_k\leq N} P_{i_1,\ldots,i_k}$$

is a true symmetric function.

The next theorem gives an efficient method for computing the H_k.

Theorem 12.1.4. *For* $1 \leq i_1 < \cdots < i_k \leq N$

$$P_{i_1,\ldots,i_k} = (1, i_1).(2, i_2).\cdots.(k, i_k)Q_k,$$

where $Q_k = P_{1,2,\ldots,k}$ *and we have*

$$Q_k = \partial_k Q_{k-1}.$$

The H_i were first computed in the monomial basis m_λ using code specifically written for this application in C++ in the small computer algebra system Gb; then the polynomials were expressed in the e_i basis. If we set $z_N = 0$ and $z_{N-1} = 1$ prior to computing the H_i, the reformulated system $\tilde{S}_N$ consists of the polynomials $H_1, \ldots, H_N, P_{N-1}, P_N$ in the variables $e_1, \ldots, e_{N-2}$. It turns out that $\tilde{S}_N$ is easier to solve: it takes 2 minutes to compute a Gröbner basis for $N = 10$ with FGb, while the calculation for M_7 was unsuccessfully stopped after 2000 seconds.

Harm Derksen's formulation. Our second reformulation was found by Harm Derksen, and appeals to the structure of the polynomial f in (5). First, a lemma.

Lemma 12.1.1. *For any* $(z_1, \ldots, z_N) \in \mathbf{C}^N$,

$$\sum_{j=1}^{N} \frac{1}{z_j^2} = \frac{e_{N-1}^2 - 2e_N e_{N-2}}{e_N^2}.$$

Proof. Since $\sum_{j=1}^{N}(1/z_j) = e_{N-1}/e_N$ and $\sum_{i>j}(1/z_i z_j) = e_{N-2}/e_N$, we have

$$\begin{aligned}\sum_{j=1}^{N}\frac{1}{z_j^2} &= (\sum_{j=1}^{N}\frac{1}{z_j})^2 - 2\sum_{i>j}\frac{1}{z_i z_j}\\ &= \frac{e_{N-1}^2}{e_N^2} - 2\frac{e_{N-2}}{e_N}.\end{aligned}$$

Theorem 12.1.5. *$(z_1, \ldots, z_N)$ is a solution to M_N if and only if*

$$\begin{cases} f \text{ is squarefree and} \\ 3(f''^2 - 4f'f''' \text{ is divisible by } f, \end{cases}$$

where $f = \prod_{i=1}^{N}(x - z_i) = x^N - e_1 x^{N-1} + e_2 x^{N-2} - \cdots + (-1)^N e_N$.

Proof. Let S_r be the rth elementary symmetric polynomial in $x - z_1, \ldots, x - z_N$. Note that replacing x by z_i in S_r gives the rth elementary symmetric polynomial in $z_i - z_1, \ldots, z_i - z_{i-1}, z_i - z_{i+1}, \ldots, z_i - z_N$, which we denote by E_r^i. Furthermore, the kth derivative of f is $f^{(k)} = k!S_{N-k}$ so that $f^{(k)}(z_i) = k!E_{N-k}^i$. Set $h := 3(f''^2 - 4f'f'''$. Then

$$\begin{aligned}h(z_i) &= 3(2E_{N-2}^i)^2 - 4E_{N-1}^i(3!E_{N-3}^i)\\ &= 12((E_{N-2}^i)^2 - 2E_{N-1}^i E_{N-3}^i).\end{aligned} \tag{12.7}$$

By Lemma 12.1.1

$$f_i = \sum_{j\neq i}\frac{1}{(z_i - z_j)^2} = \frac{(E_{N-2}^i)^2 - 2E_{N-1}^i E_{N-3}^i}{(E_{N-1}^i)^2}, \tag{12.8}$$

so that $h(z_i)$ is a constant multiple of the numerator of f_i. Therefore f divides h and the z_i are distinct $\iff$ $h(z_i) = 0$ for all i and the z_i are distinct $\iff$ $f_i(z_1, \ldots, z_N) = 0$ for all i $\iff$ $(z_1, \ldots, z_N)$ is a solution of M_N.

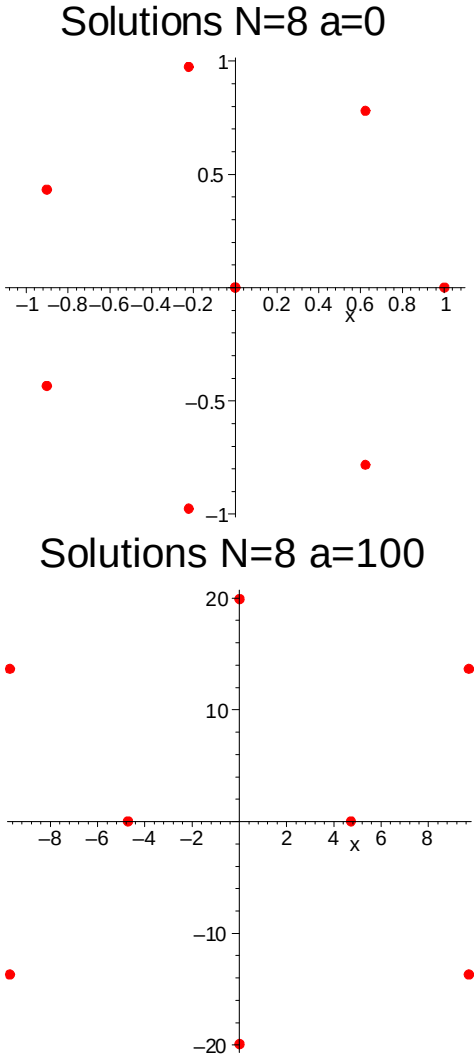
Let r be the remainder of dividing h by f, and let c_j, $1 \leq j \leq deg(r)$, be the coefficient of x^j in r. Then each c_j is a polynomial in the e_i and Theorem 12.1.5 implies the system $c_j(e_1, \ldots, e_N) = 0$, $1 \leq j \leq deg(r)$, is equivalent to M_N.

Computations with Singular (Greuel G.-M. and Pfister G. and Schoenemann H., 2002) using the formulation of Theorem 12.1.5 reveal a one-dimensional family of solution shapes for $N = 8$:

Proposition 12.1.2. *The coordinates of a solution to M_8 are given by the roots of the polynomial*

$$t^8 + \frac{28}{5}t^6 a + 14t^4 a^2 + 28t^2 a^3 - t - 7a^4,$$

where a is arbitrary. Setting $a = 0$, the roots form a regular heptagon with a point in the center. Varying a deforms this into irregular hexagons with two points in the interior.



This is a one dimensional family of solution shapes.

Proposition 12.1.3. *There are no solutions to M_N for $N = 3, 4, 6, 7, 9, 10, 11,$ and 12.*

Proof. For $N = 3, 4$, short (less than one minute by Maple on a Sun Ultra-5) Gröbner bases computations show that M'_N, hence M_N, has no solutions. For the remaining N, computations using one or both of the above reformulations show that there are no solutions of the equivalent systems. For $N > 7$ we

use FGb for the computations. When $N > 9$ another difficulty arises in the computation: it is impossible to compute the discriminant of $g = x^{N-2} - e_1x^{N-3} + e_2x^{N-4} - \cdots + (-1)^N e_{N-2}$. At the beginning we add only the condition $g(0) = e_{N-2} \neq 0$ and $g(1) \neq 0$ and we compute a lexicographical Gröbner basis. Finally we remove the bad solutions.

12.1.3 Regular solutions

The geometry of the solutions known thus far lead one to ask: What other regular polygons are solution shapes (with or without a point in the center)? What about two regular polygons, or n regular polygons? We use the notation $[n, m, p]$ to denote a solution shape consisting of n regular concentric m-gons and $p = 1$ or 0 as there is or is not a point in the center. Thus a solution $[n, m, p]$ will be a solution for M_{nm+p}. We begin this section by trying to find "by hand" some regular solutions and then give a more systematic way to find these solutions.

One regular m-gon: $[1, m, p]$. Since the solutions are invariant under translation and multiplication by complex numbers, it suffices to examine the m'th roots of unity.

The main lemma we need is

Lemma 12.1.2. *Let ω be a primitive m'th root of unity. Then*

$$\begin{aligned}
\sum_{j=1}^{m} \frac{1}{(\omega^j)^2} &= 0; \\
\sum_{j=1}^{m-1} \frac{1}{(\omega^j - 1)^2} &= -\frac{(m-1)(m-5)}{12}; \\
\sum_{j=1}^{m} \frac{1}{(\frac{a}{b}\omega^j - 1)^2} &= \frac{mb^m(b^m + a^m(m-1))}{(b^m - a^m)^2}.
\end{aligned} \tag{12.9}$$

Proof. From Lemma 12.1.1 we know

$$\sum_{j=1}^{N} \frac{1}{z_j^2} = \frac{e_{N-1}^2 - 2e_N e_{N-2}}{e_N^2},$$

where the e_i are the elementary symmetric polynomials in the z_j. The polynomials with roots ω^j $(1 \leq j \leq m)$, $\omega^j - 1$ $(1 \leq j \leq m-1)$, and $\frac{a}{b}\omega^j - 1$ $(1 \leq j \leq m)$ are

$$\begin{aligned}
&P(X) = X^m - 1 \text{ and} \\
&P(X) = \frac{(X+1)^m - 1}{X} = X^{m-1} + mX^{m-2} + \ldots + \binom{m}{3}X^2 + \binom{m}{2}X + m, \\
&P(X) = (X+1)^m - (\frac{a}{b})^m
\end{aligned}$$

(12.10)

respectively. Substituting in the corresponding values of e_N, e_{N-1} and e_{N-2} gives the result.

We first consider the case $p = 0$: $[1, m, 0]$, i.e., $N = m$. Let $z_i = \omega^i$ for all i, where ω is a primitive m'th root of unity. Then the i'th equation is

$$f_i = \sum_{j \neq i} \frac{1}{(z_i - z_j)^2} = \sum_{j \neq i} \frac{1}{(\omega^i - \omega^j)^2} = \frac{1}{(\omega^i)^2} \sum_{j \neq i} \frac{1}{(\omega^{j-i} - 1)^2} = \frac{1}{(\omega^i)^2} \sum_{j=1}^{m-1} \frac{1}{(\omega^j - 1)^2}.$$

By Lemma 12.1.2, for all i the i'th equation is zero if and only if

$$\sum_{j=1}^{m-1} \frac{1}{(\omega^j - 1)^2} = -\frac{(m-1)(m-5)}{12} = 0,$$

i.e., if and only if $m = 5$ or $m = 1$. Thus the regular pentagon is the only solution shape for this case.

If $p = 1$ we are looking at polygons of the shape $[1, m, 1]$ with $N = m + 1$ and then we have $z_i = \omega^i$ for $i = 1, \ldots, m$ and $z_{m+1} = 0$. Then the $m + 1$st equation

$$f_{m+1} = \sum_{j=1}^{m} \frac{1}{(\omega^j - 0)^2} = 0$$

by lemma 12.1.2. For $i = 1, \ldots, m$, the i'th equation is

$$f_i = \sum_{j \neq i} \frac{1}{(z_i - z_j)^2} = \sum_{j \neq i} \frac{1}{(\omega^i - \omega^j)^2} + \frac{1}{(\omega^i)^2} = \frac{1}{(\omega^i)^2} \left(\sum_{j \neq i} \frac{1}{(\omega^{j-i} - 1)^2} + 1 \right)$$
$$= \frac{1}{(\omega^i)^2} \left(\sum_{j=1}^{m-1} \frac{1}{(\omega^j - 1)^2} + 1 \right). \tag{12.11}$$

So for all $i = 1, \ldots, m$ the i'th equation is zero if and only if

$$\sum_{j=1}^{m-1} \frac{1}{(\omega^j - 1)^2} = -\frac{(m-1)(m-5)}{12} = -1,$$

i.e., $m = 7$ or $m = -1$. Therefore the regular heptagon with a point in the center is the only solution shape in this case.

Two regular m-gons [2, m, x]. Again we may fix one m-gon, P_1, to be the m'th roots of unity. We introduce a new complex variable, x, to describe the second m-gon, $P_2 = xP_1$, where multiplication of a polygon P with x means x times each vertex of the polygon.

Proposition 12.1.4. *There are no solution shapes of the form* $[2, m, 0]$ *or* $[2, m, 1]$.

Proof. We include in square brackets facts for the case $[2, m, 1]$. Let

$$z_i = \begin{cases} \omega^i & \text{if } i = 1, \ldots, m; \\ x\omega^i & \text{if } i = m+1, \ldots, 2m; \\ [0 & \text{if } i = 2m+1]. \end{cases} \tag{12.12}$$

Dividing by $(\frac{1}{\omega^i})^2$ in the i'th equation when $i = 1, \ldots, m$, or $(\frac{1}{x\omega^i})^2$ when $i = m+1, \ldots, 2m$, we get two equations in one unknown:

$$-\frac{(m-1)(m-5)}{12}[+1] + \frac{m(1+x^m(m-1))}{(1-x^m)^2} = 0 \tag{12.13}$$

$$-\frac{(m-1)(m-5)}{12}[+1] + \frac{mx^m(x^m+m-1)}{(x^m-1)^2} = 0, \tag{12.14}$$

where we have used the third part of lemma 12.1.2. Subtracting one equation from the other gives

$$1 - x^{2m} = 0, \tag{12.15}$$

so the solution set would have to consist of $2m$th roots of unities. But we have already seen that in the single polygon case the only solutions are $m = 5$ and $m = 7$, neither of which is divisible by two. Therefore no shapes of the form $[2, m, 0]$ or $[2, m, 1]$ can be a solution.

The Generalization. Using the differential equation of Theorem 12.1.5 we can find some more conditions not only for the case of regular polygons but for any set of roots to a polynomial $N(X)$. For the case of regular polygons this raises the chances of successful computations since we can add the new equations to our old systems.

Definition 12.1.1. *Let* N, M, P *be univariate polynomials of degree* n, m, p. *We use the notation* $[N, M, P]$ *to denote the set of solutions of* M_{nm+p} *with the shape* $P(X)N(M(X))$. *In the particular case* $P(X) = X^p$, $M(X) = X^m$ *we use the simplified notation* $[N, m, p]$.

Theorem 12.1.6. *Let* $N(x) = \sum_{i=0}^{n} a_i X^i$ *be a square free polynomial of degree* n *such that* $a_0 \neq 0$. *Then* $[N, m, p]$ *(with* $m > 1$*) is a solution of* M_{nm+p} *if and only if* $p \leq 1$ *and* $N(X)$ *divides*

$$\sum_{\substack{i=1\\j=1}}^{n} ija_ia_j\,(mi-1)\,(3\,mj+5-4\,mi)\,X^{i+j} \text{ if } p=0$$

and $N(X)$ *divides*

$$\sum_{\substack{i=0\\j=1}}^{n} a_ia_jjm(jm+1)(im+1)(3im-4jm+4)X^{i+j} \text{ if } p=1.$$

Proof. Let $f(X) = X^pN(X^m)$. We know from theorem 12.1.5 that f is a solution of M_{nm+p} if and only if f is square free and $U(X) = 3(f''^2 - 4f'f'''$ is divisible by $f(X)$. The first condition is true as soon as $p \leq 1$ since 0 is not a root of $N(X)$.

Considering the case $p = 0$, we find:

$$U(X) = 3\left(\sum_{i=1}^{n} im\,(mi-1)\,a_iX^{mi-2}\right)^2 - 4\left(\sum_{i=1}^{n} ima_iX^{mi-1}\right)\left(\sum_{i=i_3}^{n} im\,(mi-1)\,(mi-$$

where $i_3 = 2$ if $m = 2$ and $i_3 = 1$ else. Since X and $f(X)$ are relative prime, f divides U iff f divides $X^4U = V$ with

$$V(X) = 3\left(\sum_{i=1}^{n} im\,(mi-1)\,a_iX^{mi}\right)^2 - 4\left(\sum_{i=1}^{n} ima_iX^{mi}\right)\left(\sum_{i=i_3}^{n} im\,(mi-1)\,(mi-2)\,a_i$$

hence $V = W(X^m)$ is divisible by $N(X^m)$ iff $W(X)$ is divisible by $N(X)$. We can rewrite the sum:

$$W(X) = m^2\sum_{\substack{i=1\\j=1}}^{n} ija_ia_j\,(mi-1)\,(3\,mj+5-4\,mi)\,X^{i+j}.$$

We consider now the case $p = 1$ and find:

$$U(X) = 3\left(\sum_{i=1}^{n} a_i(im+1)(im)X^{im-1}\right)^2 - 4\left(\sum_{i=0}^{n} a_i(im+1)X^{im}\right)\left(\sum_{i=1}^{n} a_i(im+1)(im)\right.$$

must be divisible by X and $N(X^m)$, so that $m > 2$ and $V_1(X) = X^2U(X)$ should be divisible by $N(X^m)$.

$$V_1(X) = 3\left(\sum_{i=1}^{n} a_i(im+1)(im)X^{im}\right)^2 - 4\left(\sum_{i=0}^{n} a_i(im+1)X^{im}\right)\left(\sum_{i=1}^{n} a_i(im+1)(im)(i\right.$$

This equivalent to divisibility of

$$W_1(X) = 3\left(\sum_{i=1}^{n} a_i(im+1)(im)X^i\right)^2 - 4\left(\sum_{i=0}^{n} a_i(im+1)X^i\right)\left(\sum_{i=1}^{n} a_i(im+1)(im)(im$$
$$= \sum_{\substack{i=0\\j=1}}^{n} a_i a_j jm(jm+1)(im+1)(3im-4jm+4)X^{i+j}.$$

Note 12.1.1. We can always suppose that $N(X) = X^n + X^{n-1} + \sum_{i=0}^{n-2} a_i X^i$.

Note 12.1.2. In the following we give an explicit value to n and p and we consider m as a variable.

Corollary 12.1.1. *There are no solutions of the form* $[N, 2, 1]$.

Proof. From the proof of Theorem 12.1.6, $f(X) = XN(X)$ does not divide $U(X)$

because X does not divide $U(X)$.

Corollary 12.1.2. *For* $deg(N) = 1$, $[N, m, 0]$ *is a solution iff* $(m-1)(m-5) = 0$ *and* $N(X) = 1 + X$.

Proof. We apply the theorem 12.1.6 to $N = 1 + X$ and we find $W(X) = -X^2(m-1)(m-5)$.

Corollary 12.1.3. *For* $deg(N) = 1$, $[N, m, 1]$ *is a solution iff* $m = 7$ *and* $N(X) = 1 + X$.

Proof. We apply the theorem 12.1.6 to $N = 1 + X$ and we find

$$W_1(X) = X(-4 + 4m^2) + X^2(m^3 - 2m^2 - 7m - 4)$$

and the remainder of W_1 divided by N should be zero:

$$-m(-7+m)(m+1)x.$$

Corollary 12.1.4. $deg(N) = 2$, $[N, m, 0]$ *there is no solution.*

Proof. We apply the theorem 12.1.6 to $N = a_0 + X + X^2$ and we find

$$W(X) = -\left(4\,(2\,m-1)\,(2\,m-5)\,X^2 + 4\,(m-1)\,(4\,m-5)\,X + (m-1)\,(m-5)\right)X^2$$

and the remainder of W divided by N should be zero:

$$-\left(-5 - m^2 + 18\,m - 60\,a_0 m + 16\,a_0 m^2 + 20\,a_0\right)X$$
$$-\left(-m^2 + 18\,m - 5 + 16\,a_0 m^2 - 48\,a_0 m + 20\,a_0\right)a_0 = 0.$$

We can can compute a lexicographical Gröbner of the coefficients:

$$\left[20\,a_0 - m^2 + 18\,m - 5, m\left(m^2 - 18\,m + 5\right)\right]$$

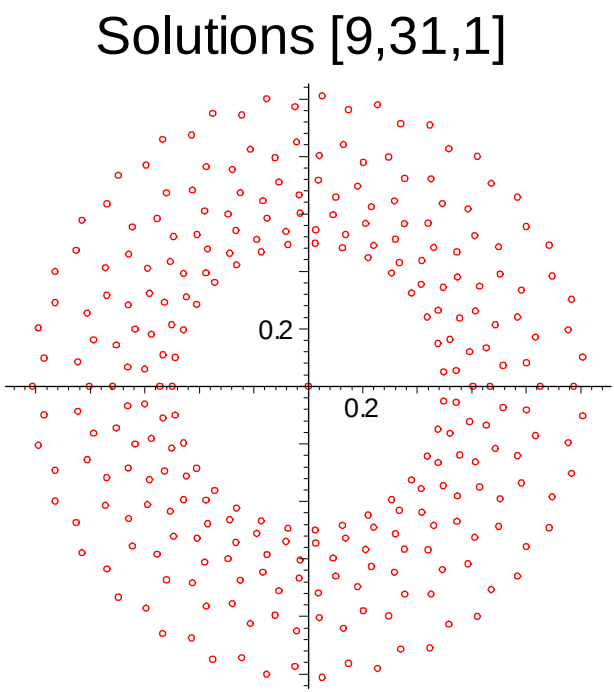
and the number of solutions is 0.

Summary of the regular solutions

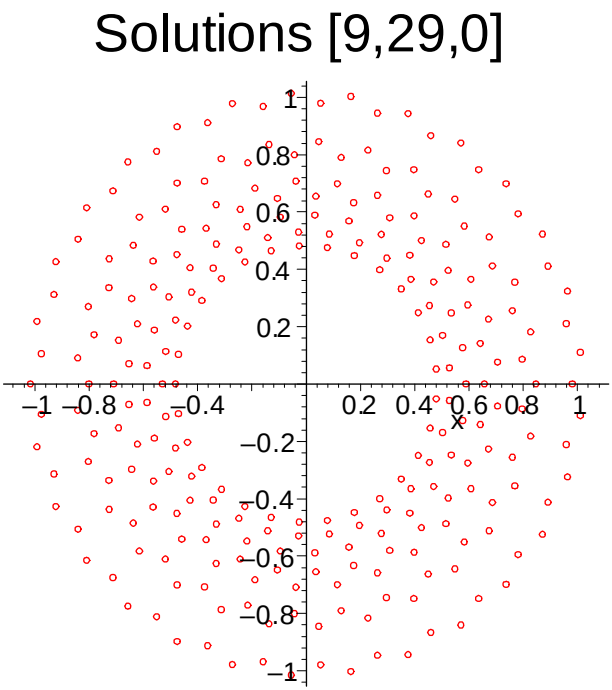
An extended version of this paper including a complete list of solutions, pictures and all the polynomials can be found at `http://calfor.lip6.fr/~jcf/MICE/m`
We summarize all the results:

Theorem 12.1.7. *For fixed values of n and p we give all the possible values of m and for each m all the solutions $[n, m, p]$. The results are summarized in the following table.*

Shape	*Values of m*	*Values of N*
$[1, m, 0]$	$m = 5$	$N = 5$
$[1, m, 1]$	$m = 7$	$N = 8$
$[2, m, 0]$	$\emptyset$	
$[2, m, 1]$	$\emptyset$	
$[3, m, 0]$	$m = 7,\ m = 11$	$N = 21,\ N = 33$
$[3, m, 1]$	$m = 5,\ m = 13$	$N = 16,\ N = 40$
$[4, m, 0]$	$m = 2,\ m = 4$	$N = 8,\ N = 16$
$[4, m, 1]$	$m = 5$	$N = 21$
$[5, m, 0]$	$m = 13,\ m = 17$	$N = 65,\ N = 85$
$[5, m, 1]$	$m = 11,\ m = 19$	$N = 56,\ N = 96$
$[6, m, 0]$	$\emptyset$	
$[6, m, 1]$	$\emptyset$	
$[7, m, 0]$	$m = 19,\ m = 23$	$N = 133,\ N = 161$
$[7, m, 1]$	$m = 17,\ m = 25$	$N = 119,\ N = 175$
$[8, m, 0]$	$m = 5,\ m = 7$	$N = 40,\ N = 56$
$[8, m, 1]$	$m = 4,\ m = 8$	$N = 33,\ N = 65$
$[9, m, 0]$	$m = 25,\ m = 29$	$N = 225,\ N = 261$
$[9, m, 1]$	$m = 23,\ m = 31$	$N = 208,\ N = 280$



one regular solution for $N = 280$.



Corollary 12.1.5. *Using this information we could find the following solution families:*

$$f(x) = -32\lambda\, x^5 + \lambda + x^{16} + x^{11} + \frac{11}{8}x^6 - \frac{11}{128}x$$

$$f(x) = -\lambda\, x + 25\,\lambda\, x^8 + x^{21} + x^{14} - \frac{13}{10}\,x^7 + \frac{13}{400}$$

for $N = 16$ *and* $N = 21$.

Conjecture 12.1.1. For n odd, there will be solutions for $[n, m, 0]$ with $m = 3n - 2$ and $m = 3n + 2$ and for $[n, m, 1]$ with $m = 3n - 4, m = 3n + 4$.

We have a new application of computer algebra in biological physics. We were able to solve the system completely up to $N = 12$ using the symmetry and the most recent techniques for the Gröbner bases computation. Starting with solution shapes of regular polygons we found solution families for $N = 8, 16, 21$ as well as single solutions for N up to 280 for which we have reason to assume that they are part of solution families as well.

From the biophysical point of view, solutions for N about 1000 are needed since there are thousands of proteins in a cell membrane. But even small numbers of proteins can give some interesting insights. We have extended the results in the original paper (Kim *et al.* , 1998) from $N = 5$ to 12.

This work is a particular instance of the more general problem of finding a global minimum of an energy function and in particular we want to point out similar work related to the classification of the stable solutions of the n body problem.

13. Codes Correcteurs

En collaboration avec D. Augot et M. Bardet ((Bardet, 2004; D. *et al.* , 2003) et le mémoire HDR de D. Augot).

Un autre problème traité par les techniques de bases de Gröbner est le problème du décodage des codes cycliques au-delà de la distance minimale (avec D. Augot et M. Bardet). *A priori* la correction des codes ne relève pas de la cryptographie, mais les techniques algébriques sont identiques: en effet, le problème du décodage des codes cycliques peut se récrire en un système algébrique sur un corps fini de caractéristique 2 (comme AES) dont les solutions sont étroitement liées à l'erreur qui s'est produite. Les travaux précédents ont montré que le calcul d'une base de Gröbner de ce système algébrique permet de décoder jusqu'à la distance minimale du code. Le calcul de base de Gröbner peut se faire soit en prétraitement (décodage formel), les paramètres étant considérés comme des variables, soit pour chaque mot à décoder (décodage en ligne), en calculant pour chaque mot les paramètres et en les substituant dans le système. Dans le cas du décodage formel, il a été montré qu'il est possible d'obtenir à partir de la base de Gröbner formelle des formules de décodage pour les coefficients du polynôme localisateur. Malheureusement, il devient rapidement impossible de calculer cette base de Gröbner formelle, même pour des codes de petite longueur.

Motivés par le problème du décodage des codes à résidus quadratiques (codes RQ), pour lesquels il n'existe aucun algorithme général de décodage actuellement, nous améliorons les résultats sur plusieurs points. D'abord, nous introduisons des systèmes modifiés, sans équation de degré élevé, pour lesquels le calcul de la base de Gröbner est plus facile. Ceci permet de calculer la base de Gröbner formelle pour des codes de plus grande longueur.

Nous montrons sur l'exemple du code RQ de type [41,21,9] que les formules deviennent rapidement de grande taille, et sont donc inutilisables pour le décodage. Cela indique que les efforts développés pour le décodage algébrique des codes cycliques par formules ont peu de chance d'aboutir. L'autre approche (décodage en ligne) est plus efficace d'un point de vue informatique. En utilisant des méthodes générales de compilation pour des systèmes avec paramètres, nous améliorons l'efficacité des calculs. Nous donnons de nombreux exemples (codes BCH de longueur 75, 511, codes RQ de

longueur 41, 73, 89, 113, et un code de longueur 75 qui n'appartient à aucune classe connue de codes cycliques).

Cette technique de décodage des codes cycliques avec des bases de Gröbner s'applique à n'importe quel code, est entièrement automatique, et permet de décoder au-delà de la capacité de correction du code.

14. Robotique

En collaboration avec D. Lazard (Faugère J.C. and Lazard D., 1995), F. Rouillier (Faugère J.C. and Rouillier F., 2005; Faugère & Rouillier, 2005) et J.P. Merlet (Faugère J.C. and Merlet J.P. and Rouillier F., 2007). Ouvrage de reference (Merlet J.P., 1990)

15. Ridges

16. Synthèse de bancs de filtres et ondelettes bidimensionnels

En collaboration avec F. Rouillier et F. Moreau de Saint Martin (tiré des articles (Faugère J.C. and Moreau de Saint-Martin F. and Rouillier F., 1996; Faugère J.C. and Moreau de Saint-Martin F. and Rouillier F., 1998)).

16.1 Contexte générale de l'application

16.1.1 Introduction

Le but de ce travail est d'étudier de nouvelles techniques de synthèse de bancs de filtres et ondelettes bidimensionnels. On s'attache plus particulièrement au cas de l'échantillonnage $2I$ qui consiste à décomposer à chaque étape le signal en 4 sous-signaux. Cette configuration présente les avantages suivants :

- Comparaisons faciles avec les systèmes "séparables" couramment utilisés ;
- Simplicité d'insertion dans un schéma de compression d'images ;
- Possibilité de disposer simultanément de l'orthogonalité et de la phase linéaire.

La première phase consiste en une étude approfondie d'une famille paramétrée d'ondelettes orthogonales et à phase linéaire proposée dans (Kovacevic & Vetterli, 1995).

16.2 Bancs de filtres et ondelettes

16.2.1 Éléments de traitement numérique du signal

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 strandard en pratique (transmission de données, compression, ...).

La theorie de l'information permet de dire que sous certaines conditions on peut 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 par conséquent 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 (suffisement 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 polyôme de Laurent.

La transformée de Fourier d'une convolution étant égale au produit des transformées de Fourier (lorque les fonctions sont suffisement régulières), lorsque l'on considère les représentations fréquentielles, l'image de $\hat{X}(z)$ par un filtre linéraire 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. $1/2 + n$, $n \in \mathbb{Z}$).

16.2.2 Bancs de filtres et ondelettes monodimensionnels

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* (Pour la frequence 0, la reponse du filtre est nulle).

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 par exemple considéré comme un shéma de base pour la compression de données si on suppose, par exemple, nulle 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).

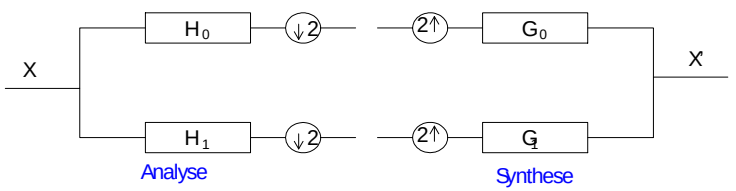


Fig 16.2.2: Décomposition par des filtres *passe-bande*

La figure 16.2.2 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 :

$$\begin{aligned} \hat{X}' = {} & 1/2 G_0(Z)[\hat{X}(Z)H_0(Z) + \hat{X}(-Z)H_0(-Z)] \\ & +1/2 G_1(Z)[\hat{X}(Z)H_1(Z) + \hat{X}(-Z)H_1(-Z)] \end{aligned},$$

cette condition s'exprime simplement en annulant le coefficient de $\hat{X}(-Z)$ dans :

$$\hat{X}' = 1/2\hat{X}(Z)[G_0(Z)H_0(Z) + G_1(Z)H_1(Z)] + \\ 1/2\hat{X}(-Z)[G_0(Z)H_0(-Z) + G_1(Z)H_1(-Z)]$$

En posant :

$$G_0(Z)H_0(-Z) + G_1(Z)H_1(-Z) = 0 \,, \tag{16.1}$$

nous obtenons, à un facteur retard près (terme de la forme Z^{-d} , d impair) :

$$G_1(Z) = H_0(-Z) \;\; H_1(Z) = -G_0(-Z) \tag{16.2}$$

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 (16.1) 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 16.2.2 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 16.2.2) peut être itérée comme décrit par la figure **??**, on définit ainsi un *banc de filtres itérés*.

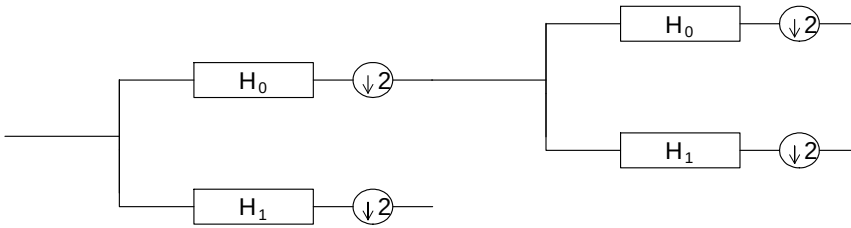


Fig **??**: Itération de filtres *passe-bande*

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 **??**, où, dans le cas de k itérations, le filtre H s'exprimera simplement par :

$$H(z) = H_0(Z)H_0(Z^2)\dots H_0(Z^{2k-1})$$

La fonction limite du processus itéré établit alors le lien entre l'étude du banc de filtres itérés et la théorie des *ondelettes*.

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 selon le cas étudié.

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 :

$$H_0(z) = \begin{bmatrix} \cos\theta_0 & \sin\theta_0 \\ -\sin\theta_0 & \cos\theta_0 \end{bmatrix} \left(\prod_{k=1}^{K-1} \begin{bmatrix} 1 & 0 \\ 1 & z^2 \end{bmatrix} \begin{bmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{bmatrix} \right) \begin{bmatrix} 1 \\ z \end{bmatrix} \quad (16.3)$$

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.

16.3 Ondelettes orthogonales pour l'échantillonnage $2I$

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 $1D$.

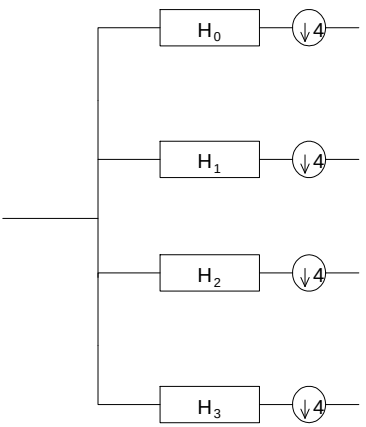
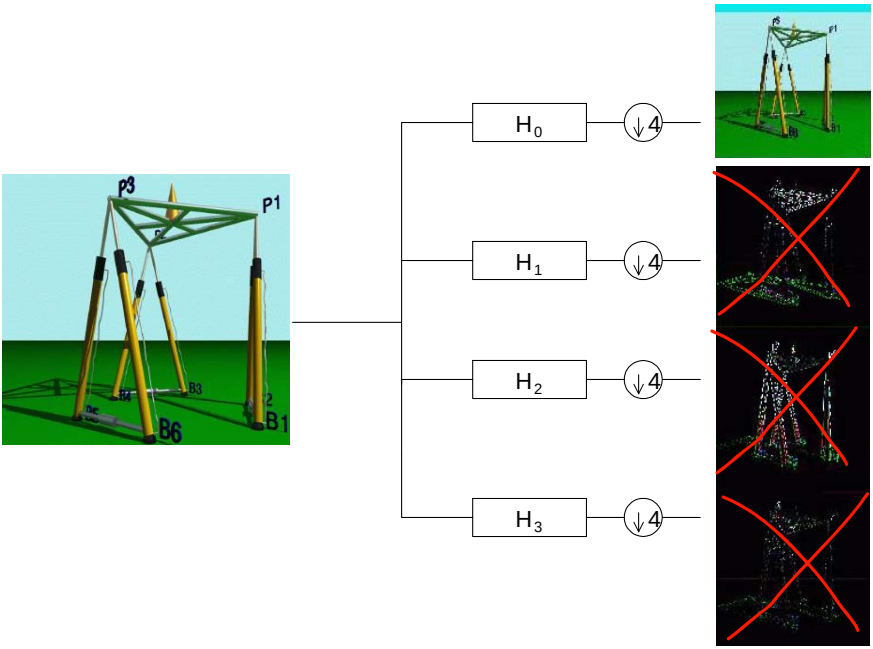


Fig 16.3: Décomposition d'un signal $2D$

Pour compresser une image on applique un filtre **??** et on ne garde que la composante principale H_0:



Si l'on se place dans le cadre d'un échantillonnage de type $2I$ (système à 4 sous bandes séparable), la figure 16.3 résume une situation que nous pouvons décrire de la façon suivante :

- La représentation temporelle d'un signal $2D$ est la donnée d'une suite $X = (x_{i,j})_{(i,j)\in\mathbb{Z}^2}$ déléments 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'epression de $\hat{X}$.
- Dans le cas $2D$, la notion de filtre *passe-bas* est similaire à celle du cas $1D$ (réponse en frequence faible sauf autour des points entiers), la définition d'un filtre *passe-haut* n'ayant, en general, pas de sens precis.

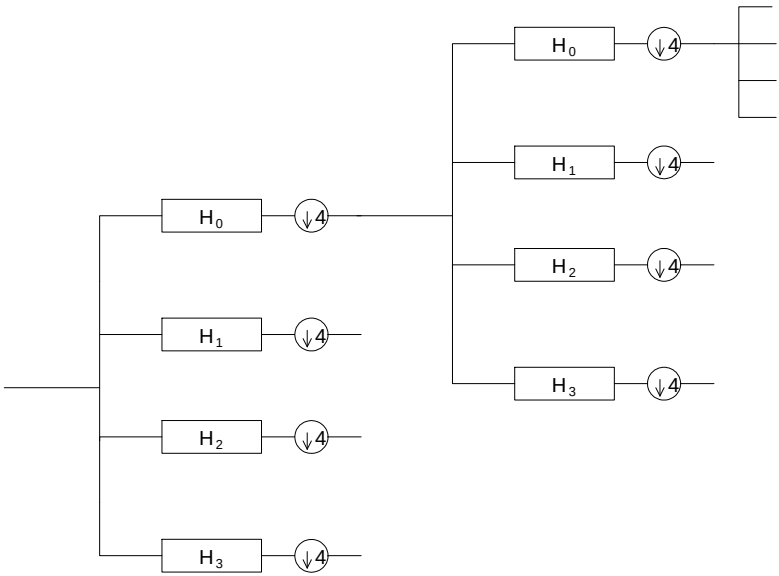


Fig 16.3: Itération de filtres *passe-bande*

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 16.3), 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, lorque $k \to \infty$, établit le lien avec la théorie des *ondelettes*.

Dans le cas $2D$, nous n'avons plus de description complète de l'espace (simplement paramètrée) des filtres à étudier et nous limitons l'étude à celle d'une cascade proposée par (Kovacevic & Vetterli, 1995) 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:

$$R_i = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i & 0 & 0 \\ \sin\alpha_i & \cos\alpha_i & 0 & 0 \\ 0 & 0 & \cos\beta_i & -\sin\beta_i \\ 0 & 0 & \sin\beta_i & \cos\beta_i \end{bmatrix} \tag{16.4}$$

$$W = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{16.5}$$

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{16.6}$$

$$D(z_1, z_2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & z_1 & 0 & 0 \\ 0 & 0 & z_2 & 0 \\ 0 & 0 & 0 & z_1 z_2 \end{bmatrix} \tag{16.7}$$

Avec ces définitions on construit des matrices MH_0 en posant :

$$MH_0(z) = R_0 W P \prod_{i=1}^{k} D(z_1, z_2) P W R_i W P \tag{16.8}$$

et on considère les filtres H_0 définis par :

$$H_0(z) = H_{0,0}(z_1^2, z_2^2) + H_{0,1}(z_1^2, z_2^2) z_1 + H_{0,2}(z_1^2, z_2^2) z_2 + H_{0,3}(z_1^2, z_2^2) z_1 z_2 \tag{16.9}$$

(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 :

- régularite de la fonction limite du processus de subdivision. Pour cela il faut imposer un certain nombre de proprietes 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 caractérise l'ordre de platitude du filtre, cela signifie que les équations suivantes sont satisfaites:

$$\forall k_1, k_2,\ k_1 + k_2 \leq N, \quad \frac{\partial^{k_1+k_2} H_0}{\partial z_1^{k_1} \partial z_2^{k_2}}(-1,-1) = 0, \tag{16.10}$$

$$\frac{\partial^{k_1+k_2} H_0}{\partial z_1^{k_1} \partial z_2^{k_2}}(-1,1) = 0, \tag{16.11}$$

$$\frac{\partial^{k_1+k_2} H_0}{\partial z_1^{k_1} \partial z_2^{k_2}}(1,-1) = 0 \tag{16.12}$$

- Le plus possible de composantes nulles sur les bandes hautes frequences. 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 équations.

Notre but sera par conséquent, pour K fixé, de trouver la valeur maximale de N telle que le système constitué de ces équations admette des solutions réelles.

16.4 Synthèses pour la cascade proposée par Kovacevic et Vetterli

16.4.1 Étude de l'exemple obtenu par Kovacevic et Vetterli

Notre première démarche a été de vérifier que nous pouvions retrouver des résulats connus pour une valeur maximale de N avant de nous lancer dans des cas nouveaux.

Pour l'instant, seul le cas $K = 3$ à été traité dans la littérature (voir (Kovacevic & Vetterli, 1995)), pour la cascade de Vetterli-Kovacevic et une seule famille de filtres à été exhibée (correspondant à $N = 2$). Les figures **??** et **??** illustrent respectivement la représentation fréquentielle du filtre H_0 obtenu et le filtre itéré limite induit.

Notre première idée a été de vérifier que le problème n'admettait pas de solutions pour $N = 3$. Ceci à été simplement vérifiable en calculant la base de Groebner du système correspondant. Cette dernière étant égale à [1], le syst'eme n'admettait donc pas de solutions complexes et a fortiori pas de solutions réelles. Notons que c'est le cas même si on ne considère que les équations portant sur le filtre H_0.

Nous nous avons alors fixé $N = 2$ et obtenu un système zéro-dimensionnel. 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, et non numérique, les coefficients des polynômes les définissant, et par conséquent les coordonnées du système étudié. En remarquant que certaines de celles-ci semblaient algébriques, 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 suffisement importantes pour que les algorithmes de factorisation ne puissent effectuer l'opération rapidement.

Lorsqu'un idéal n'est pas en position générique, le calcul d'ensembles triangulaires permet, en général de décomposer un système 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 de la 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 déja connue, les caractéristique du nouveau filtre étant données par les figures **??** et **??**.

Ce nouvel exemple de filtre n'apporte pas de réel progrès en pratique puisque la figure **??** 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'excèdent pas quelques secondes (base de Groebner comprise).

16.5 Recherche de la platitude maximale pour une taille donnée

Aucun exemple n'est connu à ce jour 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 momment les valeurs maximales de N pour $K = 4, 5, 6, 7, 8$, mais aucune des valeurs de N 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 $K = 8,\ N = 6$ occupe plusieurs dizaines de Méga-octets sur disque.

On obtient donc les résultats suivants :

K	N maximal	dimension
3	2	0
4	2	4
5	3	2
6	3	
7		
8		

Ces résultats ont été obtenus par l'application directe des techiques de calcul de bases de Groebner et de recherche des zéros réels de systèmes polynômiaux jusqu'à $K = 6$. Ensuite on a fait appel au changement de variables présenté dans le paragraphe suivant.

16.5.1 Changements de variables

Nous présentons ici le changement de variables qui permet de simplifier considérablement les équations. Ces changements de variables sont dûs à Jean-Charles Faugère.

Le principe du premier changement de variable, c'est de remplacer des matrices réelles 4×4 par des matrices complexes 2×2. Pour que des transformations linéaires de $\mathbb{R}^4$ puissent s'identifier à des transformations linéaires de $\mathbb{C}^2$ il faut qu'elles soient de déterminant positif. Ce n'est pas le cas de toutes les matrices du produit, donc on les regroupe avec leurs voisines pour les ramener à des matrices de déterminant positif.

Plus précisément, c'est la matrice P qui pose problème. On considère donc:

$$PD(z_1, z_2)P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & z_1 & 0 & 0 \\ 0 & 0 & z_1 z_2 & 0 \\ 0 & 0 & 0 & z_2 \end{bmatrix} \tag{16.13}$$

De même, on considère :

$$P \begin{bmatrix} 1 \\ z_1 \\ z_2 \\ z_1 z_2 \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ z_1 z_2 \\ z_2 \end{bmatrix} \tag{16.14}$$

On peut alors considérer, au lieu de

$$H(z) \begin{bmatrix} 1 \\ z_1 \\ z_2 \\ z_1 z_2 \end{bmatrix} \tag{16.15}$$

la cascade de traitements suivante : on part du vecteur

$$u_0 = \begin{bmatrix} 1 + iz_1 \\ (z_1 + i)z_2 \end{bmatrix} \tag{16.16}$$

et on lit dans l'ordre le produit de matrices. Quand on lit W, on multiplie le vecteur courant par

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{16.17}$$

Quand on lit R_l, on multiplie le vecteur courant par :

$$\begin{bmatrix} e^{i\alpha_l} & 0 \\ 0 & e^{i\beta_l} \end{bmatrix} \tag{16.18}$$

Quand on lit $PD(z_1, z_2)P$, on applique le traitement suivant au vecteur courant $u = (u(1), u(2))'$:

$$u := \begin{bmatrix} \mathcal{R}(u(1)) + iz_1\mathcal{I}(u(1)) \\ z_1z_2\mathcal{R}(u(2)) + iz_2\mathcal{I}(u(2)) \end{bmatrix} \tag{16.19}$$

Cette procédure permet de diviser le nombre de variables par 2. On obtient alors quelque chose qui s'écrit comme un polynôme en $e^{i\alpha}$, $e^{-i\alpha}$, $e^{i\beta}$, $e^{-i\beta}$, mais en plus de nouvelles simplifications apparaissent quand on utilise les identités sur les exponentielles, notamment grâce au fait que l'on calcule les dérivées en des points particuliers.

Plus précisément, une première astuce de calcul consiste à calculer des développements limités, ce qui coûte bien moins cher que de calculer des dérivées avant de les évaluer en certains points. Cela permet en particulier d'écrire directement les équations à satisfaire sans même développer les filtres complètement. Comme on regarde des dérivées en (-1,-1),(1,-1), (1,1) et (-1,1), on observe que tout peut s'écrire comme somme de termes du type $\exp(i\alpha_0 + \beta_1 + \alpha_2 + \alpha_3)$.

Les changements de variables ulterieurs permettent notamment dans les cas (5,3) et (7,4) de donner une expression explicite des inconnues en fonction de deux paramètres. Cela permet d'écrire tous les coefficients de tous les filtres comme des fonctions explicites des deux degrés de liberté restant.

16.5.2 Optimisation des degrés de liberté résiduels

De nombreux critères peuvent être imaginés pour l'optimisation des degrés de libertés restant :

- Imposer la réponse fréquentielle de certains filtres en certains points ;
- Minimiser le manque de symétrie de certains filtres ;
- Minimiser l'énergie de certains filtres dans certaines gammes de fréquences ;

- Maximiser un critère assez connu en traitement du signal dit "gain de codage sur modèle auto-régressif d'ordre 1" ; On parlera ici plutôt de concentration de l'énergie, mais c'est comparable.
- Minimiser le rayon spectral d'une matrice pour maximiser la régularité.

Réponse fréquentielle :. Imposer la réponse fréquentielle en certains points est a priori un critère un peu arbitraire. Il existe cependant des points qui jouent un rôle particulier. Ce sont par exemple les points $(2m\pi/3, 2n\pi/3)$, pour la raison suivante.

> Pour comprendre le phénomène, on peut plus légèrement considérer le cas mondimensionnel, où un rôle particulier est joué par le point $2\pi/3$.

On considère la transformée de Fourier de la fonction limite du processus de subdivision. On peut écrire :

$$\hat{\phi}\left(\frac{2^{k+1}\pi}{3}\right) = G_0\left(\frac{2^k\pi}{3}\right)\cdots G_0\left(\frac{2\pi}{3}\right)\hat{\phi}\left(\frac{2\pi}{3}\right) \tag{16.20}$$

Donc, pour tout α :

$$\left(|\omega|^\alpha\,\hat{\phi}\right)\left(2^{2k+1}\frac{2\pi}{3}\right) = \left(\frac{4\pi}{3}\right)^\alpha \hat{\phi}\left(\frac{2\pi}{3}\right) 2^\alpha \left(2^\alpha G_0\left(\frac{2\pi}{3}\right)\right)^{2k} \tag{16.21}$$

D'après un résultat de Cohen et Conze [1], pour que la fonction limite soit α-hölderienne [2], il est nécessaire que $\hat{\phi}$ soit uniformément bornée, donc que :

$$\left|G_0\left(\frac{2\pi}{3}\right)\right| \leq 2^{-\alpha} \tag{16.22}$$

Cela justifie que l'on souhaite particulièrement contrôler ce point.

En pratique, on intègre facilement les équations correspondant aux points $(2\pi/3, 0)$ et $(0, 2\pi/3)$ dans l'optimisation des degrés de liberté résiduels. Cependant les résultats en sont décevants. Pour un système (5,3), où il reste deux degrés de liberté explicites, cela conduit à une optimisation numérique simple.

En fréquence, la contrainte se traduit par une remontée importante des lobes secondaires, et du point de vue de la régularité, l'exposant de Sobolev n'est pas très bon.

Symétrie des filtres. Dans le cas de cette cascade, deux filtres sont structurellement symétriques par rapport au centre de leur support, et les deux autres sont antisymétriques. Pour accroître la symétrie des filtres, on cherche donc à ce que les deux filtres symétriques par rapport au centre soient aussi symétriques par rapport aux axes de symétrie du support que possible. C'est une fonctionnelle quadratique élémentaire en les coefficients des filtres. Le problème reste donc polynômial.

[1] NDLR: J. P. Conze est à Rennes I et A. Cohen à Paris VI: longue tradition de fructueuses collaborations...

[2] Cela donne un sens à C^α pour α non entier.

Critère fréquentiel et "gain de codage". Il s'agit de minimiser l'énergie du filtre passe-bas sur une partie du plan fréquentiel. Cette partie est en général définie par un paramètre arbitraire. Ce paramètre étant choisi, la fonction à minimiser est simplement quadratique en les coefficients des filtres.

Le critère dit de "concentration de l'énergie" s'écrit lui aussi comme une fonction quadratique des coefficients des filtres, dépendant aussi d'un paramètre arbitraire.

16.6 Suite

A major tool for the representation of 2-D signals such as images is the 4-band perfect reconstruction filter bank. Basically, it consists of four 2-dimensional filters H_0, H_1, H_2, H_3. The input signal is filtered by each filter separately, and each filtered signal is downsampled by two horizontally and vertically, thus generating 4 subsignals, that can be considered as a non-redundant representation of the original signal. This paper addresses the issue of designing these filters. The most common approach consists in designing separable filters, i.e. 2-D filters obtained by tensor product of 1-D filters, since many design techniques are available for 1-D perfect reconstruction filter banks.

In the design, since a perfect reconstruction filter bank implements signal decomposition onto a basis, we want this basis to be orthogonal : it also provides the energy preservation property between space and transform domains. It is also desirable, in image processing, to use linear-phase filters, which means (at least) centro-symmetric or centro-antisymmetric. A major point, at this stage, is that these two properties cannot be simultaneously achieved by separable filter banks, except in the Haar case. Thus, most filter banks used until now are separable and either orthogonal or linear-phase. Since we want the filter bank to yield both properties, we shall consider nonseparable filter banks in this paper. It is already known that nonseparable orthogonal linear-phase filter banks exist (Kovacevic & Vetterli, 1995; Stanhill & Zeevi, 1995a), but few design examples are available. Actually, the major design techniques for orthogonal nonseparable filter banks (not necessarily linear-phase) are the following :

- Straightforward formulation of the design as an optimization of the filters' coefficients under the quadratic constraints for orthogonality (implying perfect-reconstruction) (Viscito & Allebach, 1991).
- Optimization of cascade forms ensuring orthogonality structurally (Kovacevic & Vetterli, 1995; Stanhill & Zeevi, 1995b; Stanhill & Zeevi, 1995a). It is to be noticed that no complete cascade is until now available in the multidimensional case, due to the lack of a factorization theorem. The main difficulty here appears when trying to optimize the parameters of the cascade.

- State-space matrix representations (Venkataranam & Levy, 1994b) look to be a very promising approach to the design of orthogonal multidimensional filter banks (Venkataranam & Levy, 1995).

The design technique we propose derives from the cascade form approaches. It consists in looking at the optimization issue as solving a set of polynomial equations and in solving these equations using the computer algebra techniques known as Gröbner bases.

It is well known that such filter banks may generate wavelet bases (Cohen & Daubechies, 1993; Kovacevic & Vetterli, 1992). In this case, the scaling function ϕ is the limit of the subdivision process based on H_0 and satisfies the two-scale equation :

$$\phi(x,y) = \sum_{i,j} \phi(2x-i, 2y-j) H_0(i,j) \tag{16.23}$$

The wavelets are linear combinations of integer translates of ϕ :

$$\psi_1(x,y) = \sum_{i,j} \phi(2x-i, 2y-j) H_1(i,j) \tag{16.24}$$

$$\psi_2(x,y) = \sum_{i,j} \phi(2x-i, 2y-j) H_2(i,j) \tag{16.25}$$

$$\psi_3(x,y) = \sum_{i,j} \phi(2x-i, 2y-j) H_3(i,j) \tag{16.26}$$

It can be proven (Cohen *et al.* , 1995) that the resulting wavelets can exist and be $N-1$ times continuously differentiable only if the polynomials $H_0, \dots H_3$ vanish as well as their derivatives up to order N at these aliasing frequencies (of flatness order N. It is known for 1-D dyadic systems that, in practice, imposing these vanishing moments is an efficient way to obtain some regularity (Daubechies, 1992), so that we expect that designing maximally flat non-separable filter banks will provide regular wavelet bases.

More precisely, we consider a particular family of nonseparable filter banks for sampling matrix $2I$, holding structurally orthogonality and phase-linearity. This family (Kovacevic & Vetterli, 1995) is defined by polynomial matrix products including some angles that can be chosen arbitrarily. The coefficients of H_0, H_1, H_2, H_3, seen as polynomials in variables z_1 and z_2 are thus polynomials in terms of the cosines and sines of the angle parameters. The flatness equations for the polynomials $H_0, \dots H_3$ are linear combinations of the coefficients of the polynomials $H_0, \dots H_3$. Thus, the resulting system is polynomial w.r.t. sines and cosines. This shows the principle leading to use techniques for solving polynomial systems in this signal processing context. It should be emphasized that the application of Gröbner basis techniques to filter bank design is very different from (Park *et al.* , 1996).

In practice the straightforward application of existing algorithms to the polynomial system obtained by writing the flatness equations in terms of the

cascade parameters cannot design filters with support larger than 6×6 and two degrees of flatness. In order to design filter banks with higher regularity, we propose a substitution of variables, splitting the system into two smaller subsystems, which makes possible the design of filters with support up to 16×16 and five degrees of flatness.

The paper is then divided as follows. In section 16.7, we present the cascade form we use in the sequel, borrowed from (Kovacevic & Vetterli, 1995). We show how the issue of maximizing the flatness of filters of given size can be turned into solving a polynomial system. Section 16.7.1 is devoted to the estimation of the regularity of 2-D wavelets. The algorithm derives from mathematical results (Cohen *et al.* , 1995), and will be used as an analysis tool for the resulting wavelets and filter banks. In section **??**, we briefly present computer algebra tools for solving polynomial systems, the Gröbner basis notion, and the algorithms that are currently available. Using these tools, we then present, in section 16.8, a substitution of variables and a strategy to solve the problem. This provides the minimal size for achieving a given flatness order, the number of remaining free degrees, and a parametrization of the family of the corresponding filters for this flatness order. Examples for different filters' sizes are described in section 16.9 : they are obtained through an optimization of the remaining free degrees and we describe the resulting filter banks in terms of regularity, frequency characteristics and performances in image compression.

16.7 Descriptiom mathématique du problème

The principle of our approach consists in considering a cascade form ensuring structurally orthogonality and linear-phase and including some degrees of freedom which we optimize so as to maximize the flatness. Straightforward approaches might have been considered, but writing the orthogonality equations in the space domain leads to a system with very high numbers of variables and equations. This cascade form let us reduce the number of variables, and allows to choose them freely, although making any optimization a non-linear problem. We now describe the problem in more detail. This cascade is not complete (i.e : not every orthogonal linear phase filter bank can be written under this form). We first define this family of filter banks. Let us define

$$R_i = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i & 0 & 0 \\ \sin\alpha_i & \cos\alpha_i & 0 & 0 \\ 0 & 0 & \cos\beta_i & -\sin\beta_i \\ 0 & 0 & \sin\beta_i & \cos\beta_i \end{bmatrix} \qquad W = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{16.27}$$

$$P = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,0\,1 \\ 0\,0\,1\,0 \end{bmatrix} \qquad D(z_1, z_2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & z_1 & 0 & 0 \\ 0 & 0 & z_2 & 0 \\ 0 & 0 & 0 & z_1 z_2 \end{bmatrix} \tag{16.28}$$

$$\mathcal{H}(z) = R_1 W P \prod_{i=2}^{K} D(z_1, z_2) P W R_i W P \tag{16.29}$$

The filters $H_0 \dots H_3$ are defined as

$$H_i(z) = \mathcal{H}_{i,0}(z_1^2, z_2^2) + \mathcal{H}_{i,1}(z_1^2, z_2^2) z_1 + \mathcal{H}_{i,2}(z_1^2, z_2^2) z_2 + \mathcal{H}_{i,3}(z_1^2, z_2^2) z_1 z_2 \tag{16.30}$$

where $\mathcal{H}_{i,j}$ denotes the (i, j) component of the matrix $\mathcal{H}$.

It is easy to check that the resulting filters H_0 and H_1 are centrosymmetric while H_2 and H_3 are centro-antisymmetric. Each filter then has linear phase and a square support of size $2K \times 2K$. It can also be checked that the filter bank is orthogonal, meaning that we represent the input signal on an orthogonal basis. Orthogonality implies that the analysis–synthesis system is perfect reconstructing (when there is no quantization of the subband signals). These properties are ensured structurally, whatever K might be. For given K, we have to choose the angles $\alpha_1, \dots \alpha_K$ and $\beta_1, \dots, \beta_K$. We aim at having maximally flat filters, while flatness of order N for the filters $H_0 \dots H_3$ at given points writes as follows : for all $k_1, k_2, k_1 + k_2 \leq N$,

- $\frac{\partial^{k_1+k_2} H_0}{\partial z_1^{k_1} \partial z_2^{k_2}}$ vanishes at $(1, -1)$, $(-1, -1)$, $(-1, 1)$;
- $\frac{\partial^{k_1+k_2} H_1}{\partial z_1^{k_1} \partial z_2^{k_2}}$ vanishes at $(1, -1)$, $(1, 1)$, $(-1, 1)$;
- $\frac{\partial^{k_1+k_2} H_2}{\partial z_1^{k_1} \partial z_2^{k_2}}$ vanishes at $(1, 1)$, $(-1, -1)$, $(-1, 1)$;
- $\frac{\partial^{k_1+k_2} H_3}{\partial z_1^{k_1} \partial z_2^{k_2}}$ vanishes at $(1, -1)$, $(-1, -1)$, $(1, 1)$.

Straightforwardly, the coefficients of filters $H_0, \dots H_3$ are polynomial w.r.t. the parameters $\cos \alpha_i, \sin \alpha_i, \cos \beta_i, \sin \beta_i$, and the flatness equations are linear w.r.t. the coefficients of the filters, making the whole system of equations, described by K and N, a set of polynomial equations.

16.7.1 Regularity estimates for two-dimensional wavelets

We look for wavelets that are continuously differentiable, as many times as possible. However, among the functions that are continuous but not continuously differentiable for instance, some are more regular than others. There are two common definitions of regularity, corresponding to two families of functions' spaces : the Sobolev (resp. Hölder) exponent of a function refers to

the index of the smallest Sobolev (Hölder) space it belongs to. Estimating the Sobolev and Hölder exponents of wavelets is not an easy task, but efficient algorithms exist for 1-D wavelets (Daubechies, 1992; Rioul, 1992).

There are, however, few regularity estimates for nonseparable wavelets in literature : Villemoes proposes in (Villemoes, 1992) an estimate of Sobolev exponent in the quincunx case, and in (Villemoes, 1997) an estimate of Hölder exponent in the quincunx case, if this exponent is less than 1. Cohen *et al* and Gröchenig propose in (Cohen *et al.* , 1995) an estimate of Sobolev exponent for general sampling matrices, which we will present more precisely now for the sampling matrix $2I$. A lower bound of the Hölder exponent can also be calculated (Moreau de Saint-Martin, 1997), on the basis of (Cohen & Daubechies, 1995).

The estimate needs a preliminary condition : The scaling funtion ϕ and its integer translates must be a Riesz basis of the subspace they span. Assume that the so-called "Cohen's condition" (Cohen & Daubechies, 1993) is met, which is always the case in practice, since we consider low-pass filters that do not vanish in their pass-band. We then define the transfer operator T associated to $|H_0|^2$, which acts on $2\pi\mathbb{Z}^2$-periodic functions as :

$$Tf(\omega) = \sum_{k=0}^{3} \left| H_0\left(\frac{\omega + 2\pi e_k}{2}\right)\right|^2 f\left(\frac{\omega + 2\pi e_k}{2}\right) \tag{16.31}$$

where

$$\{e_0, e_1, e_2, e_3\} = \{(0,0), (0,1), (1,0), (1,1)\}$$

The finite dimensional space F of trigonometric polynomials whose Fourier coefficients are supported in $[0, 2K-1]^2$ is stable by T. For the sequel T denotes the restriction to F. If the spectral radius of T is 1, if the eigenvalue 1 is the only unitary eigenvalue and has multiplicity 1, then the integer translates of the scaling function are a Riesz basis of the subspace they span. This property is assumed in the sequel.

The Sobolev exponent of the scaling function ϕ is defined as follows, where Φ denotes the Fourier transform of ϕ :

$$s_2 = \sup\left\{ s > 0; \int \left(1 + \|\omega\|^2\right)^s |\Phi(\omega)|^2 \, d\omega < \infty \right\} \tag{16.32}$$

We assume that H_0 has flatness order N, which means that N is the largest integer such that :

$$\forall k \in \{1, \ldots, 3\}, \forall q_1, q_2, \quad q_1 + q_2 \leq N, \frac{\partial^{q_1+q_2} H_0}{\partial \omega_1^{q_1} \partial \omega_2^{q_2}}(\pi e_k) = 0 \tag{16.33}$$

The Sobolev exponent s_2 of ϕ is given as :

$$s_2 = -\frac{\log \rho}{2 \log 2} \tag{16.34}$$

where ρ denotes the spectral radius of T restricted on the subspace E of trigonometric polynomials vanishing at $(0,0)$ as well as all their partial derivatives up to order N.

It can be shown (Moreau de Saint-Martin, 1997) that the spectrum of T is made of the spectrum of T restricted to E and of the eigenvalues 2^{-k} for $0 \leq k \leq N$, with multiplicity $k+1$. For example, the Matlab implementation of the Sobolev estimate consists in writing the matrix of T as :

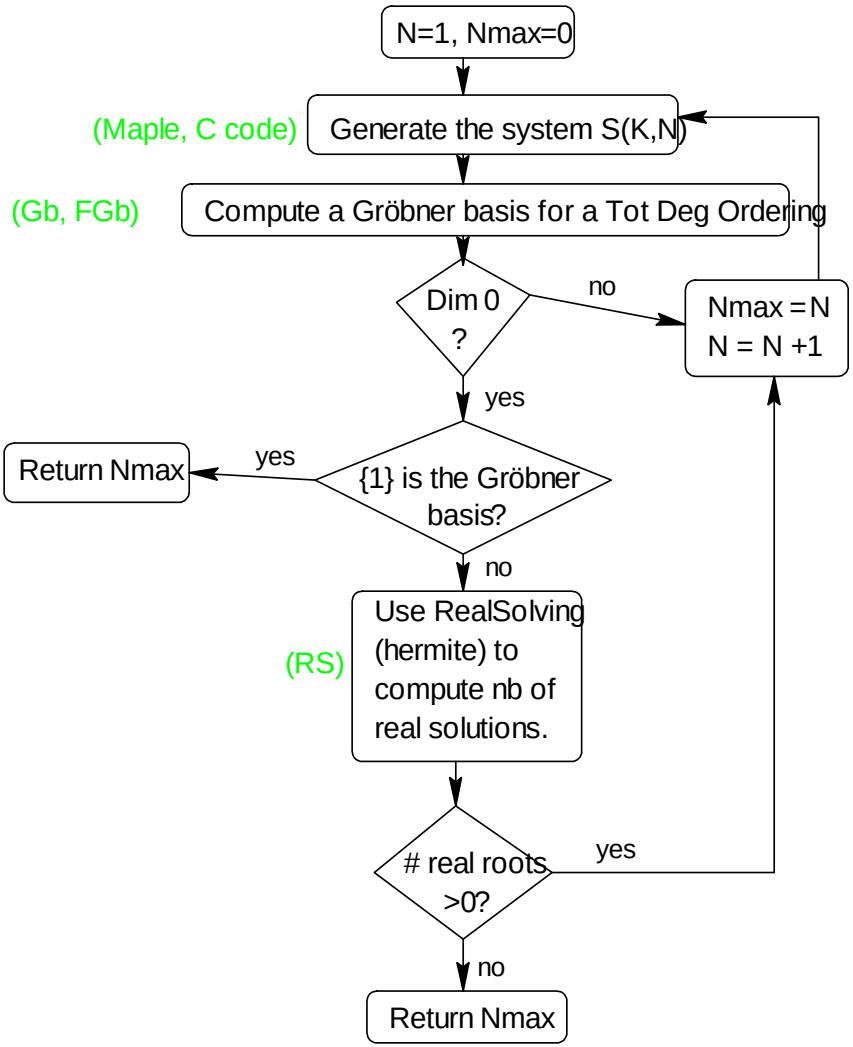
$$A(i,j) = a(2(i/(4K-1)) - (j/(4K-1)), 2(i\%(4K-1)) - (j\%(4K-1))) \tag{16.35}$$

where $a(k,l)$ denote the Fourier coefficients of $|H_0(\omega)|^2$, / the integer division and % the remainder of the integer division. The spectrum is calculated and ρ is extracted from the set of eigenvalues.

16.8 Conception de filtres optimaux dans la famille de Kovacevic Vetterli

Solving the problem of obtaining maximally flat filters divides into many parts:

1. Generating the system of equations.
2. Substituting the variables by using computation of Gröbner basis techniques.
3. Determining, for given K the highest possible, N.
4. Describing the family of the filters satisfying equations (K,N).
5. Choosing the remaining free degrees so as to obtain good properties.
6. Analyzing the resulting filters in terms of time–frequency localization, of regularity and of performances in an image compression system.



16.8.1 Approche naive

The only example provided in (Kovacevic & Vetterli, 1995) is a solution to the (3,2) system. In fact, this (3,2) problem can be solved by using computer algebra software as a "black box" : let us write the equations as a polynomial system in cosines and sines of the angles. There are 6 angles, so that there are 12 variables in the polynomial system. For each $H_i, \partial H_i/\partial z_1, \partial H_i/\partial z_2$, we consider vanishing moments equations at 3 points, which makes 36 equations

altogether, and we have to add the 6 equations imposing $\sin^2 + \cos^2 = 1$. Gröbner basis computation can then be carried out, and a zero-dimensional system is obtained. Triangular systems (in this case the Rational Univariate Representation) provide the roots explicitely, since the first polynomial factorizes in terms of degree four. We thus obtain 64 explicit solutions in the form $[\cos\alpha_1, \ldots, \cos\beta_3, \sin\alpha_1, \ldots, \sin\beta_3]$ e.g. :

$$\left[\frac{\sqrt{2}}{8} + \frac{\sqrt{15}\sqrt{2}}{8}, \frac{7}{8}, \frac{1}{4}, -\frac{\sqrt{2}}{8} + \frac{\sqrt{15}\sqrt{2}}{8}, \frac{\sqrt{15}}{8}, \frac{\sqrt{15}}{4}, -\frac{\sqrt{2}}{2}, -1, -1, \frac{\sqrt{2}}{2}, 0, 0\right].$$

These solutions correspond basically to 2 different filter banks. However, the solution which is different from the one presented in (Kovacevic & Vetterli, 1995) is not better than the previous one in terms of space-frequency localization or of regularity.

Actually, this is the only example where the straightforward approach can be applied, because in other cases the system is not zero-dimensional.

16.8.2 Génération des équations

It takes few lines of code to implement directly in Maple (or a similar system) the matrix formulation described in section 16.7. But for $K \geq 6$ and $N \geq 4$ it takes several hours to generate the equations and for $K \geq 8$ it is impossible to obtain the results.

The first idea consists in working with 2×2 complex matrices instead of 4×4 real matrices. It is easy to identify a linear transform in $\mathbb{R}^4$ with a linear transform in $\mathbb{C}^2$ if it holds a positive determinant. Not all of the matrices in the product have a positive determinant, making the identification a little more tricky. More precisely the matrix P will be considered simultaneously with a neighbour matrix so as to make the identification possible, as follows :

$$PD(z_1, z_2)P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & z_1 & 0 & 0 \\ 0 & 0 & z_1 z_2 & 0 \\ 0 & 0 & 0 & z_2 \end{bmatrix} \tag{16.36}$$

and similarly

$$P \begin{bmatrix} 1 \\ z_1 \\ z_2 \\ z_1 z_2 \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ z_1 z_2 \\ z_2 \end{bmatrix} \tag{16.37}$$

We will then consider, instead of

$$H(z_1^2, z_2^2) \begin{bmatrix} 1 \\ z_1 \\ z_2 \\ z_1 z_2 \end{bmatrix} \tag{16.38}$$

the following cascade of processings. Using Eq. 16.37, we start with the vector

$$u_0 = \begin{bmatrix} 1 + iz_1 \\ (z_1 + i)z_2 \end{bmatrix} \tag{16.39}$$

we consider the matrices one after the other. When considering W, the current vector is multiplied by

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{16.40}$$

When considering R_l, the current vector is multiplied by

$$\begin{bmatrix} e^{i\alpha_l} & 0 \\ 0 & e^{i\beta_l} \end{bmatrix} \tag{16.41}$$

When considering $PD(z_1, z_2)P$, the following processing is applied to the current vector $u = (u(1), u(2))'$:

$$u := \begin{bmatrix} \mathcal{R}(u(1)) + iz_1\mathcal{I}(u(1)) \\ z_1z_2\mathcal{R}(u(2)) + iz_2\mathcal{I}(u(2)) \end{bmatrix} \tag{16.42}$$

This approach lets us divide the number of variables by 2 and decrease dramatically the number of terms in each equations (because we replace products of sums by products of monoms). The resulting equations are polynomials in $e^{i\alpha_l}$ and $e^{i\beta_l}$.

We also notice that we consider the derivatives at given points and so it is useless to compute huge polynomials in z_1, z_2, calculate the derivatives and then evaluate them at these points afterwards. A much more efficient way consists in dealing with finite order expansions at these points, directly providing the equations for all derivatives. In addition, as we consider derivatives at points (-1,-1),(1,-1), (1,1) and (-1,1), we observe that each equation writes as a sum of terms such as $e^{\imath(\pm\gamma_1\pm\gamma_2\cdots\pm\gamma_K)}$ where γ_i denotes either α_i or β_i; thus each term can be coded efficiently as a 32 bit word for $K \leq 16$ and the multiplication of the two terms becomes a simple bitwise operation. The previous cascade of processings was implemented in C, and the resulting program generates quickly the equations.

16.8.3 Changement de variables

However, there are still a large number of equations: the size of the generated system is more than 60 mega bytes for $K = 8$! We now propose a substitution of variables. Combined with an efficient use of Gröbner bases tools (in particular we use the reduction function as a simplification process), it provides a much nicer formulation of the issue. Basically, we solve the issue by induction for N (detailed below), and we propose the following substitution of variables :

$$\alpha_1 = a_1 - \frac{a_2 - b_2}{2} \tag{16.43}$$

$$\beta_1 = \frac{a_2 + b_2}{2} \tag{16.44}$$

$$\alpha_k = \frac{a_k - b_k}{2} - \frac{a_{k+1} - b_{k+1}}{2} \text{ for } k = 2 \dots K-1 \tag{16.45}$$

$$\beta_k = \frac{a_{k+1} + b_{k+1}}{2} - \frac{a_k + b_k}{2} \text{ for } k = 2 \dots K-1 \tag{16.46}$$

$$\alpha_K = \frac{a_K - b_K}{2} \tag{16.47}$$

$$\beta_K = b_1 - \frac{a_K + b_K}{2} \tag{16.48}$$

and we obtain the following results :

Theorem 16.8.1. *(i)Equations can be reduced to 2 sub-systems, one depending only on the a_k, and the other one only on b_k. Both sub-systems are identical. In each sub-system there are only K variables (compared to $4K$ in the first formulation).*
(ii) a_1 and b_1 are equal to $-\frac{\pi}{4}$ mod π.

Hence we make a second subtitution of variables :

$$A_1 = B_1 = \frac{1}{2}, \quad A_k = e^{i(a_k - a_1)}, \quad B_k = e^{i(b_k - b_1)} \text{ for } \quad k = 2 \dots K \tag{16.49}$$

Now, let us briefly describe the induction. For $N = 1$, we observe directly that the system $S(K, 1)$ reduces to the constraint on a_1 and b_1, as already mentioned in (Kovacevic & Vetterli, 1995). Assume that the reverse lexicographic degree Gröbner basis $\mathrm{GB}_A(K, N-1)$ of the system $S_A(K, N-1)$ is known. It provides directly the reverse lexicographic degree Gröbner basis $\mathrm{GB}(K, N-1)$ of the system $S(K, N-1)$. Using finite order expansions, $S(K, N)$ is obtained as equations in A_k, $1/A_k$, B_k, $1/B_k$ and i with equation $i^2 + 1 = 0$. This system is reduced by the family $\mathrm{GB}(K, N-1)$, and further reduced by using the fact that $1/B_k$ is the inverse of B_k. At this point, we observe that the resulting system can be split into two subsystems, $S_A(k, N)$ depending on A_k and $1/A_k$ and $S_B(K, N)$ depending on B_k and $1/B_k$.

By this inductive calculation, we obtain the following as an equivalent system to the system (K, N) :

Theorem 16.8.2. *For $K \leq 10$,$N \leq 6$, the system (K, N) that $A_2, \dots, A_K$ must satisfy is $\{E_2, \dots, E_N\}$ where:*

$$E_2 \quad \sum_{k=1}^{K} A_k = 0$$

$$E_3 \quad \sum_{k=2}^{K} \frac{1}{A_k} \left(\sum_{j=1}^{k-1} A_j - \sum_{j=k+1}^{K} A_j \right) = -\frac{1}{4}$$

$$E_4 \quad \sum_{k=2}^{K-1} \frac{1}{A_k} \left(\sum_{j=k+1}^{K} A_j \right) \left(\sum_{j=1}^{k-1} A_j \right) = -\frac{1}{8}$$

$$E_5 \quad \sum_{i=4}^{K} \sum_{j=2}^{i-1} \frac{1}{A_i A_j} \left(\sum_{k=1}^{j-1} A_k - \sum_{i+1}^{K} A_k \right) \left(\sum_{k=j+1}^{i-1} A_k \right) = \frac{4K-7}{32}$$

$$E_6 \quad \sum_{k=4}^{K} \frac{1}{A_k} \sum_{i=2}^{k-2} \frac{1}{A_i} \left(\sum_{j=i+1}^{k-1} A_j \right) \left(2 \left(\sum_{j=1}^{i-1} A_j \right) + 4 \left(\sum_{j=k+1}^{K} A_j \right) \left(\sum_{j=2}^{i-1} A_j \right) \right) = 3$$

These equations have been checked by computer algebra for $K \leq 10$, and we conjecture that they hold for any K. For $N < 4$ the computation can be done by hand.

This means that all solutions to the problem (K, N) correspond to solutions to the above equations where all coordinates *lie on the unit circle*.

Unfortunately all of the methods described in section **??** give all of the complex solutions of an algebraic system ($|z|^2 = 1$ is not an algebraic equation). So we use the following trick: as the polynomials have rational coefficients, the conjugate of a solution is a solution, hence for a solution with coordinates on the unit circle, the set of the inverses of the coordinates is also a solution. We introduce the new variables A'_k and the equations $A_k A'_k - 1 = 0$; then A'_k must satisfy the same equations as A_k. The resulting system contains all of the admissible solutions we look for (where all coordinates yield modulus 1), and other parasite solutions, but this trick helps in reducing dramatically the number of "bad" solutions.

For a further description of the solutions of this system, we need a list of triangular systems (we got previous results by using only reverse lexicographic degree Gröbner bases) whose computation is much easier. For systems such as (8,5), the computation of the lexicographic Gröbner basis is a very difficult task, making it necessary to use recent and efficient computer algebra algorithms.
Let $\mathcal{S}_{K,N}$ be the system with variables A_k and A'_k (where $A_k A'_k = 1$).
The computation is then carried out in five steps :

1. Computation of the Gröbner basis of $\mathcal{S}_{K,N}$ with DRL order, the variables being ordered as
   $[A_1, \ldots, A_K, A'_1, \ldots, A'_K]$.
2. Computation of the Gröbner basis of $\mathcal{S}_{K,N}$ with a (DRL,DRL) order and two blocks $[A'_1, \ldots, A'_K]$ and $[A_1, \ldots, A_K]$.
3. In the resulting system just keep the equations involving the A_k only. This a DRL Gröbner basis (elimination theorem).
4. Computation of the lexicographic Gröbner basis of this subsystem "by change of ordering".
5. From the latter basis compute a list of triangular systems

The only restriction is that this algebraic approach describes all of the complex solutions of these systems, while we are only interested in the unitary ones. This point has to be considered for each particular system.

16.8.4 Maximal flatness for given K and maximally flat filter banks

Proposition 16.8.1. *For a given $K < 9$, we can compute the maximal N such that the system $\mathcal{S}_{K,N}$ has admissible solutions. The exact values $Nmax$ are reported in Table 16.8.4.*

The method is as follows : first we show how to find a candidate for $Nmax$ and then we prove (with computations) that it was correct.

Let us consider the following procedure:

ComputeNmax(integer K)
  $N = 1$
  repeat
    Compute $G_{K,N}$ a DRL basis of $\mathcal{S}_{K,N}$
    if $G_{K,N} = \{1\}$ **or** $G_{K,N}$ is zero-dimensional **then return** $N - 1$
    end if
    $N = N + 1$
  end repeat

If $Nmax = \text{ComputeNmax}(K)$, $3 < K < 9$, we claim that $\mathcal{S}_{K,Nmax+1}$ has no admissible solutions.

If $G_{K,Nmax+1} = \{1\}$ then the statement is obvious ; otherwise we introduce the new variable c_K and the equation $2c_K - A_K - \frac{1}{A_K} = 0$, so that $c_K = \cos(A_K)$ is a real number with modulus less than 1 for any unitary A_K ; by calculating a Gröbner basis, a degree-2 equation in c_K is obtained, so that we just have to check by hand that this equation has no real solution with modulus less than 1.

Lemma 16.8.1. •

(i) *$\mathcal{S}_{3,2}$ is zero-dimensional and $\mathcal{S}_{K,Nmax}$ is positive-dimensional for $3 < K < 9$.*

(ii) *For $K < 9$, $\mathcal{S}_{K,Nmax}$ has admissible solutions.*

When we want to prove that solutions with modulus 1 to the system exist (the system is generally not zero-dimensional) we have to add extra equations to make the system zero-dimensional. We rewrite the system so that we look for real roots, and calculate how many there are. For each system we have found a zero-dimensional system holding modulus 1 roots by adding equations setting some coordinates to 1.

There are many interesting facts reported in following Table

K	3	4	5	6	7	8
maximal N	2	2	3	3	4	5
dimension	0	2	2	4	2	2

Maximal flatness N for a filter bank with size $2K$ from (Kovacevic & Vetterli, 1995) and dimension of the corresponding system.

The first fact is that it is possible to achieve high flatness if the filters' support is large enough. We conjecture that an arbitrary flatness is possible, and we prove it until 5. The second fact is that on the whole the systems are not zero-dimensional. There are an infinite number of maximally flat wavelets, so that we will look for wavelets optimizing a property amongst this set of maximally flat wavelets. This is to be compared to the 1-D dyadic case, where the maximally flat orthogonal wavelets of given size are finitely many (Daubechies, 1988).

16.8.5 Optimizing the remaining parameters

In terms of computer algebra, this means that the tools developped for zero-dimensional systems cannot be used directly, and this reminds the applicative need for extending the zero-dimensional tools to arbitrary systems. Therefore, the recent algorithms for the computation of the lexicographic Gröbner basis, and the triangular systems are of the highest interest to us. They provide explicitly all of the parameters as a function of 2 or 4 free degrees. The following theorem reduces the number of systems to be studied :

Theorem 16.8.3. *For $2 < K < 9$, all of the admissible solutions of $\mathcal{S}_{K,Nmax}$ are solutions of **one** triangular system T_K (see **??**). The number of free parameters in T_K is equal to the number of free parameters in the whole system $\mathcal{S}_{K,Nmax}$.*

This is a by product of the triangular systems theory. The optimization among the maximally flat wavelets consists in choosing these free degrees according to a criterion. Many possible criteria can be considered, and we have mainly considered 3 of them :

Symmetry : The low-pass filter is structurally centro-symmetric, but we would like to be closer to an axial symmetry for the low-pass filter. This can be turned into a quadratic criterion.

Frequency selectivity : Frequency selectivity : we minimize the energy of the low-pass filter outside $[-a, a] \times [-a, a]$, which means maximizing :

$$\int_{-a}^{a}\int_{-a}^{a} |H_0(\omega_1,\omega_2)|^2 d\omega_1 d\omega_2 = \sum_{i,j,k,l} H_0(i,j)H_0(k,l)\left(\int_{-a}^{a}\cos((i-k)\omega_1)d\omega_1\right)\left(\int_{-a}^{a}\cos \right. \tag{16.50}$$

Energy compaction : We consider the energy compaction for a class of signals :

$$\sum_{i,j}\sum_{k,l} H_0(i,j)H_0(k,l)\mathbb{E}[x(i,j)x(k,l)] \tag{16.51}$$

with the correlations given by $\mathbb{E}[x(i,j)x(k,l)] = \rho^{|i-k|+|j-l|}$ and $\rho = 0.9$.

These criteria are optimized with respect to remaining free degrees with a conjugate gradient algorithm.

16.9 Examples de filtres

For each example we give:

- The shape of the solutions i.e. of T_K
- Optimized filters for various criteria and their Sobolev exponent.

16.9.1 Examples for $N = 2$

After the the substitution of variables, the system (3,2) is very easy to solve, because we just have to find two unitary complex points A_2 and A_3 such that $A_2 + A_3 = -\frac{1}{2}$. There are a finite number of solutions, and they are easy to calculate : we, of course, obtain the filter banks we got by the straightforward application of computer algebra tools looking for real roots of polynomial systems.

If we increase K, we are still able to describe the sets of solutions. For $K = 4$, $N = 2$ is still the maximal possible flatness. There is, for both subsystems, a degree of freedom, which can be seen geometrically. Its optimization for various criteria is easy to carry out. For all criteria mentionned above, the resulting filters hold Sobolev exponents larger than 1, so that the corresponding scaling function and wavelets are at least continuous.

For $K = 5$, we know that the maximally flat wavelets hold $N = 3$. However, we might choose to relax some of the flatness constraints so as to improve some other characteristics. E.g., we study the system (5,2). It can be shown to be of dimension 4, and it has a very nice geometric interpretation, depicted in Fig. 16.9.1:

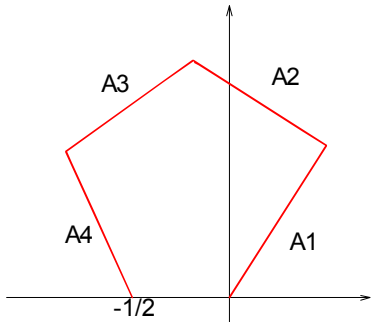


Fig 16.9.1: Geometric interpretation for solving the system (5,2).

It is then clear that for each subsystem two parameters can be chosen, if the angle between A_1 and A_4 is not to close to π. However the optimization of the 4 free degrees is a difficult task, and the solutions to $(5, 3)$ have nice properties in terms of symmetry, frequency localization and energy compaction, so that we could not improve numerically these characteristics by relaxing some of the flatness constraints.

16.9.2 Examples with $N = 3$

The minimal filter size in order to achieve 3 vanishing moments is 10×10 ($K = 5$). The solutions of the system (5,3) are described by the lexicographic Gröbner basis. In each subsystem, a variable can be freely chosen in $[-s_1, s_1]$ modulo 2π where $s_1 \sim 2.652$. The next variable is obtained by solving a second degree equation, and the other ones by linear equations.

For the system (6,3), it is possible to calculate the lexicographic Gröbner basis and even to write the system as a union of triangular systems, one of them yielding 4 free degrees. However, the filters we obtained by the optimization procedure of the free degrees are similar to the one obtained in the (5,3) case.

16.9.3 Examples with $N = 4$

The smallest size for achieving 4 vanishing moments is 14×14. Once again, the lexicographic basis provides all variables as a function of 2 free degrees. It consists of an equation in A_6 of degree 4 with coefficients that are polynomials in A_7. The kth equation is implying $A_{7-k}, \dots A_7$ and is linear in A_{7-k}.

However, the computation of the variables from the free degrees includes solving a degree-4 equation, which means, in general, choosing one of the 4 solutions. This is done numerically, and the problem is that numerical

algorithms cannot follow a root, which implies that the variables are not continuous functions of the degrees of freedom. This makes the optimization of the remaining degrees of freedom more difficult, but in any case interesting filter banks can be designed. Different filter banks are obtained, depending on the chosen root of the degree-4 equation. For instance, in figure 16.9.3, we present filters optimizing the symmetry criterion for different roots.

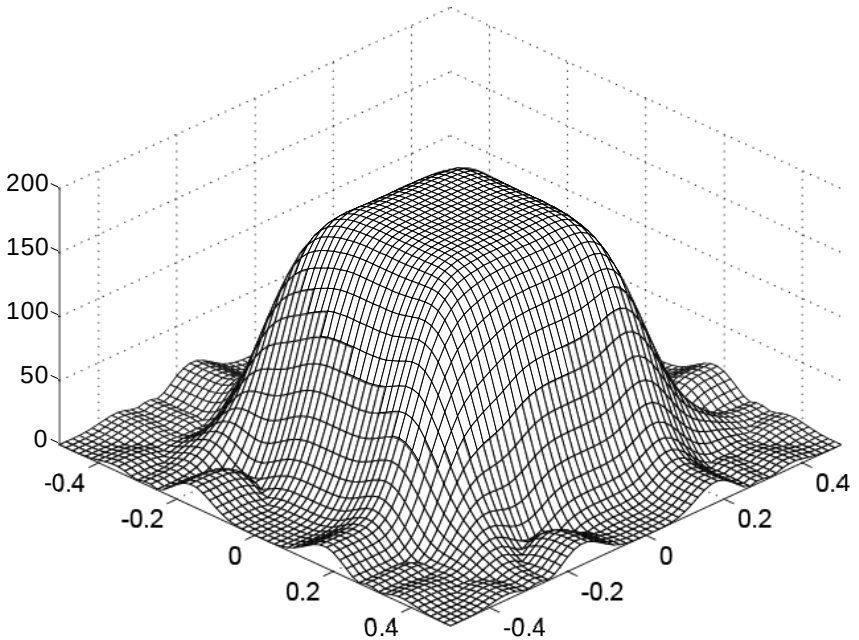


Fig 16.9.3: Frequency response of a 14×14 filter with flatness order 4, optimized w.r.t. symmetry criterion (third root in maple sense).

They hold similar qualities in terms of symmetry, but are qualitatively different. They hold similar Sobolev exponents, between 1.78 and 1.80.

16.9.4 Examples with $N = 5$

Calculating the Gröbner basis for the system (8,5) cannot be done without recent algorithms. Despite the numerical solving of a 4th degree equation, very interesting filters have been obtained, such as those depicted in Fig. 16.9.4:

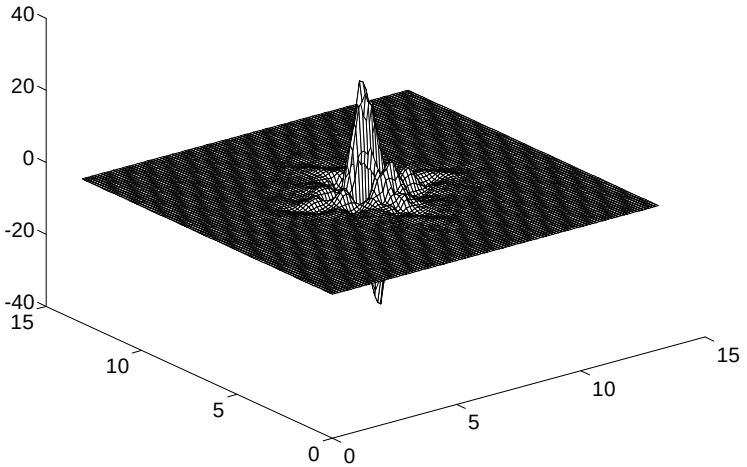


Fig 16.9.4: scaling function and wavelets H_2 associated to the 16×16 filter bank depicted in Fig. 7. It is a solution to the (8,5) problem, whose Sobolev exponent is 2.11.

The corresponding scaling function and wavelets are orthogonal, linear-phase and continously differentiable.

16.9.5 Application à la compression d'images

We now present some experimental results on the application of these filter banks holding simultaneously orthogonality and phase-linearity to image compression. We introduced these new filters in the compression scheme presented in (Onno, 1996). It consists of scalar quantization of the subband signals followed by a Universal Variable Length Coding (Delogne & Macq, 1991). The wavelet packet tree is chosen according to a rate-distortion criterion (Ramchandran & Vetterli, 1993), as well as the quantization steps (Shoham & Gersho, 1988). We consider `Lena256` at 0.5 bpp.

With Daubechies' filters with length from 6 to 16, we obtain a Peak Signal to Noise Ratio between 30.40 and 30.75 dB. With our nonseparable filters, we obtain 28.4 dB with the 6×6 filter bank, around 29.6 dB for the 8×8 filter banks, and between 30.42 and 30.57 dB for the 10×10 to 16×16 filter banks. Different filter banks hold very close performances, so that changing the image or the coding scheme may change the hierarchy. In other words all these filters may be considered as equivalent.

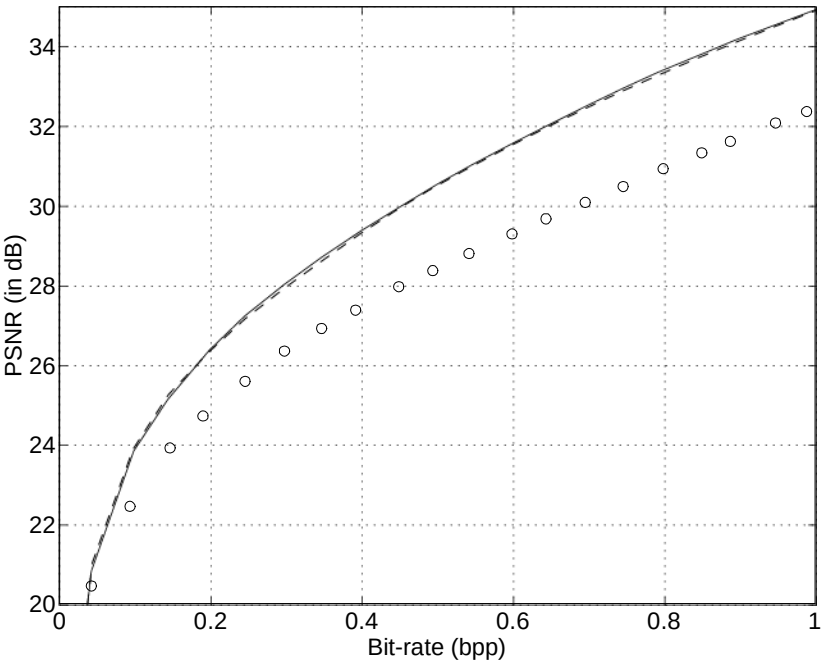


Fig 16.9.5: Rate-distortion curves for separable Daubechies filters of length 14 (dashed line), nonseparable filter with support 6×6 from(Kovacevic & Vetterli, 1995) (circles) and nonseparable filter bank from figure 16.9.3 (solid line).

This is illustrated by figure 16.9.5, depicting the rate-distortion curves for separable Daubechies filters of length 14, nonseparable filter with support 6×6 from (Kovacevic & Vetterli, 1995) and nonseparable filter bank from figure 16.9.3.

Among the non-separable filter banks, the best results are obtained with the 10×10 filter bank which has been optimized w.r.t. energy compaction. The 14×14 filter bank in Fig. 16.9.3 follows, and then the other 10×10 and 14×14 filter banks, and then the 16×16 filter bank. This means that 16×16 does not ensure enough resolution in space domain.

These results do not take into account the subjective quality of the images. Subjectively, the images encoded by the non-separable filter bank schemes cannot be said to be better or worse than the ones from good separable schemes.

16.10 Conclusion

The results of our study concern four scientific communities :

- From a computer algebra applications point of view, it is one of the major applications of computer algebra in digital signal processing, and this area looks to be a promising application field for computer algebra, especially the filter bank area. In addition, the problem has been solved using the most recent techniques for the Gröbner bases computation.
- From the point of view of the wavelet field, it is the first example of bidimensional orthogonal linear-phase wavelets which are continuously differentiable.
- From the point of view of the filter bank design, the design examples are convincing : the resulting filters have good frequency characteristics, and it has been shown that the remaining free degrees can be chosen so as to optimize various criteria.
- From the image compression point of view, it is interesting to have new filter banks holding simultaneously orthogonality, centrosymmetry and regularity, and in addition with good frequency characteristics. Relating our results to these of (Venkataranam & Levy, 1994a) suggests that improved compression efficiency can be achieved by future work on design techniques for such non separable filters.

Further work might also include the study of other cascade forms for filter banks, or a proof that arbitrarily high flatness and regularity can be achieved in this filter bank family.

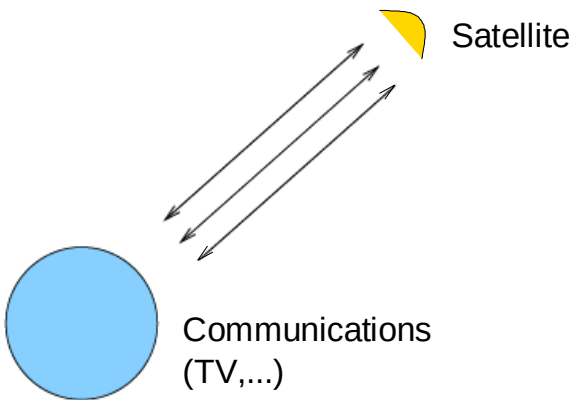
17. Synthèse de filtres hyperfréquences

En collaboration avec F. Rouillier et F. Seyfert (tiré en partie des articles (Faugère J.C. and Moreau de Saint-Martin F. and Rouillier F., 1996; Faugère J.C. and Moreau de Saint-Martin F. and Rouillier F., 1998; Faugère *et al.* , 2005; Faugère *et al.* , à paraître)).

17.1 Contexte général de l'application

17.1.1 Introduction aux filtres à cavités

Dans le domaine des télécommunications spatiales (transmission par satellite),



les contraintes spécifiques des matériels embarqués (robustesse, ...) conduisent à réaliser des filtres à cavités résonnantes bimodes dans le domaine hyperfréquences. Ces filtres sont constitués d'une succession de corps creux cylindriques reliés (couplés magnétiquement) entre eux par des iris. L'onde électromagnétique qui traverse ces cavités satisfait aux équations de Maxwell. Dans la plage de fréquences d'utilisation du filtre, le champ électrique dans chacune des cavités peut être vu comme se décomposant selon deux modes orthogonaux dans le plan perpendiculaire à l'axe des cavités.

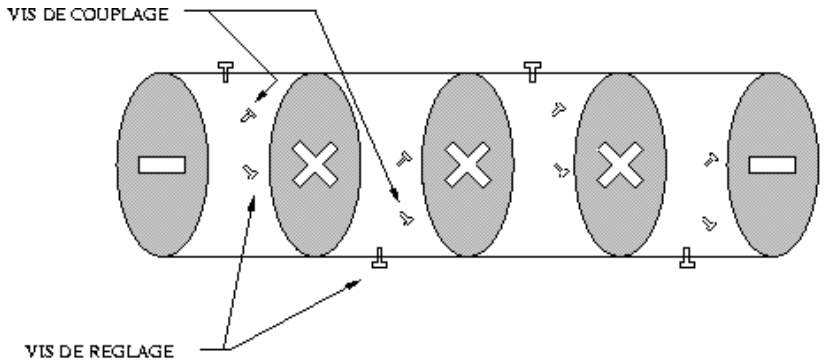


Fig 17.1.1: Cavité munie de vis.

Chacune des cavités est munie de trois vis (fig. 17.1.1) : les vis horizontales et verticales permettent, en perturbant la géométrie de la cavité, d'ajuster les fréquences de chacun des deux modes. La dernière vis permet, elle, d'ajuster le couplage des deux modes entre eux. Enfin, les iris réalisent un couplage entre modes de cavités adjacentes. Lors de la synthèse du filtre, la géométrie des iris est déterminée et respectée autant que possible à l'usinage.

In reference (Cameron *et al.* , 2002), a synthesis method for the 'Box Section' configuration for microwave filters was introduced. Box sections are able to realize a single transmission zero each, and have an important advantage that no 'diagonal' inter-resonator couplings are required to realize the asymmetric zero, as would the equivalent trisection. Also the frequency characteristics are reversible by retuning the resonators alone (Faugère *et al.* , 2005), retaining the same values and topology of the inter-resonator couplings.

The first feature leads to particularly simple coupling topologies, and is suitable for realization in the very compact waveguide or dielectric dual-mode resonator cavity, whilst the ability to reverse the characteristics by retuning makes the box-filter useful for diplexer applications, the same structure being usable for the complementary characteristics of the two channel filters.
Reference (Cameron *et al.* , 2002) continued on to introduce the extended box configuration for filter degrees N > 4, able to realize a maximum of (N–2)/2 (N even) or $(N-3)/2$ (N odd) symmetric or asymmetric transmission zeros. Fig. 1 gives extended box networks of even degree 4 (basic box section), 6, 8 and 10, showing the particularly simple ladder network form of the extended box configuration. In each case, the input and output are from opposite corners of the ladder network. The extended box network also retains the property of giving lateral inversion of the frequency characteristics by retuning of the resonators alone.

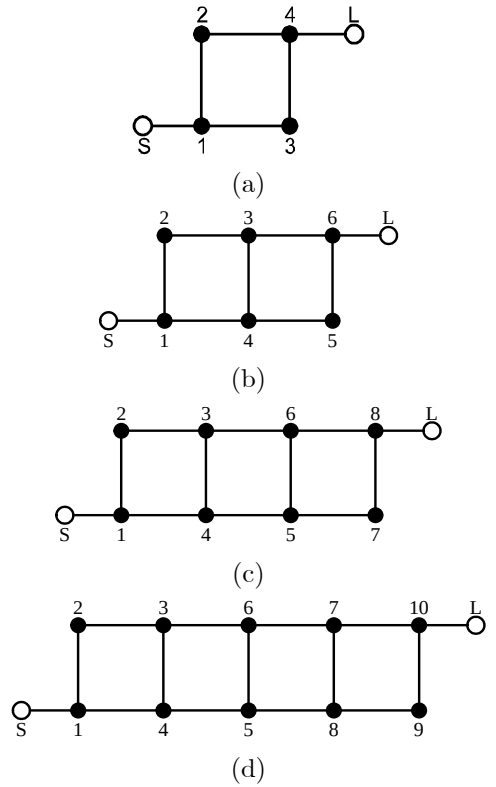


Fig. 1. Coupling and routing diagrams for extended box section networks:
(a) 4^{th} degree (basic box section) (b) 6^{th} degree (c) 8^{th} degree (d) 10^{th} degree.

The prototype coupling matrix for the extended box network may be easily synthesized in the folded or 'arrow' forms. However it appears that there is no simple closed form equation or procedure that may be used to transform the folded or arrow coupling matrix to the extended box form. In (Cameron *et al.* , 2002) a method was described which is essentially the reverse of the general sequence that reduces any coupling matrix to the folded form, for which a regular sequence of rotation pivots and angles does exist. Using this method means that some of the rotation angles cannot be determined by calculation from the pre-transform coupling matrix (as can be done from the 'forward' method) and so they have to be determined by optimization. Other methods (eg. (Macchiarella, 2003; Amari, 2000)) are also known to produce a solution.

Although most target coupling matrix configurations (eg propagating inline) have one or two unique solutions, the extended box configuration is

distinct in having multiple solutions, all returning exactly the same performance characteristics under analysis as the original prototype folded or arrow configuration. The solutions converged upon by existing optimization methods tend to be dependent upon the starting values given to the coupling values or rotation angles, and it can never be guaranteed that all possible solutions have been found. In (Faugère *et al.* , 2005) an approach based on computer algebra was outlined that allows to compute all the solutions for a given coupling matrix topology, including those with complex values (which of course are discarded from the solutions considered for the realization of the hardware). In this paper we detail the latter procedure as well as a modification in the choice of the set of algebraic equations to solve that leads to an important improvement of the algorithm's efficiency in practice.

Having a range of solutions enables a choice to be made of the coupling value set most suited to the technology it is intended to realize the filter with. Considerations influencing the choice include ease of the design of the coupling elements, minimization of parasitic couplings or resonator frequency offsets. Some of the coupling matrix solutions may contain coupling elements with values small enough to be ignored without damage to the overall electrical performance of the filter, so simplifying the manufacture and tuning processes.

In the following section we describe the multi-solution synthesis method, applicable to the extended box network and others that support multiple solutions. Finally we apply our procedure to the synthesis of filtering characteristics of degree 8 and 10. We demonstrate how the ability to choose among several coupling matrices simplifies the practical realization of the filter in dual-mode waveguide or dielectric resonator cavities. In particular an approximate synthesis technique based on a post-processing optimization step is presented and improves the approach of reference (Faugère *et al.* , 2005).

17.1.2 A General Framework for the Coupling Matrix Synthesis Problem

In this section we work with a fixed coupling topology, that is we are given a set of independent non-zero couplings associated to a low pass prototype of some filter with N resonators. Starting with numerical values for the couplings (coupling matrix M) and the i/o loads (R_1,R_2) one can easily compute the admittance matrix using following formula,

$$Y(s) = C(sI - jM)^{-1}C^t = \sum_{k=0}^{\infty} \frac{Cj^k M^k C^t}{s^{k+1}} \tag{17.1}$$

with:

$$C = \begin{bmatrix} \sqrt{R_1} & ...0... & 0 \\ 0 & ...0... & \sqrt{R_N} \end{bmatrix}$$

The coupling matrix synthesis problem is actually about inverting the latter procedure: given an admittance matrix we want to find values for the input/output loads and couplings that realize it. In order to formalize this we give a name to the mapping that builds the admittance matrix from the free electrical parameters and we define:

$$\mathrm{T{:}p} = (\sqrt{R_1}, \sqrt{R_N}..M_{i,j}) \rightarrow \ (CC^t, ...CM^kC^t, ...CM^{2N-1}C^t)$$

The above definition is justified by the fact that the admittance matrix is entirely determined by the first 2N coefficients of its power expansion at infinity (Kailath, 1980).
Now suppose that each of the electrical parameters move around in the complex plane: what about the corresponding set of admittance matrices? The latter can be identified with the image by T of C^r (C is here the field of complex numbers) where r is the number of free electrical parameters. We call this set V (= $\mathrm{T}(C^r)$) and refer to it as the set of admissible admittance matrices with respect to the coupling topology.
In this setting the coupling matrix synthesis problem is the following: given an element w in V compute the solution set of:

$$T(p) = w \tag{17.2}$$

Now from the definition of T it follows that equation (17.2) is a non-linear polynomial system with r unknowns, namely: the square roots of the i/o loads and the free couplings of the topology. From the polynomial structure of the latter system we can deduce following mathematical properties (we will take them here for granted):

- Equation (17.4) has a finite number of solutions for all generic w in V (generic means for almost all w in V) if and only if the differential of T is generically of rank r. In this case we will say that the coupling topology is non-redundant.
- The number of complex solutions of the equation (17.4) is generically constant with regard to w in V. Because of the sign symmetries this number is a multiple of 2^N and can therefore be written as $\mathrm{m}2^N$. The number m is the number of complex solutions up to sign symmetries and we will call it the "reduced order" of the coupling geometry.

Remarks:
The non-redundancy property ensures that a coupling geometry is not over-parameterized which would yield a continuum of solutions to our synthesis

problem. We illustrate this with the 6^{th} degree topology of Fig. 2.

- if no diagonal couplings are present (as suggested by the grey dots in Fig. 2), the topology is redundant, ie the synthesis problem admits an infinite number of solutions.
- If, for example, the coupling (1,4) is removed, the topology becomes non-redundant and is adapted to a 6-2 symmetric filtering characteristic. In this case the resulting coupling topology is the so called "arrow form" for which the coupling matrix synthesis problem is known to have only one solution. The reduced order of the latter topology is therefore 1.
- Finally, if diagonal couplings are allowed, the topology becomes non-redundant, and is actually the 6^{th} degree extended box topology of Fig. 1 and is adapted to a 6-2 asymmetric filtering characteristic. We will see in the following section that its reduced order is 8.

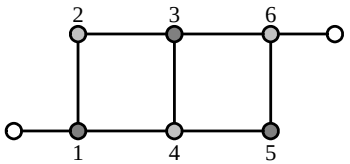


Fig. 2. Redundant topology

The use of the adjective "generic" in the latter statements is necessary for their mathematical correctness. In fact properties concerning parameterised algebraic systems are often true for all possible values of the parameters but an exceptional set. An example of this is given by following polynomial:

$p(x) = ax^2 + 1$.

The latter polynomial has two distinct roots for almost all complex values of the parameter a: the exceptional parameter set where the latter property does not hold is characterised by the equation a=0 and is very "thin" (or non-generic) as a subset of the complex plan.

The constructive nature of our framework for the synthesis problem depends strongly on our ability to invert numerically the mapping T, i.e compute the solution set of equation (17.4). In the next section we briefly explain how this can be done using Groebner basis computations

17.1.3 Nombre de solutions.

Table I summarizes the reduced order and the number of real solutions observed for a particular filtering characteristic for each of the extended box networks of Fig.1. The synthesis method is not limited to the case of extended box topologies: Table I also mentions the case of a 10^{th} degree topology (Fig. 3) adapted to 10-8 symmetric characteristics. The reduced order of the latter is equal to 3 and is therefore much smaller than the reduced order of 384 of its 10^{th} degree extended box analogue. This is something we observed empirically by testing our method on various networks: topologies adapted to

asymmetric characteristics seem to have a much higher reduced order than those adapted to symmetric ones.

Whereas the reduced order depends only on the coupling geometry, the number of real solutions depends on the prototype characteristic the network is realizing (position of transmission zeros (TZs), return loss, etc...) and is, by definition, bounded from above by the reduced order. One can even construct some coupling topologies and some filtering characteristics for which the synthesis problem admits only complex solutions. An academic example of this is given by the topology of Fig. 4 and the filtering characteristic, the canonical coupling matrix in arrow form of which is given on Fig. 5. In this latter case the reduced order of the coupling topology is 2 but both solutions to the synthesis problem are complex and equal to the matrix of Fig. 6 and to its conjugate.

TABLE I
Reduced Order & Observed Number of Real Solutions

Topology	Max. No. of TZs	Reduced Order	Observed No. of Real Solutions
Fig. 1(a)	1	2	2
Fig. 1(b)	2	8	6
Fig. 1(c)	3	48	16
Fig. 1(d)	4	384	36, 58
Fig. 3	8	3	1

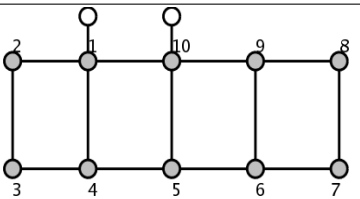


Fig. 3. Coupling topology adapted to 10-8 symmetric characteristics.

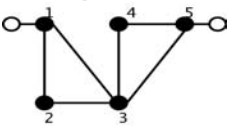


Fig. 4. Academic example of a 5^{th} degree coupling topology adapted to 5-2 asymmetric characteristics.

0	0.4	0	0	0	
0.4	0.3	0.1	0	0.1	
0	0.1	.2	0.2		0.2
0	0	0.2	0.2	1	
0	0.1	0.2	1		0.1

Fig. 5. Canonical coupling matrix in "arrow form" of a 5-2 filtering function, admitting only complex coupling matrices when using the topology of Fig. 4.

0	0.41-0.001j	0.006+0.074j	0	0	
0.41-0.001j	0.3-0.035j	0.079+0.031j	0	0	
0.006+0.074j	0.079+0.031j	.099-0.2j	0.3-0.075j		0.043-0.54
0	0	0.3-0.075j	0.3+0.23j	1.2+0.02j	
0	0	0.043-0.54j	1.2+0.02j		0.1

Fig. 6. Complex solution to the synthesis problem with coupling topology of Fig. 4 and coupling matrix in canonical arrow form of Fig.

17.2 Examples

17.2.1 Filtre de degré 8

As an application we will consider the synthesis of an 8^{th} degree filter in extended box configuration (see Fig. 1c). Using a computer algebra system (eg. Maple) we check that that this topology is non-redundant and from the application of the minimum path rule we conclude that the set of admissible admittances consists of rational reciprocal matrices of degree 8 with at most 3 transmission zeros. Using classical quasi-elliptic synthesis techniques an eighth degree filtering characteristic is designed with a 23dB return loss and three prescribed TZs producing one rejection lobe level of 40dB on the lower side and two at 40dB on the upper side (see Fig. 7a).

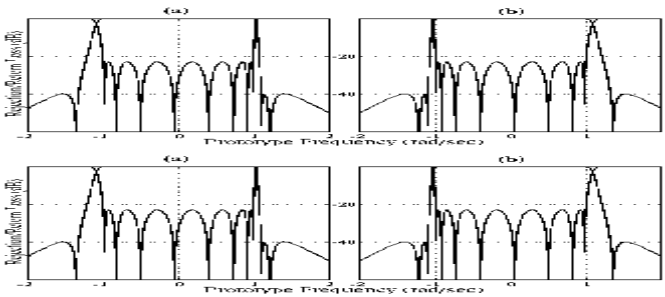


Fig. 7. (a) Original and (b) inverted rejection and return loss performance of an 8-3 asymmetric characteristic in extended box configuration.

Now computing the 2N first terms of the power expansion of the admittance matrix yields the left hand term of (17.4) which in turn could be solved using Groebner basis computations. At this point it is important to mention that the complexity of the Groebner basis computations of a system increases

with its total number of complex solutions. The natural sign symmetries of the system derived from equation (17.4) tend to artificially increase the latter (total number of solutions = $m2^N$) and may dramatically increase the computation time of the corresponding Groebner basis. Before continuing on with the synthesis we therefore explain how a rewriting of equation (17.4) allows us to get rid of these unwanted sign symmetries.
An alternative to equation (17.4) to invert the mapping T is to use an algebraic version of the approach presented in (Cameron, 1999) that is based on similarity transforms. If M is a coupling matrix in canonical form realizing the admittance matrix then equation (17.4) is "equivalent" to the following matrix equation where the unknown is a similarity transform P.

$$\begin{array}{ll} P = \begin{pmatrix} 1 \ldots 0 \ldots 0 \\ \vdots \quad H \quad \vdots \\ 0 \ldots 0 \ldots 1 \end{pmatrix} & (a) \\ H^t H = Id & (b) \\ \forall (i,j) \in I \quad (P^t M P)_{i.j} = 0 & (c) \end{array} \tag{17.3a}$$

In the latter, I is the set of indices corresponding to the couplings that must be zero in the target topology (in our example $I = \{(1,3),(1,5),(1,6)\ldots\ldots\}$). If P is a solution of (17.4) it is readily seen that all the similarity transforms that are obtained from P by inverting some of the columns vectors of the submatrix H are also solutions of (17.4). In order to break these symmetries the "trick" is to slightly modify (17.4b). We denote by h_i the i^{th} column vector of H. Some of the equations of (17.4b) indicate that the vectors h_i are unitary with regard to the Euclidean norm. We replace these normalizing equations by:

$$u_i^t h_i = 1 \tag{17.4}$$

where u_i is a randomly-chosen vector. We call (17.4') the resulting system. It can be verified that for a generic choice of the u_i's, all the solutions of (17.4) that are equivalent up to sign changes of their column vectors correspond to a single solution of (metricconverterProductID3'3'). More precisely to every set of solutions of (17.4) of the form:

$$H = (\pm h_1, \pm h_2 \ldots \pm h_i \ldots) \tag{17.5}$$

there corresponds a unique solution $G = (g_1 \ldots g_i \ldots)$ of (17.4') where the column vectors g_i are given by:

$$g_i = \frac{h_i}{u_i^t h_i} \tag{17.6}$$

With regard to the Groebner basis computation system, (17.4') has shown to be much more tractable than the algebraic system derived from equation (17.4).
Getting back to our 8^{th} degree example, we compute M the associated coupling matrix in arrow form and set up (17.4'). The latter is an algebraic system of linear and quadratic equations in the entries of H. The computation of its Groebner basis leads to the following result:

- the reduced order of the topology is 48
- for this particular filtering characteristic, 16 of the 48 solutions are real-valued.

Only the real solutions have a physical interpretation and are therefore of practical interest.

The criterion used to choose the best coupling matrix out of the 16 realizable ones will depend on the hardware implementation of the filter. Having in mind a realization with dual mode cavities, we choose to select solutions where the asymmetry between the two "arms" of each cross-iris is maximized in order to minimize parasitic couplings. The best ratios between couplings of the relevant pairs $(M_{14}, M_{23}), (M_{36}, M_{45})$ and (M_{57}, M_{68}) are found for the solution shown in Fig.8a, where each cross-iris has one of its coupling values at least 5 times larger than the other one.

0.0107	-0.2904	0	-0.8119	0	0	0	0
-0.2904	-0.9804	0.1081	0	0	0	0	0
0	0.1081	0.0605	0.5475	0	0.5984	0	0
0.8119	0	0.5475	0.1384	-0.0663	0	0	0
0	0	0	-0.0663	0.0152	0.5334	0.6782	0
0	0	0.5984	0	0.5334	0.0226	0	-0.1260
0	0	0	0	0.6782	0	0.0113	0.8530
0	0	0	0	0	-0.1260	0.8530	0.0107

(a)

0.0107	0.0001	0	-0.2464	0	0	0	0
0.0001	-0.9590	0.2094	0	0	0	0	0
0	0.2094	0.0498	0.4681	0	-0.4681	0	0
-0.2464	0	0.4681	0.0115	0.3744	0	0	0
0	0	0	0.3744	-0.0439	0.3744	0.8165	0
0	0	-0.4681	0	0.3744	0.0115	0	0.8623
0	0	0	0	0.8165	0	0.1975	0.0001
0	0	0	0	0	0.8623	0.0001	0.0107

(b)

Fig. 8. 'NxN' coupling matrices for an 8-3 asymmetric prototype: a) extended box configuration, b) 'cul-de-sac' configuration. $R_1 = R_N = 1.0878$

Fig. 8b illustrates that sometimes solutions emerge which have very small values for certain couplings (M_{12} and M_{78} in this case), which may be safely

omitted for the implementation without damaging the final response of the network. In this case a quasi cul-de-sac network is produced, similar to the 8-3 example given in (Cameron *et al.* , 2002). In fact one can show that with some renumbering, the cul-de-sac network of (Cameron *et al.* , 2002) is a sub-topology of the extended box where the couplings M_{12} and M_{78} are set to zero. The cul-de-sac topology is more restrictive than the extended box one in the sense that it is only adapted for the synthesis of auto-reciprocal characteristics, i.e. such that S_{11}=S_{22} holds. However our current filtering characteristic is, up to numerical errors, auto-reciprocal and this explains why in this example a quasi cul-de-sac network is found among all possible coupling matrices.
Finally it is shown that only the resonators need to be retuned in order to obtain an inverted characteristic. Fig. 7(b) shows the rejection and return loss obtained from the coupling matrices of Fig. 8 when the signs of their diagonal elements $M_{i,i}$ are changed (see (Faugère *et al.* , 2005) for details).

17.2.2 Filtre de degré 10

We consider the synthesis of a 10^{th} degree filter in the extended box topology of Fig. 1(d). Using our procedure we check that this topology is non-redundant and that it is adapted to asymmetric characteristics with up to 4 TZs. A filtering characteristics is designed with a 23dB return loss, 2 TZs at +j1.10929 and +j1. 19518 to give two 50dB rejection lobes on the upper side, and 2 more complex zeros at ±0.75877j0.13761 for group delay equalization purposes (see Fig. 10).

The corresponding coupling matrix in arrow form is determined and the computation of a Groebner basis of system (17.4) yields the following:

- The reduced order of the topology is 384
- For our specific filtering characteristic 36 real and therefore realizable solutions are found

When realized with dual mode cavities this topology requires 4 cross-irises. Our aim is to demonstrate how our exhaustive approach may allow the "replacement" of a cross-iris by an iris with a single arm as well as to simplify the future computer-aided tuning process of the filter.
Amongst all the possible coupling matrices the one with the smallest coupling corresponding to an iris is selected, which leads to the matrix of Fig. 11 where M_{45} is equal to -0.001. Setting M_{45} to zero yields a small but undesirable variation of the return loss as well as of the upper-band rejection lobes. The remaining couplings are therefore re-tuned thanks to an optimization step that minimizes the discrepancy between the original response and the one obtained by imposing that M_{45} be zero (see Fig. 12 for the resulting coupling matrix). A quasi perfect fit is obtained between the two responses: the least square error between the two return losses on the normalized broadband [-3,3] equals $8.83.10^{-5}$ (on the Bode plot there is visually no difference).
Finally the simplified coupling topology of Fig. 9 is considered as a new

topology in its own right. Using our procedure its reduced order is found to be equal to 2 and a second equivalent coupling matrix with the same coupling topology is computed (see Fig. 13). With regard to the "iris asymmetry criterion" of the last section the latter matrix is the best one.

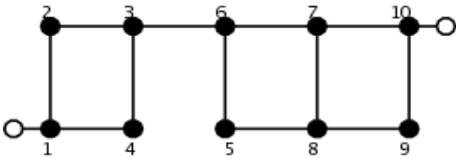
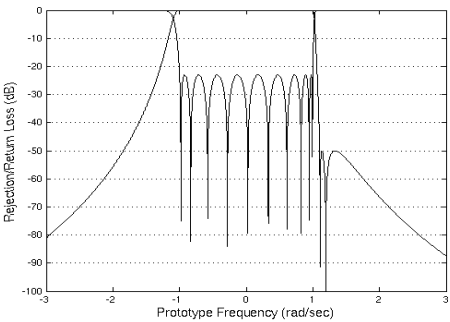


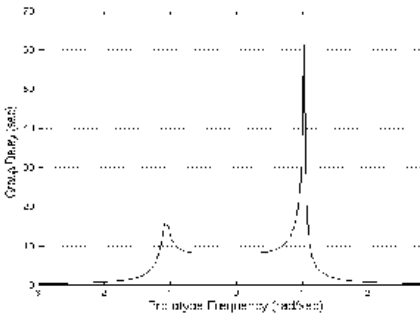
Fig. 9 Simplified 10^{th} degree topology.

Note that besides the removal of a cross-iris we have also lowered the reduced order of our target topology from 384 to 2. This is important if one wants to use a computer aided tuning process (Seyfert *et al.* , 2003) that typically identifies a coupling matrix from measured data. In the cases of topologies with multiple solutions, such a tool will return a set of equivalent coupling matrices and leave to the user the 'expert' task of choosing the "right" one. This can be done by using some extra information concerning the physical device, like for example an a priori estimation of the coupling value realizable by some irises. Nevertheless the latter task is of course much easier to carry out with a short list of equivalent coupling matrices than with a huge one.

17.2.3 Les matrices de couplage en degré 10



(a)



(b)

Fig. 10. 10-2-2 asymmetric characteristic: (a) rejection and return loss (b) group delay.

0.0145	0.7712	0	0.3879	0	0	0	0	0	0	
0.7712	0.2493	-0.5232	0	0	0	0	0	0	0	
0	-0.5232	0.0554	0.1925	0	-0.5393	0	0	0		0
0.3879	0	0.1925	-0.9071	-0.0010	0	0	0	0	0	
0	0	0	-0.0010	-0.7492	-0.2683	0	0.3110	0		0
0	0	-0.5393	0	-0.2683	0.0437	-0.4668	0	0		0
0	0	0	0	0	-0.4668	0.3195	-0.4934	0		-0.2040
0	0	0	0	0.3110	0	-0.4934	-0.1000	0.4827		0
0	0	0	0	0	0	0	0.4827	-0.0021		0.8388
0	0	0	0	0	0	-0.2040	0	0.8388		0.0145

Fig. 11. Coupling matrix of the 10-2-2 characteristic of Fig. 10 with the extended box topology and a "small" M_{45} coupling, R_1=R_N= 1.04326.

0.0161	0.7655	0	0.4053	0	0	0	0	0	0
0.7655	0.2705	-0.5173	0	0	0	0	0	0	0
0	-0.5173	0.0560	0.2057	0	-0.5386	0	0	0	0
0.4053	0	0.2057	-0.8923	0	0	0	0	0	0
0	0	0	0	-0.7810	-0.2512	0	0.2968	0	0
0	0	-0.5386	0	-0.2512	0.0445	-0.4761	0	0	0
0	0	0	0	0	-0.4761	0.2867	-0.5041	0	-0.1984
0	0	0	0	0.2968	0	-0.5041	-0.0850	0.4851	0
0	0	0	0	0	0	0	0.4851	0.0016	0.8427
0	0	0	0	0	0	-0.1984	0	0.8427	0.0173

Fig. 12. Coupling matrix of the 10-2-2 characteristic of Fig. 10 with a simplified topology, (i.e. M_{45}=0) , R_1=1.0969, R_N=1.0963.
0.01610.405300.7655000000

0.4053	-0.8923	-0.2057	0	0	0	0	0	0	0
0	-0.2057	0.0560	0.5173	0	-0.5386	0	0	0	0
0.7655	0	0.5173	0.2705	0	0	0	0	0	0
0	0	0	0	-0.7810	-0.2512	0	0.2968	0	0
0	0	-0.5386	0	-0.2512	0.0445	-0.4761	0	0	0
0	0	0	0	0	-0.4761	0.2867	-0.5041	0	0.1984
0	0	0	0	0.2968	0	-0.5041	-0.0850	-0.4851	0
0	0	0	0	0	0	0	-0.4851	0.0016	0.8427
0	0	0	0	0	0	0.1984	0	0.8427	0.0173

Fig. 13 Coupling matrix with a simplified topology and the most asymmetric irises, R_1=1.0969, R_N=1.0963.

17.3 Le Problème

17.3.1 Notations

$\{1; n\} = \{1, 2, \ldots, n\}$
K a field (in practice the field of rational coefficients).

n integer.

If A is $n \times n$ matrix with coefficients in K then ${}^TK$ is the transposed matrix. We denote by $A_{i,j}$ the elements of A; $\mathrm{Tr}(A) = \sum_{i=1}^{n} A_{i,i}$ is the trace of A. $\mathrm{col}(A, j)$ is the j column of A.

I_n is the $n \times n$ identity matrix.

K^n a K-vector space and e_i the canonical basis (hence $e_1 = \mathrm{Col}(I_n, 1) = {}^T(1, 0, \ldots, 0)$ and $e_n = \mathrm{Col}(I_n, n) = {}^T(0, \ldots, 0, 1)$).

If $v = \sum_{i=1}^{n} v_i e_i$ and $w = \sum_{i=1}^{n} w_i e_i$ are vectors then we define the usual scalar product:

$$\langle v, w \rangle = \sum_{i=1}^{n} v_i w_i$$

Definition 17.3.1. *Let $\mathcal{J}$ be a subset of $\{1; n\}^2$. We say that A a $n \times n$ symmetric matrix A has shape $\mathcal{J}$ if:*

$$A_{i,j} = 0 \text{ if } (i, j) \in \mathcal{J}$$

In other words, $\mathcal{J}$ is the list of zero elements in the matrix A.

17.4 Le problème mathématique

Let n be an integer,

Problem 17.4.1. F a $n \times n$ symmetric matrix is given and a subset $\mathcal{J}$ of $\{1; n\}^2$ is fixed. The problem is to find a $n \times n$ symmetric matrix A with shape $\mathcal{J}$ such that there exists an orthogonal $n \times n$ matrix such that:

$$A = {}^TP.F.P \quad \text{with } {}^TP.P = P.{}^TP = I_n$$

In other words the matrix A is *partially* given and we shall find all the possible equivalent matrices of F with a prescribed shape.

Note 17.4.1. If A is a solution of Problem 17.4.1 then the characteristic polynomial of F and M are the same; equivalently $Tr(A^i) = Tr(F^i)$ for all i.

Note 17.4.2. In general there is not *one* solution to the previous problem.

In the following we will show that in the particular case of "coupling matrices", Problem 17.4.1 is equivalent to the following question (see proposition 17.4.2):

Problem 17.4.2. Let assume that $R \in K^{n-1}$, $L \in K^{n-1}$ and $M \in K^{n-1}$ are given and a subset $\mathcal{J}$ of $\{1; n\}^2$ is fixed. The problem is to find a $n \times n$ symmetric matrix A with shape $\mathcal{J}$ such that:

$$\left\langle A^i e_1, e_1 \right\rangle = L_i \quad \left\langle A^i e_n, e_n \right\rangle = R_i \quad \left\langle A^i e_1, e_n \right\rangle = M_i$$

The problem stated in the first definition is very general and in this article we address only a a particular instance of this problem related to the synthesis of coupling matrices for microwave filters; consequently in the rest of the paper we assume that F is an "arrow" matrix: $F_{i,j} = 0$ for all $j + 1 < i < n$

$$F = \begin{pmatrix} . & . & 0 & 0 & 0 & . \\ . & . & . & 0 & 0 & . \\ 0 & . & . & . & 0 & . \\ 0 & 0 & . & . & . & . \\ 0 & 0 & 0 & . & . & . \\ . & . & . & . & . & . \end{pmatrix} \text{ where a dot . means a non zero element}$$

An important parameter is the number of zeros of the last row of F:

$$N_z = \#\max_j \left(\{j \in \{1; n\} \mid F_{n,1} = \cdots = F_{n,j} = 0\}\right)$$

In practice A is a *coupling matrix*, thus the typical shape for A is a tridiagonal symmetric matrix with few extra non zero elements.

Example 17.4.1. Here is an example of coupling matrix with $n = 8$ and 4 non zeros elements $A_{2,5} \neq 0$ $A_{1,3} \neq 0$ $A_{3,6} \neq 0$ $A_{5,8} \neq 0$.

$$A_8 = \begin{pmatrix} . & . & . & 0 & 0 & 0 & 0 & 0 \\ . & . & . & 0 & . & 0 & 0 & 0 \\ . & . & . & . & 0 & . & 0 & 0 \\ 0 & 0 & . & . & . & 0 & 0 & 0 \\ 0 & . & 0 & . & . & . & 0 & . \\ 0 & 0 & . & 0 & . & . & . & 0 \\ 0 & 0 & 0 & 0 & 0 & . & . & . \\ 0 & 0 & 0 & 0 & . & 0 & . & . \end{pmatrix}$$

For this example, the corresponding arrow form of the matrix has $N_z = 3$ zeros. We will use this example in several place in the paper to illustrate various ways of converting Problem 17.4.1 and 17.4.2 into algebraic equations.

Moreover another constraint coming from the application is that P, the similarity transform, has the following shape:

$$P = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & P' & & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \text{ with } {}^T P'.P' = P'^T P' = I_{n-2}$$

17.4.1 La forme flèche d'une matrice

Compute the arrow form of a matrix A is very easy and can be done with a slightly modified version of the Gram-Schmidt algorithm:

Algorithme 38 *Compute Arrow form*
Input: *A* $n \times n$ *matrix*
Output: *F, P*
$w_0 = e_n$ $w_1 = e_1$
for i ***from*** 2 ***to*** $n-1$ ***do***
$w_i = Aw_{i-1}$
$w_i = w_i - \sum_{j=1}^{i-1} \langle w_i, w_j \rangle w_j$
$N_i = \langle w_i, w_i \rangle^{\frac{1}{2}}$
$w_i = \frac{w_i}{N_i}$
P *the matrix whose columns are* $w_1, w_2, \ldots, w_{n-1}, w_0$
$F = {}^T P.A.P$
Return *F, P*

Proof. Clearly $\langle w_i, w_j \rangle = 0$ if $i \neq j$ thus ${}^T P.P = I_n$. Moreover F is the matrix representation of A in the basis $[w_1, \ldots, w_{n-1}, w_0]$ so that (recall that $w_1 = e_1$ and $w_0 = e_n$):

$$
\begin{aligned}
&Ae_1 = N_2 w_2 + \langle Ae_1, e_1 \rangle e_1 + \langle Ae_1, e_n \rangle e_n \\
&Aw_2 = N_3 w_3 + \langle Aw_2, e_1 \rangle e_1 + \langle Aw_2, w_2 \rangle w_2 + \langle Aw_2, e_n \rangle e_n \\
&Aw_3 = N_4 w_4 + \langle Aw_3, e_1 \rangle e_1 + \langle Aw_3, w_2 \rangle w_2 + \langle Aw_3, w_3 \rangle w_3 + \langle Aw_3, e_n \rangle e_n \\
&\vdots
\end{aligned}
$$

but if we compute $\langle Aw_3, e_1 \rangle = \langle w_3, Ae_1 \rangle = \langle w_3, N_2 w_2 + \S < Ae_1, e_1 \rangle e_1 + \langle Ae_1, e_n \rangle e_n >= 0$. And by induction we have that:

$$
F = \begin{pmatrix}
\langle Ae_1, e_1 \rangle & \langle Aw_2, e_1 \rangle & 0 & \cdots \\
N_2 & \langle Aw_2, w_2 \rangle & \langle Aw_3, w_2 \rangle & \cdots \\
\langle Aw_2, e_1 \rangle & N_3 & \langle Aw_3, w_3 \rangle & \cdots \\
0 & \langle Aw_3, w_2 \rangle & N_4 & \cdots \\
0 & 0 & \cdots & \cdots \\
\vdots & \vdots & \vdots & \cdots \\
\langle Ae_1, e_n \rangle & \langle Aw_2, e_n \rangle & \langle Aw_3, e_n \rangle & \cdots
\end{pmatrix}
$$

Corollary 17.4.1. *We have the relation between A and F:*

$$F_{i,j} = \langle Aw_i, w_j \rangle = \langle w_i, Aw_j \rangle$$

where w_i *are the columns of the orthogonal matrix P. Similarly*

$$A_{i,j} = \langle Fv_i, v_j \rangle = \langle v_i, Fv_j \rangle$$

where v_i *are the rows of the orthogonal matrix P.*

Proposition 17.4.1. *Up to a change of sign the arrow form of matrix A is unique. More precisely if we fix the direction of each column $col(F,i)$, for $i = 2, \ldots, n-2$ the result is unique.*

Proof. From the description of algorithm 38 it is clear that we can take $N_i = \pm \langle w_i, w_i \rangle^{\frac{1}{2}}$; hence for a given matrix A there are 2^{n-2} arrow forms. Suppose that (F (resp. F') is an arrow form of A such that $F = {}^TP.A.P$ (resp. $F'^TP'.A.P'$). We denote by w_i (resp. v_i) the i-th column of P (so that $w_1 = v_1 = e_1$ and $w_n = v_n = e_n$) and by φ the linear map whose matrix is A w.r.t. the canonical basis $(e_1, \ldots, e_n)$. W.r.t. the basis $(w_1, \ldots, w_n)$ (resp. $(v_1, \ldots, v_n)$) the matrix of φ is F (resp. F') so that:

$$\begin{array}{rl}
\varphi(e_1) &= \varphi(v_1) = \sum_{j=1}^n F'_{1,j} v_j = F'_{1,1} e_1 + F'_{1,2} v_2 + F'_{1,n} e_n \\
&= \varphi(w_1) = \sum_{j=1}^n F_{1,j} w_j = F_{1,1} e_1 + F_{1,2} w_2 + F_{1,n} e_n \\
\vdots & \\
\varphi(v_i) &= \sum_{j=1}^n F'_{i,j} v_j = F'_{i,i-1} v_{i-1} + F'_{i,i} v_i + F'_{i,i+1} v_{i+1} + F'_{i,n} e_n \\
\varphi(w_i) &= \sum_{j=1}^n F_{i,j} w_j = F_{i,i-1} w_{i-1} + F_{i,i} w_i + F_{i,i+1} w_{i+1} + F_{i,n} e_n \\
\vdots & \\
\varphi(e_n) &= \varphi(v_n) = \sum_{j=1}^n F'_{n,j} v_j \\
&= \varphi(w_n) = \sum_{j=1}^n F'_{n,j} w_j
\end{array}$$

Using the orthogonality conditions we have successively that:

$$\begin{array}{l}
F'_{1,1} = \langle \varphi(e_1), e_1 \rangle = F_{1,1} \\
F'_{1,n} = \langle \varphi(e_1), e_n \rangle = F_{1,n} \\
F'_{1,2} v_2 = \varphi(e_1) - F'_{1,1} e_1 - F'_{1,n} e_n = F_{1,2} w_2 \\
\text{since } \langle v_2, v_2 \rangle = 1 = \langle w_2, w_2 \rangle \text{ we have } |F'_{1,2}| = |F_{1,2}|
\end{array}$$

Let ϵ_2 be $\frac{F'_{1,2}}{F_{1,2}} = \pm 1$ so that $v_2 = \epsilon_2 w_2$. By induction, we may assume that $v_j = \epsilon_j w_j$ for $j \leq i$ (with the convention $\epsilon_1 = 1$). Using again the orthogonality conditions we have that:

$$\begin{array}{rl}
F'_{i,i-1} & = \langle \varphi(v_i), v_{i-1} \rangle = \epsilon_i \epsilon_{i-1} \langle \varphi(w_i), w_{i-1} \rangle = \epsilon_i \epsilon_{i-1} F_{i,i-1} \\
F'_{i,i} & = \langle \varphi(v_i), v_i \rangle = \epsilon_i \epsilon_i \langle \varphi(w_i), w_i \rangle = F_{i,i} \\
F'_{i,n} & = \langle \varphi(v_i), e_n \rangle = \epsilon_i \langle \varphi(w_i), e_n \rangle = \epsilon_i F_{i,n} \\
F'_{i,i+1} v_{i+1} & = \varphi(v_i) - F'_{i,i-1} v_{i-1} - F'_{i,i} v_i - F'_{i,n} e_n \\
& = \epsilon_i \varphi(w_i) - (\epsilon_i \epsilon_{i-1} F_{i,i-1})(\epsilon_{i-1} w_{i-1}) - F_{i,i}(\epsilon_i w_i) - (\epsilon_i F_{i,n}) e_n \\
& = \epsilon_i \left(\varphi(w_i) - F_{i,i-1}) w_{i-1}\right) - F_{i,i} w_i - F_{i,n} e_n) \\
& = \epsilon_i F_{i,i+1} w_{i+1} \\
\text{since} & \quad \langle v_{i+1}, v_{i+1} \rangle = 1 = \langle w_{i+1}, w_{i+1} \rangle \text{ we have } |F'_{i,i+1}| = |F_{i,i+1}|
\end{array}$$

hence $v_{i+1} = \epsilon_{i+1} w_{i+1}$.

From the equations (**??**) the following lemma is obvious:

Lemma 17.4.1. *If (??) holds then for any matrix A:*

$$A_{1,1} = \left({}^T PAP\right)_{1,1} \; A_{n,n} = \left({}^T PAP\right)_{n,n}$$
$$A_{1,n} = \left({}^T PAP\right)_{1,n} \; A_{n,1} = \left({}^T PAP\right)_{n,1}$$

We restate now Problem 17.4.1 in the particular case of interest of the coupling matrices:

Problem 17.4.3. F a $n \times n$ arrow form matrix is given and a subset $\mathcal{J}$ of $\{1; n\}^2$ is fixed. The problem is to find a $n \times n$ symmetric matrix A with shape $\mathcal{J}$ such that: there exists an orthogonal $n \times n$ matrix such that:

$$A = {}^T P.F.P \;\text{ with } {}^T P.P = P.{}^T P = I_n,\; col(P,1) = e_1,\; col(P,n) = e_n$$

Proposition 17.4.2. *Let us assume that $n > 3$. Generically Problem 17.4.2 is equivalent to Problem 17.4.3.*

Proof. a) Suppose that we can solve Problem 17.4.3.
We assume that $R \in K^{n-1}$, $L \in K^{n-1}$ and $M \in K^{n-1}$ and $\mathcal{J}$ are given. We first construct F an arrow form matrix such that $\left\langle F^i e_1, e_1 \right\rangle = L_i \;\; \left\langle F^i e_n, e_n \right\rangle = R_i \;\; \left\langle F^i e_1, e_n \right\rangle = M_i$. This can be done by solving a simple algebraic system (triangular system): the variables are non zero elements of the matrix (when $n > 3$ the number of variables is thus $3(n-1)$). Generically the system is a zero dimensional system with 2^{n-2} solutions. In a second step we solve the problem 17.4.1 for the matrix F to find a matrix A with the prescribed shape $\mathcal{J}$. From lemma 17.4.1 it is obvious that A is also a solution of problem 17.4.2.

b) Suppose now that we can solve problem 17.4.2.
If F is an arrow form matrix we solve the problem 17.4.2 for the 3 lists: $R = (\left\langle F^i e_1, e_1 \right\rangle)_i$, $S = (\left\langle F^i e_n, e_n \right\rangle)_i$, $M = (\left\langle F^i e_1, e_n \right\rangle)_i$. Let A be a solution. We define successively the vectors $w_2, w_3, \ldots$:

$$N_2 w_2 = Ae_1 - (\langle Ae_1, e_1 \rangle e_1 + \langle Ae_1, e_n \rangle e_n)$$
$$N_3 w_3 = Aw_2 - (\langle Aw_2, e_1 \rangle e_1 + \langle Aw_2, w_2 \rangle w_2 + \langle Aw_2, e_n \rangle e_n)$$
$$N_{i+1} w_{i+1} = Aw_i - \textstyle\sum_{j=i-1,i,n} \langle Aw_i, w_j \rangle w_j$$
$$\vdots$$

where N_i is such that $\langle w_i, w_i \rangle = 1$. Let P be the matrix whose columns are $[e_1, w_2, \ldots, w_{n-1}, e_n]$. From the proof of algorithm 38 we know that P is orthogonal and that

$$F'^T PAP = \begin{pmatrix} \langle Ae_1, e_1\rangle & \langle Aw_2, e_1\rangle & 0 & \cdots \\ N_2 & \langle Aw_2, w_2\rangle & \langle Aw_3, w_2\rangle & \cdots \\ 0 & N_3 & \langle Aw_3, w_3\rangle & \cdots \\ 0 & 0 & N_4 & \cdots \\ 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \cdots \\ \langle Ae_1, e_n\rangle & \langle Aw_2, e_n\rangle & \langle Aw_3, e_n\rangle & \cdots \end{pmatrix}$$

By definition $\langle Ae_1, e_1\rangle = \langle Fe_1, e_1\rangle = F_{1,1}$; we compute $N_2 \langle Aw_2, e_1\rangle = \langle N_2 w_2, Ae_1\rangle = \langle A^2 e_1, e_1\rangle - \langle Ae_1, e_1\rangle^2 - \langle Ae_1, e_n\rangle^2 = \langle F^2 e_1, e_1\rangle - \langle Fe_1, e_1\rangle^2 - \langle Fe_1, e_n\rangle^2 = (F_{1,1}^2 + F_{1,2}^2 + F_{1,n}^2) - F_{1,1}^2 - F_{1,n}^2 = F_{1,2}^2$. On the other hand $N_2^2 = \langle N_2 w_2, N_2 w_2\rangle = \langle N_2 w_2, Ae_1\rangle = F_{1,2}^2$. If we take $N_2 = F_{1,2}$ we have that $\langle Aw_2, e_1\rangle = F_{1,2}$. In the same for the other coefficients of F' we can proof that $F' = F$; hence A, P is a solution of problem 17.4.3.

17.5 Mise en équations algébriques

The goal of this section is to explain how derive an algebraic set of equations. Since we have two equivalent Problems (17.4.3 and 17.4.2) we have two different set of equations and variables. We evaluate the difficulty of solving the corresponding set of algebraic equations by computing a Gröbner basis for a DRL ordering in the polynomial ring $R = \mathbb{F}_{65521}[x_1, \ldots]$. In particular we compute the Gröbner basis for the example 17.4.1: it takes from 4185 sec to 2.7 sec depending for this small example !

In the rest of this section we may assume that F, the arrow form matrix, is fixed.

17.5.1 En utilisant la formulation du problème 17.4.2

In that case the problem is to find a matrix A with a prescribed shape such that

$$\left\langle A^i e_1, e_j\right\rangle = \left\langle F^i e_1, e_j\right\rangle \quad j \in 1\{1, n\} \quad i = 1, 2, \ldots \tag{17.7}$$

The variables are the non zero element of A; with respect to the number of variables this solution seems the most promising since the number of variables is proportional to n. The first disadvantage of equations (17.7) is that the degree of these is equations is very big: more precisely the equation $\left\langle A^i e_1, e_j\right\rangle - \left\langle F^i e_1, e_j\right\rangle = 0$ is an algebraic equation of degree i.

Note 17.5.1. It is also possible to add the trace equations:

$$Tr(A^i) = Tr(F^i)\ i = 1, 2, \ldots \tag{17.8}$$

Example	Eqs (17.7) + Traces F_4	Number of solutions
Ex 17.4.1: $n = 8$ $N_z = 2$	3006.4 sec	79×2^6

Note 17.5.2. A consequence of the proposition 17.4.1 is that number of solutions of the corresponding algebraic system is always a multiple of 2^{n-2}.

17.5.2 En utilisant la formulation du problème 17.4.3

The idea is to generate an overdertemined system of low degree equations (in practice the total degree of each equation should be less than 4). The variables are now the non zero elements of the matrix P (hence the number of variables is about $(n-2)^2$). Obvious equations are $(1 < i, j < n)$:

$$\begin{array}{l} \left({}^T P.P\right)_{i,j} = 0 \ \text{ if } i \neq j \\ \left({}^T P.P\right)_{i,i} = 1 \\ \left({}^T P.F.P\right)_{i,j} = 0 \ \text{ if } (i,j) \notin \mathcal{J} \end{array} \tag{17.9}$$

Example	Eqs (17.9)
Ex 17.4.1: $n = 8$ $N_z = 2$	Too large
Nb of solutions	?

Normalization equations:

${}^T P$

As already noticed in the previous section, equations $({}^T P.P)_{i,i} = \sum_{j=2}^{n-1} P_{j,i}^2 = 1$ induced "parasite" solutions: if $\epsilon_i = \pm 1$ then by multiplying the column i of P by ϵ_i we obtain another solution. A more efficient technique is to first compute an non normalized orthogonal matrix P' removing equations $\left({}^T P.P\right)_{i,i} = 1$; in a second step it is easy to obtain from any matrix P' an orthogonal matrix P providing that P' is non singular. In order to specify efficiently that the norm of the j-th column of P' is non zero we first add the inequation $det(P) \neq 0$:

$$u.det(P) - 1 = 0 \tag{17.10}$$

and we replace the previous quadratic equations by linear equations: $\sum_{j=2}^{n-1} \alpha_{j,i} P_{j,i} = 1$ where the $\alpha_{i,j}$ are random numbers. A variant is to take simply $P_{i,i} = 1$.

Formulation	Formulation bis
$\sum_{j=2}^{n-1} \alpha_{j,i} P_{j,i} = 1$	$P_{i,i} = 1$
$({}^T P.P)_{i,j} = 0 \ if \ i \neq j$	
$({}^T P.F.P)_{i,j} = 0 \ if \ (i,j) \notin J$	

(17.11)

Example	Eqs (17.10)+(17.11) F_4	Nb of solutions
Ex 17.4.1: $n = 8$ $N_z = 2$	4185 sec	79

Power equations: it is possible to improve very much the computation of Gröbner basis by taking into account the following relations:

$$A^k = {}^T P.F^k.P \quad k = 1, 2, 3, \ldots$$

Note that these equations are not *necessary* from a strict mathematical point of view because they belong the ideal generated by equations (17.11) but they are used only to *speedup* the computation. We include the equations (17.11):

$$({}^T P.F^k.P)_{i,j} = 0 \text{ if } (i,j) \notin \mathcal{J},\ k = 1, 2, 3, \ldots \tag{17.12}$$

Now the algebraic system is very overdertemined, hence it we can remove equations (17.10).

Example	Eqs (17.11)+(17.12) F_4	Nb of solutions
Ex 17.4.1: $n = 8$ $N_z = 2$	71.8 sec	79

Minor equations: we can add even more equations from the fact that:

$$P.A = F.P$$

and if $W = {}^T col(A, j)$ is the j-th column of A we have

$$\sum_{i=1}^{n} W_i \operatorname{Col}(P, i) = \operatorname{Col}(F.P, j)$$

and since $W_i = 0$ if $(i,j) \in \mathcal{J}$ we have that the matrix M_j whose columns are $[\operatorname{Col}(F.P, j), \operatorname{Col}(P, i)\ (i,j) \in \mathcal{J}]$ is not full rank.

$$\text{all minors of } M_j \text{ are equals to } 0 \tag{17.13}$$

The final step is to combine "minor equations" and "power equations" by using the relation $P.A^k = F^k.P$ and thus computing the minors of the matrix whose columns are $[\operatorname{Col}(F^k.P, j), \operatorname{Col}(P, i)\ (A^k)_{i,j} \neq 0]$. Since for large value of k A^k is not sparse we take $k \leq 3$ or 4.

Example	Eqs (17.11)+(17.12)+(17.13) F_4	Eqs (17.11)+(17.12)+(1
Ex 17.4.1: $n = 8$ $N_z = 2$	10.1 sec	2.7 sec
$n = 8$ $N_z = 3$	0.01 sec	0.01 sec
$n = 10$ $N_z = 4$	1202 sec	304 sec

Example	Nb of (useful) variables	Eqs of	Eqs of deg 2	Eqs of deg 3	Total deg 4
Ex 17.4.1: $n = 8$ $N_z = 2$	20	45	10	30	109
$n = 10$ $N_z = 4$	32	210	228	294	948

Conclusion

In this paper, a new method for the synthesis of the full range of coupling matrices for networks that support multiple solutions is presented. This procedure yields an exhaustive list of all the solutions to the synthesis problem. Based on the latter, an approximate synthesis technique is derived which allows the reduction of the constructional complexity of high-degree asymmetric filters in dual-mode technologies. In addition it has been shown that a knowledge of which solutions are possible is important when reconstructing the coupling matrix from measured data, during development or computer-aided tuning (CAT) processes. Un aspect fondamental est la mise en équation du problème pour utiliser la structure.

A software called Dedale-HF and dedicated to the presented exhaustive synthesis technique has recently been released and is accessible under:

`http://www-sop.inria.fr/apics/Dedale`

Part IV

Annexes

18. Challenges

18.1 Challenge 1 HFE

The first HFE Challenge (see the description in chapter 8) was proposed in (Patarin, 1996c) with a (symbolic) prize of 500$. This correspond to a HFE($d = 96, n = 80$) problem and can be downloaded from (Patarin, 1996a). For this problem, the exhaustive search attack require $\geq 2^{80}$ operations, hence is not feasible.

We have computed a Gröbner basis of this system with the algorithm F_5 (in fact a special version for $\mathbb{F}_2$) implemented in the FGb (Fast Gb) Gröbner program (written in C). As explained in section **??** the most time consuming part is linear algebra: for this example we have solve a 307126×1667009 matrix over $\mathbb{F}_2$. The total running time was 187892 sec (≈ 2 days and 4 hours) on an HP workstation with an alpha EV68 processor at 1000 Mhz and 4Go bytes of RAM. Some care has been taken for the memory management since the size of the process was 7.65 Giga bytes.

For this algebraic systems (Patarin, 1996a) we found that there were *four solutions*:

$$X = 644318005239051140554718 \quad X = 934344890045941098615214$$
$$X = 102267771362902876120304 6 \; X = 103704608265180114959467 0$$

where $X = \sum_{i=1}^{80} x_i 2^{i-1}$.

It must be emphasized that this computation is far beyond the capacity of all the other implementations and algorithms for computing Gröbner basis as is made clear by the following table (B. stands for the Buchberger algorithm):

Because 80 equations of degree 2 was a previously untractable problem, this Gröbner computation represents a breakthrough in research on polynomial system solving.

18.2 Challenge 1 de H. Dobbertin

CryptoChallenge 11 is broken or an efficient attack of the C^* cryptosystem.

	Algo	10	12	16	18	19	21	33
Maple	B.	71.7 s	587.9 s	×	×	×	×	×
Magma	B.	×	1.5 s	135.4 s	1900 s	×	×	×
Gb	B.	×	0.8 s	105 s	443.2 s	×	×	×
Singular	B.	×	0.7 s	55.5 s	334.3 s	663.4 s	×	×
FGb	F_4	×	×	12.3 s	70.5 s	133.2 s	436.9 s	×
FGb	$F_5/2$	×	×	×	**0.9 s**	**1.5 s**	**4.25 s**	**442.7 s**

Table 18.1. Comparison of various algorithms and implementations 2003 (PC PIII 700 Mhz)

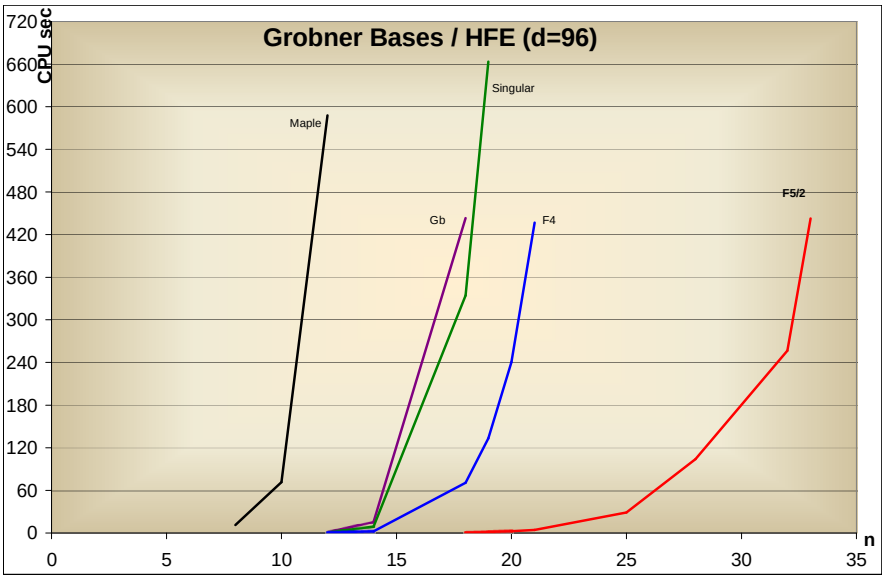


Fig. 18.1. Comparaison des temps de calcul pour HFE avec différents systèmes de Calcul Formel.

18.2.1 Description

Mystery Twister was an international crypto competition organized by the

Cryptology and IT Security (CITS) Research Group
`www.cits.ruhr-uni-bochum.de`

of the Ruhr-University Bochum (Hans Dobbertin).

Most users are not aware of how strong or weak particular encryption systems are or as how reliably a digital certificate can be assessed. For many people the role of cryptography - a fundamental and very important tool for IT security - is non-transparent and sometimes even considered as being a bit magical and mystical ... or simply "cryptic".

The level of acceptance of applications like digital signatures, home banking and e-commerce is not increasing at the expected rate. There are certainly

also other major causes, but not least a lack of transparency is responsible. As a result, a large part of the innovative potential of IT is not being exploited. The CITS research group, have taken the initiative to organize the project MysteryTwister (see `http://www.mystery-twister.com/`).

The C^* crypto system. Let $q = 2^m$, $D = 4$, $\mathbb{K}$ is the finite field $\mathbb{F}_q$ and $\mathbb{L} = \mathbb{F}_{2^{n\,m}}$ an extension field of $\mathbb{K}$.

$$\pi(x) = x^d,\ d = \sum_{k<D} q^{r_k} \qquad r_0 < r_1 < \cdots < r_{D-1} < n$$

Proposition 18.2.1. *$\pi(x)$ is bijective on $\mathbb{K}$ if and only if* $\gcd(d, q^n - 1) = 1$

The private key consists of two bijective $\mathbb{K}$-affine mappings S and T on L, each represented by a non-singular $n \times n$ matrix over $\mathbb{K}$ and a vector in $\mathbb{K}^n$. The encryption is defined as

$$E(x) = S(\pi(T(x))), \quad x \in L.$$

Knowing S and T allows to decrypt, since one-to-one affine mappings and power functions can easily be inverted. The public key is the multi-variate representation of $E(x)$ on $\mathbb{K}^n$, leading to a collection of n polynomials (of total degree 4):

$$\text{(Pub)} \qquad E_i(x_0, \ldots, x_n)\ i = 0, \ldots, (n-1)$$

Challenge 11. The challenge 11 is a particular instance of the C^* cryptosystem: $m = 5$ and $n = 16$ and one has to solve the following algebraic system

$$(\mathcal{S}) \qquad E_i(x_0, \ldots, x_n) = Q_i,\ i = 0, \ldots, (n-1)$$

where Q_i is explicitly given: $(Q_0, \ldots, Q_{n-1}) = (w^2+1, w^2+1, w^3+w, w^4+w^3+w^2, w^2+1, w^4+w^3+w+1, w^3+w, w^3+w^2+w, 1, w^4, w^3+w^2+w+1, w^2+w+1, w^4+w^2+w+1, w^4+w^3+1, w, w^4+w^3+w^2+w+1) \in K^n$

and E_i is a polynomial of degree 4 in $x_0, \ldots, x_{n-1}$ and coefficients in $\mathbb{K}$. To "break" the challenge we have to compute

$$\mathcal{V}_{\mathbb{K}} = \{(x_i) \in \mathbb{K}^n \mid E_i(x) = Q_i\}$$

Proposition 18.2.2. *$\mathcal{V}_{\mathbb{K}}$ is of dimension* 0 *and degree* 1.

Proof. From the proposition 18.2.1.

18.2.2 Gröbner bases attack

Solutions in the ground field: If $\overline{\mathbb{K}}$ is the algebraic closure of $\mathbb{K}$ then a Gröbner basis computation of $\mathcal{S}$ gives a description of

$$\mathcal{V}_{\bar{K}} = \{(x_i) \in \overline{\mathbb{K}}^n \mid E_i(x) = Q_i\}$$

m	n	$degree(\mathcal{V}_{\bar{K}})$	$degree(\mathcal{V}_K)$
5	8	32	1
5	9	76	1
5	10	88	1
5	11	4	1
5	12	112	1
5	13	628	1
5	14	568	1
5	15	5324	1
5	16	6208	1
6	8	32	1
6	10	88	1
6	14	568	1

We deduce from the previous experiments that computing directly the Gröbner basis of $\mathcal{S}$ contains parasite solutions.

First attack. To force the solutions to be in $\mathbb{K}$ we can add the "field equations" $x_i^q = x_i$ but since $q = 32$ this is useless in this case. The other solutions is to give a value to one (or several) variable:

Algorithm 1st attack
for i **from** 0 **to** $m - 2$ **do**
Substitute $x_n = w^i$ in $\mathcal{S}$
Compute G_i a Gröbner basis of this system
if $G_i \neq \{1\}$
Add the field equations $x_j^q = x_j$ to G_i
Compute G_i' a Gröbner basis of this system
This is the solution of the algebraic system.

Of course all the computations can be computed in parallel of different computers. In our case, we used the F_4 algorithm we found a solution for $n = 20$:

$x_1 = w^{27}, x_0 = w^{16}, x_2 = w^{18}, x_3 = w^{17}, x_4 = w^{25}, x_8 = w^{16}, x_9 = w^{22}, x_{10} = w^{28}, x_{11} = w^{12}, x_{15} = w^{20}, x_{14} = w^{18}, x_{13} = w^{16}, x_{12} = w^{11}, x_5 = w^4, x_7 = w^{29}, x_6 = w^{29}$

It takes about 3 hours of CPU (Intel Pentium Xeon 2.8 Ghz) to compute the Gröbner basis G_i. Thus the total sequential time is $32 \times 3 = 96$ hours of computation and 3 hours of parallel CPU time.

New attack. We use a different strategy: compute the Gröbner basis of the whole system $\mathcal{S}$ using the F_5 algorithm; in a second step (the fastest part of the computation) we select the solutions in $\mathbb{K}^n$. The computation can be carried out on a single PC (2.8 Ghz) with 2Go bytes of memory and it takes 11473.8 sec to compute the Gröbner basis; the number of solution is 6208 in $\overline{\mathbb{K}}^n$ and the size of the Gröbner basis is 266 Mbytes.

18.2.3 Complexity of the attack

It takes about 3 hours of CPU (Intel Pentium Xeon 2.8 Ghz) to compute the Gröbner basis. For the challenge 1/HFE the total CPU is about 48 hours (Alpha DS25 1 Ghz).

However to compare more precisely this is interesting to compare two parameters:

- The number of arithmetical operations.
- The maximal degree occurring in the computation of the Gröbner bases.

Number of operations. The total number of operations (XOR with 64 bits words) to break HFE challenge 1 is about $2^{44.46}$.

The total number of operations (multiplication if $\mathbb{F}_{32}$) to break C^* challenge 11 is about $2^{41.2}$ in less that 3 hours 11 mins.

Maximal degree. The maximal degree occurring in the computation of a Gröbner basis is a very important since

- It is an estimate of the non randomness of an algebraic system. The regularity D of a generic system is given by the Macaulay bound and in our case we should have: $1 + \sum_{i=1}^{n}(d_i - 1) = 3n + 1$
- It gives a (rough) estimate of the complexity of the Gröbner basis computation: $O\left(\binom{n+D}{n}^{\omega}\right)$ where N^ω is the cost of the multiplication of two $N \times N$ matrices.

d	16	17	33	96	128	129	257	384	512	513
Max degree	3	4	4	4	4	5	5	5	5	6

Maximal degree for HFE, $d = 2^i + 2^j$ GF(2)

From the previous table we see that when d (degree of the univariate polynomial) is fixed the corresponding HFE can be solved in polynomial time (see also the paper Faugère/Joux Crypto 2003).

n	8	9	10	11	12	13	14	15	**16**
Max degree	6	7	7	6	6	6	8	8	8
Nb of $*$	$2^{20.7}$	$2^{22.4}$	$2^{23.8}$	$2^{24.8}$	$2^{26.1}$	$2^{29.1}$	$2^{31.5}$	$2^{37.9}$	$2^{41.2}$

Maximal degree for C^*, $d = 1 + q + q^5 + q^7$, $q = 2^5 = 32$

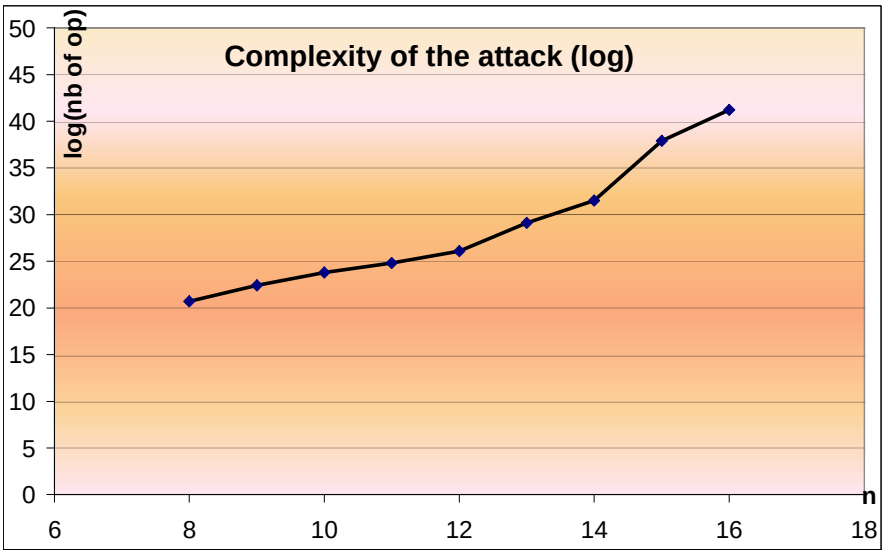


Fig. 18.2. C*: extrapolation complexité de l'attaque.

New attack using F_5

n	8	10	12	13	14	15	16	18
Max degree	6	6	6	6	7	7	7	7 or 8
Nb of $*$ (parallel)	$2^{20.7}$	$2^{23.8}$	$2^{26.1}$	$2^{29.2}$	$2^{30.0}$	$2^{37.8}$	$\times$	$\times$
Nb of $*$ (sequent)	$2^{25.7}$	$2^{28.8}$	$2^{31.1}$	$2^{34.2}$	$2^{35.0}$	$2^{42.8}$	$\times$	$\times$

Maximal degree for C^*, $d = 1 + q + q^5 + q^7$, $q = 2^5 = 32$,
Specialise one variable.
First attack using F_4

Note 18.2.1. Note that for $n = 18$, $gcd(d, q^n - 1) = 19$. The test has been done by specialising 2 and 3 variables.

We see from the previous table that C^* *perhaps* can be solved in polynomial time when d is fixed since the maximal degree occurring in the computation does not depends on n. Of course a mathematical proof is needed.

Note 18.2.2. The complexity does not depend on $m > 2$ the size of the finite field $\mathbb{F}_{2^m}$ (In fact the complexity depends on m but slightly).

Proof. This is the the result of experiments for $m = 3, 4, 5, 6, 7$. For $m = 2$ it is necessary to add the field equations $x_i^4 - x_i$ and the Gröbner computations behave differently.

18.2.4 Conclusion

If we extrapolate the previous results we found:

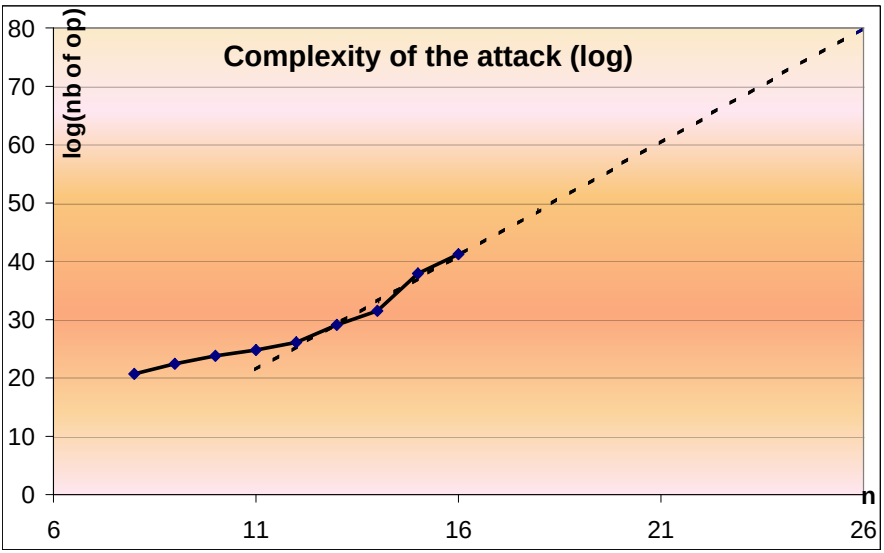


Fig. 18.3. C*: extrapolation complexité de l'attaque.

The CryptoChallenge 11 can	be broken easily $2^{41.1}$ (about 3 hours)
To improve the	security of C^* it seems that Increase the size of $\mathbb{K} = GF(2^m)$ is *useless* Increase the value $n \geq 26$ (?) can lead to a much more difficult algebraic system.

Despite a substantial amount of real computer simulations, *major uncertainties* remain.

19. Logiciels

19.1 Utilisation des logiciels FGb et Gb en Maple

Les logiciel Gb et FGb sont des logiciels écrit en C++/C dédiés pour le calcul des bases de Groebner. Gb contient des implantations des algorithmes de Buchberger et FGLM alors que FGb regrouppe les algorithmes F_4, F_5. Afin de faciliter l'usage de ces logiciels une interface a été réalisée avec le système de Calcul Formel généraliste Maple et le code objet de Gb et FGb peut être chargé dynamiquement dans le noyau de Maple. À partir de la version 11 de Maple ces logiciels (ainsi que le package RS de Fabrice Rouillier) seront distribués direcement avec Maple. Pour les versions 9.5 et 10 on peut charger FGb sous forme de librairie dynamique (disponiblité pour 7 architectures) et de packages Maple à l'adresse suivante :

`http://fgbrs.lip6.fr/jcf/Software/FGb/Download/`

On peut utiliser ces logiciels au moyen de trois interfaces:

- Interface incluse dans le logiciel Maple à partir de la version 11 (package `fgbrs).` Un sous-ensemble des fonctions.
- Gb interface de haut niveau. On peut choisir plusieurs algorithmes pour calculer une base de Gröbner (Buchberger ou F_4 par exemple).
- FGb interface de base niveau (pour experts)

19.1.1 Gb interface

Permet d'utiliser FGb au travers de l'interface commune Gb. La liste des fonctions est:

List of functions:

- gbasis computes a Groebner basis.
- mingbasis computes a Groebner basis from a non reduced Groebner basis.
- normalf computes the normal Form.
- fglm computes a Groebner basis with the FGLM algorithm. *(ZERO DIM)*
- lextriangular computes a list of triangular sets with the Lazard algorithm. *(ZERO DIM)*

- hilbert computes the hilbert polynomial of a Groebner basis.
- dimension computes the dimension of the ideal.
- vdegree computes the degree of the ideal.
- LM computes the leading monomial of a polynomial.
- LT computes the leading term of a polynomial.

<*ZERO DIM*> means that the algorithm apply only if the ideal is a zero dimensional ideal.

The constructors are available:

- pretend assumes that a list of polynomial is a Groebner basis.
- DRL polynomial ring with DRL ordering
- Lex polynomial ring with Lex ordering

Administrative functions:

- advance For experts only ! Change the association betwen Maple and a serveur.
- GbHome change the location of Gb home directory
- Version the current version of then interface Gb Maple.

Exemple:

```
> with(FGb):
   FGb/Mapleinterface package Version 1.34
   JC Faugere (jcf@calfor.lip6.fr)
   Type ?FGb for documentation
> with(Gb):
Gb/Maple interface package Version 0.92
JC Faugere (Jean-Charles.Faugere@lip6.fr)
Type ?Gb for documentation
> Id1:=gbasis([x1+x2+x3+x4,x1*x2+x1*x4+x2*x3+x3*x4,
  x1*x2*x3+x1*x2*x4+x1*x3*x4+x2*x3*x4,
  x1*x2*x3*x4-1],DRL([x1,x2,x3,x4])):
FGb: 0.01 sec 7 polys
FGb: 0.01 sec Maple: 0.05 sec
> hilbert(Id1,T);
FGb: 0.00 sec 2 polys
FGb: 0.00 sec Maple: 0.00 sec
```

$$\frac{-T^6 - T^5 + T^3 + 2\,T^2 + 2\,T + 1}{1 - T}$$

```
> abs(subs(T=1,numer(%)));
                              4
```

19.1.2 FGb interface

(experts only)

```
> with(FGb):
    FGb/Mapleinterface package Version 0.25
    JC Faugere (jcf@calfor.lip6.fr)
    Type ?FGb for documentation
> factor(fgb_gbasis([x1+x2+x3+x4,x1*x2+x1*x4+x2*x3+x3*x4,
          x1*x2*x3+x1*x2*x4+x1*x3*x4+x2*x3*x4,x1*x2*x3*x4-1],
                                    0,[x1,x2],[x3,x4]));
```

$$[(x_2+x_4)^2, x_1+x_2+x_3+x_4, (x_4-1)(x_4+1)\left(x_4^2+1\right)(x_2+x_4),$$
$$x_2\,x_3-x_2\,x_4+x_3^2x_4^4+x_3\,x_4-2\,x_4^2,$$
$$(x_4-1)(x_4+1)\left(x_4^2+1\right)(x_3\,x_4-1)(x_3\,x_4+1),$$
$$(x_3\,x_4+1)(x_3+x_4)(x_3\,x_4-1)]$$

19.2 Implantation. Gestion mémoire.

19.2.1 Exemple de mini implantation

Il est instructif de réaliser une mini implantation de l'algorithme de Buchberger. Ceci est très facile à faire dans un langage interpreté comme Maple ou Mupad. Les performances sont évidemment médiocre.

```
topReduction:=proc(f,F,ordre)
  local m1,i,m2,u:
  m1:=ordre(f):
  for i from 1 to nops(F) do
    m2:=ordre(F[i]):
    u:=m1/m2:
    if (type(u,polynom)) then
      RETURN(expand(f-u*F[i])):
    fi:
  od:
  f:
end:
reduction:=proc(f0,F,ordre)
  local q,f:
  f:=f0:
  while (f<>0) do
    q:=topReduction(f,F,ordre):
    if (q=f) then
      RETURN(f):
    else
      f:=q:
    fi:
  od:
  f:
end:
spol:=proc(p1,p2,ordre)
  local m1,m2,m:
  m1:=ordre(p1):
  m2:=ordre(p2):
  m:=lcm(m1,m2):
  expand((m/m1)*p1-(m/m2)*p2):
end:
buchberger:=proc(F,ordre)
  local G,P,n,p,f,i,j:
  n:=nops(F):
```

```
G:=F:
P:={seq(seq([F[i],F[j]],j=(i+1)..n),i=1..n)}:
while nops(P)>0 do
  p:=op(1,P):
  P:=P minus {p}:
  f:=spol(p[1],p[2],ordre):
  printf("%a -> %a",p,f);
  f:=reduction(f,G,ordre):
  printf(" -> %a\n",f):
  if (f<>0) then
    G:=[op(G),f]:
    P:=P union {seq([G[i],f],i=1..n)}:
    n:=n+1:
  fi:
od:
G:
end:
lexico:=proc(p)
  global vars:
  if (type(p,'+')) then
    op(1,sort(p,vars,plex)):
  else
    p:
  fi:
end:
```

et l'exécution du programme:

```
> vars:=[x,y,z]:
> ex1:=[x^2*y-1,x*y^2-3]:
> G:=buchberger(ex1,lexico);

[x^2*y-1, x*y^2-3] --> 3*x-y --> 3*x-y
[x^2*y-1, 3*x-y] --> x*y^2-3 --> 0
[x*y^2-3, 3*x-y] --> y^3-9 --> y^3-9
[x^2*y-1, y^3-9] --> 9*x^2-y^2 --> 0
[x*y^2-3, y^3-9] --> 9*x-3*y --> 0
[3*x-y, y^3-9] --> 27*x-y^4 --> 0
```

$$G := [x^2y - 1, xy^2 - 3, 3\,x - y, y^3 - 9]$$

les deux réductions à zéro suivantes montrent que la base n'est pas complétement réduite:

```
> reduction(G[1],[G[3],G[4]],lexico);

0

> reduction(G[2],[G[3],G[4]],lexico);

0
```

19.2.2 Implantation efficace

L'aspect purement informatique dans le travail d'implantation d'un algorithme tel que celui de Buchberger n'est pas à negliger: entre une implantation similaire d'un même algorithme dans un langage interprété comme

Maple et un langage de bas niveau comme C on peut observer un gain dans le temps de calcul en faveur de ce dernier de plus de 300.

19.3 Documentation

19.3.1 FGb[fgb_gbasis]

FGb[fgb_gbasis] - Compute a reduced Groebner basis with the C FGb program.
Calling Sequence:

fgb_gbasis(lpol,charac,vars1,vars2)
or
fgb_gbasis(lpol,charac,vars1,vars2,options)

Parameters:
lpol - a list of polynomials
charac - Z/pZ of Z if 0
vars1 - block 1 of variables
vars2 - block 2 of variables (set to [] if you need only one block)

optional parameter: options options is a set of equations for instance {"record"=0,"step"=8,"block"=0,"verb"=0}
Description:

The function gbasis computes a Groebner basis of the list of polynomials l with respect to the term order ord. The result of this computation is the Groebner basis. The result of this calculation is remembered during the session, so that another call to gbasis does not recompute the Groebner basis.

Example:

```
> with(FGb):
    FGb/Maple interface package Version 0.28
    JC Faugere (jcf@calfor.lip6.fr)
    Type ?FGb for documentation
> {1d}{vars:=[x1, x2, x3, x4, x5]:}{%
}
> {1d}{eqs:=[x3+x4+x5+x1+x2, x3*x4+x5*x4+x5*x1+x1*x2+x2*x3,
x3*x4*x5+x4*x5*x1+x5*x1*x2+x1*x2*x3+x2*x3*x4,
x3*x4*x5*x1+x4*x5*x1*x2+x5*x1*x2*x3+x1*x2*x3*x4+x2*x3*x4*x5,
x1*x2*x3*x4*x5-1]:}{%
}
```

```
> Gb1:=fgb_gbasis(eqs,0,vars,[]):
FGb: 0.01 sec 20 polys
FGb: 0.01 sec Maple: 0.25 sec
```

Number of elements in Groebner basis:

```
> nops(Gb1);
```

20

See Also: with, FGb, FGb[fgb_gbasis_lm]

19.4 Stratégies de calcul

We give for each question what is the best strategy. Be careful that these "useful tips" are only advices and not theorems: when we recommand a method this is the result of our experience so it is valid for *most* of the problems but it is always possible to find a counter example. The second point is that the answer depends on the software Gb or FGb.

Rule1

In a first time try to compute modulo p where p is a small prime.

The computations modulo p are always faster and give with a very high probability the number and the shape of solutions. Moreover if you use FGb the time of computation is approximately

(Time modulo p) $\times$ (Size of the biggest coefficient in the result)

What is the best object I can compute to solve a system

Gb	FGb
A list of Triangular Sets (Zero dim only)	A decomposition into primes

What is the best method to compute a list of Triangular Sets (Gb only)

a) Compute a DRL Gbasis modulo p. Check if it is a zero dimensional ideal.
b) Compute a DRL Gbasis with integer coefficients.
c) Change the ordering with FGLM.
d) Apply lextriangular on the resulting gbasis.

20. Décomposition en ensembles triangulaires

Dans la suite de cette section on se restreint au cas où le système à étudier est de dimension zéro c'est à dire le cas où le système admet un nombre *fini* de solutions. Le but est de transformer une base de Gröbner pour l'ordre lexicographique en liste d'ensemble triangulaires ((Lazard, 1992))

20.1 Ensembles Triangulaires

Dans la suite on prend l'ordre lexicographique pour les variables $x_1 > \cdots > x_n$.

Definition 20.1.1. *Si p est un polynôme, $mainVar(p)$ désigne la plus grande variable $x_1 > \cdots > x_n$ qui apparait dans p.*

Definition 20.1.2. *Si $t = [f_1, \ldots, f_n]$ est une liste de polynômes, on dit que c'est un ensemble triangulaire si*

(i) $mainVar(f_i) = x_i$
(ii) *f_i est unitaire en tant que polynôme en x_i; c'est à dire $f_i = x_i^{w_i} + r_i(x_i, \ldots, x_n)$ et $r_i <_{lex} f_i$.*

Definition 20.1.3. *Un idéal I de $k[x_1, \ldots, x_n]$ est* maximal *si $I \neq k[x_1, \ldots, x_n]$ si J idéal tel que $I \subset J$ implique $J = I$ ou $J = k[x_1, \ldots, x_n]$.*

Proposition 20.1.1. *Soit I un idéal de $k[x_1, \ldots, x_n]$ tel que $I \neq k[x_1, \ldots, x_n]$*

(i) *I est premier ssi $k[x_1, \ldots, x_n]/I$ est un anneau intégre*
(ii) *I est maximal ssi $k[x_1, \ldots, x_n]/I$ est un corps*

Theorem 20.1.1. *Si I est un idéal de dimension 0 alors*

$$\sqrt{I} = M_1 \cap \cdots \cap M_r$$

où M_i est un idéal maximal.

Proposition 20.1.2. *(k corps). Si M est un idéal maximal de $k[x_1, \ldots, x_n]$ alors il existe un système de générateurs triangulaire de M.*

Proof. On sait que $A = k[x_1, \ldots, x_n]/M$ est un corps. Comme il est engendré par un nombre fini d'équations c'est une extension algébrique de k. On note $A_i = k(\tilde{x}_1, \ldots, \tilde{x}_i)$ où $\tilde{x}_i$ est l'image x_i dans A. A_i est un sous corps de A. Comme $A_i = A_{i-1}(\tilde{x}_i)$ c'est une extension algébrique simple: soit p_i le polynôme minimal unitaire de $\tilde{x}_i$ (à coefficients dans A_{i-1}). Donc $A_i = A_{i-1}/p_i$. Par récurrence on a $A = k[x_1, \ldots, x_n]/(p_1, \ldots, p_n)$ et $T = (p_1, \ldots, p_n)$ est un ensemble triangulaire.

Proposition 20.1.3. *Si I est un idéal de dimension zéro alors*

$$\sqrt{I} = Id(T_1) \cap \cdots \cap Id(T_r)$$

où T_i sont des ensembles triangulaires.

20.2 Théorèmes de structure

On a dit 2.9 que la forme d'une base de Gröbner pour l'ordre lexicographique était en général particulièrement simple (voir la forme des équations 2.5), c'est donc en général un ensemble triangulaire. Sans restrictions, la forme d'une base lexicographique G d'un idéal I est la suivante:

$$\begin{cases} P_n(x_n), \\ P_{n-1,1}(x_{n-1}, x_n) \\ \cdots \\ P_{n-1,k_{n-1}}(x_{n-1}, x_n) \\ P_{n-2,1}(x_{n-2}, x_{n-1}, x_n) \\ \cdots \\ P_{n-2,k_{n-2}}(x_{n-2}, x_{n-1}, x_n) \\ \cdots \\ P_{1,1}(x_1, \ldots, x_{n-1}, x_n) \\ \cdots \\ P_{1,k_1}(x_1, \ldots, x_{n-1}, x_n) \end{cases} \tag{20.1}$$

on a un premier théorème de structure ((Lazard, 1988)) pour les polynômes en deux variables de la base:

Theorem 20.2.1. *Pour tout $i \in \{1, \ldots, k_{n-1}\}$:*
On pose $V_0 = P_n$, on a:

$$\begin{gathered} P_{n-1,i}(x_{n-1}, x_n) = V_i(x_n) \times \left(x_{n-1}^{r_i} + \textstyle\sum_{j=0}^{r_i-1} U_{i,j}(x_n) x_{n-1}^j \right) \\ avec\ deg(V_1) > \cdots > deg(V_{k_i-1}) > deg(V_{k_i}) = 0 \\ r_1 < r_2 < \cdots < r_{k_{n-1}} \\ et\ V_{k_{n-1}} |\ \cdots |\ V_2 |\ V_1 |\ V_0 \\ de\ plus\ P_{n-1,j} \in Id(\tfrac{V_{i-1}}{V_i}, \tfrac{P_{n-1,i}}{V_i}) pour\ k_{n-1} \geq j > i \end{gathered}$$

il découle immédiatement de ce théorème une décomposition partielle de $V(I)$ en ensemble triangulaires:

Corollary 20.2.1.

$$V\left(I \cap k[x_{n-1}, x_n]\right) = \bigcup_{i=1}^{k_{n-1}} \left(Id\left(\frac{V_{i-1}}{V_i}, \frac{P_{n-1,i}}{V_i} \right) \right)$$

Il est à noter que cette décomposition *ne preserve pas* les multiplicités des racines. On peut donc remplacer dès le depart le polynôme univarié $P_n = V_0$ par sa forme sans carré:

$$\frac{P_n(x_n)}{gcd(P_n(x_n), \frac{\partial P_n(x_n)}{\partial x_n})}$$

où à factoriser $P_n = U_1^{\alpha_1} \cdots U_r^{\alpha_r}$.

On peut calculer la base lexicographique de Cyclic 6 (voir la définition 2.8.3 page 43):

$G = [g_0, g_1, g_2, g_3, \ldots]$ (les autres polynômes dépendent de x_4, x_3, x_2, x_1) avec:

$g_0 = {x_6}^{48} - 2554\,{x_6}^{42} - 399710\,{x_6}^{36} - 499722\,{x_6}^{30} + 499722\,{x_6}^{18} + 399710\,{x_6}^{12} + 2554\,{x_6}^{6} - 1$

$g_1 = 1387545279120\,{x_5}^{2}{x_6}^{12} - 1387545279120\,{x_5}^{2} + 4321823003\,x_5\,{x_6}^{43} - 11037922310209\,x_5\,{x_6}^{37} - 1727510711947989\,x_5\,{x_6}^{31} - 2165150991154425\,x_5\,{x_6}^{25} - 5114342560755\,x_5\,{x_6}^{19} + 2162682824948601\,x_5\,{x_6}^{13} + 1732620732685741\,x_5\,{x_6}^{7} + 13506088516033\,x_5\,x_6 + 24177661775\,{x_6}^{44} - 61749727185325\,{x_6}^{38} - 9664106795754225\,x_6$ $12090487758628245\,{x_6}^{26} - 8787672733575\,{x_6}^{20} + 12083693383005045\,{x_6}^{14} + 9672870290826025\,{x_6}^{8} + 68544102808525\,{x_6}^{2}$

$g_2 = 25438330117200\,{x_5}^{3}{x_6}^{6} + 25438330117200\,{x_5}^{3} + 76314990351600\,{x_5}^{2}{x_6}^{7} + 76314990351600\,{x_5}^{2}x_6 - 1594966552735\,x_5\,{x_6}^{44} + 4073543370415745\,x_5\,{x_6}^{38} + 637527159231148925\,x_5\,{x_6}^{32} + 797521176113606525\,x_5\,{x_6}^{26} + 530440941097175\,x_5\,{x_6}^{20} - 797160527306433145\,x_5\,{x_6}^{14} - 638132320196044965\,x_5\,{x_6}^{8} - 4510507167940725\,x_5\,{x_6}^{2} - 6036376800443\,{x_6}^{45} + 15416903421476909\,{x_6}^{39} + 2412807646192304449\,{x_6}^{33} + 3017679923028013705\,{x_6}^{27} + 1422320037411955\,{x_6}^{21} - 3016560402417843941\,{x_6}^{15} - 2414249368183033161\,{x_6}^{9} - 16561862361763873\,{x_6}^{3}$

$g_3 = 1322793166094400\,{x_5}^{6} - 3968379498283200\,x_6\,{x_5}^{5} + 3968379498283200\,{x_6}^{2}{x_5}^{4} - 5291172664377600\,{x_6}^{3}{x_5}^{3} - 230166010900425600\,{x_5}^{2}{x_6}^{10} - 226197631402142400\,{x_5}^{2}{x_6}^{4} - 152375364610443885\,x_5\,{x_6}^{47} + 389166626064854890415\,x_5\,{x_6}^{41} + 609060978413605589877616736793460879869727 5\,x_5\,{x_6}^{29} + 27855066785995181125\,x_5\,{x_6}^{23} - 76144952817052726093362989246351754697 5\,x_5\,{x_6}^{11} - 411415071682002547795\,x_5\,{x_6}^{5} - 209493533143822353504597949056058 6\,{x_6}^{36} + 837379474964973553146\,{x_6}^{30} + 1048895070842133715701\,{x_6}^{24} - 167117997269207870\,{x_6}^{18} - 1047937257813906155 14\,{x_6}^{12} - 83842685189903180394\,{x_6}^{6} - 569978796672974242$

On applique le théorème: $g_1 = 1387545279120\,\left({x6}^{12} - 1\right)\,{x5}^{2} + \left(4321823003\,\ {x6}^{43} + \cdots\right.$ $24177661775\,{x6}^{44} + \cdots + 68544102808525\,{x6}^{2}$ donc en posant $V_1 = -1 + {x6}^{12}$ on sait V_1 divise g_1 et g_0 et on trouve le premier ensemble triangulaire:

$T_1 = [\frac{g_0}{V_1}, \frac{g_1}{V_1}] = [{x6}^{36} - 2554\,{x6}^{30} - 399709\,{x6}^{24} - 502276\,{x6}^{18} - 399709\,{x6}^{12} - 2554\,{x6}^{6} + 1, 1387545279120\,{x5}^{2} + \left(4321823003\,{x6}^{31} - 11037922310209\,{x6}^{25} + \cdots - 135\right.$ $24177661775\,{x6}^{32} - 61749727185325\,{x6}^{26} - 9664082618092450\,{x6}^{20} - 12152237485813570$ $9672870290826025\,{x6}^{8} - 68544102808525\,{x6}^{2}]$

on trouve de même $V_2 = {x6}^{6} + 1$ et $V_3 = 1$ et les ensembles triangulaires

$T_2 = [\frac{V_1}{V_2}, \frac{g_2}{V_2}]$ que l'on peut simplifier en $T_2 = [\frac{V_1}{V_2}, Reduction(\frac{g_2}{V_2}, \frac{V_1}{V_2})] = [{x6}^{6} - 1, {x5}^{3} + 3\,{x5}^{2}x6 - 3\,x5\,{x6}^{2} - {x6}^{3}]$ et $T_3 = [\frac{g_0}{V_1}, \frac{g_1}{V_1}]$ que l'on simplifie aussi en $T_3 = [\frac{V_2}{V_3}, Reduction(\frac{g_3}{V_3}, \frac{V_2}{V_3})] = [{x6}^{6} + 1, {x5}^{6} - 3\,x6\,{x5}^{5} + 3\,{x6}^{2}{x5}^{4} - 4\,{x6}^{3}{x5}^{3} + 3\,{x5}^{2}{x6}^{4} - 3\,x5\,{x6}^{5} - 1]$

On remarque les systèmes triangulaires T_2 et T_3 sont condérablement simplifiés par rapport à la base de Gröbner initiale.

20.3 Théorème de Structure

Dans le cas de deux variables $k[y,x]$ on a des théorèmes précis.

Theorem 20.3.1. *Si G est une base de Gröbner d'un idéal $Id(F)$ de $k[y,x]$ alors*

$$|G| \leq mindeg(F) + 1$$

Si G est une base de Gröbner pour un ordre du degré d'un idéal $Id(F)$ de $k[y,x]$ et D degré maximal des polynômes en cours de calcul:

$$D \leq 2\, maxdeg(F) - 1$$

Exemple: $[y,x]$
Le théorème suivant est de D. Lazard:

Theorem 20.3.2. *$G = [f_0, \ldots, f_k]$ une base de Gröbner minimale pour l'ordre lexicographique $x > y$ alors*

$$f_i = H_i G_{i+1} \cdots G_{k+1} P \quad pour \quad i = 0, \ldots, k$$

avec

$G_i \in k[x] (i = 1, \ldots, k+1)$
$H_0 = 1$
$P = primpart(f_0, y)$
$G_{i+1} \cdots G_{k+1} = content(f_i, y)$
H_i *unitaire en* y *de degré* d_i
$d_0 < d_1 < \cdots < d_k$
$H_{i+1} \in Id(H_i, H_{i-1}G_i, \ldots, H_1 G_2 \cdots G_i, H_0 G_1 G_2 \cdots G_i)$

Réciproquement si G vérifie les conditions alors c'est une base de Gröbner minimale pour l'ordre lexicographique $x > y$.

Lemma 20.3.1. *On peut supposer que $P = G_{k+1} = 1$.*

Proof. Soit $d = \text{pgcd}(f_0, \ldots, f_k)$ alors $G_{k+1} = \text{content}(d, y)$ et $P = \frac{d}{G_{k+1}}$. On pose $g_i = \frac{f_i}{d}$. Alors $(g_0, \ldots, g_k)$ base de Gröbner minimale ssi $(f_0, \ldots, f_k)$ est une base de Gröbner minimale.

Dans la suite on suppose que $P = G_{k+1} = 1$.

Lemma 20.3.2. $d_i < d_{i+1}$

Proof. Alors on a nécessairement (car les f_i sont triés) $d_i = d_{i+1}$. Par suite f_i est top réductible par f_{i+1} ou l'inverse.

Lemma 20.3.3. *Soit* $g_i = coeff(f_i, y^{d_i})$. *Alors* g_{i+1} *divise* g_i.

Proof. ..

Exemple: G est la base de Gröbner lexico de cyclic 8 $\cap K[x_7, x_8]$.
$G = [f_0, \ldots, f_5]$

```
P:=Primpart(f0,x7) mod m0:
```

$$\left({x_8}^{28} + 4079\,{x_8}^{20} + {x_8}^{12}\right){x_7}^{28} + \cdots$$

```
G123456:=sort(Quo(f0,P,x7) mod m0);
```

$$x_8^{368} + \cdots$$

```
H:=Quo(f1,P,x7) mod m0;

H1:=sort(collect(expand(Primpart(H,x7) mod m0),x7),x7);
```

$$x_7 + 1498x_8^{73} + \cdots$$

```
G23456:=sort(Quo(H,H1,x7) mod m0):
G1:=sort(Normal(G123456/G23456) mod m0);
```

$$x_8^{80} + \cdots$$

```
H:=Quo(f2,P,x7) mod m0;
H2:=sort(collect(expand(Primpart(H,x7) mod m0),x7),x7);
```

$$x_7^2 + (x_8^{257} + \cdots)x_7 + (x_8^{258} + \cdots)$$

```
G3456:=sort(Quo(H,H2,x7) mod m0):
G2:=sort(Normal(G23456/G3456) mod m0);
```

$$x_8^{224} + \cdots$$

```
H:=Quo(f3,P,x7) mod m0;

H3:=sort(collect(expand(Primpart(H,x7) mod m0),x7),x7);
```

$x_7^3 + (x_8^{217} + \cdots)x_7^2 + (x_8^{218} + \cdots) + x_7 + (x_8^{219} + \cdots)$

```
G456:=sort(Quo(H,H3,x7) mod m0):
G3:=sort(Normal(G3456/G456) mod m0);
```

$x_8^{48} + \cdots$

```
H:=Quo(f4,P,x_7) mod m0;

H4:=sort(collect(ex_pand(Primpart(H,x_7) mod m0),x_7),x_7);
```

$x_7^6 + (\cdots)x_7^5 + \cdots$

```
G56:=sort(Quo(H,H4,x7) mod m0):
G4:=sort(Normal(G456/G56) mod m0);
```

$x_8^8 - 1$

```
H:=Quo(f5,P,x7) mod m0;
H5:=sort(collect(expand(Primpart(H,x7) mod m0),x7),x7);
```

$x_7^{16} + \cdots$

```
G6:=sort(Quo(H,H5,x7) mod m0);
G5:=sort(Normal(G56/G6) mod m0);
```

$G_5 = x_8^8 + 1$

$G_6 = 1$

20.4 Thèorème de Gianni Kalkebrenner

Pour continuer à décomposer la base lexicographique pour les variables x_i avec $i < n - 1$ il nous faut une généralisation du théorème 20.2.1:

Definition 20.4.1. *Si p est un polynôme, et $x_i = mainVar(p)$ sa variable principale, on note $\mathcal{LC}(p)$ le coefficient de p en tant que polynôme en x_i.*

Theorem 20.4.1. *Soit f un morphisme d'anneau de $k[x_1, \ldots, x_n]$ dans un corps K (quelconque) tel que $f(P_{i,j}) = 0$ pour $m \leq i \leq n, 1 \leq j \leq k_i$. Alors le premier élément de G (dans l'ordre indiqué dans 20.1) dont l'image par f est non nul est P_{m-1,j_0} tel que $f(\mathcal{LC}(P_{m-1,j})) = 0$ pour $j < j_0$ et $f(\mathcal{LC}(P_{m-1,j_0})) \neq 0$. De plus, $f(P_{m-1,j_0})$ est le pgcd de tous les $fP_{m-1,j}$ pour $k_{m-1} \geq j > j_0$.*

20.5 Algorithme Lextriangular

L'algorithme suivant de D. Lazard est une implantation possible du théorème de Gianni Kalkebrenner:

Nous donnons maintenant une version plus explicite de cet algorithme.

Lextriangular
Input$G$ base lexicographique $x_1 > \cdots > x_n$
Output: liste d'ensembles triangulaires.
$P = G \cap \mathbf{Q}[x_n]$
$\mathcal{L} = \emptyset$
$todo = [[x_{n-1}, [], [P]]]$
while $todo \neq \emptyset$ **do**
  $[x_i, l, T] = first(todo)$
  $todo = rest(todo)$
  if $l = \emptyset$ **then**
    $l = sort([g \in G$ tel que $\mathrm{mainVar}(g) = x_i])$
  Donc $l = [g_1, \ldots, g_r]$ avec $\mathrm{LT}(g_1) < \cdots < \mathrm{LT}(g_r)$
  $j = min\{j \in \{1, \ldots, r\}$ tel que $\varphi(\mathrm{LeadingCoeff}(g_j, x_i), T) \neq 0\}$
  $c = \varphi(\mathrm{LeadingCoeff}(g_j, x_i), T)$ (ainsi $c \neq 0$)
  $T' = T \backslash [T[1]]$
  $(u, v) = \varphi_1(T[1], [c] \cup T')$
  Donc $T[1] = u + v\,c + h$ avec $h \in Id(T')$
  if $u \neq 0$ **then**
    if $i > 1$ **then**
      $todo = todo \cup [x_{i-1}, [], \varphi(v\,g_j, T) \cup T]$
    else
      $\mathcal{L} = \mathcal{L} \cup \{\varphi(v\,g_j, T) \cup T\}$
  else
    $todo = todo \cup [x_i, [g_{j+1}, \ldots, g_r], [c] \cup T'] \cup [x_i, [g_j, \ldots, g_r], [v] \cup T']$
return $\mathcal{L}$

Exemple: la base pour l'ordre lexicographique $x_1 > \cdots > x_5$ de Cyclic 5 est:

$$x_1 + x_2 + x_3 + x_4 + x_5 = 0$$

$$275\,{x_2}^2 + 825\,x_5\,x_2 + 550\,x_5\,{x_4}^6 + 1650\,{x_5}^2{x_4}^5 + 275\,{x_5}^3{x_4}^4 - 550\,{x_5}^4{x_4}^3 + 275\,{x_4}^2 + \left(-566\,{x_5}^{11} - 69003\,{x_5}^6 + 69019\,x_5\right)x_4 + 179073\,{x_5}^2 - 1467\,{x_5}^{12} - 178981\,{x_5}^7 = 0$$

$$(275\,x_3 - 275\,x_5)\,x_2 + 275\,{x_3}^2 + 550\,x_5\,x_3 - 330\,x_5\,{x_4}^6 - 1045\,{x_5}^2{x_4}^5 - 275\,{x_5}^3{x_4}^4 + 275\,{x_5}^4{x_4}^3 - 550\,{x_4}^2 + \left(334\,{x_5}^{11} + 40722\,{x_5}^6 - 40726\,x_5\right)x_4 + 105776\,{x_5}^7 - 105873\,{x_5}^2 + 867\,{x_5}^{12} = 0$$

$$(275\,x_4 - 275\,x_5)\,x_2 - 110\,x_5\,{x_4}^6 - 440\,{x_5}^2{x_4}^5 - 275\,{x_5}^3{x_4}^4 + 275\,{x_5}^4{x_4}^3 + = 0$$

$$\left(124\,{x_5}^{11} + 15092\,{x_5}^6 - 15106\,x_5\right)x_4 + 42218\,{x_5}^7 + 346\,{x_5}^{12} - 42124\,{x_5}^2 = 0$$

$$\left(55\,{x_5}^5 - 55\right)x_2 - 144\,x_5 + {x_5}^{11} + 143\,{x_5}^6 = 0$$

$$275\,{x_3}^3 + 550\,x_5\,{x_3}^2 - 550\,{x_5}^2x_3 + 275\,{x_5}^2{x_4}^6 + 550\,{x_5}^3{x_4}^5 - 550\,{x_5}^4{x_4}^4 + 550\,x_5\,{x_4}^2 + \left(-232\,{x_5}^{12} - 28336\,{x_5}^7 + 28018\,{x_5}^2\right)x_4 + 69307\,{x_5}^3 - 69289\,{x_5}^8 - 568\,{x_5}^{13} = 0$$

$$(275\,x_4 - 275\,x_5)\,x_3 + 440\,x_5\,{x_4}^6 + 1210\,{x_5}^2{x_4}^5 - 275\;{x_5}^4{x_4}^3 + 275\,{x_4}^2 + \left(-442\,{x_5}^{11} - 53911\,{x_5}^6 + 53913\,x_5\right)x_4 - 136763\,{x_5}^7 - 1121\,{x_5}^{12} + 136674\,{x_5}^2 \;=\; 0$$

$$\left(55\,{x_5}^5 - 55\right)x_3 - 144\,x_5 + {x_5}^{11} + 143\,{x_5}^6 = 0$$

$$55\,{x_4}^7 + 165\,x_5\,{x_4}^6 + 55\,{x_5}^2{x_4}^5 - 55\,{x_4}^2 + \left(-398\,{x_5}^{11} - 48554\,{x_5}^6 + 48787\,x_5\right)x_4 - 127116\,{x_5}^7 + 128103\,{x_5}^2 - 1042\,{x_5}^{12} = 0$$

$$\left({x_5}^5 - 1\right)\left(55\,{x_4}^2 - x_5\,\left(2\,{x_5}^5 + 233\right)x_4 - {x_5}^2\left(8\,{x_5}^5 + 987\right)\right) = 0$$

$${x_5}^{15} + 122\,{x_5}^{10} - 122\,{x_5}^5 - 1 = 0$$

Voici une trace de l'execution de l'algorithme Lextriangular:

P= $[x_5^{15} + \ldots]$
todo = $< x_4, [], \mathrm{T}= [x_5^{15} + \ldots] >$
first(todo)= $< x_4, [], \mathrm{T}= [x_5^{15} + \ldots] >$
l = $[55\,x_4^2(x_5^5 - 1) + \ldots, x_4^7 + \ldots]$
Premier LeadingCoeff non nul: $x_5^5 - 1$
$T[1] = {x_5}^{15} + 122\,{x_5}^{10} - 122\,{x_5}^5 - 1 = 0 + (x_5^{10} + \ldots)(x_5^5 - 1) + \ldots$
$u = 0$
todo =todo $\cup < x_4, [x_4^7 + \ldots], \mathrm{T}= [x_5^5 - 1] > \cup < x_4, [55(x_5^5 - 1)x_4^2 + \ldots, x_4^7 + \ldots], \mathrm{T}= [x_5^{10} + \ldots] >$
first(todo)= $< x_4, [x_4^7 + \ldots], \mathrm{T}= [x_5^5 + \ldots] >$
Premier LeadingCoeff non nul: 1
todo = todo $\cup < x_3, [], \mathrm{T}= [x_4^7 + \ldots, x_5^5 - 1] >$
first(todo)= $< x_3, [], \mathrm{T}= [x_4^7 + \ldots, x_5^5 + \ldots] >$
l = $[x_3\,x_4 + \ldots, x_3^3 + \ldots]$
Premier LeadingCoeff non nul: $x_4 + \ldots$
$x_4^7 + \ldots = 0 + (x_4^6 + \ldots)(x_4 + \ldots) + \ldots$
$u = 0$
todo = todo $\cup < x_3, [x_3^3 + \ldots], \mathrm{T}= [x_4 + \ldots, x_5^5 + \ldots] > \cup < x_3, [x_3\,x_4 + \ldots, x_3^3 + \ldots], \mathrm{T}= [x_4^6 + \ldots, x_5^5 + \ldots] > \ldots$
first(todo)= $< x_3, [x_3^3 + \ldots], \mathrm{T}= [x_4 + \ldots, x_5^5 + \ldots] >$
l = $[x_3^3 + \ldots]$
Premier LeadingCoeff non nul: 1
todo =todo $\cup < x_2, [], \mathrm{T}= [x_3^3 + \ldots, x_4 + \ldots, x_5^5 + \ldots] >$
first(todo)= $< x_2, [], \mathrm{T}= [x_3^3 + \ldots,$

$x_4 + \ldots, x_5^5 + \ldots] >$
l = $[x_2\, x_3 + \ldots, x_2^2 + \ldots]$
Premier LeadingCoeff non nul: $x_3 + \ldots$
$x_3^3 + \ldots = 0 + (x_3^2 + \ldots)(x_3 + \ldots) + \ldots$
$u = 0$
todo = todo $\cup <$ x_2, $[x_2^2 + \ldots]$, T= $[x_3 + \ldots, x_4 + \ldots, x_5^5 + \ldots] > \cup <$ x_2, $[x_2\, x_3 + \ldots, x_2^2 + \ldots]$, T= $[x_3^2 + \ldots, x_4 + \ldots, x_5^5 + \ldots] >$
first(todo)= $<$ x_2, $[x_2^2 + \ldots]$, T= $[x_3 + \ldots, x_4 + \ldots, x_5^5 + \ldots] >$
l = $[x_2^2 + \ldots]$
Premier LeadingCoeff non nul: 1
todo = *todo*$\cup <$ x_1, [], T= $[x_2^2 + \ldots, x_3 + \ldots, x_4 + \ldots, x_5^5 + \ldots] >$
first(todo)= $<$ x_1, [], T= $[x_2^2 + \ldots, x_3 + \ldots, x_4 + \ldots, x_5^5 + \ldots] >$
l = $[x_1 + \ldots]$
Premier LeadingCoeff non nul: 1
$\mathcal{L} = \mathcal{L} \cup [x_1 + \ldots, x_2^2 + \ldots, x_3 + \ldots, x_4 + \ldots, x_5^5 + \ldots]$
first(todo)= $<$ x_2, $[x_2\, x_3 + \ldots, x_2^2 + \ldots]$, T= $[x_3^2 + \ldots, x_4 + \ldots, x_5^5 + \ldots] >$
l =$[x_2\, x_3 + \ldots, x_2^2 + \ldots]$
Premier LeadingCoeff non nul: $x_3 + \ldots$
$x_3^2 + \ldots = x_5^2 + \ldots + (x_3 + \ldots)(x_3 + \ldots) + \ldots$
$x_5^5 + \ldots = 1 + (x_5^3 + \ldots)(x_5^2 + \ldots) + \ldots$
$(x_5^3 + \ldots)(x_3^2 + \ldots) = 1 + (x_3\, x_5^3 + \ldots)(x_3 + \ldots) + \ldots$
$u \neq 0$
$u^{-1} = v = x_3\, x_5^3 + \ldots$
$\varphi(g_j, T) = x_2\, x_3 + \ldots$
$\varphi(vg_j, T) = x_2 + \ldots$
todo = todo $\cup <$ x_1, [], T= $[x_2 + \ldots, x_3^2 + \ldots, x_4 + \ldots, x_5^5 + \ldots] >$
first(todo) = $<$ x_1, [], T= $[x_2 + \ldots, x_3^2 + \ldots, x_4 + \ldots, x_5^5 + \ldots] >$
l = $[x_1 + \ldots]$
Premier LeadingCoeff non nul: 1
$\mathcal{L} = \mathcal{L} \cup [x_1 + \ldots, x_2 + \ldots, x_3^2 + \ldots, x_4 + \ldots, x_5^5 + \ldots]$
first(todo)= $<$ x_3, $[x_3\, x_4 + \ldots, x_3^3 + \ldots]$, T= $[x_4^6 + \ldots, x_5^5 + \ldots] >$
l = $[x_3\, x_4 + \ldots, x_3^3 + \ldots]$
Premier LeadingCoeff non nul: $x_4 + \ldots$
$x_4^6 + \ldots = x_5 + \ldots + (x_4^5 + \ldots)(x_4 + \ldots) + \ldots$
$x_5^5 + \ldots = 1 + (x_5^4 + \ldots)(x_5 + \ldots) + \ldots$
$(x_5^4 + \ldots)(x_4^6 + \ldots) = 1 + (x_4^5\, x_5^4 + \ldots)(x_4 + \ldots) + \ldots$
$u \neq 0$
$v = u^{-1} = x_4^5\, x_5^4 + \ldots$
$\varphi(g_j, T) = x_3\, x_4 + \ldots$
$\varphi(vg_j, T) = x_3 + \ldots$
todo = todo $\cup <$ x_2, [], T= $[x_3 + \ldots, x_4^6 + \ldots, x_5^5 + \ldots] >$
first(todo)= $<$ x_2, [], T= $[x_3 + \ldots, x_4^6 + \ldots, x_5^5 + \ldots] >$
l = $[x_2\, x_4 + \ldots + \ldots, x_2\, x_3 + \ldots, x_2^2 + \ldots]$
Premier LeadingCoeff non nul: $x_4 + \ldots$
$x_4^6 + \ldots = x_5 + \ldots + (x_4^5 + \ldots)(x_4 + \ldots) + \ldots$
$x_5^5 + \ldots = 1 + (x_5^4 + \ldots)(x_5 + \ldots) + \ldots$
$(x_5^4 + \ldots)(x_4^6 + \ldots) = 1 + (x_4^5\, x_5^4 + \ldots)(x_4 + \ldots) + \ldots$
$u \neq 0$
$v = u^{-1} = x_4^5\, x_5^4 + \ldots$
$\varphi(g_j, T) = x_2\, x_4 + \ldots$
$\varphi(vg_j, T) = x_2 + \ldots$
todo = todo $\cup <$ x_1, [], T= $[x_2 + \ldots + \ldots, x_3 + \ldots, x_4^6 + \ldots, x_5^5 + \ldots] >$
first(todo)= $<$ x_1, [], T= $[x_2 + \ldots + \ldots, x_3 + \ldots, x_4^6 + \ldots, x_5^5 + \ldots] >$
l = $[x_1 + \ldots]$
Premier LeadingCoeff non nul: 1
$\mathcal{L} = \mathcal{L} \cup [x_1 + \ldots, x_2 + \ldots, x_3 +$

$\ldots, x_4^6 + \ldots, x_5^5 + \ldots]$
first(todo)= $< x_4, [x_4^2\, x_5^5 + \ldots + \ldots,$ $x_4^7 + \ldots], \mathrm{T}= [x_5^{10} + \ldots] >$
l = $[x_4^2\, x_5^5 + \ldots, x_4^7 + \ldots]$
Premier LeadingCoeff non nul: $x_5^5 + \ldots$
$x_5^{10} + \ldots = 1 + (x_5^5 + \ldots)(x_5^5 + \ldots) + \ldots$
$u \neq 0$
$v = u^{-1} = x_5^5 + \ldots$
$\varphi(g_j, T) = x_4^2\, x_5^5 + \ldots$
$\varphi(vg_j, T) = x_4^2 + \ldots$
todo = todo $\cup <$ x_3, [], T= $[x_4^2 + \ldots, x_5^{10} + \ldots] >$
first(todo)= $< x_3$, [], T= $[x_4^2 + \ldots, x_5^{10} + \ldots] >$
l = $[x_3\, x_5^5 + \ldots, x_3\, x_4 + \ldots, x_3^3 + \ldots]$
Premier LeadingCoeff non nul: $x_5^5 + \ldots$
$x_5^{10} + \ldots = 1 + (x_5^5 + \ldots)(x_5^5 + \ldots) + \ldots$
$u \neq 0$
$v = u^{-1} = x_5^5 + \ldots$
$\varphi(g_j, T) = x_3\, x_5^5 + \ldots$
$\varphi(vg_j, T) = x_3 + \ldots$
todo = todo $\cup <$ x_2, [], T= $[x_3 + \ldots, x_4^2 + \ldots, x_5^{10} + \ldots] >$
first(todo)= $< x_2$, [], T= $[x_3 + \ldots, x_4^2 + \ldots, x_5^{10} + \ldots] >$
Search =$[x_2\, x_5^5 + \ldots, x_2\, x_4 + \ldots, x_2\, x_3 + \ldots, x_2^2 + \ldots]$
Premier LeadingCoeff non nul: $x_5^5 + \ldots$
$x_5^{10} + \ldots = 1 + (x_5^5 + \ldots)(x_5^5 + \ldots) + \ldots$
$u \neq 0$
$v = u^{-1} = x_5^5 + \ldots$
$\varphi(g_j, T) = x_2\, x_5^5 + \ldots$
$\varphi(vg_j, T) = x_2 + \ldots$
todo = todo $\cup <$ x_1, [], T= $[x_2 + \ldots, x_3 + \ldots, x_4^2 + \ldots, x_5^{10} + \ldots] >$
first(todo)= $< x_1$, [], T= $[x_2 + \ldots, x_3 + \ldots, x_4^2 + \ldots, x_5^{10} + \ldots] >$
l = $[x_1 + \ldots + \ldots]$
Premier LeadingCoeff non nul: 1
$\mathcal{L} = \mathcal{L} \cup [x_1 + \ldots, x_2 + \ldots, x_3 + \ldots + \ldots, x_4^2 + \ldots, x_5^{10} + \ldots]$
Fin decomposition

20.6 Formule de Breguet

Le programme suivant permet de définir la formule de Breguet ainsi que les paramètres qui ont illustré l'introduction:

20.6.1 #ATMOSPHERE STANDARD

```
#altitude (m)
    H:=13750;

    #pression statique(Pa)
    p:=5479.4;

    #Masse Volumique ro(kg.m-3)
    ro:=2.0625*exp(-1.5769*(10^(-4))*13750);
    #Temperature absolue (K)
```

```
T:=216.65;

#Constante des gaz parfaits (m2.K-1.S-2)
R:=287;
#Acceleration gravitationnelle (m.s-2)
g:=9.806;
```

20.6.2 #AERODYNAMIQUE

```
#Coefficient de trainee totale a Cz nul
Cx0:=0.018;
#Coefficient de portance
Cz:=1.25;
```

20.6.3 #MASSE

```
#masse initiale (kg)
minit:=9200;
#coef masse a vide
kv:=mv/minit

#coef charge utile
#kcu:=mcu/minit
kcu:=0.0247;

# coef masse du fuselage
#kf:=mf/minit
kf:=0.147;

#coef masse du train d'atterissage
#kt:=mt/minit
kt:=0.032;

#coef de la masse des empennages
#ke:=me/minit
ke:=0.05;

#rapport de la masse moteur sur sa puissance(kg.W-1)
mumot:=0.00235;

#masse moteur
#mmot:=mumot*Pmot
#masse du caisson de la voilure par unite de surface alaire (kg.m-2)
ms:=7.67;
```

```
#coef de la masse voilure Kr
Kr:=8.74*(10^(-5));

#rendement nuprop
nuprop:=0.85;

#Consommation specifique
Csp:=(2.2498*10^(-4));
```

20.6.4 Formule

```
psi:=(S^(1/2))/(U1+U0/L):
    kv:=U4*S^(1/2)*L^(3/2)+U3*S^(1/2)+U6*S*L^(3/2)+U5*S+U7*S*L^(1/2)+U8
1)):
    den:=kv+U2:
    phi:=(den)^(-1/2)-1:
    E:=phi*psi:
    ll:=expand(map(numer,[diff(E,L),diff(E,S)])):
    l0:=subs(csgn(s)=1,csgn(l)=1,simplify(subs(S=s^2,L=l^2,ll))):
    den1:=subs(csgn(s)=1,csgn(l)=1,simplify(subs(S=s^2,L=l^2,den))):
    l1:=subs((A^2)^(1/2)=A,subs(den1=A^2,l0)):
    l2:=factor([op(l1),A^2*l^2-den1*l^2]):
    l3:=[l2[1]/(s*l^3),l2[2]/(l^2),l2[3]]:
    U1:=Cx0*Csp*(g^(3/2))*(minit^(1/2))/(nuprop*((2*ro)^(1/2))*(Cz^(3/2))):
    U0:=Cz^2*Csp*(g^(3/2))*(minit^(1/2))/(Pi*nuprop*((2*ro)^(1/2))*(Cz^(3/2))):
    U4:=Kr*(kf+ke+kt+kcu):
    U3:=(sqrt(2)*mumot*(g^(3/2)))/(nuprop*(ro^(1/2))*(Cz^(3/2))*(minit^(1/2))):
    U6:=U3*Kr*Cx0:
    U5:=ms/minit:
    U8:=U3*Cz^2/Pi:
    U7:=U8*Kr:
    U3:=U3*Cx0:
    U2:=kf+ke+kt+kcu:
```

21. Résolution réelle

Ce cours chapitre présente le principaux outils pour calculer les racines réelles d'un système algébrique dans le cas d'un système ayant un nombre fini de solutions (voir la thèse (Rouillier F., 1996) et HDR de Fabrice Rouillier).

21.1 La Représentation Univariée Rationnellle

The Rational Univariate Representation (Rouillier, 1999) is, with the end-user point of view, the simplest way for representing symbolically the roots of a zero-dimensional system without loosing information (multiplicities or real roots) since one can get all the information on the roots of the system by solving univariate polynomials.

Given a zero-dimensional system $I = \mathrm{Id}(p_1, \ldots, p_s)$ where the $p_i \in \mathbb{Q}[x_1, \ldots, x_n]$, a *Rational Univariate Representation (RUR)* of $V_{\mathbb{R}}(I)$ has the following shape :

$$\begin{cases} f_t(T) = 0 \\ x_1 = \frac{g_{t,x_1}(T)}{g_{t,1}(T)} \\ \cdots \\ x_n = \frac{g_{t,x_n}(T)}{g_{t,1}(T)} \end{cases}$$

où $f_t, g_{t,1}, g_{t,x_1}, \ldots, g_{t,x_n} \in \mathbb{Q}[T]$ (and T is a new variable). It is uniquely defined w.r.t. a given polynomial t which separates $V(I)$ (injective on $V(I)$), the polynomial f_t being necessarily the characteristic polynomial of m_t (see above section) in $\mathbb{Q}[x_1, \ldots, x_n]/I$ (Rouillier, 1999). The RUR defines a bijection between the roots of $\mathcal{F}$ and those of f_t preserving the multiplicities and the real roots :

$$\begin{array}{ccc} V_{\mathbb{R}}(\mathcal{S}) & \approx & V_{\mathbb{R}}(f_t) \\ \alpha = (\alpha_1, \ldots, \alpha_n) & \rightarrow & t(\alpha) \\ (x_1(\alpha) = \frac{g_{t,x_1}(t(\alpha))}{g_{t,1}(t(\alpha))}, \ldots, x_n(\alpha) = \frac{g_{t,x_n}(t(\alpha))}{g_{t,1}(t(\alpha))}) & \leftarrow & t(\alpha) \end{array}$$

In (Rouillier, 1999), a strategy is proposed for computing a RUR for any system (a RUR-Candidate and a separating element), but there are special

cases where it can be computed differently. When x_1 is separating $V_{\mathbb{R}}(I)$ and the system is said to be in *shape position* (see equations 2.5 in section 2.9.3).

As shown in (Rouillier, 1999), if the system is in shape position, $g_{x_1,1} = f'_{x_1}$ and we have $f_{x_1} = f$ and $f_i(x_1) = g_{x_1,x_i}(x_1)g_{x_1,1}(x_1) \bmod f$. Thus the RUR associated with x_1 and the lexicographic Gröbner basis are equivalent up to the inversion of $g_{x_1,1} = f'_{x_1}$ modulo f_{x_1}. In the rest of this paper we call this object a RR-Form of the corresponding lexicographic Gröbner basis. The RUR is well known to be smaller than the lexicographic Gröbner basis in general and thus will be our priviligied object. Note that it is easy to check that a system is in shape position once knowing a RUR-Candidate (and so to check that x_1 separates $V_{\mathbb{R}}(I)$): it is necessary and sufficient that f_{x_1} is square-free.

These results have many practical drawbacks since, the systems which are often in shape position. We thus can multiply the strategies for computing a RUR : one can compute a "modified" lexicographic Gröbner directly using (Faugère J.C., 1999) for example or by change of ordering like in (Faugère *et al.* , 1993) or a RUR using the algorithm from (Rouillier, 1999).

21.2 Isolation des racines réelles

Computing a RUR reduces the resolution of a zero-dimensional system to solving one polynomial in one variable (f_t) and to evaluating n rational fractions

$$\left(\frac{g_{t,x_i}(T)}{g_{t,1}(T)}, i = 1 \dots n\right)$$

at its roots (note that if one simply want to compute the number of real roots of the system there is no need to consider the rational coordinates). Our goal is to compute all the real roots of the system (and only the real roots), providing a numerical approximation with an arbitrary precision (set by the user) of the coordinates.

In practice, the computation of the RUR is not the end point of the work : approximating the roots of f_t is not sufficient to provide accurate numerical approximations of the roots of the initial system and, moreover, not sufficient to guarantee the sign of the coordinates. Also a naive algorithm which would consist in "plugging" numerical approximations of the roots into the $f_1, \dots, f_s$ will not give, in most cases, any suitable information. If one is only interested in the signs of the f_i one could imagine computing

$$f_i\left(\frac{g_{t,x_1(T)}}{g_{t,A}(T)}, \dots, \frac{g_{t,x_n(T)}}{g_{t,1}(T)}\right)$$

and studying the values of these polynomials at the roots of I. Again, this would lead to very hard computations since such a plug induces multiplying large polynomials modulo f_t.

The isolation of the real roots of f_t can be done using the algorithm proposed in (Rouillier & Zimmermann, 2003) : the output will be a list l_{f_t} of intervals with rational bounds such that for each real root α of f_t, there exists a unique interval in l_{f_t} which contains α. The second step consists in refining each interval in order to ensure that it does not contain any real root of $g_{t,1}$. Since f_t and $g_{t,1}$ are co-prime this computation is easy and we then can ensure that the rational functions can be evaluated using interval arithmetics without any cancellation of the denominator. This last evaluation is performed using multi-precision arithmetics (MPFI package). En pratique, finir ici ! the precision needed for the computations is poor and, moreover, the rational functions defined by the RUR are stable under numerical evaluation, even if their coefficients are huge (rational numbers), and thus this part of the computation is still efficient. For increasing the precision of the result, it is only necessary to decrease the length of the intervals in l_{f_t} which can easily be done by bisection. Note that is is quite simple to certify the sign of the coordinates : one simply have to compute some gcds and split, when necessary the RUR.

22. Algèbre linéaire

Résoudre un système d'équations linéaires

$$Ax = b$$

est aussi résoudre un système d'équations algébriques dont toutes les équations sont de degré un. On a vu que l'algorithme FGLM (3.1 page 52) ramenait le problème du calcul d'une base de Gröbner à un problème d'algèbre linéaire. On verra aussi que l'algorithme F4 (4 page 79) repose entièrement sur une algèbre linéaire efficace.

22.1 Méthode de Gauss

Il est commode d'ajouter la colonne b dans la matrice A. On suppose donc que $A = (a_{i,j})$ est une matrice $n \times m$ (avec $m > n$). Après la $k-$ième étape de calcul on note $A^{(k)} = (a_{i,j}^{(k)})$ la matrice transformée. On suppose que les entrées de la matrice initiale A sont dans un corps k.

Gauss
Input: A une matrice $n \times m$
Output: une matrice triangulaire
for $k = 1, \ldots, n$ **do**
  for $j = k, \ldots, m$ **do**
    if $a_{k,k} = 0$ **then**
      inverser les lignes pour que $a_{k,k} \neq 0$
    $a_{k,j} := \frac{a_{i,j}}{a_{k,k}}$
  for $i = (k+1), \ldots, n$ **do**
    pivot $:= -a_{i,k}$
    for $j = (k+1), \ldots, m$ **do**
      $a_{i,j} := a_{i,j} + \text{pivot} \times a_{k,j}$
return A

Il est clair que le nombre d'opérations ($\times$ ou $+$) est:

$$\sum_{k=1}^{n}(m-k)+\sum_{k=1}^{n-1}\sum_{i=k+1}^{n}(m-k)=\frac{1}{6}\,n\,(n+1)\,(3\,m-n-2)$$

Lorsque $m=n+1$ on trouve:

$$\frac{1}{6}\,n\,(n+1)\,(2\,n+1)\approx\frac{n^3}{3}$$

Lorsque $m=2n$ (inversion de matrice) on trouve:

$$\frac{1}{6}\,n\,(n+1)\,(5\,n-2)\approx\frac{5}{6}n^3$$

Le résultat de l'algorithme de Gauss est de fournir une matrice triangulaire

$$A^{(n-1)}=\begin{pmatrix}1 & a_{1,2}^{(n-1)} & \dots & a_{1,n}^{(n-1)} & \dots & a_{1,m}^{(n-1)}\\ 0 & 1 & \dots & a_{2,n}^{(n-1)} & \dots & a_{2,m}^{(n-1)}\\ \vdots & \ddots & \dots & \vdots & \dots & \vdots\\ 0 & 0 & \dots & 1 & \dots & a_{n,m}^{(n-1)}\end{pmatrix}$$

Il reste à obtenir une matrice diagonale:

Backward substitution
Input: A une matrice $n\times m$triangulaire
Output: une matrice diagonale
for $k=n,\dots,2$ **do**
  for $i=1,\dots,(k-1)$ **do**
    pivot $:=-a_{i,k}$
    for $j=(n+1),\dots,m$ **do**
      $a_{i,j}:=a_{i,j}+\text{pivot}\times a_{k,j}$
return A

dont le coût est

$$\sum_{k=2}^{n}\sum_{i=1}^{k-1}(m-n)=\frac{n\,(n-1)\,(n-m)}{2}$$

lorsque $m=n+1$ le coût de cette diagonalisation est négligeable:

$$\frac{n\,(n-1)}{2} \approx \frac{n^2}{2}$$

mais pas lorsque $m = 2n$:

$$\frac{n^2\,(n-1)}{2} \approx \frac{n^3}{2}$$

$$\begin{pmatrix} 1 & 0 & \dots & 0 & b_{1,n+1} & \dots & b_{1,m} \\ 0 & 1 & \dots & 0 & b_{2,n+1} & \dots & b_{2,m} \\ \vdots & \ddots & \dots & \vdots & \dots & \dots & \vdots \\ 0 & 0 & \dots & 1 & b_{n,n+1} & \dots & b_{n,m} \end{pmatrix}$$

et le vecteur $[b_{1,n+1}, b_{2,n+1}, \dots, b_{n,n+1}]$ est solution.

22.2 Gauss sur les entiers

Lorsque le corps k est le corps $\mathbf{Q}$ des rationnels le coût de l'algorithme devient prohibitif (croissance des coefficients et normalisation des fractions). Une méthode naive consiste à multiplier toutes les lignes de la matrice A par le contenu et donc d'obtenir une matrice A à coefficients dans $\mathbf{Z}$ et d'appliquer la version suivante de l'algorithme de Gauss:

Gauss **Z**
Input: A une matrice $n \times m$ coefficients entiers
Output: une matrice triangulaire
for $k = 1, \dots, (n-1)$ **do**
  for $i = (k+1), \dots, n$ **do**
    for $j = (k+1), \dots, m$ **do**
      $a_{i,j} := a_{k,k} \times a_{i,j} - a_{i,k} \times a_{k,j}$
return A

Le résultat retourné est une matrice à coefficients entiers.

22.3 Algorithme de Bareiss

L'algorithme précédant est totalement inefficace à cause de la croissance des coefficients.

Gauss Bareiss
Input: A une matrice $n \times m$ coefficients entiers
Output: une matrice triangulaire
Den := 1
for $k = 1, \ldots, (n-1)$ **do**
  for $i = (k+1), \ldots, n$ **do**
    for $j = (k+1), \ldots, m$ **do**
      $a_{i,j} := \frac{a_{k,k} \times a_{i,j} - a_{i,k} \times a_{k,j}}{\text{Den}}$
  Den := $a_{k,k}$
return A

En utilisant les notations $A^{(k)}$ on a de manière équivalente:

$$a_{i,j}^{(k)} = \frac{1}{a_{k-1,k-1}^{(k-2)}} \begin{vmatrix} a_{k,k}^{(k-1)} & a_{k,j}^{(k-1)} \\ a_{i,k}^{(k-1)} & a_{i,j}^{(k-1)} \end{vmatrix}$$

L'intérêt de cette méthode est qu'on peut exprimer facilement les données intermédiaires en fonction de la matrice de départ:

$$a_{i,j}^{(k)} = \begin{vmatrix} a_{1,1} & \cdots & a_{1,k} & a_{1,j} \\ \cdots & \cdots & \cdots & \cdots \\ a_{i-1,1} & \cdots & a_{i-1,k} & a_{i-1,j} \\ a_{i,1} & \cdots & a_{i,k} & a_{i,j} \end{vmatrix}$$

Inégalité d'Hadamard:

$$|\det(A)| \leq \sqrt{\prod_j \sum_i |a_{i,j}|^2}$$

En particulier si $|a_{i,j}| \leq B$ alors

$$|\det(A)| \leq n^{\frac{n}{2}} B^n$$

on en déduit si B est une borne des coefficients de la matrice initiale A:

$$|a_{i,j}^{(k-1)}| \leq k^{\frac{k}{2}} B^k$$

on note $M(t)$ le coût de la multiplication de deux entiers $u \leq t$:

$$M(t) = \log_b(t)^\alpha \quad \alpha \leq 2$$

avec $b = 2^{32}$ la taille d'un mot machine. Le coût du calcul d'un quotient est $\frac{7}{2}M(t)$. Le calcul de $a_{i,j}^{(k)}$ nécessite donc

$$\begin{array}{ll} 2M(t) + \frac{7}{2}M(t^2) & \text{avec } t = k^{\frac{k}{2}}B^k \\ = (2 + 72^{\alpha-1})log(t)^\alpha & \\ = (2 + 72^{\alpha-1})(klog(B) + \frac{k}{2}log(k))^\alpha & \end{array}$$

et il faut effectuer $(n-k)(m-k)$ opérations à l'étape k. Le coût total est donc:

$$(2 + 72^{\alpha-1}) \sum_{k=1}^{n-1} (n-k)(m-k)(klog(B) + \frac{k}{2}log(k))^\alpha$$

soit lorsque $\alpha = 2$ et $m = n + 1$:

$$\left(1/30\ (\ln(B))^2 + 1/30\ \ln(B)\ln(n) - \frac{47}{1800}\ \ln(B) + \frac{1}{120}\ (\ln(n))^2 - \frac{47}{3600}\ \ln(n) + \frac{1489}{21600}\right.$$

donc en résumé

$$1/30\ \ln(B)^2 n^5$$

lorsque $\alpha = 1$ et $m = n + 1$:

$$\left(1/12\ \ln(B) + 1/24\ \ln(n) - \frac{13}{288}\right) n^4 + O(n^3) \approx 1/12\ \ln(B) n^4$$

22.4 Méthode p-adique

On revient au problème

$$Ax = b$$

où A est une matrice $n \times n$ et b un vecteur $n \times 1$. On choisit p un nombre premier tel que $det(A) \neq 0$ modulo p. Soit $C = A^{-1}$ modulo p. On calcule alors

$$\begin{cases} b_0 = b \\ x_i = Cb_i \text{modulo } p \\ b_{i+1} = \frac{b_i - Ax_i}{p} \end{cases}$$

On pose

$$\tilde{x} = \sum_{i=0}^{m-1} x_i p^i$$

alors

$$A\tilde{x} = \sum_{i=0}^{m-1} p^i A x_i = \sum_{i=0}^{m-1} p^i (b_i - p b_{i+1}) = b - p^m b_m$$

donc $\tilde{x}$ est solution modulo p^m. On peut démontrer qu'il suffit de calculer les itérations jusqu'au rang $m = 2\log_p(\tilde{B})$ où $\tilde{B}$ est une borne de $|x|$ pour que x soit solution . En utilisant Hadamard: $m = 2n\log_p(nB)$.

On a donc à l'issue de cette méthode trouvé un entier $\tilde{x}$ vérifiant la relation modulo p^m et l'on recherche l'élément $x \in \mathbf{Q}$ tel que $x = \tilde{x} \bmod p^m$ c'est à dire un rationnel $\frac{a}{b}$ tel que $\frac{a}{b} = \tilde{x} \bmod p^m$. Pour cela on va utiliser les fractions continues.

Une fraction continue est un élément de la forme :

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \cdots}}}$$

que l'on note par commodité $[a_0, a_1, \ldots, a_n]$.

$$\begin{aligned}
&[a_0] = \tfrac{a_0}{1} \\
&[a_0, a_1, \ldots, a_n] = \left[a_0, a_1, \ldots, a_{n-1}, a_{n-1} + \tfrac{1}{a_n}\right] \\
&[a_0, a_1, \ldots, a_n] = a_0 + \tfrac{1}{[a_1, \ldots, a_n]} = [a_0, [a_1, \ldots, a_n]]
\end{aligned}$$

On définie le $k-$ème "convergent" de $[a_0, a_1, \ldots, a_n]$ comme étant $[a_0, a_1, \ldots, a_k]$.

Theorem 22.4.1.

$$\begin{array}{lll} p_0 = a_0 & p_1 = a_1 a_0 + 1 & p_k = a_k p_{k-1} + p_{k-2} \\ q_0 = 1 & q_1 = a_1 & q_k = a_k q_{k-1} + q_{k-2} \end{array}$$

alors

$$\frac{p_k}{q_k} = [a_0, a_1, \ldots, a_k]$$

Theorem 22.4.2. *Soit une fraction $\frac{a}{b}$ telle que*

(i) $a \leq M - 1$, $b \leq M - 1$
(ii) $pgcd(a, b) = 1$

(ii) $b \neq 0$
(iii) $p^m \geq 2M^2$
(iv) $\frac{a}{b} = u$ *modulo* p^m

alors il existe i *tel que* $a = a_i$ *et* $b = b_i$ *où les* a_i, b_i *sont produit par l'algorithme d'Euclide sur* $\mathbf{N}$.

Euclide généralisé sur $\mathbf{N}$

Input: $\begin{cases} m \\ 0 \geq u < m \\ B \text{ borne du théorème précédent} \end{cases}$
Output: $b/c = u$ modulo p^m
$a_{-1} := p^m$
$b_{-1} := 0$
$c_{-1} := -1$
$a_0 := u$
$b_0 := 1$
$c_0 := 0$
$i := 1$
while $a_{i-1} \neq 0$ **do**
  $q_i :=$ quotient $(\frac{a_{i-2}}{a_{i-1}})$
  $a_i := a_{i-2} - q_i a_{i-1}$
  $b_i := b_{i-2} - q_i b_{i-1}$
  $c_i := c_{i-2} - q_i c_{i-1}$
  if $a_i \leq B$ **and** $b_i \leq B$ **then return** $\frac{a_i}{b_i}$
if $B \neq 0$ **then ERROR else return** $\frac{a_i}{b_i}$

Note 22.4.1. Cet algorithme peut être généralisé à une réduction modulo un entier quelconque v.

Le même type de résultat existe pour les polynômes :

Theorem 22.4.3. *Soit une fraction* $\frac{a(x)}{b(x)}$ *telle que*

(i) $deg(a(x)) \leq M-1$, $deg(b(x)) \leq M-1$
(ii) $pgcd(a(x), b(x)) = 1$
(iii) x *ne divise pas* $b(x)$
(iv) $n \geq 2M-1$
(v) $\frac{a(x)}{b(x)} = H(x)$ *modulo* x^m

alors il existe i *tel que* $a(x) = a_i(x)$ *et* $b = b_i(x)$ *où les* a_i, b_i *sont produit par l'algorithme d'Euclide.*

Euclide généralisé

Input: $\begin{cases} m \\ H(x) \text{ polynôme de degré} < m \\ B \text{ borne sur le degré} \end{cases}$

Output: $P(x)/Q(x) = H(x)$ modulo x^m

$a_{-1} := x^m$
$b_{-1} := 0$
$c_{-1} := -1$
$a_0 := H(x)$
$b_0 := 1$
$c_0 := 0$
$i := 1$
while $a_{i-1} \neq 0$ **do**
  $q_i :=$ quotient $(\frac{a_{i-2}}{a_{i-1}})$
  $a_i := a_{i-2} - q_i a_{i-1}$
  $b_i := b_{i-2} - q_i b_{i-1}$
  $c_i := c_{i-2} - q_i c_{i-1}$
  if $deg(a_i) \leq B$ **and** $deg(b_i) \leq B$ **then return** $\frac{a_i}{b_i}$
if $B \neq 0$ **then ERROR else return** $\frac{a_i}{b_i}$

Theorem 22.4.4. *La suite* $(\frac{b_1}{c_1}, \frac{b_2}{c_2}, \ldots, \frac{b_i}{c_i})$ *est la suite des "réduites" du développement en fraction continue de* $\frac{x^m}{H(x)}$ *(resp.* $\frac{p^m}{u}$*).*

Pour la méthode p-adique on applique donc l'algorithme d'Euclide sur $\mathbf{N}$ à p^m et $(\tilde{x})_i$ et arrête l'algorithme dès que $a_i < p^{\frac{m}{2}}$.

On démontre que

$$|b_i| \leq nB \frac{1 - (1/p)^i}{1 - 1/p} \leq 2nB$$

Le coût de l'évaluation $b_{i+1} = \frac{b_i - Ax_i}{p}$ est donc $O(n^2 \log(nB))$ à l'étape i et donc en tout

$$mO(n^2 \log(nB)) = O(n^3 \log^2(nB))$$

opérations pour utiliser l'algorithme d'Euclide sur $\mathbf{N}$: $O(n^2 \log^2(nB))$ pour une composante $(\tilde{x})_i$. La complexité globale est donc

$$O(n^3 \log^2(nB))$$

22.5 Algorithme de Lanczos (*en anglais*)

This is the case of conjugate gradient method or the Lanczos algorithm (Montgomery P., 1995) since after n steps the result is exact:

We define $\langle u, v\rangle$ to be the scalar product.

Lanczos algorithm

Input: $\begin{cases} A \text{ a } n \times n \text{ matrix} \\ w \text{ a } n \times 1 \text{ vector} \end{cases}$

$w_0 := w$, $v_1 := Aw_0$, $q_1 := \frac{-1}{\langle w_0, v1\rangle}$

$w_1 := v_1 + q_1\langle v_1, v_1\rangle w_0$ and $x := \langle w_0, w_0\rangle q_1 w_0$

for $j := 2, 3, \ldots, (n+1)$ **do**

  $v_2 := Aw_1$ and $r := \langle w_1, v_2\rangle$

  if $r \neq 0$ **then**

    $q_j := \frac{-1}{\langle w_1, v_2\rangle}$, $x := x + q_j\langle w_1, w\rangle w_1$

    $w_2 := v_2 + q_j\langle v_2, v_2\rangle w_1 + q_{j-1}\langle v_2, v_1\rangle w_0$

  else

    return $-x$

  fi

  $(w_0, w_1, w_2) \leftarrow (w_1, w_2, w_0)$ and $(v_1, v_2) \leftarrow (v_2, v_1)$

Note that sometimes the algorithm may fail since we can divide by zero.

22.6 Algorithme de Wiedemann (*en anglais*)

Another well known iterative algorithm is the Wiedemann algorithm (Wiedemann D., 1986; Kaltofen E., 1991), which uses the efficient Berlekamp and Massey algorithm (L., 1969) to recover the solution. The berlekamp algorithm take a list of numbers $(s_0, \ldots, s_{2n-1})$ and returns a polynomial $f(x) = X^n + \sum_{i=0}^{n-1} f_i X^i$ such that $s_{i+n} = \sum_{i=0}^{n-1}(-f_i)s_{i+j}$ for $i \geq 0$

Wiedemann algorithm

Input: $\begin{cases} A \text{ a } n \times n \text{ matrix} \\ b \text{ a } n \times 1 \text{ vector} \end{cases}$

Choose u a random $n \times 1$ vector

Compute $y_i := A^i b$ for $i = 0, \cdots, (n-1)$

Compute $s_i := \langle A^i b, u\rangle$ for $i = 0, \cdots, (2n-1)$

$f(x) := X^n + \sum_{i=0}^{n-1} f_i X^i := Berlekamp(s_0, s_1, \ldots, s_{2n-1})$

$x := \sum_{i=1}^{n} \frac{-f_i}{f_0} y_{i-1}$

return x

In the previous form the last algorithm is probabilistic but it is possible to derive a deterministic one. Note also that there exists a more efficient version

of the Wiedemann algorithm: the Wiedemann algorithm by blocks (but we have not implemented this version).

The key operation in these algorithms is the multiplication $A \times y$. It is very easy to implement this operation efficiently to take advantage of the sparsity of A and to obtain a complexity of $O(|A|)$ for computing $A \times y$ where $|A|$ is the number of non zero elements in A; thus the global complexity for solving the system is $O(n|A|)$ instead of $O(n^3)$.

22.7 Décomposition LU (*en anglais*)

These methods try to decompose the input matrix A into a product LU where L (resp. U) is a lower (resp. upper) triangular matrix with the additional constraint that the number of non zero elements in L and U minus the number of non zero elements in A is as smallest as possible (this number is called the number of "fill-ins"). The first step of these algorithms is to *predict* the shape of L and U without computation, or more precisely with a *symbolic* factorization of the matrix A (George A. and Liu J. W. H., 1981; Reid J.K., 1971; Rose D. and Willoughby R.A., 1972; Duff I.S. and Reid J.K., 1984). We try to sketch the algorithm for understanding the basic idea: we try to discover the shape of L and U for the following matrix A:

$$A = \begin{bmatrix} ? & ? & 0 & 0 & 0 \\ ? & ? & ? & 0 & 0 \\ 0 & 0 & ? & ? & 0 \\ 0 & 0 & 0 & ? & ? \\ & 0 & 0 & 0 & ? \end{bmatrix}$$

where ? means a non zero element (but we do not know its real value). Let

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ L_{21} & 1 & 0 & 0 & 0 \\ L_{31} & L_{32} & 1 & 0 & 0 \\ L_{41} & L_{42} & L_{43} & 1 & 0 \\ L_{51} & L_{52} & L_{53} & L_{54} & 1 \end{bmatrix} \text{ and } U = \begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} & U_{15} \\ 0 & U_{22} & U_{23} & U_{24} & U_{25} \\ 0 & 0 & U_{33} & U_{34} & U_{35} \\ 0 & 0 & 0 & U_{44} & U_{45} \\ 0 & 0 & 0 & 0 & U_{55} \end{bmatrix}$$

then we compute $M := LU - A$ and we have the equation $M = 0$. Consequently

$$M_{1k} = U_{1k} - A_{1k}$$

and the first row of U is obviously the first row of A. Accordingly, the shape of the first row of U is:

$$\times \ \times \ 0 \ 0 \ 0$$

Here $\times$ means a (possibly) non zero element and 0 is a zero element. The next equation to consider is $M_{21} = L_{21}U_{11} - A_{21} = L_{21} \times - \times = 0$ so we can deduce that $L_{21} = \times$. In the same way, we deduce from $M_{22} = 0 = L_{21}U_{12} + U_{22} - A_{22} = \times \times + U_{22} - \times$, that $U_{22} = \times$. By applying the following symbolic rule:

$$\begin{array}{c} \times + \times \rightarrow \times \\ \times + 0 \rightarrow \times \\ 0 + 0 \rightarrow 0 \\ \times * \times \rightarrow \times \\ \times * 0 \rightarrow 0 \end{array}$$

we can estimate the shape of L and U:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \times & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \times & \times & \times & \times & 1 \end{bmatrix} \text{ and } U = \begin{bmatrix} \times & \times & 0 & 0 & 0 \\ 0 & \times & \times & 0 & 0 \\ 0 & 0 & \times & \times & 0 \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}$$

and estimate the number of fill-ins: $5 + 9 - 11 = 3$. We will use a similar idea in our algorithm to generate a matrix (and also to reduce it). It must be emphasized that the shape of L and U could be over estimated (it is correct if all the non zero coefficients in A are chosen at random). Practical experience will show that this overhead is small.

The second step of the sparse LU decomposition algorithms is to compute permutation of the rows (columns operations are not allowed for comptuing Gröbner bases) to minimize the number of fill-ins. This is often done by using graph algorithms (George A. and Liu J. W. H., 1981; George A. and Gilbert J.R. and Liu J. W. H., 1993).

The third step is to solve the system $LUx = b$ by solving first $Ly = b$ and then $Ux = y$.

References

Adams, W., & Loustaunau, P. 1994. *An introduction to Gröbner Bases.* Graduate Studies in Mathematics, vol. 3. American Mathematical Society.

Amari, A. 2000. Synthesis of Cross-Coupled Resonator Filters Using an Analytical Gradient-Based Optimization Technique. *IEEE Trans. Microwave Theory Tech.*, **MTT-48**(Sept), 1559–1564.

Arita, S. 1999. Algorithms for Computations in Jacobian Group of C_{ab} Curve and Their Application to Discrete-Log Based Public Key Cryptosystems. *IEICE Transactions*, **J82-A**(8), 1291–1299. In Japanese. English translation in the proceedings of the Conference on The Mathematics of Public Key Cryptography, Toronto 1999.

Ars, G., Faugère, J.-C., Imai, H., Kawazoe, M., & Sugita, M. 2004. Comparison between XL and Gröbner Basis Algorithms. *In:* LEE, Pil Joong (ed), *AsiaCrypt 2004.* Lectures Notes in Computer Science. Springer. Jeju Island, KOREA.

Ars, G. and Faugère, J.-C. 2005. Algebraic immunities of functions over finite fields. *In:* PURH, Presses Universitaires (ed), *Boolean Function : Cryptography and Applications.*

Attardi G. and Traverso C. 1994. A strategy-accurate parallel Buchberger algorithm. *Pages 12–21 of:* Hoon Hong (ed), *Pasco'94.* Lecture Notes in Computing, vol. 5. World Scientific.

Auzinger and Stetter H. 1998. An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. *Int. Series of Numerical Math.*, **86**, 11–30.

Bardet, M., Faugère, J.C, & B., Salvy. 2003a (Dec). *Complexity of Gröbner basis computation for semi-regular overdetermined sequences over GF(2) with solutions in GF(2).* Tech. rept. RR-5049. INRIA.

Bardet, M., Faugère, J.C, & B., Salvy. 2004 (Nov.). On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. *Pages 71–75 of:* Valibouze, A (ed), *ICPSS Paris.*

Bardet, Magali. 2004. *Étude des systèmes algébriques surdeterminés. Applications aux codes correcteurs et à la cryptographie.* Ph.D. thesis, Université Paris 6.

Bardet, Magali, Faugère, Jean-Charles, & Salvy, Bruno. 2003b. *Complexity of Gröbner bases computation of generic systems.* in preparation.

Bardet, Magali, Faugère, Jean-Charles, & Salvy, Bruno. 2005. Asymptotic Behaviour of the Index of Regularity of Semi-Regular Quadratic Polynomial Systems. *In: Proceedings of the8th MEGA (Effective Methods in Algebraic Geometry)*. 15 pages.

Basiri, A., & Faugère, J.-C. 2003a. Changing the ordering of Gröbner Bases with LLL: Case of Two Variables. *Pages 23–29 of:* Sendra, J. Rafael (ed), *Proceedings of ISSAC*. ACM Press.

Basiri, A., Enge, A., Faugère, J.-C., & Gürel, N. 2002a (November). *The arithmetic of Jacobian groups of superelliptic cubics*. Tech. rept. INRIA, available from *http://www.inria.fr/rrrt/rr-4618.html*.

Basiri, A., Enge, A., Faugère, J.-C., & Gürel, N. 2002b. *Fast Arithmetics for Superelliptic Cubics*. Extended version, available from *http://-www.lix.polytechnique.fr/Labo/Andreas.Enge/vorabdrucke/super.ps*.

Basiri, A., Enge, A., Faugère, J.-C., & Gürel, N. 2003. *Implementing the Arithmetic of Superelliptic Cubics*. In preparation.

Basiri, A., Enge, A., Faugère, J.C., & Gürel., N. 2004. The arithmetic of Jacobian groups of superelliptic cubics. *Mathematics of Computation*, **74**(July), 389–410.

Basiri, Abdolali. 2003. *Bases de Gröbner et LLL. Arithmétique rapide des courbes Cab*. Ph.D. thesis, Universite Paris 6.

Basiri, Abdolali, & Faugère, Jean-Charles. 2003b. Changing the ordering of Gröbner bases with LLL: case of two variables. *Pages 23–29 of: Proceedings of the 2003 international symposium on Symbolic and algebraic computation*. ACM Press.

Bauer, M.-L. 2001. *The Arithmetic of Certain Cubic Function Fields*. Preprint.

Becker, Eberhard, Mora, Teo, Marinari, Maria Grazia, & Traverso, Carlo. 1994. The shape of the Shape Lemma. *Pages 129–133 of: Proceedings of the international symposium on Symbolic and algebraic computation*. ACM Press.

Becker T. and Weispfenning V. 1993. *Groebner Bases, a Computationnal Approach to Commutative Algebra*. Graduate Texts in Mathematics. Springer-Verlag.

Berbain, Côme, Gilbert, Henri, & Patarin, Jacques. 2006. QUAD: A Practical Stream Cipher with Provable Security. *Pages 109–128 of:* Vaudenay, Serge (ed), *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*. Lecture Notes in Computer Science, vol. 4004. Springer.

Bermejo, I., & Gimenez, P. 2001. Computing the Castelnuovo-Mumford regularity of some subschemes of $\mathbb{P}_n^K$ using quotients of monomial ideals. *J. Pure Appl. Algebra*, **164**(1-2), 23–33.

Biham, E. 2000. Cryptanalysis of Patarin's 2-Round Public Key System with S Boxes (2R). *Pages 408–416 of: Advances in Cryptology – CRYPTO 2000*. Lecutures Notes in Computer Science, vol. 1807. Springer Verlag.

Billet, O. 2005. *Cryptologie multivariable*. Ph.D. thesis, Université de Versailles.

Billet, Olivier, & Gilbert, Henri. 2003. A traceable block cipher. *Pages 331–346 of:* Laih, Chi-Sung (ed), *Advances in cryptology – ASIACRYPT 2003*. Lecutures Notes in Computer Science, vol. 2894. Springer.

Biryukov, A., Cannière, C. De, Braeken, A., & Preneel, B. 2003. A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms. *Pages 33–50 of: Advances in Cryptology – EUROCRYPT 2003*. Lecutures Notes in Computer Science, vol. 2656. Springer Verlag.

Bourgeois, G. 2006 (Juin). *Attaque algébrique de NTRU à l'aide des vecteurs de Witt*. http://arxiv.org/ftp/cs/papers/0605/0605136.pdf.

Buchberger, B. 1983. A note on the complexity of constructing Gröbner bases. *Pages 137–145 of: van Hulzen J.A. (ed.) EUROCAL '83, European Computer Algebra Conference*. Lecture Notes in Computer Science, vol. 162. Springer.

Buchberger, B. 1987. History and Basic Features of the Critical-Pair/Completion Procedure. *Journal of Symbolic Computation*, **3**(1 and 2), 3–38.

Buchberger B. 1965. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. Ph.D. thesis, Innsbruck.

Buchberger B. 1970. An Algorithmical Criterion for the Solvability of Algebraic Systems. *Aequationes Mathematicae*, **4**(3), 374–383. (German).

Buchberger B. 1976. A theoretical basis for the reduction of polynomials to canonical forms. *ACM SIGSAM Bull.*, **39**, 19–29.

Buchberger B. 1979. A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Basis. *Pages 3–21 of: Proc. EUROSAM 79*. Lect. Notes in Comp. Sci., vol. 72. Springer Verlag.

Buchberger B. 1985. Gröbner Bases : an Algorithmic Method in Polynomial Ideal Theory. *Chap. 6, pages 184–232 of:* Reidel Publishing Company (ed), *Recent trends in multidimensional system theory*. Bose.

Cameron, R. J., Harish, A. R., & Radcliffe, C. J. 2002. Synthesis of advanced microwave filters without diagonal cross-couplings. *IEEE Trans. Microwave Theory Tech.*, **MTT-50**, 2862–2872.

Cameron, R.J. 1999. General Coupling Matrix Synthesis Method for Chebyshev Filtering Functions. *IEEE Trans. Microwave Theory Tech.*, **MTT-47**(April), 433–442.

Caniglia L. and Galligo A. and Heintz J. 1988. Some new effectivity bounds in computational geometry. *Pages 131–152 of: Proceedings of AAECC-6*. Lect. Notes in Comp. Sci., vol. 357. Springer Verlag.

Caniglia L. and Galligo A. and Heintz J. 1991. Equations for the projective closure and effective Nullstellensatz. *Discrete Applied Math.*, **33**, 11–23.

Cannon J. 1998 (Feb). *The Magma Computational Algebra System 2.20-7.* `http://www.maths.usyd.edu.au:8000/u/magma/`.

Char B. and Geddes K. and Gonnet G. and Leong B. and Monagan M. and Watt S. 1991. *Maple V Library Reference Manual.* Spinger-Verlag. Third Printing, 1993.

Chester, C., Friedman, B., & F.Ursell. 1957. An extension of the method of steepest descents. *Proc. Camb. Philos. Soc*, **53**, 599–611.

Cohen, A., & Daubechies, I. 1993. Non-Separable Bidimensional Wavelet Bases. *Rev. Mat. Iberoamericana*, **9**(1), 51–137.

Cohen, A., & Daubechies, I. 1995. "A new technique to estimate the regularity of refinable functions". *pre-print.*

Cohen, A., Gröchenig, K., & Villemoes, L. 1995. "Regularity of Multi-variate Refinable Functions". *pre-print.*

Cohen, Henri, Frey, Gerhard, Avanzi, Roberto, Doche, Christophe, Lange, Tanja, Nguyen, Kim, & Vercauteren, Frederik (eds). 2006. *Handbook of elliptic and hyperelliptic curve cryptography.* Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL.

Courtois, N. 2001a. *La sécurité des primitives cryptographiques basées sur des problèmes algébriques multivariables: MQ, IP, MinRank, HFE.* Ph.D. thesis, Université de Toulon.

Courtois, N., Goubin, L., & Patarin, J. 2004. *SFLASH, a Fast Asymmetric Signature Scheme for low-cost Smartcards – Primitive Specification and Supporting Documentation.* Tech. rept. Minrank. http://www.minrank.org/sflash-b-v2.pdf.

Courtois, Nicolas, Shamir, Adi, Patarin, Jacques, & Klimov, A. 2000. Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations. *Pages 392–407 of: Eurocrypt'2000.* Lecutures Notes in Computer Science, vol. 1807. Springer Verlag.

Courtois, Nicolas T. 2001b. The security of Hidden Field Equations (HFE). *Pages 266–281 of: Cryptographers' Track RSA Conference.* Lecutures Notes in Computer Science, vol. 2020.

Cox, D., Little, J., & O'Shea, D. 1992. *Ideals, Varieties and Algorithms.* Springer Verlag, New York.

Cox, D., Little, J., & O'Shea, D. 1998. *Using Algebraic Geometry.* Springer Verlag, New York.

Cox, David, Little, John, & O'Shea, Donal. 1996. *Ideals, Varieties, and Algorithms.* Undergraduate Texts in Mathematics. Springer-Verlag. second edition.

D., Augot, M., Bardet, & J.C., Faugère. 2003. Efficient Decoding of (binary) Cyclic Codes beyond the correction capacity of the code using Gröbner bases. *In: IEEE International Symposium on Information Theory (Japan).* ISIT 2003.

D., Bayer, & M., Stillman. 1987. A theorem on refining division orders by the reverse lexicographic orders. *Duke J. Math.*, **55**, 321–328.

Daubechies, I. 1988. "Orthogonal bases of compactly Supported Wavelets". *Communications on Pure and Applied Mathematics*, **XLI**, 909–996.

Daubechies, I. 1992. *"Ten Lectures on Wavelets".* CBMS-NSF Series in Applied Mathematics, no. 61. Philadelphia: SIAM.

Davenport J.H. and Siret Y. and Tournier E. 1993. *Calcul Formel.* Masson. 2^e édition révisée.

Delogne, P., & Macq, B. 1991. "Universal variable length coding for an integrated approach to image coding". *Annales des télécommunications*, **46**(7-8).

Dickerson, M. 1989 (July). *The functional Decomposition of Polynomials.* Ph.D. thesis, Cornell University, Ithaca. TR 89-1023.

Diffie, W., & Hellman, M.-E. 1976. New directions in cryptography. *IEEE Transactions on Information Theory*, **IT-22**(6), 644–654.

Dubois, V., Granboulan, L., & Stern, J. 2006. An Efficient Provable Distinguisher for HFE. *Pages 156–167 of: Proceedings of ICALP 2006, Part II.* Lecutures Notes in Computer Science, vol. 4052.

Duff I.S. and Reid J.K. 1984. The multifrontal solution of unsymmetric sets of linear equations. *SIAM J. Sci. Statist. Comput.*, **5**(3), 633–641.

Duquesne, Sylvain, & Lange, Tanja. 2006. Arithmetic of hyperelliptic curves. *Pages 303–353 of: Handbook of elliptic and hyperelliptic curve cryptography.* Discrete Math. Appl. (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL.

Eisenbud, David. 1995. *Commutative Algebra with a View Toward Algebraic Geometry.* Graduate Texts in Mathematics, vol. 150. Springer-Verlag.

Enge, A. 2001. A General Framework for Subexponential Discrete Logarithm Algorithms in Groups of Unknown Order. *Pages 133–146 of:* Blokhuis, A., Hirschfeld, J. W. P., Jungnickel, D., & Thas, J. A. (eds), *Finite Geometries.* Developments in Mathematics, vol. 3. Dordrecht: Kluwer Academic Publishers.

Enge, A. 2002. Computing Discrete Logarithms in High-Genus Hyperelliptic Jacobians in Provably Subexponential Time. *Mathematics of Computation*, **71**(238), 729–742.

Enge, A., & Gaudry, P. 2002. A general framework for subexponential discrete logarithm algorithms. *Acta Arithmetica*, **102**(1), 83–103.

Faugère, J.-C., Gianni, P., Lazard, D., & Mora, T. 1993. Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. *Journal of Symbolic Computation*, **16**, 329–344.

Faugère, Jean-Charles. 1994. Parallelization of Gröbner bases. *Pages 124–132 of:* Hong, Hoon (ed), *Parallel and Symbolic Computation.* Lecture Notes in Computing, vol. 5. World Scientific.

Faugère, J.-C. and Perret L. 2005. *Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects.* Tech. rept. Universitè Catholique de Louvain. submitted to EuroCrypt 2006.

Faugère J.C. 1999. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, **139**(1–3), 61–88.

Faugère J.C. and Lazard D. 1995. The Combinatorial Classes of Parallel Manipulators. *Mechanism and Machine Theory*, **30**(6), 765–776.

Faugère J.C. and Merlet J.P. and Rouillier F. 2007. On solving the direct kinematics problem for parallel robots . *International Journal of Robotics Research.* under revision.

Faugère J.C. and Moreau de Saint-Martin F. and Rouillier F. 1996. *Synthèse de bancs de filtres et ondelettes bidimensionnels par le calcul formel.* Rapport Interne CCETT . CNET.

Faugère J.C. and Moreau de Saint-Martin F. and Rouillier F. 1998. Design of nonseparable bidimensional wavelets and filter banks using Gröbner bases techniques. *IEEE SP Transactions on Signal Processing*, **46**(4). Special Issue on Theory and Applications of Filter Banks and Wavelets.

Faugère J.C. and Rouillier F. 2005 (october). Polynomial System Solving for Industrial Problem. *In: SIAM Conference on Mathematics for Industry.* Detroit, Michigan.

Faugère, J.C., Gianni, P., Lazard, D. and Mora T. 1993. Efficient Computation of Zero-Dimensional Gröbner Basis by Change of Ordering. *Journal of Symbolic Computation*, **16**(4), 329–344.

Faugère, J.-C., & Joux, A. 2003. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner bases. *Pages 44–60 of:* Boneh, Dan (ed), *Advances in Cryptology - CRYPTO 2003.* LNCS, vol. 2729. Springer.

Faugère, J.-C., Hering, M., & Phan, J. 2000. The Membrane Inclusions Curvature Equations. *In:* Singer, M. (ed), *Mega 2000 Bath (UK).*

Faugère, J.-C., Hering, M., & Phan, J. 2003. The membrane inclusions curvature equations. *Advances in Applied Mathematics*, **31**(4), 643–658.

Faugère, Jean-Charles. 2006 (6). *Groebner Bases attack – 2R.*

Faugère, Jean-Charles, & Perret, Ludovic. 2006a. Cryptanalysis of 2R– schemes. *Pages 357–372 of:* Dwork, Cynthia (ed), *Advances in Cryptology CRYPTO 2006.* Lecture Notes in Computer Science, vol. 4117. Springer-Verlag.

Faugère, Jean-Charles, & Perret, Ludovic. 2006b. Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. *In:* Vaudenay, Serge (ed), *EuroCrypt 2006 Advances in Cryptology.* Lecture Notes in Computer Science, vol. 4004. Springer-Verlag.

Faugère, Jean-Charles, & Perret, Ludovic. 2006c. Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. *In:* Bernstein, D.J. (ed), *International Workshop on Post-Quantum Cryptography 2006.*

Faugère, Jean-Charles, & Rouillier, Fabrice. 2005. Outils de calcul formel performants pour la résolution de certains problèmes en robotique. *Pages 89–99 of: Journées Nationales de Recherche en Robotique.*

Faugère, Jean-Charles, Cameron, R.j., & Seyfert, Fabien. 2005. Coupling matrix synthesis for a new class of microwave filter configuration. *Pages 119–124 of: IEEE MTT-S International Microwave Symposium digest*, vol. 1. Institute of Electrical and Electronics Engineers, New-York, NC, ETATS-UNIS (1977). isbn: 0-7803-8846-1.

Faugère, Jean-Charles, Rouillier, Fabrice, Seyfert, Fabien, & Cameron, R.j. à paraître. An Exhaustive Approach to the Coupling Matrix Synthesis Problem Application to the Design of High Degree Asymmetric Filters. *International Journal of RF and Microwave Computer-Aided Engineering Special Issue on RF and Microwave Filters, Modeling and Design Pierre Jarry and Hussein Baher Guest Editors*, 1.

Faugère J.C. 2002. A new efficient algorithm for computing Gröbner bases without reduction to zero F5. *Pages 75–83 of:* T. Mora (ed), *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation.* ACM Press.

Felke, P. 2005 (March). On certain Families of HFE-type Cryptosystems. *In: Proceedings of WCC05, International Workshop on Coding and Cryptography.*

Flon, S., & Oyono, R. 2003. *Fast Arithmetic on Jacobians of Picard curves.* Available from, *http://citeseer.nj.nec.com/577354.html.*

Flon, Stéphane, Oyono, Roger, & Ritzenthaler, Christophe. 2004. *Fast addition on non-hyperelliptic genus* 3 *curves.* Cryptology ePrint Archive, Report 2004/118. http://eprint.iacr.org/.

Fortin, S. 1996. *The Graph Isomorphism problem.* Tech. rept. 20. University of Alberta.

Fraenkel, A. S., & Yesha, Y. 1979. Complexity of problems in games, graphs and algebraic equations. *Discrete Appl. Math.*, **1**(1), 15–30.

Fraenkel, A. S., & Yesha, Y. 1980. Complexity of solving algebraic equations. *Inform. Process. Lett.*, **10**(4), 178–179.

Fröberg, R. 1985. An inequality for Hilbert series of graded algebras. *Math. Scand.*, **56**(2), 117–144.

Fröberg, R. 1997. *An introduction to Gröbner Bases.* Pure and Applied Mathematics. John Wiley and Sons Ltd. Chichester.

Fulton, W. 1969. *Algebraic Curves.* Mathematics Lecture Note Series. Benjamin Inc.

G., Ars. 2005 (June). *Applications des bases de Gröbner à la cryptographie.* Ph.D. thesis, Université de Rennes 1.

Galbraith, S.-D., Paulus, S., & Smart, N.-P. 2002. Arithmetic on Superelliptic Curves. *Mathematics of Computation*, **71**(237), 393–405.

Garey, M. R., & Johnson, D. B. 1979. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman.

Gaudry, P. 2000 (Décembre). *Algorithmique des courbes hyperelliptiques et applications à la cryptologie*. Ph.D. thesis, ECOLE POLYTECHNIQUE. Available from, *www.lix.polytechnique.fr/Labo/Pierrick.Gaudry/*.

Gebauer, R., & Moller, H. M. 1986 (July). Buchberger's Algorithm and Staggered Linear Bases. *Pages 218–221 of: Proceedings of the 1986 Symposium on Symbolic and Algebraic Computation*.

Gebauer, R., & Möller, H.M. 1988. On an Installation of Buchberger's Algorithm. *Journal of Symbolic Computation*, **6**(2 and 3), 275–286.

Geiselmann, W., Steinwandt, R., & Beth, T. 2001. Attacking the Affine Parts of SFLASH. *In: Cryptography and Coding, 8th IMA International Conference*. Lecutures Notes in Computer Science, vol. 2260. Springer Verlag.

George A. and Gilbert J.R. and Liu J. W. H. 1993. *Graph theory and sparse matrix computation*. New York : Springer-Verlag.

George A. and Liu J. W. H. 1981. *Computer solution of large sparse positive definite systems*. Englewood Cliffs, N.J. : Prentice-Hall.

Gerdt V.P. 1995. Involutive Polynomial Bases. *In: PoSSo on software*. Paris, F.

Gianni P. and Mora T. 1987. Algebraic Solution of Systems of Polynomial Equations Using Gröbner Bases. *Pages 247–257 of: AAECC–5*. Lect. Notes in Comp. Sci., vol. 356.

Giovini A. and Mora T. and Niesi G. and Robbiano L. and Traverso C. 1991. One sugar cube, please, or Selection strategies in the Buchberger Algorithm. *In:* S. M. Watt (ed), *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*. ACM Press.

Giusti, M. 1994. Some effectivity problems in polynomial ideal theory. *Pages 159–171 of: Proc. Int. Symp. on Symbolic and Algebraic Computation EUROSAM 84, Cambridge (England)*. LNCS, vol. 174. Springer.

Goubin, L., & Patarin, J. 1997. Trapdoor One-way Permutations and Multivariate Polynomials. *Pages 356–368 of: Information and Communication Security, First International Conference (ICICS'97*. Lecutures Notes in Computer Science, vol. 1334. Springer–Verlag.

Greuel G.-M. and Pfister G. and Schoenemann H. 2002 (July). *SINGULAR 2.0*. `http://www.singular.uni-kl.de/`.

Gutierrez, J., Rubio, R., & von zur Gathen, J. 2003. Multivariate Polynomial Decomposition. *Algebra in Engineering, Communication and Computing*, **14**(1), 11–31.

Harasawa, R., & Suzuki, J. 2000. Fast Jacobian Group Arithmetic on C_{ab} Curves. *Pages 359–376 of:* Bosma, Wieb (ed), *Algorithmic Number Theory — ANTS-IV*. Lecture Notes in Computer Science, vol. 1838. Berlin: Springer-Verlag.

Hashemi, Amir. 2006. *Structure et complexité des base de Gröbner*. Ph.D. thesis, Université Paris 6.

Hoffman., M. 1982. *Group-theoretic algorithms and Graph Isomorphism.* Lecutures Notes in Computer Science. Springer–Verlag. Chap. 1.

Hoffstein, Jeffrey, Pipher, Jill, & Silverman, Joseph H. 1998. NTRU: A Ring-Based Public Key Cryptosystem. *Pages 267–288 of:* Buhler, J.P. (ed), *Algorithmic Number Theory (ANTS III), Portland, OR.* Lecutures Notes in Computer Science, vol. 1423. Springer-Verlag.

J., Apel, & R., Hemmecke. 2002. *Detecting unnecessary reductions in an involutive basis computation.* Tech. rept. RISC Linz Report Series.

J., Sylvester. 1853. On a theory of syzygetic relations of two rational integral functions, comprising an application to the theory of Sturm's functions, and that of the greatest Algebraical Common Measure. *Philosophical Trans.*, **143**, 407–548.

J.Buchmann, A.Pychkine, & R.P.Weinmann. 2005. Block ciphers sensitive to Gröbner Attacks. *Pages 313–31 of: In The Cryptographers' Track at the RSA Conference 2005.* Lecutures Notes in Computer Science.

J.Buchmann, A.Pychkine, & R.P.Weinmann. 2006. A Zero-Dimensional Groebner Basis for AES-128. *Pages 313–31 of: In Fast Software Encryption: 13th International Workshop, FSE 2006, Graz, Austria.* Lecutures Notes in Computer Science.

J.C., Faugère. 2001 (mai). Optimisation globale et calcul formel. *In:* Rouillier, F. (ed), *Congrès National de Mathématiques Appliquées et Industrielles.* Utilisation du calcul formel en calcul scientitfique.

Joux, A., & Stern, J. 1998. Lattice reduction: a toolbox for the cryptanalyst. *Journal of Cryptology*, **11**(3), 161–185.

Joux, A., Granboulan, L., & Stern, J. 2006. Inverting HFE is Quasipolynomial. *Page ?? of:* Dwork, Cynthia (ed), *Advances in Cryptology CRYPTO 2006.* Lecture Notes in Computer Science, vol. 4117. Springer-Verlag.

Kailath, T. 1980. *Linear Systems.* Prentice Hall.

Kaltofen E. 1991. On Wiedemann's Method of Solving Sparse Linear Systems. *LNCS*, **539**, 29–38. AAECC-9.

Katsura K. 1986. Theory of spin glass by the method of the distribution function of an effective field. *Progress of Theoretical Physics*, **87**, 139–154. Supplement.

Kim, K.S., J., Neu, & G., Oster. 1998. Curvature-mediated interactions between membrane proteins. *Biophysical Journal*, **75**, 2274–2291.

Kipnis, Aviad, & Shamir, Adi. 1999. Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. *Pages 19–30 of: Advances in Cryptology – CRYPTO 1999.* Lecutures Notes in Computer Science, vol. 1666.

Koblitz, N. 1987. Elliptic curve cryptosystems. *Mathematics of Computation*, **48**(177), 203–209.

Koblitz, N. 1989. Hyperelliptic Cryptosystems. *Journal of Cryptology*, **1**, 139–150.

Kovacevic, J., & Vetterli, M. 1992. "Nonseparable Multidimensional Perfect Reconstruction Filter Banks and Wavelet Bases for R^n". *IEEE Transactions on Information Theory*, **38**(2), 533–555.

Kovacevic, J., & Vetterli, M. 1995. "Nonseparable Two- and Three-Dimensional Wavelets". *IEEE Transactions on Signal Processing*, **43**(5), 1269–1272.

Kozen, D., & Landau, S. 1989. Polynomial Decomposition Algorithms. *Journal of Symbolic Computation*, **7**(5), 445–456.

L., Massey J. 1969. Shift-Register Synthesis and BCH Decoding. *IEEE Trans. on Information Theory*, **15**(1), 122–127.

Lachartre, Sylvain. 2006. *Algèbre linéaire dans la résolution de systèmes polynomiaux. Application en cryptographie.* Ph.D. thesis, Université de Paris 6.

Lang, S. 2002. *Algebra.* Graduate Texts in Mathematics, vol. 211. Springer-Verlag New York. third edition.

Lazard, D. 1983. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. *Pages 146–157 of: van Hulzen J.A. (ed.) EUROCAL '83, European Computer Algebra Conference.* Lecture Notes in Computer Science, vol. 162. Springer-Verlag.

Lazard, D. 1988. Quantifier Elimination:Optimal Solutions for Two Classical Examples. *Journal of Symbolic Computation*, **5**(1 and 2), 261–266.

Lazard, D. 1992. Solving zero-dimensional algebraic systems. *Journal of Symbolic Computation*, **13**(2), 117–132.

Lazard D. 1981. Resolution des systemes d'equations algebriques. *Theor. Comp. Science*, **15**, 77–110.

Lazard D. 1983. Gaussian Elimination and Resolution of Systems of Algebraic Equations. *Pages 146–157 of: Proc. EUROCAL 83.* Lect. Notes in Comp. Sci, vol. 162.

Lenstra, A.-K. 1985. Factoring multivariate polynomials over finite fields. *Journal of Computer and System Sciences*, **30**(2), 235–248.

Lenstra, A.-K., Lenstra, H.-W., & Lovász, L. 1982. Factoring polynomials with rational coefficients. *Math. Ann.*, **261**, 515–534.

Macaulay, F.S. 1916. *The algebraic theory of modular systems.* Cambridge Mathematical Library., vol. xxxi. Cambridge University Press.

Macaulay, F.S. 1927. Some properties of enumeration in the theory of modular systems. *Proc. London Math. Soc.*, **26**, 531–55.

Macchiarella, G. 2003 (June). A powerful tool for the synthesis of prototype filters with arbitrary topology. *Pages 1721–1724 of: IEEE MTT-S Int. Microwave Symp. Dig.*, vol. 3.

Mahler, K. 1941. An analogue of Minkowski's geometry of numbers in a field of series. *Pages 488–522 of: Annals of Math*, vol. 42.

Matsumoto, T., & Imai, H. 1985. Algebraic Methods for Constructing Asymmetric Cryptosystems. *Pages 108–119 of: Algebraic and Error-*

Correcting Codes. Proc. Third Intern. Conf., Grenoble. Lecutures Notes in Computer Science. Springer–Verlag.

Matsumoto, Tsutomu, & Imai, Hideki. 1988. Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption. *Pages 419–453 of: Advances in Cryptology – EUROCRYPT 1988.* Lecutures Notes in Computer Science, vol. 330. Springer Verlag.

Merlet J.P. 1990. *Les robots parallèles.* Robotique. Hermes.

Miller, V. 1987. Use of elliptic curves in cryptography. *Pages 417–426 of:* Odlyzko, A. M. (ed), *Advances in Cryptology – CRYPTO 86.* Lecture Notes in Computer Science, vol. 263. Berlin: Springer-Verlag.

Miura, S. 1998. Linear Codes on Affine Algebraic Curves. *IEICE Transactions*, **J81-A**, 1398–1421. In Japanese. English summary by Ryutaroh Matsumoto available at *http://www.rmatsumoto.org/cab.html*.

Miura, S., & Kamiya, N. 1993 (June). Geometric Goppa codes on some maxima curves and their minimum distance. *Pages 85–86 of: IEEE Worksop on Information Theory.*

Möller H.M. 1993. Systems of algebraic equations solved by means of endomorphisms. *Pages 43–56 of: Proceedings of AAECC-10.* Lect. Notes in Comp. Sci., vol. 673.

Montgomery P. 1995. A Block Lanczos Algorithm for Finding Dependencies over GF(2). *LNCS*, May, 106–120.

Mora, T. and Möller, H.M. and Traverso, C. 1992. Gröbner Bases Computation Using Syzygies. *Pages 320–328 of:* Wang, Paul S. (ed), *ISSAC 92.* ACM Press.

Moreau de Saint-Martin, F. 1997. *"Bancs de filtres et ondelettes : synthèses, adaptativité et applications".* Ph.D. thesis, CEREMADE, Université Paris IX-Dauphine.

Mourrain, Bernard, & Trébuchet, Philippe. 2005. Generalized Normal Forms and Polynomial System Solving. *Pages 253–260 of:* Krauers, M. (ed), *International Symposium on Symbolic and Algebraic Computation.* ACM Press.

Nessie. 2004. *https://www.cosic.esat.kuleuven.be/nessie/deliverables/decision-final.pdf.*

Nguyen, P.-Q., & Stern, J. 2000. Lattice Reduction in Cryptology: An Update. *Pages 85–112 of: Proceedings of ANTS IV.* Lecture Notes in Computer Science, vol. 1838. Springer-Verlag.

Nguyen, P.-Q., & Stern, J. 2001. The Two Faces of Lattices in Cryptology. *Pages 146–180 of: Proceedings of CALC '01.* Lecture Notes in Computer Science, vol. 2146. Springer-Verlag.

Nussbaumer, H.-J. 1981. *Fast Fourier Transform and Convolution Algorithms.* Springer Verlag.

Onno, P. 1996. *"Bancs de filtres et quantification vectorielle sur réseau : Étude conjointe pour la compression d'images".* Ph.D. thesis, Université de Rennes I, France.

Pardue, K., & Richert, B. 2003. *Syzygies of semi-regular sequences.* Preprint available at: http://www.math.lsa.umich.edu/ brichert/publications.

Park, H., Kalker, T., & Vetterli, M. 1996. Groebner Bases and Multidimensional FIR Multirate Systems. *Journal of Multidimensional Systems and signal.*

Patarin, Jacques. 1995a. Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88. *Pages 248–261 of: Proc. of the 15th Annual International Cryptology Conference on Advances in Cryptology - CRYPTO'95*, vol. 963.

Patarin, Jacques. 1995b. *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88.* Extended version.

Patarin, Jacques. 1996a. *HFE first challenge.* http://www.minrank.org/challenge1.txt.

Patarin, Jacques. 1996b. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. *Pages 33–48 of: Advances in Cryptology – EUROCRYPT 1996.* Lecutures Notes in Computer Science, vol. 1070.

Patarin, Jacques. 1996c. *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms.* Extended version Available from `http://www.minrank.org/hfe/`.

Patarin, Jacques, & Goubin, Louis. 1997a. Asymmetric Cryptography with S-Boxes. *Pages 369–380 of: Information and Communication Security, First International Information and Communications Security Conference.* Lecutures Notes in Computer Science.

Patarin, Jacques, & Goubin, Louis. 1997b. *Asymmetric Cryptography with S-Boxes.* Extended version.

Patarin, Jacques, Goubin, Louis, & Courtois, Nicolas. 1998a. Improved Algorithms for Isomorphisms of Polynomials. *Advances in Cryptology - EUROCRYPT 1998*, **1403**, 184–200.

Patarin, Jacques, Goubin, Louis, & Courtois, Nicolas. 1998b. *Improved Algorithms for Isomorphisms of Polynomials – Extended Version.*

Paulus, S. 1998. Lattice Basis Reduction in Function Fields. *Pages 567–575 of:* Buhler, J. P. (ed), *Algorithmic Number Theory — ANTS-III.* Lecture Notes in Computer Science, vol. 1423. Berlin: Springer-Verlag.

Pelzl, J., Wollinger, T., Guajardo, J., & Paar, C. 2003. *Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves.* Available from, *http://citeseer.nj.nec.com/pelzl03hyperelliptic.html.*

Ramchandran, K., & Vetterli, M. 1993. "Best wavelet packet bases in a rate-distortion sense". *IEEE Transactions on Image Processing*, **2**(2), 160–175.

Reid J.K. 1971. *Large Sparse Sets of Linear Equations.* Academic Press . London and New York.

Rioul, O. 1992. "Simple regularity criteria for subdivision schemes". *SIAM Journal Math. Anal.*, **23**(6), 1544–1576.

Rose D. and Willoughby R.A. 1972. *Sparse Matrices and their applications.* Plenum Press.

Rouillier, F. 1999. Solving zero-dimensional systems through the rational univariate representation. *Journal of Applicable Algebra in Engineering, Communication and Computing*, **9**(5), 433–461.

Rouillier, F., & Zimmermann, P. 2003. Efficient Isolation of Polynomial Real Roots. *Journal of Computational and Applied Mathematics*, **162**(1), 33–50.

Rouillier F. 1996. *Algorithmes efficaces pour l'étude des zéros réels des systèmes plynomiaux.* Ph.D. thesis, Université de Rennes I.

Seyfert, F., Baratchart, L., Marmorat, J.P., Bila, S., & Sombrin, J. 2003 (June). Extraction of Coupling Parameters For Microwave Filters: Determination of a Stable Rational Model from Scattering Data. *Pages 25–28 of: IEEE MTT-S Int. Microwave Symp. Dig.*, vol. 3.

Shoham, Y., & Gersho, A. 1988. "Efficient bit allocation for an arbitrairy set of quantizers". *IEEE Transactions on Acousticcs, Speech and Signal Processing*, **36**(Sept.).

Shoup, V. 2003. *NTL 5.3.1, a Library for doing Number Theory.* `http://www.shoup.net/ntl`.

Smart, Nigel, Vercauteren, Fre, & Silverman, Joseph H.. 2005. An algebraic approach to NTRU ($q = 2^n$) via Witt vectors and overdetermined systems of nonlinear equations. *Pages 278–298 of: Security in Communication Networks (SCN 2004).* Lecutures Notes in Computer Science, vol. 3352. Springer-Verlag.

Stanhill, D., & Zeevi, Y. Y. 1995a. "Two-Dimensional Linear-Phase Orthogonal Filter-Banks and Wavelets". *pre-print.*

Stanhill, D., & Zeevi, Y. Y. 1995b. "Two-Dimensional Orthogonal Wavelets with Vanishing Moments". *pre-print.*

Steel, Allan. 2004. *Allan Steel's Gröbner Basis Timings Page.* http://magma.maths.usyd.edu.au/users/allan/gb/.

Stewart, Ian. 1989. *Galois Theory.* second edition edn. CRC Press.

Stichtenoth, H. 1993. *Algebraic function field and codes.* Springer-verlage.

Sugita, Makoto, Kawazoe, Mitsuru, Imai, Hideki, Watanabe, Hideyuki, & Mitsunari, Shigeo. 2006. Implementation of F4 algorithm, and Experimental Cryptanalysis of Toyocrypt and 58-round SHA-1 using Gröbner Bases. *In: Linz Workshop D1.*

Szanto, A. 2004. Multivariate subresultants using Jouanolou s resultant matrices. *Journal of Pure and Applied Algebra.* to appear.

the arithmetic of $C_{3,4}$ curves., Implementing. 2004. A. Basiri and A. Enge and J.C. Faugère and N. Gürel. *Pages 57–71 of:* Buell, Duncan (ed), *Sixth Algorithmic Number Theory Symposium, ANTS VI proceedings (Vermont, USA, June 13–18, 2004).* Lectures Notes in Computer Science, vol. 3076. Springer-Verlag.

Traverso C. 1998. Gröbner trace algorithms. *Pages 125–138 of:* Gianni P. (ed), *ISSAC '88, Roma*, vol. 8. Lect. Notes in Comp.Sc.

V. Carlier and H. Chabanne and E. Dottax. 2005. Grey Box Implementation of Block Ciphers Preserving the Confidentiality of their Design. *In:* PURH, Presses Universitaires (ed), *Boolean Function : Cryptography and Applications*.

v.-z. Gathen, J., & Gerhard, J. 1999. *Modern Computer Algebra*. Cambridge universty press.

Van der Waerden B.L. 1991. *Algebra*. Springer-Verlag. seventh edition.

Venkataranam, S., & Levy, B. C. 1994a. "Nonseparable Orthogonal Linear Phase Perfect Reconstruction Filter Banks and their Application to Image Compression". *Proc. of Int. Conf. on Image Processing*, 334–338.

Venkataranam, S., & Levy, B. C. 1994b. "State Space Representations of 2-D FIR Lossless Matrices". *IEEE Transactions on Circuits and Systems–II: Analog and DIgital Signal Processing*, **41**(2), 117–131.

Venkataranam, S., & Levy, B. C. 1995. "A Comparison of Design Methods for 2-D FIR Orthogonal Perfect Reconstruction Filter Banks". *IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing*, **42**(8), 525–536.

Villemoes, L. F. 1992. "Sobolev regularity of wavelets and stability of iterated filter banks". *Pages 243–251 of:* Meyer, Yves, & Roques, Sylvie (eds), *Progress in wavelet analysis and applications*. Editions frontières. Proceedings of Int. Conf. "Wavelets and Applications", Toulouse, France, 1992.

Villemoes, L. F. 1997. "Continuity of nonseparable quincunx wavelets". *Appl. Comput. Harmon. Anal.*

Viscito, E., & Allebach, J. P. 1991. "The Analysis and Design of Multidimensional FIR Perfect Reconstruction Filter Banks for Arbitrary Sampling Lattices". *IEEE Transactions on Circuits and Systems*, **38**(1), 29–41.

Von Zur Gathen, J. 1984. Hensel and Newton methods in valuation rings. *Mathematics of Computation*, **42**(166), 637–661.

Von zur Gathen, J. 1990. Functional Decomposition of Polynomials:The Tame Case. *Journal of Symbolic Computation*, **9**(3), 281–300.

Von Zur Gathen, J. 1990. Functional Decomposition of Polynomials:The Wild Case. *Journal of Symbolic Computation*, **10**(5), 437–452.

von zur Gathen, Joachim, & Gerhard, Jürgen. 1999. *Modern Computer Algebra*. Cambridge Press.

V.P. Gerdt and Yu.A.Blinkov. 1998. Involutive Bases of Polynomial Ideals. *athematics and Computers in Simulation*, **45**, 519–542.

Wiedemann D. 1986. Solving Sparse Linear Equation Over Finite Fields. *IEEE Transaction on Information Theory*, **IT-32**(1), 54–62.

Ye, D.F., Lam, K.Y., & Dai, Z.D. 1999. Cryptanalysis of "2R" Schemes. *Pages 315–325 of: Advances in Cryptology – CRYPTO 1999*. Lecutures Notes in Computer Science, vol. 1666.

Ye, D.F., Lam, K.Y., & Dai, Z.D. 2001. Decomposing Attacks on Asymmetric Cryptography Based on Mapping Compositions. *Journal of Cryptology*, **14**, 137–150.

Index