Thèse de doctorat de l'École polytechnique

Pour obtenir le titre de Docteur en Sciences

Spécialités : mathématiques et informatique

Grégoire Lecerf

Une alternative aux méthodes de réécriture pour la résolution des systèmes algébriques

Soutenue le 14 septembre 2001

devant le jury composé de :

Michel Demazure (président),
Jean-Charles Faugère (examinateur),
Marc Giusti (directeur),
Monique Lejeune-Jalabert (examinatrice),
Bruno Salvy (co-directeur),
Michael Stillman (rapporteur),
Jean-Claude Yakoubsohn (rapporteur).

Une alternative aux méthodes de réécriture pour la résolution des systèmes algébriques

Grégoire Lecerf

Cette thèse a été composée avec les logiciels (LᴬTEX. L'auteur de TEX, D. E. Knuth et de LATEX L. Lamport sont responsables de la qualité typographique de ce document, mais pas des fautes qu'il contient.

Version du 9 novembre 2001.

Table des matières

Remerciements

Qu'il me soit permis, au seuil de cette thèse, d'adresser ma plus vive reconnaissance à Marc Giusti et Bruno Salvy, sans qui ce travail n'aurait pu voir le jour.

Outre tous ceux dont les noms, égrenés au fil des pages, montrent assez la dette intellectuelle que j'ai contractée à leur égard, je voudrais remercier ici tous les membres du groupe TERA. Les conseils savants et la bienveillance de Joos Heintz et Luis-Miguel Pardo m'ont été particulièrement précieux.

Éric Schost et Lutz Lehmann ont été les premiers à utiliser le prototype Kronecker puis à y contribuer. L'équipe de John Cannon m'a accueilli et s'est intéressé à mes recherches. Allan Steel a apporté de spectaculaires améliorations à Magma, contribuant ainsi aux performances de mon code. Je leur en suis très reconnaissant.

Je remercie les rapporteurs Michael Stillman et Jean-Claude Yakoubsohn pour leur effort de lecture. Michel Demazure, Jean-Charles Faugère et Monique Lejeune-Jalabert m'ont fait l'honneur de participer à mon jury, je les en remercie.

Ma dette de reconnaissance ne serait pas acquittée si je ne rappelais ici tout ce que je dois au laboratoire GAGE et l'unité de services MEDICIS. Marc Giusti, François Ollivier, Teresa Gómez Díaz et Nicole Dubois en assurent le bon fonctionnement avec une grande compétence.

Cette thèse a été financée une année par l'École Normale Supérieure de Paris puis trois années par le Rectorat de Versailles. Mes ressources matérielles, logicielles et mes déplacements ont été généreusement pris en charge par le Centre National de la Recherche Scientifique et l'École polytechnique.

Je n'ai jamais été assez loin pour bien sentir l'application de l'algèbre à la géométrie. Je n'aimais point cette manière d'opérer sans voir ce qu'on fait, et il me sembloit que résoudre un problème de géométrie par les équations, c'étoit jouer un air en tournant une manivelle. La première fois que je trouvai par le calcul que le carré d'un binôme étoit composé du carré de chacune de ses parties, et du double produit de l'une par l'autre [...] je n'en voulus rien croire jusqu'à ce que j'eusse fait la figure.

J.-J. Rousseau, *Les confessions*

Chapitre I

Introduction

Introduite par Hironaka au milieu des années 1960, la notion de base standard d'un idéal dans un anneau de polynômes connaît depuis les travaux de Buchberger un engouement particulier en mathématiques et informatique. La construction effective d'une telle base est désormais une fonctionnalité essentielle dans tous les logiciels de calcul formel. Les algorithmes sous-jacents sont sans cesse améliorés et permettent de traiter des problèmes concrets inaccessibles aux méthodes numériques. Et pourtant la complexité de ces algorithmes est doublement exponentielle dans le pire des cas. Dans les années 1990, Giusti et Heintz montrent que les problèmes d'élimination peuvent être ramenés dans une classe polynomiale en représentant les polynômes éliminant par des calculs d'évaluation. Sur la base de leurs travaux, ma thèse aboutit à un algorithme de calcul d'une décomposition en composantes équidimensionnelles de l'ensemble des solutions d'un système d'équations et d'inéquations polynomiales. Sa complexité est polynomiale en un « degré intrinsèque » et est la meilleure connue actuellement. J'ai implanté cet algorithme dans le système de calcul formel Magma. Nous avons appelé mon paquetage Kronecker, en hommage à l'illustre mathématicien pour ses travaux sur l'élimination. Les performances de mon logiciel sont à la hauteur des meilleures implantations dans le langage C de calcul de bases standard.

I.1 Résultat principal

Très souvent, on rencontre un système d'équations et d'inéquations polynomiales, modélisant le problème que l'on étudie. Qu'il s'agisse de situations physiques ou bien de questions mathématiques abstraites on cherche une description des solutions. Par exemple deux points P_i, $i = 1,2$, d'une même partie solide d'un mécanisme de l'espace, à coordonnées (x_i,y_i,z_i), sont reliés par une relation de la forme

$$(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 = l^2,$$

où l représente la distance qui sépare les points. S'ils ne font plus partie d'un même solide et si leur distance l peut varier on peut avoir envie d'imposer la condition $l \neq 0$.

Par résoudre un tel système nous entendons trouver une description des solutions dans la clôture algébrique du corps de base k, en terme d'extension algébrique de k. Les solutions isolées sont codées par une *représentation de Kronecker*. Une telle représentation se compose d'une forme linéaire u séparant les points solutions, son polynôme minimal q dans $k[T]$ et n polynômes $w_1,\ldots,w_n$ de $k[T]$ correspondant aux coordonnées des points solutions :

$$q(u) = 0, \qquad \left\{ \begin{array}{rcl} q'(u)x_1 & = & w_1(u), \\ & \vdots & \\ q'(u)x_n & = & w_n(u). \end{array} \right.$$

Une composante algébrique solution de dimension positive r est codée par une représentation de Kronecker d'un ensemble de points formés par l'intersection transverse de cette composante avec une variété linéaire affine de dimension $n-r$. Le nombre de points de cette intersection est le *degré géométrique* de l'ensemble solution. Ce codage est bon dans le sens où n'importe quelle information sur cet ensemble solution peut être retrouvée rapidement à condition de ne pas oublier le système de départ. Cette représentation s'appelle une *fibre de remontée*.

L'aboutissement de cette thèse est un nouvel algorithme pour résoudre les systèmes algébriques. Avant de pouvoir énoncer nos résultats de complexité il nous faut préciser le codage des entrées, des sorties et le modèle de complexité. Le système est composé de s équations polynomiales $f_1 = \cdots = f_s = 0$ et une inéquation polynomiale $g \neq 0$ en n variables $x_1,\ldots,x_n$ et à coefficients dans un corps k.

Complexité de l'entrée : les polynômes d'entrée $f_1,\ldots,f_n$ et g sont vus comme des fonctions polynomiales et sont codés par un *calcul d'évaluation* (*straight-line program* en anglais) de taille borné par L. On note d un majorant du degré des polynômes f_i.

Complexité intrinsèque : notre algorithme résout le système progressivement, en commençant par la première équation, puis ajoute la seconde, la troisième etc. Le nombre de solutions de ces problèmes intermédiaires, en terme de degré géométrique, gouverne la complexité globale. Nous notons $\mathcal{V}_i$ la variété algébrique clôture de Zariski de l'ensemble des solutions du système intermédiaire $f_1 = \cdots = f_i = 0$, $g \neq 0$. Pour une composante $\mathcal{W}$ irréductible de $\mathcal{V}_i$ nous notons $\deg(\mathcal{W})$ le degré géométrique classique de $\mathcal{W}$, $\mathrm{mul}_{f_1,\ldots,f_i}(\mathcal{W})$ la multiplicité de $\mathcal{W}$ comme solution du système $f_1 = \cdots = f_i = 0$ et enfin $\deg^a_{f_1,\ldots,f_i}(\mathcal{W})$, le *degré algébrique* de $\mathcal{W}$, représente le produit $\mathrm{mul}_{f_1,\ldots,f_i}(\mathcal{W})\deg(\mathcal{W})$. La grandeur δ_i sera la somme des degrés algébriques des composantes irréductible de $\mathcal{V}_i$ et δ^a le maximum des δ^a_i, pour $i = 1,\ldots,s$.

Comme modèle de complexité nous comptons le nombre d'opérations effectuées dans le corps de base k : additions, multiplications, divisions et tests d'égalité coûtent $\mathcal{O}(1)$. La fonction $\mathcal{U}(a)$ représente $a\log(a)^2\log(\log(a))$ et Ω est une constante (provenant de l'algèbre linéaire sur un anneau) inférieure à 4. Pour simplifer la présentation nous supposons que $\Omega \geq 3$. Nous pouvons maintenant énoncer le résultat principal de cette thèse.

Théorème A *Soit k un corps de caractéristique zéro, $f_1,\ldots,f_s$ et g des fonctions polynomiales de $k[x_1,\ldots,x_n]$ données par un calcul d'évaluation de taille borné par L. Il existe un algorithme probabiliste qui calcule la décomposition équidimensionnelle de la clôture de Zariski de l'ensemble des zéros du système*

$$f_1 = \cdots = f_s = 0, \qquad g \neq 0$$

avec une complexité, en cas de succès, dans

$$\mathcal{O}\Big(s\log(d)n^4(nL+n^{\Omega})\mathcal{U}(d\delta^a)^3\Big).$$

La probabilité de succès repose sur des choix d'éléments de k. Les mauvais choix sont contenus dans des fermés algébriques stricts.

I.2 Résultats Antérieurs

L'algorithme de cette thèse est essentiellement nouveau mais puise son inspiration dans les travaux du début des années 1990 de Giusti, Heintz et leurs collaborateurs.

Plaçons nous tout d'abord dans une situation simplifiée. Étant donnés n polynômes $f_1,\ldots,f_n$ et g à coefficients entiers en les indéterminées $x_1,\ldots,x_n$, on cherche à résoudre le système d'équations et d'inéquations

$$(S) \qquad f_1=\cdots=f_n=0, \quad g\neq 0.$$

Nous notons $\mathcal{V}_i$ la $i$ème *variété intermédiaire* :

$$\mathcal{V}_i := \overline{\mathcal{V}(f_1,\ldots,f_i)\backslash\mathcal{V}(g)},$$

c'est la réunion des composantes irréductibles solutions de $f_1=\cdots=f_i=0$ qui ne sont pas incluses dans l'hypersurface définie par $g=0$. La variété $\mathcal{V}_n$ représente la clôture de Zariski de l'ensemble des solutions du système (S).

Résumé dans le Théorème B ci-dessous, Giusti, Heintz et leurs collaborateurs proposent au début des années 1990 un nouvel algorithme itératif basé sur les résolutions successives des variétés intermédiaires $\mathcal{V}_i$. Leur algorithme fonctionne sous deux hypothèses :

(H_1) *Hypothèse de régularité :* chaque $\mathcal{V}_i$ est équidimensionnelle de codimension i.

(H_2) *Hypothèse de réduction :* la matrice jacobienne formée par les $f_1,\ldots,f_i$ est de rang plein sur $\mathcal{V}_i$.

Dans les calculs intermédiaires chaque $\mathcal{V}_i$ est représentée par une *résolution géométrique.* Une résolution géométrique d'une variété équidimensionnelle $\mathcal{W}$ de dimension r est la donnée de :

- Une matrice $n\times n$ inversible M à entrées dans $\mathbb{Q}$ de telle sorte que les coordonnées y définies par $y:=M^{-1}x$ soient en *position de Noether projective* pour $\mathcal{W}$. Cela signifie géométriquement que le morphisme de projection π

$$\begin{array}{rccc}\pi: & \mathcal{W} & \longrightarrow & \mathbb{C}^r\\ & (y_1,\ldots,y_n) & \longmapsto & (y_1,\ldots,y_r),\end{array}$$

  est surjectif, fini et de degré *exactement* le degré $\deg(\mathcal{W})$ de la variété $\mathcal{W}$.
- Une forme linéaire $u:=\lambda_{r+1}y_{r+1}+\cdots+\lambda_n y_n$, à coefficients λ_i dans $\mathbb{Q}$, séparant les composantes irréductibles de $\mathcal{W}$. Si l'on note $R:=\mathbb{Q}[y_1,\ldots,y_r]$ et K le corps des fractions de R, alors les puissances $1,u,u^2,\ldots,u^{\deg(\mathcal{W})-1}$ forment une base de $B':=K\otimes\mathbb{Q}[\mathcal{W}]$ vu comme K-espace vectoriel.

- Le polynôme minimal q de u dans B'.
- Les paramétrisations des coordonnées de $\mathcal{W}$:

$$y_{r+1} = v_{r+1}(T), \ldots, y_n = v_n(T),$$

  où les v_i appartiennent à $K[T]$ et ont un degré en T strictement inférieur à $\deg(\mathcal{W})$.

D'un point de vue algorithmique, les polynômes intervenant dans une résolution géométrique sont considérés comme développés dans la base monomiale en la variable principale T et leurs coefficients sont stockés par des calculs d'évaluation.

Pour les mesures de complexité nous adoptons les notations suivantes :

Complexité de l'entrée :

- d est un majorant du degré des polynômes d'entrée ;
- h un majorant de leur hauteur arithmétique (la « taille » des coefficients) ;
- L est la taille du calcul d'évaluation par lequel ils sont donnés.

Complexité intrinsèque :

- δ est le maximum des degrés des variétés intermédiaires $\mathcal{V}_i$, i allant de 1 à n ;
- η est (essentiellement) le maximum des hauteurs des polynômes intervenant dans la résolution géométrique du système.

On a alors le théorème suivant [GHH+97, GHMP97, GHM+98] :

Théorème B [Giusti, Hägele, Heintz, Montaña, Morais, Morgenstern, Pardo] *Il existe une machine de Turing qui calcule une résolution géométrique du système (S) satisfaisant les hypothèses (H_1) et (H_2) en temps (nombre de bits manipulés) polynomial en $ndhL\delta\eta$, de manière probabiliste avec une erreur bornée. De plus, si l'on étend cette représentation en évaluation aux entiers, la dépendance en η disparaît.*

I.3 Contributions

C'est sur la base des travaux menant au Théorème B ci-dessus que j'ai débuté mon travail de thèse. Je présente maintenant chronologiquement mes contributions.

I.3.1 Déforestation

J'ai commencé ma thèse par une *étude de faisabilité* : j'ai implanté dans le système de calcul formel Maple l'algorithme de calcul de la dimension d'une variété projective proposé par Giusti et Heintz [GH93]. Il s'agit d'un algorithme simple manipulant des calculs d'évaluation. Ce travail a été mené avec Giusti, Hägele, Marchand et Salvy. Le résultat est publié dans [GHL+00] et fait l'objet de l'Annexe A. En faisant le lien avec une méthode issue de l'informatique théorique, appelée *déforestation*, nous montrons que, pour la classe des algorithmes d'élimination nous concernant, l'usage des calculs d'évaluation peut et doit être évité dans les calculs intermédiaires, sans perte d'efficacité théorique, au moyen d'une transformation syntaxique sur le code de l'algorithme. Dans le code transformé, seuls les polynômes d'entrée sont encore stockés par évaluation.

Nous avons ensuite entrepris un prototype en Maple de l'algorithme de résolution sous-jacent au Théorème B. Nos techniques de déforestation nous ont permis de supprimer tous les calculs d'évaluation intervenant dans les calculs intermédiaires. Seules les équations d'entrée sont données en évaluation, les structures de données utilisées ensuite sont les polynômes à une ou deux variables et à coefficients dans $\mathbb{Z}/p\mathbb{Z}$ (où p est un « petit » nombre premier), sur lesquels existe une algorithmique efficace. Seulement, les algorithmes menant au Théorème B avaient une complexité peu raffinée, l'exposant intervenant dans celle-ci était de l'ordre de 8. Nos résultats pratiques étaient alors fort peu satisfaisants.

I.3.2 Cas régulier réduit

Nous avons fait des pas importants dans l'amélioration de la complexité de l'algorithme de résolution. Géométriquement nous avons entériné la *fibre de remontée* comme la « bonne » représentation pratique des variétés : rappelons que celle-ci a été introduite dans les travaux conduisant au Théorème B comme une étape permettant de « comprimer » les résolutions géométriques des variétés $\mathcal{V}_i$. Cette compression était le point crucial pour maintenir la taille des résolutions géométriques successives polynomiale en la quantité δ.

En utilisant les notations ci-dessus, une fibre de remontée pour la variété $\mathcal{W}$ est une résolution géométrique de la variété zéro-dimensionnelle $\mathcal{W}_P := \mathcal{W} \cap \mathcal{V}(y_1 = P_1, \ldots, y_r = P_r) = \pi^{-1}(P)$, où $P := (P_1, \ldots, P_r)$ est un point de $\mathbb{Q}^r$ choisi suffisamment générique de sorte que le cardinal de $\mathcal{W}_P$ soit $\deg(\mathcal{W})$ et que π soit lisse en chaque point de $\mathcal{W}_P$. Le point P est appelé le *point de remontée*.

Notre algorithme calcule alors une fibre de remontée pour $\mathcal{V}_{i+1}$ à partir d'une fibre de $\mathcal{V}_i$, pour i de 1 à n par la méthode des *courbes de remontée* ; c'est *l'étape incrémentale* constituée des trois algorithmes :

- *Remontée :* à partir d'une fibre de remontée de $\mathcal{V}_i$, on choisit un point P' dans $\mathbb{Q}^r$ différent de P et on calcule une résolution géométrique de la variété image réciproque de la droite (PP') par π. Il s'agit d'une courbe tracée sur $\mathcal{V}_i$, appelée une *courbe de remontée.* Cette partie du calcul repose essentiellement sur une variante de la méthode de Newton. L'hypothèse de réduction (H_2) est cruciale.
- *Intersection :* on coupe cette courbe avec l'hypersurface définie par l'équation suivante $f_{i+1} = 0$: dans ce contexte notre méthode est nouvelle et, dans un sens, optimale. Sous l'hypothèse (H_1) cette intersection est un nombre fini de points.
- *Nettoyage :* la description de la fibre de remontée de $\mathcal{V}_{i+1}$ est enfin obtenue de la précédente intersection en supprimant les points contenus dans l'hypersurface $g = 0$.

Pour représenter nos courbes de remontée nous utilisons une représentation de Kronecker des paramétrisations (appelée aussi « représentation univariée rationnelle »). Dans le cadre de la définition de la résolution géométrique de $\mathcal{W}$ ci-dessus, il s'agit d'une paramétrisation de la variété $\mathcal{W}$ sous la forme :

$$q(u) = 0, \quad \left\{ \begin{array}{rcl} \frac{\partial q}{\partial T}(u) y_{r+1} & = & w_{r+1}(u), \\ \cdots & & \\ \frac{\partial q}{\partial T}(u) y_n & = & w_n(u). \end{array} \right.$$

L'intérêt de cette représentation réside en ce que les degrés totaux des polynômes q et w_i sont bornés par $\deg(\mathcal{W})$.

Finalement, notre résultat de complexité dans ce cadre particulier est le suivant [GLS01] :

Théorème C [Giusti, Lecerf, Salvy] *Si k est un corps de caractéristique zéro et $f_1, \ldots, f_n$, g sont $n+1$ polynômes de $k[x_1, \ldots, x_n]$ de degré au plus d, donnés par un calcul d'évaluation de complexité séquentielle L et satisfaisant les hypothèses (H_1) et (H_2), alors nous disposons d'un algorithme probabiliste calculant une résolution géométrique de $\overline{\mathcal{V}(f_1, \ldots, f_n) \setminus \mathcal{V}(g)}$ en $\mathcal{O}(n(nL + n^4)\mathcal{U}(d\delta)^2)$ opérations arithmétiques sur le corps k. La probabilité de succès repose sur le choix d'éléments de k, les mauvais choix sont contenus dans un fermé algébrique strict.*

Lorsque le corps de base est le corps des nombres rationnels, la résolution s'effectue d'abord modulo un nombre premier de petite taille en conservant la complexité ci-dessus, puis la résolution modulaire est remontée en une résolution rationnelle en $\mathcal{O}((nL + n^\Omega)\mathcal{U}(D)\mathcal{U}(\eta))$ opérations en bits, où D représente le nombre de solutions du système et η la taille des entiers de la sortie. Cette dernière partie peut donc être considérée comme *optimale* vis-à-vis de la taille de la sortie.

J'ai réalisé un *prototype* en Magma, appelé Kronecker. Une fois stabilisé et documenté [Lec99a, Lec99b], mon code a été rendu disponible. Je l'ai présenté au mois de juillet 1999 à la conférence internationale FoCM'99. Les performances de mon implantation étaient comparables sur de nombreux exemples aux meilleurs logiciels de calculs de bases de Gröbner disponibles. Les résultats de [GHL+00] se trouvaient ainsi confirmés et notre approche validée.
Ces résultats ont fait l'objet de la publication [GLS01] et sont au cœur du Chapitre II.

I.3.3 Décomposition équidimensionnelle

À partir de septembre 1999, j'ai porté mes efforts dans deux directions, théorique et pratique, pour étendre la méthode aux cas ne satisfaisant pas les hypothèses de régularité et de réduction (H_1) et (H_2).

J'ai proposé un algorithme théorique pour le calcul de la décomposition en composantes équidimensionnelles d'une variété algébrique décrite par un nombre fini d'équations polynomiales. Ce résultat généralise le Théorème B en levant les hypothèses sur les équations. La complexité que j'obtiens est polynomiale dans le nombre d'équations, la taille du calcul d'évaluation en entrée, le maximum des degrés des équations et un degré intrinsèque généralisant la quantité δ. Dans le pire des cas cette complexité est polynomiale en d^n. La meilleure borne connue auparavant était exponentielle (polynomiale en d^{n^2}).

L'idée du nouvel algorithme théorique repose sur le premier théorème de Bertini : les équations d'origine sont remplacées par des combinaisons linéaires génériques $g_i :=$

$t_{i,1}f_1 + \cdots + t_{i,s}f_s$ pour i de 1 à $n+1$. Les variétés intermédiaires $\mathcal{V}_i$ deviennent $\mathcal{V}_i := \mathcal{V}(g_1, \ldots, g_i)$, l'algorithme reste essentiellement itératif, la grandeur δ est alors le maximum des degrés géométriques des variétés intermédiaires. Dans le théorème suivant le modèle de complexité non-uniforme permet de tester si un calcul d'évaluation représente le polynôme nul avec une complexité polynomiale dans la taille du calcul.

Théorème D *Soient $f_1, \ldots, f_s$ des polynômes de $k[x_1, \ldots, x_n]$ avec k un corps de caractéristique nulle. Il existe un sous ensemble dense G de $k^{(n+1)s}$ et un algorithme déterministe tels que : pour tout $(t)_{i,j}$ dans G, l'algorithme calcule une décomposition équidimensionnelle du système $f_1 = \cdots = f_s = 0$ avec une complexité non-uniforme polynomiale en $Lnsd\delta$, en terme du nombre d'opérations arithmétiques sur le corps de base k.*

Si α est un entier positif et si E est un ensemble de points de taille $3\alpha(d+1)^{3n}$ dans k, un point $(t)_{i,j}$ choisi aléatoirement dans $E^{(n+1)s}$ est dans G avec une probabilité de succès au moins $1 - 2(n+1)/\alpha$.

D'un point de vue pratique, j'ai distribué en avril 2000 une deuxième version du logiciel Kronecker mettant en place l'algorithme fonctionnant sans hypothèse restrictive sur le système d'entrée. Cette version était alors nettement supérieure à tous les logiciels communément disponibles dans les systèmes de calcul formel généralistes tels que Maple, Mathematica, Axiom... et offre des performances très intéressantes sur de nombreux exemples par rapport à des logiciels spécialisés tels que Singular, Magma et même Gb de Faugère et RealSolving de Rouillier. Les composantes de dimension positive solutions du système sont aussi codées par des fibres de remontée. Deux possibilités sont alors offertes à l'utilisateur :

- Une *fonction de passage* permet de calculer n'importe quelle autre fibre $\pi^{-1}(P')$; dotée de cette fonctionnalité notre sortie est donc parfaitement équivalente à l'idée théorique.
- Une *fonction d'écriture* qui permet de calculer efficacement une résolution géométrique codée par des polynômes à plusieurs variables en représentation classique (creuse). Ce travail a été effectué par Schost [Sch00b, Sch00a]. Pour obtenir un algorithme rapide (linéaire dans la taille de la sortie) nous avons dû concevoir un algorithme de multiplication rapide de séries à plusieurs variables. Ce travail est présenté dans l'Annexe C.

Ces résultats ont été publiés dans [Lec00] et ont fait l'objet d'un exposé à la conférence internationale ISSAC'2000. Cette article constitue l'Annexe B.

I.3.4 Opérateur de Newton quadratique en présence de multiplicité

L'algorithme sous-jacent au Théorème D présente l'inconvénient de devoir remplacer les équations de départ par des combinaisons linéaires génériques de celles-ci : la complexité d'évaluation d'une seule équation devient celle de tout le système d'entrée. Ceci accroît la complexité pratique de la méthode. De plus, la structure même de l'algorithme une fois déforesté est très compliquée.

Pour pallier ces inconvénients je propose une généralisation de l'opérateur de Newton adaptée aux situations singulières, afin de s'affranchir de l'hypothèse (H_2) de réduction.

Le nouvel algorithme de résolution en découlant est bien plus naturel, élégant et efficace.

Le point crucial est donc maintenant le suivant : si $\mathcal{W}$ est une composante multiple de $\mathcal{V}_i$ donnée par une fibre de remontée alors l'opérateur de Newton classique servant à produire une courbe de remontée ne s'applique plus. Je *généralise* cet opérateur de Newton et montre que le surcoût est polynomial en la multiplicité.

Ce problème est le cœur du Chapitre III. Voici une présentation sommaire de notre résultat. Si x^* est une racine isolée de multiplicité M d'un système $f_1 = \cdots = f_s = 0$, où les f_i sont des fonctions polynomiales données par un calcul d'évaluation de complexité L et de degré total au plus d, alors il existe un voisinage V de x^* et un algorithme N tels que si $z \in V$ alors la convergence de la suite des itérés $N^j(z)$ est quadratique vers x^* et la complexité de N est dans

$$\mathcal{O}\Big(\log(d) n^4 (nL + n^{\Omega}) M^2 \Big).$$

Notre étude se place dans le cadre d'une topologie $\mathfrak{m}$-adique pour lequel le voisinage V est $x^* + \mathfrak{m}$.

L'affranchissement de l'hypothèse (H_1) est relativement aisé et la présentation de l'algorithme complet menant au Théorème A se trouve dans le Chapitre IV.

I.3.5 Implantation

Nos algorithmes sont implantés dans le système de calcul formel Magma [BC89, BC90, BCM94, BC95, CP96, BCP97]. Le passage d'estimations théoriques de complexité à une implantation que l'on souhaite efficace est rarement une mince affaire. Il est important de souligner que notre méthodologie à toujours fait précéder la pratique à la théorie. Les algorithmes que nous donnons sont tous parfaitement réalistes. Malgré tout, les constantes cachées dans les $\mathcal{O}$ et les seuils de validité des algorithmes asymptotiquement rapides sous-jacents à nos estimations de complexité sont pour certains d'entre-eux trop élevés par rapport à la gamme des problèmes que l'on peut traiter à l'heure actuelle. Nous avons donc développé des algorithmes alternatifs. Ceux-ci font l'objet du Chapitre V.

Durant mon séjour d'un mois à l'université de Sydney au sein de l'équipe de Cannon qui développe le logiciel Magma (du 15 août au 15 septembre 2000), des progrès ont été réalisés dans plusieurs directions : j'ai contribué à l'élaboration d'un domaine de calculs d'évaluation et à l'amélioration de l'arithmétique des polynômes à deux variables ; de plus, j'ai pu analyser en détail le temps passé dans chaque fonction de mon code Kronecker et réaliser des améliorations, éclairé par les conseils des développeurs de Magma. Sur des exemples de taille moyenne, le gain de temps obtenu par rapport à la version 0.16 de Kronecker est entre 2 et 5.

I.3.6 Applications

D'un point de vue *applicatif*, je me suis essentiellement concentré sur les systèmes référencés afin de mettre au point mon logiciel. Mes dernières performances ont permis de traiter le système connu sous le nom de cyclique 8 en 30 minutes et cyclique 9 en 5

jours, en utilisant les machines les plus puissantes de l'UMS MEDICIS. Ces performances étaient jusque là l'apanage exclusif du logiciel FGb de Faugère.
Des applications de mes méthodes ont été développées par Schost, Lehmann et Waissbein sur la base de Kronecker :

- Schost a pu répondre à des questions pratiques de cryptographie, en collaboration avec Gaudry [GS00, Gau00, Sch00b] : études de courbes elliptiques et hyperelliptiques, construction d'équations modulaires et calcul du polynôme minimal des j-invariants des courbes elliptiques quotients d'une courbe à Jacobienne décomposable.
- En collaboration avec Rouillier et Safey el Din, Schost a effectué des calculs de géométrie réelle sur le problème d'interpolation de Birkhoff [RSS01, Saf01, Sch00b].
- Les méthodes de géométrie réelle du groupe de travail TERA sont implantées par Lehmann. Les résultats récents, en collaboration avec Waissbein, concernant des optimisations d'ondelettes utilisées pour de la compression d'image sont très encourageants [BB01].
- Aussi, dans le thème de la compression d'image, nous avons résolu avec Schost des systèmes de difficulté élevée, proposés par Mallat (CMAP, École polytechnique).

I.3.7 Publications

Voici un résumé de l'ensemble de mes publications et productions logicielles.

Publications avec comité de lecture anonyme

M. Giusti, K. Hägele, G. Lecerf, J. Marchand, et B. Salvy, *The Projective Noether Maple Package: Computing the Dimension of a Projective Variety.* Journal of Symbolic Computation, pp. 291–307, vol. 30, n°3, septembre 2000.

M. Giusti, G. Lecerf et B. Salvy, *A Gröbner Free Alternative for Polynomial System Solving.* Journal of Complexity, vol. 17, n°1, pp. 154–211, mars 2001.

G. Lecerf, *Computing an Equidimensional Decomposition of an Algebraic Variety by means of Geometric Resolutions.* Actes de la conférence ISSAC 2000 (ACM), pp. 209–216, 2000.

G. Lecerf, *Quadratic Newton Iterator for Systems with Multiplicity.* Manuscrit de 40 pages, avril 2001. À paraître dans Journal of FoCM.

Prépublication

G. Lecerf et É. Schost, *Fast Multivariate Power Series Multiplication in Characteristic Zero.* Manuscrit de 6 pages, avril 2001.

Logiciels

- *The Projective Noether Package.* Paquetage pour Maple offrant diverses stratégies de calcul de la dimension d'une variété projective, 1997. Réalisé au laboratoire GAGE et disponible avec sa documentation à
http://tera.medicis.polytechnique.fr/pub/tera/soft/pnp/

- *Kronecker*, un paquetage pour Magma servant à résoudre des systèmes d'équations polynomiales. Réalisé au laboratoire GAGE et disponible avec sa documentation à http://kronecker.medicis.polytechnique.fr/ :
  - version 0.1, cas régulier réduit, juillet, 1999.
  - version 0.16, cas général, décomposition équidimensionnelle, avril 2000. Contributions d'É. Schost.
  - version 0.166, algorithme de déflation rapide, décomposition équidimensionnelle, en cours. Contributions de L. Lehmann.

I.4 Structure de la thèse

Le Chapitre II de cette thèse traite le cas particulier de la résolution d'une suite régulière réduite menant au Théorème C. Nous détaillons précisément les aspects qui détachent notre travail des méthodes historiques sous-jacentes au Théorème B. Aussi nous donnons une vision d'ensemble simplifiée des calculs effectués par l'algorithme. Nous exhibons quelques temps de calcul et comparaisons de notre programme Kronecker dans sa version 0.1 avec les logiciels qui étaient les plus compétitifs à ce moment là. Le Chapitre III est consacré à la généralisation de l'opérateur de Newton, clef de voûte pour étendre les résultats du Chapitre II. Cette extension est traitée dans le Chapitre IV. Le Théorème A y est prouvé. Le Chapitre V discute de l'implantation en Magma de l'algorithme et donne quelques performances et points de comparaison.

Les annexes A et B sont des travaux publiés. La première introduit la notion de *déforestation* pour les algorithmes d'élimination en calcul formel et la seconde est la preuve du Théorème D. Enfin, la dernière annexe est un résultat technique concernant la multiplication des séries à plusieurs variables : nous montrons que, tronquée en degré total, la multiplication des séries a une complexité linéaire à des facteurs logarithmiques près.

I.5 Perspective historique

Afin de mettre en lumière les caractéristiques nouvelles de notre algorithme nous présentons une vue d'ensemble des problématiques liées à la résolution pratique des systèmes algébriques.

Depuis l'aube des mathématiques de nombreuses méthodes ont été proposées pour obtenir ou bien des solutions approchées, dans le cadre de l'*analyse numérique*, ou bien des solutions exactes, dans le cadre du *calcul formel*. Un précurseur des méthodes formelles modernes est certainement Léopold Kronecker à la fin du XIXème siècle. Donner une vue d'ensemble des méthodes de résolution de systèmes polynomiaux n'est pas un exercice facile. On doit tenir compte de plusieurs aspects : la représentation du système d'entrée, la représentation de la sortie et le modèle de complexité (incluant le caractère probabiliste ou déterministe). Historiquement un certain nombre de ces méthodes apparaissent pour calculer les points isolés solutions d'un système puis sont généralisées pour calculer toutes les solutions (c'est le cas de notre méthode). De ce fait, nous ne distinguerons pas les algorithmes spécialisés des algorithmes généralistes.

Commençons par examiner les différents codages usuels pour les polynômes à plusieurs variables donnés en entrée. On distingue les polynômes en *écriture* de ceux en *évaluation*. Dans le premier cas les polynômes sont vus comme vecteurs de leurs coefficients. On peut alors décider de stoker tous les coefficients, c'est une représentation *dense* ou bien ne stocker que ceux qui ne sont pas nuls, c'est une représentation *creuse*. La représentation par évaluation consiste à stocker un programme permettant d'évaluer le ou les polynômes en un point donné (si cette évaluation a un sens). Un stockage par évaluation semble une représentation naturelle pour les méthodes numériques tandis qu'une représentation en écriture semble relever du calcul formel. Notre algorithme tend à montrer que cette distinction n'a pas lieu d'être. Concernant la complexité de l'entrée on notera d une borne sur le degré des polynômes, n le nombre de variables et L la complexité d'évaluation.

Abordons le problème crucial de la représentation des sorties. En toute généralité, par solution d'un système algébrique il faut entendre un couple constitué d'une description mathématique (sémantique) et d'une représentation informatique (syntaxique). Cette première doit contenir toute l'information sur l'ensemble des solutions au sens géométrique ou bien algébrique. La représentation informatique doit permettre la manipulation et la réponse aisée à des questions concernant l'ensemble des solutions. Pour des systèmes à coefficients des nombres entiers, les méthodes numériques fournissent des ensembles de points approchant les solutions dans le corps des nombres complexes. Les méthodes formelles calculent soit des polynômes appartenant à l'idéal de départ soit des polynômes minimaux correspondant aux extensions algébriques engendrées par les solutions.

Du point de vue de la complexité il est important de souligner l'aspect suivant. La notion de solution isolée (de dimension zéro) est un concept qui nous apparaît relativement clair à l'heure actuelle. Une description par solution numérique approchée avec suffisamment de précision (dans un bassin d'attraction de l'opérateur de Newton par exemple) est *équivalente* à une description en terme d'élément primitif et d'extension de corps. Nous reviendrons sur ce point plus loin. Dans ce cas la complexité connue dans le pire des cas est polynomiale en d^n (et donc dans la taille de l'entrée, si n est fixé). Pour les ensembles solutions de dimension positive le problème est de tout autre difficulté. Une représentation en écriture de polynômes éliminant pour décrire un tel ensemble conduit inéluctablement à une complexité au moins $d^{\mathcal{O}(n^2)}$. En effet considérons un système avec m équations ($m < n$) à coefficients génériques de degré exactement d. Un polynôme provenant de l'élimination de m variables est de degré au moins d^m et contient donc au moins $\binom{d^m+n-m}{n-m}$ monômes. En fixant n et en prenant pour m la partie entière de $n/2$, ce nombre combinatoire conduit ainsi à une estimation en $d^{\mathcal{O}(n^2)}$.

Si l'on souhaite des algorithmes généralistes de complexité polynomiale en d^n dans le pire des cas il faut donc trouver une alternative à la représentation en écriture. Comme le montre notre algorithme la représentation par évaluation est une bonne réponse. L'idée mathématique géométrique sous-jacente est fort simple : c'est la notion de *point générique* d'une variété irréductible. La donnée d'un point suffisamment générique sur une composante de dimension positive est suffisante pour retrouver toute l'information nécessaire sur cette composante, si peu que l'on se souvienne du système d'entrée. Cette idée est le pendant non-archimédien de la notion de bassin d'attraction pour l'opérateur de Newton : les points génériques s'obtiennent comme ensembles de solutions isolées du système après spécialisation de certaines variables dans des coordonnées génériques. La valuation est celle des séries en les variables spécialisées autour des points solutions. Et l'opérateur de Newton permet de calculer un développement en série entière à précision arbitraire des composantes solutions dans un voisinage d'un point générique. Encore faut-il disposer d'un opérateur de Newton pour faire face aux composantes multiples ! La réponse que je fournis à ce problème est la contribution clef de ma thèse et donne toute la cohérence à mon approche.

Il arrive que le passage d'une estimation théorique de complexité à un programme efficace représente un véritable défi. Réciproquement des algorithmes de complexités moins bonnes en théorie, ou même sans estimation très précise connue, peuvent se révéler des outils très efficaces. La grande variété des algorithmes de résolution et des implantations laissent à penser qu'il existe toujours un exemple pour lequel un programme de résolution arrivera plus vite à la réponse qu'un autre programme. Il est donc important de tester les programmes sur un grand nombre d'exemples afin de pouvoir en dégager une plage d'utilisation optimale. Les besoins de tels programmes sont réels et les applications nombreuses. Lorsque l'on développe un programme de résolution on observe souvent que chaque problème concret, poussant souvent les ressources à l'extrême, est l'occasion d'optimisations spécifiques et de mise en place d'heuristiques. Les effets finissent par cacher les performances réelles et la course à la programmation domine quelquefois la réflexion. Nous avons pensé notre code sans optimisation ni heuristique, sa modularité permet sur une application fixée de faire un grand nombre de réglages. Cette philosophie est rendue possible par l'environnement convivial du système de calcul formel généraliste Magma. Un tel confort est rarement possible lorsqu'il s'agit de programmes autonomes écrits en langage de bas niveau comme C.

Venons en enfin à la zoologie des différentes méthodes de résolution. Je distinguerai le cadre archimédien, propre aux méthodes numériques approchées, du cadre non-archimédien, lié au calcul exact. Aussi, il faut faire une place aux méthodes de *conversion* d'un codage vers un autre.

I.5.1 Méthodes numériques

Situation à une variable. Chercher des approximations des racines complexes d'un polynôme à coefficients entiers est un problème ancien. Il faut néanmoins attendre les années 1990 pour avoir des algorithmes asymptotiquement optimaux (à des facteurs logarithmiques près). Principalement les meilleures méthodes connues sont dues à Schönhage [Sch82, Sch87] et Pan [Pan95, Pan96]. Depuis les travaux historiques de Gauss de la preuve du théorème fondamental de l'algèbre par un argument d'homotopie, les méthodes par déformation sont nombreuses, offrent un véritable intérêt pratique, et sont toujours d'actualité [Sma86]. Citons parmi les travaux récents ceux de Yakoubsohn [Yak95, Yak00]. L'opérateur de Newton est au cœur d'un grand nombre d'algorithmes, soulignons les récentes avancées de Hubbard, Kim, Schleicher et Sutherland [KS94, HSS00].

Situation à plusieurs variables. Dans cette situation nous ne connaissons pas d'algorithmes optimaux. Les principales méthodes fonctionnent par homotopie. Les premiers programmes remontent à la fin des années 1970 avec les travaux de Drexler [Dre77] et Garcia & Zangwill [GZ79]. Chercher une bonne homotopie (en particulier un bon système de départ) est un problème difficile. La théorie des zéros approchés de Smale [Sma81, Sma86] et ses collaborateurs offre un cadre de travail effectif en terme de machine de Turing réelles et conduit à des méthodes robustes. Le théorème de Bernshtein [Ber75] contient implicitement une méthode d'homotopie pour les systèmes creux, optimale si les coefficients des polynômes sont suffisamment génériques. Cette homotopie a été raffinée et mise en pratique par Verschelde [Ver96] et ses collaborateurs. Le logiciel de Verschelde PHC [Ver99] écrit en Ada offre des performances inégalées. À l'origine conçues pour les systèmes de dimension zéro torique, de récentes extensions [SV00, SVW01a] permettent le calcul de la décomposition en composantes équidimensionnelles et même irréductibles. Le codage des composantes de dimension positive se fait par un ou plusieurs points génériques approchés. Cette sortie est donc très proche de la nôtre. Seulement, à défaut d'utiliser un opérateur de Newton pour les composantes multiples, des polynômes éliminant à plusieurs variables sont interpolés pour répondre aux tests d'inclusion prévenant les redondances dans la sortie.

I.5.2 Méthodes formelles

Nous distinguerons les méthodes *classiques* en écriture, de nos méthodes en évaluation.

Méthodes par écriture. Sous ce terme je tente de regrouper les algorithmes de calcul formel manipulant les polynômes en écriture.

On attribue communément la notion d'*ensemble caractéristique*, très proche de celle d'*ensemble triangulaire*, à Ritt [Rit32, Rit66] dans le contexte de l'algèbre différentielle. Ensuite plusieurs approches ont été proposées. Chacune d'entre elles correspond à des propriétés spécifiques des ensembles et aussi à la façon de les calculer. Les principales écoles sont celles de Wu [Wu86], Lazard [Laz91, Laz92], Kalkbrener [Kal91], Wang [Wan93], Duval et ses collaborateurs [DDD85, Duv94, Día94]. Des travaux de comparaison et d'unification ont été entrepris par Aubry, Lazard, Moreno Maza [ALM99] ainsi que par Dellière [Del99]. Les complexités de ces méthodes sont souvent mal connues.

Les *bases standard* introduites par Hironaka en 1964 sont apparues en 1965 dans le domaine du calcul formel par l'algorithme de Buchberger, sous le nom de *bases de Gröbner*. Il serait trop long de vouloir retracer les questions et résultats qui lui sont rattachés. Nous renvoyons les lecteurs à des ouvrages de synthèses, tels que [BW93]. Les bases de Gröbner sont désormais un outil vital dans tous les systèmes de calcul formel actuels. Bayer et Stillman [BS88] ont montré que l'ordre *grevlex* est « le plus efficace ». Les bornes supérieures de complexité connues reflètent mal le comportement pratique des implantations. D'un point de vue pratique, les bases de Gröbner de Magma sont les meilleures des logiciels diffusés. Mais les méthodes développées par Faugère, améliorant l'algorithme de Buchberger par une utilisation pertinente des techniques d'algèbre linéaire creuse et un codage fin en C, dépassent largement tous les autres logiciels [Fau99].

Depuis la construction du résultant par la matrice de Sylvester les méthodes basées sur l'algèbre linéaire et la construction de matrices de Macaulay [Mac16] ou de variantes foisonnent. C'est le cas des travaux d'Emiris, Mourrain et Pan [EM99b, BMP98, MP00]. Les constructions de matrices de type Macaulay sont coûteuses puisque les tailles de celles-ci sont des nombres combinatoires dont l'estimation en d^n cache des constantes exponentielles en n. Par exemple, la taille d'une matrice de Macaulay est le nombre de monômes en degré $nd - n + 1$ (régularité de Hilbert), c'est à dire $\binom{nd+1}{n} \in \mathcal{O}((ed)^n)$. Il faut noter que les récents travaux de Mourrain et Trébuchet conduisent à un véritable d^n [MT00] pour une intersection complète.

Le cas particulier des systèmes creux connaît un engouement certain. De la construction du résultant creux [GKZ94], aux algorithmes proposés par Canny & Emiris [CE93, EC95b] et Rojas [Roj94, Roj99, Roj00], les complexités en terme de volume mixte raffinent les bornes en d^n. Notons que le calcul du volume mixte ne semble plus désormais un facteur limitant [LL01].

Méthodes par évaluation. C'est ici qu'intervient l'originalité de notre travail. Comme pour les méthodes numériques notre algorithme bénéficie de la complexité d'évaluation du système. Par exemple, pour des problèmes issus de la mécanique, les équations sont souvent des produits scalaires et donc s'évaluent bien. Concernant la présence d'une inéquation $g \neq 0$ notre méthode est l'une des rares à en tirer efficacement parti. Pour l'algorithme de Burchberger il faut ajouter une nouvelle variable z et ajouter l'équation $zg = 1$. Notre algorithme dépend d'un degré δ^a, grandeur géométrique, et non combinatoire, qui compte les solutions avec multiplicité. Aucun facteur parasite exponentiel n'intervient. D'un point de vue pratique, notre méthode n'utilise que des polynômes en deux variables au plus dans le cas d'une suite régulière réduite et en trois variables dans le cas général (pour les tests l'inclusion).

I.5.3 Algorithmes de conversion

Pour le problème particulier du calcul de base de Gröbner on sait que l'ordre de Bayer et Stillman [BS88] est dans un certain sens optimal. Il est alors intéressant transformer une base pour cet ordre en une base pour un autre ordre (ordre d'élimination par exemple). Pour les systèmes de dimension zéro on peut citer les algorithmes FGLM [FGLM93], RUR (représentation univariée rationnelle) [Rou96, Rou99] et une variante plus rapide [BSS01]. En dimension positive l'algorithme Gröbner walk [CKM97] est implanté en Magma et affiche de bonnes performances.

En matière de résolution de systèmes polynomiaux la séparation entre calcul numérique et formel n'est pas aussi grande qu'on pourrait le croire ; des ponts existent. Il faut commencer par noter que certaines méthodes en calcul formel peuvent être exécutées avec des nombres flottants à précision fixée. C'est le cas par exemples des méthodes basées sur l'algèbre linéaire telles que celles de Mourrain & Pan. Par ailleurs les résolutions par éléments primitifs ramènent par une correspondance birationnelle les problèmes à plusieurs variables à une situation en une variable. Un grand intérêt de cette sortie est de pouvoir calculer aisément les solutions numériques approchées.

On peut croire que les algorithmes numériques sont plus rapides et ont une meilleure complexité que les algorithmes formels. Ceci semble un leurre : les programmes de Faugère affichent des performances inégalées par les numériciens et les travaux théoriques de Pardo et ses collaborateurs [CHMP01, CMPS01, Cas01] montrent que le nombre de décimales nécessaires pour représenter correctement un nombre algébrique solution (un zéro approchée dans la théorie de Smale) est polynomial en la hauteur du point et donc en la taille des entiers intervenant dans le polynôme minimal décrivant l'extension algébrique correspondante. D'un point de vue constructif le passage d'un nombre flottant algébrique à son polynôme minimal peut se faire avec LLL [LLL82]. Ces mêmes travaux fournissent aussi des éléments de comparaison précis entre l'homotopie de Smale et notre algorithme de résolution.

I.6 Perspectives

L'extension de notre algorithme de décomposition équidimensionnelle en décomposition irréductible se pose naturellement. Nous présentons quelques idées sur lesquelles nous travaillons. Du point de vue non-archimédien notre méthode de résolution n'est qu'une suite d'homotopies. Il est alors immédiat d'envisager d'adapter les homotopies utilisées par les numériciens à l'aide de nos outils non archimédiens et obtenir ainsi de nouveaux algorithmes de résolution formelle. Malheureusement il n'est pas clair à l'heure actuelle si cela peut mener à un algorithme de meilleure complexité que celui donné dans cette thèse. D'autant plus qu'il nous semble possible de faire reculer le goulot d'étranglement de notre algorithme.

I.6.1 Décomposition en composantes irréductibles

Nous abordons d'un point de vue pratique le calcul de la décomposition en composantes irréductibles dans le Chapitre V. En collaboration avec Pardo et San Martín (université de Cantabrie, Santander, Espagne), les études théoriques sont maintenant bien avancées mais de nombreux progrès restent à faire d'un point de vue algorithmique et pratique. Détaillons les deux approches que nous envisageons.

Le premier point de vue consiste à calculer, à la $i$ème étape de l'algorithme de résolution, une décomposition en composantes irréductibles de la variété $\mathcal{V}_i$. Les calculs avec les composantes irréductibles sont parallélisables, les degrés des variétés sont plus petits, les temps de calcul sont améliorés (c'est le cas des systèmes cyclique). La difficulté est de recombiner les fibres lors des étapes de remontée : en général les composantes irréductibles d'une fibre ne sont pas en bijection avec celles d'une courbe de remontée associée. Sur le corps $\mathbb{Z}/p\mathbb{Z}$ (où p est un nombre premier) nous montrons que l'approche naïve combinatoire a une complexité polynomiale en moyenne. Ce résultat n'est pas satisfaisant dans de nombreuses situations. Par exemple, si à l'issue de la résolution modulo un « petit » nombre premier p la sortie comporte plusieurs centaines de composantes irréductibles (modulo p) qui ne sont pas les spécialisations des composantes sur $\mathbb{Q}$, alors les méthodes combinatoires sont trop coûteuses. La solution que nous utilisons actuellement est basée sur la récente méthode de van Hoeij [Hoe00], qui utilise l'algorithme LLL. Nos résultats deviennent alors satisfaisants.

Le deuxième point de vue consiste à calculer pour chaque composante équidimensionnelle de $\mathcal{V}_i$ une seule composante irréductible et d'explorer l'arbre combinatoire menant à toutes les solutions du système. L'intérêt de cette approche est de fournir rapidement une seule solution du système puis d'autres à la demande de l'utilisateur. La difficulté maintenant est la suivante : étant donnée uniquement une composante irréductible d'une fibre de remontée comment calculer la composante irréductible d'une courbe de remontée associée ? Il s'agit d'un problème d'approximation algébrique. On peut alors utiliser des algorithmes de calculs d'approximants de Padé-Hermite ou bien encore l'agorithme LLL. La mise en œuvre directe de ces méthodes ne conduit pas à des complexités pratiques satisfaisantes lorsque l'on souhaite toutes les solutions du système. Néanmoins la nouvelle amélioration de LLL proposée par Koy et Schnorr [KS01a, KS01b] nous encourage à poursuivre dans cette direction.

I.6.2 Goulot d'étranglement

La complexité de notre algorithme généraliste (Théorème A) de décomposition équidimensionnelle fait intervenir un facteur $(\delta^a)^3$, alors que dans le cas particulier du Théorème C celle-ci n'est qu'en δ^2. Le seul sous algorithme en $(\delta^a)^3$ et le calcul de différence entre deux composantes équidimensionnelles présenté en §IV.3 : le problème consiste à calculer les composantes d'une variété qui ne sont pas incluses dans une autre. Ces composantes sont, bien entendu, données par des fibres de remontée. Il est naturel de conjecturer que l'on peut résoudre ce problème en seulement $(\delta^a)^2$.

I.6.3 Homotopies non archimédiennes

L'homotopie est un cadre unifiant les méthodes de résolution des systèmes d'équations polynomiales non basées sur la réécriture. Qu'il s'agisse de méthodes numériques ou de notre méthode formelle, la résolution du système s'effectue par approximations successives des solutions de systèmes déformations du système d'origine. Schématiquement, la différence principale entre les points de vue numérique et formel réside dans le caractère archimédien ou non de la métrique utilisée. Le cas archimédien correspond au cadre de l'analyse numérique, le cas non archimédien aux topologies $\mathfrak{m}$-adiques utilisées en calcul formel. Le cas non archimédien évite les questions liées à la stabilité et à la précision numérique, les complexités booléennes des algorithmes sont plus simples à obtenir. Alors que les méthodes numériques sont particulièrement gênées lorsque deux chemins d'une homotopie se croisent ou lorsqu'un chemin part à l'infini, il n'en est rien de notre approche non archimédienne.

Une fois ce cadre posé, il est intéressant d'étudier les différentes homotopies possibles pour aboutir à une résolution formelle. Le cas le plus simple est le suivant : si F représente le vecteur $(f_1,\dots,f_n)$ on considère l'homotopie $F(t,x) := F(x) - (1-t)F(P)$ où P est un point choisi au hasard dans $\mathbb{Q}^n$. Lorsque $t = 0$, le point $x = P$ est une solution du système, en $t = 1$ on retrouve le système d'origine.

On regarde le système $F(t,x) = 0$ dans $\mathbb{Q}[t,x_1,\dots,x_n]$ et on cherche à calculer les courbes algébriques solutions passant par le point $(0,P)$. On peut procéder comme suit : avec l'opérateur de Newton on calcule la série en t solution au voisinage de P à un certain ordre, on cherche ensuite son polynôme minimal et c'est à nouveau un problème d'approximation algébrique. Le Théorème C me fait espérer une réalisation de ces calculs en temps linéaire en la complexité d'évaluation du système et d^{2n} (rappelons que d est un majorant du degré des équations). Ceci constituerait une amélioration notable des algorithmes connus et rendrait très efficace cette méthode fort simple.

Une autre homotopie particulièrement intéressante est celle introduite par D. N. Bernshtein [Ber75] pour la résolution des systèmes creux. De façon succincte, le résultat est le suivant : un système d'équations creuses voit son nombre de solutions borné par un certain volume mixte. Cette borne est atteinte pour des coefficients numériques génériques. La preuve originale par D. N. Bernhstein fait intervenir une homotopie qui a été ensuite reprise dans de nombreux travaux d'analyse numérique. Ces résultats ont culminé avec l'implantation par Vershelde d'une homotopie numérique suivant cette idée ; son logiciel est certainement le meilleur dans sa catégorie mais malheureusement aucune analyse de complexité fine n'a été entreprise à ce jour.

Notre approche non archimédienne de cette homotopie permettrait d'obtenir d'une façon similaire les solutions formelles du système ; je pense pouvoir obtenir une complexité linéaire dans la complexité d'évaluation du système et dans le carré du volume mixte (avec un algorithme probabiliste). Cela constituerait un résultat améliorant nettement ceux des approches par le résultant creux. Il reste cependant des obstacles de taille à surmonter.

Chapitre II

Resolution of a Reduced and Regular Sequence

Given a system of polynomial equations and inequations with coefficients in the field of rational numbers, we show how to compute a geometric resolution of the set of common roots of the system over the field of complex numbers. A geometric resolution consists of a primitive element of the algebraic extension defined by the set of roots, its minimal polynomial and the parametrizations of the coordinates. Such a representation of the solutions has a long history which goes back to Leopold Kronecker and has been revisited many times in computer algebra.

We introduce a new generation of probabilistic algorithms where all the computations use only univariate or bivariate polynomials. We give a new codification of the set of solutions of a positive dimensional algebraic variety relying on a new global version of Newton's iterator. Roughly speaking the complexity of our algorithm is polynomial in some kind of degree of the system, in its height, and linear in the complexity of evaluation of the system.

We present our implementation in the Magma system which is called Kronecker in homage to his method for solving systems of polynomial equations. We show that the theoretical complexity of our algorithm is well reflected in practice and we exhibit some cases for which our program is more efficient than the other available software.

Parts of this chapter have been published in [GLS01].

II.1 Introduction

We are interested in solving systems of polynomial equations, possibly including inequations. Let $f_1, \ldots, f_n$ and g be polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$ such that the system $f_1 = \cdots = f_n = 0$ with $g \neq 0$ has only a finite set of solutions over the field $\mathbb{C}$ of complex numbers. We show how to compute a representation of this set in the form

$$q(T) = 0, \quad \left\{ \begin{array}{rcl} x_1 & = & v_1(T), \\ & \vdots & \\ x_n & = & v_n(T), \end{array} \right. \tag{II.1}$$

where q is a univariate polynomial with coefficients in $\mathbb{Q}$ and the v_i, $1 \leq i \leq n$, are univariate rational functions with coefficients in $\mathbb{Q}$.

Let us sketch our algorithm, which is incremental in the number of equations to be solved. At step i we have a resolution

$$q(T) = 0, \quad \left\{ \begin{array}{lcl} x_{n-i+1} & = & v_{n-i+1}(T), \\ & \vdots & \\ x_n & = & v_n(T), \end{array} \right. \tag{II.2}$$

of the solution set of

$$f_1 = \cdots = f_i = 0, \quad g \neq 0, \qquad x_1 = a_1, \ldots, x_{n-i} = a_{n-i},$$

where q is a univariate polynomial over $\mathbb{Q}$, the v_j are univariate rational functions over $\mathbb{Q}$ and the a_j are chosen generic enough in $\mathbb{Q}$. The variable T represents a linear form separating the solutions of the system: the linear form takes different values when evaluated on two different points that are solution of the system. From there, two elementary steps are performed. The first step is a Newton-Hensel lifting of the variable x_{n-i} in order to obtain a geometric resolution

$$Q(x_{n-i}, T) = 0, \quad \left\{ \begin{array}{lcl} x_{n-i+1} & = & V_{n-i+1}(x_{n-i}, T), \\ & \vdots & \\ x_n & = & V_n(x_{n-i}, T), \end{array} \right. \tag{II.3}$$

of the 1-dimensional solution set of

$$f_1 = \cdots = f_i = 0, \quad g \neq 0, \qquad x_1 = a_1, \ldots, x_{n-i-1} = a_{n-i-1},$$

where Q is polynomial in T and rational in x_{n-i} and the V_j are bivariate rational functions over $\mathbb{Q}$. The second step is the intersection of this 1-dimensional set with the solution set of the next equation $f_{i+1} = 0$, which leads to a geometric resolution like (II.2) for step $i + 1$.

At step i the system $f_1, \ldots, f_i$ defines a positive dimensional variety, the new codification of its resolution we propose here consists of a specialization of some variables and a resolution of the zero-dimensional specialized system. This representation makes the link between the positive and zero dimensions and relies on two main ideas: the *Noether position* and the *lifting fiber* (see §II.3).

II.1.1 History

The representation of a variety in the form of (II.1) above has a long history. To the best of our knowledge the oldest trace of this representation is to be found in Kronecker's work at the end of the 19th century [Kro82] and a few years later in König's [Kön03]. Their representation is naturally defined for positive dimensional algebraic varieties, for instance for a variety of codimension i it has the form:

$$q(x_1, \ldots, x_{n-i}, T) = 0, \quad \left\{ \begin{array}{lcl} \frac{dq}{dT} x_{n-i+1} & = & w_{n-i+1}(x_1, \ldots, x_{n-i}, T), \\ & \vdots & \\ \frac{dq}{dT} x_n & = & w_n(x_1, \ldots, x_{n-i}, T), \end{array} \right. \tag{II.4}$$

where $q, w_{n-i+1}, \ldots, w_n$ are polynomials in $x_1, \ldots, x_{n-i}$ and T with coefficients in $\mathbb{Q}$ and such that q is square free. A good summary of their work can be found in Macaulay's book [Mac16].

This representation has been used in computer algebra as a tool to obtain complexity results by many authors, in the particular zero-dimensional case: Chistov, Grigoriev [CG82], Canny [Can88], Gianni, Mora [GM89], Kobayashi, Fujise, Furukawa [KFF88], Heintz, Roy, Solerno [HRS90], Lakshman, Lazard [LL91], Renegar [Ren92], Giusti, Heintz [GH93], Alonso, Becker, Roy, Wörman [ABRW96] and many others. From a practical point of view, the computation of such a representation is always relying on Gröbner basis computations [BW93], either with a pure lexicographical elimination order, or with an algorithm of change of basis [FGLM93, CKM97] or from any basis using a generalization of Newton's formulæ by Rouillier [Rou96, Rou99].

In 1995, Giusti, Heintz, Morais and Pardo [GHMP95, Par95] rediscovered Kronecker's approach without any prior knowledge of it and improved the space complexity but not the running time complexity. A first breakthrough was obtained by Giusti, Hägele, Heintz, Montaña, Morais, Morgenstern and Pardo [GHH$^+$97, GHM$^+$98]: there exists an algorithm with a complexity roughly speaking polynomial in the degree of the system, in its height and in the number of the variables. Then, in [GHMP97], it is announced that the height of the integers does not appear in the complexity if the integers are represented by **straight-line programs**. For exact definitions and elementary properties of the notion of straight-line programs we refer to [Str72, Gat86, Sto89, Hei89]. A good historical presentation of all these works can be found in [CHMP01] and a didactic presentation of the algorithm is in [Mor97]. We recall the main statement of these works:

Theorem [GHMP97] *Let g and $f_1, \ldots, f_n$ be polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$. Suppose that $f_1, \ldots, f_n$ define a reduced regular sequence in the open subset $\{g \neq 0\}$ of $\mathbb{C}^n$ and are of degree at most d, coded by straight-line programs of size at most L and height at most h. There is a bounded error Turing machine that outputs a geometric resolution of $\mathcal{V}(f_1, \ldots, f_n) \backslash \mathcal{V}(g)$. The time complexity of the execution is in $L(nd\delta h)^{\mathcal{O}(1)}$ if we represent the integers of the output by straight-line programs.*

The **reduced** and **regular** hypothesis means that each variety

$$\mathcal{V}_i = \overline{\mathcal{V}(f_1, \ldots, f_i) \backslash \mathcal{V}(g)}, \qquad 1 \leq i \leq n,$$

has dimension $n - i$ and for each $1 \leq i \leq n-1$ the localized quotient

$$\left(\mathbb{Q}[x_1, \ldots, x_n]/(f_1, \ldots, f_i)\right)_g$$

is reduced. Using the Jacobian criterion, this is equivalent to the situation when the Jacobian matrix of $f_1, \ldots, f_i$ has full rank at each generic point of $\mathcal{V}_i$. This condition is not really restrictive since we can perform a generic linear combination of the equations to recover this situation, as showed in [KP94, Proposition 37]. The number δ is defined as $\max(\deg(\mathcal{V}_1), \ldots, \deg(\mathcal{V}_{n-1}))$, it is bounded by d^n, by Bézout's theorem [Hei83] (see also [Vog84] and [Ful84]). A precise definition of geometric resolutions is given in §II.3.2.

The geometric resolution returned by the algorithm underlying the above theorem has its integers represented by means of straight-line programs, the manipulation of such a representation has been studied in [HM97] and [Häg98]. The size of the integers of the intermediate computations is bounded by the one of the output. In [CHMP01, Theorem 20] this result has been refined, showing how to compute efficiently only the solutions of height bounded by a given value.

II.1.2 Contributions

We have transformed this algorithm in order to obtain a new and simpler one, as above, without using straight-line programs anymore, neither for multivariate polynomials nor for integer numbers. We give a new estimate of the exponents of the complexity of Theorem [GHMP97] above improving the results of [HMW01].

One main step of this transformation is obtained by a technique reminiscent of the deforestation [Wad90, CDPR98], that we had already used in [GHL$^+$00] to replace straight-line programs by an efficient use of specialization. We only need polynomials in at most two variables. From a geometrical point of view our algorithm only needs to compute the intersection of two curves. This improvement has been independently discovered in [HMW01, Remark 13].

The second step is the use of Kronecker's form (II.4) to represent geometric resolutions, leading to a lower total degree complexity in the positive dimensional case. In [Rou96, Rou99], this representation has also been used and its good behavior in practice in the zero dimensional case has been observed.

The third step is the use of a global Newton iterator presented in §II.2.2 and §II.4. This improves the original algorithm of [Mor97, §4.2.1] by avoiding to compute a geometric resolution of each $\mathcal{V}_i$ from a *lifting fiber* (see §II.3.4) by means of primitive elements computations in two variables [Mor97, Lema 54].

The fourth simplifying step is the use of a simple technique to intersect a variety by a hypersurface, which was already present in Kronecker's method presented in §II.2.2 and §II.6. This improves [HMW01, §4.2] by avoiding the use of primitive elements computations in two variables which is used twice in [Mor97]. This technique first appeared in [GH93] and developed in [KP96].

The last step is the intensive use of modular arithmetic: the resolution is computed modulo a small prime number, the integers are lifted at the end by our global Newton iterator. Hence the cost of integer manipulations is quite optimal: we never use integers more than twice as large as the ones contained in the output.

II.1.3 Results

We present three new results: the first one gives a new arithmetic complexity in terms of number of operations in the base field $\mathbb{Q}$, the second one is more realistic and takes care of the bit length of the integers and the third one consists in an implementation of our algorithm which demonstrates its tractability and efficiency.

For our complexity measurement we use the class of functions $\mathcal{U}$ defined by

$$\mathcal{U}(n) := \mathcal{O}(n \log^2(n) \log\log(n)).$$

As recalled in §II.3.5, if R is any unitary ring, it represents the complexity of the arithmetic operations in $R[T]$ for polynomials of degree at most n in terms of operations in R: addition, multiplication, division, resultant (if R is integral), greatest common divisor and interpolation (if R is a field). It is also the bit complexity of the arithmetic operations of the integers of bit-size at most n: addition, multiplication, division, greatest common divisor. The class $\mathcal{O}(n^\Omega)$ represents the complexity of the arithmetic operations of the matrix with coefficients in R of size $n \times n$ in terms of arithmetic operations in R: addition, multiplication, determinant and adjoint. We know that Ω is less than 4.

Theorem 1 *Let k be a field of characteristic zero, let $f_1, \ldots, f_n, g$ be polynomials in the ring $k[x_1, \ldots, x_n]$ of degree at most d and given by a straight-line program of size at most L, such that $f_1, \ldots, f_n$ defines a reduced regular sequence in the open subset $\{g \neq 0\}$. The geometric resolution of the variety $\mathcal{V}(f_1, \ldots, f_n) \backslash \mathcal{V}(g)$ can be computed with $\mathcal{O}(n(nL + n^\Omega)(\mathcal{U}(d\delta))^2)$ arithmetic operations in k, where $\delta = \max(\deg(\mathcal{V}_1), \ldots, \deg(\mathcal{V}_{n-1}))$. There is a probabilistic algorithm performing this computation. Its probability of returning correct results relies on choices of elements of k. Choices for which the result is not correct are enclosed in strict algebraic subsets.*

The fact that bad choices are enclosed in strict algebraic subsets implies that almost all random choices lead to a correct computation. In this sense we can say that our probabilistic algorithm has a low probability of failure. Our algorithm is not Las Vegas, but it satisfies a weaker property: one can check that the geometric resolution it returns satisfies the input equations; if it does some of the solutions have been found but not necessarily all of them. In the special case when the output contains $\deg(f_1)\deg(f_2)\cdots\deg(f_n)$ solutions Bézout's theorem implies that all of them have been found.

In order to compare the complexity of our algorithm to Gröbner bases computations we apply our complexity theorem to the case of systems of polynomials $f_1, \ldots, f_n$ given by their dense representation:

Corollary 1 *Let $f_1, \ldots, f_n$ be a reduced regular sequence of polynomials of $k[x_1, \ldots, x_n]$ of degree at most d. Assume that d is at least n, then the geometric resolution of $\mathcal{V}(f_1, \ldots, f_n)$ can be computed with $\mathcal{O}(d^{3(n+\mathcal{O}(1))})$ arithmetic operations in k with the probabilistic algorithm of Theorem 1.*

Proof. By Bézout's inequality $d\delta$ is at most d^n, so $\mathcal{U}(d\delta)$ is in $d^{n+\mathcal{O}(1)}$. And L is at most $n\binom{d+n}{n}$, which is in $d^{n+\mathcal{O}(1)}$. □

Our algorithm does not improve drastically the worst case complexity in case of dense input systems; its efficiency fully begins when either the complexity of evaluation of the input system is small or when the hypersurface $g = 0$ contains several components of each $\mathcal{V}_i$, i.e. δ is small with respect to d^n.

These results are proved for a field of characteristic 0 and are not valid for fields of positive characteristic. However, when k is equal to $\mathbb{Q}$, it is tempting to compute resolutions in $\mathbb{Z}/p\mathbb{Z}$ for some prime numbers p. We have a result in this direction: from a resolution computed modulo a *lucky prime number* p we can deduce the resolution in $\mathbb{Q}$, and p can be chosen small with respect to the integers of the output.

Theorem 2 *Assume that* k *is* $\mathbb{Q}$, $\mathcal{V} = \overline{\mathcal{V}(f_1, \ldots, f_n) \backslash \mathcal{V}(g)}$ *is zero-dimensional and the algebra* $(\mathbb{Q}[x_1, \ldots, x_n]/(f_1, \ldots, f_n))_g$ *is reduced.*

Let u *be a primitive element of the extension* $\mathbb{Q} \to \mathbb{Q}[\mathcal{V}]$, $q(T)$ *its monic minimal polynomial in* $\mathbb{Q}[T]$. *Let* D *be the degree of* q, D *is equal to* $\deg(\mathcal{V})$. *Let* $w_i(T)$, $1 \leq i \leq n$, *be polynomials of* $\mathbb{Q}[T]$ *of degree strictly less than* D *such that* $q'(u)x_i - w_i(u)$ *is equal to zero in* $\mathbb{Q}[\mathcal{V}]$.
If we are given

- η *the bit-size of the integers of the polynomials* q *and* w_i;
- *a prime number* p *not dividing any denominator appearing in* q *and the* w_i *and such that* $\log(p) < \eta$;
- q_p *and* $w_{1,p}, \ldots, w_{n,p}$ *polynomials in* $\mathbb{Z}/p\mathbb{Z}[T]$, *images of* q *and* $w_1, \ldots, w_n$,

such that

- q'_p *is invertible modulo* q_p;
- *for each* i, $1 \leq i \leq n$, $f_i(w_{1,p}/q'_p, \ldots, w_{n,p}/q'_p) \equiv 0 \ [q_p]$;
- *the Jacobian matrix* J *of the* f_i *is invertible:* $\det(J(w_{1,p}/q'_p, \ldots, w_{n,p}/q'_p))$ *is invertible modulo* q_p,

then the polynomials q *and the* w_i *can be reconstructed in the bit-complexity*

$$\mathcal{O}((nL + n^{\Omega})\mathcal{U}(D)\mathcal{U}(\eta)).$$

From a practical point of view we combine the algorithms related to Theorems 1 and 2 in the following way: first choose at random a small prime number, compute a geometric resolution of the input system modulo p and then lift the integers to get the geometric resolution in $\mathbb{Q}$.

The problem of choosing a prime number for which this algorithm leads to a correct result is similar to the problem of computing the greatest common divisor of two univariate polynomials over $\mathbb{Q}$ by means of modular computations and Hensel's lifting (for example see [DST93, §4.1.1] or [GCL94, §7.4]). The description of the probability of choosing a *lucky* p is out of the scope of this work but such considerations are as in [Häg98, HM97, HMPS97].

The probability of failure of the algorithm given in [HMW01] has been studied using Zippel-Schwartz's zero test [Zip79, Sch80] for multivariate polynomials. We could use the same analysis here to quantify the probability mentioned in Theorem 1, but this has no practical interest without the quantification of the probability of choosing a lucky prime number p. These probabilities will be studied in forthcoming works.

II.1.4 Implementation: the Kronecker Package

One aim of our implementation is to demonstrate that our algorithm has a practical interest and is competitive with the other methods. We have implemented our algorithm within the computer algebra system Magma [Mag, CP96, BCP97], the package has been called Kronecker [Lec99c] version 0.1 and is available with its documentation at http://kronecker.medicis.polytechnique.fr.

We compare our implementation to Gröbner bases computations for total degree orders and algorithms of change of bases. Given a Gröbner basis of a zero-dimensional polynomial equation system one can deduce a Rational Univariate Representation of the zeros via the algorithm proposed in [Rou96, Rou99]. We also compare our implementation to the one of [Rou96].

This chapter is organized as follows. The next section is devoted to an informal presentation of the whole algorithm reflecting the actual computations performed in a generic situation. We then give definitions and introduce our encoding of the solutions. The next three sections are devoted to the formal presentation and proofs of our Newton iterator and the intersection algorithm. Section II.7 presents the whole algorithm and specifies the random choices. The last part provides some practical aspects of our implementation in the Magma system and comparisons with other methods for solving systems of polynomial equations.

II.2 Description of the Algorithm

We first give an informal presentation of the probabilistic aspects of our algorithm. Then we show the actual computations that are performed in a generic case, forgetting the modular computational aspects for the moment. All these points are detailed in the next sections.

II.2.1 Outlook of the Probabilistic Aspects

Let k be a field of characteristic zero, and $f_1, \ldots, f_n, g$ be polynomials in the ring $k[x_1, \ldots, x_n]$ under the hypotheses of Theorem II.1.1. The system

$$\mathcal{S} = \{f_1 = \cdots = f_n = 0, g \neq 0\}$$

has only a finite set of solutions in the n-affine space over an algebraic closure of k. Our algorithm is parametrized by three **parameters** N, L, C, called respectively the **Noether points**, **lifting points** and **Cayley points**: they are functions returning tuples of integers (see §II.7.2). Once the parameters are fixed this specifies a deterministic algorithm $\mathcal{A}_{\mathsf{N},\mathsf{L},\mathsf{C}}$ for the resolution of $\mathcal{S}$. For a proper choice of these parameters, the algorithm $\mathcal{A}_{\mathsf{N},\mathsf{L},\mathsf{C}}$ computes a resolution of the set of solutions of the system in the form:

$$q(T) = 0, \quad \left\{ \begin{array}{rcl} x_1 & = & v_1(T), \\ & \vdots & \\ x_n & = & v_n(T), \end{array} \right. \tag{II.5}$$

where $q, v_1, \ldots, v_n \in k[T]$ and T represents a k-linear form in the x_i.

The time complexity of the execution of $\mathcal{A}_{\mathsf{N,L,C}}$ for such a proper choice is $L(ndh\delta)^{\mathcal{O}(1)}$. It has been shown in [GHH$^+$97] that the choices of the parameters can be done using Correct Test Sequences of size polynomial in the sequential complexity of the algorithm. In [HMW01, Theorem 5] it is shown using Zippel-Schwartz's equality test [Zip79, Sch80] that the choices can be done at random in a set of integers of size polynomial in the sequential complexity of $\mathcal{A}_{\mathsf{N,L,C}}$ with a uniformly bounded probability of failure less than 1/2.

In the case of our algorithm, we precise these parameters in §II.7 and we show that we can choose them in a Zariski open subset of the space of choices. In particular this means that any random choice suits the input system.

We say that our algorithm is **semi-numerical** since it is parametrized by some initial choices in the same way as some numerical algorithms are. Our advantage over numerical algorithms is the *certification* of the result. In [CHMP01] a comparison is made between our method and the numerical approach using homotopy and the approximate zero theory introduced by Smale [Sma81, Sma86].

II.2.2 Description of the Computations

We present now the actual computations performed by our algorithm in a generic case. Our algorithm is incremental in the number of equations to be solved. Let $\mathcal{S}_i$ be the system of polynomial equations

$$x_1 = \cdots = x_{n-i} = f_1 = \cdots = f_i = 0, \quad g \neq 0.$$

The algorithm solves $\mathcal{S}_1, \ldots, \mathcal{S}_n$ in sequence. We enter step i with a solution of $\mathcal{S}_i$ in the form

$$q(T) = 0, \quad \left\{ \begin{array}{lcl} x_{n-i+1} & = & T, \\ x_{n-i+2} & = & v_{n-i+2}(T), \\ & \vdots & \\ x_n & = & v_n(T), \end{array} \right. \tag{II.6}$$

with the property that x_{n-i+1} separates the points of $\mathcal{S}_i$. We want to compute such a solution for $\mathcal{S}_{i+1}$. The computation divides into three main parts: the lifting, the intersection and the cleaning steps.

The Lifting Step

Starting from (II.6), we compute a solution of the system $\mathcal{S}_i'$

$$x_1 = \cdots = x_{n-i-1} = f_1 = \cdots = f_i = 0, \quad g \neq 0,$$

in the form

$$Q(x_{n-i}, T) = 0, \quad \left\{ \begin{array}{lcl} x_{n-i+1} & = & T, \\ \frac{\partial Q}{\partial T}(x_{n-i}, T) x_{n-i+2} & = & W_{n-i+2}(x_{n-i}, T), \\ & \vdots & \\ \frac{\partial Q}{\partial T}(x_{n-i}, T) x_n & = & W_n(x_{n-i}, T), \end{array} \right. \tag{II.7}$$

such that the W_j and Q are polynomials in x_{n-i} and T. The solution

$$\mathbf{v} = (T, v_{n-i+2}(T), \ldots, v_n(T))$$

from (II.6) can be seen as an approximated solution of $\mathcal{S}'_i$ at precision $\mathcal{O}(x_{n-i})$. We now show how it can be lifted to a solution at precision $\mathcal{O}(x_{n-i}^2)$.

We first compute, with the classical Newton method,

$$\begin{pmatrix} V_{n-i+1}(x_{n-i}, T) \\ \vdots \\ V_n(x_{n-i}, T) \end{pmatrix} = \mathbf{v}^t - J(0, \ldots, 0, x_{n-i}, \mathbf{v})^{-1} \begin{pmatrix} f_1(0, \ldots, 0, x_{n-i}, \mathbf{v}) \\ \vdots \\ f_i(0, \ldots, 0, x_{n-i}, \mathbf{v}) \end{pmatrix},$$

modulo $q(T)$ and at precision $\mathcal{O}(x_{n-i}^2)$, where J is the Jacobian matrix of $f_1, \ldots, f_i$ with respect to $x_{n-i+1}, \ldots, x_n$. The parametrization

$$q(T) = 0, \quad \left\{ \begin{array}{rcl} x_{n-i+1} & = & V_{n-i+1}(x_{n-i}, T), \\ & \vdots & \\ x_n & = & V_n(x_{n-i}, T), \end{array} \right. \tag{II.8}$$

is a solution of $\mathcal{S}'_i$ at precision $\mathcal{O}(x_{n-i}^2)$. The expression V_{n-i+1} can also be written

$$V_{n-i+1}(x_{n-i}, T) = T + x_{n-i}\Delta(T) + \mathcal{O}(x_{n-i}^2).$$

Hence

$$T = x_{n-i+1} - x_{n-i}\Delta(x_{n-i+1}) + \mathcal{O}(x_{n-i}^2).$$

Substituting the right-hand side for T in q and the V_j we get:

$$q(x_{n-i+1}) - x_{n-i}(q'(x_{n-i+1})\Delta(x_{n-i+1}) \bmod q(x_{n-i+1})) + \mathcal{O}(x_{n-i}^2) = 0$$

and

$$x_j = V_j(x_{n-i}, x_{n-i+1}) - x_{n-i}(\frac{\partial V_j}{\partial T}\Delta(x_{n-i+1}) \bmod q(x_{n-i+1})) + \mathcal{O}(x_{n-i}^2),$$

for $n - i + 1 \leq j \leq n$, which is an approximated solution of $\mathcal{S}'_i$ at precision $\mathcal{O}(x_{n-i}^2)$. We continue this process up to a certain precision. At the end, multiplying both sides of the parametrization of the coordinates by the derivative of q with respect to T and reducing the right-hand side with respect to q, we get the resolution (II.7) exactly. Section II.4 gives the full description of this method.

Compared to the original algorithm in [Mor97], this method shortcuts the reconstruction of the whole geometric resolution from a fiber by means of primitive element computations in two variables. Compared to [HMW01], we only need to perform the lifting at precision the degree of the current variety. This method also applies for integers to lift a geometric resolution known modulo a prime number p, see §II.4.6.

The Intersection Step

To the solution (II.7) of $\mathcal{S}'_i$ we add the new equation $f_{i+1} = 0$. Let X be a new variable, we first perform the following change of variables in the power series ring $k[[t]]$:

$$x_{n-i} = X - tx_{n-i+1} + \mathcal{O}(t^2).$$

This leads to a new parametrization in the form

$$Q_t(X,T) = 0, \qquad \begin{cases} x_{n-i+1} &= T, \\ x_{n-i+2} &= V_{t,n-i+2}(X,T), \\ &\vdots \\ x_n &= V_{t,n}(X,T), \end{cases} \tag{II.9}$$

where Q_t is a polynomial in X and T and the $V_{t,j}$ are polynomial in T and rational in X with coefficients in $k[[t]]$ at precision $\mathcal{O}(t^2)$. Then we compute

$$\begin{aligned} A(X) \quad &= \mathrm{Resultant}_T(Q_t(X,T), \\ &f_{i+1}(0,\dots,0,X-tT,T,V_{t,n-i+2}(X,T),\dots,V_{t,n}(X,T))). \end{aligned}$$

The resultant $A(X)$ is indeed in $k[X][[t]]$ and replacing X by $x_{n-i} + tx_{n-i+1}$ in

$$A(X) = a_0(X) + ta_1(X) + \mathcal{O}(t^2) = 0,$$

we get:

$$a_0(x_{n-i}) = 0, \qquad a_0'(x_{n-i})x_{n-i+1} + a_1(x_{n-i}) = 0,$$

which gives the desired resolution of $\mathcal{S}_i \cup \{f_{i+1} = 0\}$. If a_0 is not relatively prime with its first derivative a_0', we replace a_0 by its square free part a_s and let $a_m = a_0/a_s$, a_m divides a_0' and a_1. The parametrization becomes: $a_0'/a_m x_{n-i+1} + a_1/a_m = 0$. Then a_0'/a_m is relatively prime with a_0. These computations are described in more detail in §II.6.

This method simplifies considerably the ones given in the original algorithm of [Mor97] and [HMW01] relying on primitive element computations in two variables.

The Cleaning Step

We now have a resolution of $\mathcal{S}_i \cup \{f_{i+1} = 0\}$ in the form

$$q(T) = 0, \qquad \begin{cases} x_{n-i} &= T, \\ x_{n-i+1} &= v_{n-i+1}(T), \\ x_{n-i+2} &= v_{n-i+2}(T), \\ &\vdots \\ x_n &= v_n(T), \end{cases} \tag{II.10}$$

where q and the v_j are new polynomials in T. To get a resolution of $\mathcal{S}_{i+1}$ we must remove the points contained in the hypersurface $g = 0$. To do this, we compute the greatest common divisor:

$$c(T) = \mathrm{gcd}_T(q, g(0,\dots,0,T,v_{n-i+1},\dots,v_n)).$$

Then we just have to replace q by q/c and reduce the parametrizations v_j by the new polynomial q. This algorithm relies on Proposition 14: it simplifies [Mor97, §4.3.1].

The rest of this chapter is devoted to the justifications of these computations and to the comparison of practical results with some other methods.

II.3 Definitions and Basic Statements

One key feature of our algorithm is an effective use of the *Noether Normalization Lemma* also seen geometrically as a *Noether Position*. It allows us to represent a positive dimensional variety as a zero-dimensional one.

II.3.1 Noether Position, Primitive Element

Let k be a field of characteristic 0. Let $x_1, \ldots, x_n$ be indeterminates over k. Let $\mathcal{V}$ be a r-dimensional k-variety in $\overline{k}^n$, where $\overline{k}$ is the algebraic closure of k and $\mathfrak{I} = \mathfrak{I}(\mathcal{V})$ the annihilating ideal of $\mathcal{V}$.

We say that a subset of variables $Z = \{x_{i_1}, \ldots, x_{i_k}\}$ is *free* when $\mathfrak{I} \cap k[x_{i_1}, \ldots, x_{i_k}] = (0)$. A variable is **dependent** or **integral** with respect to a subset of variables Z if there exists in $\mathfrak{I}(\mathcal{V})$ a monic polynomial annihilating it and whose coefficients are polynomial in the variables of Z only.

A **Noether normalization** of $\mathcal{V}$ consists of a k-linear change of variables, transforming the variables $x_1, \ldots, x_n$ into new ones, $y_1, \ldots, y_n$, such that the linear map from $\overline{k}^n$ to $\overline{k}^r$ $(r \leq n)$ defined by the forms $y_1, \ldots, y_r$ induces a finite surjective morphism of affine varieties $\pi : \mathcal{V} \to \overline{k}^r$. This is equivalent to the fact that the variables $y_1, \ldots, y_r$ are free and $y_{r+1}, \ldots, y_n$ dependent with respect to the first ones. In this situation we say that $y_1, \ldots, y_n$ are in Noether position.

If B is the coordinate ring $k[\mathcal{V}]$, then a Noether normalization induces an integral ring extension $R := k[y_1, \ldots, y_r] \to B$. Let K be the field of fractions of R and B' be $K \otimes_R B$, B' is a finite-dimensional K-vector space.

Example 1 Consider $f = x_1 x_2$ in $\mathbb{Q}[x_1, x_2]$, f defines a hypersurface in the affine space of dimension two over the complex numbers. The variable x_1 is free but x_2 is not integral over x_1. This hypersurface is composed of two irreducible components $x_1 = 0$ and $x_2 = 0$. When specializing the variable x_1 to any value p_1 in k^*, $f(p_1, x_2)$ has one irreducible factor only. Let us take $y_1 = x_1 - x_2$ and $y_2 = x_2$ then f becomes $(y_1 + y_2)y_2 = y_2^2 + y_1 y_2$. The variable y_2 is integral over y_1: we have a Noether position of this hypersurface; we can specialize y_1 to 0 in f, there remains two irreducible components.

Example 2 Consider the hypersurface given by the equation $x_2 - x_1^2 = 0$. The variables x_1, x_2 are in Noether position but when specializing x_1 to a point of k, for instance 0, the fiber contains only one point while the hypersurface has degree 2. The vector space B' is $k(x_1)[x_2]/(x_2 - x_1^2)$ and has dimension one only.

The degeneration of the dimension of B' in the last example does not occur when working with projective varieties, so if we want to avoid it in affine spaces we need a kind of stronger Noether position.

We say that the variables $y_1, \ldots, y_n$ are in **projective Noether position** if they define a Noether position for the projective Zariski closure of $\mathcal{V}$. More precisely, let x_0 be a new variable, to any polynomial f of $k[x_1, \ldots, x_n]$, we write $f^h(x_0, \ldots, x_n)$ the homogenization of f with respect to x_0, $\mathfrak{I}^h$ denotes the ideal of the homogenized polynomial of $\mathfrak{I}$ and $\mathcal{V}^h$ the variety associated to $\mathfrak{I}^h$, which corresponds to the projective Zariski of $\mathcal{V}$. We say that the variables $y_1, \ldots, y_n$ are in projective Noether position with respect to $\mathcal{V}$ when $x_0, y_1, \ldots, y_n$ are in Noether position with respect to $\mathcal{V}^h$.

In the whole thesis we only use projective Noether positions, so we only say Noether position. We write $\mathfrak{I}'$ for the extension of $\mathfrak{I}$ in $K[y_{r+1}, \ldots, y_n]$. $\mathfrak{I}'$ is a zero-dimensional radical ideal. We are interested in some particular bases of B'. A k-linear form $u = \lambda_{r+1} y_{r+1} + \cdots + \lambda_n y_n$ such that the set of the powers $1, u, \ldots, u^{\deg(\mathcal{V})-1}$ form a basis of the vector space B' is called a **primitive element** of the variety $\mathcal{V}$.

In general we do not know any efficient way to compute in B. Even when it is a free module we do not know bases of small size [ADS98]. The next two propositions give some properties of computations in B'.

Proposition 1 *With the above notations, assume that $\mathcal{V}$ is r-equidimensional. If the variables $x_1, \ldots, x_n$ are in projective Noether position with respect to $\mathcal{V}$ then the dimension of B' is the degree of $\mathcal{V}$.*

We recall a result [SS96, Proposition 1], itself a continuation of [CGH88, Remark 9]:

Proposition 2 *Let $\mathfrak{I}$ be a radical ideal of $k[x_1, \ldots, x_n]$ such that the variety $\mathcal{V} = \mathcal{V}(\mathfrak{I})$ is r-equidimensional and the variables $x_1, \ldots, x_n$ are in Noether position. Let f be an element of $k[x_1, \ldots, x_n]$ and $\overline{f}$ its class in the quotient ring B. Let T be a new variable, then there exists a monic polynomial $F \in R[T]$ which satisfies $F(\overline{f}) = 0$ and whose total degree is bounded by $\deg(\mathcal{V})\deg(f)$.*

An alternative proof of this proposition is given in [Mor97, Corolario 21]. The next corollary expresses that minimal and characteristic polynomials in B' have their coefficients in R.

Corollary 2 *Let $\mathfrak{I}$ be a radical ideal of $k[x_1, \ldots, x_n]$ such that $\mathfrak{I}$ is r-equidimensional and the variables $x_1, \ldots, x_n$ are in Noether position. Let f be a polynomial in $k[x_1, \ldots, x_n]$. Then the characteristic polynomial χ of the endomorphism of multiplication by f in B' belongs to $k[x_1, \ldots, x_r][T]$. Its coefficient of degree i in T has degree at most $(\delta - i)\deg(f)$, where $\delta = \dim(B')$. In the case when $f = u$ is a primitive element of $\mathcal{V}$ we have $\chi(\overline{u}) = 0$.*

Proof. Let F be an integral dependence relation of f modulo the ideal $\mathfrak{I}$ of degree bounded by $\deg(\mathcal{V})\deg(f)$ from Proposition 2, M_f be the endomorphism of multiplication by f in B' and μ its minimal polynomial. First we note that $F(M_f) = 0$, thus μ divides F. The polynomials μ and F being monic we deduce using Gauss lemma that μ is in $R[T]$ and so is χ. If $f = u$, $\deg_T(\mu) = \deg_T(\mathcal{V})$ and thus $\mu = F$.

Let us now prove the bound on the degrees, to do this we homogenize the situation: let x_0 be a new variable and f^h denotes the homogenized polynomial of f, $\mathfrak{I}^h$ the homogenized ideal of $\mathfrak{I}$. Let now B' be $k(x_0, \ldots, x_r)[x_{r+1}, \ldots, x_n]/\mathfrak{I}^h$ and $\chi(T)$ the characteristic polynomial of the endomorphism of multiplication by f^h in B'. It is sufficient to prove that the coefficient of degree i in T of χ is homogeneous of degree $(\delta - i)\deg(f)$. To do this let $\overline{K}$ be the algebraic closure of $k(x_0, \ldots, x_r)$ and $Z_1, \ldots, Z_\delta$ be the zeroes of $\mathfrak{I}^h$ in $\overline{K}$. The following formula holds:

$$\chi(x_0, \ldots, x_r, T) = \prod_{i=1}^{\delta} (T - f^h(x_0, \ldots, x_r, Z_i)).$$

Hence, if t is a new variable we have

$$\begin{aligned}\chi(tx_0, \ldots, tx_r, T) &= \prod_{i=1}^{\delta} (T - f^h(tx_0, \ldots, tx_r, tZ_i)) \\ &= \prod_{i=1}^{\delta} (T - t^{\deg(f)} f^h(x_0, \ldots, x_r, Z_i)).\end{aligned}$$

Expanding this last expression, we get the claimed bound on the degrees of the coefficients in T of χ, this concludes the proof. □

II.3.2 Geometric Resolutions

Let $\mathcal{V}$ be a r-equidimensional algebraic variety and $\mathfrak{I}$ its annihilator ideal in $k[x_1, \ldots, x_n]$. A **geometric resolution** of $\mathcal{V}$ is given by:

- an invertible $n \times n$ square matrix M with entries in k such that the new coordinates $y = M^{-1}x$ are in Noether position with respect to $\mathcal{V}$;
- a primitive element $u = \lambda_{r+1} y_{r+1} + \cdots + \lambda_n y_n$ of $\mathcal{V}$;
- the minimal polynomial $q(T) \in R[T]$ of u in B', monic in T, and
- the **parametrization** of $\mathcal{V}$ by the zeros of q, given by polynomials

  $$v_{r+1}(y_1, \ldots, y_r, T), \ldots, v_n(y_1, \ldots, y_r, T) \in K[T],$$

  such that $y_j - v_j(y_1, \ldots, y_r, u) \in \mathfrak{I}'$ for $r + 1 \leq j \leq n$, where $\mathfrak{I}'$ is the extension of $\mathfrak{I}$ in $k(y_1, \ldots, y_r)[y_{r+1}, \ldots, y_n]$ and $\deg_T(v_j) < \deg_T(q)$.

Given a primitive element u, its minimal polynomial q is uniquely determined up to a scalar factor. The parametrization can be expressed in several ways. In the definition of geometric resolutions the parametrization of the algebraic coordinates has the form

$$y_j = v_j(T), \qquad r + 1 \leq j \leq n.$$

However, given any polynomial $p(T) \in K[T]$ relatively prime with $q(T)$ another parametrization is given by:

$$p(T)y_j = v_j(T)p(T), \qquad r+1 \le j \le n.$$

One interesting choice is to express the parametrization in the following way:

$$\frac{\partial q}{\partial T}(T)y_j = w_j(T), \qquad r+1 \le j \le n, \tag{II.11}$$

with $\deg_T w_j < \deg_T q$. We call a parametrization in the form of Equation (II.11) a **Kronecker parametrization**.

Proposition 3 *The polynomial q has its coefficients in R and in the Kronecker parametrization II.11 the polynomials w_i have also their coefficients in R instead of K. The* total *degree of q and the w_i is bounded by* $\deg_T(q)$. *Moreover $q(u)$ and $\frac{dq}{dT}(u)x_j - w_j(u)$ belong to $\mathfrak{I}$, for $r+1 \le j \le n$.*

In particular the discriminant of q with respect to T is a multiple of any denominator appearing in any kind of parametrization.

> **Example 3** Let $f_1 = x_3^2 + x_1x_2 + 1$ and $f_2 = x_2^2 + x_1x_3$, the variables x_1, x_2, x_3 are in Noether position, x_2 is a primitive element and we have the following Kronecker parametrization
>
> $$x_2^4 + x_1^3x_2 + x_1^2 = 0,$$
>
> $$(4x_2^3 + x_1^3)x_3 = 4x_1x_2 + 3x_1^2x_2^2.$$

The following is a fundamental result.

Proposition 4 *Given a Noether position and a primitive element, any r-equidimensional algebraic variety $\mathcal{V}$ admits a unique geometric resolution.*

The proofs of the last two propositions are given in the next section.

> **Example 4** Here is an example of an ideal which is not Cohen-Macaulay: in the polynomial ring $k[x_1, x_2, x_3, x_4]$, consider
>
> $$\mathfrak{I} = (x_2x_4, x_2x_3, x_1x_4, x_1x_3).$$
>
> $\mathfrak{I}$ is radical 2-equidimensional. A Noether position is given by $x_1 = y_3 - y_1, x_2 = y_4 - y_2, x_3 = y_3, x_4 = y_4$. The generating equations become $y_4^2 - y_2y_4, y_3y_4 - y_2y_3, y_3y_4 - y_1y_4, y_3^2 - y_1y_3$. For any $\lambda_3, \lambda_4 \in k$ and $u = \lambda_3y_3 + \lambda_4y_4$ we have $u^2 - \lambda_4y_2u - \lambda_3y_1u \in \mathfrak{I}$.

II.3.3 Generic Primitive Elements

Assume that $\mathfrak{I}$ is radical and equidimensional of dimension r and the variables $x_1, \ldots, x_n$ are in Noether position. The minimal polynomial of a generic primitive element is of great importance in algebraic geometry and computer algebra. It was already used by Kronecker as an effective way to compute geometric resolutions.

Let Λ_i be new variables, $i = r+1, \ldots, n$, $k_\Lambda = k(\Lambda_{r+1}, \ldots, \Lambda_n)$, and $R_\Lambda = k_\Lambda[x_1, \ldots, x_r]$. Let $\mathfrak{I}_\Lambda$ be the extension of $\mathfrak{I}$ in $k_\Lambda[x_1, \ldots, x_n]$. Let $u_\Lambda = \Lambda_{r+1}x_{r+1} + \cdots + \Lambda_n x_n$. The objects indexed with Λ are related to objects defined over k_Λ. The generic linear form u_Λ is a primitive element of $\mathfrak{I}_\Lambda$, let U_Λ be its characteristic polynomial in the algebra $B'_\Lambda := k_\Lambda(x_1, \ldots, x_r)[x_{r+1}, \ldots, x_n]/\mathfrak{I}_\Lambda$. From Corollary 2, the polynomial $U_\Lambda(x_1, \ldots, x_r, T)$ is square free, monic in T, of total degree equal to the degree of the variety corresponding to $\mathfrak{I}$, has its coefficients in R_Λ and we have

$$U_\Lambda(x_1, \ldots, x_r, u_\Lambda) \in \mathfrak{I}_\Lambda.$$

Differentiating U_Λ with respect to $\Lambda_{r+1}, \ldots, \Lambda_n$, we deduce the following geometric resolution of $\mathfrak{I}_\Lambda$:

$$\begin{array}{l} U_\Lambda(x_1, \ldots, x_r, T) = 0, \\ \left\{ \begin{array}{rcl} \frac{\partial U_\Lambda}{\partial T}(x_1, \ldots, x_r, T)x_{r+1} & = & -\frac{\partial U_\Lambda}{\partial \Lambda_{r+1}}(x_1, \ldots, x_r, T), \\ & \vdots & \\ \frac{\partial U_\Lambda}{\partial T}(x_1, \ldots, x_r, T)x_n & = & -\frac{\partial U_\Lambda}{\partial \Lambda_n}(x_1, \ldots, x_r, T). \end{array} \right. \end{array} \tag{II.12}$$

This proves Propositions 3 and 4.

Example 5 In the previous example with $\lambda_3\lambda_4 \neq 0$, we deduce the parametrization

$$u^2 - \lambda_4 y_2 u - \lambda_3 y_1 u = 0, \quad \left\{ \begin{array}{rcl} (2u - \lambda_4 y_2 - \lambda_3 y_1)y_3 & = & y_1 u, \\ (2u - \lambda_4 y_2 - \lambda_3 y_1)y_4 & = & y_2 u. \end{array} \right.$$

II.3.4 Lifting Fibers

Instead of processing the representation of univariate polynomials over the free variables we make an intensive use of specialization. Thanks to our lifting process presented in §II.4 we do not lose anything.

From now on we assume that $\mathcal{V}$ is an r-equidimensional variety which is a sub-variety of $\mathcal{V}(e_1, \ldots, e_t)$, where $\mathbf{e} = e_1, \ldots, e_t$ is a sequence of polynomials in $k[x_1, \ldots, x_n]$. We call such a sequence of polynomials an **annihilating system** of $\mathcal{V}$. Let $y_1, \ldots, y_n$ be new coordinates bringing $\mathcal{V}$ into a Noether position. We recall that π represents the finite projection morphism onto the free variables.

A point $p = (p_1, \ldots, p_r)$ in k^r is called a **lifting point** if the points of the fiber $\pi^{-1}(p)$ are geometrically smooth on $\mathcal{V}$ and smooth for the projection π.

We define a **fiber** of $\mathcal{V}$ the following record:

- an annihilating system $\mathbf{e} = (e_1, \ldots, e_t)$ of $\mathcal{V}$;
- an invertible $n \times n$ square matrix M with entries in k such that the new coordinates $y = M^{-1}x$ are in Noether position with respect to $\mathcal{V}$;
- a **specialization point** $p = (p_1, \ldots, p_r)$ for $\mathcal{V}$;
- a primitive element $u = \lambda_{r+1}y_{r+1} + \cdots + \lambda_n y_n$ of $\mathcal{V}_p = \pi^{-1}(p)$;
- the minimal polynomial $q(T) \in k[T]$ annihilating u over the points of $\mathcal{V}_p$;
- $n - r$ polynomials $\mathbf{v} = (v_{r+1}, \ldots, v_n)$ of $k[T]$, of degree strictly less than $\deg_T(q)$, giving the parametrization of $\mathcal{V}_p$ by the zeros of q: $y_j - v_j(u) = 0$ for all $r+1 \leq j \leq n$ and all roots u of q.

We say that a fiber is a **lifting fiber** if its specialization point is a lifting point. We have the following relations between the components of a fiber:

$$u(v_{r+1}(T), \ldots, v_n(T)) = T,$$

$$\mathbf{e} \circ M(p_1, \ldots, p_r, v_{r+1}(T), \ldots, v_n(T)) \equiv 0 \bmod q(T).$$

The following proposition explains the one to one correspondence between geometric resolutions and lifting fibers. The specialization of the free variables at a lifting point constitutes the main improvement of complexity of our algorithm: compared to rewriting techniques such as Gröbner bases computations, we do not have to store multivariate polynomials, but only univariate ones.

Proposition 5 *For any lifting fiber encoding a variety $\mathcal{V}$ there exists a unique geometric resolution of $\mathcal{V}$ for the same Noether position and primitive element. The specialization of the minimal polynomial and the parametrization of this geometric resolution on the lifting point gives exactly the minimal polynomial and the parametrization of the lifting fiber. We have $\deg(\mathcal{V}_p) = \deg(\mathcal{V})$.*

Proof. First, the equality $\deg(\mathcal{V}_p) = \deg(\mathcal{V})$ is a direct consequence of the definition of the degree and the choice of p.

Suppose now that the primitive element u for $\mathcal{V}_p$ is not primitive for $\mathcal{V}$. We can choose a primitive element u' of $\mathcal{V}$ which is also a primitive element for $\mathcal{V}_p$. The specialization of the corresponding Kronecker parametrization of $\mathcal{V}$ with respect to u' gives a parametrization of $\mathcal{V}_p$. Using the powers of u' as a basis of B', we can compute the minimal polynomial of u, of degree strictly less than δ. Its denominators do not vanish at p, hence its specialization at p gives an annihilating polynomial of u for $\mathcal{V}_p$ of degree strictly less than δ. This leads to a contradiction. This concludes the proof. $\square$

We say that two lifting fibers are **equivalent** if they represent the same variety $\mathcal{V}$. We say that F is **isolated** if $\mathcal{V}$ is isolated in $\mathcal{V}(\mathbf{e})$, where $\mathbf{e}$ denotes the annihilating system of F. In this situation we speak about a **lifting system**. All the lifting fibers occuring during the resolution process of this chapter are isolated. Moreover the variety $\mathcal{V}$ of dimension r is always a component of $\mathcal{V}(f_1, \ldots, f_{n-r})$. From the reduced and regularity hypotheses $f_1, \ldots, f_{n-r}$ of a lifting system for $\mathcal{V}$: a lifting point p is a point where the Jacobian matrix of $f_1, \ldots, f_{n-r}$ with respect to the dependent variables $y_{r+1}, \ldots, y_n$ is invertible at each point of $\pi^{-1}(p)$. When dealing without hypotheses on the input system in Chapter IV the situation gets a bit coarser.

We show that lifting points and primitive elements can be chosen at random with a low probability of failure in practice.

Proposition 6 *With the above notations and assumptions, the points*

$$(p_1, \ldots, p_r, \lambda_{r+1}, \ldots, \lambda_n) \in k^n$$

such that $(p_1, \ldots, p_r)$ is not a lifting point or $u = \lambda_{r+1}y_{r+1} + \cdots + \lambda_n y_n$ is not a primitive element for $\mathcal{V}_p$ are enclosed in a strict subset of k^n which is algebraic.

Proof. Let J be the Jacobian matrix of $f_1, \ldots, f_{n-r}$ with respect to the variables $y_{r+1}, \ldots, y_n$ and $F(T)$ be an integral dependency relation of $\det(J)$ modulo $\mathcal{V}$. By hypothesis $\det(J)$ is not a zero divisor in B. Hence the constant coefficient $A(y_1, \ldots, y_r)$ of F is not zero and satisfies $A \in \mathfrak{I} + (\det(J))$. Each point p such that $A(p) \neq 0$ is a lifting point.

Now fix a lifting point p and consider U_Λ of §II.3.3 for $\mathcal{V}_p$, then any point $\Lambda_{r+1} = \lambda_{r+1}, \ldots, \Lambda_n = \lambda_n$ such that the discriminant of U_Λ does not vanish is a primitive element of $\mathcal{V}_p$. □

Notations for the Pseudo-Code: For the pseudo-code of the algorithms we use the following notations. If F denotes a fiber: $F_{ChangeOfVariables}$ is M, $F_{PrimitiveElement}$ is u, $F_{LiftingPoint}$ is p, $F_{MinimalPolynomial}$ is q, $F_{Parametrization}$ is $\mathbf{v}$ and $F_{Equations}$ is $\mathbf{e}$. Moreover we denote by $\deg(F)$ the degree of the fiber that is $\deg_T(F_{MinimalPolynomial})$ and if F is a lifting fiber $\dim(F)$ the dimension of the algrebraic variety that F represents.

II.3.5 Complexity Notations

We now discuss the complexity of integer and polynomial arithmetic. In the whole thesis $\mathcal{U}(n)$ denotes

$$\boldsymbol{\mathcal{U}(n) := \mathcal{O}\Big(n \log^2(n) \log\log(n)\Big)}$$

and represents the bit-complexity of the arithmetic operations (addition, multiplication, quotient, remainder and gcd) of the integers of bit-size n and the complexity of the arithmetic operations of the polynomials of degree n in terms of number of operations in the base ring. Many authors have contributed to these topics. Some very good historical presentations can be found in the books of Aho, Hopcroft, Ullman [AHU74], Bürgisser, Clausen, Shokrolahi [BCS97], Bini, Pan [BP94] among others.

Let R be a unitary commutative ring, the Schönhage-Strassen polynomial multiplication [SS71, Sch77, Nus80] of two polynomials of $R[T]$ of degree at most n costs $\mathcal{O}(n \log(n) \log\log(n))$ arithmetic operations in R. The division of polynomials has the same complexity as the multiplication [BM72, Str73]. The greatest common divisor of two polynomials of degree at most n over a field K can be computed in $\mathcal{U}(n)$ arithmetic operations in K [MB73]. The resultant, the sub-resultants and the interpolation can also be computed within the same complexity [LR96, Fid87].

The Schönhage-Strassen algorithm [SS71] for multiplying two integers of bit-size at most n has a bit-complexity in $\mathcal{O}(n\log(n)\log\log(n))$. The division has the same complexity as the multiplication [Sie72]. The greatest common divisor has complexity $\mathcal{U}(n)$ [Sch71].

Let R be a unitary ring, the multiplication of two $n \times n$ matrices can be done in $\mathcal{O}(n^\omega)$ arithmetic operations in R. The exponent ω can be taken less than 2.376 [CW90]. If R is a field, Bunch and Hopcroft showed that matrix inversion is not harder than the multiplication [BH74]. According to [BH74], the converse fact is due to Winograd.

In our case, R is a k-algebra $k[T]/q(T)$, where q is a square-free monic polynomial of $k[T]$, so we can not apply the results of [BH74] to compute the inverse of a matrix. Let $\mathcal{O}(n^{\Omega'})$ denote the complexity of the elementary operations on $n \times n$ matrices over any commutative ring R in terms of arithmetic operations in R: addition, multiplication, determinant and adjoint matrix. In fact, Ω' is at most $1/2+\omega$ [PS78], if division by $n!$ is allowed in R. Otherwise, if R is any commutative ring, then we only know that $\Omega' \leq 1+\omega$ [Abd97, Ber84, Csa76, Lev40], see also [TER98, Mat97]. In order to simplify our complexity estimates, the constant Ω denotes $\max(\Omega', 3)$ in the whole thesis.

II.4 Global Newton Lifting

In this section we present the new global Newton-Hensel iterator. First, through an example, we recall the Newton-Hensel method in its local form and show the slight modification we make in order to globalize it. Then we give a formal description and proof of the method. We apply it in the case of isolated lifting fibers in order to compute lifted curves. In the case $k = \mathbb{Q}$, we present a method to compute a geometric resolution in $\mathbb{Q}$, knowing one over $\mathbb{Z}/p\mathbb{Z}$, for a prime integer p.

II.4.1 Local Newton Iterator

We recall here the classical Newton iterator, along with an example. Let

$$\begin{cases} f_1(x_1,x_2,t) &=& (x_1-1)^2+(x_2-1)^2-4-t-t^2, \\ f_2(x_1,x_2,t) &=& (x_1+1)^2+(x_2+1)^2-4-t. \end{cases}$$

Suppose that we have solved the zero-dimensional system obtained by specializing t to 0. The variable x_1 is a primitive element and we thus have the geometric resolution

$$T^2-1=0, \quad \begin{cases} x_1 = T, \\ x_2 = -T. \end{cases} \tag{II.13}$$

Let $\mathbb{Q}[a]$ be the extension $\mathbb{Q}[T]/(T^2-1)$ of $\mathbb{Q}$. In $\mathbb{Q}[a]$ the point $\mathbf{X}_0 = (a, -a)$ is a solution of the system $f_1 = f_2 = 0$ for $t = 0$. Hence in the formal power series ring $\mathbb{Q}[a][[t]]$, it is a solution of the system at precision $\mathcal{O}(t)$. If the Jacobian matrix of f_1

and f_2 with respect to the variables x_1 and x_2 evaluated at $\mathbf{X}_0$ is invertible, the classical Newton method lifts the solution to a solution at an arbitrary precision by computing the sequence $\mathbf{X}_n$ given by

$$\mathbf{X}_{n+1} := \mathbf{X}_n - J(\mathbf{X}_n)^{-1}\mathbf{f}(\mathbf{X}_n), \qquad n \geq 0.$$

Then $\mathbf{X}_n$ is the solution of the system at the precision $\mathcal{O}(t^{2^n})$. In our example we have

$$\mathbf{X}_2 := \begin{pmatrix} a + \frac{1}{4}\,at + \left(-\frac{1}{8} + \frac{3}{32}\,a\right)t^2 - \frac{3}{128}\,at^3 + \mathcal{O}\left(t^4\right) \\ \\ -a - \frac{1}{4}\,at - \left(\frac{1}{8} + \frac{3}{32}\,a\right)t^2 + \frac{3}{128}\,at^3 + \mathcal{O}\left(t^4\right) \end{pmatrix}.$$

II.4.2 From Local to Global Lifting

The above method allows a local study of the positive dimensional variety in the neighborhood of $t = 0$ but does not lead to a finite representation of a solution of the input system, since the parametrization is given by infinite series over an algebraic extension of $\mathbb{Q}$. The variety $\mathcal{V}(f_1, f_2)$ has the resolution

$$T^2 - 1 - \frac{1}{2}t + \left(\frac{1}{4}T - \frac{1}{4}\right)t^2 + \frac{1}{32}t^4 = 0, \quad \left\{ \begin{array}{l} x_1 = T, \\ x_2 = -T - \frac{1}{4}t^2. \end{array} \right. \tag{II.14}$$

We now show how we perform the lifting on this example. We lift our resolution (II.13) when $t = 0$ step by step to get (II.14).

After the first step of Newton's iterator, when $T^2 - 1 = 0$, $\mathbf{X}_1$ is $(T(1 + t/4 + \mathcal{O}(t^2)), -T(1 + t/4 + \mathcal{O}(t^2)))$. We deduce that $T = x_1(1 - t/4 + \mathcal{O}(t^2))$ and thus

$$x_1^2 - 1 - \frac{1}{2}t + \mathcal{O}(t^2) = 0 \qquad \text{and} \qquad x_2 = -x_1 + \mathcal{O}(t^2),$$

which is the approximation of (II.14) at precision $\mathcal{O}(t^2)$.

We repeat this technique with the new resolution

$$q(T) = T^2 - 1 - \frac{1}{2}t = 0, \quad \left\{ \begin{array}{l} x_1 = T, \\ x_2 = -T. \end{array} \right.$$

We perform another step of Newton's iterator over $\mathbb{Q}[[t]][T]/q(T)$ at the point $(T, -T)$ at precision $\mathcal{O}(t^4)$. We get the following refinement of the parametrization

$$\left\{ \begin{array}{l} x_1 = T + \left(\frac{1}{8}T - \frac{1}{8}\right)t^2 - \frac{1}{16}t^3T + \mathcal{O}(t^4) \\ x_2 = -T + \left(-\frac{1}{8}T - \frac{1}{8}\right)t^2 + \frac{1}{16}t^3T + \mathcal{O}(t^4) \end{array} \right.$$

Thus

$$T = x_1 + \left(\frac{1}{8} - \frac{1}{8}x_1\right)t^2 + \frac{1}{16}x_1t^3 + \mathcal{O}(t^4)$$

and we deduce:

$$T^2 - 1 - \frac{1}{2}t + \left(\frac{1}{4}T - \frac{1}{4}\right)t^2 + \mathcal{O}(t^4) = 0, \quad \left\{ \begin{array}{l} x_1 = T, \\ x_2 = -T - \frac{1}{4}t^2 + \mathcal{O}(t^4). \end{array} \right.$$

ALGORITHM II.1: **Global Newton Iterator**

`GlobalNewton`($\mathbf{f}, \mathbf{x}, u, q, \mathbf{v}$,StopCriterion)

- $\mathbf{x}$ is the list of variables.
- $\mathbf{f}, u, q, \mathbf{v}$ are the ones of (I1), (I2), (I3), (I4) and satisfy (H1), (H2) and (H3).
- StopCriterion is a function returning a boolean. Its arguments are taken from the local variables $\mathbf{f}$, $\mathbf{x}$, u, Q, $\mathbf{V}$ and k below. It returns whether the lifted parametrization $Q(u) = 0$, $\mathbf{x} = \mathbf{V}$ at precision k is sufficient of not.

The procedure returns Q a polynomial and $\mathbf{V}$ as in (O1) and (O2), giving a solution of $\mathbf{f}$ modulo I^{κ}, where κ is implicitely fixed by StopCriterion.

$J \leftarrow$ JacobianMatrix($\mathbf{f}$,$\mathbf{x}$);
$k \leftarrow 1$; $Q \leftarrow q$; $\mathbf{V} \leftarrow \mathbf{v}$;
while not StopCriterion$(\mathbf{f}, \mathbf{x}, u, Q, \mathbf{V}, k)$ **do**
  $k \leftarrow 2k$;
  $\mathbf{V} \leftarrow \mathbf{V} - J(\mathbf{V})^{-1}\mathbf{f}(\mathbf{V}) \bmod Q$;
  $\Delta \leftarrow u(\mathbf{V}) - T$;
  $\mathbf{V} \leftarrow \mathbf{V} - \left(\frac{d\mathbf{V}}{dT}\Delta \bmod Q\right)$;
  $Q \leftarrow Q - \left(\frac{\partial Q}{\partial T}\Delta \bmod Q\right)$;
od;
return$(Q, \mathbf{V})$;

Finally, the next step leads to the resolution

$$T^2 - 1 - \frac{1}{2}t + \left(\frac{1}{4}T - \frac{1}{4}\right)t^2 + \frac{1}{32}t^4 + \mathcal{O}(t^8) = 0,$$

$$\begin{cases} x_1 = T, \\ x_2 = -T - \frac{1}{4}t^2 + \mathcal{O}(t^8), \end{cases}$$

which is the desired resolution, we can remove the $\mathcal{O}(t^8)$. In general, to decide when the lifting is finished, there are two solutions: either we know the required precision in advance, this is the case in §II.4.5, or no *a priori* bound is known, this the case in §II.4.6. In the last case, the only way to decide if the resolution is correct is to check whether the lifting equations vanish on the resolution or not.

II.4.3 Description of the Global Newton Algorithm

Let R be a commutative integral ring, I an ideal of R. We now give a formal presentation of our lifting process passing from a resolution known at precision I to one at precision I^2.

The lifting algorithm takes as input:

(I1) $\mathbf{f} = (f_1, \ldots, f_n)$, n polynomials in $R[x_1, \ldots, x_n]$;

(I2) $u = \lambda_1 x_1 + \cdots + \lambda_n x_n$ a linear form in the x_i, with λ_i in R;

(I3) $q(T)$ a monic polynomial of degree $\delta \geq 1$ in $R[T]$;

(I4) $\mathbf{v} = (v_1(T), \ldots, v_n(T))$, n polynomials of degrees strictly less than δ in $R[T]$.

Let J be the Jacobian matrix of $f_1, \ldots, f_n$ with respect to the variables $x_1, \ldots, x_n$:

$$J_{(i,j)} = \frac{\partial f_i}{\partial x_j}.$$

In $(R/I)[T]/(q(T))$, we make the following assumptions:

(H1) $\mathbf{f}(\mathbf{v}) \equiv \mathbf{0}$;

(H2) $T \equiv u(\mathbf{v})$;

(H3) $J(\mathbf{v})$ is invertible.

Then the following objects exist and we give formulae to compute them:

(O1) Q, a monic polynomial of degree δ, such that $Q \equiv q \bmod R[T]I$;

(O2) $\mathbf{V} = (V_1, \ldots, V_n)$, n polynomials in $R[T]$ of degrees strictly less than δ such that for all i, $1 \leq i \leq n$, we have $V_i \equiv v_i \bmod R[T]I$, and verifying

$$\mathbf{f}(\mathbf{V}) \equiv 0 \text{ and } T \equiv u(\mathbf{V}) \text{ in } (R/I^2)[T]/(Q(T)).$$

The coefficients of Q and V are uniquely determined by the above conditions modulo I^2.
Proof. This process is summarized in Algorithm II.1, the notations being the ones of the end of §II.3.4. The proof divides into two parts and is just the formalization of the computations of §II.4.2.

First we perform a classical Newton step to compute the vector $\mathbf{w} = (w_1, \ldots, w_n)$ of polynomials of degrees strictly less than δ in $R[T]$ such that:

(C) $\mathbf{w} \equiv \mathbf{v} \bmod R[T]I$ and $\mathbf{f}(\mathbf{w}) \equiv 0$ in $(R/I^2)[T]/(q(T))$.

We recall that this can be done by writing the first order Taylor expansion of $\mathbf{f}$ between the points $\mathbf{v}$ and $\mathbf{w}$. The condition (C) implies that:

$$\mathbf{f}(\mathbf{w}) \equiv \mathbf{f}(\mathbf{v}) + J(\mathbf{v}) \cdot (\mathbf{w} - \mathbf{v}) \text{ in } (R/I^2)[T]/(q(T)).$$

According to hypothesis (H3), we deduce the existence and uniqueness of $\mathbf{w}$ modulo $R[T]I$:

$$\mathbf{w} \equiv \mathbf{v} - J(\mathbf{v})^{-1} \cdot \mathbf{f}(\mathbf{v}) \text{ in } (R/I^2)[T]/(q(T)).$$

According to hypothesis (H2) we can write $u(\mathbf{w})$ as

$$u(\mathbf{w}) = T + \Delta(T),$$

where $\Delta(T)$ is a polynomial in $R[T]$ of degree strictly less than δ, with all its coefficients in I.

The second part is a consequence of the following equality between ideals in the quotient ring $R/I^2[T, U, x_1, \ldots, x_n]$:

$$\begin{aligned}&(q(T), U - T - \Delta(T), x_1 - w_1(T), \ldots, x_n - w_n(T))\\ &\quad = (Q(U), T - U + \Delta(U), x_1 - V_1(U), \ldots, x_n - V_n(U)),\end{aligned}$$

where

$$\begin{aligned}Q(U) &= q(U) - (q'(U)\Delta(U) \bmod q(U)),\\ V_i(U) &= w_i(U) - (w_i'(U)\Delta(U) \bmod q(U)), \qquad i = 1, \ldots, n.\end{aligned}$$

□

We now turn to the evaluation of the complexity of Algorithm II.1. Let $\mathsf{a}(h)$ be the cost of the arithmetic operations in R/I^h, where h is a positive integer. Recall that $\mathcal{U}$ is the complexity of the arithmetic operations in $R[T]$ in terms of operations in the base ring R, where R denotes here any commutative ring. Let L be the number of operations required to evaluate $f_1, \ldots, f_n$. Using the notations of §II.3.5, we have the following complexity estimate:

Proposition 7 *According to the above notations, the complexity of Algorithm II.1 returning a solution of $f_1, \ldots, f_n$ at precision I^κ (where κ is a power of 2) is in*

$$\mathcal{O}\left((nL + n^\Omega)\mathcal{U}(\delta) \sum_{j=0}^{\log_2(\kappa)} \mathsf{a}(2^j)\right).$$

Proof. Thanks to [BS83], we only need at most $5L$ operations to evaluate the gradient of a straight-line program of size L. Thus the evaluation of the polynomials $\mathbf{f}$ and the Jacobian matrix J of Algorithm II.1 has complexity $\mathcal{O}(nL)$. Then, the core of the loop requires $\mathcal{O}(n^\Omega)$ operations to compute the inverse of the Jacobian matrix and $\mathcal{O}(n^2)$ other operations to update Q and $\mathbf{V}$, so at step k of the loop $\mathcal{O}(nL + n^\Omega)$ arithmetic operations are done in $R/I^k[T]$ modulo Q. □

In practice there are many possible improvements. An important one consists in taking better care of the precision, for instance to compute the solution at precision $2k$, we just need to know the value of the Jacobian matrix at precision k, since the value of $f_1, \ldots, f_n$ has valuation at least k. Another one can be obtained by inverting the value of the Jacobian matrix by means of a Newton iterator: let J_k be the value of the Jabobian matrix at step k and J_k^{-1} be its inverse then we have $J_{2k}^{-1} = J_k^{-1} + J_k^{-1}(\mathrm{Id}_n - J_{2k}J_k^{-1})$. These techniques are described in [Zip93].

II.4.4 Recovering a Geometric Resolution

Our iterator allows to compute a whole geometric resolution from an isolated lifting fiber. In the frame of §II.3.4, taking $R = k[y_1 - p_1, \ldots, y_r - p_r]$ and $I = (y_1 - p_1, \ldots, y_r - p_r)$ we can apply our iterator with an isolated lifting fiber, in order to lift the parametrization using the lifting equations. But in this case by Propositions 3 and 5 we know that there exists a parametrization of the variety with total degree bounded by $\delta = \deg(\mathcal{V})$, in the form

$$q(T) = 0, \quad \begin{cases} \frac{\partial q}{\partial T} y_{r+1} & = & w_{r+1}(T), \\ & \vdots & \\ \frac{\partial q}{\partial T} y_n & = & w_n(T). \end{cases} \tag{II.15}$$

We can compute q and the w_i in the following way: first we apply our iterator until precision $\delta + 1$ is reached and get a resolution in the form

$$Q(T) + \mathcal{O}(I^{\delta+1}) = 0, \quad \begin{cases} y_{r+1} & = & V_{r+1}(T) + \mathcal{O}(I^{\delta+1}), \\ & \vdots & \\ y_n & = & V_n(T) + \mathcal{O}(I^{\delta+1}). \end{cases}$$

Then let

$$W_i = V_i(T) \frac{\partial Q}{\partial T} \bmod Q(T), \qquad r + 1 \leq i \leq n,$$

the unicity of the geometric resolution lying over the lifting fiber implies that $Q - q, W_{r+1} - w_{r+1}, \ldots, W_n - w_n \in I^{\delta+1}$, whence we deduce q and the w_i.

In practice we are not interested in the lifting of a lifting fiber to its corresponding geometric resolution since it would imply storing multivariate polynomials. Indeed we do not need to lift the fiber over the whole space of the free variables but just over one line containing the lifting point.

II.4.5 Lifted Curves

Let F be an isolated lifting fiber of the variety $\mathcal{V}$ as in §II.3.4, δ its degree and $p' \in k^r$ a point different from p. We are interested in computing the geometric resolution of $\pi^{-1}(D)$, where D denotes the line (pp').

First we notice that the variety $\mathcal{V}_D = \pi^{-1}(D)$ is 1-equidimensional of degree $\delta = \deg(\mathcal{V})$. The restriction $\pi_D : \mathcal{V}_D \to D$ is a finite surjective morphism of degree δ, smooth for $t = 0$. The variety $\mathcal{V}_D = \pi^{-1}(D)$ is called a **lifted curve** of the lifting fiber F.

Let $g_1, \ldots, g_{n-r}$ be the equations of F expressed in the Noether coordinates y_i:

$$g_j = f_j \circ M(y_1, \ldots, y_n)^t.$$

Let also $h_1, \ldots, h_{n-r}$ be the polynomials in $k[t, y_{r+1}, \ldots, y_n]$ defined by:

$$h_i = g_i((p'_1 - p_1)t + p_1, \ldots, (p'_r - p_r)t + p_r, y_{r+1}, \ldots, y_n).$$

ALGORITHM II.2: **Lift Curve**

`LiftCurve`(F, p')

- F is an isolated lifting fiber of dimension r.
- p' is a point in k^r different from the lifting point of F.

The procedure returns the Kronecker parametrization $q, \mathbf{w}$ of the geometric resolution of the lifted curve for the line (pp'), as in §II.4.5.

> $r \leftarrow \dim(F)$;
> $\delta \leftarrow \deg(F)$;
> $\mathbf{g} \leftarrow F_{Equations} \circ F_{ChangeOfVariables}$;
> $\mathbf{h} \leftarrow \mathbf{g}((p'_1 - p_1)t + p_1, \ldots, (p'_r - p_r)t + p_r, y_{r+1}, \ldots, y_n)$;
> StopCriterion $\leftarrow ((k) \mapsto k > \delta)$
> Q,$\mathbf{V}$:=`GlobalNewton`$(\mathbf{h}, [y_{r+1}, ..., y_n], F_{PrimitiveElement},$
> $\quad F_{MinimalPolynomial}, F_{Parametrization}$,StopCriterion$)$;
> $\mathbf{W} \leftarrow [z\frac{\partial Q}{\partial T} \bmod Q : \ z \in \mathbf{V}]$;
> $q \leftarrow$ Truncate$(Q, t^{\delta+1})$;
> $\mathbf{w} \leftarrow$ [Truncate$(z, t^{\delta+1}) : \ z \in \mathbf{W}$];
> **return**$(q, \mathbf{w})$;

From the isolated lifting fiber F we deduce a lifting fiber of $\mathfrak{I}_D$ directly.

Proposition 8 *The variables $t, y_{r+1}, \ldots, y_n$ are in Noether position for $\mathcal{V}_D$, the polynomials h_i define a lifting system for $\mathcal{V}_D$, $t = 0$ is a lifting point and the primitive element of F is primitive for the fiber $t = 0$.*

We can apply the method of the previous section and get the geometric resolution of $\mathcal{V}_D$ in the form

$$q(t,T) = 0, \quad \left\{ \begin{array}{lcl} \frac{\partial q}{\partial T} y_{r+1} & = & w_{r+1}(t,T), \\ & \vdots & \\ \frac{\partial q}{\partial T} y_n & = & w_n(t,T). \end{array} \right. \tag{II.16}$$

This process is summarized in Algorithm II.2.

In order to evaluate the complexity of this algorithm, let L be the number of operations required to evaluate $f_1, \ldots, f_n$ and let the notations be as in §II.3.5. We have the following complexity estimate:

Proposition 9 *Using the above notations and assumptions, the number of operations that Algorithm II.2 performs on elements of R is in*

$$\mathcal{O}((nL + n^{\Omega})\mathcal{U}(\delta)^2).$$

ALGORITHM II.3: **Lifting of Integers**

`LiftIntegers`(F)

- F is an isolated lifting fiber over $\mathbb{Z}/p\mathbb{Z}$.

The procedure returns F', the fiber over $\mathbb{Q}$ lying over F. p must be lucky.

$\delta \leftarrow \deg(F)$;
$\mathbf{f} \leftarrow F_{Equations} \circ F_{ChangeOfVariables}$;
StopCriterion $\leftarrow ((\mathbf{f}, \mathbf{x}, u, Q, \mathbf{V}, k) \mapsto$
  $q \leftarrow$ RationalReconstruction(Q);
  $\mathbf{w} \leftarrow$ RationalReconstruction$([z\frac{\partial q}{\partial T} \bmod q : z \in \mathbf{V}])$;
  if $\mathbf{f}(\mathbf{w}/\frac{\partial q}{\partial T}) \bmod q = 0$ **then**
    $Q \leftarrow q$;$\mathbf{V} \leftarrow \mathbf{w}$;
    return(true);
  else return(false);
  fi;)
$q, \mathbf{w} \leftarrow$ `GlobalNewton`$(\mathbf{f}, \mathbf{x}, F_{PrimitiveElement},$
  $F_{MinimalPolynomial}, F_{Parametrization},$StopCriterion$)$;
$F' \leftarrow F$;
$F'_{MinimalPolynomial} \leftarrow q$;
$F'_{Parametrization} \leftarrow \mathbf{w}$;
return(F');

Proof. We just apply Proposition 7 to the case $\mathtt{a} = \mathcal{U}$. We have to bound the sum:

$$\sum_{j=0}^{\log_2(\kappa)} \mathcal{U}(2^j) \leq \mathcal{U}(\kappa) \sum_{j=0}^{\log_2(\kappa)} 1/2^j \in \mathcal{O}(\mathcal{U}(\kappa)).$$

The precision κ of the last step verifies $\delta < \kappa \leq 2\delta$. Hence $\mathcal{U}(\kappa) \leq \mathcal{U}(2\delta) \in \mathcal{O}(\mathcal{U}(\delta))$. □

Of course, in practice we take κ the biggest power of two less than $\delta+1$, $\kappa \leq \delta+1 < 2\kappa$, we lift C up to precision κ and the last step of the lifting is performed at precision $\delta+1$ only.

II.4.6 Lifting the Integers

We assume here that $k = \mathbb{Q}$. The lifting of the free variables of the previous section can be used for integers as well. If we have a geometric resolution of a zero dimensional variety computed modulo a prime number p we can lift it to precision p^k. If there exists a geometric resolution with rational coefficients lying over the modular one, then the lifting process can stop and we can recover the rational numbers of the geometric resolution.

Here we take $R = \mathbb{Z}$ and $I = p\mathbb{Z}$ where p is a prime number. We assume that we have computed a geometric resolution of a zero dimensional $\mathbb{Q}$-variety in $\mathbb{Z}/p\mathbb{Z}$, that we have $f_1, \ldots, f_n$, n polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$ such that their Jacobian matrix is invertible over the modular resolution, and that the degree of the modular resolution is δ, the degree of the $\mathbb{Q}$-variety. In this case there exists a unique rational geometric resolution lying over the modular one; the lifting process gives the p-adic expansions of its rational coefficients at any required precision.

In [Dix82] Dixon gave a Padé approximant method for integers, see also [HM97] and [Häg98] for related results.

Proposition 10 *[Dix82] Let $s, h > 1$ be integers and suppose that there exist integers f, g such that*

$$gs \equiv f(\text{mod } h) \quad \text{and} \quad |f|, |g| \leq \lambda\sqrt{h},$$

where $\lambda = 0.618\ldots$ is a root of $\lambda^2 + \lambda - 1 = 0$. Let w_i/v_i $(i = 1, 2, \ldots)$ be the convergents to the continued fraction of s/h and put $u_i = v_i s - w_i h$. If k is the least integer such that $|u_k| < \sqrt{h}$, then $f/g = u_k/v_k$.

We assume we have a function called RationalReconstruction computing the unique rational f/g for any s in $\mathbb{Z}/p^k\mathbb{Z}$ with bit complexity in $\mathcal{O}(\mathcal{U}(k\log(p)))$. Such a complexity can obtained combining Dixon's algorithm [Dix82] and a fast Gcd algorithm for integers as discussed in §II.3.5, see [BP94, p.247]. This function returns an error if no such rational number exists. Thus we can stop the lifting when the rational reconstruction of each coefficient of the current resolution leads to a parametrization over $\mathbb{Q}$ of $\mathcal{V}$ satisfying all the equations f_i. This process is summarized in Algorithm II.3.

Proposition 11 *Assume that the geometric resolution lying over the modular one has height at most η with $\log(p) \leq \eta$, then it can be computed in bit complexity*

$$\mathcal{O}((nL + n^\Omega)\mathcal{U}(\delta)\mathcal{U}(\eta)).$$

Proof. We apply Proposition 7 with $\mathsf{a}(k)$ being the bit complexity of the arithmetic in $\mathbb{Z}/p^k\mathbb{Z}$: we can take $\mathsf{a}(k) = \mathcal{U}(k\log(p))$. Choose κ a power of two, such that $4\eta \geq \kappa\log(p) > 2\eta$ and apply Algorithm II.1 until precision $k = \kappa$: since $\sum_{j=0}^{\log_2(\kappa)} \mathcal{U}(\log(p)2^j) \in \mathcal{U}(\kappa\log_2(p))$, then the complexity is in $\mathcal{O}((nL + n^\Omega)\mathcal{U}(\delta)\mathcal{U}(\eta))$. The rational reconstruction for each coefficient of the Kronecker parametrization is in $\mathcal{O}(n\delta\mathcal{U}(\eta))$ □

Theorem 2 is a direct corollary of this lemma.

This result does not give the complexity of Algorithm II.3 because it forgets the verification that the rational reconstructed parametrization satisfies the equations. This verification could be done in $\mathbb{Q}[T]/q(T)$ but it would involve a growth of the size of the integers in the intermediate computations. So, in practice, we prefer to choose another prime number $p' \neq p$ and we perform this verification in $\mathbb{Z}/p'\mathbb{Z}$. The study of the probability of success of this method is out of the scope of this work (see [HM97] and [Häg98] for results related to this question).

II.5 Changing a Lifting Fiber

From any given isolated lifting fiber one can change it to another one, more precisely we can make any linear change of the free variables, or compute a fiber for specialization point or for another primitive element. These three operations on lifting fibers are crucial for the algorithm since it may appear that a given lifting fiber may not be generic enough for computing the intersection of its corresponding variety by a given hypersurface. In this section we assume we are given a lifting fiber with the same notations as in §II.3.4.

II.5.1 Changing the Free Variables

Let $\mathcal{V}$ be a r-equidimensional variety given by an isolated lifting fiber and f be a given polynomial in $k[x_1, \ldots, x_n]$. We are interested in having a Noether position of $\mathcal{V} \cap \mathcal{V}(f)$.

Proposition 12 *Let $\mathcal{V}$ be a r-equidimensional variety of degree δ such that the variables $x_1, \ldots, x_n$ are in Noether position, and f a polynomial in $k[x_1, \ldots, x_n]$ of total degree d such that $\mathcal{V}(f)$ intersects $\mathcal{V}$ regularly. For almost all choices of $(p_1, \ldots, p_{r-1}) \in k^{r-1}$ the change of variables $x_1 = y_1 + p_1 y_r, \ldots, x_{r-1} = y_{r-1} + p_{r-1} y_r, x_r = y_r, \ldots, x_n = y_n$ brings the new coordinates y_i into a Noether position with respect to $\mathcal{V} \cap \mathcal{V}(f)$.*

Proof. Let $\mathfrak{I} = \mathfrak{I}(\mathcal{V})$, x_0 be a new variable, the exponent ^{h} is related to the homogenized objects as in §II.3.1. The ideal $\mathfrak{I}^h$ is in Noether position, let F be an integral dependence relation for f^h given by Proposition 2. Its total degree is bounded by δd and $F(f^h)$ belongs to $\mathfrak{I}^h$. Let $A \in k[x_0, \ldots, x_r]$ be the constant coefficient of F, it belongs to $\mathfrak{I}^h + (f^h)$. Since f intersects $\mathcal{V}$ regularly, F can be chosen such that $A \neq 0$. Let m be the valuation of A with respect to x_0, we define B by A/x_0^m, B is in $(\mathfrak{I}^h + (f^h)) : x_0^\infty$, which is the homogenized ideal of $\mathfrak{I} + (f)$. Let B_0 be the constant coefficient of B with respect to x_0, it is homogeneous and not zero, we can choose a point $p = (p_1, \ldots, p_{r-1})$ in k^{r-1} such that $B_0(p_1, \ldots, p_{r-1}, 1)$ is not zero. Then the change of variables $x_1 = y_1 + p_1 y_r, \ldots, x_{r-1} = y_{r-1} + p_{r-1} y_r, x_r = y_r, \ldots, x_n = y_n$, is such that the new variable y_r is integral over $x_0, y_1, \ldots, y_{r-1}$ for $\mathcal{V} \cap \mathcal{V}(f)$. We deduce that the variables $x_0, y_1 \ldots, y_n$ are in Noether position with respect to $\mathcal{V} \cap \mathcal{V}(f)$. $\square$

The operations to perform such a change of variables are described in Algorithm II.4. Its complexity is in $\mathcal{O}(n^\Omega)$, the complexity of performing linear algebra in dimension n, it is not significative in the whole algorithm.

II.5.2 Changing the Specialization Point

We are now interested in computing a fiber F' on another given specialization point p', assuming that the primitive element of F remains primitive for F'.

We use the method of §II.4.5 to compute the geometric resolution of the lifted curve corresponding to the line (pp') in the form of Equation (II.16). The specialization of this parametrization for $t = 1$ is the one of F'. But in order to get the minimal polynomial we need to take a squarefree part and then reduce the parametrizations. We postone the proof of this last operation in §II.6.5. The method is summarized in Algorithm II.5. Its complexity is the same as in Proposition 9.

ALGORITHM II.4: **Change Free Variables**

`ChangeFreeVariables`(F, p)

- F is a fiber of dimension r.
- p is a point in k^{r-1}.

The procedure performs the linear change of the free variables of F: $y_1 \leftarrow y_1 + p_1 y_r, \dots, y_{r-1} \leftarrow y_{r-1} + p_{r-1} y_r$.

$r \leftarrow \dim(F)$;
$N \leftarrow \mathrm{Id}_n \in \mathrm{SquareMatrix}(n)$;
for i **from** 1 **to** $r-1$ **do**$N[i, r] \leftarrow p_i$; **od**;
$F_{ChangeOfVariables} \leftarrow F_{ChangeOfVariables} \circ N$;
$N \leftarrow \mathrm{SubMatrix}(N, 1..r, 1..r)$;
$F_{LiftingPoint} \leftarrow N^{-1} F_{LiftingPoint}$;

ALGORITHM II.5: **Change Specialization Point**

`ChangeSpecializationPoint`(F, p')

- F is an isolated lifting fiber of dimension r.
- $p' \in k^r$ is a new specialization point, such that $F_{PrimitiveElement}$ remains primitive over p'.

At the end F contains the fiber for p'.

$q, \mathbf{w} \leftarrow$ `LiftCurve`(F, p');
$q, \mathbf{w} \leftarrow \mathrm{subs}(t = 1, q, \mathbf{w})$;
$M \leftarrow \gcd(q, q')$;
$q \leftarrow q/M$; $p \leftarrow q'/M$; $\mathbf{w} \leftarrow \mathbf{w}/M$;
$\mathbf{v} \leftarrow [z/p \bmod q \;:\; z \in \mathbf{w}]$;
$F_{MinimalPolynomial} \leftarrow q$;
$F_{Parametrization} \leftarrow \mathbf{v}$;
$F_{LiftingPoint} \leftarrow p'$;

II.5.3 Changing the Primitive Element

We show how we compute a lifting fiber F' for another given primitive element $u' = \lambda'_{r+1} y_{r+1} + \cdots + \lambda'_n y_n$. The method is summarized in Algorithm II.6.

Algorithm II.6: **Change Primitive Element**

`ChangePrimitiveElement`(F, u')

- F is a lifting fiber of dimension r.
- u' is a lucky new primitive element.

At the end F contains the lifting fiber for u'.

> $q \leftarrow F_{MinimalPolynomial}$;
> $\mathbf{v} \leftarrow F_{Parametrization}$;
> $u \leftarrow F_{PrimitiveElement}$;
> # Let t be a new variable the computations are in $k[t]/(t^2)$.
> $u'_t \leftarrow u' + tu$;
> $U'_t \leftarrow \mathrm{Resultant}_T(q, S - u'_t(\mathbf{v}))$;
> $Q \leftarrow \mathrm{subs}(t = 0, U'_t)$;
> $V \leftarrow -\mathrm{Coefficient}(U'_t, t)/Q' \bmod Q$;
> $\mathbf{V} \leftarrow [z(V) \bmod Q \ : \ z \in \mathbf{v}]$;
> $F_{MinimalPolynomial} \leftarrow Q$;
> $F_{Parametrization} \leftarrow \mathbf{V}$;
> $F_{PrimitiveElement} \leftarrow u'$;

Let t be a new variable, we extend the base field k to the rational function field $k_t = k(t)$. Let $u'_t = u' + tu$ and $\mathfrak{I}_t$ the extension of $\mathfrak{I}$ in k_t. We can compute the characteristic polynomial U'_t of u'_t such that $U'_t(u'_t) \in \mathfrak{I}_t$ and deduce the Kronecker parametrization of $\mathfrak{I}_t$ with respect to u_t in the same way as in §II.3.3. The characteristic polynomial can be computed by means of a resultant:

$$U'_t(S) = \mathrm{Resultant}_T(q(T), S - u'_t(v_{r+1}, \ldots, v_n)).$$

In order to get the new parametrization, we only need to know the first order partial derivative with respect to t at the point $t = 0$. So the resultant can be computed modulo t^2. If we use a resultant algorithm performing no division on its base ring, this specialization over the non-integral ring $k[t]/(t^2)[S]$ does not create any problem.

A problem comes from the fact that we are interested in using resultant algorithms for integral rings since they have better complexity. In order to explain how this can work under some genericity conditions, we come back to the notations of §II.3.3. Then we take u' generic: $u_\Lambda = \Lambda_{r+1}x_{r+1} + \cdots + \Lambda_n x_n$, the Λ_i being new variables, we can compute

$$U_\Lambda(S) = \mathrm{Resultant}_T(q(T), S - u_\Lambda(v_{r+1}, \ldots, v_n)),$$

in the integral ring $k[\Lambda_{r+1}, \ldots, \Lambda_n][S]$. Let Φ be the ring morphism of specialization:

$$\begin{array}{rrcl} \Phi: & k[\Lambda_{r+1}, \ldots, \Lambda_n][S] & \to & k[t]/(t^2)[S], \\ & \Lambda_i & \mapsto & \lambda'_i + t\lambda_i \end{array}$$

If u' is chosen generic enough the specialization Φ commutes with the resultant computation. The justification of this fact is based on the remark that the specialization commutes when all the equality tests on elements of $k[t]/(t^2)$ can be done on the coefficients of valuation 0 and give the same answer as the corresponding test in $k[\Lambda_{r+1},\ldots,\Lambda_n]$. The λ'_i for which this condition does not apply satisfy algebraic equations in $k[\Lambda_{r+1},\ldots,\Lambda_n]$. A choice of u' such that the specialization Φ commutes with a given resultant algorithm is said to be **lucky** for this computation. One can find in [GCL94, §7.4] a systematic discussion about this question.

In order to estimate the complexity of this method, recall that $\mathcal{U}(\delta)$ is the complexity of the resultant of two univariate polynomials of degrees at most δ in terms of arithmetic operations in the base ring and also the complexity of the arithmetic operations on univariate polynomials of degree δ, as in §II.3.5.

Proposition 13 *Let u' be a lucky primitive element for Algorithm II.6, then the complexity of Algorithm II.6 is in $\mathcal{O}(n\delta\mathcal{U}(\delta))$.*

Proof. In the resultant computation of U'_t the variable S if free thus its specialization commutes with the resultant. The degree of U'_t in S is δ. So we can compute U'_t for $\delta+1$ distinct values of S and interpolate in k the polynomials q' and v'. The cost of interpolation in degree δ is in $\mathcal{U}(\delta)$ [BP94, p. 25].

Then the computation of $\mathbf{v}'$ requires to compute the powers $v^2,\ldots,v^{\delta-1}$ modulo q', this involves a cost in $\mathcal{O}(\delta\mathcal{U}(\delta))$. Finally we perform n linear combinations of these powers, which takes $\mathcal{O}(n\delta^2)$ operations. □

II.6 Computation of an Intersection

We show in this section how we compute an isolated lifting fiber of the intersection by a hypersurface of a r-equidimensional variety given an isolated lifting fiber. We use Kronecker's method: when performing an elimination, the parametrization of the coordinates are given at the same time as the eliminating polynomial. The computational trick consists in a slight change of variables called Liouville's substitution [Mac16, p.15] and the use of first order Taylor expansions.

Example 6 Suppose we want a geometric resolution of two equations f_1 and f_2, intersecting regularly, in $k[x_1,x_2]$. Let Λ_1 and Λ_2 be new variables and $u_\Lambda = \Lambda_1x_1+\Lambda_2x_2$. We can compute $U_\Lambda(T)$, the eliminating polynomial of u_Λ:

$$U_\Lambda(T) = \mathrm{Resultant}_{x_1}\Big(f_1(x_1,\frac{T-\Lambda_1x_1}{\Lambda_2}), f_2(x_1,\frac{T-\Lambda_1x_1}{\Lambda_2})\Big).$$

The expression $U_\Lambda(u_\Lambda)$ belongs to the ideal (f_1,f_2), and f_1, f_2 have a common root if and only if $U_\Lambda(u_\Lambda)$ vanishes. Taking the first derivatives in the Λ_i we deduce that

$$\frac{\partial U_\Lambda}{\partial T}x_1 + \frac{\partial U_\Lambda}{\partial \Lambda_1} \in (f_1,f_2),$$

and

$$\frac{\partial U_\Lambda}{\partial T}x_2 + \frac{\partial U_\Lambda}{\partial \Lambda_2} \in (f_1, f_2).$$

If U_Λ is square free, then the common zeros of f_1 and f_2 are parameterized by

$$U_\Lambda(T) = 0, \quad \left\{ \begin{array}{rcl} \frac{dU_\Lambda}{dT}(T)x_1 & = & -\frac{dU_\Lambda}{d\Lambda_1}(T), \\ \frac{dU_\Lambda}{dT}(T)x_2 & = & -\frac{dU_\Lambda}{d\Lambda_2}(T). \end{array} \right. \tag{II.17}$$

For almost all values λ_1, λ_2 in k of Λ_1, Λ_2, the specialization of (II.17) gives a geometric resolution of f_1, f_2. So, letting $\Lambda_i = \lambda_i + t_i$, in order to get a geometric resolution we only need to know U_Λ at precision $\mathcal{O}((t_1, t_2)^2)$.

Our aim is to generalize the method of this example for the intersection of a lifted curve with an hypersurface.

Let $\mathfrak{I}$ be a 1-equidimensional radical ideal in $k[y, x_1, \ldots, x_n]$ such that the variables $y, x_1, \ldots, x_n$ are in Noether position and assume that we have a geometric resolution in the form

$$q(y,T) = 0, \quad \left\{ \begin{array}{rcl} \frac{dq}{dT}(y,T)x_1 & = & w_1(y,T), \\ & \vdots & \\ \frac{dq}{dT}(y,T)x_n & = & w_n(y,T). \end{array} \right. \tag{II.18}$$

The variable T represents the primitive element u. Let f be a given polynomial in the ring $k[y, x_1, \ldots, x_n]$ intersecting $\mathfrak{I}$ regularly, which means that $\mathfrak{I} + (f)$ is 0-dimensional. We want to compute a geometric resolution of $\mathfrak{I} + (f)$.

II.6.1 Characteristic Polynomials

In the situation above one can easily compute an eliminating polynomial in the variable y, using any elimination process. First we invert q' modulo q and compute

$$v_i(y,T) = w_i(y,T)q'^{-1}(y,T) \bmod q(y,T), \text{ for } 1 \leq i \leq n.$$

The elimination process we use is given in the following:

Proposition 14 *The characteristic polynomial of the endomorphism of multiplication by f in $B' = k(y)[x_1, \ldots, x_n]/\mathfrak{I}$ belongs to $k[y][T]$ and its constant coefficient with respect to T is given by*

$$A(y) = \mathrm{Resultant}_T(q, f(y, v_1, \ldots, v_n)),$$

up to its sign. Moreover the set of roots of $A(y)$ is exactly the set of values of the projection on the coordinate y of the set of roots of $\mathfrak{I} + (f)$.

Proof. We already know from Corollary 2 that A belongs to $k[y]$ and has degree bounded by $\deg(f)\delta$, $\delta = \deg(\mathcal{V})$. Let π be the finite projection onto the coordinate y. Let y_0 be a point of $\overline{k}$ and $\{Z_1, \dots, Z_s\} = \pi^{-1}(y_0)$ of respective multiplicity $m_1, \dots, m_s$, $s \leq \delta$ and $m_1 + \cdots + m_s = \delta$, where the multiplicity of Z_i is defined as $m_i = \dim_k(k[y, x_1, \dots, x_n]/(\mathfrak{I} + (y - y_0))_{Z_i})$. First we prove that

$$A(y) \in \mathfrak{I} + (f), \tag{II.19}$$

which implies that any root of $\mathfrak{I} + (f)$ cancels A, and then the formula

$$A(y_0) = \prod_{j=1}^{s} f(Z_j)^{m_j}, \tag{II.20}$$

which implies that when y_0 annihilates A at least one point in the fiber annihilates f.

The ideal $\mathfrak{I}$ being 1-equidimensional and the variables being in Noether position, the finite $k[y]$-module $B = k[y, x_1, \dots, x_n]/\mathfrak{I}$ is free of rank δ (combine [Kun85, Example 2, p.187] and the proof of [GHS93, Lemma 3.3.1] or [AS95, Lemma 5]). Since any basis of B induces a basis for $B' = k(y) \otimes B$, the characteristic polynomials of the endomorphism of multiplication by f in B and B' coincide, Cayley-Hamilton theorem applied in B implies (II.19).

For the formula (II.20), let $B_0 = \overline{k}[y, x_1, \dots, x_n]/(\mathfrak{I} + (y - y_0))$, B_0 is a $\overline{k}$-vector space of dimension δ. Let $e_1, \dots, e_\delta$ be a basis of B, their specialization for $y = y_0$ leads to a set of generators of B_0 of size δ thus it is a basis of B_0. We deduce that $A(y_0)$ is the constant coefficient of the characteristic polynomial of the endomorphism of multiplication by f in B_0, whence formula (II.20). □

From a computational point of view, the variable y belongs to $k(y)$ and if we take $p \in k$ such that the denominators of the v_i do not vanish at p we can perform the computation of the resultant in $k[[y - p]]/((y - p)^{\delta d + 1})$, since A has degree at most δd. This method works well if we use a resultant algorithm performing no test and no division. So we are in the same situation as in §II.5.3, we want to use an algorithm with tests and divisions in order to get a better complexity, and this is possible if p is generic enough. The values of p for which this computation gives the good result are said to be *lucky*. Unlucky p are contained in a strict algebraic closed subset of k. In Algorithm II.7 we suppose that the last coordinate of the lifting point is lucky.

As in §II.3.5, $\mathcal{U}$ denotes respectively the complexity of univariate polynomial arithmetic and the resultant computation.

Lemma 1 *Let L be the complexity of evaluation of f, d the total degree of f and δ the degree of q, then $A(y)$ can be computed in $\mathcal{O}((L + n^2)\mathcal{U}(\delta)\mathcal{U}(d\delta))$ arithmetic operations in k.*

Proof. Let $p \in k$ be generic enough, we perform the computation with y in $k[[y - p]]$ at precision $\mathcal{O}((y - p)^{d\delta + 1})$. First we have to compute each v_i from the w_i, this is done by performing an extended GCD between q and $\frac{dq}{dT}$. The cost of the extended GCD is the same as $\mathcal{U}$. Then we evaluate f modulo q, and perform the resultant computation, whence the complexity. □

II.6.2 Liouville's Substitution

We are now facing two questions: first the variable y is probably not a primitive element of $\sqrt{\mathfrak{I}+(f)}$, so we are looking for an eliminating polynomial of $\lambda y+u$ and secondly we want the parametrization of the coordinates with respect to the linear form $\lambda y+u$, for the same cost. Liouville's substitution answers both problems, for almost all $\lambda \in k$.

Let Y be a new variable, the substitution consists in replacing y by $(Y-T)/\lambda$ in both the parametrization and the polynomials of the ideal $\mathfrak{I}$. So we need some more notations: let $q_Y(Y,T) = q((Y-T)/\lambda, T)$, $p_Y(Y,T) = q'((Y-T)/\lambda, T)$, $w_{Y,i}(Y,T) = w_i((Y-T)/\lambda, T)$, $1 \leq i \leq n$ and $\mathfrak{I}_Y = (e((Y-T)/\lambda, T),\ e \in \mathfrak{I})$. In order to apply Proposition 14, we must ensure that the parametrization of $\mathfrak{I}_Y$ we get is still valid. Indeed, this is true for almost all $\lambda \in k$.

A point λ is said to be a **Liouville point** with respect to the above geometric resolution of $\mathfrak{I}$ when it is not zero, and when q_Y is monic in T, of the same degree as q in T, square-free and relatively prime with p_Y.

Lemma 2 *With the above notations, if λ is a Liouville point then the variables $Y, x_1, \ldots, x_n$ are in Noether position with respect to $\mathfrak{I}_Y$ and*

$$q_Y(Y,T)=0, \quad \left\{ \begin{array}{rcl} p_Y(Y,T)x_1 & = & w_{Y,1}(Y,T), \\ & \vdots & \\ p_Y(Y,T)x_n & = & w_{Y,n}(Y,T), \end{array} \right. \tag{II.21}$$

is a geometric resolution of $\mathfrak{I}_Y$ for the primitive element u.

Proof. First we prove that Y is free in $\mathfrak{I}_Y$. Let $h \in k[Y]$ such that $h(Y) \in \mathfrak{I}_Y$. This implies that $q(y,T)$ divides $h(\lambda y+T)$ and so q_Y divides $h(Y)$. Since q_Y is monic in T this implies that $h=0$.

Now we prove that the x_i are dependent over Y. Let $\mathfrak{J} = \mathfrak{I}_Y + (T-u) \subset k[Y, x_1, \ldots, x_n, T]$ and h a bivariate polynomial such that $h(y, x_i) \in \mathfrak{I}$ is monic in x_i, of total degree bounded by $\deg_{x_i}(h)$. We have $h((Y-T)/\lambda, x_i) \in \mathfrak{J}$, and since $q_Y(Y,T) \in \mathfrak{J}$ has a total degree bounded by δ, there exists a polynomial H such that $H(Y, x_i) \in \mathfrak{J}$, monic in x_i, of total degree bounded by its partial degree in x_i. We deduce that $Y, x_1, \ldots, x_n$ are in Noether position with respect to $\mathfrak{I}_Y$.

The conditions that q_Y is square-free and relatively prime with p_Y imply that u remains a primitive element and that we can invert p_Y modulo q_Y. The parametrization of (II.21) is a geometric resolution of $\mathfrak{I}_Y$. $\square$

Lemma 3 *Almost all elements $\lambda \in k$ are Liouville points.*

Proof. We write the proof replacing λ by $1/\lambda$ and then Y by λY, thus q_Y becomes $q(Y-\lambda T, T)$. The discriminant of q_Y and the resultant of q_Y with p_Y are now polynomials in λ and Y and do not vanish for $\lambda = 0$. Hence almost all choices of λ satisfy the last two conditions of the definition of a Liouville point given just above. For the first one, let us consider $h(y,T)$, the homogeneous part of q of maximal degree δ, then the coefficient of T^δ in q_Y is $h(-\lambda, 1)$, which does not vanish when $\lambda = 0$. $\square$

ALGORITHM II.7: **Kronecker Intersection**

`KroneckerIntersect`(C, λ, f)

- C is a geometric resolution of $\mathfrak{I}$, 1-equidimensional, with a first order generic parametrization.
- λ is a Liouville point for C.
- f is a polynomial.

The procedure returns the constant coefficient of the characteristic polynomial of the endomorphism of multiplication by f in $k[y, x_1, \ldots, x_n]/\mathfrak{I}$.

$q \leftarrow C_{MinimalPolynomial}$;
$\mathbf{w} \leftarrow C_{Parametrization}$;
$q_Y \leftarrow q((Y-T)/\lambda, T)$;
$p_Y \leftarrow \frac{dq}{dT}((Y-T)/\lambda, T)$;
$\mathbf{w}_Y \leftarrow \mathbf{w}((Y-T)/\lambda, T)$;
$\mathbf{v}_Y \leftarrow \mathbf{w}_Y p_Y^{-1} \bmod q_Y$;
$A \leftarrow \mathrm{Resultant}_T(q_Y, f((Y-T)/\lambda, \mathbf{v}_Y))$;
return(A);

Lemma 4 *Let $\lambda \in k\setminus\{0\}$ and p be a polynomial in $k[y,T]$ of total degree bounded by δ and stored in a two dimensional array of size $\mathcal{O}(\delta^2)$. The polynomial $p_Y(Y,T) = p((Y-T)/\lambda, T) \in k[Y,T]$, can be computed in $\mathcal{O}(\delta\mathcal{U}(\delta))$ arithmetic operations in k.*

Proof. We can write $p = p_0 + p_1 + \cdots + p_\delta$, where each p_i is homogeneous of degree i. So we can suppose that p is homogeneous of degree i. To compute $p_Y(Y,T) = p((Y-T)/\lambda, T)$ we first note that since p_Y is homogeneous of degree i we just compute $p_Y(Y,1) = p((Y-1)/\lambda, 1)$. But $p(y,1)$ is a polynomial in $k[y]$ in which we have to perform a linear transformation. We refer to [BP94, pp. 15–16]: the cost of the linear substitution is $\mathcal{U}(i)$. Thus the sum of complexities for each i is in $\mathcal{O}(\sum_{i=0}^{\delta}\mathcal{U}(i)) \subset \mathcal{O}(\delta\mathcal{U}(\delta))$. □

II.6.3 Computing the Parametrization

Combining the two previous sections, we are now able to describe the core of our intersection method, which is summarized in Algorithm II.7.

Proposition 14 and Lemma 2 lead to:

Proposition 15 *If λ is a Liouville point for the given geometric resolution of $\mathfrak{I}$, then the polynomial A returned by Algorithm II.7 applied on $\mathfrak{I}_Y$ and f_Y satisfies*

$$A(\lambda y + u) \in \mathfrak{I}$$

and its set of roots is exactly the set of values of the linear form $\lambda y + u$ on the points of $\mathfrak{I} + (f)$.

Proof. From Proposition 14 we have $A(Y) \in \mathfrak{I}_Y + (f_Y)$ and over each root of A lies a zero of $\mathfrak{I}_Y + (f_Y)$. Replacing Y by $\lambda y + u$ leads to $A(\lambda y + u) \in \mathfrak{I}$ and a zero $(z_Y, z_1, \ldots, z_n)$ of $\mathfrak{I}_Y$ lying over z_Y, a root of A, induces a zero of $\mathfrak{I}$, namely $((z_Y - u(z_1, \ldots, z_n))/\lambda, z_1, \ldots, z_n)$. □

This is not sufficient to describe the points of $\mathfrak{I} + (f)$: the parametrization of the coordinates are still missing. Let $t_y, t_1, \ldots, t_n$ be new variables and $k_t = k(t_y, t_1, \ldots, t_n)$, let $\mathfrak{I}_t$ be the extension of $\mathfrak{I}$ in k_t and $u_t = u + t_1 x_1 + \cdots + t_n x_n$, we assume that we have the geometric resolution of $\mathfrak{I}_t$ with respect to u_t:

$$q_t(y,T) = 0, \quad \begin{cases} x_1 & = & v_{t,1}(y,T), \\ & \vdots & \\ x_n & = & v_{t,n}(y,T). \end{cases} \tag{II.22}$$

If λ is a Liouville point for $\mathfrak{I}$ then $\lambda + t_y$ is a Liouville point for $\mathfrak{I}_t$. So we can apply Algorithm II.7 in this situation, we get a polynomial $A_t \in k_t[T]$ such that $A_t((\lambda + t_y)y + u) \in \mathfrak{I}_t$ and we can write

$$A_t = A + t_y A_y + t_1 A_1 + \cdots + t_n A_n + \mathcal{O}((t_y, t_1, \ldots, t_n)^2),$$

where A, A_y and the A_i are polynomials over k. We deduce that

$$A(\lambda y + u), \quad A'(\lambda y + u)y + A_y(\lambda y + u), \quad A'(\lambda y + u)x_i + A_i(\lambda y + u),$$

$1 \le i \le n$, belong to $\mathfrak{I}$. The computation has to be handled at precision $\mathcal{O}((t_y, t_1, \ldots, t_n)^2)$ only, so we are faced with the same problem as in §II.5.3: if we use a resultant algorithm without division there is no difficulty, but if we want to benefit from the better complexity of an algorithm for an integral ring we have to make some genericity restriction on the choices of u and λ. We will also speak about *lucky* choices for Algorithm II.7. We call the parametrization (II.22) at precision $\mathcal{O}((t_y, t_1, \ldots, t_n)^2)$ the *first order generic parametrization* associated to parametrization (II.18).

Proposition 16 *With lucky u and λ, Algorithm II.7 has complexity in*

$$\mathcal{O}(n(L + n^2)\mathcal{U}(\delta)\mathcal{U}(d\delta)),$$

in terms of number of arithmetic operations in k.

Proof. This is a direct consequence of Lemma 1 replacing k by $k[t_y, t_1, \ldots, t_n]/(t_y, t_1, \ldots, t_n)^2$. The n Liouville's substitutions are insignificant. □

In §II.6.5 we explain how to deduce a geometric resolution from A, A_y and the A_i.

II.6.4 Lifting a First Order Genericity

Now, we have to explain how to compute the first order generic parametrization (II.22) from (II.18), that we use in the previous section.

The ideal $\mathfrak{I}$ is given by the geometric resolution of equations (II.18). Let $B_t = k_t \otimes B$, in B_t we have $x_i = v_i(y,u)$ so $u_t = u + t_1 v_1(y,u) + \cdots + t_n v_n(y,u)$. But at the first order in the t_i we have $u_t = u + t_1 v_1(y,u_t) + \cdots + t_n v_n(y,u_t) + \mathcal{O}((t_y,t_1,\ldots,t_n)^2)$, we deduce that $u = u_t - (t_1 v_1(y,u_t) + \cdots + t_n v_n(y,u_t)) + \mathcal{O}((t_y,t_1,\ldots,t_n)^2)$. We can replace u in the parametrization:

$$\begin{aligned} q_t(y,T) &= q(y,T) - (\frac{dq}{dT}(y,T)(t_1 v_1(y,T) + \cdots + t_n v_n(y,T)) \bmod q(y,T)) \\ &= q(y,T) - (t_1 w_1(y,T) + \cdots + t_n w_n(y,T)) + \mathcal{O}((t_y,t_1,\ldots,t_n)^2) \end{aligned}$$

$$\begin{aligned} v_{t,i}(y,T) &= v_i(y,T) - \frac{dv_i}{dT}(y,T)(t_1 v_1(y,T) + \cdots + t_n v_n(y,T)) \\ & \quad \bmod q_t(y,T) + \mathcal{O}((t_y,t_1,\ldots,t_n)^2), \qquad 1 \le i \le n. \end{aligned}$$

Computations are summarized in Algorithm II.8. As in the previous subsection we perform the computations in

$$k_{y,t} = k[y,t_y,t_1,\ldots,t_n]/((y-p)^{d\delta+1} + (t_y,t_1,\ldots,t_n)^2),$$

with a lucky choice of p in order to inverse $\frac{dq}{dT}$ modulo q with an extended GCD algorithm of complexity $\mathcal{U}(\delta)$. In this situation we have the following complexity estimate:

Proposition 17 *Algorithm II.8 has complexity in $\mathcal{O}(n^2\mathcal{U}(\delta)\mathcal{U}(d\delta))$, in terms of number of arithmetic operations in k.*

Proof. The arithmetic operations in $k_y = k[y]/(y-p)^{d\delta+1}$ have cost in $\mathcal{O}(\mathcal{U}(d\delta))$ in terms of arithmetic operations in k. The computation of $\mathbf{v}$ requires $\mathcal{O}(n\mathcal{U}(\delta))$ in k_y. Then the computation of $\mathbf{v}_t$ requires $\mathcal{O}(n)$ operations in $k_{t,y}/(q_t)$, this is in $\mathcal{O}(n^2\mathcal{U}(d\delta)\mathcal{U}(\delta))$. □

II.6.5 Removing the Multiplicities

The output of Algorithm II.7 is not yet a parametrization of the roots of $\mathfrak{I} + (f)$: it may happen that A_0 has multiplicities. We give a simple method to remove them and thus get a geometric resolution of $\sqrt{\mathfrak{I} + (f)}$.

Assume now that $\mathfrak{I}$ is 0-dimensional, that we have a primitive element $u = \lambda_1 x_1 + \cdots + \lambda_n x_n$ of $\mathcal{V} = \mathcal{V}(\mathfrak{I})$ and that at precision $\mathcal{O}((t_1,\ldots,t_n)^2)$ we have an eliminating polynomial A_t of $u_t = u + t_1 x_1 + \cdots + t_n x_n$, coming from Algorithm II.7, such that

$$A_t(u_t) \in \mathfrak{I}_t + \mathcal{O}((t_1,\ldots,t_n)^2),$$

and the roots of A_t are the values of u_t over the points of $\mathcal{V}$. Let $Z_1,\ldots,Z_\delta$ be the points of $\mathcal{V}$, then for some integers $m_i > 0$ we have

$$A_t(T) = \prod_{j=1}^{\delta} (T - u_t(Z_j))^{m_j} + \mathcal{O}((t_1,\ldots,t_n)^2).$$

ALGORITHM II.8: **Lift First Order Genericity**

`LiftFirstOrderGenericity`(C)

- C is a geometric resolution of $\mathfrak{I}$, one equidimensional.

The procedure returns a geometric resolution C' of $\mathfrak{I}_t$ for the primitive element $u_t = u + t_1 x_1 + \cdots t_n x_n$, at precision $\mathcal{O}((t_1, \ldots, t_n)^2)$.

$C' \leftarrow C$;
$q \leftarrow C_{MinimalPolynomial}$;
$\mathbf{w} \leftarrow C_{Parametrization}$;
$q_t \leftarrow q - (t_1 w_1 + \cdots + t_n w_n)$;
$\mathbf{v} \leftarrow (\frac{dq}{dT})^{-1} \mathbf{w} \bmod q$;
$\mathbf{v_t} \leftarrow \mathbf{v} - \frac{d\mathbf{v}}{dT}(t_1 v_1 + \cdots + t_n v_n) \bmod q_t$;
${C'}_{MinimalPolynomial} \leftarrow q_t$;
${C'}_{Parametrization} \leftarrow \mathbf{v_t}$;
${C'}_{PrimitiveElement} \leftarrow C_{PrimitiveElement} + t_1 x_1 + \cdots + t_n x_n$;
$\mathbf{return}(C')$;

Now if we write $A_t = A_0 + t_1 A_1 + \cdots + t_n A_n + \mathcal{O}((t_1, \ldots, t_n)^2)$, with A_i polynomials in $k[T]$ we have:

$$A_0(T) = \prod_{j=1}^{\delta} (T - u(Z_j))^{m_j},$$

$$A_i(T) = -\sum_{i=1}^{\delta} \Big(x_i(Z_j) m_i (T - u(Z_i))^{m_i - 1} \prod_{j=1, j \neq i}^{\delta} (T - u(Z_j))^{m_j} \Big), \quad 1 \leq i \leq n.$$

We deduce the following proposition:

Proposition 18 *With the above notations, let $M = \gcd(A_0, A_0')$ then M divides A_0, A_0' and the A_i. Let $q = A_0/M$, $p = A_0'/M$ and $w_i = -A_i/M$, $1 \leq i \leq n$, then*

$$q(u) = 0, \quad \begin{cases} p(u) x_1 & = & w_1(u), \\ & \vdots & \\ p(u) x_n & = & w_n(u), \end{cases} \tag{II.23}$$

is a geometric resolution of $\mathcal{V}$.

This process is summarized in Algorithm II.9.

Proposition 19 *Let δ be the degree of A_t in T then the complexity of Algorithm II.9 is in*

$$\mathcal{O}(n\mathcal{U}(\delta)),$$

in terms of arithmetic operations in k.

ALGORITHM II.9: **Remove Multiplicity**

`RemoveMultiplicity`(A_t)

- A_t is an annihilating polynomial of a primitive element u_t modulo $\mathfrak{I}$, coming from Algorithm II.7.

The procedure returns $q, \mathbf{v}$, a parametrization of $\mathcal{V}(\mathfrak{I})$ for the primitive element u.

\# We write $A_t = A_0 + t_1 A_1 + \cdots + t_n A_n + \mathcal{O}((t_1, \ldots, t_n)^2)$.
$M \leftarrow \gcd(A_0, A_0')$;
$q \leftarrow A_0/M$;
$p \leftarrow A_0'/M$;
$\mathbf{w} \leftarrow [-A_1/M, \ldots, -A_m/M]$;
$\mathbf{v} \leftarrow \mathbf{w}/p \bmod q$;
return$(q, \mathbf{v})$;

Note that this method to remove the multiplicity does not work for any kind of parametrization. For example, consider $\mathfrak{I} = (x_1^2, x_2^2)$, x_1 is a primitive element and we have $x_1^4 \in \mathfrak{I}$ and $4x_1^3x_2 - x_1^2 \in \mathfrak{I}$, but x_1^3 does not divide x_1^2.

II.6.6 Removing the Extraneous Components

Let $\mathcal{V}$ be a 0-dimensional variety given by a geometric resolution:

$$q(u) = 0, \quad \left\{ \begin{array}{lcl} x_1 & = & v_1(u), \\ & \vdots & \\ x_n & = & v_n(u). \end{array} \right. \tag{II.24}$$

Let g be a given polynomial in $k[x_1, \ldots, x_n]$, we are interested in computing a geometric resolution of $\mathcal{V} \backslash \mathcal{V}(g)$. The computations are presented in Algorithm II.10.

Proposition 20 *The parametrization*

$$Q(u) = 0, \quad \left\{ \begin{array}{lcl} x_1 & = & V_1(u), \\ & \vdots & \\ x_n & = & V_n(u), \end{array} \right. \tag{II.25}$$

returned by Algorithm II.10 is a geometric resolution of $\mathcal{V} \backslash \mathcal{V}(g)$.

Proposition 21 *Let L be the complexity of evaluation of g, Algorithm II.10 has a complexity in*

$$\mathcal{O}((L + n^2)\mathcal{U}(\delta)),$$

in terms of arithmetic operations in k.

ALGORITHM II.10: **Cleaning**

`Clean`(F, g)

- F is a geometric resolution of dimension 0.
- g is a polynomial.

At the end F contains a geometric resolution for the variety composed of the points outside $g = 0$.

$q \leftarrow F_{MinimalPolynomial}$;
$\mathbf{v} \leftarrow F_{Parametrization}$;
$e \leftarrow \mathrm{Gcd}(q, g(\mathbf{v}))$;
$q \leftarrow q/e$;
$F_{MinimalPolynomial} \leftarrow q$;
$F_{Parametrization} \leftarrow \mathbf{v} \bmod q$;

In order to apply this method in the situation of a lifting fiber we must ensure that the choice of the lifting point is not too bad.

> **Example 7** In $k[x_1, x_2]$, $\mathcal{V} = \mathcal{V}(x_2)$ and $g = x_1$, the choice of $x_1 = 0$ as a lifting point is not a proper choice to compute $\mathcal{V} \backslash \mathcal{V}(g)$.

We now show that almost all choices are correct. Let $\mathcal{V}$ be a r-equidimensional variety given by a lifting fiber, it is sufficient to take the lifting point p of the fiber outside $\pi(\overline{\mathcal{V} \backslash \mathcal{V}(g)} \cap \mathcal{V}(g))$, since then

$$\overline{\mathcal{V} \backslash \mathcal{V}(g)} \cap (x_1 - p_1, \ldots, x_r - p_r) = \mathcal{V} \cap (x_1 - p_1, \ldots, x_r - p_r) \backslash \mathcal{V}(g).$$

A lifting point is said to be a **cleaning point** with respect to the polynomial g when $p \notin \pi(\overline{\mathcal{V} \backslash \mathcal{V}(g)} \cap \mathcal{V}(g))$.

Proposition 22 *The lifting points that are not cleaning points are enclosed in an algebraic closed set.*

Proof. The hypersurface $g = 0$ intersects regularly $\overline{\mathcal{V} \backslash \mathcal{V}(g)}$. This intersection has dimension $r - 1$, the closure of its projection is a strict algebraic subset of k^r. □

II.6.7 Summary of the Intersection

We are now able to put §II.6.3, §II.6.4, §II.6.5 and §II.6.6 together in order to compute a geometric resolution of $\sqrt{\mathfrak{I} + (f)}$. The whole process of intersection is summarized in Algorithm II.11.

ALGORITHM II.11: **One Dimensional Intersect**

`OneDimensionalIntersect`(C, f, λ, g)

- C is a geometric resolution of $\mathfrak{I}$, 1-equidimensional.
- f is a polynomial intersecting C regularly.
- λ is a Liouville point of C. Let u be the primitive element of C, $\lambda y + u$ is a primitive element of $\sqrt{\mathfrak{I} + (f)}$, g is a polynomial.

The procedure returns F, a geometric resolution of $\mathcal{V}(\mathfrak{I} + (f)) \backslash \mathcal{V}(g)$.

$C_t \leftarrow$ `LiftFirstOrderGenericity`(C);
$A_t \leftarrow$ `KroneckerIntersect`(C_t, f, λ);
$q, \mathbf{v} \leftarrow$ `RemoveMultiplicity`(A_t);
$F_{ChangeOfVariables} \leftarrow C_{ChangeOfVariables}$;
$F_{PrimitiveElement} \leftarrow \lambda y + C_{PrimitiveElement}$;
$F_{MinimalPolynomial} \leftarrow q$;
$F_{Parametrization} \leftarrow \mathbf{v}$;
$F_{Equations} \leftarrow C_{Equations}, f$;
`Clean`(F, g);
return(F);

Proposition 23 *Let C be a geometric resolution of a 1-equidimensional ideal $\mathfrak{I}$, u its primitive element, and $\lambda \in k$ a Liouville point for C such that $v = \lambda y + u$ is a primitive element of $\sqrt{\mathfrak{I} + (f)}$. If u, λ and p_r are lucky for Algorithm II.11, then it returns a geometric resolution of $\sqrt{\mathfrak{I} + (f)}$. Its complexity is in*

$$\mathcal{O}(n(L + n^2)\mathcal{U}(\delta)\mathcal{U}(d\delta)),$$

in terms of arithmetic operations in k.

II.7 The Resolution Algorithm

In this section we present the whole resolution algorithm. Let $f_1, \ldots, f_n \in k[x_1, \ldots, x_n]$ be a reduced regular sequence of polynomials outside the hypersurface defined by the polynomial g. That is, if we write $\mathcal{V}_i = \overline{\mathcal{V}(f_1, \ldots, f_i) \backslash \mathcal{V}(g)}$ we have the following situation: for $1 \leq i \leq n$, $\mathcal{V}_i$ is $(n-i)$-equidimensional and for $1 \leq i \leq n-1$, the quotient $(k[x_1, \ldots, x_n]/(f_1, \ldots, f_i))_g$ localized at g is reduced, by the Jacobian criterion this means that the Jacobian matrix of $f_1, \ldots, f_i$ has full rank at each generic point of $\mathcal{V}_i$.

The algorithm is incremental in the number of equations: we solve $\mathcal{V}_1, \ldots, \mathcal{V}_n$ in sequence. We encode each resolution by an isolated lifting fiber. So we need to choose at step i a Noether position for $\mathcal{V}_i$, a lifting point and a primitive element. These choices can be done at random with a low probability of failure, since bad choices are enclosed in strict algebraic subsets.

First we explain the incremental step of the algorithm, then we summarize all the conditions of genericity required by the geometry and the luckiness needed when using an algorithm designed for an integral ring in a non integral one. In §II.7.3 we discuss the special case when k is $\mathbb{Q}$.

II.7.1 Incremental Step

Let F_i be an isolated lifting fiber of $\mathcal{V}_i$, in this section we present our method to compute F_{i+1} from F_i, if F_i is generic enough. If this is not the case, we use the techniques of §II.5 to change the fiber.

We assume that we are given an isolated lifting fiber F for an r-equidimensional variety $\mathcal{V}$, a polynomial f intersecting $\mathcal{V}$ regularly and a polynomial g. Let $\mathfrak{I} = \mathfrak{I}(\mathcal{V})$. We want to compute a lifting fiber for the $(r-1)$-equidimensional variety $\overline{\mathcal{V} \cap \mathcal{V}(f) \backslash \mathcal{V}(g)}$. For the sake of simplicity we assume that the variables $x_1, \ldots, x_n$ are in Noether position for $\mathcal{V}$, let $p = (p_1, \ldots, p_r)$ be a lifting point of $\mathcal{V}$, u a primitive element, q its minimal polynomial on the p-fiber, $x_{r+1} = v_{r+1}(T), \ldots, x_n = v_n(T)$ the parametrization of the dependent variables and $f_1, \ldots, f_{n-r}$ the lifting equations.

In order to apply Algorithm II.11 we need to show that the lifted curve intersects regularly the hypersurface $\mathcal{V}(f)$. Let C be the lifted curve of F in the direction of x_r. Namely, let D be the line containing p with direction x_r, $\mathfrak{I}_D = \mathfrak{I} + (x_1 - p_1, \ldots, x_{r-1} - p_{r-1})$ can be seen as a 1-equidimensional ideal of $k[x_r, \ldots, x_n]$. Thanks to the techniques of §II.4.5 we can compute a geometric resolution C of $\mathfrak{I}_D$ from F.

If the variables $x_1, \ldots, x_n$ are in Noether position for $\mathcal{V} \cap \mathcal{V}(f)$ then there exists a polynomial $A \in k[x_1, \ldots, x_r]$ monic in x_r such that $A \in \mathfrak{I} + (f)$. This implies that $A(p_1, \ldots, p_{r-1}, x_r) \in \mathfrak{I}_D + (f(p_1, \ldots, p_{r-1}, x_r, \ldots, x_n))$. Hence $f(p_1, \ldots, p_{r-1}, x_r, \ldots, x_n)$ intersects regularly the lifted curve, Algorithm II.11 applies.

Algorithm II.11 applied on C, $f(p_1, \ldots, p_{r-1}, x_r, \ldots, x_n)$, $\lambda \in k$ and $g(p_1, \ldots, p_{r-1}, x_r, \ldots, x_n)$ returns an isolated lifting fiber of $\overline{(\mathcal{V} \cap \mathcal{V}(f)) \backslash \mathcal{V}(g)}$ for the lifting point $(p_1, \ldots, p_{r-1})$ and primitive element $\lambda x_r + u$, if the following conditions hold:

- the Noether position of F is also a Noether position of $\mathcal{V} \cap \mathcal{V}(f)$;
- $(p_1, \ldots, p_{r-1})$ is a lifting point for $\mathcal{V} \cap \mathcal{V}(f)$;
- λ is a Liouville point for C;
- $\lambda y_r + u$ is a primitive element of $\mathcal{V} \cap \mathcal{V}(f)$;
- $(p_1, \ldots, p_{r-1})$ is a cleaning point for $\overline{\mathcal{V} \cap \mathcal{V}(f) \backslash \mathcal{V}(g)}$;
- p_r is lucky for Algorithms II.8 and II.7;
- u and λ_r are lucky for Algorithm II.7.

We have seen that each of the above conditions is generic. If one of them were failing the techniques of §II.5 would recover a good situation.

Example 8 Here is an example where we need to change the primitive element: in $k[t, x_1, x_2]$, let $\mathcal{V}$ be given by the union of two lines D_1 and D_2 parametrized as follows: $(x_1 = 1, x_2 = t)$ and $(x_1 = -1, x_2 = -t)$. The variables t, x_1, x_2 are in Noether position, $t = 0$ is a lifting point and x_2 a primitive element for $t = 0$. Intersecting $\mathcal{V}$ by the equation $x_2 = 0$ the two points solution are $(t = 0, x_1 = 1, x_2 = 0)$ and $(t = 0, x_1 = -1, x_2 = 0)$. For any value of $\lambda \in k$ the linear form $\lambda t + x_2$ does not separate these two points.

II.7.2 Parameters of the Algorithm

We call the choices on which the algorithm depends its **parameters**. These are functions determining the choices of the Noether positions, lifting points and primitive elements of the fibers $F_1, \ldots, F_n$. In order to make the algorithm compute a correct result, they have to satisfy a few requirements. We have discussed them part by part, we now summarize them.

At step i of the algorithm we have a lifting fiber F_i of $\mathcal{V}_i$, we want to compute a lifting fiber for $\mathcal{V}_{i+1}$. For this we need to choose:

- a Noether position of $\mathcal{V}_{i+1}$, it is determined by a point N^{i+1} in k^{n-i-1} called the $(i+1)$th **Noether point**;
- a **lifting point** L^{i+1} for $\mathcal{V}_{i+1}$;
- a primitive element $u = \lambda_{n-i} y_{n-i} + \cdots + \lambda_n y_n$ for the corresponding fiber, the point $\mathsf{C}^{i+1} = (\lambda_{n-i}, \ldots, \lambda_n)$ is called the $(i+1)$th **Cayley point**.

These three functions N, L, C constitute the parameters of the algorithm. As seen in the previous subsection, the computations require some more restricting conditions. We distinguish three kinds of restrictions: the first ones are concerned with the geometry of the system, the second ones are also related to the geometry but are specific to the algorithm and the third ones are related with the luckiness of some specializations using algorithms designed for integral rings in case of non integral ones. Namely, let $r = n - i$, we gather all the conditions necessary for the execution and correctness of the whole algorithm:

- The pure geometric restrictions of the algorithm are:
  - Assume that $x_1, \ldots, x_n$ are in Noether position for $\mathcal{V}_i$, the change of variables $x_1 = y_1 + \mathsf{N}_1^{i+1} y_r, \ldots, x_{r-1} = y_{r-1} + \mathsf{N}_{r-1}^{i+1} y_r, x_r = y_r, \ldots, x_n = y_n$ brings the new coordinates y_i into Noether position for $\mathcal{V}_i \cap \mathcal{V}(f_{i+1})$;
  - The lifting point $\mathsf{L}^{i+1} = (p_1, \ldots, p_r)$ is chosen in k^r instead of k^{r-1}, the $r-1$ first coordinates are a lifting point of $\mathcal{V}_i \cap \mathcal{V}(f_{i+1})$ and a cleaning point with respect to g;

- The Cayley point $\mathsf{C}^{i+1} = (\lambda_r, \ldots, \lambda_n)$ is such that the linear form $\lambda_r y_r + \cdots + \lambda_n y_n$ is primitive for $\mathcal{V}_i \cap \mathcal{V}(f_{i+1})$ for the lifting point L^{i+1}.

- The geometric restrictions specific to the algorithm are:
  - L^{i+1} is a lifting point of $\mathcal{V}_i$ for the new coordinates y;
  - $u = \lambda_{r+1} y_{r+1} + \cdots + \lambda_n y_n$ is a primitive element of $\mathcal{V}_i$ for the lifting point L^{i+1}, and λ_r is a Liouville point for the lifted curve $\mathcal{V}_i \cap (y_1 - p_1, \ldots, y_{r-1} - p_{r-1})$.

- The luckiness restrictions are:
  - u is lucky for Algorithms II.6 and II.7;
  - p_r is lucky for Algorithm II.7 and II.8;
  - λ_r is lucky for Algorithm II.7.

We have seen along the previous sections that all these restrictions are contained in a Zariski open subset of the space they are lying in. This means that any random choice of these parameters leads to a correct computation with a high probability of success.

The complete algorithm is summarized in Algorithm II.12. Let

$$\delta = \max(\deg(\mathcal{V}_1), \ldots, \deg(\mathcal{V}_{n-1}))$$

, d be the maximum of the degrees of the f_i, L the complexity of evaluating $f_1, \ldots, f_{n-r+1}$ and g; $\mathcal{U}$ as before. Combining Propositions 9, 13 and 23 we get Theorem 1.

The *Initialization* step of Algorithm II.12 consists in initializing F as an isolated lifting fiber of the whole space. This particular case must be handled by each sub-functions of the algorithm, for the sake of clarity we do not give more details about this.

II.7.3 Special Case of the Integers

The complexity of our algorithm is measured in terms of number of arithmetic operations in k. When $k = \mathbb{Q}$ this model does not reflect the real behavior of the method. We now give a method which is efficient in practice, leading to a good running time complexity.

Assume that the input polynomial system $f_1, \ldots, f_n$ is reduced over each point of $\mathcal{V}_n$. Choose now at random a prime number p large enough so that the geometric resolution computed in $\mathbb{Z}/p\mathbb{Z}$ by algorithm II.12 is the modular trace of the one computed over $\mathbb{Q}$. It is clear that such prime numbers exist. Now we can apply Algorithm II.3 to recover the geometric resolution over $\mathbb{Q}$.

In a future work, one could prove that p can be chosen small enough. Steps have already be done into this direction by Hägele, Krick, Morais, Pardo and Sombra [Häg98, HMPS00, KPS01]. In [Sch00b, Sch00a] Schost proposes a definite answer for the lifting problem, but the method he uses could be extended to the whole algorithm.

ALGORITHM II.12: **Geometric Solve**

`GeometricSolve`$(\mathbf{f}, g)$

- $\mathbf{f}$ is a reduced regular system of n equations in n variables.
- g is a polynomial.

The procedure returns a geometric resolution of the roots of $\mathbf{f} = 0, g \neq 0$.

> $F \leftarrow$ Initialization;
> **for** i **from** 1 **to** n **do**
>   `ChangeFreeVariables`(F, N^i);
>   `ChangeLiftingPoint`(F, L^i);
>   `ChangePrimitiveElement`(F, C^i);
> $C \leftarrow$ subs($t = y_r$,`LiftCurve`$(F, \mathsf{L}^i + (0, \ldots, 0, 1))$);
>   # Consider C as the geometric resolution of the
>   # corresponding lifted curve to perform:
> $F \leftarrow$ `OneDimensionalIntersect`$(C, f_i, \mathsf{C}^i, g)$;
> **od**;
> **return**(F);

II.8 Practical Results

We have implemented our algorithm within the Magma computer algebra system. The package has been called Kronecker version 0.1 [Lec99c] and is available from July 1999, with its documentation [Lec99a, Lec99b] at:

http://kronecker.medicis.polytechnique.fr.

Before presenting some data reporting performances of our method compared to some other ones, we discuss the relevance of such comparisons.

II.8.1 Relevance of the Comparisons

In computer algebra the best softwares for polynomial solving are based on rewriting techniques. These methods are all deterministic algorithms, so we have to keep in mind that we compare these deterministic algorithms to our probabilistic one. There is a special case when the final number of solutions of the system is equal to the Bézout number of the system, namely $\deg(f_1) \cdots \deg(f_n)$, then we get a deterministic result and the comparison is fair.

We can compare our implementation to Gröbner bases computations and algorithms of change of bases. To compute a Gröbner basis we have several possible choices concerning the elimination order and the algorithm of change of bases. We focus our attention to *grevlex* orders (graded reverse lexicographical order) and *plex* (pure lexicographical order). It is important to notice that our result is stronger than a grevlex basis but weaker than a plex one. One interesting comparison is with a RUR (Rational Univariate Representation) computation [Rou96]: the RUR given in output corresponds exactly to a Kronecker parametrization of the solutions. The software we have retained for these comparisons is: Magma, Gb [Fau95, Fau97] and RealSolving [Rou, Rou96]. To the best of our knowledge they are the best among the most commonly available software for polynomial system solving.

II.8.2 Systems of Polynomials of Degree 2

We begin with systems composed of n equations in n variables of degree $d = 2$ for different heights h, representing the maximum number of decimal digits of the coefficients of the equations. The number of solutions of the systems is the Bézout number $D = 2^n$.

The following table has be realized with a Compaq Alpha EV6, 500 MHz, 128 Mb of MEDICIS [Med]. The column Gb + Realsolving means that the computations have been done using successively Gb for computing a Gröbner basis for grevlex ordering and Real Solving for computing the RUR from the basis. We have used the interface available within the Mupad computer algebra system [MuP96, FR98]. Each entry of the column contains the respective times for each part of the computation. The columns Magma grevlex and lex correspond respectively to Gröbner bases computations for grevlex and lex ordering. Note that Magma uses the Gröbner Walk algorithm in the lex case.

The notation $> 128Mb$ means that the computation can not be performed within $128Mb$.

n	h	Kronecker 0.1	Gb grevlex	+	Real Solving	Magma grevlex	Magma lex
4	4	5.4 s	0.5s	+	0.5s	0.3s	1.1s
4	8	6 s	1s	+	1.3s	0.4s	2.2s
4	16	7.5s	2.5s	+	3.7s	0.8s	6s
4	32	11.7s	7s	+	9.3s	1.8s	20s
5	4	29.5s	5s	+	18s	2s	44s
5	8	42.2s	17s	+	57s	5s	155s
5	16	78s	65s	+	180s	15s	563s
5	32	196s	244s	+	592s	46s	2064s
6	4	186s	209s	+	$> 128Mb$	58s	3855s
6	8	335s	773s	+	$> 128Mb$	175s	14112s
6	16	875s	2999s	+	$> 128Mb$	552s	54703s
6	32	2312s	5652s	+	$> 128Mb$	1750s	

This first comparison reveals that our method is faster that Gb + Realsolving, but the more striking is that we are able to compute the same output as Gb + Realsolving even faster than the computation of the grevlex Gröbner basis. Moreover, in this case our result is deterministic since the number of solutions found is equal to the Bézout number of the system.

II.8.3 Camera Calibration (Kruppa)

The original problem comes from [Kru13] and has been introduced in computer vision in [MF92]. It is composed of 5 equations in 5 variables. Each equation is a difference of two products of two linear forms. The parameter h is the size of the integers of the input system. The systems have 32 solutions. The comparisons are as above, on the same machine.

h	Kronecker 0.1	Gb grevlex + Realsolving	Magma grevlex	Magma lex
25	43s	18s + 36s	5s	118s
60	228s	195s + 716s	56s	2482s

II.8.4 Products of Linear Forms

The last example we give is not completely generic. We take 7 equations in 7 variables with integers coefficients of size 18, each equation is a product of two linear forms minus a constant coefficient. The system has 128 solutions, the integers of the output have approximately 8064 decimal digits. The computations have been done using a DEC Alpha EV56, 400 MHz, 1024 Mb of MEDICIS.

Kronecker 0.1	Gb grevlex	Magma grevlex
5h	∞	13.6h

It illustrates the good properties of the practical complexity of our approach.

Chapitre III

Quadratic Newton Iteration for Systems with Multiplicity

Newton's iterator is one of the most popular components of polynomial equation system solvers, either from the numeric or symbolic point of view. This iterator usually handles smooth situations only (when the Jacobian matrix associated to the system is invertible). This is often a restrictive factor. Generalizing Newton's iterator is still an open problem: how to design an efficient iterator with a quadratic convergence even in degenerate cases? We propose an answer for a $\mathfrak{m}$-adic topology when the ideal $\mathfrak{m}$ can be chosen generic enough: compared to a smooth case we prove quadratic convergence with a small overhead that grows with the square of the multiplicity of the root.

III.1 Introduction

Let us consider the polynomial equation system $f_1 = f_2 = f_3 = 0$ in the unknowns x_1, x_2, x_3, where:

$$\begin{aligned} f_1 &:= 2\,x_1 + 2\,x_1^2 + 2\,x_2 + 2\,x_2^2 + x_3^2 - 1, \\ f_2 &:= (x_1 + x_2 - x_3 - 1)^3 - x_1^3, \\ f_3 &:= (2\,x_1^3 + 5\,x_2^2 + 10\,x_3 + 5\,x_3^2 + 5)^3 - 1000\,x_1^5. \end{aligned}$$

It is easy to check that $x^* := (0, 0, -1)$ is an isolated multiple root of this system. Assume that, for a given prime number p, we are given an approximate root $z := (z_1, z_2, z_3)$ in $\mathbb{Z}^3$, where $z_1 = 0 \bmod p$, $z_2 = 0 \bmod p$ and $z_3 = -1 \bmod p$. Our problem is to recover x^* from z. Let F denote the sequence of the above polynomials f_1, f_2, f_3 in $\mathbb{Z}[x_1, x_2, x_3]$.

If x^* were a simple root then the classical Newton iterator N would solve our problem:

$$N(z) := z - \left(\frac{dF}{dx}(z)\right)^{-1} F(z).$$

The iterator N is well-defined over the ring of the p-adic integers $\mathbb{Z}_p$ if the Jacobian matrix $\frac{dF}{dx}$ is invertible at z. Except for a finite number of primes p this condition is satisfied and the sequence $(N^\kappa(z))_{\kappa\geq 0}$ of the iterated of z converges to x^* quadratically in $\mathbb{Z}_p^3$, that is:

$$N^\kappa(z) - x^* \in p^{2^\kappa}\mathbb{Z}_p^3.$$

Then, using a rational reconstruction algorithm [Dix82] (Proposition 10), one can recover x^* from a p-adic precise enough approximation.

One can check that the multiplicity M of x^* is 18. This can be computed as the dimension of the $\mathbb{Q}$-algebra $\mathbb{Q}[[x_1, x_2, x_3{+}1]]/(F)$ thanks to the software Singular [GPS01]. Moreover the above system has 54 isolated solutions counted with multiplicity; x^* is the only multiple root.

The purpose of this article is the construction of an iterator $\widetilde{N}$ that generalizes N for multiple roots. The validity of $\widetilde{N}$ still depends on the choice of a lucky prime p. We show that there exists only a finite number of unlucky p. The number of operations in $\mathbb{Z}_p$ executed by $\widetilde{N}$ is linear in the evaluation complexity of the input system F and in the square of the multiplicity (up to logarithmic factors), but it is important to underline that the algorithm does not compute the multiplicity and does not need to know it.

III.1.1 Main Result

Let $\mathfrak{o}$ be a Noetherian domain and k its field of fractions. The reader may keep in mind that our cases of interest are $\mathfrak{o} = K[t]$, $\mathfrak{o} = K[t_1, \ldots, t_m]$ (where K is a field) and $\mathfrak{o} = \mathbb{Z}$ (as in the above situation). We denote by $\overline{k}$ the algebraic closure of k. We are given $f_1, \ldots, f_s$ polynomials in $\mathfrak{o}[x_1, \ldots, x_n]$. Let x^* be an isolated point (for the Zariski topology, see for instance [Mat86, §4]) of multiplicity M of the algebraic variety $\{x \in \overline{k}^n,\ f_1(x) = \cdots = f_s(x) = 0\}$. The multiplicity M is the dimension of the $\overline{k}$-algebra $\overline{k}[[x_1 - x_1^*, \ldots, x_n - x_n^*]]/(f_1, \ldots, f_s)$. Roughly speaking, we are concerned with the following lifting problem: can we recover x^* from one of its approximations modulo a maximal ideal $\mathfrak{m}$ of $\mathfrak{o}$?

We propose a partial answer to this question: our method works for some lucky ideals $\mathfrak{m}$ only. The first restrictions on $\mathfrak{m}$ are natural and concern the specialization of the root x^* modulo $\mathfrak{m}$.

Let Q be a *monic irreducible* polynomial of $k[T]$. We denote by u the image of T in the algebraic extension $k(u) := k[T]/(Q(T))$. We assume that $x^* \in k(u)^n$ and:

> (H_Q) There exists $\rho_Q \in \mathfrak{o}$ such that ρ_Q is a unit in $\mathfrak{o}/\mathfrak{m}$, $\rho_Q Q \in \mathfrak{o}[T]$ and the discriminant of $\rho_Q Q$ is a unit in $\mathfrak{o}/\mathfrak{m}$.

Concerning the classical mathematical background we refer to [Mat86, §8]. We denote by $\hat{\mathfrak{o}}$ the completion of $\mathfrak{o}$ with respect to the $\mathfrak{m}$-adic topology. Any a in $\mathfrak{o}$ that is a unit in $\mathfrak{o}/\mathfrak{m}$ is a unit in $\mathfrak{o}/\mathfrak{m}^\kappa$ for any integer $\kappa \geq 1$ and is also a unit in $\hat{\mathfrak{o}}$. Under hypothesis (H_Q), the quotient ring $A := \hat{\mathfrak{o}}[T]/(Q(T))$ is well-defined as an $\hat{\mathfrak{o}}$-algebra of dimension the degree of Q and inherits the complete and separated $(\mathfrak{m}A)$-adic topology. Multiplication in A is continuous: if $z_i \in \mathfrak{m}^{\kappa_i}A$, for $i = 1, 2$ and integers $\kappa_i \geq 0$, then $z_1 z_2 \in \mathfrak{m}^{\kappa_1+\kappa_2}A$.

From a practical point of view the situation is the following. We want to compute in A but knowing only an approximation $q \in \mathfrak{o}[T]$ of Q, that is $\rho_Q(Q - q)$ has all its

coefficients in $\mathfrak{m}$. But it suffices to observe that A is isomorphic to $\hat{\mathfrak{o}}[T]/(q(T))$ (see Proposition 24 below) to make the computations in A effective.
In order to embed x^* in A we need the following hypothesis (H_{x^*}). We denote by p the canonical projection from $k[T]$ onto $k[u]$. Let $p^{-1} : k[u] \to k[T]$ be the linear map such that $p^{-1}(e)$ is the unique polynomial of degree less than the degree of Q and such that its projection in $k[u]$ is e.

> (H_{x^*}) There exists ρ_{x^*} in $\mathfrak{o}$ such that ρ_{x^*} is a unit in $\mathfrak{o}/\mathfrak{m}$ and the polynomial $\rho_{x^*} p^{-1}(x_i^*)$ is in $\mathfrak{o}[T]$, for $i = 1, \ldots, n$.

We still write x^* for the image of x^* in A when there is no danger of confusion. Let $z \in A^n$ be an approximation of x^* modulo $\mathfrak{m}$. In the case $M = 1$ (we say that x^* is a simple root) it is well-known that Newton's iterator answers our lifting problem ([Lan93, XII, §7] for instance). Let F denote the vector of polynomials $(f_1, \ldots, f_s)$ and $\frac{dF}{dx}$ the Jacobian matrix of F. If

> (H_J) The determinant of $\frac{dF}{dx}(x^*)$ in a unit in $A/\mathfrak{m}$,

then Newton's iterator N is well-defined in any neighborhood of x^*:

$$N(z) = z - \left(\frac{dF}{dx}(z)\right)^{-1} F(z).$$

Its convergence is **quadratic**, that is for all $\kappa \geq 1$:

$$z - x^* \in (\mathfrak{m}^\kappa A)^n \implies N(z) - x^* \in (\mathfrak{m}^{2\kappa} A)^n.$$

If the multiplicity M is greater than 1 then Newton's iterator is not defined anymore at x^*. Our aim is the generalization of this iterator N in order to handle multiple roots. We introduce an iterator $\widetilde{N}$ answering this problem and prove that the overhead is mainly M^2. Our algorithm works under some genericity conditions: the characteristic of k must be big enough, the coordinates must be generic enough and the maximal ideal $\mathfrak{m}$ must satisfy (H_Q), (H_{x^*}) and other conditions generalizing (H_J). The change of coordinates is represented by a matrix $\mathcal{M}$ of $GL_n(k)$ (the group of invertible $n \times n$ matrices over k). The complexity model we use is stated in §III.4.1. The constant Ω comes mainly from linear algebra complexity: $3 \leq \Omega < 4$. We refer to §III.4.1 for further details.

Theorem 3 *Let $\mathfrak{o}$ be a Noetherian domain and k its field of fractions. There exists a deterministic algorithm performing the following task. The inputs of the algorithm are:*

- *A sequence $f_1, \ldots, f_s$ of polynomials in $\mathfrak{o}[x_1, \ldots, x_n]$ given by a straight-line program of length L.*
- *$q, v_1, \ldots, v_n$, a sequence of polynomials in $\mathfrak{o}[T]$;*
- *A matrix $\mathcal{M}$ of $GL_n(k)$.*
- *A maximal ideal $\mathfrak{m}$ of $\mathfrak{o}$.*

We assume that the input satisfies the following hypotheses:

- *q is monic.*
- *There exists polynomials $Q, V_1, \ldots, V_n$ in $k[T]$ such that:*
  - *Q is monic and irreducible.*
  - *The point $x^* = (V_1(u), \ldots, V_n(u))$ in $k(u) := k[T]/(Q(T))$ is an isolated root of the system $f_1 = \cdots = f_s = 0$ with multiplicity M.*
  - *There exist ρ_Q and ρ_{x^*} in $\mathfrak{o}$ and being units in $\mathfrak{o}/\mathfrak{m}$ such that the polynomials $\rho_Q Q$ and $\rho_{x^*} V_i$ have all their coefficients in $\mathfrak{o}$ and $\rho_Q(Q-q)$ and $\rho_{x^*}(V_i - v_i)$, for $i = 1, \ldots, n$, have all their coefficients in $\mathfrak{m}$.*
- *The characteristic $\mathrm{char}(k)$ is either 0 or at least $M+1$.*
- *The matrix $\mathcal{M}$ is outside of an algebraic hypersurface of $GL_n(k)$ depending on the input polynomials and the root x^*.*
- *The ideal $\mathfrak{m}$ does not contain an element $a \neq 0$ of $\mathfrak{o}$ that depends on the input polynomials, the root x^* and $\mathcal{M}$.*

Let z be the image of $(v_1, \ldots, v_n)$ in $A := \hat{\mathfrak{o}}[T]/(q(T))$, we still write x^* for its image in A. Let κ be a lower bound on the precision of z as an approximation of x^* in A: $z - x^* \in (\mathfrak{m}^\kappa A)^n$. Under the above conditions the algorithm computes polynomials $\widetilde{v}_1, \ldots, \widetilde{v}_n$ in $\mathfrak{o}/(\mathfrak{m}^{2\kappa})[T]$ such that the image $\widetilde{N}(z)$ of $(\widetilde{v}_1, \ldots, \widetilde{v}_n)$ in A approximates x^* at precision at least 2κ, that is: $\widetilde{N}(z) - x^* \in (\mathfrak{m}^{2\kappa} A)^n$. The algorithm performs

$$\mathcal{O}\Big(n^3(nL + n^\Omega)M^2 \log(nM)\Big)$$

arithmetic operations in $A/(\mathfrak{m}^{2\kappa}A)$.

As an immediate consequence of the above theorem, the sequence $(\widetilde{N}^l(z))_{l \geq 0}$ converges quadratically to x^*: $\widetilde{N}^l(z) - x^* \in (\mathfrak{m}^{2^l\kappa} A)^n$.

It is important to notice that the theorem does not state neither the existence of a lucky $\mathfrak{m}$ nor generic enough changes of coordinates. If k has characteristic zero then almost all random choices of coordinates are generic enough. In the case when $\mathfrak{o} = K[t_1, \ldots, t_n]$ (where K is a field) the maximal ideals $\mathfrak{m} = (t_1 - p_1, \ldots, t_n - p_n)$ that are not lucky come from points $(p_1, \ldots, p_n)$ included in an algebraic hypersurface in K^n. Hence, if K has characteristic zero then almost all maximal ideals are lucky. In the case when $\mathfrak{o} = \mathbb{Z}$ only a finite number of maximal ideals are not lucky. We leave probability estimates for further work and focus only on the algorithm.

From Chapter II we recall that for the special case $M = 1$ the complexity of Newton's iterator is in $\mathcal{O}(nL + n^\Omega)$. Focusing on the dependency on M, the overhead of $\widetilde{N}$ grows with M^2 up to logarithmic factors.

The complexity model and the algorithm are presented in §III.4. The basic tools necessary for the algorithm are given in §III.2. The mathematical idea is developed in §III.3.

In §III.5 we use the classical dynamic evaluation framework to handle reducible sets of roots. In the next paragraphs we give some details about the important consequences in the field of polynomial systems solving. We conclude this introduction with the presentation of the new underlying ideas along with examples.

III.1.2 Motivation for Polynomial System Solving

In Chapter II we present an algorithm to solve systems of polynomial equations and inequations: a practical variant of the *geometric resolution algorithm*. We recall the main framework of this algorithm. Let k be a field of characteristic zero. We are given polynomials $f_1, \ldots, f_n, g$ in $k[x_1, \ldots, x_n]$ and are interested in computing a description of the set of roots of the system $f_1 = \cdots = f_n = 0$, $g \neq 0$. Let $\mathcal{V}(f_1, \ldots, f_i)$ (resp. $\mathcal{V}(g)$) denote the algebraic variety solution of $f_1 = \cdots = f_i = 0$ (resp. $g = 0$). Let $\mathcal{V}_i$ be the Zariski closure of $\mathcal{V}(f_1, \ldots, f_i) \backslash \mathcal{V}(g)$, for $i = 1, \ldots, n$. Our algorithm is incremental in the number of equations to be solved, we compute a description of $\mathcal{V}_{i+1}$ from one of $\mathcal{V}_i$. Our method works under the following restrictive hypotheses:

(R_1) *Regularity hypothesis:* each $\mathcal{V}_i$ is equidimensional of dimension $n - i$;

(R_2) *Reduction hypothesis:* the Jacobian matrix of $f_1, \ldots, f_i$ has full rank when evaluated at $\mathcal{V}_i$.

We assume that the affine coordinates are generic enough (if this is not the case then replace them by a random affine transformation). Roughly speaking here is the incremental step of our solver: at step i, the variety $\mathcal{V}_i$ is represented by the finite set of solutions of the system $f_1 = \cdots = f_i = x_1 = \cdots = x_{n-i} = 0, g \neq 0$, called a *lifting fiber*. From this set of points we compute the curves solutions of the system $f_1 = \cdots = f_i = x_1 = \cdots = x_{n-i-1} = 0$, $g \neq 0$. This curve is parametrized by the variable x_{n-i}, it is called a *lifting curve*. This computation is called the *lifting step*. Then we compute the intersection of this curve with the next equation $f_{i+1} = 0$, this is the *intersection step*: according to hypothesis (R_1) this yields a finite set of points from which we remove the solutions of $g = 0$, this is the *cleaning step*. This way, we obtain a lifting fiber for $\mathcal{V}_{i+1}$, which represents the solutions of the system $f_1 = \cdots = f_{i+1} = x_1 = \cdots = x_{n-i-1} = 0$, $g \neq 0$.

We focus on the lifting step: the one of Chapter II relies on Newton's iterator, the invertibility of the Jacobian matrix occurring in this iterator is equivalent to hypothesis (R_2). This is a restrictive factor of the method. Theorem 3 remedies this problem. Full details about the complete resulting solver are given in the next chapter: we show how to get rid of hypothesis (R_1) as well and how to compute the equidimensional algebraic decompositions of the $\mathcal{V}_i$ in sequence.

III.1.3 Brief History

For centuries Newton's method has certainly been the most famous approach for solving equations and systems of equations numerically, but the idea of using it in a symbolic solver is more recent. We refer to Schost's thesis [Sch00b, Chapitre 6] for a detailed historical presentation.

In the non-archimedian case of $\mathfrak{I}$-adic topologies, where $\mathfrak{I}$ is an ideal of a ring R, we attribute the introduction of Newton's method in computer algebra to Zassenhaus [Zas69] for greatest common divisor computations (known as Hensel's lemma). In the field of polynomial equation system solving, the earliest occurrence of Newton's iterator seems to be due to Trinks [Tri85] in 1985: he proposed to lift to the rational numbers of a shape-lemma [GM89] Gröbner basis which is only known modulo a lucky prime number.

In 1988, Winkler [Win88] generalized Trinks' approach for the computations of p-adic approximations of Gröbner bases. The choice of a lucky p is then discussed in several papers: Gianni [Gia87], Kalkbrener [Kal87, Kal97], Pauer [Pau92], Gräbe [Grä93], Assi [Ass94], and more recently Gianni, Fortuna and Trager [GFT00]. In [Nau98] Nauheim proposes a method to handle lifting with an unlucky p.

The *geometric resolution algorithm* we are concerned with has been introduced by Giusti, Heintz, Morais, Morgenstern and Pardo [GHM$^+$98] in the early 1990s: Newton's iterator is used to compress the straight-line programs encoding the resolutions of the intermediate varieties $\mathcal{V}_i$ in the incremental solving process. The representation of each $\mathcal{V}_i$ has a size polynomial in its degree. The resulting solver has a complexity mainly polynomial in the maximum of the degrees of the intermediate varieties $\mathcal{V}_i$. This was a breakthrough in theoretical complexity. This idea has been developed in a series of papers [Mor97, GHH$^+$97, GHMP97, Häg98, HKP$^+$00]. From a practical point of view, the geometric resolution algorithm has been turned into an efficient software called Kronecker [Lec99c]. It has been designed by Giusti, Lecerf and Salvy [GLS01] and implemented in the Magma computer algebra system [BC89, BC90, BCM94, BC95, CP96, BCP97]: the theoretical algorithm has been completely redesigned and simplified. We improved its complexity dramatically. We introduced, independently of Heintz, Matera and Waissbein [HMW01], the notion of *lifting curves*.

I propose in Appendix B a theoretical generalization of the geometric resolution algorithm for computing an equidimensional decomposition of the solution set of any polynomial equation system (removing the above regular reduced restrictive hypotheses (R_1) and (R_2)). My approach is based on Bertini's first theorem as initiated in [KP96, Mor97]: the input system is replaced with generic linear combinations of the given equations. I prove that the only lifting to perform concerns the smooth components and can be done using the classical Newton iterator. But my algorithm presents two main drawbacks. On the one hand, the substitution of the original system by linear combinations of the original equations spoils the evaluation complexity of the intermediate systems. On the other hand the theoretical description does not lead to an easy implementation as in [GLS01] when applying the *deforestation* idea introduced in [GHL$^+$00] to eliminate straight-line programs in the intermediate computations. Jeronimo and Sabia also propose a generalization of the algorithm in [JS00]. Their approach and purposes are different: they provide an idealistic description of each equidimensional component instead of a geometric resolution. This is a less convenient output for numerical solving.

The purpose of this chapter is the generalization of the algorithm presented in Chapter II, but without mixing the equations of the input system and keeping the natural incremental resolution process. The main problem is to deal with situations featuring multiple components.

From a numerical point of view the one dimensional case is now well understood as demonstrated in Yakoubsohn's recent work [Yak00]. But in several variables there exists no satisfying generic Newton iterator handling multiplicities. In [MS95] Möller and Stetter postprocess Gröbner bases numerically in order to compute all the roots of a zero-dimensional polynomial equation system. In [Ste96] Stetter exploits some Gröbner bases in order to obtain local information about a cluster of roots. As pointed out to me by Schost, a generic numerical iterator has been proposed by Ojika, Watanabe and Mitsui in [Oji82, OWM83, Oji87]. Their idea consists in replacing the original system by another one for which the considered singular root has smaller multiplicity. This is done by differentiating well chosen equations. After a finite number of steps they obtain a system for which the considered root is simple. The computations are done mixing numerical and symbolical manipulations. It is a pity that their study lacks stability and complexity analyses. Since they call their algorithm the *Modified Deflation Algorithm*, we will refer to our method as a **deflation algorithm**.

III.1.4 Presentation of the Method

Before entering the mathematical framework of our algorithm we introduce the basic ideas along examples.

Example 1 We start with the easiest case, with one variable only: $n = 1$, $\mathfrak{o} = \mathbb{Q}[t]$, $k = \mathbb{Q}(t)$ and one polynomial $f(x)$ in $\mathfrak{o}[x]$. Let $p \in \mathbb{Q}$ and $\mathfrak{m} = (t - p)$, we are given an algebra $A = \hat{\mathfrak{o}}[T]/(q(T))$ as defined above in §III.1.1 and an approximation $z \in A$ of a root x^* of f of multiplicity M. If M is known and is greater than 1 we can replace f by its $(M-1)$st derivative $\widetilde{f} := \frac{\partial^{M-1} f}{\partial x^{M-1}}$ and then use the classical Newton iterator (if $\widetilde{f}'(x^*)$ is invertible in A). Practically we proceed this way: to evaluate $\widetilde{f}$ and its first order derivative at the point z, we take a new variable dx and evaluate f in the power series ring $A[[dx]]$ at $z + dx$ and precision $\mathcal{O}(dx^{M+1})$:

$$f(z + dx) = \sum_{i=0}^{M} \frac{1}{i!} \frac{d^i f}{dx^i}(z) dx^i + \mathcal{O}(dx^{M+1}),$$

so that introducing the function **coeff** to extract the coefficient of its first argument with respect to its second one, we deduce the iterator:

$$\widetilde{N}(z) := z - \widetilde{f}/\widetilde{f}'(z) = z - \frac{1}{M} \frac{\operatorname{coeff}(f(z + dx), dx^{M-1})}{\operatorname{coeff}(f(z + dx), dx^M)}. \tag{III.1}$$

Now let us assume that M is *a priori* unknown. Our strategy is to determine M and then to use the iterator $\widetilde{N}$. The success of the algorithm relies on the choice of $\mathfrak{m}$.

We compute the multiplicity M_p of z as a root of f in $A/\mathfrak{m}A$. For this purpose we evaluate f at $z + dx$ in the power series ring $A/\mathfrak{m}A[[dx]]$:

$$M_p := \mathrm{val}_{dx} f(z + dx),$$

where val denotes the valuation function. Assuming that $M_p = M$ and that $\frac{d^M f}{dx^M}(x^*)$ is a unit in $A/\mathfrak{m}A$ then the sequence $(\widetilde{N}^\kappa(z))_\kappa$ converges quadratically to x^*.

As for the justification of this algorithm we observe that $M_p = M$ if and only if $\frac{\partial^M f}{\partial x^M}(z)$ is a unit in $A/\mathfrak{m}A$. This condition generalizes (H_J). In particular this proves that, except for a finite set of choices, almost all p yield a correct multiplicity.

Example 2 We now take $n = 2$, $\mathfrak{o} = \mathbb{Q}[t]$, $f_1 = (x_1 - t)^3 - x_2^2$, $f_2 = x_2^4 + 6t^5 x_1 - t^6$. One can check that the set of roots of

$$x_1 = 0,\ x_2^2 + t^3 = 0 \tag{III.2}$$

is an isolated component of the variety defined by $f_1 = f_2 = 0$ of multiplicity $M = 2$. This is confirmed by the following computations.

Let p be a point in $\mathbb{Q}$ and $\mathfrak{m} = (t-p)$. We are given $q(T) = T^2 + p^3$, the corresponding algebra $A = \mathbb{Q}[[t-p]][T]/(q(T))$ and the approximate root $z = (0, u)$, where u denotes the image of T in A. In order to satisfy (H_Q) and (H_{x^*}) it suffices that $p \neq 0$.

Let dx_2 be a new variable. First we compute an approximation of $y_1 \in A[[dx_2]]$, the power series solution of $f_1(y_1, u + dx_2) = 0$ using an effective version of the implicit function theorem:

$$y_1 = (t-p) + \frac{2}{3p^2} u\, dx_2 - \frac{1}{9p^2} dx_2^2 + \mathcal{O}((t-p)^2, dx_2^3).$$

Substituting x_1 by the above approximation of y_1 in the other equation $f_2 = 0$ we get:

$$0 = 20p^2 u(t-p) dx_2 - \frac{10}{3} p^2 (2p + (t-p)) dx_2^2 + \mathcal{O}((t-p)^2, dx_2^3).$$

Differentiating the above equation with respect to dx_2 yields a linear equation in dx_2:

$$0 = 20p^2 u(t-p) - \frac{20}{3} p^2 (2p + (t-p)) dx_2 + \mathcal{O}((t-p)^2, dx_2^2).$$

This equation admits a unique solution of valuation 1 and precision $\mathcal{O}((t-p)^2)$:

$$dx_2 = \frac{3}{2p} u(t-p) + \mathcal{O}((t-p)^2).$$

We deduce that $(0, u(1 + \frac{3}{2p}(t-p)))$ is an approximate root at precision $\mathcal{O}((t-p)^2)$. From $x_2^* = u(1 + \frac{3}{2p}(t-p)) + \mathcal{O}((t-p)^2)$ we deduce that $u = x_2^*(1 - \frac{3}{2p}(t-p)) + \mathcal{O}((t-p)^2)$. Substituting u by this value in $u^2 + p^3 = 0$ we recover an approximation of (III.2):

$$x_1 = 0, \quad x_2^2 + p^3 + 3p^2(t-p) = 0 + \mathcal{O}((t-p)^2).$$

We could repeat this process once more and reach the precision $\mathcal{O}((t-p)^4)$. In this way we recover (III.2) completely.

Example 3 We are coming back to the example of the beginning: $n = 3$, $\mathfrak{o} = \mathbb{Z}$, $k = \mathbb{Q}$, and $F = f_1, f_2, f_3$, where

$$\begin{aligned} f_1 &:= 2\,x_1 + 2\,x_1^2 + 2\,x_2 + 2\,x_2^2 + x_3^2 - 1, \\ f_2 &:= (x_1 + x_2 - x_3 - 1)^3 - x_1^3, \\ f_3 &:= (2\,x_1^3 + 5\,x_2^2 + 10\,x_3 + 5\,x_3^2 + 5)^3 - 1000\,x_1^5. \end{aligned}$$

The root $x^* = (0, 0, -1)$ lies in k^3 and has multiplicity 18. For the sake of simplicity we chose an example without algebraic extension. The Jacobian matrix of F at x^* has rank 1. We can not treat this example now, it will serve to illustrate our algorithm later in §III.3.2 and §III.4.5.

III.2 Preliminaries

In this section we present the foundations of our generalized Newton iterator.

III.2.1 Local and Global Point of Views

We explain in §II.4 how to lift an algebraic eliminant polynomial in a global way. Informally speaking we recall that the local point of view corresponds to situations when the parameterization of the variables are known with greater precision than the minimal polynomial defining the algebra in which the computations are done. The global point of view is when the minimal polynomial is updated at each improvement of the precision. The following proposition enlightens a bit the results of §II.4. It says that both computations are equivalent.

In this chapter we do not assume having a parameterization of the roots in terms of an explicit primitive linear form so that we do not perform the globalization trick of §II.4. We prefer the local point of view for the presentation of the algorithm. But in the next chapter the global point of view comes back in §IV.2.

Proposition 24 *Let Q and q be monic polynomials in $\hat{\mathfrak{o}}[T]$ such that $Q - q$ has all its coefficients in $\mathfrak{m}$ and the discriminant of Q is a unit in $\hat{\mathfrak{o}}/\mathfrak{m}$. Then $\hat{\mathfrak{o}}[T]/(Q(T))$ is homeomorphic to $\hat{\mathfrak{o}}[T]/(q(T))$. For any integer $\kappa \geq 1$, $(\hat{\mathfrak{o}}/\mathfrak{m}^\kappa)[T]/(Q(T))$ is homeomorphic to $(\hat{\mathfrak{o}}/\mathfrak{m}^\kappa)[T]/(q(T))$.*

Proof. The idea is to construct a root of q in $\hat{\mathfrak{o}}[T]/(Q(T))$ and a root of Q in $\hat{\mathfrak{o}}[T]/(q(T))$ and use these roots to construct the homeomorphisms. We denote by U (resp. u) the image of T in $\hat{\mathfrak{o}}[T]/(Q(T))$ (resp. $\hat{\mathfrak{o}}[T]/(q(T))$). Since $Q - q$ has all its coefficients in $\mathfrak{m}$ the discriminant of q equals the discriminant of Q in $\hat{\mathfrak{o}}/\mathfrak{m}$. Hence the derivative $q'(U)$ is invertible in $\hat{\mathfrak{o}}[T]/(Q(T))$. In $\hat{\mathfrak{o}}[T]/(Q(T))$ we build the sequence $(a_\kappa)_{\kappa\geq 0}$:

$$a_0 = U, \quad a_{\kappa+1} = a_\kappa - \frac{q(a_\kappa)}{q'(a_\kappa)}, \ \kappa \geq 0.$$

Since $q(U) = 0$ modulo $\mathfrak{m}$ the sequence (a_κ) converges quadratically to a root a of q in $\hat{\mathfrak{o}}[T]/(Q(T))$:

$$a - a_\kappa \in \mathfrak{m}^{2^\kappa}\Big(\hat{\mathfrak{o}}[T]/(Q(T))\Big).$$

Since a is a root of q the following map g is well-defined as a continuous $\hat{\mathfrak{o}}$-algebra morphism:

$$\begin{array}{rccc} g: & \hat{\mathfrak{o}}[T]/(q(T)) & \to & \hat{\mathfrak{o}}[T]/(Q(T)) \\ & u & \mapsto & a \end{array}$$

Exchanging the roles of Q and q we construct b as a root of Q in $\hat{\mathfrak{o}}[T]/(q(T))$ and define h:

$$\begin{array}{rccc} h: & \hat{\mathfrak{o}}[T]/(Q(T)) & \to & \hat{\mathfrak{o}}[T]/(q(T)) \\ & U & \mapsto & b \end{array}$$

Since $q(u) = 0$ in $\hat{\mathfrak{o}}[T]/(q(T))$ then $h(g(q(u))) = q(h(g(u))) = 0$. Since $h(g(u)) = 0$ modulo $\mathfrak{m}$, it follows that $h(g(u)) = u$ in $\hat{\mathfrak{o}}[T]/(q(T))$ and in a similar way that $g(h(U)) = U$ in $\hat{\mathfrak{o}}[T]/(Q(T))$. This proves that g is an isomorphism and that $g^{-1} = h$. By construction g and h are both continuous and can be restricted modulo $\mathfrak{m}^\kappa$ for any $\kappa \geq 1$. $\square$

III.2.2 Basic Notations and Definitions

Let k be a field and S denote $k[[x_1, \ldots, x_n]]$, the power series ring in n variables over k. We denote by $\mathrm{val}(\phi)$ the **valuation** of ϕ in S. By convention the valuation of 0 is $+\infty$. For any subset Φ of S we define $\mathrm{val}(\Phi)$ as the minimum of the valuations of its elements. If Φ is empty, it has valuation $+\infty$. The **support** of a polynomial or a series is its set of monomials with a nonzero coefficient.

The first tool we need is a local counterpart of the classical Noether normalization lemma for algebraic varieties (see for instance [Mat86, §33]). Let Ψ be an ideal of S of valuation m, we say that a variable x_i is in **Weierstraß position** if there exists an element of Ψ of valuation m and having x_i^m in its support.

Like Noether positions, Weierstraß positions are easy to obtain: if m is the valuation of the ideal Ψ (assume that $\Psi \neq (0)$), then there exists an element ψ in Ψ of valuation m. Let ψ_m be the homogeneous component of valuation m of ψ and $a_1, \ldots, a_{n-1}$ in k such that $\psi_m(a_1, \ldots, a_{n-1}, 1)$ is not zero (assume that such a point exists), then the following change of variables puts x_n into Weierstraß position: replace x_i by $x_i + a_i x_n$ for $1 \leq i \leq n-1$. In particular, if k has characteristic zero it is always possible to find such a_i.

Lemma 5 *Let Ψ be an ideal of S, there exists an algebraic hypersurface of k^{n-1} such that for any element $(a_1, \ldots, a_{n-1})$ outside of it the following change of variables yields a Weierstraß position for x_n: replace x_i by $x_i + a_i x_n$ for $1 \leq i \leq n-1$.*

Let Φ be a subset of S, we define its **first partial derivative** $\frac{\partial \Phi}{\partial x_i}$ with respect to the variable x_i as the set $\Phi \cup \{\frac{\partial \phi}{\partial x_i},\ \phi \in \Phi\}$. If Ψ is an ideal of S then so is $\frac{\partial \Psi}{\partial x_i}$. Moreover the derivative of the ideal generated by Φ is generated by the derivative of Φ.

Lemma 6 *Let $\alpha_i \geq 0$, $i = 1, \ldots, n$. If $\alpha_l \geq 1$ then*

$$\frac{\partial}{\partial x_l}(x_1^{\alpha_1}, \ldots, x_n^{\alpha_n}) = (x_1^{\alpha_1}, \ldots, x_{l-1}^{\alpha_{l-1}}, x_l^{\alpha_l - 1}, x_{l+1}^{\alpha_{l+1}}, \ldots, x_n^{\alpha_n}).$$

Corollary 3 *Let $\alpha_i \geq 0$, $\beta_i \geq 0$, for $i = 1, \ldots, n$. Let $\pi_1 = (x_1^{\alpha_1}, \ldots, x_n^{\alpha_n})$ and $\pi_2 = (x_1^{\beta_1}, \ldots, x_n^{\beta_n})$, if $\alpha_l \geq 1$ and $\beta_l \geq 1$ then*

$$\frac{\partial}{\partial x_l}(\pi_1 \cap \pi_2) = \frac{\partial \pi_1}{\partial x_l} \cap \frac{\partial \pi_2}{\partial x_l}.$$

III.2.3 Gradient of an Ideal

In order to compute effectively in S we need to fix the precision of the series. The precisions used by our algorithm are built on what we call the *gradient* of an ideal. This construction is motivated by the following situation.

Let $\mathfrak{o}$ be a Noetherian domain, k be its field of fractions, f be a polynomial function in $\mathfrak{o}[x_1, \ldots, x_n]$ given by a *straight-line program* (see §III.4.1 for precise considerations) and π is a zero-dimensional monomial ideal of $S := k[[x_1, \ldots, x_n]]$ (an ideal generated by monomials). For short, we write monomials using multi-indices: if $\alpha = (\alpha_1, \ldots, \alpha_n)$, then x^α denotes $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ and $|\alpha|$ the sum of the α_i: $|\alpha| := \alpha_1 + \cdots + \alpha_n$. For each monomial x^α which does not belong to π we want to compute the corresponding partial derivative of f at a given point $a := (a_1, \ldots, a_n)$: $\frac{\partial^{|\alpha|} f}{\partial x^\alpha}(a) := \frac{\partial^n f}{\partial x_1^{\alpha_1} \cdots \partial x_n^{\alpha_n}}(a)$. It is classical to handle this situation by evaluating f at the point $(a_1 + x_1, \ldots, a_n + x_n)$ modulo the ideal π and picking up the right coefficients. Introducing the function $\mathbf{coeff}(f, x^\alpha)$ that returns the coefficient of x^α in f, we know that for all $x^\alpha \notin \pi$:

$$\frac{\partial^{|\alpha|} f}{\partial x^\alpha}(a) = (\alpha_1! \cdots \alpha_n!)\mathrm{coeff}\Big(f(a_1 + x_1, \ldots, a_n + x_n), x^\alpha\Big). \tag{III.3}$$

We are interested in extending this computation in order to compute the values of the gradients of the $\frac{\partial^{|\alpha|} f}{\partial x^\alpha}$. For this purpose we construct an ideal denoted by $\nabla_{\mathcal{L}}\pi$ (where $\mathcal{L}$ stands for the set $\{1, \ldots, n\}$, for the moment) such that the evaluation modulo $\nabla_{\mathcal{L}}\pi$ instead of π yields the gradients of $\frac{\partial^{|\alpha|} f}{\partial x^\alpha}$ for all x^α at point a via (III.3). It suffices to take $\nabla_{\mathcal{L}}\pi$ as the biggest monomial ideal not containing the $x_i x^\alpha$, for all $i \in \mathcal{L}$ and all x^α not in π.

We first prove that this construction is optimal and then give the properties used in the proof of the correctness of our algorithm presented in §III.4.

Let us now formalize the above construction. We consider the set $\mathcal{M}$ of monomials generated by n indeterminates $x_1, \ldots, x_n$ with nonnegative exponents:

$$\mathcal{M} = \{x_1^{\alpha_1} \cdots x_n^{\alpha_n},\ \alpha_i \geq 0,\ i = 1, \ldots, n\},$$

it is a semigroup (a group without an inverse operation) for the multiplication of monomials. A subset π of $\mathcal{M}$ stable under multiplication by any element of $\mathcal{M}$, that is $\mathcal{M}\pi = \pi$, is called a **stable subset**. We denote by $(x^{\beta_1}, \ldots, x^{\beta_s})$ the stable subset $\cup_{i=1}^s \mathcal{M} x^{\beta_i}$ generated by the monomials $x^{\beta_1}, \ldots, x^{\beta_s}$.

Let π be a stable subset of $\mathcal{M}$, $\overline{\pi}$ its complement (in $\mathcal{M}$) and $\mathcal{L}$ a subset of $\{1, \ldots, n\}$, we define $\nabla_{\mathcal{L}}\pi$, the **gradient** of π with respect to the variables in $\mathcal{L}$, as the complement set of $\overline{\pi} \cup \cup_{i\in\mathcal{L}} x_i\overline{\pi}$. For example, if $n = 2$, $\pi = (x_1^{\alpha_1}, x_2^{\alpha_2})$ and $\mathcal{L} = \{1, 2\}$ then $\nabla_{\mathcal{L}}\pi = (x_1^{\alpha_1+1}, x_2^{\alpha_2+1}, x_1^{\alpha_1}x_2^{\alpha_2})$.

Lemma 7 *With the above notations, $\nabla_{\mathcal{L}}\pi$ is a stable subset of $\mathcal{M}$.*

Proof. Let x^α be a monomial of $\nabla_{\mathcal{L}}\pi$, it suffices to prove that for each i the monomial x_ix^α is in $\nabla_{\mathcal{L}}\pi$. If it were not the case then x_ix^α would be either in $\overline{\pi}$ or one of the x_jx^β with $j \in \mathcal{L}$ and x^β in $\overline{\pi}$. The first situation immediately leads to a contradiction. As for the second situation, if i were equal to j we could deduce that $x^\alpha = x^\beta \in \overline{\pi}$, hence i would be different from j. From $x_ix^\alpha = x_jx^\beta$ we could construct γ such that $x^\beta = x_ix^\gamma$ and $x^\alpha = x_jx^\gamma$. Hence, x^γ would belong to $\overline{\pi}$ and x^α would be in $x_j\overline{\pi}$, which is a contradiction again. □

We extend this construction to monomial ideals. If π is a monomial ideal of S (an ideal generated by monomials) and $\mathcal{L}$ a subset of $\{1, \ldots, n\}$ we define the **gradient** of π with respect to the variables in $\mathcal{L}$, denoted by $\nabla_{\mathcal{L}}\pi$, as the monomial ideal generated by the gradient with respect to $\mathcal{L}$ of the stable subset spanned by a set of monomials generating π (this construction is independent of the choice of the generating set).

If π is a zero-dimensional ideal of S (in the sense that the quotient ring S/π has Krull dimension zero) we denote by $\deg(\pi)$ the **degree** of π that is the dimension of the finite dimensional k-algebra S/π. For a zero-dimensional ideal π the complement in $\mathcal{M}$ of $\pi\cap\mathcal{M}$ is finite: we call it the **support** of π. The support of π is a basis of the quotient ring S/π as a k-linear space. The cardinal of the support is finite and equals the degree of π. For example, if $n = 2$, $\mathrm{supp}(x_1^{\alpha_1}, x_2^{\alpha_2}) = \{x_1^ix_2^j,\ 0 \le i \le \alpha_1 - 1,\ 0 \le j \le \alpha_2 - 1\}$.

Proposition 25 *For two subsets $\mathcal{L}_1$ and $\mathcal{L}_2$ of $\{1, \ldots, n\}$ and for any monomial ideal π:*

$$\nabla_{\mathcal{L}_1\cup\mathcal{L}_2}\pi = \nabla_{\mathcal{L}_1}\pi \cap \nabla_{\mathcal{L}_2}\pi.$$

Proof.

$$\begin{aligned}
&\mathrm{supp}(\nabla_{\mathcal{L}_1\cup\mathcal{L}_2}\pi)\\
&\quad= \mathrm{supp}(\pi) \cup \cup_{i\in\mathcal{L}_1\cup\mathcal{L}_2} x_i\mathrm{supp}(\pi)\\
&\quad= \mathrm{supp}(\pi) \cup \Big(\cup_{i\in\mathcal{L}_1} x_i\mathrm{supp}(\pi)\Big) \cup \Big(\cup_{i\in\mathcal{L}_2} x_i\mathrm{supp}(\pi)\Big)\\
&\quad= \mathrm{supp}(\nabla_{\mathcal{L}_1}\pi) \cup \mathrm{supp}(\nabla_{\mathcal{L}_2}\pi).
\end{aligned}$$

□

Let $\pi = (x_1^{\alpha_1}, \ldots, x_n^{\alpha_n})$, for $\alpha_i \ge 0$, $i = 1, \ldots, n$. Let $l \in \{1, \ldots, n\}$ then

$$\nabla_{\{l\}}\pi = (x_1^{\alpha_1}, \ldots, x_{l-1}^{\alpha_{l-1}}, x_l^{\alpha_l+1}, x_{l+1}^{\alpha_{l+1}}, \ldots, x_n^{\alpha_n}),$$

we deduce:

Corollary 4 *Let $\alpha_i \ge 0$, $i = 1, \ldots, n$ and $\mathcal{L}$ be a subset of $\{1, \ldots, n\}$. We have the following formula:*

$$\nabla_{\mathcal{L}}(x_1^{\alpha_1}, \ldots, x_n^{\alpha_n}) = \cap_{i\in\mathcal{L}}(x_1^{\alpha_1}, \ldots, x_{i-1}^{\alpha_{i-1}}, x_i^{\alpha_i+1}, x_{i+1}^{\alpha_{i+1}}, \ldots, x_n^{\alpha_n}).$$

Combined with Corollary 3 we deduce:

Corollary 5 *Let $\alpha_i \geq 0$, $i = 1, \ldots, n$, $\mathcal{L}$ be a subset of $\{1, \ldots, n\}$ and l such that $\alpha_l \geq 1$. We have the following equality:*

$$\frac{\partial}{\partial x_l} \nabla_{\mathcal{L}}(x_1^{\alpha_1}, \ldots, x_n^{\alpha_n}) = \nabla_{\mathcal{L}}\Big(\frac{\partial}{\partial x_l}(x_1^{\alpha_1}, \ldots, x_n^{\alpha_n})\Big).$$

From Corollary 4 we also deduce:

Corollary 6 *If $\alpha_i \geq 0$, for all $i = 1, \ldots, n$, then*

$$\nabla_{\{1,\ldots,n\}}(x_1^{\alpha_1}, \ldots, x_n^{\alpha_n}) \cap k[[x_2, \ldots, x_n]] = \nabla_{\{2,\ldots,n\}}(x_2^{\alpha_2}, \ldots, x_n^{\alpha_n}).$$

We give two useful bounds on the degree of the gradients:

Proposition 26 *According to the above notations, if π is zero-dimensional of degree M then $\nabla_{\mathcal{L}}\pi$ is zero-dimensional and*

$$\deg(\nabla_{\mathcal{L}}\pi) \leq (1 + \#\mathcal{L})\deg(\pi),$$

where $\#\mathcal{L}$ denotes the cardinal of the set $\mathcal{L}$.
If π is generated by $x_1^{\alpha_1}, \ldots, x_n^{\alpha_n}$, with $\alpha_i \geq 1$, for $i = 1, \ldots, n$, we have:

$$\deg(\pi) = \alpha_1 \cdots \alpha_n, \quad \deg(\nabla_{\mathcal{L}}\pi) = \deg(\pi)\Big(1 + \sum_{i \in \mathcal{L}} \frac{1}{\alpha_i}\Big).$$

Proof. The proof is straightforward from the definition of the gradient. □

In our complexity estimates we use the first bound only. But one has to keep in mind that it is not sharp at all as soon as $\pi \subset (x_1, \ldots, x_n)$.

Proposition 27 *For any monomial ideal π and two subsets $\mathcal{L}_1 \subseteq \mathcal{L}_2$ of the set $\{1, \ldots, n\}$ the following inclusions hold:*

$$\pi^2 \subseteq \nabla_{\mathcal{L}_2}\pi \subseteq \nabla_{\mathcal{L}_1}\pi \subseteq \pi.$$

Proof. The inclusions $\nabla_{\mathcal{L}_2}\pi \subseteq \nabla_{\mathcal{L}_1}\pi \subseteq \pi$ are true by construction. We prove that $\pi^2 \subseteq \nabla_{\mathcal{L}_2}\pi$. Let x^α be a monomial in the support of π and $i \in \mathcal{L}_2$. If the monomial $x_i x^\alpha$ were in π^2 we could write it as the product of two monomials of π: $x_i x^\alpha = x^\beta x^\gamma$. One of them would be a multiple of x_i, say x^β, then x^α would be a multiple of x^γ. This is a contradiction. □

The last technical result about gradients of ideals we need is:

Proposition 28 *For any positive integer $\lambda \geq 1$ the following inclusion holds:*

$$(x_1^\lambda, x_2, \ldots, x_n)\nabla_{\{1,\ldots,n\}}(x_1^\lambda, x_2, \ldots, x_n) \subseteq \nabla_{\{1,\ldots,n\}}(x_1^{2\lambda}, x_2, \ldots, x_n).$$

Proof. Let ζ_λ denote $(x_1^\lambda, x_2, \ldots, x_n)$. By construction we have:

$$\mathrm{supp}(\nabla_{\{1,\ldots,n\}}\zeta_\lambda) = \{x_1^j,\ 0 \le j \le \lambda - 1\} \cup \{x_1^j x_i,\ 1 \le i \le n,\ 0 \le j \le \lambda - 1\}.$$

We want to prove that any monomial of the support of $\nabla_{\{1,\ldots,n\}}\zeta_{2\lambda}$ is not in $\mathfrak{I} = \zeta_\lambda \nabla_{\{1,\ldots,n\}}\zeta_\lambda$; let x^α be one of these monomials. First, if x^α is x_1^j, with $j \le 2\lambda$, then it can not be in $\mathfrak{I}$, since the smallest power of x_1 in $\nabla_{\{1,\ldots,n\}}\zeta_\lambda$ is $\lambda + 1$ and λ in ζ_λ. If $x^\alpha = x_1^j x_i$, with $0 \le j \le 2\lambda - 1$ and $i \ne 1$, were in $\mathfrak{I}$ then it could only be the product of x_1^l and $x_1^m x_i$, with $m \ge \lambda$ and $l \ge \lambda$, which is not possible. □

III.2.4 Deflation Lemma

The deflation lemma is the key for bounding the complexity of the deflation algorithm of §III.4: it shows that our deflation process has a good complexity behavior.

Lemma 8 *Let k be a field and Ψ be an ideal of $S := k[[x_1, \ldots, x_n]]$ of valuation m such that*

- *S/Ψ is a finite dimensional k-vector space of dimension $M \ge 1$;*
- *x_n is in Weierstraß position with respect to Ψ;*
- *Either k has characteristic zero or $m + 1 \le \mathrm{char}(k)$.*

We define $\widetilde{\Psi}$, the ***deflated ideal*** *of Ψ by*

$$\widetilde{\Psi} := \frac{\partial^{m-1}\Psi}{\partial {x_n}^{m-1}};$$

this is an ideal containing Ψ of valuation 1 and the dimension $\widetilde{M}$ of the k-vector space $S/\widetilde{\Psi}$ satisfies the following inequality:

$$1 \le \widetilde{M} \le M/m.$$

Proof. Since $\widetilde{\Psi}$ contains Ψ, the quotient $S/\widetilde{\Psi}$ is a finite dimensional k-vector space. Now we order the monomials according to the *anti-graded lexicographic order* defined by $x_1^{\alpha_1} \cdots x_n^{\alpha_n} > x_1^{\beta_1} \cdots x_n^{\beta_n}$ if $\alpha_1 + \cdots + \alpha_n < \beta_1 + \cdots + \beta_n$ or if $\alpha_1 + \cdots + \alpha_n = \beta_1 + \cdots + \beta_n$ and $(\alpha_1, \ldots, \alpha_n)$ is greater than $(\beta_1, \ldots, \beta_n)$ for the pure lexicographic order:

$$\begin{array}{cccccccccc} 1 & > & x_n & > & x_{n-1} & > & \cdots & > & x_1 & \\ & > & x_n^2 & > & x_n x_{n-1} & > & x_n x_{n-2} & > & \cdots & > x_n x_1 \\ & & & > & x_{n-1}^2 & > & \cdots & & & \end{array}$$

This order is compatible with the differentiation with respect to x_n: if both the derivatives of two monomials are not zero, they are in the same order as the monomials.
We denote by $\mathrm{lm}(\Psi)$ (resp. $\mathrm{lm}(\widetilde{\Psi})$) the monomial ideal constituted by the leading monomials of Ψ (resp. $\widetilde{\Psi}$) according to the above order. Note that the cardinal of the complement of $\mathrm{lm}(\Psi)$ (resp. $\mathrm{lm}(\widetilde{\Psi})$) is M (resp. $\widetilde{M}$). Let T be the subset of the complement of $\mathrm{lm}(\Psi)$ composed of the monomials having the power $m - 1$ with respect to x_n:

$$T := \{x_1^{\alpha_1} \cdots x_n^{\alpha_n} \notin \mathrm{lm}(\Psi),\ \alpha_n = m - 1\}.$$

Since M is at least $m|T|$ (where $|T|$ denotes the cardinal of T), it suffices to prove that $\widetilde{M}$ is at most $|T|$. Let A be a leading monomial $x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n}$ of Ψ such that $\alpha_n = m-1$ then $\frac{\partial^{m-1}A}{\partial {x_n}^{m-1}}$ is not zero (for we have $\mathrm{char}(k) \geq m+1$) and is a leading monomial of $\widetilde{\Psi}$ (Note that $x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n}$ cannot be x_n^{m-1}).
Now using the monomial $A = x_1^{\alpha_1}\cdots x_{n-1}^{\alpha_{n-1}}x_n^m$ which belongs to $\mathrm{lm}(\Psi)$ for any tuple $(\alpha_1,\ldots,\alpha_{n-1})$ by the Weierstraß position property and from the fact that $\mathrm{char}(k) \geq m+1$, we deduce that the monomial $x_1^{\alpha_1}\cdots x_{n-1}^{\alpha_{n-1}}x_n$ is a leading monomial of $\widetilde{\Psi}$. We deduce that the complement of $\mathrm{lm}(\widetilde{\Psi})$ is included in the set

$$\{x_1^{\alpha_1}\cdots x_{n-1}^{\alpha_{n-1}} \quad | \quad x_1^{\alpha_1}\cdots x_{n-1}^{\alpha_{n-1}}x_n^{m-1} \in T\}.$$

We are done. □

We exhibit an example where the Weierstraß condition is a necessary hypothesis: let $n := 2$ and $\Psi := (x_1x_2, x_1^a, x_2^b)$, with $a \geq 3$ and $b \geq 3$. With $m = 2$ and $M = a+b-1$, the ideal Ψ satisfies the hypotheses of the deflation lemma save the Weierstraß position. Its corresponding deflated ideal $\widetilde{\Psi} = \frac{\partial \Psi}{\partial x_2}$ is (x_1, x_2^{b-1}) and has degree $\widetilde{M} := b-1$ so that the conclusion of the lemma holds if and only if $b \leq a+1$.

III.3 Deflation Sequence

We recall that $F := \{f_1,\ldots,f_s\}$ is a finite subset of $\mathfrak{o}[x_1,\ldots,x_n]$ and x^* is an isolated root of $f_1 = \cdots = f_s = 0$ with multiplicity M. First we construct a sequence of ideals of $\overline{k}[[x_1 - x_1^*,\ldots,x_n - x_n^*]]$ starting from F with decreasing multiplicity. Then we examine the branchings and deduce conditions under which the computations remain valid when replacing $\overline{k}$ by $A = \hat{\mathfrak{o}}[T]/(q(T))$ and x^* by an approximate root. We assume that either $\mathrm{char}(k) = 0$ or $\mathrm{char}(k) \geq M+1$.

III.3.1 Exact Construction

In this subsection we assume that we are given a root $x^* \in \overline{k}^n$ of $f_1,\ldots,f_s$ isolated for the Zariski topology, and of multiplicity M. We denote by Φ the set F viewed as a subset of $S := \overline{k}[[x_1 - x_1^*,\ldots,x_n - x_n^*]]$. The ideal generated by the elements of Φ in S is not trivial and the quotient ring S/Φ is a finite dimensional $\overline{k}$-vector space of dimension M ($M \geq 1$). We construct a sequence of *deflated ideals*. This sequence starts from Φ, each step is achieved by combining the action of a well chosen differentiation and the elimination of well chosen variables.

By induction we define the sequences of integers R_i and the deflated subsets Φ_i of S, for $i \geq 1$ as follows:

- $R_1 := 1$;
- $\Phi_1 := \Phi$.

At step $i \geq 1$ we know R_i and Φ_i, we now describe how we compute R_{i+1} and Φ_{i+1}. We use the following notations:

- $S_i := \overline{k}[[x_{R_i} - x_{R_i}^*,\ldots,x_n - x_n^*]]$;

- $M_i := \dim_{\overline{k}}(S_i/\Phi_i)$;
- $m_i := \mathrm{val}(\Phi_i)$.

We assume that

(W_i) x_{R_i} is in Weierstraß position with respect to Φ_i.

The ideal Φ_i satisfies the conditions of the deflation lemma (Lemma 8), we deflate it once. We write $\frac{\partial \widetilde{\Phi}_i}{\partial \{x_{R_i}, \ldots, x_n\}}$ for the Jacobian matrix of the elements of $\widetilde{\Phi}_i$ with respect to the variables $x_{R_i}, \ldots, x_n$.

- $\widetilde{\Phi}_i := \frac{\partial^{m_i - 1}}{\partial {x_{R_i}}^{m_i - 1}} \Phi_i$;
- $\widetilde{M}_i := \dim(S_i/\widetilde{\Phi}_i)$;
- $r_i := \mathrm{rank}\Big(\frac{\partial \widetilde{\Phi}_i}{\partial \{x_{R_i}, \ldots, x_n\}}(x^*_{R_i}, \ldots, x^*_n)\Big)$, this is the rank of the set of the gradient vectors at x^* of the elements of $\widetilde{\Phi}_i$.

According to Lemma 8 the following properties hold:

- $1 \leq r_i \leq n - R_i + 1$;
- $1 \leq \widetilde{M}_i \leq M_i/m_i$.

We set $R_{i+1} := R_i + r_i$ and extract a subset Ω_i of cardinal r_i from $\widetilde{\Phi}_i$ such that: the gradient of Ω_i at $(x^*_{R_i}, \ldots, x^*_n)$ has rank r_i and, up to a permutation of the variables, there exist power series $y_{R_i}, \ldots, y_{R_{i+1}-1}$ in S_{i+1} satisfying $x_j = y_j$ in S_i/Ω_i, for $R_i \leq j \leq R_{i+1} - 1$ (thanks to the implicit function theorem). We define the elimination map G_i as follows:

$$\begin{array}{rccl} G_i : & S_i & \to & S_{i+1} \\ & \phi & \mapsto & \phi(y_{R_i}, \ldots, y_{R_{i+1}-1}, x_{R_{i+1}}, \ldots, x_n). \end{array}$$

From a practical point of view the rank computation and the extraction of such a subset Ω_i can be achieved using classical Gaussian elimination. The lucky choice of the maximal ideal $\mathfrak{m}$, involved with this part of the computation, relates to all the equality tests and inversions. We discuss these aspects more in Proposition 37 of §III.4.3.

Lemma 9 *The following property holds:*

$$\mathrm{val}(G_i(\widetilde{\Phi}_i)) \geq 2.$$

Proof. From Lemma 8, $\widetilde{\Phi}_i$ has valuation 1. Let ϕ be an element of $\widetilde{\Phi}_i$ of valuation 1; since the gradient of ϕ is a linear combination of the gradients of the elements of Ω_i, the valuation of $G_i(\phi)$ is at least 2. □

Corollary 7 $m_i \geq 2$, *for* $i \geq 2$.

Last, we define

$$\Phi_{i+1} := G_i(\widetilde{\Phi}_i).$$

The above construction stops once we have exhausted all the variables, that is when $R_{i+1} = n+1$. We let ν be such that $R_{\nu+1} = n+1$ and call it the **depth** of the deflation. The main output of this process is the sequence $(\Omega_i)_{i=1,\ldots,\nu}$ such that

$$\begin{cases} \Omega_1(x^*_{R_1}, \ldots, x^*_n) = 0, \\ \Omega_2(x^*_{R_2}, \ldots, x^*_n) = 0, \\ \cdots \\ \Omega_\nu(x^*_{R_\nu}, \ldots, x^*_n) = 0, \end{cases}$$

and the Jacobian matrix of the union of the Ω_i is invertible at x^*. Our algorithm presented in §III.4 consists essentially in applying the classical Newton iterator on the above system. The difficulty is to find an efficient way for evaluating the Ω_is and their gradient vectors in a neighborhood of x^*.

The sequence $m_1, \ldots, m_\nu$ is called the **multiplicity sequence** associated to the deflation. The crucial quantity appearing in the complexity estimate of our algorithm is $m_1 \cdots m_\nu$. Noting that M_{i+1} equals $\widetilde{M_i}$, we deduce the following proposition:

Proposition 29 *For $1 \le i \le \nu$ we have the following inequalities:*

$$m_1 \cdots m_{i-1} M_i \le M, \qquad m_1 \cdots m_\nu \le M.$$

Concerning the Weierstraß conditions (W_i), we successively apply Lemma 5 to deduce:

Proposition 30 *The linear changes of variables for which not all the (W_i) hold are enclosed in an algebraic hypersurface of $GL_n(k)$.*

From a computational point of view we first perform a generic linear change of coordinates and then apply a deterministic deflation process (performing no random choice). With respect to this generic linear change of coordinates we can speak about a generic multiplicity sequence as the multiplicity sequence found on a Zariski open subset of $GL_n(k)$. We detail this point of view later in §III.3.5 more precisely.

Block Notations It is natural to introduce the following block notation, for each i, $1 \le i \le \nu$:

- $X_i := x_{R_i}, \ldots, x_{R_{i+1}-1}$;
- $Y_i := y_{R_i}, \ldots, y_{R_{i+1}-1}$.

By convention we consider that Y_0 is the empty sequence.

III.3.2 Example 3 continued

We are coming back to the example of the beginning: $n = 3$, $\mathfrak{o} = \mathbb{Z}$, $k = \mathbb{Q}$, and $F = f_1, f_2, f_3$, where

$$\begin{aligned} f_1 &:= 2\,x_1 + 2\,x_1^2 + 2\,x_2 + 2\,x_2^2 + x_3^2 - 1, \\ f_2 &:= (x_1 + x_2 - x_3 - 1)^3 - x_1^3, \\ f_3 &:= (2\,x_1^3 + 5\,x_2^2 + 10\,x_3 + 5\,x_3^2 + 5)^3 - 1000\,x_1^5. \end{aligned}$$

The root $x^* = (0, 0, -1)$ lies in k^3 and has multiplicity 18. We illustrate the construction of the deflation sequence. At x^* we examine the rank of the Jacobian matrix of f_1, f_2, f_3:

$$\frac{df_1}{dx}(x^*) = (2,2,2),\ \frac{df_2}{dx}(x^*) = (0,0,0),\ \frac{df_3}{dx}(x^*) = (0,0,0).$$

We find that $m_1 = 1$, $\widetilde{\Phi}_1 = \Phi_1$, x_1 is in Weierstraß position, $r_1 = 1$ and $\Omega_1 = \{f_1\}$. We compute y_1 as the power series solution of $f_1(y_1, x_2, x_3) = 0$ in $k[[x_2, x_3 + 1]]$ such that $y_1 = 0 + \mathcal{O}(x_2, x_3 + 1)$:

$$\begin{aligned} y_1 = {} & (x_3+1) - \frac{3}{2}(x_3+1)^2 + 3\,(x_3+1)^3 - \frac{33}{4}(x_3+1)^4 \\ & + x_2\Big(-1 + 2\,(x_3+1) - 7\,(x_3+1)^2 + 26\,(x_3+1)^3 - \frac{203}{2}(x_3+1)^4\Big) \\ & + x_2^2\Big(-2 + 8\,(x_3+1) - 40\,(x_3+1)^2 + 196\,(x_3+1)^3 - 949\,(x_3+1)^4\Big) \\ & + x_2^3\Big(-4 + 32\,(x_3+1) - 216\,(x_3+1)^2 + 1320\,(x_3+1)^3 - 7610\,(x_3+1)^4\Big) \\ & + x_2^4\Big(-12 + 136\,(x_3+1) - 1148\,(x_3+1)^2 + 8360\,(x_3+1)^3 - 55710\,(x_3+1)^4\Big) \\ & + x_2^5\Big(-40 + 592\,(x_3+1) - 6008\,(x_3+1)^2 + 50736\,(x_3+1)^3 - 383124\,(x_3+1)^4\Big) \\ & + x_2^6\Big(-144 + 2624\,(x_3+1) - 31104\,(x_3+1)^2 + 298592\,(x_3+1)^3 \\ & -2517368\,(x_3+1)^4\Big) + \mathcal{O}(x_2^7, (x_3+1)^5). \end{aligned}$$

Substituting x_1 by y_1 in F we deduce $\Phi_2 = \{\phi_1, \phi_2, \phi_3\}$ where:

$$\begin{aligned} \phi_1 = {} & 0 + \mathcal{O}(x_2^7, (x_3+1)^5), \\ \phi_2 = {} & -(x_3+1)^3 + \frac{9}{2}(x_3+1)^4 \\ & + x_2\Big(3\,(x_3+1)^2 - 15\,(x_3+1)^3 + \frac{255}{4}(x_3+1)^4\Big) \\ & + x_2^2\Big(-3\,(x_3+1) + \frac{45}{2}(x_3+1)^2 - 123\,(x_3+1)^3 + \frac{2367}{4}(x_3+1)^4\Big) \\ & + x_2^3\Big(1 - 18\,(x_3+1) + 135\,(x_3+1)^2 - 822\,(x_3+1)^3 + \frac{9357}{2}(x_3+1)^4\Big) \\ & + x_2^4\Big(6 - 84\,(x_3+1) + 708\,(x_3+1)^2 - 5112\,(x_3+1)^3 + 33699\,(x_3+1)^4\Big) \\ & + x_2^5\Big(24 - 360\,(x_3+1) + 3636\,(x_3+1)^2 - 30480\,(x_3+1)^3 + 228324\,(x_3+1)^4\Big) \\ & + x_2^6\Big(84 - 1560\,(x_3+1) + 18468\,(x_3+1)^2 - 176520\,(x_3+1)^3 \\ & +1480290\,(x_3+1)^4\Big) + \mathcal{O}(x_2^7, (x_3+1)^5), \\ \phi_3 = {} & 5000\,x_2(x_3+1)^4 \end{aligned}$$

$$+x_2^3\Big(-10000\,(x_3+1)^3+95375\,(x_3+1)^4\Big)$$
$$+x_2^4\Big(10000\,(x_3+1)^2-130000\,(x_3+1)^3+1031450\,(x_3+1)^4\Big)$$
$$+x_2^4\Big(-5000\,(x_3+1)+107875\,(x_3+1)^2-1113950\,(x_3+1)^3+8959725\,(x_3+1)^4\Big)$$
$$+x_2^5\Big(1000-50000\,(x_3+1)+774250\,(x_3+1)^2-8043910\,(x_3+1)^3+\frac{137720739}{2}(x_3+1)^4\Big)$$
$$+x_2^6\Big(10125-319550\,(x_3+1)+4815785\,(x_3+1)^2-53037098\,(x_3+1)^3$$
$$+\frac{978677779}{2}(x_3+1)^4\Big)+\mathcal{O}(x_2^7,(x_3+1)^5).$$

We see that $m_2=3$ and that x_2 is in Weierstraß position. We deduce $\widetilde{\Phi}_2$ at precision $\mathcal{O}(x_2^5,(x_3+1)^5)$, it contains 9 elements:

$$\widetilde{\Phi}_2=\left\{\phi_1,\phi_2,\phi_3,\frac{\partial\phi_1}{\partial x_2},\frac{\partial\phi_2}{\partial x_2},\frac{\partial\phi_3}{\partial x_2},\frac{\partial^2\phi_1}{\partial {x_2}^2},\frac{\partial^2\phi_2}{\partial {x_2}^2},\frac{\partial^2\phi_3}{\partial {x_2}^2}\right\}.$$

The gradients at point $(0,-1)$ of these elements are respectively

$$(0,0),\ (0,0),\ (0,0),\ (0,0),\ (0,0),\ (0,0),\ (0,0),\ (6,-6),\ (0,0).$$

Therefore we get $r_2=1$, $\Omega_2=\{\frac{\partial^2\phi_2}{\partial {x_2}^2}\}$ and compute y_2 as the power series solution of $\frac{\partial^2\phi_2}{\partial {x_2}^2}(y_2,x_3)=0$ in $k[[x_3+1]]$ such that $y_2=0+\mathcal{O}(x_3+1)$:

$$y_2=(x_3+1)-\frac{3}{2}(x_3+1)^2+3\,(x_3+1)^3+\frac{9}{4}(x_3+1)^4+\mathcal{O}((x_3+1)^5).$$

Substituting x_2 by y_2 in $\widetilde{\Phi}_2$ we find $\Phi_3=\{\varphi_1,\ldots,\varphi_9\}$ where

$$\begin{array}{rcl}\varphi_l & = & \phi_l(y_2,x_3),\ \text{ for } l=1,2,3,\\ \varphi_l & = & \frac{\partial\phi_l}{\partial dx_2}(y_2,x_3),\ \text{ for } l=4,5,6,\\ \varphi_l & = & \frac{\partial^2\phi_l}{\partial d{x_2}^2}(y_2,x_3),\ \text{ for } l=7,8,9.\end{array}$$

We compute $\varphi_l=0+\mathcal{O}((x_3+1)^5)$ for $l=1,\ldots,8$ and $\varphi_9=9000(x_3+1)^4+\mathcal{O}((x_3+1)^5)$. We deduce that $m_3=4$ and $\widetilde{\Phi}_3$:

$$\widetilde{\Phi}_3=\Big\{\frac{\partial^j\varphi_l}{\partial d{x_3}^j},\ l=1,\ldots,9,\ j=0,\ldots,3\Big\}.$$

All the elements of $\widetilde{\Phi}_3$ equal $0+\mathcal{O}((x_3+1)^2)$ except $\frac{\partial^3\varphi_9}{\partial d{x_3}^3}=216000(x_3+1)+\mathcal{O}((x_3+1)^2)$. It is obvious that x_3 is in Weierstraß position. We deduce $r_3=1$ and finally the depth $\nu=3$. As expected, the product $m_1m_2m_3=12$ is bounded by the multiplicity $M=18$.

III.3.3 Smoothness Hypotheses

We revisit the construction of the deflation sequence presented above. We gather conditions on $\mathfrak{m}$ under which the Y_i are well-defined over A. Theses conditions are weak in the sense that they do not allow the computations of the ranks and the selections of Ω_i

over A instead of $k(u)$. Stronger conditions are established in Proposition 37 of §III.4.3 during the presentation of the algorithm.

Let $A_i := A[[x_{R_i} - x^*_{R_i}, \ldots, x_n - x^*_n]]$, for $i = 1, \ldots, \nu$. We assume by induction on i that Y_i is well-defined in A_{i+1}. Since Y_0 is the empty sequence, this is trivially true for $i = 0$. Let us assume that this holds for $i \geq 1$. To go from step $i-1$ to i we assume that the Jacobian matrix of Ω_i with respect to the variables in X_i evaluated at x^* is invertible in $A/\mathfrak{m}A$:

(H_{Ω_i}) The determinant of $\frac{\partial \Omega_i}{\partial X_i}(x^*_{R_i}, \ldots, x^*_n)$ is a unit of $A/\mathfrak{m}A$.

In particular, from the implicit function theorem this condition implies that Y_i belongs to A_{i+1}. We denote by (H_Ω) the set of hypotheses (H_{Ω_i}) for $i = 1, \ldots, \nu$. We recall from §III.1.1 that p is the canonical projection from $k[T]$ to $k[u]$ and p^{-1} the linear map such that $p \circ p^{-1} = \mathrm{Id}$ and $p^{-1}(z)$ has degree strictly less than $\deg(Q)$. We define $a_i \in k$ as the resultant of $p^{-1}(\det(\frac{\partial \Omega_i}{\partial X_i}(x^*_{R_i}, \ldots, x^*_n)))$ with Q. We can write $a_i = \rho_{\Omega_i}/\overline{\rho}_{\Omega_i}$ with $\rho_{\Omega_i} \in o$, and $\overline{\rho}_{\Omega_i} \notin \mathfrak{m}$.

Proposition 31 *Let $\rho_\Omega = \rho_{\Omega_1} \cdots \rho_{\Omega_\nu}$. For any maximal ideal $\mathfrak{m}$ of $\mathfrak{o}$ that does not contain ρ_Ω, hypothesis (H_Ω) is satisfied.*

Indeed (H_Ω) generalizes (H_J) of §III.1.1 in the following sense:

Proposition 32 *Under hypothesis (H_Ω). Let z be an approximation of x^* (that is $z - x^* \in (\mathfrak{m}A)^n$) such that*

$$\begin{cases} \Omega_1(z_{R_1}, \ldots, z_n) = 0, \\ \Omega_2(z_{R_2}, \ldots, z_n) = 0, \\ \cdots \\ \Omega_\nu(z_{R_\nu}, \ldots, z_n) = 0, \end{cases}$$

then $z = x^*$.

Proof. The proof is immediate since the Jacobian matrix of the above system is invertible at x^* modulo $\mathfrak{m}A$. □

In the example of §III.3.2 the computation of the deflation sequence can be performed modulo any prime number p different from 2, 3 and 5: we need to invert 2, 6 and $216000 = 2^6 3^3 5^3$.

III.3.4 Nested Coordinates

The idea behind our lifting algorithm is to use the classical Newton iterator on the system of equations of Proposition 32. Hence, we only need to compute values and gradients of the Ω_i at various points in a neighborhood of x^* efficiently. Computing these values straightforwardly as in §III.3.2 is not the best way: the requested precision with respect to the variable x_2 is $m_2 + m_3 = 7$ whereas the method we are to present requires precision $\nabla_{\{1,2,3\}}(x_1^{m_1}, x_2^{m_2}, x_3^{m_3})$ only. Our method relies on what we call the *nested coordinates* associated to the deflation sequence.

We introduce a new set of variables $dx_1, \ldots, dx_n$ with its associated blocks dX_i (as in §III.3.1). For $i = 1, \ldots, \nu$:

- $dX_i := dx_{R_i}, \ldots, dx_{R_{i+1}-1}$.
- The associated power series rings over $\overline{k}$: $dS_i := \overline{k}[[dx_{R_i}, \ldots, dx_n]]$.
- The associated power series rings over A: $dA_i := A[[dx_{R_i}, \ldots, dx_n]]$.

We also introduce the function $\mathbf{coeff}_i(f, dx^\alpha)$ returning the coefficient of f with respect to the monomial dx^α, seen as a power series in the variables $dX_1, \ldots, dX_i$ only.
The definition of Φ_{i+1} of §III.3.1 yields the following recursive formula:

$$\begin{aligned}
\Phi_{i+1} &= G_i(\widetilde{\Phi}_i) \\
&= \frac{\partial^{m_i-1}\Phi_i}{\partial {x_{R_i}}^{m_i-1}}(Y_i, x_{R_{i+1}}, \ldots, x_n) \\
&= \{\alpha_i! \operatorname{coeff}_i(\phi(Y_i + dX_i, x_{R_{i+1}}, \ldots, x_n), dx_{R_i}^{\alpha_i}), \\
&\qquad \phi \in \Phi_i,\ 0 \le \alpha_i \le m_i - 1\}.
\end{aligned}$$

Solving this recurrence leads to a fast evaluation scheme for the Φ_is. For this purpose and each $i = 1, \ldots, \nu$, we introduce the ith **nested coordinates** Y^i as the sequence $(Y_1^i, \ldots, Y_i^i)$ with entries in

$$\overline{k}[[dX_1, \ldots, dX_i]][[x_{R_{i+1}} - x^*_{R_{i+1}}, \ldots, x_n - x^*_n]].$$

The entries of Y^i are recursively defined from the last one to the first one:

$$\begin{aligned}
Y_i^i &:= Y_i(x_{R_{i+1}}, \ldots, x_n), \\
Y_{i-1}^i &:= Y_{i-1}(Y_i^i + dX_i, x_{R_{i+1}}, \ldots, x_n), \\
&\cdots \\
Y_1^i &:= Y_1(Y_2^i + dX_2, \ldots, Y_i^i + dX_i, x_{R_{i+1}}, \ldots, x_n).
\end{aligned}$$

By convention we set $Y^0(x_1, \ldots, x_n)$ to be the empty sequence. Under hypothesis (H_Ω), Y^i is well-defined over A instead of $\overline{k}$. Next proposition describes how nested coordinates are related. Note that we use the comma operator for the concatenation of sequences.

Proposition 33 *Let V denote $x_{R_{i+2}}, \ldots, x_n$. The nested coordinates are related to each other by the following formula:*

$$Y^{i+1} = Y^i(Y_{i+1}(V) + dX_{i+1}, V), Y_{i+1}(V),$$

for $i = 0, \ldots, \nu - 1$.

The deflated sets Φ_i can be computed by evaluating the input set F at the nested coordinates.

Proposition 34 *For $i = 0, \ldots, \nu - 1$, the $(i+1)$st deflated set Φ_{i+1} is given by:*

$$\begin{aligned}
\Phi_{i+1} \;=\; \Big\{ &\alpha_1! \cdots \alpha_i! \operatorname{coeff}_i\Big(\phi(Y_1^i + dX_1, \ldots, Y_i^i + dX_i, x_{R_{i+1}}, \ldots, x_n), \\
&dx_{R_1}^{\alpha_1} \cdots dx_{R_i}^{\alpha_i}\Big),\quad \phi \in F,\ 0 \le \alpha_j \le m_j - 1,\ 1 \le j \le i \Big\}.
\end{aligned}$$

Proof. The proof is done by induction on i. If $i = 0$, we have $\Phi_1 = F$ and Y^0 is the empty sequence, the formula is true. Assume that the formula holds for $i \geq 0$. First we have:

$$\begin{aligned}\Phi_{i+2} &= \frac{\partial^{m_{i+1}-1}\Phi_{i+1}}{\partial {x_{R_{i+1}}}^{m_{i+1}-1}}(Y_{i+1}, x_{R_{i+2}}, \ldots, x_n) \\ &= \Big\{\alpha_{i+1}!\,\mathrm{coeff}_{i+1}(\phi(Y_{i+1} + dX_{i+1}, x_{R_{i+2}}, \ldots, x_n), dx_{R_{i+1}}^{\alpha_{i+1}}), \\ &\qquad \phi \in \Phi_{i+1},\ 0 \leq \alpha_{i+1} \leq m_{i+1} - 1\Big\}.\end{aligned}$$

Using the induction hypothesis giving Φ_{i+1} and letting

$$V_i := Y_{i+1} + dX_{i+1}, x_{R_{i+2}}, \ldots, x_n,$$

we obtain:

$$\begin{aligned}\Phi_{i+2} &= \Big\{\alpha_1! \cdots \alpha_{i+1}!\,\mathrm{coeff}_{i+1}\Big(\phi(Y_1^i(V_i) + dX_1, \ldots, Y_i^i(V_i) + dX_i, V_i), \\ &\qquad dx_{R_1}^{\alpha_1} \cdots dx_{R_{i+1}}^{\alpha_{i+1}}\Big),\ \phi \in F,\ 0 \leq \alpha_j \leq m_j - 1,\ 1 \leq j \leq i+1\Big\}.\end{aligned}$$

Using Proposition 33 we recognize the desired formula for $i + 1$. □

At last, we show in §III.4 how to compute Y^ν efficiently:

$$\begin{aligned}Y_\nu^\nu &= Y_\nu(), \\ Y_{\nu-1}^\nu &= Y_{\nu-1}(Y_\nu^\nu + dX_\nu), \\ &\cdots \\ Y_1^\nu &= Y_1(Y_2^\nu + dX_2, \ldots, Y_\nu^\nu + dX_\nu).\end{aligned}$$

But before this, it remains to make the deflation process deterministic with respect to the choices of the sets Ω_i and to present the complexity model we consider.

Example 3 (continued from §III.3.2) Here are the nested coordinates Y^3:

$$\begin{aligned}Y_3^3 &= -1, \\ Y_2^3 &= y_2(Y_3^3 + dx_3) \\ &= dx_3 - \frac{3}{2}dx_3^2 + 3\,dx_3^3 + \frac{9}{4}dx_3^4 + \mathcal{O}(dx_3^5), \\ Y_1^3 &= y_1(Y_2^3 + dx_2, Y_3^3 + dx_3) \\ &= -\frac{21}{2}{dx_3}^4 + dx_2\Big(-1 - 2\,dx_3 + 3\,dx_3^2 - 6\,dx_3^3 - \frac{51}{2}dx_3^4\Big) \\ &\quad + dx_2^2\Big(-2 - 4\,dx_3 + 2\,dx_3^2 - 126\,dx_3^4\Big) \\ &\quad + dx_2^3\Big(-4 - 16\,dx_3 + 8\,dx_3^3 + 18542\,dx_3^4\Big) \\ &\quad + dx_2^4\Big(-12 - 64\,dx_3 - 48\,dx_3^2 + 19120\,dx_3^3 - 351000\,dx_3^4\Big) \\ &\quad + dx_2^5\Big(-40 - 272\,dx_3 + 11032\,dx_3^2 - 162096\,dx_3^3 + 1733652\,dx_3^4\Big) \\ &\quad + dx_2^6\Big(-144 + 2624\,dx_3 - 31104\,dx_3^2 + 298592\,dx_3^3 - 2517368\,dx_3^4\Big) \\ &\quad + \mathcal{O}(dx_2^7, dx_3^5).\end{aligned}$$

III.3.5 Tracing the Deflation

During the deflation process presented in §III.3 the choice of the Ω_is are not really specified: different choices may be possible for the Weierstraß positions and the variables with respect to which the implicit function theorem is applied. We adopt the following point of view: the deflation process is seen as a deterministic procedure. In particular the variables on which the implicit function theorem is applied are not chosen at random. But the deflation process is parametrized by the change of the coordinates. From this point of view, Proposition 30 tells us that the deflation process works well for almost all choices of this parameter.

In order to explain how we track the deflation process in practice, we introduce notations. For $i = 1, \ldots, \nu$, we let $\boldsymbol{\pi_i}$ denote the monomial ideal

$$\pi_i := (dx_{R_i}^{m_i}, dx_{R_i+1}, \ldots, dx_{R_{i+1}-1})$$

and $\boldsymbol{\pi_{i,j}}$ denote the monomial ideal $\pi_i + \cdots + \pi_j$, for $i \leq j$. We use these ideals over various base rings. We shall specify the considered base ring in each case.

From Proposition 34 it follows that each element of Φ_{i+1} is determined by an element of F and a monomial dx^α in the support of $\pi_{1,i}$, where $\pi_{1,i}$ is seen as an ideal in the variables $dX_1, \ldots, dX_i$. In the same way, to each element of $\widetilde{\Phi}_{i+1}$ corresponds a triple $\tau := (l, dx^\alpha, \mu)$ in $\{1, \ldots, s\} \times \operatorname{supp}(\pi_{1,i}) \times \{0, \ldots, m_{i+1} - 1\}$: this element of $\widetilde{\Phi}_{i+1}$ is the μth derivative with respect to $dx_{R_{i+1}}$ of the element of Φ_{i+1} corresponding to f_l and dx^α.

The **trace** of the deflation is a sequence $(\mathcal{T}_i)_{1 \leq i \leq \nu}$; each $\mathcal{T}_i$ is itself a sequence $\tau_1, \ldots, \tau_{r_i}$ of r_i elements of $\{1, \ldots, s\} \times \operatorname{supp}(\pi_{1,i-1}) \times \{0, \ldots, m_i - 1\}$. Each $\tau_j = (l, dx^\alpha, \mu)$ is bijectively related to an element of Ω_i. That is: $\Omega_i = \{\omega_{\tau_j},\ \tau_j \in \mathcal{T}_i\}$, where

$$\begin{aligned} \gamma_\tau &= f_l(Y_1^{i-1} + dX_1, \ldots, Y_{i-1}^{i-1} + dX_{i-1}, x_{R_i}, \ldots, x_n) \text{ and} \\ \omega_{\tau_j} &= \alpha_1! \cdots \alpha_{i-1}! \, \frac{\partial^\mu}{\partial {dx_{R_i}}^\mu} \operatorname{coeff}_{i-1}(\gamma_\tau, dx^\alpha). \end{aligned} \tag{III.4}$$

The smoothness hypothesis (H_Ω) is equivalent to the fact that the trace of the deflation sequence over $\mathfrak{o}/\mathfrak{m}$ is the same as over $\overline{k}$.

It is important to notice that there exists a Zariski open subset of $GL_n(k)$ for the change of coordinates on which the trace is constant: we call this constant the **generic trace**. From now on we say that the change of coordinates taken as parameter of the deflation process is **generic enough** if the Weierstraß positions are satisfied and if the trace of the deflation process equals the generic trace.

The generic trace depends on the deflation algorithm used: two different deflation programs may lead to distinct generic traces according to their choices of the subsets Ω_i. Therefore our lifting process has two stages: first we must compute a generic trace, then we can apply the classical Newton iterator on the corresponding system of Ω_i. The smoothness conditions (H_Ω) restricting the choices of $\mathfrak{m}$ ensure that the second stage works fine but not necessarily the first one. We come back to this point in §III.4.3.

Example 3 (continued from §III.3.2) The trace is $\mathcal{T}_1 = (1,1,0)$, $\mathcal{T}_2 = (2, dx_1^0, 2)$, $\mathcal{T}_3 = (3, dx_1^0 dx_2^2, 3)$. For instance the element of Ω_3 is obtained from the evaluation of f_3 this way:

$$\begin{aligned}
\Omega_3(Y_3^3 + dX_3) &= \left\{ 2\frac{\partial^3}{\partial {dx_3}^3} \mathrm{coeff}_2(f_3(Y_1^3 + dX_1, Y_2^3 + dX_2, Y_3^3 + dX_3), dx_1^0 dx_2^2) \right\} \\
&= \left\{ 2\frac{\partial^3}{\partial {dx_3}^3} \mathrm{coeff}_2(4500\, dx_2^2 dx_3^4 + \mathcal{O}(dx_2^3, dx_3^5), dx_1^0 dx_2^2) \right\} \\
&= \left\{ 2\frac{\partial^3}{\partial {dx_3}^3} 4500\, dx_3^4 + \mathcal{O}(dx_3^5) \right\} \\
&= \left\{ 216000\, dx_3 + \mathcal{O}(dx_3^2) \right\}.
\end{aligned}$$

III.4 Algorithm

We are now ready to present our algorithm. It is based on the following idea: we compute the nested coordinates Y^ν associated to the input system F at the root x^* in A modulo $\mathfrak{m}A$ first and then lift these coordinates modulo $(\mathfrak{m}A)^\kappa$ for arbitrary integer κ. During the first stage we compute a generic trace as defined in §III.3.5, $\mathfrak{m}$ must be lucky and the coordinates generic to ensure the correctness of the answer. Once a generic trace is found together with the nested coordinates modulo $\mathfrak{m}A$ then we are sure that the lifting process works fine. We first detail the complexity model we use and the costs of each elementary operation.

III.4.1 Complexity Model

Two data structures are used by the algorithm. First the input polynomials $f_1, \ldots, f_s \in \mathfrak{o}[x_1, \ldots, x_n]$ are given by a **straight-line program** [Str72, Gat86, Sto89, Hei89] of length L containing neither test nor branching: for any ring R, for any partially defined ring morphism c from $\mathfrak{o}$ to R and any point $a := (a_1, \ldots, a_n)$ in R^n we can compute the values $f_1(a), \ldots, f_s(a)$ performing L binary operations in R (additions, subtractions and multiplications) and at most L calls to c. Therefore each evaluation costs at most L times the maximum of the cost of an elementary binary arithmetic operation in R or call to c.

The second data structure we use is for multivariate power series. For any ring R and any zero-dimensional monomial ideal π of $S := R[[x_1, \ldots, x_n]]$, we need to compute in S/π. We count the number of arithmetic operations performed in R ($+$, $\times$, $=$) but also the arithmetic operations with the exponents of the monomials.

Proposition 35 *For a given π, we can construct S/π in*

$$\mathcal{O}\Big(n^2 \deg(\pi)^2 \log(\deg(\pi))\Big).$$

Then we have the following complexity estimates.

- *The construction of an element given by a list of l terms (couple of coefficient and exponent) costs $\mathcal{O}\Big(nl \log(\deg(\pi))\Big)$.*

- *The extraction of a coefficient costs* $\mathcal{O}\Big(n\log(\deg(\pi))\Big)$.
- *The cost of each elementary binary arithmetic operations* $(+, -, \times, =)$ *in* S/π *is in* $\mathcal{O}\Big(\deg(\pi)^2\Big)$.
- *The inverse costs* $\mathcal{O}\Big(\deg(\pi)^2\log(\deg(\pi))\Big)$.
- *For any* i, $1 \leq i \leq n$, *and any* $\psi \in S/\pi$, $\frac{\partial\psi}{\partial x_i}$ *costs* $\mathcal{O}\Big(\deg(\pi)\Big)$.
- *For any* $x^\alpha \notin \pi$ *and any* $\psi \in S/\pi$, $\frac{\partial^{|\alpha|}\psi}{\partial x^\alpha}$ *costs* $\mathcal{O}\Big(\deg(\pi)^2\Big)$.
- *For any* $x^\alpha \in R[[x_1, \ldots, x_i]]$ *with* $x^\alpha \notin \pi$ *and any* $\psi \in S/\pi$, *the extraction of the coefficient of the monomial* x^α *of* ψ *seen as an element of* $R[[x_{i+1}, \ldots, x_n]][[x_1, \ldots, x_i]]$ *costs* $\mathcal{O}\Big(n\deg(\pi)\log(\deg(\pi))\Big)$.

Proof. We use a dense representation and naive algorithms for the arithmetic operations. The dense representation is implemented as follows: first we choose a total order on the monomials such that each comparison between two monomials of $\mathrm{supp}(\pi)$ costs $\mathcal{O}(n)$ (for instance we can take the lexicographical order). Then we sort the support of π according to this order, we obtain an array E representing a map from the integer range $N := [1, \ldots, \deg(\pi)]$ to $\mathrm{supp}(\pi)$. Each element of S/π is stored in an array of size $\deg(\pi)$: the ith entry is its coefficient with respect to the monomial $E(i)$. During the initialization of S/π we also precompute the multiplication table of the monomials of the support of π: this is a two dimensional array $F : N \times N \to N$, such that $F(i,j)$ is zero if $x^\beta := E(i)E(j)$ is in π and $E^{-1}(x^\beta)$ otherwise. Let us detail the costs of these precomputations:

- The construction of E costs $\mathcal{O}(n\deg(\pi)\log(\deg(\pi)))$, using a classical fast sorting algorithm.
- Each call to E has cost $\mathcal{O}(1)$.
- Each call to E^{-1} has cost $\mathcal{O}(n\log(\deg(\pi)))$, by dichotomic search.
- F can be built with cost $\mathcal{O}(n\deg(\pi)^2\log(\deg(\pi)))$, performing one call to E^{-1} for each possible product of the monomials of the support.
- Each call to F costs $\mathcal{O}(1)$.

Finally this part of initialization of S/π requires

$$\mathcal{O}\Big(n\deg(\pi)^2\log(\deg(\pi))\Big)$$

operations. Let us now specify the costs of the arithmetic operations in S/π:

- The construction of an element given by a list of l terms requires l calls to E^{-1}.

- The function coeff just performs one call to E^{-1}.
- Binary additions (resp. subtractions) are done in $\deg(\pi)$ binary additions (resp. subtractions) in R.
- A binary multiplication requires at most $\deg(\pi)^2$ binary additions and multiplications in R and also $\deg(\pi)^2$ calls to the multiplication table F.

If $\psi \in S/\pi$ is a unit, we compute its inverse thanks to the classical iterator $N(z) = z + z(1 - z\psi)$, starting from the inverse of the constant coefficient of ψ. The number of iterations is in $\mathcal{O}(\log(\deg(\pi)))$.

In order to speed up the derivations we compute look-up tables $D_1, \ldots, D_n$. For $i = 1, \ldots, n$:

$$\begin{array}{rrcl} D_i: & N & \to & N \times \mathbb{N} \\ & l & \mapsto & \left(E^{-1}(\frac{\partial x^\alpha}{\partial x_i}), \alpha_i\right), \text{ where } x^\alpha = E(l). \end{array}$$

The construction of each D_i requires $\mathcal{O}(n\deg(\pi)\log(\deg(\pi)))$. The cost of the construction of all the D_i dominates the cost of the initialization of S/π.

Let $\psi \in S/\pi$. In order to compute $\frac{\partial \psi}{\partial x_i}$ we differentiate ψ term by term: for each $l \in N$ we look up $(m, \lambda) = D_i(l)$ and set the mth entry of $\frac{\partial \psi}{\partial x_i}$ to λ times the lth entry of ψ. The total cost is in $\mathcal{O}(\deg(\pi))$.

We deduce that for any $x^\alpha \notin \pi$ the cost of $\frac{\partial^{|\alpha|}\psi}{\partial x^\alpha}$ is in $\mathcal{O}(\deg(\pi)^2)$, for we have $|\alpha| \leq \deg(\pi)$.

As for the last statement of the proposition it suffices to select in ψ the monomials matching x^α and construct the answer in S/π. This is done within $\mathcal{O}(n\deg(\pi)\log(\deg(\pi)))$.

□

For the sake of clarity, we denote by $\mathcal{C}_S(\pi)$ the following expression

$$\boldsymbol{\mathcal{C}_S(\pi) := \deg(\pi)^2 \log(\deg(\pi)).}$$

It denotes the maximum of the numbers of binary additions, subtractions, multiplications and inversion in R needed to compute a binary addition, subtraction, multiplication or inverse in S/π. Unfortunately, we do not know better complexity result in general. We only know an optimal algorithm (up to logarithmic factors and when R is a field of characteristic zero) when π is a power of $(x_1, \ldots, x_n)$ [LS01].

Concerning the complexity of linear algebra we refer to §II.3.5: Ω is a constant such that matrix determinant requires $\mathcal{O}(n^\Omega)$ arithmetic operations in the base ring. The reader can keep in mind that Ω is less than 4. In order to simplify the presentation we take $\Omega \geq 3$. We discuss the practical relevance of Ω in Chapter V.

III.4.2 Incremental Lifting Step

The method we propose to compute the nested coordinates follows an approximation and correction scheme. The function `IncrementalLift` presented below is the core of our algorithm: for a given i it lifts an approximate value of Y^i. The input and output precisions are governed by integers $\kappa \geq 1$ and $\lambda \geq 1$. We define the ideals ζ_1 and ζ_2:

$$\begin{aligned}\zeta_1 &:= (dx_{R_{i+1}}^{\lambda}, dx_{R_{i+1}+1}, \ldots, dx_n) + \mathfrak{m}^{\kappa},\\ \zeta_2 &:= (dx_{R_{i+1}}^{2\lambda}, dx_{R_{i+1}+1}, \ldots, dx_n) + \mathfrak{m}^{2\kappa}.\end{aligned}$$

We lighten the notations: $\nabla\pi_{l_1,l_2}$ stands for $\nabla_{\{R_{l_1},\ldots,R_{l_2+1}-1\}}\pi_{l_1,l_2}$ and we write $\nabla\zeta_i$ instead of $\nabla_{\{R_{i+1},\ldots,n\}}\zeta_i$, for $i = 1, 2$. Quantities appearing in the core of the function that are not input nor local variables refer to global variables. As discussed in §III.3.5 we assume that the coordinates are generic enough.

`IncrementalLift`

Input:

- $W_{i+1} := z_{R_{i+1}}, \ldots, z_n$ in a neighborhood of $x^*_{R_{i+1}}, \ldots, x^*_n$, let $dW_{i+1} := z_{R_{i+1}} + dx_{R_{i+1}}, \ldots, z_n + dx_n$.
- Two integers $\kappa \geq 1$ and $\lambda \geq 1$.
- $\mathcal{T}_1, \ldots, \mathcal{T}_i$, the first i elements of the generic trace.
- $Z_1 := Y^i_1(dW_{i+1}), \ldots, Z_i := Y^i_i(dW_{i+1})$ at precision $\pi_{1,i} + \nabla\zeta_1$.

Output:

- $Y^i(dW_{i+1})$ at precision $\pi_{1,i} + \nabla\zeta_2 + \mathfrak{m}^{\kappa}\zeta_1$.

This subroutine will be used in two situations: either we want to improve the precision λ with respect to the variables $dx_{R_{i+1}}$ and $\kappa = 1$ holds (§III.4.3) or we want to improve the precision κ with respect to $\mathfrak{m}$ and $i = \nu$ holds (§III.4.6). Our presentation combines both these situations.

Algorithm:

We improve the precision of the nested coordinates in sequence. At step l we know an improved value of Y^l. More precisely:

($\mathtt{L}_l$) At step l we know $Y^l(Z_{l+1}+dX_{l+1}, \ldots, Z_i+dX_i, dW_{i+1})$ at precision $\pi_{1,l}+\sigma_{l+1}$, where
$$\sigma_{l+1} := \nabla\pi_{l+1,i} + \pi_{l+1,i}\zeta_1 + \nabla\zeta_2 + \mathfrak{m}^{\kappa}\zeta_1.$$

The computation goes from ($\mathtt{L}_0$) to ($\mathtt{L}_i$). Along the presentation of the algorithm we introduce the quantities θ, θ^{Ω}, θ_{Ω} and θ^{∇}. They are detailed in Lemma 10 below.

The initialization is trivial since Y^0 is the empty sequence. At the end (when $l = i$) we get $Y^i(dW_{i+1})$ at precision $\pi_{1,i} + \sigma_{i+1} = \pi_{1,i} + \nabla\zeta_2 + \mathfrak{m}^{\kappa}\zeta_1$.

Going from ($\mathtt{L}_l$) to ($\mathtt{L}_{l+1}$) is performed by the following computations:

1. First we let $dW_{l+2} := Z_{l+2} + dX_{l+2}, \dots, Z_i + dX_i, dW_{i+1}$ and we compute for each $j = 1, \dots, s$:

$$\gamma_j := f_j\Big(Y_1^l(Z_{l+1} + dX_{l+1}, dW_{l+2}) + dX_1, \dots,\\ Y_l^l(Z_{l+1} + dX_{l+1}, dW_{l+2}) + dX_l, Z_{l+1} + dX_{l+1}, dW_{l+2}\Big).$$

2. Since $\mathrm{coeff}_l(\gamma_j, dx^\alpha)$ is known at precision σ_{l+1}, then for each $\tau := (j, dx^\alpha, \mu) \in \mathcal{T}_{l+1}$ and according to formula (III.4), we deduce the value

$$\omega_\tau(Z_{l+1} + dX_{l+1}, dW_{l+2}) := \alpha_1! \cdots \alpha_l! \, \frac{\partial^\mu \mathrm{coeff}_l(\gamma_j, dx^\alpha)}{\partial {dx_{R_{l+1}}}^\mu} \tag{III.5}$$

at precision at least

$$\theta := \frac{\partial^{m_{l+1}-1} \sigma_{l+1}}{\partial {dx_{R_{l+1}}}^{m_{l+1}-1}}.$$

3. We deduce $\Omega_{l+1}(Z_{l+1}, dW_{l+2}) = (\omega_\tau(Z_{l+1}, dW_{l+2}),\ \tau \in \mathcal{T}_{l+1})$ at precision at least $\theta^\Omega := \theta \cap dA_{l+2}$ and valuation at least $\theta_\Omega := \pi_{l+2,i} + \nabla\zeta_1$ (by hypothesis).

4. We also deduce

$$\frac{\partial \omega_\tau}{\partial dX_{l+1}}(Z_{l+1}, dW_{l+2}) := \left(\frac{\partial \omega_\tau}{\partial dx_j}(Z_{l+1}, dW_{l+2}),\ j = R_{l+1}, \dots, R_{l+2} - 1\right)$$

at precision θ^∇.

5. We are looking for a value dZ_{l+1} of dX_{l+1} such that $Z_{l+1} + dZ_{l+1}$ is $Y_{l+1}(dW_{l+2})$ at precision σ_{l+2}. This value satisfies the following equation obtained from the first order Taylor expansion of Ω_{l+1} at (Z_{l+1}, dW_{l+2}):

$$\Omega_{l+1}(Z_{l+1} + dZ_{l+1}, dW_{l+2}) \in \sigma_{l+2}$$

$$\Longleftrightarrow$$

$$\Omega_{l+1}(Z_{l+1}, dW_{l+2}) + \frac{\partial \Omega_{l+1}}{\partial dX_{l+1}}(Z_{l+1}, dW_{l+2}) dZ_{l+1} \in \sigma_{l+2}.$$

We prove in Lemma 10 below that $\theta_\Omega^2 + \theta^\nabla \theta_\Omega + \theta^\Omega \subseteq \sigma_{l+2}$. Therefore dZ_{l+1} exists, is unique at precision σ_{l+2} and is given by:

$$dZ_{l+1} := -\left(\frac{\partial \Omega_{l+1}}{\partial dX_{l+1}}(Z_{l+1}, dW_{l+2})\right)^{-1} \Omega_{l+1}(Z_{l+1}, dW_{l+2}).$$

6. From the value $Y_{l+1}(dW_{l+2}) = Z_{l+1} + dZ_{l+1}$ at precision σ_{l+2} and using Proposition 33, it remains to compute $Y^l(Y_{l+1}(dW_{l+2}) + dX_{l+1}, dW_{l+2})$ at precision

$\pi_{1,l+1} + \sigma_{l+2}$ in order to reach $(\mathtt{L}_{l+1})$. For this purpose we use the following first order Taylor expansion formula:

$$\begin{aligned} Y^l(Z_{l+1} + dX_{l+1} + dZ_{l+1}, dW_{l+2}) &= Y^l(Z_{l+1} + dX_{l+1}, dW_{l+2}) \\ &+ \frac{\partial Y^l(Z_{l+1} + dX_{l+1}, dW_{l+2})}{\partial dX_{l+1}} dZ_{l+1} + \mathcal{O}((dZ_{l+1})^2) \end{aligned} \tag{III.6}$$

We know dZ_{l+1} at precision σ_{l+2} and each of its entries is in θ_Ω. The derivative $\frac{\partial Y^l(Z_{l+1}+dX_{l+1},dW_{l+2})}{\partial dx_j}$ for $dx_j \in dX_{l+1}$ is known at precision $\theta' := \frac{\partial(\pi_{1,l}+\sigma_{l+1})}{\partial dx_j}$. From Corollary 5 and Proposition 27 we deduce that

$$\begin{aligned} \theta' &\subseteq \pi_{1,l} + \tfrac{\partial \nabla \pi_{l+1,i}}{\partial dx_j} + \zeta_1 \\ &\subseteq \pi_{1,l} + \nabla_{\{R_{l+1},\ldots,R_{i+1}-1\}} \tfrac{\partial \pi_{l+1,i}}{\partial dx_j} + \zeta_1 \\ &\subseteq \pi_{1,l} + \nabla_{\{j\}} \tfrac{\partial \pi_{l+1,i}}{\partial dx_j} + \zeta_1 \\ &\subseteq \pi_{1,l} + \pi_{l+1,i} + \zeta_1. \end{aligned}$$

Therefore the precision of Y^l we reached is in

$$\pi_{1,l+1} + (\pi_{l+2,i} + \zeta_1)(\pi_{l+2,i} + \nabla\zeta_1),$$

which is itself included in $\pi_{1,l+1} + \sigma_{l+2}$, using again Lemma 10 below.

Lemma 10 *The following inclusions hold $\theta^\Omega \subseteq \sigma_{l+2}$, $\theta^\nabla \theta_\Omega \subseteq \sigma_{l+2}$ and $\theta_\Omega^2 \subseteq \sigma_{l+2}$.*

Proof. First we prove that $\theta^\Omega \subseteq \sigma_{l+2}$. Combining Corollary 5 and Lemma 6 we get that:

$$\frac{\partial^{m_{l+1}-1} \nabla \pi_{l+1,i}}{\partial d{x_{R_{l+1}}}^{m_{l+1}-1}} = \nabla_{\{R_{l+1},\ldots,R_{i+1}-1\}}(dX_{l+1} + \pi_{l+2,i}).$$

We deduce that

$$\begin{aligned} \theta \subseteq \frac{\partial^{m_{l+1}-1}\sigma_{l+1}}{\partial d{x_{R_{l+1}}}^{m_{l+1}-1}} &= \nabla_{\{R_{l+1},\ldots,R_{i+1}-1\}}(dX_{l+1} + \pi_{l+2,i}) \\ &\quad + (dX_{l+1} + \pi_{l+2,i})\zeta_1 + \nabla\zeta_2 + \mathfrak{m}^\kappa \zeta_1. \end{aligned} \tag{III.7}$$

As for the projection $\theta^\Omega = \theta \cap dS_{l+2}$ we use Corollary 6:

$$\nabla_{\{R_{l+1},\ldots,R_{i+1}-1\}}(dX_{l+1} + \pi_{l+2,i}) \cap dS_{l+2} = \nabla \pi_{l+2,i}.$$

This leads to

$$\theta^\Omega = \nabla\pi_{l+2,i} + \pi_{l+2,i}\zeta_1 + \nabla\zeta_2 + \mathfrak{m}^\kappa\zeta_1 = \sigma_{l+2}.$$

From (III.7) we deduce roughly that

$$\theta^\nabla \subseteq \pi_{l+2,i} + \zeta_1.$$

Finally

$$\theta^\nabla \theta_\Omega \subseteq (\pi_{l+2,i} + \zeta_1)(\pi_{l+2,i} + \nabla\zeta_1) = \pi_{l+2,i}^2 + \pi_{l+2,i}\zeta_1 + \zeta_1\nabla\zeta_1,$$

and using Propositions 27 and 28 yields

$$\theta^{\nabla}\theta_{\Omega} + \mathfrak{m}^{\kappa}\zeta_1 \subseteq \nabla\pi_{l+2,i} + \pi_{l+2,i}\zeta_1 + \nabla\zeta_2 = \sigma_{l+2}.$$

□

For complexity estimate we recall that the constant Ω is less than 4 and at least 3 (see definition in §III.4.1). We observe that all the computations can be done modulo the ideal $\nabla_{\{1,\ldots,n\}}(\pi_{1,i} + \zeta_2) + \mathfrak{m}^{2\kappa}$:

Lemma 11 *For any $i = 0, \ldots, \nu$ and any $l = 1, \ldots, i$, the following inclusion holds:*

$$\nabla_{\{1,\ldots,n\}}(\pi_{1,i} + \zeta_2) \subseteq \pi_{1,l} + \sigma_{l+1}.$$

Proof. It suffices to prove that

$$\nabla_{\{1,\ldots,n\}}(\pi_{1,i} + \zeta_2) \subseteq \pi_{1,l} + \nabla\pi_{l+1,i} + \pi_{l+1,i}\zeta_1 + \nabla\zeta_2.$$

We prove the reverse inclusion for the support of the ideals. We write $dx^{\alpha}dx^{\beta}dx^{\gamma}$ a monomial of the support of $\pi_{1,l} + \nabla\pi_{l+1,i} + \pi_{l+1,i}\zeta_1 + \nabla\zeta_2$, where dx^{α} is in the variables $dX_1, \ldots, dX_l$, dx^{β} in the variables $dX_{l+1}, \ldots, dX_i$ and dx^{γ} in the variables $dx_{R_{i+1}}, \ldots, dx_n$. Necessarily we have: $dx^{\alpha} \in \operatorname{supp}(\pi_{1,l})$, $dx^{\beta} \in \operatorname{supp}(\nabla\pi_{l+1,i})$, $dx^{\gamma} \in \operatorname{supp}(\nabla\zeta_2)$. We examine the two possible cases. In the first case $dx^{\beta} \in \operatorname{supp}(\pi_{l+1,i})$, we conclude thanks to Corollary 6. The second situation is $dx^{\beta} \in \pi_{l+1,i}$: necessarily we have $dx^{\gamma} \in \operatorname{supp}(\zeta_1) \subseteq \operatorname{supp}(\zeta_2)$, hence $dx^{\alpha}dx^{\beta}dx^{\gamma} \in \operatorname{supp}(\nabla_{\{1,\ldots,n\}}(\pi_{1,i} + \zeta_2))$. □

Proposition 36 *Let c denote $nm_1 \cdots m_i\lambda$. In terms of arithmetic operations in $A/(\mathfrak{m}^{2\kappa}A)$, the complexity of the function* `IncrementalLift` *is in*

$$\mathcal{O}\Big((nL + n^{\Omega})c^2\log(c)\Big).$$

Proof. Let $S := A/(\mathfrak{m}A)^{2\kappa}[[x_1 - x_1^*, \ldots, x_n - x_n^*]]$ and $\pi = \nabla_{\{1,\ldots,n\}}(\pi_{1,i} + \zeta_2)$. Using Lemma 11, we perform the computations is S/π. Let $\mathcal{C} := \mathcal{C}_S(\pi)$. From Proposition 26 we know that $\deg(\pi) \in \mathcal{O}(c)$. Using Proposition 35 we deduce that $\mathcal{C} \in \mathcal{O}(c^2\log(c))$ and that the initialization of S/π is done within $\mathcal{O}(n^2\mathcal{C})$.
We analyze the complexity for going from $(\mathtt{L}_l)$ to $(\mathtt{L}_{l+1})$ in terms of the number of operations in $A/(\mathfrak{m}^{2\kappa}A)$.

Step 1 costs $\mathcal{O}(L\mathcal{C})$. Since $n \leq c$, coeff_l requires $\mathcal{O}(\mathcal{C})$. Therefore Step 2 costs $\mathcal{O}(r_{l+1}\mathcal{C})$. Step 3 can be done within $\mathcal{O}(r_{l+1}\mathcal{C})$. Step 4 performs r_{l+1}^2 calls to coeff, this needs $\mathcal{O}(r_{l+1}^2\mathcal{C})$. In Step 5 one has to inverse an $r_{l+1} \times r_{l+1}$ invertible matrix over a ring it can be done in $\mathcal{O}(r_{l+1}^{\Omega}\mathcal{C})$. Last, Step 6 performs at most nr_{l+1} differentiations and n matrix vector products in dimension r_{l+1}. This yields $\mathcal{O}(nr_{l+1}^2\mathcal{C})$. Summing all these costs yields $\mathcal{O}((L + r_{l+1}^{\Omega} + nr_{l+1}^2)\mathcal{C})$. Now summing for l from 0 to i and the assumption $\Omega \geq 3$ yield the claimed bound $\mathcal{O}((nL + n^{\Omega})\mathcal{C})$. □

III.4.3 Computation of the Nested Coordinates

Now we explain how to compute the nested coordinates over the residue field $\mathfrak{o}/\mathfrak{m}$ together with a generic trace. We call this part of the algorithm `NestedCoordinates`. This function is always called before entering a lifting process. Since this function is used in §IV.2 we specify its input and output completely.

`NestedCoordinates`

Input:

- $f_1, \dots, f_s$ polynomials in $\mathfrak{o}[x_1, \dots, x_n]$.
- A monic polynomial q in $\mathfrak{o}/\mathfrak{m}[T]$. We let $A := \hat{\mathfrak{o}}[T]/(q)$.
- An approximation x^* in $(A/\mathfrak{m}A)^n$ of an isolated root with multiplicity M of $f_1 = \cdots = f_s = 0$.

Output:

- If the coordinates are generic enough and if $\mathfrak{m}$ is lucky then the function returns the nested coordinates $Y^\nu()$ in $\mathfrak{o}/\mathfrak{m}[[dx_1, \dots, dx_n]]$ at precision $\pi_{1,\nu}$. The function also returns a generic trace for the deflation.

Algorithm:

In order to shorten the notations we let

$$dW_{i+1} := x^*_{R_{i+1}} + dx_{R_{i+1}}, \dots, x^*_n + dx_n,$$

$$\zeta_\lambda := (dx^\lambda_{R_{i+1}}, dx_{R_{i+1}+1}, \dots, dx_n) \text{ and } \nabla\zeta_\lambda := \nabla_{\{R_{i+1},\dots,n\}}\zeta_\lambda.$$

The computation of the nested coordinates is incremental: we get the ith nested coordinates $Y^i(dW_{i+1})$ in sequence for $i = 0, \dots, \nu$. The initialization (for $i = 0$) is trivial since Y^0 is the empty sequence.

We enter the ith step of the computation with

($\mathtt{C}_i$) We know

- $r_1, \dots, r_i$;
- $m_1, \dots, m_i$;
- $Y^i(dW_{i+1})$ at precision $\pi_{1,i} + \nabla\zeta_1$.

We explain how to go from ($\mathtt{C}_i$) to ($\mathtt{C}_{i+1}$). All the calls to `IncrementalLift` are done at precision $\kappa = 1$.

In a first stage we search the value of m_{i+1}. These computations are organized around one main loop. We initialize $\lambda = 1$ and enter the following loop:

1. repeat:
   - Compute $\Phi_{i+1}(dW_{i+1})$ at precision $\nabla\zeta_\lambda$ using the formula of Proposition 34.

- Find the smallest $l \leq \lambda$ such that the monomial $dx_{R_{i+1}}^l$ is in the support of the elements of $\Phi_{i+1}(dW_{i+1})$. If such an l exists then it equals m_{i+1}. In this case break the loop.
- Call `IncrementalLift` in order to reach the precision $\nabla\zeta_{2\lambda}$ and the replace λ by 2λ.

The value of m_{i+1} is known. It remains to compute r_{i+1} and $\mathcal{T}_{i+1}$.

2. Compute
$$\widetilde{\Phi}_{i+1}(dW_{i+1}) = \frac{\partial^{m_{i+1}-1}}{\partial {dx_{R_{i+1}}}^{m_{i+1}-1}}\Phi_{i+1}(dW_{i+1})$$
at precision $\nabla\zeta_1$.

3. Build the $(sm_1 \cdots m_{i+1}) \times (n - R_{i+1} + 1)$ matrix J:
$$J := \frac{\partial\widetilde{\Phi}_{i+1}(dW_{i+1})}{\partial\{dx_{R_{i+1}}, \ldots, dx_n\}}(0, \ldots, 0).$$
This is the row matrix of the gradients of the elements of $\widetilde{\Phi}_{i+1}$ at x^*.

4. Let
$$dW_{i+2} := x^*_{R_{i+2}} + dx_{R_{i+2}}, \ldots, x^*_n + dx_n.$$
Compute the rank r_{i+1} of the matrix J and the expression of $Y_{i+1}(dW_{i+2})$ at precision $\nabla_{\{R_{i+2},\ldots,n\}}(dx_{R_{i+2}}, \ldots, dx_n)$. Deduce the trace $\mathcal{T}_{i+1}$. We let
$$dZ_{i+1} := Y_{i+1}(dW_{i+2}) - (x^*_{R_{i+1}}, \ldots, x^*_{R_{i+2}-1}).$$

5. We deduce the value
$$Y^{i+1}(dW_{i+2}) = (Y^i(Y_{i+1}(dW_{i+2}) + dX_{i+1}, dW_{i+2}), Y_{i+1}(dW_{i+2}))$$
at precision $\nabla_{\{R_{i+2},\ldots,n\}}(dx_{R_{i+2}}, \ldots, dx_n)$ using the following first order Taylor expansion:
$$Y^i(Y_{i+1}(dW_{i+2}) + dX_{i+1}, dW_{i+2}) = Y^i(dW_{i+1}) + \frac{\partial Y^i(dW_{i+1})}{\partial dX_{i+1}}dZ_{i+1} + \mathcal{O}((dZ_{i+1})^2).$$

The computation of the rank and the determination of $\mathcal{T}_{i+1}$ can be done using the classical Gaussian elimination process for instance. Indeed we can use any linear algebra algorithm. But it is important to notice that, according to this choice, the luckiness of $\mathfrak{m}$ differs. We can only say that:

Proposition 37 *There exists an element $a \neq 0$ in $\mathfrak{o}$ such that any $\mathfrak{m}$ that does not contain a is lucky for* `NestedCoordinates`.

Proof. Formally, one can think about the execution of `NestedCoordinates` over $k(u)$ instead of A. The function performs a finite number of equality tests and divisions. The maximal ideal $\mathfrak{m}$ is lucky if this mental computation can be specialized modulo $\mathfrak{m}$. Therefore the element a is a multiple of all the denominators of all the elements of k occurring in this computation. $\square$

Proposition 38 *If the coordinates are generic enough and if* $\mathfrak{m}$ *is lucky then the function* `NestedCoordinates` *requires*

$$\mathcal{O}\Big(n^3(nL+n^\Omega)M^2\log(nM)\Big)$$

arithmetic operations in $A/\mathfrak{m}A$.

Proof. Let $c_\lambda := nm_1\cdots m_i\lambda$ and $\mathcal{C}_\lambda = c_\lambda^2\log(c_\lambda)$. From Proposition 36 the call of `IncrementalLift` with input precision $\nabla\zeta_\lambda$ and output precision $\nabla\zeta_{2\lambda}$ costs $\mathcal{O}((nL+n^\Omega)\mathcal{C}_{2\lambda})$ operations in $A/\mathfrak{m}A$. The computation of $\Phi_{i+1}(dW_{i+1})$ requires $\mathcal{O}(L\mathcal{C}_{2\lambda}+sm_1\cdots m_inc_{2\lambda}\log(c_{2\lambda}))\subseteq\mathcal{O}((L+s)\mathcal{C}_{2\lambda})\subseteq\mathcal{O}(L\mathcal{C}_{2\lambda})$.
We deduce that the total cost of the loop is in $\mathcal{O}((nL+n^\Omega)\mathcal{C}_{m_{i+1}})$.

The cardinal of Φ_{i+1} equals $sm_1\cdots m_i$. The computation of the value $\widetilde{\Phi}_{i+1}(dW_{i+1})$ requires $\mathcal{O}(sm_1\cdots m_{i+1}c_{m_{i+1}})\subseteq\mathcal{O}(s\mathcal{C}_{m_{i+1}})$. The construction of J requires

$$\mathcal{O}(sm_1\cdots m_{i+1}nc_{m_{i+1}}\log(c_{m_{i+1}}))\subseteq\mathcal{O}(s\mathcal{C}_{m_{i+1}}).$$

The Gaussian elimination can be done within

$$\mathcal{O}(sm_1\cdots m_{i+1}n^2)\subseteq\mathcal{O}(s\mathcal{C}_{m_{i+1}}).$$

Last, updating the value of Y^i requires

$$\mathcal{O}(nr_{i+1}c_{m_{i+1}}+nr_{i+1}\mathcal{C}_{m_{i+1}})\subseteq\mathcal{O}(nr_{i+1}\mathcal{C}_{m_{i+1}}).$$

The total of these last operations is in $\mathcal{O}((s+nr_{i+1})\mathcal{C}_{m_{i+1}})$.
We conclude that going from Step $(\mathtt{C}_i)$ to $(\mathtt{C}_{i+1})$ requires $\mathcal{O}((nL+n^\Omega)\mathcal{C}_{m_{i+1}})$. It remains to sum these costs from $(\mathtt{C}_0)$ to $(\mathtt{C}_\nu)$ to get the claimed bound. $\square$

III.4.4 An *a posteriori* probabilistic luckiness test

We give a variant of the function `NestedCoordinates` taking the generic trace as argument and returning the nested coordinates. Computations are essentially the same as in `NestedCoordinates` but no rank determination is necessary.

`NestedCoordinatesWithTrace`

Input:

- $f_1, \ldots, f_s$ polynomials in $\mathfrak{o}[x_1, \ldots, x_n]$.
- a monic polynomial q in $\mathfrak{o}/\mathfrak{m}[T]$. We let $A := \hat{\mathfrak{o}}[T]/(q)$.
- A approximation x^* in $(A/\mathfrak{m}A)^n$ of an isolated root of $f_1 = \cdots = f_s = 0$.
- The generic trace $\mathcal{T}$.

Output:

- If the coordinates are generic enough and if $\mathfrak{m}$ satisfies the smoothness hypotheses of §III.3.3 then the function returns the nested coordinates $Y^\nu()$ in $\mathfrak{o}/\mathfrak{m}[[dx_1, \ldots, dx_n]]$ at precision $\pi_{1,\nu}$.

Proposition 39 *If the coordinates are generic enough and if* $\mathfrak{m}$ *is lucky then* `Nested-CoordinatesWithTrace` *requires*

$$\mathcal{O}\Big(n^3(nL + n^\Omega)M^2 \log(nM)\Big)$$

arithmetic operations in $A/\mathfrak{m}A$.

Once we know the generic trace it is sometimes useful to have a probabilistic test of the smoothness hypotheses for another maximal ideal $\mathfrak{m}$. From a practical point of view, `NestedCoordinatesWithTrace` raises a “division by zero” error message if the inversion of an element which is not zero is not possible. This message occurs when $\mathfrak{m}$ does not satisfy the smoothness hypotheses.

Proposition 40 *Let* $\mathcal{T}$ *be the generic trace of* x^* *if the coordinates are generic enough then if* `NestedCoordinates` *does not raise “division by zero” then the smoothness hypotheses* (H_Ω) *are satisfied.*

Proof. The only inversions occurring in `NestedCoordinates` are the ones of the Jacobian matrices of the Ω_i. These inversions involve determinants only. This corresponds to the smoothness hypotheses exactly. □

III.4.5 Example 3 continued from §III.3.2

We illustrate `NestedCoordinates` on Example 3 of §III.3.2. In order to make the reading of the computation easier we work over k instead of $\mathfrak{o}/\mathfrak{m}$, for some lucky maximal ideal $\mathfrak{m}$. Moreover we do not change the coordinates to generic ones since we know from III.3.2 that all the Weierstraß positions we need are satisfied.

We enter `NestedCoordinates` at step $(\mathtt{C}_0)$. It is easy to check that $m_1 = 1$, $r_1 = 1$ and compute

$$Y_1^1(dx_2, -1 + dx_3) = -dx_2 + dx_3 + \mathcal{O}((dx_1) + \nabla_{\{2,3\}}(dx_2, dx_3)).$$

We arrive at step ($\mathtt{C}_1$).

We compute $Y_1^1(dx_2,-1+dx_3) = Y_1(dx_2,-1+dx_3)$ as the solution of $f_1(Y_1^1(dx_2,-1+dx_3),dx_2,-1+dx_3)$ satisfying the condition $Y_1^1(dx_2,-1+dx_3)=0$ at precision $\mathcal{O}((dx_1)+\nabla_{\{2,3\}}(dx_2^3,dx_3))$. After 2 calls to `IncrementalLift` we obtain:

$$\begin{aligned} Y_1^1(dx_2,-1+dx_3) &= dx_3 + dx_2(-1+2\,dx_3) + dx_2^2(-2+8\,dx_3) - 4\,dx_2^3 \\ &\quad +\mathcal{O}((dx_1)+\nabla_{\{2,3\}}(dx_2^3,dx_3)). \end{aligned}$$

Substituting x_1 by $Y_1^1(dx_2,-1+dx_3)$ in F, we get $\gamma_1,\gamma_2,\gamma_3$:

$$\begin{aligned} \gamma_1 &= f_1(Y_1^1(dx_2,-1+dx_3)+dx_1,dx_2,-1+dx_3) \\ &= 0+\mathcal{O}((dx_1)+\nabla_{\{2,3\}}(dx_2^3,dx_3)), \\ \gamma_2 &= f_2(Y_1^1(dx_2,-1+dx_3)+dx_1,dx_2,-1+dx_3) \\ &= -3\,dx_2^2dx_3 + dx_2^3 + \mathcal{O}((dx_1)+\nabla_{\{2,3\}}(dx_2^3,dx_3)), \\ \gamma_2 &= f_3(Y_1^1(dx_2,-1+dx_3)+dx_1,dx_2,-1+dx_3) \\ &= 0+\mathcal{O}((dx_1)+\nabla_{\{2,3\}}(dx_2^3,dx_3)). \end{aligned}$$

Then we deduce $\Phi_2 = \{\phi_1,\phi_2,\phi_3\}$ where

$$\begin{aligned} \phi_1(dx_2,-1+dx_3) &= \mathrm{coeff}_1(\gamma_1,dx_1^0) \\ &= 0+\mathcal{O}(\nabla_{\{2,3\}}(dx_2^3,dx_3)), \\ \phi_2(dx_2,-1+dx_3) &= \mathrm{coeff}_1(\gamma_2,dx_1^0) \\ &= -3\,dx_2^2dx_3 + dx_2^3 + \mathcal{O}(\nabla_{\{2,3\}}(dx_2^3,dx_3)), \\ \phi_3(dx_2,-1+dx_3) &= \mathrm{coeff}_1(\gamma_3,dx_1^0) \\ &= 0+\mathcal{O}(\nabla_{\{2,3\}}(dx_2^3,dx_3)). \end{aligned}$$

We find that $m_2 = 3$ and we have to study the rank r_2 of the gradients of the elements of $\widetilde{\Phi}_2$:

$$\widetilde{\Phi}_2(dx_2,-1+dx_3) = \left\{\phi_1,\phi_2,\phi_3,\frac{\partial\phi_1}{\partial x_2},\frac{\partial\phi_2}{\partial x_2},\frac{\partial\phi_3}{\partial x_2},\frac{\partial^2\phi_1}{\partial {x_2}^2},\frac{\partial^2\phi_2}{\partial {x_2}^2},\frac{\partial^2\phi_3}{\partial {x_2}^2}\right\}.$$

All the elements of $\widetilde{\Phi}_2$ equal $0+\mathcal{O}(\nabla_{\{2,3\}}(dx_2,dx_3))$ except $\frac{\partial^2\Phi_2}{\partial {dx_2}^2}$: we deduce that $r_2=1$ and examine

$$\frac{\partial^2\phi_2}{\partial {dx_2}^2}(dx_2,-1+dx_3) = -6\,dx_3 + 6\,dx_2 + \mathcal{O}(\nabla_{\{2,3\}}(dx_2,dx_3)).$$

This yields the first order approximation of $Y_2^2(-1+dx_3)$:

$$Y_2^2(-1+dx_3) = Y_2(-1+dx_3) = dx_3 + \mathcal{O}((dx_1,dx_2^3)+\nabla_{\{3\}}(dx_3)).$$

We can now update $Y_1^2(1+dx_3)$:

$$\begin{aligned}
Y_1^2(-1+dx_3) &= Y_1^1(Y_2^2(-1+dx_3)+dx_2,-1+dx_3)\\
&= Y_1^1(dx_2,-1+dx_3)+dx_3\frac{\partial Y_1^1(dx_2,1+dx_3)}{\partial dx_2}\\
&= Y_1^1(dx_2,-1+dx_3)+dx_3\Big(-1-4\,dx_2-12\,dx_2^2\\
&\quad +\mathcal{O}((dx_1,dx_2^3)+(dx_3))\Big)\\
&= dx_2(-1-2\,dx_3)+dx_2^2(-2-4\,dx_3)\\
&\quad +\mathcal{O}((dx_1,dx_2^3)+\nabla_{\{3\}}(dx_3)).
\end{aligned}$$

Step ($\mathtt{C}_2$) of `NestedCoordinates` is reached.

We check that the precision is not enough and enter `IncrementalLift` again. Let z_1 and z_2 denote the former approximations of $Y_1^2(-1+dx_3)$ and $Y_2^2(-1+dx_3)$ respectively. The evaluation of f_1 yields:

$$\begin{aligned}
&f_1(z_1+dx_1,z_2+dx_2,-1+dx_3)=3\,dx_3^2+8\,dx_2^2dx_3^2+8\,dx_2^3\\
&\quad +dx_1(2-4\,dx_2-8\,dx_2^2)\\
&\quad +\mathcal{O}(\nabla_{\{1,2\}}(dx_1,dx_2^3)+(dx_1,dx_2^3)(dx_3)+\nabla_{\{3\}}(dx_3^2)).
\end{aligned}$$

We deduce

$$\begin{aligned}
dz_1 &= -\frac{3\,dx_3^2+8\,dx_2^2dx_3^2+8\,dx_2^3}{2-4\,dx_2-8\,dx_2^2}+\mathcal{O}(\nabla_{\{2\}}(dx_2^3)+(dx_2^3)(dx_3)+\nabla_{\{3\}}(dx_3^2))\\
&= -\frac{3}{2}dx_3^2-3\,dx_2dx_3^2-16\,dx_2^2dx_3^2-4\,dx_2^3\\
&\quad +\mathcal{O}(\nabla_{\{2\}}(dx_2^3)+(dx_2^3)(dx_3)+\nabla_{\{3\}}(dx_3^2)).
\end{aligned}$$

Therefore we arrive at Step ($\mathtt{L}_1$) with

$$\begin{aligned}
&Y_1^1(z_2+dx_2,-1+dx_3)=z_1+dz_1\\
&\quad =-\frac{3}{2}dx_3^2+dx_2(-1-2\,dx_3-3\,dx_3^2)+dx_2^2(-2-4\,dx_3-16\,dx_3^2)\\
&\quad -4\,dx_2^3+\mathcal{O}((dx_1)+\nabla_{\{2\}}(dx_2^3)+(dx_2^3)(dx_3)+\nabla_{\{3\}}(dx_3^2)).
\end{aligned}$$

We deduce the new value of Ω_2:

$$\begin{aligned}
&\Omega_2(z_2+dx_2,-1+dx_3)=\\
&\quad\Big\{\tfrac{\partial^2}{\partial {dx_2}^2}\mathrm{coeff}_1(f_2(Y_1^1(z_2+dx_2,-1+dx_3)+dx_1,z_2+dx_2,-1+dx_3),1)\Big\}\\
&\qquad =\Big\{9\,dx_3^2+6\,dx_2+\mathcal{O}(\nabla_{\{2\}}(dx_2)+(dx_2)(dx_3)+\nabla_{\{3\}}(dx_3^2))\Big\}.
\end{aligned}$$

We get $dz_2=-\frac{3}{2}dx_3^2+\mathcal{O}(\nabla_3(dx_3^2))$ and improve the precision of $Y_2^2(-1+dx_3)$:

$$\begin{aligned}
Y_2^2(-1+dx_3) &= z_2+dz_2\\
&= dx_3-\frac{3}{2}dx_3^2+\mathcal{O}((dx_1,dx_2^3)+\nabla_3(dx_3^2)).
\end{aligned}$$

The new value of Y_1^2 becomes:

$$\begin{aligned}
Y_1^2(-1+dx_3) &= Y_1^1(z_2+dz_2+dx_2,-1+dx_3)\\
&= Y_1^1(z_2+dx_2,-1+dx_3)+dz_2\frac{\partial Y_1^1(z_2+dx_2,1+dx_3)}{\partial dx_2}\\
&= Y_1^1(z_2+dx_2,-1+dx_3)-\frac{3}{2}dx_3^2\Big(-1-4\,dx_2\\
&\quad -12\,dx_2^2+\mathcal{O}((dx_1)+\nabla_{\{2\}}(dx_2^2)+(dx_3))\Big)\\
&= Y_1^1(z_2+dx_2,-1+dx_3)+\frac{3}{2}dx_3^2+6\,dx_2dx_3^2\\
&\quad +18\,dx_2^2dx_3^2+\mathcal{O}((dx_1,dx_2^3)+\nabla_{\{3\}}(dx_3^2))\\
&= dx_2(-1-2\,dx_3+3\,dx_3^2)+dx_2^2(-2-4\,dx_3+2\,dx_3^2)\\
&\quad +\mathcal{O}((dx_1,dx_2^3)+\nabla_{\{3\}}(dx_3^2)).
\end{aligned}$$

Step ($\mathtt{L}_2$) is reached but the precision is not enough to find m_3 and r_3. We enter `IncrementalLift` again. Let z_1 and z_2 denote the last approximations of $Y_1^2(-1+dx_3)$ and $Y_2^2(-1+dx_3)$ respectively. The evaluation of f_1 yields:

$$\begin{aligned}
&f_1(z_1+dx_1,z_2+dx_2,-1+dx_3)=\\
&\quad -6\,dx_3^3+\frac{9}{2}dx_3^4+dx_2^2(-24\,dx_3^3+18\,dx_3^4)+dx_2^3(8+32\,dx_3)\\
&\quad +dx_1(2+dx_2(-4-8\,dx_3)+dx_2^2(-8-16\,dx_3))\\
&\quad +\mathcal{O}(\nabla_{\{1,2\}}(dx_1,dx_2^3)+(dx_1,dx_2^3)(dx_3^2)+\nabla_{\{3\}}(dx_3^4)).
\end{aligned}$$

We find

$$\begin{aligned}
dz_1 &= -\frac{-6\,dx_3^3+\frac{9}{2}dx_3^4+dx_2^2(-24\,dx_3^3+18\,dx_3^4)+dx_2^3(8+32\,dx_3)}{2+dx_2(-4-8\,dx_3)+dx_2^2(-8-16\,dx_3)}\\
&\quad +\mathcal{O}(\nabla_{\{2\}}(dx_2^3)+(dx_2^3)(dx_3^2)+\nabla_{\{3\}}(dx_3^4))\\
&= 3\,dx_3^3-\frac{9}{4}dx_3^4+dx_2(6\,dx_3^3+\frac{15}{2}dx_3^4)+dx_2^2(36\,dx_3^3+45\,dx_3^4)\\
&\quad +dx_2^3(-4-16\,dx_3)+\mathcal{O}(\nabla_{\{2\}}(dx_2^3)+(dx_2^3)(dx_3^2)+\nabla_{\{3\}}(dx_3^4)).
\end{aligned}$$

Therefore we arrive at Step ($\mathtt{L}_1$) with

$$\begin{aligned}
&Y_1^1(z_2+dx_2,-1+dx_3)=z_1+dz_1=\\
&\quad 3\,dx_3^3-\frac{9}{4}dx_3^4+dx_2(-1-2\,dx_3+3\,dx_3^2+6\,dx_3^3+\frac{15}{2}dx_3^4)\\
&\quad +dx_2^2(-2-4\,dx_3+2\,dx_3^2+36\,dx_3^3+45\,dx_3^4)\\
&\quad +dx_2^3(-4-16\,dx_3)+\mathcal{O}((dx_1)+\nabla_{\{2\}}(dx_2^3)+(dx_2^3)(dx_3^2)+\nabla_{\{3\}}(dx_3^4)).
\end{aligned}$$

We deduce the new value of Ω_2:

$$\Omega_2(z_2+dx_2,-1+dx_3)=$$

$$\begin{aligned}&\Big\{\tfrac{\partial^2}{\partial {dx_2}^2}\mathrm{coeff}_1(f_2(Y_1^1(z_2+dx_2,-1+dx_3)+dx_1,z_2+dx_2,-1+dx_3),1)\Big\}\\ &\qquad=\Big\{-18\,dx_3^3-\tfrac{243}{2}dx_3^4+dx_2(6+36\,dx_3)\\ &\qquad+\mathcal{O}(\nabla_{\{2\}}(dx_2^3)+(dx_2^3)(dx_3^2)+\nabla_{\{3\}}(dx_3^4))\Big\}.\end{aligned}$$

We get $dz_2 = 3dx_3^3 + \frac{9}{4}dx_3^4 + \mathcal{O}(\nabla_3(dx_3^4))$ and improve the precision of $Y_2^2(-1+dx_3)$:

$$\begin{aligned}&Y_2^2(-1+dx_3) = z_2 + dz_2 =\\ &\qquad dx_3 - \frac{3}{2}dx_3^2 + 3\,dx_3^3 + \frac{9}{4}dx_3^4 + \mathcal{O}((dx_1, dx_2^3) + \nabla_3(dx_3^4)).\end{aligned}$$

The new value of Y_1^2 becomes:

$$\begin{aligned}Y_1^2(-1+dx_3) &= Y_1^1(z_2+dz_2+dx_2,-1+dx_3)\\ &= Y_1^1(z_2+dx_2,-1+dx_3)+dz_2\frac{\partial Y_1^1(z_2+dx_2,-1+dx_3)}{\partial dx_2}\\ &= -\frac{21}{2}dx_3^4+dx_2(-1-2\,dx_3+3\,dx_3^2-6\,dx_3^3-\frac{51}{2}dx_3^4)\\ &\quad +dx_2^2(-2-4\,dx_3+2\,dx_3^2-126\,dx_3^4)\\ &\quad +\mathcal{O}((dx_1,dx_2^3)+\nabla_{\{3\}}(dx_3^4)).\end{aligned}$$

Step $(\mathtt{L}_2)$ is reached. We deduce $\Phi_3 = \{\varphi_1,\dots,\varphi_9\}$, where $\varphi_l = 0 + \mathcal{O}(\nabla_{\{3\}}(dx_3^4))$ for $l = 1,\dots,8$ and $\varphi_9 = 9000\,dx_3^4 + \mathcal{O}(\nabla_{\{3\}}(dx_3^4))$. Therefore, $m_3 = 4$, $r_3 = 1$ and $\widetilde{\Phi}_3$. All the elements of $\widetilde{\Phi}_3$ equal $0 + \mathcal{O}(\nabla_{\{3\}}(dx_3))$ except $\frac{\partial^3\varphi_9}{\partial {dx_3}^3} = 216000\,dx_3 + \mathcal{O}(\nabla_{\{3\}}(dx_3))$.

III.4.6 Lifting the Nested Coordinates

Once we know the nested coordinates Y^ν over $A/(\mathfrak{m}^\kappa A)$ we can lift them in $A/(\mathfrak{m}^{2\kappa}A)$, for any $\kappa \geq 1$.

`LiftNestedCoordinates`

Input:

- $f_1,\dots,f_s$ polynomials in $\mathfrak{o}[x_1,\dots,x_n]$.
- a monic irreducible polynomial q in $\mathfrak{o}/\mathfrak{m}[T]$. We let $A := \hat{\mathfrak{o}}/\mathfrak{m}[T]/(q)$.
- Y^ν at precision $\mathfrak{m}^\kappa A + \pi_{1,\nu}$, the nested coordinates with respect to an isolated root x^* of $f_1 = \cdots = f_s = 0$.
- The generic trace $\mathcal{T}$ of the deflation computed by `NestedCoordinates`.

Output:

- Y^ν at precision $\mathfrak{m}^{2\kappa}A + \pi_{1,\nu}$.

Algorithm: Call straightforwardly the procedure `IncrementalLift`. Recall that the coordinates must be generic enough.

Proposition 41 *If the coordinates are generic enough and if* $\mathfrak{m}$ *is lucky then the function* `LiftNestedCoordinates` *requires*

$$\mathcal{O}(n^2(nL+n^\Omega)M^2\log(nM))$$

arithmetic operations in $A/(\mathfrak{m}^{2\kappa}A)$.

III.4.7 Example 3 continued from §III.4.5

In §III.4.5 we computed over $k=\mathbb{Q}$ the nested coordinates Y^3. As discussed in §III.3.3, $\mathfrak{m}=(7)$ is a lucky maximal ideal in $\mathfrak{o}=\mathbb{Z}$. We could have computed the nested coordinates in $\mathbb{Z}/p\mathbb{Z}$, they coincide with the value modulo 7 of the ones computed over $\mathbb{Q}$:

$$\begin{aligned}
Y_1^3 &= dx_2(6+5\,dx_3+3\,dx_3^2+dx_3^3+6\,dx_3^4)+dx_2^2(5+3\,dx_3+2\,dx_3^2)\\
&\quad+\mathcal{O}(dx_1,dx_2^3,dx_3^4),\\
Y_2^3 &= dx_3+2\,dx_3^2+3\,dx_3^3+4\,dx_3^4+\mathcal{O}(dx_1,dx_2^3,dx_3^4),\\
Y_3^3 &= 6+\mathcal{O}(dx_1,dx_2^3,dx_3^4).
\end{aligned}$$

Modulo 7 our approximated root z is $(0,0,6)$. We call the function `LiftNested-Coordinates` and get modulo 7^2 the new nested coordinates Y^3:

$$\begin{aligned}
Y_1^3 &= 21\,dx_3^4+dx_2(48+47\,dx_3+3\,dx_3^2+43\,dx_3^3+27\,dx_3^4)\\
&\quad+dx_2^2(47+45\,dx_3+2\,dx_3^2)+\mathcal{O}(dx_1,dx_2^3,dx_3^4),\\
Y_2^3 &= dx_3+23\,dx_3^2+3\,dx_3^3+4\,dx_3^4+\mathcal{O}(dx_1,dx_2^3,dx_3^4),\\
Y_3^3 &= 48+\mathcal{O}(dx_1,dx_2^3,dx_3^4).
\end{aligned}$$

The new approximation of $x^*=(0,0,-1)$ we get modulo 49 is $(0,0,48)$, which is correct.

III.4.8 Summary of the Algorithm

We gather the above algorithms in order to get a proof of Theorem 3. The existence of a in the theorem comes from Proposition 31 and Proposition 37. The restriction on the characteristic of k comes from the deflation lemma 8.

In §III.3.5 it is shown that the change of coordinates that are not generic enough are contained in an algebraic hypersurface of $GL_n(k)$. From Proposition 38 the computation of the nested coordinates is done within

$$\mathcal{O}\Big(n^3(nL+n^\Omega)M^2\log(nM)\Big)$$

arithmetic operations in $A/\mathfrak{m}A$.

Once the nested coordinates are computed modulo $\mathfrak{m}$ we can lift them using the procedure `LiftNestedCoordinates`. Reaching precision $\mathfrak{m}^{2\kappa}$ from precision $\mathfrak{m}^\kappa$ costs

$$\mathcal{O}\Big(n^2(nL+n^\Omega)M^2\log(nM)\Big)$$

arithmetic operations in $A/(\mathfrak{m}^{2\kappa}A)$.

III.5 Splittings

From the beginning we have assumed that the root x^* is given by means of an irreducible minimal polynomial Q. The question arising naturally is to know what happens if Q is not irreducible any more: in this case x^* represents several irreducible sets of roots. It is a well-known fact that such a systematic extension of our method for a reducible square free polynomial Q is possible via *dynamic evaluation*. This general framework has been developed both from theoretical and practical point of views in the Axiom [Sut92] computer algebra system in a series of papers by Duval and her collaborators: Della Dora, Dellière, Dicrescenzo, Gómez Díaz, Reynaud [DDD85, Duv87, DD89, Duv89, Duv94, Día94, Duv94, DR94a, DR94b, BGW95, Duv95, Del99]. Following the dynamic evaluation scheme, we develop in this section a special function for handling a non irreducible situation and estimate its complexity.

From now on we assume that Q is not necessarily irreducible anymore: Q is square free in $k[T]$, we let $k[u] := k[T]/(Q(T))$ and x^* is given by a vector of n polynomials of $k[T]$. If $Q = Q_1 \cdots Q_r$ is the irreducible decomposition of Q in $k[T]$ then $k[u]$ is isomorphic to the Cartesian product of the field extensions $k[T]/(Q_1(T)) \times \cdots \times k[T]/(Q_r(T))$. We still assume that hypotheses (H_Q) and (H_{x^*}) hold. In consequence, (H_{Q_i}) holds and (H_{x^*}) holds in each $k[T]/(Q_i(T))$, for $i = 1, \dots, r$.

We denote by A_i the quotient $\hat{\mathfrak{o}}[T]/(Q_i(T))$ and still write x^* for the image of x^* in A_i. Note that A is isomorphic to the product $A_1 \times \cdots \times A_r$. We shall write $\mathcal{T}^i$ for the generic trace of the deflation process executed in A_i, as defined in §III.3.5. Having the same trace induces a partition $p_1, \dots, p_t$ of the set $\{1, \dots, r\}$. Let Q_{p_i} be the product of the Q_j for all $j \in p_i$. The aim of the following algorithm is to compute the Q_{p_i} and their associated generic traces $\mathcal{T}^{p_i}$. We write M_i for the multiplicity of x^* as a root of the system $f_1 = \cdots = f_s = 0$ in $k[T]/(Q_i(T))$, for $i = 1, \dots, r$. We say that x^* is **DA-irreducible** if $t = 1$.

If we execute the function `NestedCoordinates` in A directly to compute the nested coordinates modulo $\mathfrak{m}$ from the knowledge of x^* modulo $\mathfrak{m}$ we probably run into an error due to the fact that $A/\mathfrak{m}A$ is not a field. Such an error occurs when the computation requires a division by an element that is not zero and not invertible. In order to build up the splitting method we assume that the function `NestedCoordinates` raises the message "division by zero" and returns the element causing the trouble.

We introduce the function `DaSplit` which takes as input the polynomial Q known modulo $\mathfrak{m}$ and the value of x^* modulo $\mathfrak{m}$ in the algebra A. It returns the sequences $(Q_{p_i})_{i=1,\dots,t}$ and $(\mathcal{T}^{p_i})_{i=1,\dots,t}$.

`DaSplit`

Input:

- a square free monic polynomial Q in $\mathfrak{o}[T]$. We let $A = \hat{\mathfrak{o}}/\mathfrak{m}[T]/(Q)$.
- a vector $(V_1, \dots, V_n)$ of polynomials in $\mathfrak{o}[T]$ of degree less than $\deg(Q)$ such that $x^* = (V_1, \dots, V_n)$ in $A/\mathfrak{m}A$.

- a sequence $f_1, \ldots, f_s$ of polynomials such that x^* represents an isolated set of roots of $f_1 = \cdots = f_s = 0$.

Output:

- If the coordinates are generic enough and if $\mathfrak{m}$ is lucky then it returns the sequences $Q_{p_i},\ i = 1, \ldots, t$ and $\mathcal{T}^{p_i}, i = 1, \ldots, t$.

Algorithm:

1. Call `NestedCoordinates` with Q. If the error "division by zero" is raised then the function returns an element P of $\hat{\mathfrak{o}}/\mathfrak{m}[T]$ that is not divisible by Q. Let Q_e be the greatest common divisor of P and Q and $Q_n = Q/Q_e$. We have $\deg(Q_e) \geq 1$ and $\deg(Q_n) \geq 1$, this induces a splitting. If no error occurs then return Q and $\mathcal{T}$ found by `NestedCoordinates`.

2. Call recursively the function with Q_e, then with Q_n, merge the returned sequences and return.

As for complexity estimate we need to count the costs of the operations in A. We introduce in §II.3.5 the function $\mathcal{U}$ such that all the elementary arithmetic operations can be performed in A within $\mathcal{O}(\mathcal{U}(\deg(Q)))$ operations in $\hat{\mathfrak{o}}/\mathfrak{m}$.

Proposition 42 *In case of success the complexity of* `DaSplit` *is in*

$$\mathcal{O}\Big(\log(d) n^4 (nL + n^\Omega) D^2 \mathcal{U}(\deg(Q)) \Big),$$

in terms of arithmetic operations in the residue field $\mathfrak{o}/\mathfrak{m}$, *where*

$$D := \sum_{i=1}^{r} M_i \deg(Q_i).$$

Proof. There are $2t - 1$ calls to `NestedCoordinates`: t of them leading to the Q_{p_j} and $t-1$ raising the "division by zero" error. We perform the complexity analysis in the worst case as if all the computations were performed in A. First we examine the cost of the failing calls. It suffices to observe that the failing computations can be embedded in the successful ones: to each failing call we associate a successful one performing the same operations in A until the error. Therefore, in terms of arithmetic operations in A, the sum of the costs of the calls to `NestedCoordinates` raising the error is bounded by the sum of the costs of the finishing ones.

Let $M_{p_j} = \min_{i \in p_j} M_j$, for $j = 1, \ldots, t$. In terms of arithmetic operations in $\mathfrak{o}/\mathfrak{m}$, the sum of the costs of the `NestedCoordinates` finishing without any error is in

$$\mathcal{O}\Big(\sum_{j=1}^{t} n^3 (nL + n^\Omega) M_{p_j}^2 \log(n M_{p_j}) \mathcal{U}(\deg(Q)) \Big).$$

Noticing that $M_i \leq d^n$ for each $i = 1, \ldots, r$, this yields the bound $\log(nM_i) \in \mathcal{O}(\log(nd^n)) \subseteq \mathcal{O}(n\log(d))$. We conclude the proof this way:

$$\sum_{j=1}^{t} M_{p_j}^2 \leq \sum_{j=1}^{t} M_{p_j}^2 \deg(Q_{p_j})^2 \leq \left(\sum_{j=1}^{t} M_{p_j} \deg(Q_{p_j})\right)^2 \leq D^2.$$

□

Concerning the probabilistic aspects it is easy to deduce that

Proposition 43 *There exists an element $a \neq 0$ in $\mathfrak{o}$ such that for any $\mathfrak{m}$ which does not contain a* `DaSplit` *returns a correct answer if the linear change of coordinates is chosen outside an algebraic hypersurface of $GL_n(k)$.*

Chapitre IV

Equidimensional Decomposition

In this chapter we generalize the algorithm presented in Chapter II: we propose a new probabilistic method for solving any system of polynomial equations and inequations. The output is the equidimensional decomposition of the Zariski closure of the set of roots of the system. Equidimensional varieties are represented by means of lifting fibers.

We deeply use our generalization of Newton's iterator based on fast deflation as presented in the previous chapter. The solver is probabilistic, its success depends on a lucky linear change of coordinates in the input system. Its complexity is in the cube of a certain degree counting multiplicities and linear in the evaluation complexity of the input system.

If the base field is the field of the rational numbers then the computations are first performed modulo a random prime number and a random choice of coordinates. Then we search coordinates with small rational integers and lift the fibers up to the integers. We also describe how to write down the dependency in the free variables.

We illustrate our implementation available in our package Kronecker from version 0.16, written in the Magma computer algebra system.

IV.1 Introduction

Let k be a field of characteristic zero, $\overline{k}$ its algebraic closure and $f_1, \ldots, f_s, g$ polynomial functions in $k[x_1, \ldots, x_n]$ given by a straight-line program. We are interested in computing the equidimensional decomposition of the Zariski closure $\mathcal{V}$ of the set of roots of the system

$$f_1(x) = \cdots = f_s(x) = 0, \quad g(x) \neq 0.$$

The algorithm of this chapter extends the one of Chapter II straightforwardly. We let d be the maximum of the degrees of the f_i and L the evaluation complexity of $f_1, \ldots, f_s, g$. We introduce the ith *intermediate variety* $\mathcal{V}_i$ as the closure of the set of roots of $f_1 = \cdots = f_i = 0$, $g \neq 0$:

$$\mathcal{V}_i := \overline{\{z \in \overline{k}^n \mid f_1(z) = \cdots = f_i(z) = 0,\ g(z) \neq 0\}}, \quad i = 1, \ldots, s.$$

Our algorithm computes the equidimensional decompositions of the $\mathcal{V}_i$ in sequence for $i = 1, \ldots, s$. Each equidimensional component is represented by a set of lifting fibers without redundancy.

If $\mathcal{W}$ is an irreducible component of $\mathcal{V}_i$ we write $\mathrm{mul}_{f_1,\ldots,f_i}(\mathcal{W})$ for the *multiplicity* of $\mathcal{W}$ as solution set of the system $f_1 = \cdots = f_i = 0$. We introduce the *algebraic degree* $\deg^{\mathrm{a}}_{f_1,\ldots,f_i}(\mathcal{W})$ as the product $\mathrm{mul}_{f_1,\ldots,f_i}(\mathcal{W})\deg(\mathcal{W})$, where $\deg(\mathcal{W})$ is the classical geometric degree of $\mathcal{W}$. In extension we define the algebraic degree $\deg^{\mathrm{a}}_{f_1,\ldots,f_i}(\mathcal{V}_i)$ as the sum of the algebraic degrees of the irreducible components of $\mathcal{V}_i$. The new crucial quantity appearing in the complexity of our algorithm is the maximum δ^a of the algebraic degrees of the intermediates varieties, namely:

$$\delta^a := \max_{i=1,\ldots,s} \deg^{\mathrm{a}}_{f_1,\ldots,f_i}(\mathcal{V}_i).$$

We recall from §II.3.5 that we fix the function $\mathcal{U}(a) = a\log(a)^2\log\log(a)$ and Ω a constant less than 4. We are now able to state the main result of this chapter.

Theorem 4 *Let k be a field of characteristic zero and let $f_1, \ldots, f_n, g$ be polynomials in $k[x_1, \ldots, x_n]$ of degree at most d and given by a straight-line program of size at most L. There exists a probabilistic algorithm computing the equidimensional decomposition of the Zariski closure of the system*

$$f_1 = \cdots = f_s = 0, \quad g \neq 0$$

with a complexity in

$$\mathcal{O}\Big(s\log(d)n^4(nL + n^{\Omega})\mathcal{U}(d\delta^a)^3\Big),$$

in terms of arithmetic operations in k and in case of success. Equidimensional components are encoded by a set of lifting fibers. The probability of success of the algorithm depends on lucky choices of elements in k. Choices for which the result is not correct are enclosed in strict algebraic subsets.

The worst case situation for L is when the input polynomials are expanded. We deduce the following corollary.

Corollary 8 *Let k be a field of characteristic zero and $f_1, \ldots, f_n, g$ be polynomials in $k[x_1, \ldots, x_n]$ of degree at most d given as vectors of their coefficients. There exists a probabilistic algorithm computing the equidimensional decomposition of the Zariski closure of the system*

$$f_1 = \cdots = f_s = 0, \quad g \neq 0$$

with a complexity in

$$\mathcal{O}\Big(s^2 n^{11} d^{4n+\mathcal{O}(1)}\Big),$$

in terms of arithmetic operations in k and in case of success. Equidimensional components are encoded by a set of lifting fibers. The probability of success of the algorithm depends on lucky choices of elements in k. Choices for which the result is not correct are enclosed in strict algebraic subsets.

Proof. The evaluation complexity L of the system is in $\mathcal{O}(sd^{n+\mathcal{O}(1)})$ and we can bound δ^a by nd^n. □

IV.1.1 Related Results

For computing an equidimensional decomposition, the best known deterministic algorithms have a complexity upper bound asymptotically polynomial in sd^{n^2}: in [CG83, Chi96, Chi97] Chistov and Grigoriev present algorithms with such a complexity for computing a decomposition of $\mathcal{V}$ into irreducible components; Giusti and Heintz give in [GH91] another method with the same complexity, moreover their decomposition into equidimensional parts is well-parallelizable; Elkadi and Mourrain propose in [EM99a] a method based on Bézoutian matrices, their algorithm is probabilistic and has complexity polynomial in sd^{n^2}.

It is clear that sd^{n^2} is a lower bound for the problem if we represent multivariate polynomials in the output as vectors of their coefficients (see the discussion I.5 in the introduction of this thesis). We propose a first breakthrough in [Lec00]: encoding eliminant polynomials by means of straight-line programs and algebraic varieties by geometric resolutions yield a complexity mainly polynomial in d^n. The algorithm is probabilistic with a uniformly bounded error probability. Another independent approach is proposed by Jeronimo and Sabia in [JS00]: they exhibit a probabilistic algorithm for computing an equidimensional decomposition of an affine variety over a field of characteristic zero in time polynomial in sd^n, each output component described as the set of roots of $n+1$ polynomials encoded by straight-line programs. But their error probability is not uniformly bounded.

A byproduct of an equidimensional decomposition is the dimension but direct algorithms for computing the dimension with a complexity polynomial in sd^n exist: in [GH93] Giusti and Heintz give a well-parallelizable algorithm polynomial in sd^n, deterministic in a non-uniform complexity model and probabilistic in a uniform one; in 1996 Chistov [Chi96, Chi97] performs the same computation within the same complexity but deterministically for a uniform complexity model. A recent historical presentation of the work of Chistov, Grigoriev and Vorobjov can be found in [Vor99]. On the basis of Koiran's derandomization methods [Koi97] Rojas proposes in [Roj00] a deterministic algorithm for computing the dimension using toric resultant [Stu94, GKZ94, EC95a, Emi94, EP98, EM99b] with a complexity polynomial in a certain mixed volume (therefore polynomial in d^n).

From a numerical point of view, Sommese, Vershelde and Wampler propose in [SV00, SVW01a, SVW01b] numerical equidimensional and irreducible decomposition algorithms based on homotopy continuation [VC93, Ver96, Ver99]. Their output is a set of generic points on equidimensional components. Even if the complexity of their algorithm is not stated, the theoretical bottleneck is still in $\mathcal{O}(d^{n^2})$ since they interpolate multivariate eliminant polynomials in order to test inclusion of components.

Our method lies in the continuation of a series of papers initiated by Giusti, Hägele, Heintz, Montaña, Morais, Morgenstern and Pardo. The notions of geometric resolutions and intrinsic degree of a system have been introduced in [GHMP95, Par95] and the first resolution procedure for regular sequences (complete intersections) appears in 1997: the articles [GHH$^+$97, GHMP97, GHM$^+$98] present an algorithm for computing the roots of a system of polynomial equations with a complexity polynomial in an intrinsic degree of the system. In [Mor97, HMPS00] the method is generalized in order to compute the isolated roots of any system of polynomial equations and inequations (not complete intersection). In [GLS01] (Chapter II of this thesis) we simplify, redesign and improve the algorithm and explain how to implement it. Our aim in this chapter is now to extend this algorithm in a natural way in order to compute not only the isolated roots but a description of all the components.

IV.1.2 Contributions

We recall that the algorithm presented in Chapter II is designed to solve the very particular so called reduced regular situation: $s = n$, $\mathcal{V}_i$ is equidimensional of codimension i (*regularity hypothesis*) and the Jacobian matrix of $f_1, \ldots, f_i$ has full rank when evaluated at a generic point of $\mathcal{V}_i$ (*reduction hypothesis*). In this chapter we keep the same approach: incremental solving and encoding of equidimensional varieties by means of lifting fibers.

The new tools which conduct us to the generalization of the algorithm of Chapter II are the following. First the results of the previous chapter allow us to remove the reduction hypothesis. We describe in §IV.2 the new lifting process tackling multiple components. It is important to notice that we do not compute the multiplicities but only probabilistic lower bounds on them. Then the regularity hypothesis is easier to remove, this is the purpose of §IV.3: we give a minimization process which ensures that our representations of equidimensional decompositions are not redundant.

As in Chapter II, if k is the field of the rational numbers the resolution is first computed modulo a small prime number p. Then we lift the solutions in the field of p-adic integers and reconstruct the rational numbers. This is the aim of §IV.5.

The algorithm is implemented in our Magma package Kronecker since version 0.16. The implementation is described in the next chapter together with examples and timings. In IV.6 we illustrate its behaviour with a few examples.

At first reading of this chapter the reader may keep in mind that the coordinates of the input system are moved to generic affine ones. We prove that the probability of success of the resolution process relies on this choice of coordinates. In order to control the probabilistic aspects we keep the same point of view as in Chapter II: algorithms depend on parameters. Correctness relies on good choices of these parameters. The complexity we give corresponds to situations of success. In a bad situation it may appear that a computation does not stop.

IV.2 Global Newton with Multiplicity

We tackle the lifting problem for multiple components using our fast deflation algorithm of the previous chapter. This section generalizes §II.4. In Chapter II the geometric degree is the very important quantity occurring in complexity estimate. When dealing with multiple components we need to introduce a degree counting the multiplicities.

ALGORITHM IV.1: **Global Newton Iterator with Multiplicity**

`GlobalNewton`($\mathbf{e}, u, q, \mathbf{v}$,StopCriterion, $\mathcal{T}$)

- $\mathbf{e}$ is a set of polynomials.
- u is a linear form.
- q is a monic polynomial in $\mathfrak{o}[T]$.
- $\mathbf{v}$ is a sequence of polynomials in $\mathfrak{o}[T]$.
- StopCriterion is a function returning a boolean. Its arguments are taken from the local variables. It returns whether the lifted parametrization $(V_1, \ldots, V_n)$ at precision $\mathfrak{m}^\kappa$ is sufficient or not.
- $\mathcal{T}$ is a generic trace.

If (H1),...,(H7) are satisfied and if N is generic enough then the procedure returns Q and $(V_1, \ldots, V_n)$ as in (O1), (O2), where κ is implicitly fixed by StopCriterion.

\# Computations are performed in $A/(\mathfrak{m}^\kappa A)$.
    $N \leftarrow \mathsf{W}(n)$; # N is a random invertible matrix.
    # Change the coordinates to N
    $\mathbf{e} \leftarrow \mathbf{e} \circ N$; $\mathbf{v} \leftarrow N^{-1}\mathbf{v}$; $u \leftarrow u \circ N$;
    $\kappa \leftarrow 1$;
    $\mathbf{Y}, \mathcal{T} \leftarrow$ `NestedCoordinatesWithTrace`($\mathbf{e}, q, \mathbf{v}, \mathcal{T}$); # $\mathbf{Y} = (Y_1, \ldots, Y_n)$
    $\mathbf{V} \leftarrow \mathbf{v}$; $Q \leftarrow q$;
    while not StopCriterion($\mathbf{e}, u, Q, \mathbf{V}, \kappa, N$) **do**
        $\kappa \leftarrow 2\kappa$;
        $\mathbf{Y} \leftarrow$ `LiftNestedCoordinates`($\mathbf{e}, Q, \mathbf{Y}, \mathcal{T}$);
        # Y_i is a multivariate power series in n variables.
        $V_i \leftarrow$ constant coefficient of Y_i;
        $\Delta \leftarrow u(\mathbf{V}) - T$;
        $\mathbf{Y} \leftarrow \mathbf{Y} - \left(\frac{\partial Y}{\partial T}\Delta \bmod Q\right)$;
        $Q \leftarrow Q - \left(\frac{\partial Q}{\partial T}\Delta \bmod Q\right)$;
    od;
    $\mathbf{V} \leftarrow N\mathbf{V}$; # Change the coordinates back
    return($Q, \mathbf{V}$);

IV.2.1 Global Lifting

We revisit the lifting process presented in §II.4. In this section only, we use the notations of Chapter III. Let $\mathfrak{o}$ be a commutative Noetherian domain, $\mathfrak{m}$ one of its maximal ideals,

k its field of fractions and $\hat{\mathfrak{o}}$ its completion with respect to the $\mathfrak{m}$-acid topology. The lifting algorithm takes as input:

(I1) A system $\mathbf{e} = e_1, \ldots, e_r$ of polynomials in $k[x_1, \ldots, x_n]$;

(I2) A linear form $u = \lambda_1 x_1 + \cdots + \lambda_n x_n$, with $\lambda_i \in \mathbb{Z}$;

(I3) A monic squarefree polynomial q in $\mathfrak{o}[T]$;

(I4) $\mathbf{v} = (v_1, \ldots, v_n)$, n polynomials in $\mathfrak{o}[T]$ of degrees strictly less than $\deg(q)$;

(I5) $\mathcal{T}$, a *generic trace* as defined in §III.3.5.

Let $A := \hat{\mathfrak{o}}[T]/(q)$. We assume that:

(H1) There exist polynomials $\hat{Q}, \hat{V}_1, \ldots, \hat{V}_n$ in $k[T]$ such that $x^* = (\hat{V}_1, \ldots, \hat{V}_n)$ in $k[T]/\hat{Q}(T)$ represents a set of isolated roots of the system $\mathbf{e} = 0$ with respect the Zariski topology. Let $\hat{Q}_i$, for $i = 1, \ldots, r$, be the irreducible factors of $\hat{Q}$, then we denote by M_i the multiplicity of $(\hat{V}_1, \ldots, \hat{V}_n)$ as a root of $\mathbf{e} = 0$ in $k[T]/\hat{Q}_i(T)$. We say that an element in k is **well-defined** in $\hat{\mathfrak{o}}$ if its denominator is invertible modulo $\mathfrak{m}$. By extension we say that an element of $k[T]/\hat{Q}(T)$ is well-defined in A if all its coefficients are well-defined in $\hat{\mathfrak{o}}$.

(H2) x^* is *DA-irreducible* as defined in §III.5, M denotes the minimum of the multiplicities of the points represented by x^* and $\mathcal{T}$ is a generic trace for x^*.

(H3) $\hat{Q}$ is monic, well-defined in $\mathfrak{o}$ and coincides with q modulo $\mathfrak{m}A$.

(H4) x^* is well-defined in A^n and coincides with $(v_1, \ldots, v_n)$ modulo $\mathfrak{m}A$.

(H5) $u(\hat{V}_1, \ldots, \hat{V}_n) = T$.

(H6) $\mathbf{e}$ is well-defined over A.

(H7) The maximal ideal $\mathfrak{m}$ satisfies the smoothness hypotheses of §III.3.3 with respect to $\mathcal{T}$.

Then the lifting algorithm computes approximations of x^* in A at any arbitrary precision. More precisely, for any $\kappa > 0$ we can obtain:

(O1) Q a monic polynomial in $\mathfrak{o}[T]$ of degree $\deg(q)$, such that $Q = \hat{Q} \bmod \mathfrak{m}^\kappa \mathfrak{o}[T]$.

(O2) $\mathbf{V} = (V_1, \ldots, V_n)$, n polynomials in $\mathfrak{o}[T]$ of degrees strictly less than $\deg(q)$ such that $V_i = \hat{V}_i \bmod \mathfrak{m}^\kappa \mathfrak{o}[T]$ and $u(\mathbf{V}) = T \bmod \mathfrak{m}^\kappa \mathfrak{o}[T]$.

The lifting algorithm is the combination of the fast deflation algorithm of the previous chapter and the globalization trick of §II.4. It is summarized in Algorithm IV.1. We recall that the algorithm is probabilistic: in order to work properly the coordinates must be generic enough to satisfy the Weierstraß positions required in §III.3.1. This is why we introduce a new function $\mathsf{W}(n)$ returning a random $n \times n$ matrix in order to keep track of the choice of N, which is a $n \times n$ matrix with entries in $\mathfrak{o}$ and invertible modulo $\mathfrak{m}$.

In the sequel this function plays the role of a parameter of the algorithm. We change $\mathbf{e}$ to $\mathbf{e} \circ N$, $\mathbf{v}$ to $N^{-1}(\mathbf{v})$ and u to $u \circ N$ before entering the deflation routines. At the end of the lifting we convert back to the original coordinates. If N is not generic enough the function may either raise a "division by zero" error from `NestedCoordinatesWithTrace` or return a bad result. If $\mathtt{a}(h)$ denotes the cost of the arithmetic operations in $\mathfrak{o}/\mathfrak{m}^h$ then we have the following complexity estimate.

Proposition 44 *Under hypotheses* (H1),...,(H7) *there exists a Zariski open subset of* $GL_n(k)$ *such that for any matrix* N *in it which is well-defined in* A *and invertible modulo* $\mathfrak{m}$ *Algorithm IV.1 returns a correct answer, with a complexity in*

$$\mathcal{O}\Big(\log(d)n^4(nL+n^{\Omega})M^2\mathcal{U}(\deg(Q))\sum_{j=0}^{\log_2(\kappa)}\mathtt{a}(2^j)\Big).$$

Proof. The first claim reformulates the discussion of §III.3.5. The cost of the changes of variables is negligible. The complexity is essentially the sum of the complexities of the computation of the nested coordinates and the successive liftings. These complexities are the ones of Propositions 39 and 41. The deformation computations for updating $\mathbf{Y}$ and Q are done within $\mathcal{O}(n^2M)$ arithmetic operations in $A/(\mathfrak{m}^{\kappa}A)$, since the support of the multivariate power series for Q and $\mathbf{Y}$ is in $\mathcal{O}(nM)$. The value M is bounded by d^n. Therefore we bound $\log(nM)$ by $\log(n) + n\log(d) \in \mathcal{O}(n\log(d))$. $\square$

IV.2.2 Lifting fibers

We revert to the notations of the beginning of this chapter: k is any field of characteristic zero.

Let $\mathcal{W}$ be an r-equidimensional variety. We consider a geometric resolution of $\mathcal{W}$ composed of variables $y_1, \ldots, y_n$ in projective Noether position, a primitive element u, its minimal polynomial q and the parametrization of the coordinates $v_{r+1}, \ldots, v_n$ as defined in §II.3.2:

$$q(y_1,\ldots,y_r,u)=0, \quad \begin{cases} y_{r+1} &= v_{r+1}(y_1,\ldots,y_r,u),\\ &\vdots\\ y_n &= v_n(y_1,\ldots,y_r,u).\end{cases}$$

We recall that an **annihilating system** of $\mathcal{W}$ is a set of polynomials $\mathbf{e} := e_1, \ldots, e_t$ vanishing on $\mathcal{W}$. We do not assume that $\mathcal{W}$ is an isolated subset of the variety $\mathcal{V}(\mathbf{e})$.

Let $\mathcal{W}'$ be an irreducible component of $\mathcal{W}$, isolated in $\mathcal{V}(\mathbf{e})$; we denote by $\mathrm{mul}_{\mathbf{e}}(\mathcal{W}')$ the **multiplicity** of $\mathcal{W}'$ as a solution of the system $\mathbf{e} = 0$. From a computational point of view $\mathcal{W}'$ can be represented by a geometric resolution deduced from the one of $\mathcal{W}$: there exists an irreducible factor Q of q such that $\mathcal{W}'$ is parametrized by

$$Q(y_1,\ldots,y_r,u)=0, \quad \begin{cases} y_{r+1} &= V_{r+1}(y_1,\ldots,y_r,u),\\ &\vdots\\ y_n &= V_n(y_1,\ldots,y_r,u),\end{cases}$$

where the V_i are the remainders of the v_i in the division by Q. Let $K := k(y_1, \ldots, y_r)$, the multiplicity $\mathrm{mul}_{\mathbf{e}}(\mathcal{W}')$ corresponds to the dimension of the quotient $K[T]/(Q(T))[[y_{r+1} - V_{r+1}, \ldots, y_n - V_n]]/(\mathbf{e})$ seen as a $K[T]/(Q(T))$-algebra.

If $\mathcal{W}$ is isolated in $\mathcal{V}(\mathbf{e})$ then we define the **algebraic degree** $\deg^{\mathrm{a}}_{\mathbf{e}}(\mathcal{W})$ as:

$$\deg^{\mathrm{a}}_{\mathbf{e}}(\mathcal{W}) := \sum_{\mathcal{W}'} \mathrm{mul}_{\mathbf{e}}(\mathcal{W}')\deg(\mathcal{W}'),$$

where the sum is over all the irreducible components $\mathcal{W}'$ of $\mathcal{W}$.

Let $K[u] := K[T]/q(T)$ and $x^* = (v_{r+1}(u), \ldots, v_n(u))$ in $K[u]^n$. We assume that the vector x^* represents a set of isolated roots of the system $\mathbf{e} = 0$. We are in the frame of the deflation process of the previous chapter: we say that the geometric resolution is **DA-irreducible** with respect the lifting system $\mathbf{e}$ if x^* is DA-irreducible. By extension, a **generic trace** $\mathcal{T}$ of the geometric resolution is a generic trace of x^*.

The notion of lifting points becomes more tedious than in Chapter II. We recall that π denotes the canonical projection from $\mathcal{W}$ onto k^r and a **lifting point** is a point p in k^r such that the points of the fiber $\pi^{-1}(p)$ are all geometrically smooth on $\mathcal{W}$ and smooth for the projection π. This definition is sufficient to ensure that there is only one geometric resolution lying over the fiber in the sense of Proposition 5 (once a primitive element for the fiber is fixed). In particular this implies that the degree of the fiber is exactly $\deg(\mathcal{W})$.

If $\mathcal{W}$ isolated in $\mathcal{V}(\mathbf{e})$ then we strengthen the definition but distinguish two cases:

- If the geometric resolution of $\mathcal{W}$ is DA-irreducible then we say that p is a **DA-lifting point** if the function `NestedCoordinatesWithTrace` raises no "division by zero" error on x^* when executed after a generic linear change of coordinates on the point
$$q^p(u') = 0,\ y_{r+1} = v^p_{r+1}(u'), \ldots,\ y_n = v^p_n(u'),$$
where u' is a linear form separating the points of the fiber, q^p its minimal polynomial and $v^p_{r+1}, \ldots, v^p_n$ are the parametrization of the coordinates.

- If the geometric resolution of $\mathcal{W}$ is not DA-irreducible we say that p is a DA-lifting point if the function `DaSplit` of §III.5 called on x^* commutes with the specialization of the free variables at p, when executed after a generic linear change of coordinates in $\mathbf{e}$.

These definitions are technical and the lack of geometric interpretation makes them not so easy to manipulate. Anyway, in practice we are satisfied with:

Lemma 12 *In both the above cases, there exists a Zariski open subset of k^r such that any p in it is a DA-lifting point.*

Proof. This is a consequence of Propositions 31, 40 and 43. □

The above definitions are coherent. First if $\mathcal{W}$ is isolated in $\mathcal{V}(\mathbf{e})$ and has a DA-irreducible geometric resolution then a DA-lifting point p is a lifting point: let u be a primitive element for the point of the fiber and q, $v_{r+1}, \dots, v_n$ be the same as above (the parametrization of the geometric resolution lying above the lifting fiber). The fact that p is a DA-lifting point implies that the discriminant of Q is invertible at p by definition. In particular this means that the points of the fibers are smooth on $\mathcal{W}$ and for the projection π. Last if the geometric resolution of $\mathcal{W}$ if DA-irreducible and if `DaSplit` commutes with the specialization of the free variables this implies that `NestCoordinatesWithTrace` also commutes with the specialization when called with the traces found by `DaSplit`.

The very aim of this technical definition instead of being satisfied with generic choices of lifting points is the following: when working over the rational number field the resolution is performed modulo a small prime number first, then the integers are lifted. It would be very inefficient to lift integers of a fiber which has a lifting point of large height. Our strategy is this one: we assume that the resolution modulo the random prime number p is done using random numbers in the range $0, \dots, p-1$. Then we want to find coordinates, primitive elements and magic points of small height before lifting the integers. This searching phase is still done modulo p and we need criteria which ensure that the candidates are good. This is detailed in §IV.5. In particular we need a criterium for testing whether a point in $\mathbb{Z}^r$ of small height is a DA-lifting point or not. According to the above definitions we deduce from the previous chapter the following condition:

Proposition 45 *Let F be a DA-irreducible lifting fiber of lifting system* $\mathbf{e}$ *such that its corresponding algebraic set $\mathcal{W}$ is isolated in $\mathcal{V}(\mathbf{e})$. Let $\mathcal{T}$ be a generic trace of the geometric resolution lying above F. If there exists a Zariski open subset of $GL_{n-r}(k)$ of matrices N such that the function* `NestedCoordinatesWithTrace` *executed on the points of F in the coordinates defined by N and trace $\mathcal{T}$ raises no "division by zero" error then the lifting point of F is a DA-lifting point.*

Proof. This is a reformulation of Proposition 40. □

Finally we revisit the definition of a lifting fiber given in §II.3.4. We need to add a new entry for generic traces.

Let $\mathcal{W}$ be an r-equidimensional algebraic variety. A **fiber** of $\mathcal{W}$ is defined by:

- An invertible $n \times n$ square matrix M with entries in k such that the new coordinates $y = M^{-1}x$ are in *projective Noether position* with respect to $\mathcal{V}$. In the following π represents the finite projection morphism from $\mathcal{W}$ onto the free variables $y_1, \dots, y_r$. The other variables $y_{r+1}, \dots, y_n$ are called dependent;

- An *annihilating system* $\mathbf{e} = (e_1, \dots, e_t)$ of $\mathcal{W}$.

- A *specialization point* $p = (p_1, \dots, p_r)$ for $\mathcal{W}$;

- A *primitive element* $u = \lambda_{r+1}y_{r+1} + \dots + \lambda_n y_n$ of the fiber $\mathcal{W}_p = \pi^{-1}(p)$, with $\lambda_i \in k$ for $i = r+1, \dots, n$;

- The *minimal polynomial* $q(T) \in k[T]$ annihilating u over the points of $\mathcal{W}_p$;

- $n-r$ polynomials $\mathbf{v} = (v_{r+1}, \ldots, v_n)$ of $k[T]$, of degrees strictly less than $\deg(q)$, giving the *parametrization* of $\mathcal{W}_p$ by the zeros of q: $y_j - v_j(z) = 0$ for all $r+1 \leq j \leq n$ and any root z of q.

If the variety $\mathcal{W}$ represented by the fiber F is isolated in $\mathcal{V}(\mathbf{e})$ we say for short that F is **isolated**. Moreover, if the geometric resolution lying over F is **DA-irreducible** we say that F is DA-irreducible. If F is isolated and DA-irreducible it is natural to demand that the lifting point of F be an DA-lifting point. If F is isolated, we respectively denote by $\mathrm{mul}(F)$ and $\deg^{\mathrm{a}}(F)$ the **multiplicity** and the **algebraic degree** of $\mathcal{W}$ with respect to $\mathcal{V}(\mathbf{e})$. Moreover we add a new entry for storing a generic trace of a DA-irreducible lifting fiber:

- A *generic trace* with respect to the fast deflation process on the geometric resolution lying on the fiber.

For the pseudocode we still use the notations of §II.3.4, but we introduce F_{Trace} to denote the generic trace of the fiber F.

We conclude this section by a practical remark: for the sake of saving computations we could add a new entry in the fiber data structure for storing the nested coordinates. This requires taking care that no change of coordinates is done between two liftings if we want to reuse the nested coordinates. We do not detail these aspects here, this does not affect the total theoretical complexity. Moreover when $k = \mathbb{Q}$ the computations of nested coordinates are not the bottleneck of the algorithm since they are performed modulo a small prime number.

IV.2.3 Lifted Curves

Replacing the function `GlobalNewton` of §II.4 by the one of Algorithm IV.1 then `LiftCurve` remains valid if called on DA-irreducible lifting fibers. One slight modification is that `GlobalNewton` takes the trace of the fiber as a new argument. It is also important to observe that `LiftCurve` is now probabilistic since generic enough coordinates are required for computing the nested coordinates.

Proposition 46 *In case of success the complexity of* `LiftCurve` *applied on a DA-irreducible lifting fiber F is in*

$$\mathcal{O}\Big(\log(d)n^4(nL+n^{\Omega})\mathcal{U}(\deg^{\mathrm{a}}(F))^2\Big),$$

in terms of arithmetic operations in k.

IV.2.4 Splittings due to Deflation

The lifting fibers on which we want to perform a lifting must contain a generic trace. During the resolution process of §IV.4 we produce isolated fibers that are not necessarily DA-irreducible. We need to split them into DA-irreducible ones and initialize their generic trace. The definition of DA-lifting points is stated such that the execution of the splitting algorithm of the previous chapter leads to correct decompositions and traces.

ALGORITHM IV.2: **Splittings due to the deflation**

`DaSplitFiber`(F)

- F is an isolated lifting fiber with a DA-lifting point.

If N is generic enough then the function returns a set $\mathbf{F}$ of DA-irreducible lifting fibers.

$r \leftarrow \dim(F)$;
$N \leftarrow \mathsf{W}(n-r)$;
$\mathbf{g} \leftarrow F_{Equations} \circ F_{ChangeOfVariables}$;
$\mathbf{g} \leftarrow \mathbf{g}(F_{LiftingPoint}, y_{r+1}, \dots, y_n)$;
$\mathbf{g} \leftarrow \mathbf{g} \circ N$;
$\mathbf{v} \leftarrow N^{-1} F_{Parametrization}$;
$(Q_i, \mathcal{T}_i)_{i=1,\dots,t} \leftarrow$ `DaSplit`$(\mathbf{g}, F_{MinimalPolynomial}, \mathbf{v})$;
$\mathbf{F} \leftarrow \{\}$;
for i **from** 1 to t **do**
  $F' \leftarrow F$;
  $F'_{MinimalPolynomial} \leftarrow Q_i$;
  $F'_{Parametrization} \leftarrow F_{Parametrization} \bmod Q_i$;
  $F'_{Trace} \leftarrow \mathcal{T}_i$;
  $\mathbf{F} \leftarrow \mathbf{F} \cup \{F'\}$;
return$(\mathbf{F})$;

Proposition 47 *There exists a Zariski open subset of $GL_{n-r}(k)$ such that for any matrix N in it Algorithm IV.2 returns a correct answer, with a complexity in*

$$\mathcal{O}\Big(\log(d) n^4 (nL + n^{\Omega}) \deg^{\mathrm{a}}(F)^2 \mathcal{U}(\deg(F))\Big).$$

Proof. This is a direct consequence of Propositions 42 and 43. □

IV.3 Removing Redundancies

Let $\mathbf{F}$ be a set of lifting fibers. We assume that they all share the same annihilating system $\mathbf{e} = e_1, \dots, e_t$ but that they are not necessarily isolated with respect to it. Let $\mathcal{W}$ be the union of the algebraic varieties represented by the elements of $\mathbf{F}$. We assume that $\mathcal{W}$ is isolated in $\mathcal{V}(\mathbf{e})$. We say that $\mathbf{F}$ is **redundant** if there exists an irreducible component of $\mathcal{W}$ included in the algebraic sets represented by two distinct elements of $\mathbf{F}$. In this section we show how to compute a set $\mathbf{F}'$ of DA-irreducible (and in consequence isolated) lifting fibers representing $\mathcal{W}$ but without redundancy. The computation of $\mathbf{F}'$ from $\mathbf{F}$ is called the **minimization** of $\mathbf{F}$. Computing the components represented by a lifting fiber which are not included in a variety represented by another fiber relies

ALGORITHM IV.3: **Difference**

`Difference`(F^1, F^2)

- F^1 is a lifting fiber.
- F^2 is an DA-irreducible lifting fiber.

The procedure returns F a lifting fiber of the components of F^1 not included in F^2. The primitive element of F^2 must be generic enough. The lifting point of F^1 must be generic enough.

$r_1 \leftarrow \dim(F^1)$;
$r_2 \leftarrow \dim(F^2)$;
if $r_1 > r_2$ **then return**(F^1);
$M_1 \leftarrow F^1_{ChangeOfVariables}$;
$M_2 \leftarrow F^2_{ChangeOfVariables}$;
$(p_1, \ldots, p_n) \leftarrow M_2^{-1} M_1(F^1_{LiftingPoint}, F^1_{Parametrization})$;
$\mathcal{C} \leftarrow$ `LiftCurve`$(F^2, (p_1, \ldots, p_{r_2}))$ in $k[T]/(F^1_{MinimalPolynomial})$;
$\mathcal{C} \leftarrow \mathrm{subs}(t = 1, \mathcal{C})$;
$a \leftarrow F^2_{PrimitiveElement}(p_1, \ldots, p_n)$;
$q \leftarrow \mathcal{C}_{MinimalPolynomial}(a)$ viewed in $k[T]$;
$F \leftarrow F^1$;
$F_{MinimalPolynomial} \leftarrow F^1_{MinimalPolynomial}$ div $\gcd(F^1_{MinimalPolynomial}, q)$;
$F_{Parametrization} \leftarrow F^1_{Parametrization}$ mod $F^1_{MinimalPolynomial}$;
return(F);

on one lifting curve computation. Therefore it is important to ensure that the former fiber is DA-irreducible (and isolated). This process is not as simple as the one given in Appendix B.

IV.3.1 Difference between two Lifting Fibers

The elementary operation underlying the minimization process is the difference between two equidimensional components. Let $\mathcal{W}_1$ (resp. $\mathcal{W}_2$) be a r_1(resp. r_2)-equidimensional variety given by a lifting fiber F^1 (resp. F^2). We need to assume that F^2 is DA-irreducible (and isolated). We are looking for a lifting fiber of the set of components of $\mathcal{W}_1$ not included in $\mathcal{W}_2$. The idea is to compute the fiber of $\mathcal{W}_2$ at the r_2 first coordinates of $\mathcal{W}_1$ and compare the $n - r_2$ coordinates left.

First we explain the method in terms of geometric resolutions. We consider the geometric resolutions lying over F^1 and F^2: M_i is the change of variables, u_i the primitive element, Q_i the minimal polynomial and $\mathbf{V}^i = V^i_{r_i+1}, \ldots, V^i_n$ the parametrization of the geometric resolution lying over F^i, for $i = 1, 2$. If $y^i = M_i^{-1}x$, then the parametrization of $\mathcal{W}_i$ is:

$$Q_i(y_1^i, \ldots, y_{r_i}^i, u_i) = 0, \quad \left\{ \begin{array}{lcl} y_{r_i+1}^i & = & V_{r_i+1}^i(y_1^i, \ldots, y_{r_i}^i, u_i), \\ & \vdots & \\ y_n^i & = & V_n^i(y_1^i, \ldots, y_{r_i}^i, u_i). \end{array} \right.$$

We express the parametrization of F^1 in the coordinates of F^2:

$$P := M_2^{-1}M_1(y_1^i, \ldots, y_{r_1}^i, V_{r_1+1}^1, \ldots, V_{r_2+1}^1).$$

Let $\chi_t(y_1^2, \ldots, y_{r_2}^2)$ be the minimal polynomial of a generic linear form $u_t = t_{r_2+1}y_{r_2+1}^2 + \cdots + t_n y_n^2$ with respect to $\mathcal{W}_2$ where t_i are new parameters. Let $A := u_2(P)$ then the equality

$$\chi_t(P_1, \ldots, P_{r_2}, A) = 0$$

in $k(y_1^1, \ldots, y_{r_1}^1)[T]/(Q_1)$ is equivalent to the inclusion of $\mathcal{W}_1$ in $\mathcal{W}_2$ (see Lemma 18 of §B.5). Therefore the components of $\mathcal{W}_1$ included in $\mathcal{W}_2$ correspond to the greatest common divisor of Q_1 and $\chi_t(P_1, \ldots, P_{r_2}, A) = 0$.

We now consider what happens when specializing $y_1^1, \ldots, y_{r_1}^1$ to the lifting point p_1 of F_1. More precisely let M_1, p_1, u_1, q_1, $\mathbf{v}^1$ be the change of coordinates, the magic point, the primitive element, the minimal polynomial and the parametrization of F. We recall that q_1 and $\mathbf{v}_1$ can be deduced from Q_1 and $\mathbf{V}^1$ by substituting $y_1^1, \ldots, y_{r_1}^1$ by p_1.

Let $k[u] := k[T]/(q_1(T))$ and $p := M_2^{-1}M_1(p_1, \mathbf{v}^1(u))$. We compute the fiber F of $\mathcal{W}_2$ at $(p_1, \ldots, p_{r_2})$ using the algorithm of §II.5.2: it is true that $k[u]$ is not a field but neither inversion nor equality test is performed in $k[u]$. If the primitive element u_2 separates the points of F then we get the corresponding minimal polynomial $q(u_2) = 0$. Let $a := u_2(p)$ and $b := q(a)$: b is an element of $k[u]$, we can view it as an element of $k[T]$ of degree strictly less than $\deg(q)$. Let $\overline{q}$ be the greatest common divisor of b and q, F the restriction of F^1 modulo $q/\overline{q}$. If the lifting point of F_1 is generic enough then F represents the components of F_1 included in F_2. In the sequel we shall write $F =: F_1 \backslash F_2$ for this difference. The method is summarized in Algorithm IV.3.

Proposition 48 *There exists a Zariski open subset of k^n of primitive elements of F^2 and a Zariski open subset of k^{r_1} of lifting points of F^1 yielding a correct result for Algorithm IV.3. In terms of arithmetic operations in k, Algorithm IV.3 has complexity is in*

$$\mathcal{O}\Big(\log(d)n^4(nL + n^\Omega)\mathcal{U}(\deg^{\mathrm{a}}(F_2))^2\mathcal{U}(\deg(F_1))\Big).$$

IV.3.2 Minimization

We are now coming to the problem of minimizing a set of lifting fibers $\mathbf{F}$. Our method relies on the above function `Difference` and is summarized in Algorithm IV.4.

Proposition 49 *In terms of operations in k and in case of success, the complexity of Algorithm IV.4 is in*

$$\mathcal{O}\Big(\log(d)n^4(nL + n^\Omega)\mathcal{U}(\deg^{\mathrm{a}}(\mathbf{F}'))^2\mathcal{U}(D)\Big),$$

Algorithm IV.4: **Minimization**

`Minimize`($\mathbf{F}$)

- $\mathbf{F}$ is a set of lifting fibers.

The procedure returns $\mathbf{F}'$ a set of lifting fibers describing the same algebraic set as $\mathbf{F}$ but without redundancy. The procedure may fail if the lifting points and primitive elements of $\mathbf{F}$ are not generic enough.

> **for** i **from** 0 **to** n **do**
> $\mathbf{F}_i \leftarrow \{F \in \mathbf{F} | \ \dim(F) = n - i\}$;
> $\mathbf{F}'_0 \leftarrow \mathbf{F}_0$;
> **for** i **from** 1 **to** n **do**
> $\mathbf{F}'_i \leftarrow \{\}$;
>   **for** F **in** $\mathbf{F}_i$ **do**
>   **for** F' **in** $\mathbf{F}'_0 \cup \cdots \cup \mathbf{F}'_i$ **do**
> $F \leftarrow$ `Difference`(F, F');
> $\mathbf{F}'_i \leftarrow \mathbf{F}'_i \cup$ `DaSplitFiber`(F);
> $\mathbf{F}' \leftarrow \mathbf{F}'_0 \cup \cdots \cup \mathbf{F}'_n$;
> **return**$(\mathbf{F}')$;

where $D := \sum_{F \in \mathbf{F}} \deg(F)$.

Proof. First we observe that when calling `DaSplitFiber`(F) the fiber F is necessarily isolated. Its lifting point must be generic enough in order to ensure a correct result. When executing `Difference`(F, F') the lifting point of F and the primitive element of F' must be generic enough.
The sum of the costs due to the calls to `Difference` is in

$$\mathcal{O}\left(\log(d) n^4 (nL + n^\Omega) \sum_{F \in \mathbf{F}} \sum_{F' \in \mathbf{F}'} \mathcal{U}(\deg^{\mathrm{a}}(F'))^2 \mathcal{U}(\deg(F))\right)$$

$$\subseteq \mathcal{O}\left(\log(d) n^4 (nL + n^\Omega) \mathcal{U}(\deg^{\mathrm{a}}(\mathbf{F}'))^2 \mathcal{U}(D)\right).$$

The sum of the costs of `DaSplitFiber` is in

$$\mathcal{O}\left(\log(d) n^4 (nL + n^\Omega) \sum_{F' \in \mathbf{F}'} \deg^{\mathrm{a}}(F')^2 \mathcal{U}(\deg(F'))\right)$$

$$\subseteq \mathcal{O}\left(\log(d) n^4 (nL + n^\Omega) \deg^{\mathrm{a}}(\mathbf{F}')^2 \mathcal{U}(\deg(\mathbf{F}'))\right).$$

Finally, note that $\deg(\mathbf{F}') \leq D$. □

ALGORITHM IV.5: **Splitting a Lifting Fiber**

`Split`(F, f)

- F is a lifting fiber.
- f is a polynomial.

The function returns (F', F'') a couple of lifter fibers such that F' is a lifting fiber for the components of F included in $\mathcal{V}(f)$; F'' is a lifting fiber for the other components. The function may fail if the lifting point of F is not generic enough.

$q \leftarrow F_{MinimalPolynomial}$;
$\mathbf{v} \leftarrow F_{Parametrization}$;
$e \leftarrow \gcd(q, f \circ F_{ChangeOfVariables}(F_{LiftingPoint}, \mathbf{v}))$;
$q \leftarrow q/e$;
$F' \leftarrow F$; $F'' \leftarrow F$;
$F'_{MinimalPolynomial} \leftarrow e$; $F'_{Parametrization} \leftarrow \mathbf{v} \bmod e$;
$F''_{MinimalPolynomial} \leftarrow q$; $F''_{Parametrization} \leftarrow \mathbf{v} \bmod q$;
return(F', F'');

IV.4 Resolution Algorithm

We are ready to present the main function of the solver. But before we need to explain how we handle improper intersections.

IV.4.1 Splitting

Let F be a lifting fiber and f be a polynomial function given by a straight-line program of size at most L. We are interested in computing a lifting fiber for the components of the algebraic variety represented by F which are included in $\mathcal{V}(f)$, and another lifting fiber for the other components. The algorithm is very similar to the one of §II.6.6 and requires a generic enough lifting point.

Proposition 50 *The complexity of Algorithm IV.5 is in*

$$\mathcal{O}\Big((n^2 + L)\mathcal{U}(\deg(F))\Big).$$

IV.4.2 Incremental Solving

We recall that $\mathcal{V}_i$ denotes the Zariski closure of $\mathcal{V}(f_1, \ldots, f_i) \setminus \mathcal{V}(g)$, for $i = 1, \ldots, s$. By induction we assume that we have already computed a set of DA-irreducible lifting fibers $\mathbf{F}$ representing $\mathcal{V}_i$ without redundancy and having all $f_1, \ldots, f_i$ as lifting system for $i \geq 0$.

We want to compute such a representation for $\mathcal{V}_{i+1}$. Concerning the initialization of the resolution we must distinguish two cases: if $\mathcal{V}_0$ is empty (that is $g = 0$) then we set $\mathbf{F} = \{\}$ otherwise $\mathcal{V}_0$ is $\overline{k}^n$ it is pretty easy to build a lifting fiber. This particular case should be handled by each sub-function of the algorithm. For the sake of presentation we do not insist more about this.

We denote by $\mathbf{F}_j$ the subset of $\mathbf{F}$ of lifting fibers of codimension j, for $j = 0, \ldots, i$. For any F in $\mathbf{F}$ we denote by F^r a lifting fiber of the components of F which are not contained in $\mathcal{V}(f_{i+1})$ and F^i a fiber for the other components. One can compute F^r and F^i from F using Algorithm IV.5, assuming that F has a generic enough lifting point. For each $F \in \mathbf{F}$ one can compute a lifting fiber F' of the Zariski closure of $(\mathcal{W} \cap \mathcal{V}(f)) \setminus \mathcal{V}(g)$, where $\mathcal{W}$ is the algebraic set represented by F^r. This computation is achieved using the algorithm of §II.6.7. The fiber F' is not isolated with respect to $f_1, \ldots, f_{i+1}$ a priori.

In order to remove the redundancies and split the fibers into DA-irreducible ones we apply the minimization process of the previous section on the set $\mathbf{F}'$:

$$\mathbf{F}' := \{F^i \mid F \in \mathbf{F}\} \cup \{F' \mid F \in \mathbf{F}\}.$$

The method is summarized in Algorithm IV.6. As in §II.7.2 we introduce functions keeping track of all the random choices occurring in the algorithm. Let F be a fiber of dimension r:

- N^F denotes the choice of a **Noether point** in k^{n-r} of F.
- L^F denotes the choice of a **lifting point** of F.
- C^F denotes the choice of a **Cayley point** in k^{n-r}.

Proposition 51 *Let $\delta_i^a := \deg^{\mathrm{a}}_{f_1,\ldots,f_i}(\mathcal{V}_i)$. In terms of operations in k and in case of success the complexity of Algorithm IV.6 is in*

$$\mathcal{O}\left(\log(d) n^4 (nL + n^\Omega) \mathcal{U}(\deg(f_{i+1})\deg(\mathcal{V}_i)) \left(\mathcal{U}(\delta_i^a)^2 + \mathcal{U}(\delta_{i+1}^a)^2\right)\right)$$

The correctness of the result depends on lucky choices of the parameters N^F, L^F, C^F and W. Choices of these parameters correspond to choices of elements in k. Bad choices are enclosed in strict algebraic subsets of k.

Proof. From Proposition 50 the costs of the function `Split` is not significant. The cost due to `LiftCurve` is in

$$\mathcal{O}\left(\log(d) n^4 (nL + n^\Omega) \textstyle\sum_{F^r} \mathcal{U}(\deg^{\mathrm{a}}(F^r))^2\right)$$

$$\subseteq \mathcal{O}\left(\log(d) n^4 (nL + n^\Omega) \mathcal{U}(\deg^{\mathrm{a}}_{f_1,\ldots,f_i}(\mathcal{V}_i))^2\right).$$

From Proposition 23 the sum of the costs of the function `OneDimensionalIntersect` is in

$$\mathcal{O}\left(n(L + n^2) \textstyle\sum_{F \in \mathbf{F}} \mathcal{U}(\deg(F)) \mathcal{U}(\deg(f_{i+1})\deg(F))\right)$$

$$\subseteq \mathcal{O}\left(n(L + n^2) \mathcal{U}(\deg(\mathcal{V}_i)) \mathcal{U}(\deg(f_{i+1})\deg(\mathcal{V}_i))\right).$$

ALGORITHM IV.6: **Intersection**

`Intersect`$(\mathbf{F}, f, g)$

- $\mathbf{F}$ is a set of DA-irreducible lifting fibers without redundancy.
- f, g are polynomial functions in $k[x_1, \dots, x_n]$.

Let $\mathcal{W}$ be the algebraic set represented by $\mathbf{F}$. If the choices of the parameters are generic enough then the function returns a set of DA-irreducible lifting fibers describing $\overline{(\mathcal{W} \cap \mathcal{V}(f)) \setminus \mathcal{V}(g)}$ without redundancy.

$\mathbf{F}' \leftarrow \{\}$;
for F **in** $\mathbf{F}$ **do**
  `ChangeFreeVariables`(F, N^F);
  `ChangeLiftingPoint`(F, L^F);
  `ChangePrimitiveElement`(F, C^F);
  $F^r, F^i \leftarrow$ `Split`(F, f);
  $\mathcal{C} \leftarrow \mathrm{subs}(t = y_r,$`LiftCurve`$(F^r, \mathsf{L}^F + (0, \dots, 0, 1)))$;
  # Consider $\mathcal{C}$ as the geometric resolution of the
  # corresponding lifted curve and perform:
  $F' \leftarrow$ `OneDimensionalIntersect`$(\mathcal{C}, f, \mathsf{C}^F, g)$;
  $\mathbf{F}' \leftarrow \mathbf{F}' \cup \{F', F^i\}$;
return(`Minimize`$(\mathbf{F}')$);

Last, noticing that $\sum_{F' \in \mathbf{F}'} \deg(F') \leq \deg(f_{i+1}) \deg(\mathcal{V}_i)$, we deduce from Proposition 49 the cost of the minimization:

$$\mathcal{O}\left(\log(d) n^4 (nL + n^\Omega) \mathcal{U}(\deg(f_{i+1}) \deg(\mathcal{V}_i)) \mathcal{U}(\deg^{\mathrm{a}}_{f_1, \dots, f_{i+1}}(\mathcal{V}_{i+1}))^2\right).$$

□

IV.4.3 Main Function

It is straightforward to deduce the resolution algorithm from the previous function. This is detailed in Algorithm IV.7. The initialization of the incremental process consists in building a generic lifting fiber for the whole space $\overline{k}^n$. We recall that the function W is a parameter defined in §IV.2, it is a function returning random matrices bringing the variables into generic position necessary to the fast deflation algorithm.

Theorem 5 *In terms of operations in k and in case of success the complexity of Algorithm IV.7 is in*

$$\mathcal{O}\left(s \log(d) n^4 (nL + n^\Omega) \mathcal{U}(d\delta^a)^3\right),$$

ALGORITHM IV.7: **Equidimensional Decomposition**

`GeometricSolve`$(\mathbf{f}, g)$

- $\mathbf{f}$ is a sequence of polynomials in $k[x_1, \ldots, x_n]$.
- g is a polynomial in $k[x_1, \ldots, x_n]$.

The function returns the equidimensional decomposition of the system $\mathbf{f} = 0$, $g \neq 0$ encoded by a set of DA-irreducible lifting fibers without redundancy.

> $\mathbf{F} \leftarrow$ Initialization;
> **for** f **in f do**
> $\mathbf{F} \leftarrow$ `Intersect`$(\mathbf{F}, f, g)$;
> **return**$(\mathbf{F})$;

where $\delta^a := \max_{i=1,\ldots,s} \deg^{\mathrm{a}}_{f_1,\ldots,f_i}(\mathcal{V}_i)$. *The correctness of the result depends on lucky choices of the parameters* N^F, L^F, C^F *and* W. *Choices of these parameters corresponds to choices of elements in* k. *Bad choices are enclosed in strict algebraic subsets of* k.

Proof. The complexity is a direct consequence of the above proposition: it suffices to notice that

$$\deg(\mathcal{V}_i) \leq \delta^a, \quad \mathcal{U}(\deg^{\mathrm{a}}_{f_1,\ldots,f_{i+1}}(\mathcal{V}_{i+1})) \leq \delta^a, \qquad \text{for } i = 1, \ldots, s.$$

□

IV.5 Special Case of the Integers

Our geometric resolution algorithm works well on a field k of characteristic zero. If $k = \mathbb{Q}$ we pick up a random prime p and first solve the system modulo p. If p is lucky then the algorithm computes a correct resolution of the system modulo p. The luckiness of p corresponds to the commutation of the computations over $\mathbb{Q}$ and modulo p. This is why the output is correct except for a finite number of primes. In Chapter V we show the good behavior of our program: if p is taken near 2^{32} we can solve systems with up to 6000 solutions without witnessing any failure on several executions. Each lifting fiber solution has a generic enough change of variables, a generic enough primitive element and a generic enough lifting point. In this situation generic enough means random integers in the range $0, \ldots, p-1$. Before recovering informations over the field $\mathbb{Q}$ it is important to find changes of variables, lifting points and primitive elements with small height. This is the purpose of this section.

IV.5.1 Moving the Coordinates

Let $\mathcal{W}$ be an r-equidimensional variety, $\mathfrak{I}$ its annihilating ideal in $k[x_1, \ldots, x_n]$ and F one of its DA-irreducible lifting fibers. We have shown in §II.5.2 how to change the lifting point of F and its primitive element. We now face the problem of changing its variables yielding the projective Noether position. Our strategy is based on homotopy.

Let M_0 denote $F_{ChangeOfVariables}$ and M_1 be another matrix of $GL_n(k)$. Let m be the number of rows which are not zero in $M_1 - M_0$. Let t be a new variable, $M_t = M_0 + t(M_1 - M_0)$ (M_t is invertible in $k[[t]]$), $(y_1^t, \ldots, y_n^t) = M_t^{-1}(x_1, \ldots, x_n)$ and h_t represent the following evaluation morphism:

$$\begin{array}{rrcl} h_t: & k[x_1, \ldots, x_n] & \to & k[t, x_1, \ldots, x_n] \\ & x_i & \mapsto & y_i^t, \; i = 1, \ldots, n. \end{array}$$

We define $\mathfrak{I}_t$ as the image of $\mathfrak{I}$ by h_t and $\mathcal{V}_t$ the variety of $\mathfrak{I}_t$ in $\overline{k}^{n+1}$. We let $\mathcal{C}_t' = \mathcal{V}_t \cap \mathcal{V}(x_1 - p_1, \ldots, x_r - p_r)$, where p is the lifting point of F. The variety $\mathcal{C}_t'$ is has dimension 1. We denote by $\mathcal{C}_t$ the components of dimension exactly 1 and satisfying $k[\mathcal{C}_t] \cap k[t] = (0)$. The variable t is a free variable with respect to $\mathcal{C}_t$. We are in the frame of Schost's thesis [Sch00b] about systems with parameters. The degree of $\mathcal{C}_t$ is bounded by $2^m \deg(\mathcal{W})$. The degree of the generic fiber with respect to the parameter t is exactly $\deg(\mathcal{W})$. We deduce from [Sch00b, Théorème 8] that there exists a parametrization of $\mathcal{C}_t$ satisfying:

- The primitive element u of F is a primitive element of the $k(t)$ algebra $B' := k(t) \otimes k[\mathcal{C}_t]$.
- The minimal polynomial $Q(T) \in k(t)[T]$ of u in B' has degree in T exactly $\deg(\mathcal{W})$.
- There exist polynomials $V_{r+1}(T), \ldots, V_n(T)$ in $k(t)[T]$ such that $x_i = V_i(u)$ in B', for $i = r+1, \ldots, n$, and $\deg(V_i) < \deg(\mathcal{W})$.
- The numerators and the least common multiple of the denominators of the coefficients of $Q, V_{r+1}, \ldots, V_n$ have degree at most $2^m \deg(\mathcal{W})$.

Note that this result generalizes Corollary 2. Once one has computed the above parametrization of the curve $\mathcal{C}_t$ it suffices to substitute t by 1 to deduce the fiber F' of $\mathcal{W}$ for the change of coordinates M_1, at point p for the primitive element u. We reformulate Proposition 45: if $t = 1$ does not annihilate any denominator of Q and the V_i then we define q and v_i, as the values of Q and the V_i for $t = 1$. If q has degree $\deg(\mathcal{W})$ and is square free, then M_1 is a projective Noether position for $\mathcal{W}$ and u a primitive element. If the execution `NestedCoordinatesWithTrace`(F', F_{Trace}) does not raise an error after a generic change of coordinates then p is a DA-lifting point.

We now describe an algorithm for computing Q and the V_i. The point $t = 0$ determines a generic fiber for $\mathcal{C}_t$ since the Jacobian matrix of $\mathfrak{I}_t$ has full rank at each point of this fiber. Therefore we can apply our global lifting algorithm for computing the values of Q and the V_i with coefficients in $k[[t]]$ at precision $\mathcal{O}(t^{2^{m+1}\deg(\mathcal{W})+1})$. Then

---

ALGORITHM IV.8: **Moving the Coordinates**

`MoveCoordinates`(F, M_1)

- F is a DA-irreducible lifting fiber of dimension r.
- M_1 is a matrix.

The procedure returns a fiber F' equivalent to F but with change of variables M_1, if possible.

> $r \leftarrow \dim(F)$;
> $\delta \leftarrow \deg(F)$;
> $M_0 \leftarrow F_{ChangeOfVariables}$;
> $m \leftarrow$ number of nonzero rows in $M_1 - M_0$;
> $M_t \leftarrow M_0 + t(M_1 - M_0)$;
> $\mathbf{g} \leftarrow F_{Equations} \circ M_t$;
> $\mathbf{h} \leftarrow \mathbf{g}(p_1, \dots, p_r, x_{r+1}, \dots, x_n)$;
> StopCriterion $\leftarrow ((k) \mapsto k > 2^{m+1}\delta)$;
> Q,$\mathbf{V} \leftarrow$ `GlobalNewton`$(\mathbf{h}, F_{PrimitiveElement}, F_{MinimalPolynomial},$
> $\quad F_{Parametrization}$,StopCriterion$, F_{Trace})$;
> $\mathbf{W} \leftarrow [z\frac{\partial Q}{\partial T} \bmod Q : \ z \in \mathbf{V}]$;
> $q \leftarrow$ PadeApproximant(Q);
> $\mathbf{w} \leftarrow$ [PadeApproximant(z) : $z \in \mathbf{W}$];
> $F'_{MinimalPolynomial} \leftarrow \mathrm{subs}(t = 1, q)$;
> $F'_{Parametrization} \leftarrow \mathrm{subs}(t = 1, \mathbf{w})$;
> **return**(F');

---

the coefficients in $k(t)$ are recovered by a Padé approximant method due historically to Kronecker [Kro81], Hermite and Padé [Pad92]. The fast Padé approximant algorithm is due to Brent, Gustavson and Yun [BGY80]: the complexity is the same as the greatest common divisor. In Algorithm IV.8 we call the function `PadeApproximant`(z) which returns a rational function in $k(t)$ with a numerator and denominator of degree d if z is a power series at precision $2d + 1$.

Proposition 52 *In terms of arithmetic operations in* k*, the complexity of Algorithm IV.8 is in:*

$$\mathcal{O}\left(\log(d) n^4 (nL + n^\Omega) \mathrm{mul}(F)^2 \mathcal{U}(\deg(F)) \mathcal{U}(2^m \deg(F))\right).$$

IV.5.2 Changing Back the coordinates

Let F be a DA-irreducible lifting fiber and M denote $F_{ChangeOfVariables}$. We divide the matrix M into four blocks:

$$M = \begin{pmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{pmatrix},$$

such that $M_{1,1}$ is a $r \times r$ matrix. We assume that $M_{1,1}$ and $M_{2,2}$ are invertible. This is satisfied if M is a random matrix. We are looking for an invertible matrix N such that, written with the same block pattern,

$$N := \begin{pmatrix} N_{1,1} & 0 \\ N_{2,1} & N_{2,2} \end{pmatrix} \quad \text{and} \quad MN = \begin{pmatrix} Id & ? \\ 0 & Id \end{pmatrix}.$$

We get the following equations:

$$\begin{cases} M_{1,1}N_{1,1} + M_{1,2}N_{2,1} = Id, \\ M_{2,1}N_{1,1} + M_{2,2}N_{2,1} = 0, \\ M_{2,2}N_{2,2} = Id. \end{cases}$$

This system admits a unique solution N given by:

$$\begin{cases} N_{2,2} = M_{2,2}^{-1}, \\ N_{1,1} = (M_{1,1} - M_{1,2}M_{2,2}^{-1}M_{2,1})^{-1}, \\ N_{2,1} = -M_{2,2}^{-1}M_{2,1}N_{1,1}. \end{cases}$$

This comes from the following identity:

$$M = \begin{pmatrix} Id & M_{1,2}M_{2,2}^{-1} \\ 0 & Id \end{pmatrix} \begin{pmatrix} M_{1,1} - M_{1,2}M_{2,1}^{-1}M_{2,1} & 0 \\ M_{2,1} & M_{2,2} \end{pmatrix}.$$

If p denotes the magic point of F, u its primitive element, q its minimal polynomial and v the parametrization of the coordinates then the fiber F' composed of $M' = MN$, $p' = N_{1,1}^{-1}p$, $v' = N_{2,2}^{-1}v - N_{2,1}p$, $u' = u \circ N$ is a DA-irreducible lifting fiber equivalent to F.

IV.5.3 Finding a Small Noether Position

From the previous paragraph we can assume that F has a change of variables M in the form:

$$M = \begin{pmatrix} Id & ? \\ 0 & Id \end{pmatrix}.$$

We can search small Noether positions in the form

$$M' = \begin{pmatrix} Id & ? \\ 0 & Id \end{pmatrix}.$$

For such a candidate M' we try to compute the lifting fiber with Algorithm IV.8. In order to avoid a factor 2^r in the complexity we use the following strategy. We build the sequence of matrices $(M_i)_{i=1,\ldots,r}$ such that M_i contains the first i rows of M' and the other rows of M. We apply Algorithm IV.8 to go from M to M_1 then from M_1 to M_2 and so on until $M_r = M'$. If all the intermediates computed fibers are DA-irreducible lifting fibers then we are done. Otherwise we must pick up another candidate M'.

Corollary 9 *From a lifting fiber F of dimension r which has a generic enough change of coordinates, the computation of an equivalent fiber F' with a change of coordinates in the form*

$$\begin{pmatrix} Id & ? \\ 0 & Id \end{pmatrix},$$

requires at most

$$\mathcal{O}\left(r \log(d) n^4 (nL + n^{\Omega}) \mathcal{U}(\deg^{\mathrm{a}}(F))^2\right),$$

in terms of operations in $\mathbb{Z}/p\mathbb{Z}$ and in case of a lucky candidate M'.

IV.5.4 Finding a Small Primitive Element

For a lifting fiber F known modulo p we are now interested in finding a primitive element of small height. Our strategy is the naive one: we pick up at random small linear forms and test whether they separate the points of the fiber modulo p. In practice it is often a good idea to test first whether one of the coordinates is primitive.

IV.5.5 Finding a Small Lifting Point

For a lifting fiber F modulo p we want to find out a lifting point of small height. Our strategy is again the naive one: we pick up at random points of small height in $\mathbb{Z}^r$, compute the corresponding fiber using Algorithm II.5. Proposition 45 gives a criterion for checking that the candidate is a lifting point: we test whether the degree of the minimal polynomial still equals $\deg(F)$ and if the fiber is still DA-irreducible by calling `NestedCoordinatesWithTrace` after a generic linear change of coordinates.

IV.5.6 Lifting the Integers

Before lifting the integers we first call successively the three preceeding functions. Then the lifting is exactly the same as in §II.4.6 except that the StopCriterion function takes as input the local matrix N of `GlobalNewton` and tests the rational reconstruction of the p-adic integers in the original variables. This involves changing back the coordinates in each execution of StopCriterion. The complexity remains linear in the size of the integers of the output.

IV.5.7 Lifting the Free Variables

In case one would be tempted to lift the free variables and write down the geometric resolution lying over a lifting fiber in an expanded form we give a reasonable strategy. First we find small coordinates, primitive elements an magic point modulo p and lift the integers. Then, for various prime numbers p' we lift the free variables $y_1, \ldots, y_r$ in the power series ring $\mathbb{Z}/p'\mathbb{Z}[[y_1,\dots,y_r]]$ (assuming that the lifting point is 0). The requested precision is the degree of the fiber. We combine the different geometric resolution over $\mathbb{Z}/p'\mathbb{Z}[[y_1,\dots,y_r]]$ with an effective Chinese remainder theorem before reconstructing rational numbers. These computations are performed until the reconstructed geometric resolution satisfies the annihilating system. This last test can be performed fast in a probabilistic way: specialize the free variables at a random point and evaluate the system on this fiber modulo a new random prime number.

The complexity of this method relies on the complexity of multivariate power series. In Appendix C we propose an asymptotically fast multiplication for multivariate power series. Therefore the complexity of this lifting is essentially linear (up to logarithmic factors) in the size of the output.

We can find in [Sch00b] a mathematical framework to study the probability of success of this method in an even more generic situation when the variables are not necessarily in Noether position. A multivariate Padé approximant is provided.

IV.6 Examples

The algorithms have been implemented in the continuation of Kronecker version 0.1 presented in Chapter II and written in the Magma computer algebra system. Version 0.16 contains the equidimensional decomposition and a pre-version of the fast deflation algorithm. Since version 0.166 the definitive implementation of the fast deflation algorithm is integrated and we speed up the computations using factorization techniques as explained in the next chapter. But the reader must be misstaken: we do not use *true* factorization as explained in §V.5. This implies that the probabilities of success still rely on Zariski open sets and not on Hilbert's irreducibility theorem [Hil92]. Examples presented below come from different horizons. When not specified, timings concern a Celeron 400MHz running Linux with 128Mb internal memory. We use Magma version 2.6.

IV.6.1 Toy Example

The following example has been constructed by Schost as a debugging test. The problem is composed of polynomials in $\mathbb{Q}[y, y_0, z, z_0]$:

$$\begin{aligned}
f_1 &:= 1 + 2\,{y_0}^2 z^2 - 2\,{y_0}^2 y^3 + {y_0}^4 + 2\,{y_0}^2 {z_0}^2 - 2\,{y_0}^2 + 2\,{z_0}^2 z^2 \\
&\quad + z^4 + 2\,y^3 z^2 - 2\,z^2 - 2\,y^3 {z_0}^2 + {z_0}^4 - 2\,{z_0}^2 + y^6 + 2\,y^3,
\end{aligned}$$

$$\begin{aligned}
f_2 &:= y_0\,\left(6\,y^5 + 3\,\left(-2\,{y_0}^2 + 2\,z^2 + 2 - 2\,{z_0}^2\right) y^2\right) \\
&\quad - \left(4\,{y_0}^3 + 4\,y_0\,\left(z^2 - y^3 - 1 + {z_0}^2\right)\right)(1+y)\,,
\end{aligned}$$

$$\begin{aligned}
f_3 &:= \left(6\,y^5 + 3\,\left(-2\,{y_0}^2 + 2\,z^2 + 2 - 2\,{z_0}^2\right) y^2\right)(z_0 - 3) \\
&\quad - \left(4\,{z_0}^3 + 2\,z_0\,\left(2\,{y_0}^2 + 2\,z^2 - 2 - 2\,y^3\right)\right)(1+y)\,,
\end{aligned}$$

$$\begin{aligned} f_4 &:= \left(6\,y^5 + 3\,\left(-2\,y_0{}^2 + 2\,z^2 + 2 - 2\,z_0{}^2\right)y^2\right)(z-6) \\ &\quad - \left(4\,z^3 + 2\,z\left(2\,y_0{}^2 + 2\,z_0{}^2 + 2\,y^3 - 2\right)\right)(1+y)\,. \end{aligned}$$

These polynomials share subexpressions, we create the following straight-line program to evaluate them using the function `optimize` of Maple:

```
t1 := y0^2;
t2 := z^2;
t5 := y^2;
t6 := t5*y;
t9 := t1^2;
t10 := z0^2;
t16 := t2^2;
t22 := t10^2;
t24 := t5^2;
t27 := 1+2*t1*t2-2*t1*t6+t9+2*t1*t10-2*t1+2*t10*t2
       +t16+2*t6*t2-2*t2-2*t6*t10+t22-2*t10+t24*t5+2*t6;
t33 := 6*t24*y+6*(-t1+t2+1-t10)*t5;
t39 := 1+y;
f1 := t27;
f2 := y0*t33-4*(t1*y0+y0*(t2-t6-1+t10))*t39;
f3 := t33*(z0-3)-(4*t10*z0+4*z0*(t1+t2-1-t6))*t39;
f4 := t33*(z-6)-(4*t2*z+4*z*(t1+t10+t6-1))*t39;
```

This straight-line program has 100 arithmetic operations (scalar and non-scalar). The modular resolution is performed modulo the prime number $p = 377158259$. The modular resolution requires 36s. We find two components of dimension 3 with respective degrees 2 and 3, one component of dimension 2 and degree 4 and 14 isolated points. Finding small Noether positions, small lifting points and small primitive elements and lifting the integers takes about 20s. Lifting the free variables takes about 10s more. The total time is 66s. Here is for instance the minimal polynomial found for the component of dimension 2 and degree 3:

$$\begin{aligned} &T^3 + (-3/10y_1 + 6/5y_2 + 871/100)T^2 \\ &+(3/100y_1^2 - 6/25y_1y_2 - 87/50y_1 + 12/25y_2^2 + 3483/500y_2 + 12639/500)T \\ &-1/1000y_1^3 + 3/250y_1^2y_2 + 87/1000y_1^2 - 6/125y_1y_2^2 - 87/125y_1y_2 - 2523/1000y_1 \\ &+8/125y_2^3 + 1393/1000y_2^2 + 2527/250y_2 + 6113/250. \end{aligned}$$

For comparison, the function `ProbableRadicalDecomposition` of Magma takes about 10s with a *grevlex* ordering and 25s for a *lex* ordering. I stopped the function `Primary-Decomposition` after 14min and 100Mb of memory consumption.

IV.6.2 Four Bars Mechanical System

The present problem has been given to me by Rouillier in the form of 4 expanded equations in 4 variables X_1, X_2, Y_1, Y_2 of degree 4. The solution set decomposes into a curve of degree 4 and 66 isolated points. We compare timings with Magma and FGb [Fau94, Fau, Fau99] written by Faugère in C.

$$A := X_2Y_1, \quad B := X_2Y_2, \quad C := X_1Y_1, \quad D := X_1Y_2,$$

$$\begin{aligned}
f_1 := \;& A(64715623X_2 - 56396232Y_2 + 9599150X_1 - 25500000Y_1 - 1180728)\\
&+B(+9263117X_2 - 8968923Y_2 - 56800727X_1 - 7848711)\\
&+C(-6208267Y_1 + 16531076Y_2 + 7914896X_1 - 7848711)\\
&+D(50187964Y_2 - 336032X_1 + 1180728)\\
&-27049227CD - 138036964BD + 4117176A^2 + 144135801BA + 33148064CA\\
&-88850928D^2 + 85988894DA - 437457B^2 + 1692600C^2\\
&+3724274Y_2^2 + 4228769X_1^2 + 3724274Y_1^2 + 4228769X_2^2,
\end{aligned}$$

$$\begin{aligned}
f_2 := \;& A(-193710490X_2 - 58138007Y_1 - 35747673Y_2 + 53185603X_1 + 15334721)\\
&+B(127693X_2 - 6721370Y_2 + 94924825X_1 + 14096591)\\
&+C(-3975762Y_1 + 51416636Y_2 - 98785664X_1 + 14096591)\\
&+D(-53057910X_1 - 15334721 + 31771910Y_2)\\
&+263600135A^2 + 146551534AB - 79949047B^2 + 28327488X_2^2 - 63187811AC\\
&-287586667AD - 180296540BD + 51817867C^2 + 3829033Y_1^2 + 29442805CD\\
&-4144648(D^2) + +3829033(Y_2^2) + 28327488(X_1^2),
\end{aligned}$$

$$\begin{aligned}
f3 := \;& A(+63996483X_2 - 13286905Y_1 - 54118122Y_2 + 21700734X_1 + 363197)\\
&+B(+10138668X_2 - 8437590Y_2 - 57100810X_1 - 7813029)\\
&+C(-5611697Y_1 + 4849315Y_2 + 6895672X_1 - 7813029)\\
&+D(+48506424Y_2 - 11562065X_1 - 363197)\\
&+7592648AC - 28306621A^2 + 46004152AB + 4921858B^2 + 4713110X_2^2\\
&+114839711AD - 38839931BD + 2335600C^2 + 3244955Y_1^2 - 428427CD\\
&-79275631D^2 + 3244955Y_2^2 + 4713110X_1^2,
\end{aligned}$$

$$\begin{aligned}
f_4 := \;& A(-2136131X_2 + 14890455Y_1 + 9219694Y_2 - 4865320X_1 + 316794)\\
&+B(-3725955X_2 + 14240874Y_2 + 12103391X_1 - 4831066)\\
&+C(-38120520Y_1 - 649580Y_2 + 9967259X_1 - 4831066)\\
&+D(-47340215Y_2 + 1139364X_1 - 316794)\\
&+7358549A^2 + 6996942AB + 5489972B^2 + 634952X_2^2 - 30575734AC\\
&-22339292AD - 36411108BD + 39370741C^2 + 9228863Y_1^2 + 1161568CD\\
&+59841456D^2 + 9228863Y_2^2 + 634952X_1^2.
\end{aligned}$$

In the following table, PRD stands for the function `ProbableRadicalDecomposition` of Magma, AGb and FGb are developed by Faugère. Computations with Fgb has been done on the web site https://www-calfor.lip6.fr/ and the other ones at MEDICIS [Med].

Kronecker 0.166	Pentium III 600MHz	4min15s
Kronecker 0.16	Pentium III 600MHz	20min
Magma *grevlex*	Pentium III 600MHz	86min
Magma PRD *grevlex*	Pentium III 600MHz	98min
AGb F4 *grevlex*	Pentium II 400MHz	27min
FGb 1.1.1021 F4 *grevlex*	Pentium II 233MHz	1min30s
Magma PRD *lex*	EV56 500MHz	>496min, >800Mb

IV.6.3 A Problem from Invariant Theory

This system was brought to us by Thiéry (Université de Lyon I). It illustrates the efficiency of our deflation process. It is composed of 10 polynomials in

$$\mathbb{Q}[x_{12}, x_{13}, x_{23}, x_{14}, x_{24}, x_{34}, x_{15}, x_{25}, x_{35}, x_{45}].$$

$$\begin{array}{l}
x_{12} + x_{13} + x_{14} + x_{15},\\
x_{12} + x_{23} + x_{24} + x_{25},\\
x_{13} + x_{23} + x_{34} + x_{35},\\
x_{14} + x_{24} + x_{34} + x_{45},\\
x_{15} + x_{25} + x_{35} + x_{45},\\
x_{12}^2 + x_{13}^2 + x_{14}^2 + x_{23}^2 + x_{15}^2 + x_{24}^2 + x_{25}^2 + x_{34}^2 + x_{35}^2 + x_{45}^2,\\
x_{12}^3 + x_{13}^3 + x_{14}^3 + x_{23}^3 + x_{15}^3 + x_{24}^3 + x_{25}^3 + x_{34}^3 + x_{35}^3 + x_{45}^3,\\
x_{12}^4 + x_{13}^4 + x_{14}^4 + x_{23}^4 + x_{15}^4 + x_{24}^4 + x_{25}^4 + x_{34}^4 + x_{35}^4 + x_{45}^4,\\
x_{12}^5 + x_{13}^5 + x_{14}^5 + x_{23}^5 + x_{15}^5 + x_{24}^5 + x_{25}^5 + x_{34}^5 + x_{35}^5 + x_{45}^5,\\
x_{12}^6 + x_{13}^6 + x_{14}^6 + x_{23}^6 + x_{15}^6 + x_{24}^6 + x_{25}^6 + x_{34}^6 + x_{35}^6 + x_{45}^6.
\end{array}$$

The solution is the point 0. Therefore we deduce that the multiplicity is 720. In order to speed up our computations we eliminate linear variables as many as possible. There remains 5 equations in 5 variables. Our program finds 0 after 216s as the only solution. The product of the elements of the multiplicity sequence is 8 which is far from 720. This shows that our complexity estimate in terms of algebraic degree can be very pessimistic. Moreover if we add the product of the variables as an inequation the resolution takes 35s to find the empty set.

Chapitre V

Programmation

Nous avons basé l'étude de complexité de notre algorithme de résolution géométrique sur l'utilisation d'algorithmes asymptotiquement rapides pour la manipulation des polynômes. Dans ce chapitre nous discutons la pertinence de ce modèle en pratique, face à des problèmes concrets. Nous présentons notre implantation Kronecker écrite dans le système de calcul formel Magma. Nous discutons certaines accélérations possibles des calculs basées sur la factorisation. Cela nous conduit naturellement à la question du calcul de la décomposition en composantes irréductibles d'une variété algébrique. Nous donnons quelques éléments de réponse, éclairés par des implantations expérimentales. Tout au long ce chapitre nous illustrons notre propos avec divers exemples de systèmes polynomiaux.

V.1 Introduction

La première version publique de notre programme Kronecker date de juillet 1999 (version 0.1), le code source documenté comportait alors 4700 lignes et pesait 156ko. Cette première distribution correspondait strictement à l'algorithme décrit dans le Chapitre II pour résoudre un système d'équations et d'inéquations répondant à certaines conditions de généricité (*suite régulière réduite*). Le passage à la décomposition en composantes équidimensionnelles s'est effectué progressivement. La version 0.16, diffusée en avril 2000, comportait une version préliminaire de l'algorithme de dégonflement (*deflation*, présentée dans le Chapitre III) et déjà le processus de minimisation expliqué dans le Chapitre IV. Cette version documentée comportait alors 8500 lignes pour 250ko. Actuellement la version en développement est la 0.166. Elle comporte pour l'instant 9000 lignes pour 300ko.

Concernant les performances de ces versions successives, peu de changements sont à noter entre la première et la deuxième version. Dans la version courante nous avons consacré une partie de nos efforts à améliorer les performances de notre code. Ce travail s'est concrétisé lors de mon séjour d'un mois, en août 2000, dans l'équipe de John Cannon développant Magma. J'ai alors eu accès aux sources du logiciel, à des outils d'analyses de performances mais surtout à l'extrême compétence d'Allan Steel. Celui-ci a entreprit l'écriture en C d'un domaine de calculs d'évaluation (straight-line programs) pour Magma et surtout a implanté la substitution de Kronecker accélérant ainsi l'arithmétique des polynômes à deux variables. Nous discutons ces points en détail en V.2.2.

Nourri des pertinentes remarques d'Éric Schost et poussé par de nouvelles idées de Luis Migel Pardo et Jorge San Martin j'ai entrepris d'exploiter les outils de factorisation sans détériorer la nature probabiliste de l'algorithme : les succès pratiques ont été appréciables. Ces idées sont présentées en V.4.

La version actuelle comporte quelques fonctions expérimentales ouvrant la voie au calcul de la décomposition en composantes irréductibles d'une variété algébrique. Nous présentons quelques résultats partiels en V.5.

V.1.1 Choix d'un système de calcul formel

L'implantation de Kronecker s'est amorcée dans le prolongement du Projective Noether Package (cf. Annexe A) en Maple, puisque, rappelons le, ce dernier dispose d'une structure de données pour les calculs d'évaluation particulièrement efficace, implanté via des graphes acycliques orientés [GSZ95]. Ainsi la première version de Kronecker pour Maple date de décembre 1997 et s'appelait Noether. Elle comportait plus de 2000 lignes pour 60ko. Nous avions retranscrit fidèlement la version déforestée (au sens de l'Annexe A) de la thèse de Morais [Mor97]. Pour fixer les idées, le petit système suivant appelé `cyclique` 3 ne nécessitait pas moins de 8s (Celeron 400MHz sous Linux):

$$x_1 + x_2 + x_3 = 0,\ x_1x_2 + x_2x_3 + x_3x_1 = 0,\ x_1x_2x_3 = 1.$$

En octobre 1998 le même système ne prenait plus que 1,3s et l'algorithme était alors dans une version très proche de celui décrit dans le Chapitre II. Le code, très simple, tenait alors en 1000 lignes dans seulement 34ko ! Cette version, bien que commentée, n'a jamais été distribuée faute de performance. Par exemple le système suivant appelé `cyclique` 5 prenait de l'ordre d'une heure :

$$\begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 = 0, \\ x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_1 = 0, \\ (x_1x_2)x_3 + (x_2x_3)x_4 + (x_3x_4)x_5 + (x_4x_5)x_1 + (x_5x_1)x_2 = 0, \\ ((x_1x_2)x_3)x_4 + ((x_2x_3)x_4)x_5 + ((x_3x_4)x_5)x_1 + ((x_4x_5)x_1)x_2 + ((x_5x_1)x_2)x_3 = 0, \\ (((x_1x_2)x_3)x_4)x_5 = 1. \end{cases}$$

En novembre 1998 nous avons donc fait le choix de changer de système de calcul formel. Après un tour d'horizon des systèmes disponibles nous avons retenu Magma pour ses très bonnes performances en matière d'arithmétique d'entiers et de polynômes. Une première version a été rapidement écrite : le code en Maple appelait, via des fichiers textes temporaires, les routines de remontée et d'intersection écrites en Magma. Nous pouvions alors résoudre très aisément 5 équations de degré 2 génériques. Forts de ce succès nous avons définitivement entériné le choix de Magma et réécrit la totalité de l'algorithme pour arriver ainsi à la première version 0.1 en juillet 1999. Une partie non négligeable de nos efforts a dû être portée à la programmation d'un domaine de calcul d'évaluation. Nous expliquons notre solution en V.2.4.

Voici quelques points de repères concernant l'évolution des performances de notre implantation :

version	0.1	0.16	0.166
`genpol` 5.2	44s	38s	29s
`cyclique` 5	84s	28s	20s

Le système `genpol` 5.2 est constitué de 5 équations génériques en 5 variables, développées avec des coefficients aléatoires entre 0 et 99.

V.1.2 Aspects Probabilistes

L'algorithme de résolution présenté dans les chapitres précédents est probabiliste. Rappelons que la probabilité de succès repose sur un choix de coordonnées affines suffisamment génériques. Nous avons montré que les mauvais choix sont enfermés dans des parties algébriques strictes mais n'avons pas fait d'effort suffisant pour estimer la taille de ces parties. Il n'est donc en rien immédiat d'imaginer si l'algorithme échoue de l'ordre d'une fois sur deux ou bien d'une fois sur un million. Pour notre code nous avons choisi de prendre des nombres premiers de la taille d'un mot machine (i.e., 32 bits). Cette heuristique n'a pas été étudiée, car je n'ai jamais observé d'échec lors d'une résolution qui ne soit pas due à ma propre inattention. De plus, pour ce qui est de Magma, choisir un plus petit nombre premier n'accélère pas les calculs.

Néanmoins il serait intéressant de mener une étude pratique de la taille de nombres premiers minimum commençant à causer des problèmes. D'un point de vue théorique nous disposons de tous les outils pour faire cette étude. Mais il va de soi qu'une telle étude de probabilité est fastidieuse et dépend très précisément des sous-algorithmes utilisés. Pour présenter un intérêt pratique il faudrait être capable de l'automatiser. Dans sa thèse [Sch00b] Schost propose des estimations de probabilité pour son algorithme de résolution de systèmes algébriques à paramètres. Les estimations donnent des résultats a priori réalistes mais bien pessimistes par rapport à ce que l'on observe.

D'un point de vue théorique nous saurions revenir à une version non-déforestée de notre algorithme afin d'exprimer nos résultats dans le modèle de complexité non-uniforme des travaux historiques [GHH+97]. Nous pourrions obtenir les degrés et complexités d'évaluation des hypersurfaces algébriques à éviter pour les choix des paramètres de l'algorithme et par conséquent déduire les probabilités de succès. Nous pourrions prouver que ces grandeurs sont polynomiales en la grandeur δ^a définie en §IV.4.3.

V.2 Algorithmique élémentaire

Cette section concerne les aspects algorithmiques élémentaires intervenant dans l'algorithme de résolution : calculs d'évaluation, algèbre linéaire, polynômes à une et deux variables et séries à plusieurs variables. Notre objectif et de discuter la complexité pratique de l'arithmétique de ces objets et d'en déduire ainsi la complexité pratique de l'algorithme complet sur la gamme des problèmes que la technologie actuelle nous permet de traiter.

V.2.1 Algèbre linéaire

Notre algorithme de résolution utilise de l'algèbre linéaire en dimension au plus le nombre de variables n du système à résoudre. Les principales nécessités sont l'inversion et le produit de matrices sur un anneau. Nous avons fixé depuis II.3.5 les constantes ω et Ω telles que le produit matriciel en dimension n effectue $\mathcal{O}(n^{\omega})$ opérations arithmétiques dans l'anneau de base (nous savons que $\omega < 2.376$ [CW90]) et le calcul du polynôme caractéristique et donc de la comatrice peut se faire en $\mathcal{O}(n^{\Omega})$. Nous savons que $\Omega \leq \omega + 1$ [Abd97, Ber84, Csa76, Lev40].

Cette algèbre linéaire est utilisée lors des phases de remontée pour inverser les matrices Jacobiennes intervenant dans l'opérateur de Newton ou l'opérateur basé sur le Chapitre III. Dans les deux cas la situation est la suivante : on travaille dans une algèbre $A = \mathbb{Z}/p\mathbb{Z}[[t]][T]/(q(T))$ où p est un petit nombre premier et q un polynôme de $\mathbb{Z}/p\mathbb{Z}[[t]][T]$ unitaire de degré δ. On veut inverser une matrice J à coefficients dans A telle que J est inversible dans $A/(t)$ d'inverse I_0. Le calcul de I_0 se fait en $\mathcal{O}(n^{\Omega})$ opérations dans $A/(t)$. Puis on construit la suite $(I_\kappa)_\kappa$:

$$I_{\kappa+1} := I_\kappa + (\mathrm{Id} - I_\kappa J(z_{\kappa+1}))I_\kappa, \quad \kappa \geq 0.$$

La matrice I_κ est alors l'inverse de J à la précision 2^κ en t. Pour calculer une courbe de remontée la matrice J doit être inversée jusqu'en précision $\delta+1$. Cela coûte essentiellement deux multiplications de matrices. En pratique nous sommes confrontés à la situation où $\deg(q) \gg n$, ce qui implique que le temps de calcul de I_0 est négligeable et que l'essentiel est passé dans les deux dernières multiplications. Nous souhaitons montrer qu'une multiplication rapide de matrices est bénéfique même pour des petites valeurs de n lorsque l'arithmétique de l'anneau de base est coûteuse.

Pour la table suivante nous avons choisi $p = 4294967311$, $\delta = 15$ et faisons varier n de 2 à 10. Nous affichons le rapport des logarithmes des temps de calcul de la multiplication de Strassen ($\omega_s \approx 2{,}8$ [Str69]) avec la multiplication naïve. Pour ce faire j'ai utilisé la fonction Strassen accessible dans la version en développement de Magma. La première ligne donne le temps t_s en secondes pour la multiplication de Strassen, la deuxième t_c pour la multiplication classique. Ensuite nous donnons $100\,t_s/t_c$ puis $\log(t_s)/\log(t_c)$. Le dernière ligne doit converger vers $2{,}8/3 \approx 0{,}93$.

n	2	3	4	5	6	7	8	9	10
t_s	0,30	1,08	2,21	4,77	7,88	13,53	16,12	26,24	38,15
t_c	0,32	1,11	2,79	5,26	9,03	14,74	22,59	34,29	54,67
$100t_s/t_c$	94%	97%	79%	90%	87%	91%	71%	76%	69%
$\log(t_s)/\log(t_c)$	1,04	0,72	0,77	0,94	0,94	0,96	0,89	0,92	0,91

V.2.2 Arithmétiques des polynômes

Nous considérons depuis §II.3.5 que la fonction $\mathcal{U}(\delta)$ représente la complexité de l'arithmétique des polynômes à une variable et fixons sa valeur à $\delta\log(\delta)^2\log(\log(\delta))$. Rappelons en la signification : la multiplication, la division, le résultant et le plus grand commun diviseur de polynômes de degré δ s'effectue en $\mathcal{O}(\mathcal{U}(\delta))$ opérations arithmétiques dans le corps des coefficients.

Pour fixer les idées prenons $k := \mathbb{Z}/p\mathbb{Z}$ comme corps de base et examinons une situation correspondant au calcul d'une courbe de remontée. Nous sommes dans un quotient

$$A := \Big(k[[t]]/(t^\delta)\Big)[T]/(q(T))$$

où q est un polynôme unitaire de degré δ. Si les domaines de polynômes et de séries sont implantés de façon générique polymorphe (ce qui est le cas en Magma jusque la version 2.7 incluse) alors une multiplication dans A requiert $\mathcal{O}(\mathcal{U}(\delta)^2)$ opérations dans k. L'estimation asymptotique commence à être généralement effective pour $\delta \gg 100$. En fait cette estimation est bien au contraire pessimiste en pratique. En effet l'empilement naïf des domaines de séries et de polynôme est la pire approche. En pratique on ramène les opérations de ces objets à deux variables à des objets à une variable en utilisant la substitution de Kronecker [Kro82] (celle-ci est détaillée dans les ouvrages généralistes tels que [GG99, BP94] : la multiplication de deux polynômes de $k[x,y]$ de degrés en x et y bornés par δ se ramène au produit de deux polynômes à une variable de degré $2\delta^2$. Cette dernière opération est alors dans $\mathcal{O}(\mathcal{U}(\delta^2))$ et réaliste dès lors que $\delta \gg 10$. Sous notre impulsion la substitution de Kronecker est depuis la version 2.8 implantée dans Magma pour ces problèmes à deux variables.

Les calculs de plus grand commun diviseurs et résultants interviennent à plusieurs endroits dans l'algorithme. L'étape significative est le calcul de l'intersection. À ma connaissance, il n'existe pas d'implantation d'algorithmes asymptotiquement optimaux qui soit efficace pour des degrés de l'ordre de 1000 pour des polynômes à coefficients dans $\mathbb{Z}/p\mathbb{Z}$. Magma ne dispose pas de tels algorithmes en C et les implantations que nous avons mis en œuvre ne se sont pas avérées rentables. Nous renvoyons à la thèse de Montgomery [Mon92] pour une discussion fine sur ce sujet. Dans la table suivante on donne les temps de la fonction `Gcd` de Magma 2.6 pour deux polynômes de degré δ, avec $p = 4294967311$.

δ	24	48	96	192	384	768	1536
Temps (s)	0,002	0,010	0,030	0,094	0,340	1,346	5,518

V.2.3 Séries à plusieurs variables

La manipulation de séries à plusieurs variables est cruciale pour l'algorithme de dégonflement du Chapitre III. Dans cette situation nous ne connaissons pas d'algorithme asymptotiquement rapide en général (c'est-à-dire linéaire dans la taille du support des séries à des facteurs logarithmiques près). La multiplication est alors quadratique. Ceci n'est pas très gênant car d'une part les situations avec multiplicité sont relativement rares et surtout lorsqu'elles apparaissent le produit des multiplicités intermédiaires, gouvernant la taille du support des séries ne dépasse pas la dizaine sur tous nos exemples. Il n'existe pas de séries à plusieurs variables en Magma, nous avons dû utiliser une algèbre associative construite à partir de la table de multiplication des monômes du support des séries. Grâce à la représentation creuse de l'algèbre le coût d'une multiplication est véritablement quadratique.

La remontée des variables libres pour obtenir une résolution géométrique dans une représentation dense développée en §IV.5.7 utilise des séries à plusieurs variables modulo une puissance de l'idéal maximal de l'anneau des séries. Dans cette situation nous proposons dans l'Annexe C un algorithme asymptotiquement rapide. Celui-ci n'est pas implanté. Pour l'instant nous nous contentons de remonter plusieurs courbes et de les interpoler avec la fonction d'interpolation de Magma. Cette réalisation n'est donc pas des plus efficaces.

V.2.4 Calculs d'évaluation

Magma ne dispose pas de calculs d'évaluation. Nous avons donc du créer une sorte de *domaine* pour représenter, manipuler et évaluer les polynômes d'entrée.

Puisque les calculs d'évaluation ne sont nécessaires que pour stocker les polynômes d'entrée afin de les évaluer il serait possible de représenter le système d'entrée par des fonctions ordinaires. Cette vue est parfaitement réaliste mais présente des inconvénients majeurs. Si l'on attribue une fonction par polynôme d'entrée il n'est plus possible (en Magma du moins) de partager des sous-expressions communes entre plusieurs polynômes. Si par contre on ne construit qu'une seule fonction pour évaluer tous les polynômes en même temps on sera inefficace lors des phases d'intersection où il ne faut évaluer qu'un seul polynôme. Quand bien même on admettrait un codage subtil de la fonction codant le système d'entrée il se pose le problème du calcul de la matrice Jacobienne. On se verrait alors forcé d'évaluer les polynômes d'entrée dans un anneau de la forme $k[x_1,\dots,x_n]/(x_1,\dots,x_n)^2$ afin de récupérer les valeurs des dérivées partielles comme coefficients des x_i. Cette approche n'est pas bonne si l'on souhaite évaluer le gradient d'une seule équation de complexité L : il nous en coûtera de l'ordre de nL opérations alors que l'algorithme de Baur-Strassen le fait en $4L$ opérations.

Implanter des calculs d'évaluation en Magma n'est pas chose aisée, car d'une part il n'est pas possible de créer un nouveau domaine et d'autre part il n'est pas possible non plus d'avoir des fonctions avec effets de bord (variables globales). La solution que nous avons retenue relève certes du bricolage mais a largement suffi à nos besoins. Nous avons choisi le type `Rec`, pour lequel les opérations arithmétiques élémentaires n'existent pas, comme type pour nos calculs d'évaluation. Chaque calcul d'évaluation est alors un enregistrement avec un champ pointant sur le domaine dit parent et un champ adresse mémorisant la position du nœud de calcul dans le calcul d'évaluation global attaché au domaine parent. Comme domaine parent nous avons choisi `RngMPol` (anneau de polynômes à plusieurs variables). Notre constructeur de domaine est donc le suivant :

```
intrinsic BlackboxPolynomialAlgebra(R,n::RngIntElt) -> RngMPol
{ R, a ring or a field.
  n, a positive integer.

  It returns a ring of multivariate polynomials over R in n variables with
  a new attribute called Dag in which the expressions are stored.}

  A:=recformat <
   Ops,          // Set of the expressions
```

```
   Variables,     // Sequence of the variables
   Constants,     // Sequence of the constants
   Remember,      // Set of integers
   Values >;      // Sequence of values
 AddAttribute(RngMPol,"Dag");
 MP:=PolynomialAlgebra(R,n);
 MP`Dag:=rec<A|
   Ops:={@ <"Cst",[1]>,<"Cst",[2]> @},
   Variables:=[MP.i: i in [1..n]],
   Constants:= [R|0,1],
   Remember:=[], Values:=[]>;
 return(MP);
end intrinsic;
```

La fonction `Var` suivante sert de constructeur : elle retourne le nœud de calcul correspondant à sa variable en argument.

```
intrinsic Var(x::RngMPolElt) -> Rec
{ x is an element of a multivariate polynomial ring created by
  the intrinsic BlackboxPolynomialAlgebra.

  It returns a blackbox polynomial of recformat<Parent,Address>, where
  Parent is Parent(x), and Address is the position of the variable x
  in the Dag of its parent.}

  e:=<"Var",[Position(Parent(x)`Dag`Variables,x)]>;
  R:=Parent(x);
  return(IncludeInDag(R,e));
end intrinsic;
```

La fonction `IncludeInDag` est chargée d'inclure un nouveau nœud dans le `Dag` courant. Si ce nœud existe déjà elle ne fait que retourner son adresse. Voyons comment cela fonctionne sur un exemple :

```
> AttachSpec("spec");
> MP<x1,x2,x3>:=BlackboxPolynomialAlgebra(Rationals(),3);
> x1:=Var(x1); x2:=Var(x2); x3:=Var(x3);
> MP`Dag;
rec<recformat<Ops, Variables, Constants, Remember, Values> |
 Ops := {@ <Cst, [ 1 ]>, <Cst, [ 2 ]>, <Var, [ 1 ]>, <Var, [ 2 ]>,
          <Var, [ 3 ]> @},
 Variables := [ x1, x2, x3 ],
 Constants := [ 0, 1 ],
 Remember := [], Values := []>

> f:=x1*x2;
> f;
rec<recformat<Parent, Address> |
```

```
 Parent := Polynomial ring of rank 3 over Rational Field
          Lexicographical Order Variables: x1, x2, x3,
 Address := 6>

> MP‘Dag;
rec<recformat<Ops, Variables, Constants, Remember, Values> |
 Ops := {@ <Cst, [ 1 ]>, <Cst, [ 2 ]>, <Var, [ 1 ]>, <Var, [ 2 ]>,
          <Var, [ 3 ]>, <*, [ 3, 4 ]> @},
 Variables := [ x1, x2, x3 ],
 Constants := [ 0, 1 ],
 Remember := [], Values := []>
```

Les champs `Remember` et `Values` servent à stocker les résultats intermédiaires lors d'un processus d'évaluation afin d'éviter qu'une sous expression soit évaluée plusieurs fois. Pour les calculs de gradients nous utilisons l'algorithme de Baur-Strassen.

```
> InitializeEvaluation(MP,[Factorial(2000),Factorial(10001),0]);
> time tmp:=Evaluate(f);
Time: 0.140
> time tmp:=Evaluate(f);
Time: 0.000
> MP‘Dag‘Remember;
[ false, false, true, true, true, true ]
```

Il faut observer que cette implantation est ce qu'il y a de plus rudimentaire : la construction des polynômes d'entrée peut devenir relativement coûteuse sur des systèmes de plusieurs centaines de lignes et les calculs de gradients prohibitifs. Il semble naturel qu'un tel domaine se trouve dans le noyau de Magma, écrit en C. Ce travail est en cours par Allan Steel. Jusqu'en version 0.16 le mécanisme d'évaluation ne comportait aucun système à jetons (*pebble game*) pour économiser la place mémoire. La version en cours possède désormais cette fonctionnalité, ce qui rend possible la résolution de systèmes dans un espace mémoire plus réduit. Cette nouvelle implantation du domaine de calculs d'évaluation est due à Lutz Lehmann (université de Humbold, Berlin, Allemagne), motivé par des calculs de bases d'ondelettes servant à la compression d'image.

V.3 Opérateur de Newton avancé

L'opérateur de Newton que nous utilisons intensivement dans l'algorithme de résolution est sujet à de nombreuses optimisations. Nous présentons ici l'une d'entre elles, originale, qui n'améliore pas la complexité théorique mais qui s'avère intéressante en pratique lorsque la complexité d'évaluation du système d'entrée est faible.

Revenons dans le cadre de §II.4 : A est un anneau commutatif intègre et $\mathfrak{m}$ l'un de ses idéaux maximaux. Pour simplifier la présentation, on suppose que la topologie induite par $\mathfrak{m}$ sur A est séparée et complète. On considère $F := (f_1, \ldots, f_n)$ un vecteur de n fonctions polynomiales de $A[x_1, \ldots, x_n]$ données par un calcul d'évaluation de complexité L. On souhaite calculer à précision arbitraire une racine simple donnée par son approximation

z_0 modulo $\mathfrak{m}$. On note J la matrice Jacobienne de F. Il nous faut supposer que $J(z_0)$ est inversible dans $A/(\mathfrak{m}A)$: on note I_0 cet inverse. Pour les mesures de complexité $\mathsf{a}(\kappa)$ représente la complexité d'une opération arithmétique élémentaire dans $A/(\mathfrak{m}^{2^\kappa}A)$.

Rappelons l'opérateur de Newton classique : la suite définie par

$$z_{\kappa+1} = z_\kappa - J(z_\kappa)^{-1}F(z_\kappa), \quad \kappa \geq 0$$

converge quadratiquement vers une racine x^* de F de sorte que :

$$\begin{aligned} &z_\kappa - x^* \in \mathfrak{m}^{2^\kappa} \times \cdots \times \mathfrak{m}^{2^\kappa}, \\ &z_\kappa - z_{\kappa+1} \in \mathfrak{m}^{2^\kappa} \times \cdots \times \mathfrak{m}^{2^\kappa}, \\ &F(z_\kappa) \in \mathfrak{m}^{2^\kappa} \times \cdots \times \mathfrak{m}^{2^\kappa}. \end{aligned}$$

La première remarque importante classique est que la connaissance de $J(z_\kappa)^{-1}$ à la précision $\mathfrak{m}^{2^\kappa}$ est suffisante pour calculer $z_{\kappa+1}$. La deuxième remarque tout aussi classique est qu'il ne faut pas calculer $J(z_\kappa)^{-1}$ directement mais utiliser un opérateur de Newton à cette fin :

$$I_{\kappa+1} = I_\kappa + (\mathrm{Id} - I_\kappa J(z_{\kappa+1}))I_\kappa, \quad \kappa \geq 0.$$

La matrice I_κ est ainsi l'inverse de $J(z_\kappa)$ à la précision $\mathfrak{m}^{2^\kappa}$.

L'opérateur de Newton devient ainsi la suite double

$$\begin{aligned} &I_\kappa = I_{\kappa-1} + (\mathrm{Id} - I_{\kappa-1}J(z_\kappa))I_{\kappa-1}, \quad \kappa > 0, \\ &z_{\kappa+1} = z_\kappa - I_\kappa F(z_\kappa), \quad \kappa \geq 0. \end{aligned}$$

Le coût $\mathcal{C}_1(\kappa)$ pour calculer $z_{\kappa+1}$ à partir de z_κ est alors

$$\mathcal{C}_1(\kappa) \approx (nL + 2n^\omega + n^2 + n)\mathsf{a}(\kappa) + (L + n^2 + n)\mathsf{a}(\kappa+1).$$

En effet l'évaluation de la matrice Jacobienne coûte nL et le produit de matrices n^ω. Si $A = k[[t]]$ et $\mathfrak{m} = (t)$, alors $\mathsf{a}(\kappa+1)$ est proche de $2\mathsf{a}(\kappa)$. Par conséquent si $n \gg 2$ et $L \ll n^2$ le terme principal dans $\mathcal{C}_1$ est $2n^\omega\mathsf{a}(\kappa)$. Nous proposons une astuce pour amoindrir le coût de la mise à jour de l'inverse de la matrice Jacobienne. Cette astuce m'a été proposée par Bruno Salvy. Elle est moins classique mais très similaire à celle utilisée pour gagner une demie multiplication dans le calcul de la division euclidienne par la méthode du polynôme réciproque [Sie72].

L'idée clef est de retarder l'inversion de la matrice Jacobienne de la façon suivante :

$$\begin{aligned} &I_{\kappa-1} = I_{\kappa-2} + (\mathrm{Id} - I_{\kappa-2}J(z_{\kappa-1}))I_{\kappa-2}, \quad \kappa > 1, \\ &z_{\kappa+1} = z_\kappa - 2(I_{\kappa-1}F(z_\kappa)) + I_{\kappa-1}J(z_\kappa)(I_{\kappa-1}F(z_\kappa)), \quad \kappa \geq 0. \end{aligned}$$

Le coût $\mathcal{C}_1(\kappa)$ pour le calcul $z_{\kappa+1}$ à partir de z_κ est alors de l'ordre de

$$\mathcal{C}_2(\kappa) \approx nL\mathsf{a}(\kappa) + (2n^\omega + n^2 + n)\mathsf{a}(\kappa-1) + (L + 3n^2 + 2n)\mathsf{a}(\kappa+1).$$

Cette astuce nous apporte des gains sensibles dans notre programme. On pourrait penser à itérer le procédé et retarder d'avantage la mise à jour de l'inverse de la matrice Jacobienne. Notre expérience n'a pas fléchi dans cette direction.

V.4 Accélérations liées à la factorisation

La factorisation de polynômes à une variable à coefficients dans $k := \mathbb{Z}/p\mathbb{Z}$ est une opération désormais courante dans tous les systèmes de calcul formel et particulièrement efficace en Magma. Nous nous en référons aux ouvrages de synthèse tels que [GG99, BP94, GCL94]. Nous montrons comment nous avons accéléré dans notre implantation l'opérateur de Newton et l'algorithme d'intersection vu en §II.6. Les résultats de complexité que nous obtenons n'améliorent pas ceux des chapitres précédents dans le pire des cas mais les probabilités de succès des algorithmes ne sont pas détériorées pour autant : nous n'utilisons pas le théorème d'irréductibilité de Hilbert [Hil92, DST93]. Par ailleurs, puisque nous nous focalisons sur $k := \mathbb{Z}/p\mathbb{Z}$ reposer nos probabilités de succès sur le théorème d'irréductibilité de Hilbert sur $\mathbb{Q}$ puis penser trouver un nombre premier convenable n'est pas réaliste. Cette conclusion de bon sens est manifeste en pratique. C'est ainsi que nous abordons naturellement dans la section suivante un algorithme de calcul de décomposition en composantes irréductibles.

V.4.1 Remontées en parallèle

Soit F une fibre de remontée codant une variété r-équidimensionnelle $\mathcal{W}$: M représente la matrice de changement de variables définissant une position de Noether projective pour $\mathcal{W}$, $y_1, \ldots ,y_n$ les variables en position de Noether, p le point de remontée, u est l'élément primitif, q son polynôme minimal et la paramétrisation $y_{r+1} = v_{r+1}, \ldots ,y_n = v_n$. On considère la factorisation en facteurs irréductibles de $q := q_1 \cdots q_s$, où les q_i sont des polynômes irréductibles de $k[T]$. On définit la *factorisation de la fibre* F comme l'ensemble les fibres F_i construites comme suit : F_i est constituée de M, p, u, q_i, $v_{r+1}, \ldots ,v_n$ mod q_i.

Étudions comment se comporte le calcul de remontée d'une courbe vis-à-vis de la factorisation. Plaçons nous dans le cadre simple où $\mathcal{W}$ ne présente pas de multiplicité par rapport à son système de remontée. Nous notons toujours L la complexité d'évaluation du système d'entrée. En nombre d'opérations dans k, la complexité du calcul de la courbe de remontée (Proposition 9) est dans

$$\mathcal{O}\Big((nL + n^{\Omega})\mathcal{U}(\deg(\mathcal{W}))^2\Big).$$

Effectuons maintenant les remontées des courbes en parallèle sur chacun des facteurs q_i et recombinons les résultats pour retrouver la paramétrisation de la courbe de remontée. Plus précisément, pour chaque $i = 1, \ldots ,s$, l'opérateur de Newton calcule

$$Q_i(t,T) = 0,\ y_j = V_{i,j}(t,T),\ j = r+1, \ldots ,n,$$

tels que $Q_i = q_i$ mod t. Le polynôme Q_i est de degré $\deg(q_i)$ en T et les $V_{i,j}$ sont réduits par rapport à Q_i. La précision des séries en t est $\mathcal{O}(t^{\deg(\mathcal{W})+1})$. Précisons qu'il est en général faux que $Q_i(t,T) \in k[t,T]$. Le coût total de ce calcul est dans

$$\mathcal{O}\Big((nL + n^{\Omega})\mathcal{U}(\deg(\mathcal{W}))\sum_{i=1}^{s}\mathcal{U}(\deg(q_i))\Big).$$

Pour retrouver la paramétrisation algébrique

$$Q(t,u) = 0, \quad \frac{\partial Q}{\partial T}(t,u) y_j = W_j(t,U),\ j = r+1,\ldots,n$$

de la courbe de remontée il faut recombiner les paramétrisations de la façon suivante. Tout d'abord on obtient $Q = Q_1 \cdots Q_s$. Ensuite on calcule les $W_{i,j} = \frac{\partial Q_i}{\partial T} V_{i,j}$ modulo Q_i et les $\hat{Q}_i = Q/Q_i$ pour $i = 1,\ldots,s$. Enfin W_j est donné par la formule d'interpolation

$$W_j = W_{1,j}\hat{Q}_1 + \cdots + W_{s,j}\hat{Q}_s.$$

Il s'agit en fait d'un problème de restes chinois pour lequel il existe des algorithmes rapides (cf. les ouvrages généralistes [BP94, p. 27] et [GG99, p. 289]). Ce calcul de recombinaison peut se faire en

$$\mathcal{O}((n-r)\mathcal{U}(\deg(Q))^2).$$

Examinons le cas où $\mathcal{U}$ est essentiellement linéaire : les remontées en parallèle gagnent de facteurs logarithmiques seulement. En revanche si l'on a $\mathcal{U}(\delta) = \delta^2$ (ce qui est réaliste lorsque δ est petit) alors le calcul des remontées en parallèle est nettement plus intéressant.

Cette courte étude montre que l'on ne perd pas en complexité dans le pire des cas en utilisant la factorisation. D'un point de vue pratique faire les remontées en parallèle est toujours intéressant à condition que le coût de recombinaison ne devienne pas prohibitif. Pour mon implantation j'ai adopté la stratégie suivante : les facteurs plus petits qu'un certain degré sont regroupés.

V.4.2 Intersection avec factorisation

Nous proposons une alternative à la méthode d'intersection basée sur l'argument de déformation de l'élément primitif telle que présentée en §II.6. Cette méthode amoindrit la dépendance en n (nombre de variables) et présente l'avantage de ne dépendre que du degré de l'intersection et non de la prévision issue du théorème de Bézout.

Du point de vue calculatoire la situation est la suivante. On dispose d'une paramétrisation d'une courbe de remontée de degré D sous la forme

$$q(t,u) = 0, \quad \frac{\partial q}{\partial T}(t,u) y_i = w_i(t,u),\ i = 1,\ldots,n,$$

où q est un polynôme de $k[t,T]$ de degré total D et unitaire en T, les W_i ont aussi un degré total borné par D et ont un degré en T strictement inférieur à D. On souhaite calculer une paramétrisation de l'ensemble des points d'intersection de cette courbe par l'hypersurface $\mathcal{V}(f)$, où f est une fonction polynomiale de degré d intersectant proprement la courbe et donnée par un calcul d'évaluation de complexité au plus L. Ces points d'intersection peuvent être décrits par

$$Q(t) = 0, \quad \frac{\partial Q}{\partial t} y_i = W_i(t),\ i = 1,\ldots,n,$$

avec Q de degré borné par dD et W_i de degré borné par $\deg(Q)$. On suppose que t sépare les points de cette intersection, le degré de Q est alors exactement le nombre de points de l'intersection.

Parmi les méthodes que nous avons implantées, nous proposons celle qui offre les meilleures performances en Magma pour la gamme des problèmes que nous pouvons traiter à l'heure actuelle. Nous calculons Q en premier lieu, nous le factorisons, puis déduisons les W_i.

Calcul de Q

On prend S un ensemble de $D+1$ points de k et on calcule

$$\{(q(a,T),w_1(a,T),\dots,w_n(a,T)),\ a\in S\}$$

en $\mathcal{O}(nD\mathcal{U}(D))$ opérations arithmétiques dans k en utilisant des algorithmes d'évaluation rapide multipoint [AHU74, Fid72, Fid87]. Pour chaque $a\in S$ tel que $\frac{\partial q}{\partial T}(a,T)$ est inversible modulo q nous pouvons calculer

$$y_1=v_1(a,T),\dots,y_n=v(a,T),$$

puis déduire $Q(a)$ comme le résultant en T de $f(v_1(a,T),\dots,v(a,T))$ avec $q(a,T)$. L'inversion de $\frac{\partial q}{\partial T}(a,T)$ coûte $\mathcal{U}(D)$, l'évaluation de f modulo Q se fait en $\mathcal{O}(L\mathcal{U}(D))$ puis le résultant en $\mathcal{O}(\mathcal{U}(D))$. Il suffit de répéter cette opération pour tous les éléments de S puis de choisir d'autres ensembles de points S deux à deux disjoints jusqu'à l'obtention de Q par interpolation.

Les mauvais points sont contenus dans le discriminant de q, leur nombre ne peut donc dépasser D^2. Le nombre de points à parcourir au total est au plus $D^2+\deg(Q)+2$, il faut donc

$$\left\lceil\frac{D^2+\deg(Q)+2}{D+1}\right\rceil$$

ensembles S. Pour l'interpolation on utilise une méthode incrémentale naïve, il nous en coûte un total de $\mathcal{O}(\deg(Q)^2)$.

Calcul de la paramétrisation

Factorisons Q en facteurs irréductibles $Q_1,\dots,Q_r$. Pour chaque facteur Q_l nous calculons ensuite :

1. Dans $k(a_l)=k[T]/(Q_l)$ nous évaluons q et les w_i en $t=a_l$. Il ne s'agit pas vraiment d'une évaluation, il suffit de calculer le reste de chaque coefficient dans la division euclidienne par Q_l. Il en coûte
$$\mathcal{O}(nD\mathcal{U}(D)).$$
2. On cherche ensuite une paramétrisation des points de l'intersection :
   - $M:=\gcd(q(a_l,T),\frac{\partial q}{\partial T}(a_l,T))$.
   - $\overline{q}:=q/M$, $\overline{q}$ est sans carré.
   - $\overline{p}:=\frac{\partial q}{\partial T}(a_l,T)/M$.
   - $\overline{v}_i:=(w_i/M)(\overline{p}^{-1}\bmod\overline{q})$, la division par M est exacte d'après II.6.5.

Ces opérations coûtent $\mathcal{O}(n\mathcal{U}(\deg(Q_l))\mathcal{U}(D))$.

3. Enfin gcd $\Big(f(\overline{v}_1,\dots,\overline{v}_n),\overline{q}\Big)$ donne un facteur linéaire $T-\overline{V}_l$ avec une complexité dans
$$\mathcal{O}\Big(L\mathcal{U}(\deg(Q_l))\mathcal{U}(D)\Big).$$
Cette phase est sans aucun doute la plus coûteuse.
4. Il ne reste plus qu'à calculer $\overline{v}_i(\overline{V}_l)$ qui est la valeur de la coordonnée y_i modulo Q_l.

Les W_i sont récupérés par recombinaison via les restes chinois (formules d'interpolation).

V.4.3 Optimisation d'une base d'ondelettes

Avec É. Schost, nous avons résolu des systèmes posés par S. Mallat (Laboratoire CMAT, École polytechnique, France) dont les solutions servent à construire des bases d'ondelettes répondant à certains critères d'optimalité. L'un des problèmes comporte quatre équations à variables qk_0,x_1,x_2,x_3, à coefficients dans $\mathbb{Q}$ et l'inéquation $x_1x_2x_3 \neq 0$. Le problème comporte 92 solutions isolées simples. Moitié de temps de calcul de Kronecker est consacré à la résolution modulo le nombre premier 473425343, l'autre moitié pour remonter les entiers. Les calculs avec FGb sont effectués à https://calfor.lip6.fr/. Pour les calculs de bases de Gröbner nous avons introduit une nouvelle variable z et l'équation $zx_1x_2x_3=1$. Le calcul de base *lex* avec FGb échoue après 17 minutes.

Kronecker	Athlon 1GHz	7min
Magma *grevlex*	Athlon 1GHz	> 1h30min
FGb *grevlex*	Pentium II 233MHz	5min
FGb *lex*	Pentium II 233MHz	> 17 min

V.5 Vers une décomposition en irréductibles

Nous présentons quelques pistes de recherche pour calculer la décomposition en composantes irréductibles d'une variété algébrique. Nous abordons en premier lieu le problème de la factorisation d'une courbe de remontée modulo un nombre premier p. Sur l'exemple du système `cyclique` 9 nous expliquons un moyen efficace de recombiner les solutions isolées factorisées modulo p afin de remonter la décomposition irréductible sur $\mathbb{Q}$.

V.5.1 Factorisation d'une courbe de remontée

Nous nous plaçons dans le cas où le corps de base k est $\mathbb{Z}/p\mathbb{Z}$, avec p un nombre premier. Dans le cadre de II.4.5 nous souhaitons calculer une courbe de remontée d'une variété $\mathcal{W}$ equidimensionnelle de dimension r à partir d'une fibre de remontée. Le morphisme π est la projection de $\mathcal{W}$ sur l'espace k^r des variables libres, u est une forme linéaire séparant les points de la fibre, q le polynôme minimal et w les paramétrisations de Kronecker :
$$q(u)=0, \quad q'(u)y_j = w_j(u).$$

Considérons la factorisation en irréductibles de q :

$$q = \prod_{i=1}^{s} q_i.$$

À partir des composantes irréductibles de la fibre nous souhaitons calculer les composantes irréductibles d'une courbe associée. Celle-ci est paramétrée par

$$\hat{Q}(t,u) = 0, \quad \frac{\partial \hat{Q}}{\partial T}(t,u) y_j = \hat{W}_j(t,u),$$

où $\hat{Q}$ est les $\hat{W}_j$ sont des polynômes de $k[t,T]$ de degré au plus $\deg(q)$ tels que $q = \hat{Q}(0,u)$ et $w_j = \hat{W}(0,u)$. Considérons la factorisation en irréductibles de $\hat{Q}$:

$$\hat{Q} = \prod_{i=1}^{S} \hat{Q}_i.$$

En général $S < s$ et le problème consiste à trouver la partition $\alpha_1, \ldots, \alpha_S$ de l'ensemble $\{1, \ldots, s\}$ telle que

$$\hat{Q}_i(0,T) = \prod_{j \in \alpha_i} q_j(T),$$

pour chaque $i = 1, \ldots, S$. Nous proposons deux méthodes différentes. Dans un premier temps nous remontons à la précision $\mathcal{O}(t^3)$ chaque composante irréductible de la fibre et explorons toutes les possibilités de recombinaison. Dans un deuxième temps nous remontons une seule composante irréductible de la fibre et calculons la composante irréductible de la courbe à laquelle elle appartient.

Recombinaisons

La situation et les notations sont identiques à §V.4.1 mais la phase de calcul des α_i se contente de calculer les $Q_i(t,T)$ dans $k[[t]]/(t^3)$, pour $i = 1, \ldots, s$. Ensuite, nous utilisons une technique proposée par Rupprecht dans [Rup01] (voir aussi [GR01]). Le Théorème 2.9 de [Rup01] ramène le problème au calcul des plus petits ensembles α_i tels que $\prod_{j \in \alpha_i} Q_j(T)$ soit de la forme $T^D + (a_0 + a_1 t + \mathcal{O}(t^3))T^{D-1} + \cdots$. Cette méthode est valable sur $\mathbb{C}$ avec des coordonnées génériques.

Dans le pire des cas cette méthode est exponentielle en le nombre de facteurs S. Son grand mérite est d'être immédiate à programmer. Notons qu'en moyenne le comportement est polynomial puisque le nombre de facteurs est logarithmique [Knu73]. Dans la gamme des systèmes dont nous disposons cette phase combinatoire est très rapide.

Un algorithme polynomial

Une autre idée pour factoriser une courbe de remontée est de partir d'un facteur irréductible q_l de la fibre et de chercher à quelle composante $\hat{Q}_L$ il correspond. À cette fin on calcule un développement en série aux points de la fibre correspondant à q_l dans $k[T]/(q_l)[[t]]$, on pose $K := k[T]/(q_l)$:

$$y_{r+1} = v_{r+1}(t), \ldots, y_n = v_n(t),$$

où les v_i sont dans $K[[t]]/(t^{N^2})$ avec N un entier fixé. Nous commençons par calculer $U := u(v_{r+1},\dots,v_n(t))$ et cherchons des polynômes $a_i(t)$ de degré au plus $N-1$ tels que

$$U^N + \sum_{i=0}^{N-1} a_i(t)U^i \in \mathcal{O}(t^{N^2}).$$

On obtient ainsi un système linéaire avec N^2 inconnues (les coefficients de a_i) et N^2 équations. Notons P le polynôme $X^N + a_{N-1}X^{N-1} + \cdots + a_0$. Pour un tel polynôme solution nous calculons $R(t,X)$ comme le résultant de P et q_l par rapport à T. La plus petite solution P lorsque N dépasse $\deg(\hat{Q}_L)$ conduit à $R(t,X) = \hat{Q}_L$.

La recherche de P correspond à ce que l'on appelle à problème d'approximant algébrique. Bienqu'il existe des méthodes plus rapides que la résolution naïve du système linéaire : approximants de Padé-Hermite [BL94], ou algèbre linéaire structurée [BP94, Chapter 2] il n'en existe pas d'asymptotiquement optimales (à des facteurs logarithmiques près). Nous n'avons pas encore programmé cette approche.

V.5.2 Résolution des systèmes `cyclique` n

Les systèmes `cyclique` n forment très certainement la famille la plus célèbre de systèmes polynomiaux pour effectuer des bancs d'essais entre logiciels. Le problème a été introduit dans la communauté de calcul formel par Davenport [Dav86]. Le problème est issu d'une application impliquant des transformées de Fourier [Bjö90, BF91]. Fröberg conjecture (conjecture rapportée dans [Möl98]) que lorsque n est divisible par un carré alors le nombre de solutions est infini et qu'en cas d'un nombre fini de solutions ce nombre est $(2n-2)!/(n-1)!^2$. Haagerup [Haa97] montre que lorsque n est premier le nombre de solutions est toujours fini et confirme le nombre de solutions de la conjecture de Fröberg.

Voici comment le système se construit. Les variables sont $x_1,\dots,x_n$. On note M_i le monôme $x_1 \cdots x_i$ et σ le cycle $(1,2,\dots,n)$ du groupe de permutation à n éléments. La $i$ème équation f_i du $n$ème système `cyclique` est

$$f_i = \sum_{k=0}^{n-1} \sigma^k(M_i) \quad \text{for} \quad 1 \leq i \leq n-1, \qquad f_n = M_n - 1.$$

Le programme de Verschelde [Ver99] calcule numériquement toutes les solutions de ces systèmes jusque $n = 11$. Les composantes de dimensions positives isolées sont représentées par des points génériques. Lorsque $n = 11$ le système a 184 756 racines isolées. Cela ne nécessite que quelques heures [SVW01b]. Du côté formel Magma est le seul logiciel distribué à pouvoir calculer une base *grevlex* de `cyclique` 8. Seul le logiciel `FGb` est capable de traiter `cyclique` 8 en quelques dizaines de minutes, `cyclique` 9 en quelques heures [Fau00] et `cyclique` 10 [Fau01]. Pour ma part Kronecker ne peut traiter que `cyclique` 8 en quelques heures sans factorisation, en quelques dizaines de minutes avec factorisation et `cyclique` 9 avec factorisation en quelques jours.

Voici quelques détails concernant la résolution de `cyclique` 9 avec Kronecker en utilisant la factorisation par recombinaisons modulo p expliquée précédemment. Tout d'abord nous éliminons l'équation linéaire manuellement. Nous donnons alors au programme les

8 équations en 8 variables restantes et aussi l'inéquation $x_2x_3x_4x_5x_6x_7x_8x_9(x_2+x_3+x_4+x_5+x_6+x_7+x_8+x_9) \neq 0$. Les calculs ont été faits sur un Athlon à 1GHz de l'UMS MEDICIS. La résolution modulo le nombre premier $p = 141833207$ prend 94h. Moitié du temps est passé dans le calcul d'une courbe de remontée en degré 558. La factorisation ne s'opère véritablement qu'en codimension 1. Une grande partie du temps restant est consacrée aux tests d'inclusion. Finalement on trouve un degré géométrique de 6174 dont une composante de dimension 2 et degré 18 et des points isolés.

Une fois la décomposition en composantes irréductible calculée modulo le nombre premier p, il nous faut retrouver les appariements de composantes sur $\mathbb{Q}$. Nous abordons le cas des points isolés solutions (dimension 0) uniquement, la composante de dimension 2 est petite et ne pose pas de problème malgré sa multiplicité.

Sur un tel exemple les méthodes combinatoires ne sont plus praticables et une approche classique par LLL [LLL82] est trop coûteuse. Notre solution est venue de van Hoeij [Hoe00] : le problème de recombinaison est ramené à un problème de sac à dos sur les traces des facteurs modulo p. Ce dernier est résolu par LLL, la taille du réseau est le nombre de composantes irréductibles modulo p. Il faut noter que, contrairement à la situation considérée par van Hoeij, notre polynôme à factoriser n'est pas unitaire mais surtout nous ne connaissons pas le coefficient de tête. Il faut alors introduire cette indéterminée dans le réseau, ce qui fait croître sensiblement la taille des entiers p-adiques.

Il faut 7h pour remonter les entiers modulo p^{512} et encore quelques heures à LLL pour trouver les bonnes recombinaisons. La plus grosse composante irréductible est de degré 1944.

Annexe A

The Projective Noether Maple Package: Computing the Dimension of a Projective Variety

Recent theoretical advances in elimination theory use straight-line programs as a data-structure to represent multivariate polynomials. We present here the *Projective Noether Package* which is a Maple implementation of one of these new algorithms, yielding as a byproduct a computation of the dimension of a projective variety. Comparative results on benchmarks for time and space of several families of multivariate polynomial equation systems are given and we point out both weaknesses and advantages of different approaches.

Parts of this chapter have been published in [GHL+00].

A.1 Introduction

Classical methods to study and solve systems of polynomial equations are based on numerous avatars of Gröbner (standard) basis algorithms or Riquier-Janet type methods (Ritt-Wu's algorithm). All these methods use implicitly but deeply the dense or sparse representation of multivariate polynomials, which is the computer science counterpart of the expansion of these mathematical objects on the monomial basis of the polynomial algebra. Also, all these methods can be interpreted as *rewriting* techniques.

Considerable efforts have been made in order to improve both theoretical and practical aspects of these techniques and to produce efficient algorithms and implementations. Restricting to this last aspect, all most commonly available computer algebra systems offer Gröbner basis implementations.

The knowledge of a standard basis yields as a simple byproduct the dimension of the algebraic variety defined by such systems. Actually, one can show that focusing on the simpler problem of computing the dimension of a projective algebraic variety will lead to a better worst-case complexity than the whole construction of a standard basis [Giu88]. Considering the *unit cost measure* model, i.e., each arithmetic operation of the ground field is counted as one, this complexity is polynomial in the size of the intermediate expressions computed.

Having in mind the problem of determining the dimension, it seemed at that time that there was no hope to design an algorithm whose complexity is *polynomial* in the size of the *input*, since the intermediate computations are not. But this observation is only valid if we stay stuck in the dense representation context. A breakthrough was obtained by [GH93], resulting in the existence of an algorithm with polynomial behaviour w.r.t. the dense measure of the input, provided one uses a mixed representation for intermediate computations and output.

Actually the algorithm described in *loc. cit.* computes more, i.e., a change of coordinates putting the new variables in *Noether position*. Informally speaking, the variables are then separated into two subsets of different nature: the independent and dependent ones. The number of independent variables is the dimension. The key point is the introduction of a mixed data structure to represent the polynomials occurring in intermediate computations. While we use the dense representation w.r.t. the dependent variables, their coefficients (which are polynomials in the independent ones) are coded by *arithmetic circuits*, also called *straight-line programs*. This means that these latter polynomials are represented by programs evaluating them at a point (of the ground field) using only additions and multiplications (of the ground field). We will conveniently refer to this latter representation as the *evaluation data structure*.

Mixing these two data structures was successfully used in a series of theoretical papers to design a new geometric elimination algorithm (see the joint works by Giusti, Hägele, Heintz, Montaña, Morais, Morgenstern, Pardo, 1995–97 at http://tera.medicis.polytechnique.fr). An efficient implementation of the complete elimination algorithm will require some time to collect more practical experience with the first experimental prototypes. Some basic but important steps were made already, but there is still a lot of work left (see the works by Aldaz, Castaño, Hägele, Lecerf, Llovet, Martínez, Matera within the Tera project *loc. cit.*).

We present here modestly a Maple program called the *Projective Noether Package* derived from the algorithm of [GH93]. The package and its documentation are available at http://tera.medicis.polytechnique.fr/tera/soft.html. It turns out that a not so well-known functionality of Maple is the systematic use of the evaluation data structure. Consequently we can compare the more traditional algorithms already available within Maple with the new ones, experimenting with several possible strategies. Comparative results on benchmarks for time and space of several different families of multivariate polynomial equation systems are given and we point out both advantages and weaknesses of the different approaches. One of the encouraging results is provided by an example (see Section A.4) where our Maple implementation computes an upper bound for the dimension more than fifty times as fast as the available version of Faugère's Gb system [Fau95]. However, Gb computes much more, i.e., a full Gröbner basis, from which an upper bound on the dimension can be extracted.

This chapter is organized as follows. In Section A.2, we recall the main definitions and results concerning straight-line programs and Noether position and we give the theoretical algorithm from [GH93]. Section A.3 shows how straight-line programs can be handled in practice, first by exploiting the directed acyclic graphs which are fundamental to Maple, then by appealing to a mechanism called *deforestation* from theoretical computer science. In Section A.3.4, we use these techniques to cast the theoretical algorithm in a different form on which the implementation is based. It turns out that the theoretical bounds which lead to a polynomial complexity of the algorithm are much too large to be of practical use. Therefore when we compare our implementation with a Gröbner basis package, we distinguish two subtasks: computing an upper bound on the dimension and proving that the bound is reached. Our implementation is very efficient for the former task, while we can only provide a heuristic answer to the latter one, in so far as a straight-line program representing a multivariate polynomial has been evaluated to 0 at a settable number of points but has not been proved to be identically 0.

A.2 Evaluation Data Structures and Deterministic versus Probabilistic Algorithms

A.2.1 Directed Acyclic Graphs and Straight-Line Programs

Let k be an infinite effective field; this means that the arithmetic operations (addition, subtraction, multiplication, division) and basic equality checking (comparison) between elements of k are realizable by algorithms. Let $x_1, \ldots, x_n$ be indeterminates over k. A polynomial of $k[x_1, \ldots, x_n]$ is usually coded as an expanded sum of monomials and each operation on such polynomials is related to operations on the vector of their coefficients. Instead, we use multivariate polynomials represented by **straight-line programs** that compute values at points of k^n.

All algorithms below will be represented by **arithmetic networks** over k i.e., directed acyclic graphs (DAGs) whose internal nodes are labelled by arithmetic operations of k, by Boolean operations corresponding to propositional logic, and by selectors associated with equality checking of elements of k. The external nodes of the graph represent the inputs and the outputs of the network. The inputs are always elements of k and the outputs may be elements of k, Boolean values, or integers of limited range (represented by vectors of Boolean values).

Particular arithmetic networks are of special interest: **arithmetic circuits** or **straight-line programs** (SLPs), without division or branching, containing neither selectors nor (propositional) Boolean operations. Generally speaking the *size* of the DAG (or the **sequential complexity** of the arithmetic network) is nothing but the number of its nodes (thus for a SLP the number of additions and multiplications involved). For details and elementary properties of straight-line programs we refer to [Str72], [Gat86], [Sto89] or [Hei89].

A.2.2 Zero Testing: Deterministic versus Probabilistic Algorithms

Arithmetic operations on polynomials represented by straight-line programs are immediate. The only non-trivial point when dealing with this data structure is equality checking (or zero testing). One can perform this task by evaluating the SLP at sufficiently many points. This is formalized in terms of "correct test sequences" of points with coordinates from k according to a theorem of [HS82a], which we recall for completeness:

Let D and L be two positive integers, and let us define the subset $W(D,n,L)$ of polynomials of $k[x_1,\dots,x_n]$, of degree at most D, which can be coded by a SLP with at most L arithmetic operations. Furthermore given a subset Γ of k, a family $\gamma := \{\gamma_1,\dots,\gamma_m\}$ (with $\gamma_i \in \Gamma$) of m points in k^n is called a **correct test sequence** for $W(D,n,L)$ if every polynomial in the latter vanishing on the points of γ is actually identically zero.

Theorem 6 *[HS82a, Theorem 4.4] Let us fix a subset Γ of k of cardinality $\#\Gamma = 2L(D+1)^2$, and a cardinality $m := 6(L+n)(L+n+1)$. The subset $\tau(D,n,L,\Gamma) \in \Gamma^{nm}$ of correct test sequences for $W(D,n,L)$ satisfies:*

$$\#\tau(D,n,L,\Gamma) \geq (\#\Gamma)^{nm}(1-(\#\Gamma)^{-\frac{m}{6}}).$$

In other words, the ratio of incorrect test sequences over all sequences of Γ^{nm} is at most the quantity:

$$\frac{1}{(2L(D+1)^2)^{(L+n)(L+n+1)}}$$

which is uniformly bounded by $\varepsilon := 1/262144$.

Although the choice of such a correct test sequence could be done by a deterministic algorithm, the cost of doing so would exceed the main complexity class we want. Therefore the algorithms we study below will be non-uniform insofar as they depend on the choice of correct test sequences. On the other hand, the theorem allows us to randomly choose correct test sequences with a probability of failure which becomes arbitrarily small as the parameters D, n and L increase. Therefore our algorithms can be uniformly randomized within the same order of (average) complexity. In doing so we encounter the following kind of probabilistic procedure which we call a **randomized algorithm**.

A randomized algorithm has error probability bounded by some $0 \leq \varepsilon < 1/2$ when accepting an input and error probability zero when rejecting it (in our case we may choose $\varepsilon = 1/262144$). As far as our algorithms compute polynomials or rational functions the correctness of the output depends only on the correctness of previous and intermediate decisions made by probabilistic procedures and can be checked randomly. In this sense, we apply also the term *randomized procedure* to the computation of polynomials or rational functions. Thus our results are valid not only in the sense of the non-uniform complexity model, but also in the sense of probabilistic (randomized) algorithms (see [BDG95, §6.6], [GH93, §1.2.3, §1.3 and §2.2] and [FGS95, §1.3 and §2.1] for more details).

To sum up we get the weakened following form which allows a probabilistic treatment of the theorem:

Theorem 7 *[FGS95, Theorem 2.1] There exists an arithmetic network over k of size $O(Lm) = O(L(L+n)^2)$, which given any SLP (without divisions) of size at most L, checks if the n-variate polynomial it represents is identically zero. Moreover the network can be constructed by a probabilistic algorithm in sequential time $O(L(L+n)^2)$ with a probability of failure uniformly bounded by $\varepsilon := 1/262144$.*

In the sequel, we shall consider a special class of polynomials in n variables of total degree D whose complexity of evaluation L is of the same order as D (compare with $\binom{D+n}{n}$ the number of monomials of such a dense polynomial). In other words, these polynomials can be evaluated much faster than we should expect from their total degree. For this class, the above theorem yields a probabilistic zero-test of complexity polynomial in D (or L).

In practice, we encounter two difficulties when using these ideas for large values of D. First, the size L induces an important memory cost to store the SLP. Next, although polynomial, the bound $m = 6(L+n)(L+n+1)$ from Theorem 2.1 on the length of a correct test sequence is by far too large to be useful. Thus, our implementation will not be able to *certify* that such a multivariate polynomial coded as a SLP is identically zero.

However, we show below (see §A.3.2) how to evaluate the SLP (without storing it) at a number of points which can be made arbitrarily large.

A.2.3 Noether Position of Projective Varieties

Let k be an infinite effective field, and $\overline{k}$ be an algebraic closure of k. Given a set of homogeneous polynomials $f_1, \ldots, f_s$ in $k[x_0, \ldots, x_n]$, consider the projective variety $V = V(f_1, \ldots, f_s)$ generated by the f_i in projective n-space $\mathbb{P}^n(\overline{k})$. We want to calculate the dimension of the projective variety V.

There are several approaches to this problem. We distinguish two different tasks, first the computation of an upper bound; and second the certification that an integer known to be an upper bound is actually the dimension.

In [GH93] Giusti and Heintz gave an algorithm to compute the dimension, which actually computes a change of coordinates putting the new variables in Noether position. The variables $x_0, \ldots, x_r$ are said to be **independent** with respect to V if $(f_1, \ldots, f_s) \cap k[x_0, \ldots, x_r]$ is the trivial ideal (0). If moreover the canonical homomorphism

$$k[x_0, \ldots, x_r] \to k[x_0, \ldots, x_n]/(f_1, \ldots, f_s)$$

is an integral ring extension, the variables $x_0, \ldots, x_n$ are said to be in **Noether position** with respect to V. The latter condition means that the canonical images of the variables $x_{r+1}, \ldots, x_n$ satisfy integral dependence relations (in other words are algebraic integers) over $k[x_0, \ldots, x_r]$. As a consequence the dimension of V is nothing but r. In order to simplify the complexity considerations we shall suppose that the total degree d of the f_i's is at least n.

Theorem 8 *[GH93] Let $f_1, \ldots, f_s$ be homogeneous polynomials in $k[x_0, \ldots, x_n]$ of degree at most d ($\geq n$), defining a projective variety V in $\mathbb{P}^n(\overline{k})$. There exists a randomized algorithm without divisions which computes with sequential complexity $s^{O(1)}\, d^{O(n)}$ a linear change of coordinates over k such that the new variables are in Noether position with respect to V.*

Let us recall the main steps of this algorithm. It is organized around a loop, with decreasing index m starting from n. The $(n-m)$th iteration is entered with the condition: the canonical images of $x_{m+1}, \ldots, x_n$ are already algebraic integers over $R^{(m)} := k[x_0, \ldots, x_m]$.

- Let z be a new variable; we denote by $f_i^{(m)}$ the polynomial

  $$f_i(zx_0, zx_1, \ldots, zx_m, x_{m+1}, \ldots, x_n)$$

  considered as a polynomial in $R^{(m)}[x_{m+1}, \ldots, x_n, z]$. It is homogeneous of degree (with respect to the variables $x_{m+1}, \ldots, x_n, z$) the total degree of f_i.

- Let W be the projective variety in $\mathbb{P}^{n-m}(\overline{K})$ defined by $f_1^{(m)}, \ldots, f_s^{(m)}$, where $K = K^{(m)}$ is the field of fractions $k(x_0, \ldots, x_m)$ of $R^{(m)}$.

- In this situation a **criterion for independence** is that W is not empty, which can be checked by computing a Gröbner basis or by applying an effective projective Nullstellensatz as follows:

  - Let $N = 1 + \sum_{i=1}^{s}(\deg(f_i) - 1)$ (the bound of the effective projective Nullstellensatz). Create the matrix Q of the linear $R^{(m)}$-linear application $(h_1, \ldots, h_s) \mapsto h_1 f_1^{(m)} + \cdots + h_s f_s^{(m)}$, h_i being an homogeneous polynomial of degree $N - \deg(f_i)$ in $R^{(m)}[x_{m+1}, \ldots, x_n, z]$, on the monomial basis (in $x_{m+1}, \ldots, x_n, z$); the coefficients are of course in $R^{(m)}$.
  - Use Berkowitz-Mulmuley linear algebra ([Ber84], [Mul87]) to construct the DAG corresponding to the determinant of the product matrix $Q^t Q$ to check the surjectivity of Q (this determinant lives in $R^{(m)}$).
  - A probabilistic test is then performed to determine whether this DAG represents zero (W is not empty) or not, in which case a point in $\mathbb{P}^m(k)$ with homogeneous coordinates $(a_0, \ldots, a_m)$ ($a_m \neq 0$) where it is non zero is found.

- If the criterion is satisfied, we are in Noether position and we stop.

- Otherwise, there exists a nonzero homogeneous polynomial g in $k[x_0, \ldots, x_m]$ which vanishes on V and does not vanish at $(a_0, \ldots, a_m)$. After the change of coordinates $x_0 \leftarrow x_0 + a_0 x_m, \ldots, x_{m-1} \leftarrow x_{m-1} + a_{m-1} x_m, x_m \leftarrow a_m x_m$ the polynomial g becomes monic in x_m, hence the canonical image of x_m is an algebraic integer over $R^{(m-1)}$ and we can enter the $(n + 1 - m)$th iteration.

For a more complete mathematical description of this algorithm, we refer to [GH93, pp. 25–27] and the Userguide of [Lec97]. After showing in the next section how SLPs can be dealt with in practice, we discuss the implementation of this algorithm.

A.3 Evaluation Data Structure and Maple Implementation

In this section, we show how algorithms based on straight-line programs can be turned into programs. The fundamental use of directed acyclic graphs made by the computer algebra system Maple is first used to provide a straightforward implementation exhibiting the required complexity. The efficiency of this implementation is however hindered by an important consumption of memory. A method based on evaluation, and reminiscent of the technique called deforestation [Wad90, CDPR98] from theoretical computer science, can then be applied to reduce the memory used in intermediate steps and leads to faster programs.

A.3.1 Efficient Evaluation in Maple

The Maple computer algebra system is based on a systematic use of common subexpression sharing. Objects which might look like expression trees to the user are actually stored as directed acyclic graphs, where only one copy of each distinct subtree is kept. This is accomplished by maintaining a hash table of all the expressions occurring simultaneously in a session. The structure thus obtained can be viewed as a single directed acyclic graph the children of whose root correspond to all the distinct subexpressions residing simultaneously in memory. A simple consequence of this representation is that syntactic equality of expressions is reduced to checking equality of addresses and can thus be performed in constant time. This provides the basis for the efficiency of Maple's *option remember* which can be used to record the pairs (input, output) of a procedure. Conversely, this option can be used to write recursive procedures performing DAG traversals of their argument instead of tree traversals without this option. This can lead to an improved complexity of the algorithm.

For instance, it is unfortunate that up to the current version, Maple's substitution command `subs`, which is commonly used to evaluate an expression at a point, does not benefit from this nice mechanism and has a complexity related to the size of the tree instead of the size of the DAG. The use of a DAG traversal to improve the complexity can be illustrated with a simple alternate procedure:

$$
\begin{array}{l}
\mathit{dagsubs} := \mathbf{proc}(\mathit{tosubs}{::}\{\mathit{name} = \mathit{algebraic}, \mathrm{list}(\mathit{name} = \mathit{algebraic})\},\ \mathit{expr})\\
\quad \mathbf{local}\, \mathit{dosubs},\ i,\ \mathit{res};\\
\qquad \mathit{dosubs} := \mathbf{proc}(\mathit{expr})\,\mathbf{option}\, \mathit{remember};\\
\qquad\qquad \mathbf{if}\,\mathrm{nops}(\mathit{expr}) = 1\,\mathbf{then}\,\mathit{expr}\,\mathbf{else}\,\mathrm{map}(\mathrm{procname},\ \mathit{expr})\,\mathbf{fi}\,\mathbf{end};\\
\qquad \mathbf{if}\,\mathrm{type}(\mathit{tosubs},\ \mathit{name} = \mathit{algebraic})\,\mathbf{then}\,\mathit{dosubs}(\mathrm{op}(1,\ \mathit{tosubs})) := \mathrm{op}(2,\ \mathit{tosubs})\\
\qquad \mathbf{else}\,\mathbf{for}\, i\, \mathbf{in}\, \mathit{tosubs}\,\mathbf{do}\,\mathit{dosubs}(\mathrm{op}(1,\ i)) := \mathrm{op}(2,\ i)\,\mathbf{od}\,\mathbf{fi};\\
\qquad \mathit{res} := \mathit{dosubs}(\mathit{expr});\\
\qquad \mathit{dosubs} := \mathrm{subsop}(4 = \mathrm{NULL}, \mathrm{op}(\mathit{dosubs}));\\
\qquad \mathit{res}\\
\quad \mathbf{end}
\end{array}
$$

In Table A.1, we give examples of the time and memory[1] required by both `subs` and this simple `dagsubs`. The test suite is the following sequence of polynomials:

$$P_0(x) = 1, \qquad P_1(x) = x, \qquad P_{n+2}(x) = xP_{n+1}(x) + P_n(x) + 1, \quad n \geq 0.$$

[1]The tests in this chapter have been performed on a PC Pentium Pro (200MHz) with 512Mb of memory, running Linux 2.0.27 and Maple V.3. This computer forms part of the equipment of GDR Medicis: http://medicis.polytechnique.fr.

Table A.1: Substitution and DAGs

n	30	31	32	33	34
`subs`	6sec	11sec	15sec	29sec	47sec
	46Mb	75Mb	121Mb	195Mb	316Mb
`dagsubs`	6sec	9sec	12sec	19sec	37sec
	25kb	26kb	28kb	28kb	29kb

Of course the polynomials are not expanded and the substitution consists in replacing x by another variable y. The time difference is not very large, but `subs` is a function of Maple's compiled kernel, whereas `dagsubs` is interpreted. However, while `dagsubs` needs a very limited amount of memory, `subs` requires more than one thousand times this amount, and the ratio increases very fast with the index of the polynomials. This example clearly demonstrates that working with DAGs when possible can be crucial in terms of efficiency.

When the DAG is large and many evaluations of it at different values are required, for instance to prove that it is the zero polynomial with a correct test sequence, Maple also provides tools working at the DAG level (via the *option remember*) that generate optimized Fortran or C code. For instance, on the polynomial P_{10} above, Maple's `optimize` yields:

$$t_1 = x^2, \quad t_3 = x(t_1 + 2), \quad t_5 = x(t_3 + x + 1), \quad t_7 = x(t_5 + t_1 + 3),$$

$$t_9 = x(t_7 + t_3 + x + 2), \quad t_{11} = x(t_9 + t_5 + t_1 + 4), \quad t_{13} = x(t_{11} + t_7 + t_3 + x + 3),$$

$$t_{18} = x(x(t_{13} + t_9 + t_5 + t_1 + 5) + t_{11} + t_7 + t_3 + x + 4) + t_{13} + t_9 + t_5 + t_1 + 6.$$

This can then be translated into a Maple procedure (`makeproc`) or alternatively into Fortran or C code. Here is for instance the corresponding C code:

```
t1  = x*x;
t3  = x*(t1+2.0);
t5  = x*(t3+x+1.0);
t7  = x*(t5+t1+3.0);
t9  = x*(t7+t3+x+2.0);
t11 = x*(t9+t5+t1+4.0);
t13 = x*(t11+t7+t3+x+3.0);
t18 = x*(x*(t13+t9+t5+t1+5.0)+t11+t7+t3+x+4.0)+t13+t9+t5+t1+6.0;
```

All these tools performing conversions between DAGs and SLPs implement the canonical isomorphism between polynomials and polynomial functions (the ground field being infinite).

A.3.2 Deforestation

The algorithm described in Section A.2.3 computes SLPs. The only information required about some of these SLPs is whether they represent the zero polynomial and if not, a point at which they are different from zero. This is done by evaluating the SLP at one or several points (see §A.2.1). The complexity of evaluating the SLP is directly related to its size, i.e. the memory it uses. The idea of *deforestation* is to perform the same evaluations *without computing the SLP first*, thus saving memory occupied by unnecessary expression trees (whence the name deforestation).

More generally the deforestation of a program is a transformation consisting in eliminating the building of intermediate structures introduced by composition of functions.

For instance, Lisp code to compute the last element of a list could be implemented by

```
(defun last (l) (car (reverse l)))
```

where `reverse` is written

```
(defun reverse (l) (reverse2 l nil))
(defun reverse2 (l1 l2) (if l1 (reverse2 (cdr l1) (cons (car l1) l2)) l2))
```

This implementation has the disadvantage of creating an intermediate list of the same length as the argument. The deforested version would read

```
(defun last (l) (last2 l nil))
(defun last2 (l1 l2) (if l1 (last2 (cdr l1) (car l1)) l2))
```

In some cases this program transformation can be performed automatically, see [Wad90].

In the case of the algorithm of Section A.2.3 this deforestation in Section A.3.4 will mainly consist in a simple exchange of loops. Let `pol(x1,...,xn)` be any Maple procedure using only `+, -, *, :=`, integers and loops with fixed depth. Such a function has the particularity to run with any kind of entries in a ring: if its arguments are variable names it returns a DAG in which the arguments are the leaves, if its arguments are integers it returns an integer. Thus, if $x_1, \ldots, x_n$ are variable names, the code:

$p := \mathrm{pol}(x_1, \ldots, x_n)$;
for point **in** list_of_points **do**
  if *dagsubs*([**seq**($x_i = \mathrm{point}[i], i = 1, \ldots, n$)], p) $\neq 0$ **then** RETURN(point) **fi**
od;
...

can be written:

for point **in** list_of_points **do if** pol(**op**(point)) $\neq 0$ **then** RETURN(point) **fi od**;
...

The memory required by the DAG p in the first version of the code is not necessary in the second one and this has an important impact on the speed of the execution (by reducing

Table A.2: Computations on matrices with polynomial entries

dimension	10	20	50	100	200
naive (maple `det`)	∞	∞	∞	∞	∞
computation of the DAG (Berkowitz)	.08	1.5	400	> 5000	
test sequence (#pts)	$5.7\,10^{7}$	$2.3\,10^{10}$	$8.0\,10^{13}$		
evaluation at one point of the DAG (`subs`)	53	> 5000			
optimization of the DAG (`optimize`)	.02	> 5000			
evaluation at one point, deforested version	.03	.20	23	47	1050

the time spent in memory handling). In the application to the algorithm of Section A.2.3, the complexity of the deforested version is that given by Theorem 8. Note however that in a more general context, the DAG p might have smaller number of arithmetic operations.

After a detailed example illustrating the evaluation strategies presented above, the rest of this chapter describes a "manual" deforestation of the algorithm from Section A.2.3. Besides the reduction of the amount of memory used in intermediate steps, further gains are possible by substituting faster algorithms operating on constants for those operating on SLPs. Then the complexity estimates based on SLPs provide an upper bound on the complexity of the actual computation.

A.3.3 Example: Determinant of a Matrix of Polynomials

In this section, we illustrate how the systematic exploitation of the DAG structure leads to improved algorithms computing the determinants of matrices of multivariate polynomials with integer coefficients. This task is an important building block of the theoretical algorithm from Section A.2.3.

The matrices we take in our examples are square $k \times k$ matrices with polynomial entries having 3 variables, at most 6 terms, a total degree at most 5 and coefficients with 2 decimal digits. The matrices are sparse with $5k$ entries at random filled by polynomials provided by Maple's `randpoly` function. To obtain regular matrices, the diagonal is first filled with 1's. To obtain singular matrices, the columns from 1 to $k-1$ are summed into the kth one. Timings for regular and singular matrices turn out to be similar, thus we give only one table (Table A.2) of results.

Naive approach We first use Maple's `det` command, which is based on a mixture of fraction free Gaussian elimination and minor expansion. Since the resulting polynomial is always expanded, the computation is expensive because of the exponential growth of the number of multivariate monomials as the degree increases. This shows in Table A.2: on all our examples, Maple returns an error message "object too large" (abbreviated ∞ in the table).

Straight-line programs and Berkowitz's algorithm In order to overcome the exponential complexity of expanding determinants, it is natural to turn to DAGs or to straight-line programs evaluating them. Recognizing zero with this data-structure becomes an expensive operation. Thus an approach based on Gaussian elimination does not apply anymore. Several other algorithms can be applied. We use an algorithm due to [Ber84] which has the advantage of a simple description. Given an $n \times n$ matrix, it computes its characteristic polynomial (and in particular its determinant) in $O(n^4)$ arithmetic operations on the coefficients and requires neither test nor division.

On a generic square matrix of size n, the number of nodes of the DAG evaluating the expanded determinant and the one produced by Berkowitz's algorithm are indicated in Table A.3. In this case, the polynomial complexity of Berkowitz's algorithm quickly yields better results than the number $n!+1$ of monomials of the generic determinant. This is naturally reflected by the time required for the computation. For our test matrices, the results turn out to be very similar: Berkowitz's algorithm takes almost no time on matrices for which `det` cannot compute the result. This appears in the second line of Table A.2.

Evaluation and test sequences In line 4 of Table A.2, we give the time used to evaluate the DAG computed via Berkowitz's algorithm at one point. In regular cases, this is usually also the time required to prove that the matrix is regular. In the singular cases, we also show an estimate of the number of points m forming the correct test sequences for DAGs of the corresponding size. The table shows that although this approach makes it possible to deal with objects which are too large for Maple when expanded, it also rapidly produces objects with which it is impossible to proceed in a reasonable amount of time. (Part of this might be due to the limited size of the hash tables used by Maple.) Whatever the speed of evaluation, the number of points indicated on line 3 leads to the conclusion that *the theoretical bound which leads to polynomial complexity of the deterministic algorithm is much too large to be of practical use.*

Deforestation The process outlined above consists of two steps. First a DAG is constructed via Berkowitz's algorithm, then this DAG is evaluated at one or several points. Inverting the loops as discussed in §A.3.2 is then a natural strategy: we first evaluate the matrix at these points and then compute the determinant. This is clearly an improvement, since the DAG for the determinant does not need to be stored in memory anymore and the determinant can be evaluated by faster algorithms. In the singular case, we can still use the bound on the number of points which follows from considering

Table A.3: Sizes of different representations of the determinant of an $n \times n$ matrix

dimension	2	3	4	5	6	7	8	9
dag size (expanded)	3	7	25	121	721	5041	40321	362881
dag size (Berkowitz)	3	20	64	169	343	664	1104	1817

the DAG produced via Berkowitz's algorithm without actually executing this algorithm. The resulting timings appear in the last line of Table A.2 and vindicate this strategy. Preliminary experiments in C using LEDA's bignums [MNU97] seem to indicate that we can only expect an improvement of a factor four to five by performing this evaluation directly in C.

A.3.4 Deforestation of the Algorithm

The algorithm described in Section A.2.3 relies on the computation of determinants of matrices of polynomials. As shown in the previous Section A.3, this operation benefits from deforestation. We can go one step further and apply deforestation to the whole algorithm.

The deforested algorithm is organized around nested loops. The outer loop is decreasing of index m and starts from n. Its $(n-m)$th iteration is entered with the condition: the canonical images of $x_{m+1}, \dots, x_n$ are already algebraic integers over $R^{(m)} := k[x_0, \dots, x_m]$. The inner loop is indexed by a list of points l_m in $\mathbb{P}^m(k)$. Each point of homogeneous coordinates $(a_0, \dots, a_m)$ of l_m satisfies $a_m \neq 0$.

Algorithm For m from n to 0 do:

- For each point $(a_0, \dots, a_m)$ of l_m do:
  - Specialize the polynomials $f_i^{(m)}$ of $k(x_0, \dots, x_m)[x_{m+1}, \dots, x_n, z]$ to homogeneous polynomials $f_i(a_0 z, \dots, a_m z, x_{m+1}, \dots, x_n)$ of $k[x_{m+1}, \dots, x_n, z]$;
  - Apply a Gröbner basis algorithm with total degree ordering to the specialized $f_i^{(m)}$ to compute a standard basis; then determine whether the variety $W \subseteq \mathbb{P}^{n-m}(\overline{k})$ they define is empty or not.
    - $*$ If it is empty, break this inner loop;
    - $*$ If it is not empty, repeat this process with another point of l_m.
- We exit from this inner loop in two possible cases:
  - A point $(a_0, \dots, a_m)$ of l_m has been found such that the corresponding variety W is not empty. We perform the following change of variables in the input polynomials f_i: $x_0 \leftarrow x_0 + a_0 x_m, \dots, x_{m-1} \leftarrow x_{m-1} + a_{m-1} x_m, x_m \leftarrow a_m x_m$ (Recall that a_m is not zero). The variables x_m is now an algebraic integer over $R^{(m-1)}$, we can enter the $(n+1-m)$th step of the outer loop.
  - All the points of l_m are such that the corresponding W is empty. In this case we return the integer m and the polynomials f_i.

That this algorithm is the deforestation of the one of Section A.2.3 follows from the fact that the computation of a Gröbner basis in the projective case is nothing but a triangulation of the block Toeplitz matrix Q by elementary column operations.

The index m of the outer loop represents an upper bound on the dimension of the variety. As seen in the previous section, the number of points where the inner loop has to be performed when the corresponding determinant is actually zero is very large. However, this will only happen once, at the end of the algorithm, when m reaches the dimension. In practice, we shall therefore stop the inner loop after a settable number of points. We thus obtain a proved bound on the dimension and a likely value for it.

The practical complexity of the deforested algorithm inherits the good complexity behavior of the theoretical one but the non-uniform deterministic aspect is lost. It is clear that the specializations which do not lead to a correct answer are enclosed in a strict algebraic subset but we do not know any precise bound on the probability of failure. One reason is that we do not know bounds on the probability of failure of a Gröbner basis of a specialized system to be the specialization of the Gröbner basis of the system.

In the same way as the specialization of the free variables we could specialize the integers, that is, perform the computations modulo a prime number picked-up at random, but this is out of the scope of this study.

Obviously, the computation will be fast when the dimension is large, since it is reduced to Gröbner basis computations on systems of polynomials in much less variables. Thus in cases of low dimension, the timings of our method are comparable with a direct Gröbner basis computation, while our method is best suited for cases of large dimension, as exemplified in the experimental data below.

Illustration of the Algorithm Let us consider $f_1 = x_0x_1$ and $f_2 = x_0x_2$ in $k[x_0, x_1, x_2]$, we have $n = 2$. We now detail what the algorithm does:

First step: set $m = 2$; $f_1^{(2)} = x_0x_1z^2$, $f_2^{(2)} = x_0x_2z^2$ are seen as homogeneous equations in $K[z]$, where $K = k(x_0, x_1, x_2)$. The corresponding variety W is empty, the specialization $a_0 = 1$, $a_1 = 0$, $a_2 = 1$ leads to the change of variables: x_0 is replaced by $x_0 + x_2$, x_1 remains x_1 and x_2 remains x_2. The new equations are $f_1 = x_0x_1 + x_1x_2$, $f_2 = x_0x_2 + x_2^2$, the canonical image of the variable x_2 is now an algebraic integer over $k[x_0, x_1]$. We go to the second step.

Second step: set $m = 1$; $f_1^{(1)} = x_0x_1z^2 + x_1zx_2$, $f_2^{(1)} = x_2^2 + x_0zx_2$ are seen as homogeneous equations in $K[x_2, z]$, where $K = k(x_0, x_1)$. For any specialization (a_0, a_1) of (x_0, x_1) the corresponding variety W is not empty since it contains the projective point $x_2 = -a_0z$. So the algorithm returns 1 as the dimension.

A.4 Examples

A.4.1 The Behavior at Infinity of the Cyclic n-roots Systems

This system is related to the question of finiteness and structure of the corresponding set of cyclic n-roots [Bjö90]. Let $x_1, \dots, x_n$ be variables and M_i be the monomial $x_1 \cdots x_i$. Let σ be the cycle $(1, 2, \dots, n)$ of the nth permutation group. We define the ith cyclic equation of the nth system as:

$$H_n^i = \sum_{k=0}^{n-1} \sigma^k(M_i) \quad \text{for} \quad 1 \leq i \leq n-1, \qquad H_n^n = M_n.$$

Now, `infcyclic` n defines the system $\{H_n^n = H_n^{n-1} = \ldots = H_n^1 = 0\}$. For example, when $n = 3$ the system H_3^∞ is $\{x_1x_2x_3 = x_1x_2 + x_2x_3 + x_3x_1 = x_1 + x_2 + x_3 = 0\}$.

Proposition 53 (in collaboration with E. Schost) *The system H_n^∞ defines in $\mathbb{P}^{n-1}(\mathbb{C})$ a projective variety of dimension at least:*

$$r_{\inf}(n) = n - \left\lceil \frac{n}{\lfloor\sqrt{n}\rfloor} \right\rceil - \lfloor\sqrt{n}\rfloor.$$

Proof. Let $k = \lfloor\sqrt{n}\rfloor$, $l = \left\lceil \frac{n}{\lfloor\sqrt{n}\rfloor} \right\rceil$. The polynomial H_n^n is reduced to a monomial containing x_1, which shows that the variety contains a subvariety defined by $x_1 = H_n^{n-1} = \cdots = H_n^1 = 0$. Modulo $x_1 = 0$, H_n^{n-1} is again reduced to a monomial. This monomial contains x_{1+k}. Proceeding in this manner using the equations $H_n^{n-1} = 0, \ldots, H_n^{n-(l-2)k-1} = 0$ shows that the variety contains a subvariety defined by $x_1 = x_{1+k} = \cdots = x_{1+(l-1)k} = H_n^{k-1} = \cdots = H_n^1 = 0$. Then applying Krull's Lemma [Eis95, 8.2.2] completes the proof. □

The resulting lower bounds for the dimension of the system `infcyclic` H_n^∞ are given in Table A.4.

Note that it is possible to decompose by hand the system `infcyclic` and thus obtain better time/space results. This is for instance done by Maple's `gsolve` function. In our tests, we shall not allow this simplification for either our package or Gröbner basis packages. The reason for us to use the `infcyclic` system for our tests is the known lower bounds for the dimension from Table A.4.

We present the tables of experimental results obtained for the two tasks of bounding and computing the dimension on these systems.

The algorithm described in Section A.3.4 and denoted `pnp` is compared with Maple's `grobner` function (`grobner`) and with Faugère's `Gb` applied directly to the systems. The `Gb` computation uses the function `sugar`, with the optional parameter `"info"`, and a total degree ordering (degree reverse lexicographic). Time is measured in *seconds* and memory space in *kilobytes*. A time t preceded by the symbol $>$ means that the computation has been manually aborted after t seconds, the corresponding value in the column labelled `space` is the memory allocated until then. Empty entries in the tables are due to technical problems measuring very small quantities and/or their scaling with the rest of the entries in the same table.

In table A.5, UBD stands for "upper bound on the dimension". For example the first line of Table A.5 reads: the dimension of the system `infcyclic 6` has been proved to be at most 2 in 0.2 seconds with a memory space of 720kb, using our program, whereas using the Maple `grobner` function it took 0.71 seconds and 1180kb.

Table A.4: Lower bounds for the dimension of the system `infcyclic` H_n^∞

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$r_{\inf}(n)$	-1	-1	-1	0	0	1	1	2	3	3	4	5	5	6	7	8	8

Table A.5: Bounding the dimension

system / method	pnp		grobner		Gb	
UBD	Time	Space	Time	space	Time	space
`infcyclic 6`						
2	0.2	720	0.71	1180		
1	2.5	1400	8.48	1500	1.0	1200
`infcyclic 7`						
4			1.7	1400		
3	0.2	720	2.5	1500		
2	2.5	1440	65	2230	7	2200
1	900	3200	13000	6200	950	3200
`infcyclic 8`						
5			5.8	1500		
4	0.3	720	6.18	1500		
3	3.0	1500	396	2700	3	2200
2	1400	4130	> 250000	> 18000	54000	27600
`infcyclic 9`						
6			21	1900		
5	0.3	720	22	1900		
4	5.1	1700	2293	3080	3	3400
3	1600	7000	>200000	> 8000	> 108000	> 180000

Table A.6: Computing the dimension

		pnp (one pt)		grobner	
System	Dim.	Time	Space	Time	Space
`infcyclic`					
2	-1	0.00	243	0.05	105
3	-1	0.01	448	0.04	149
4	0	0.07	2165	0.13	481
5	0	6.5	1834	4	1507
6	1	107	2948	114	2424

The first task is the determination of upper bounds on the dimension of the variety. Using the algorithm of §A.3.4, we know that at step m of the iteration the dimension of the variety is at most m. It is also possible to compute upper bounds on the variety during the calculation of the Maple `grobner` function: the dimension is at most the dimension of the monomial ideal generated by the leading monomials of the S-polynomials computed when performing a Buchberger algorithm, see e.g., [CLO97].

The computation of upper bounds on the dimension of the projective variety is the strong point of our method. Where the Gröbner basis algorithms are forced to work with the complete equation system in many variables, our recursive approach yields the correct answer a lot faster. First, it can be seen that Maple's `grobner` function is the least efficient approach. We observe then, that even using this same Maple `grobner` function for the intermediate calculations in our method, the resulting performance is already competitive with the stand-alone `Gb` program. Note that any improvements in the field of Gröbner bases computations will also yield direct improvements for our method.

The second task is to determine the exact dimension of the given variety. The correct test sequences are out of reach and thus we can only evaluate a zero-polynomial given as a SLP at a certain number of points. The dimension computed is indicated in the second column of Table A.6. In the next column, we give the time necessary to evaluate the final polynomial at one point. As already discussed, this example being of low dimension, the performance of our method in this case is comparable with that of a mere Gröbner basis computation.

For the next examples, we did not try to estimate the time taken by the Gröbner basis package to bound the dimension and we only give the total time of the computation.

A.4.2 Generic Polynomials

We consider a polynomial system of three homogeneous equations in 11 variables of degree 2. The coefficients are randomly chosen integers between -99 and 99.

Our package takes 1.33s to prove that the dimension is at most 8. Then the final putative zero-polynomial is evaluated at random points in 2.22 sec. each in a constant memory of 1.9Mb. Maple's `grobner` computes a Gröbner basis for the total degree order in 10053 sec. and 5.3Mb. By extrapolation, 4528 points could be tested by our program.

A.4.3 Incomplete Determinental Ideals

Let M be a 5×3 matrix, filled with linear forms in 11 variables, with random coefficients between -10 and 10. M has ten 3×3 submatrices. Their determinants are homogeneous polynomials of degree 3 in the 11 variables. We drop one of them, in order to obtain of system with 9 polynomials.

Our package takes 1s to prove that the dimension is less than 8. As before, each random point with random coordinates between 0 and 100 is tested in 1.16s and 3.8Mb. Maple's `grobner` with total degree order computes the basis in 7093s and 8.32Mb. By extrapolation 6114 points could be tested by our program.

M is now a 7×4 matrix filled with random linear forms in eleven variables, with coefficients between -10 and 10. M has thirty five 4×4 submatrices. Their determinants are homogeneous polynomials of degree 4. We have chosen nine of them, corresponding to the following sets of number lines :

$$[1,2,3,4],[1,2,3,5],[1,2,3,6],[1,2,3,7],[1,2,4,5],$$

$$[1,2,4,6],[1,2,4,7],[1,2,5,6],[1,2,5,7].$$

We now compare Faugère's `Gb` package with a version of our program which calls `Gb` externally for the Gröbner bases computations via `gblink` [LS97].

Our program takes 0.08s and 0.6Mb to prove that the dimension is less than 8. Each point tested is, as previously, a random point with coordinates between 0 and 100. Eighty-eight have been tested in 1056s and 1.2Mb. `Gb's Grobner` with its `tgrobner` function has not finished in 3 days and 451Mb. By extrapolation at least 21600 points can be tested with our program.

A.5 Conclusion

Algorithms using SLPs to store multivariate polynomials suffer two practical problems: first, the memory management becomes prohibitive when dealing with very big SLPs and second the use of non-uniform deterministic zero tests requires evaluations of the SLPs on a very big set of points. A direct implementation of these algorithms does not turn out to be efficient. Along the example of computing a Noether position of a projective variety, we have shown how to transform the theoretical algorithm into a practical one which does not require SLPs anymore in the intermediate computations and which is very simple to implement. The practical complexity of the new algorithm inherits the one of the theoretical one. This idea should apply in a wide class of algorithms in elimination theory. (See [GLS01] for another step in this direction.)

Annexe B

Equidimensional Decomposition via Bertini's first Theorem

Let $f_1, \ldots, f_s$ be polynomials in n variables over a field of characteristic zero and d be the maximum of their total degree. We propose a new probabilistic algorithm for computing a *geometric resolution* of each equidimensional part of the variety defined by the system $f_1 = \cdots = f_s = 0$. The returned resolutions are encoded by means of *Straight-Line Programs* and the complexity of the algorithm is polynomial in a *geometric degree* of the system. In the worst case this complexity is asymptotically polynomial in sd^n.

Parts of this chapter have been published in [Lec00].

B.1 Introduction

Let k be a field of characteristic zero, $f_1, \ldots, f_s$ be s polynomials in $k[x_1, \ldots, x_n]$ and d be the maximum of their total degree. Let $\mathcal{V} = \mathcal{V}(f_1, \ldots, f_s)$ be the algebraic variety defined by the system $f_1 = \cdots = f_s = 0$ in the n dimensional affine space over an algebraic closure $\overline{k}$ of k. We are interested in computing an exact description of $\mathcal{V}$, from the computer algebra point of view.

Up to now, the best complexity upper bound known is asymptotically polynomial in sd^{n^2}: in [CG83, Chi96, Chi97] Chistov and Grigoriev present algorithms with such a complexity for computing a decomposition of $\mathcal{V}$ into irreducible components; Giusti and Heintz give in [GH91] another method with the same complexity, moreover their decomposition into equidimensional parts is well-parallelizable; Elkadi and Mourrain propose in [EM99a] a method based on Bézoutian matrices, their algorithm is probabilistic.

A byproduct of an equidimensional decomposition is the dimension but direct algorithms for computing it with a complexity polynomial in sd^n exist: in 1993 Giusti and Heintz [GH93] exhibit a well-parallelizable algorithm polynomial in sd^n, deterministic in a non-uniform complexity model and probabilistic in a uniform one; in 1996 Chistov [Chi96, Chi97] performs the same computation within the same complexity but deterministically for a uniform complexity model. A recent historical presentation of the work of Chistov, Grigoriev and Vorobjov can be found in [Vor99].

Our aim is to present an algorithm which computes a *geometric resolution* of each equidimensional part of the variety $\mathcal{V}$ with a complexity polynomial in some intrinsic degree of the system and, by the way, polynomial in sd^n in the worst case.

Our algorithm stores eliminating polynomials by means of *Straight-Line Programs* (SLP) without division; our method lies in the continuation of a series of papers initiated by Giusti, Hägele, Heintz, Montaña, Morais, Morgenstern and Pardo. The notions of geometric resolutions and intrinsic degree of a system have been introduced in [GHMP95, Par95] and the first resolution procedure for regular sequences (complete intersections) appears in 1997: the articles [GHH$^+$97, GHMP97, GHM$^+$98] present an algorithm for computing the roots of a system of polynomial equations with a complexity polynomial in an intrinsic degree of the system. In [Mor97, HMPS00] the method is generalized in order to compute the isolated roots of any system of polynomial equations and inequations (not complete intersection). In [GLS01] we simplify and redesign the algorithm and explain how to implement it. A good historical presentation of all these works can be found in [CHMP01]. Our aim is now to extend the theoretical method in order to compute not only the isolated roots but a description of all the components.

Recently and independently of our work Jeronimo and Sabia have improved the results of [GH91]: they propose in [JS00] a probabilistic algorithm for computing an equidimensional decomposition of an affine variety over a field of characteristic zero in time polynomial in sd^n, each output component is described as the set of roots of $n+1$ polynomials encoded by straight-line programs.

In the sequel, L represents the complexity of the given SLP encoding the input system $f_1, \ldots, f_s$. When considering degrees of algebraic varieties we use *Heintz' degree* [Hei83], that is the sum of the degrees of all the isolated irreducible components appearing in a minimal decomposition of the variety.

Let $t_{i,j}$ for $1 \leq i \leq n+1$ and $1 \leq j \leq s$, be some new generic parameters taking their value in k. Let g_i be the generic linear combination $\sum_{j=1}^{s} t_{i,j} f_j$, for $1 \leq i \leq n+1$. We define the *intermediate varieties* $\mathcal{V}_i$ as $\mathcal{V}(g_1, \ldots, g_i)$ for $0 \leq i \leq n+1$ and we define the *intrinsic degree* δ *of the system* as being the maximum of the degrees δ_i of $\mathcal{V}_i$, $0 \leq i \leq n$. Note that by convention $\mathcal{V}_0$ represents the whole space $\overline{k}^n$.

Theorem 9 *There exist a dense open subset* $\mathcal{G}$ *of* $k^{(n+1)s}$ *and a deterministic algorithm such that: if the values of the* $t_{i,j}$ *are taken in* $\mathcal{G}$*, it computes a* minimal geometric resolution*, encoded by SLP, of* $\mathcal{V}$ *with a non-uniform complexity polynomial in* $Lnsd\delta$*, in terms of the number of arithmetic operations in* k*.*
Let α *be a positive integer, if* Ω *is a set of points of size* $3\alpha(d+1)^{3n}$ *in* k*, a random point* $t_{i,j}$ *in* $\Omega^{(n+1)s}$ *belongs to* $\mathcal{G}$ *with probability of success at least* $1 - 2(n+1)/\alpha$*.*

The definition of geometric resolutions is stated in the next section and the complexity model in §B.4.1. Note that δ is bounded by d^n, using the Bézout inequality [Hei83].

The deterministic algorithm underlying the theorem admits a bounded error probabilistic counterpart in a uniform complexity model (see discussion in §B.4.1).

Our algorithm relies on the properties of the $\mathcal{V}_i$ and their relation with $\mathcal{V}$; we adapt the results of [GHM$^+$98] in order to perform splittings when irregular intersections occur. So this chapter is organized as follows: we first recall the definition of a geometric resolution, then we describe the properties of the generic system $g_1 = \cdots = g_{n+1} = 0$ and the last part is devoted to the description of the algorithm and the proof of Theorem 9.A Section B.5 details a technical result contained in [GHM$^+$98] but without an independent statement.

B.2 Definitions

Let $\mathcal{W}$ be a r-equidimensional algebraic variety and $\mathfrak{I}$ its annihilating ideal in the polynomial ring $k[x_1, \ldots, x_n]$.

Let M be an invertible $n \times n$ square matrix with entries in k, we say that the new coordinates $y = M^{-1}x$ are in *Noether position* with respect to $\mathcal{W}$ if the following two conditions hold: the variables $y_1, \ldots, y_r$ are *free*, which means that the canonical projection from $\mathcal{W}$ to the affine space generated by $y_1, \ldots, y_r$ is surjective, and the ring morphism

$$R := k[y_1, \ldots, y_r] \to k[y_1, \ldots, y_n]/(f_1, \ldots, f_s) =: B$$

defines an integral ring extension.

Let K be the field of fractions of R and B' the finite-dimensional K-vector space $K \otimes B$. A linear form u is said to be a primitive element of $\mathcal{W}$ if its set of powers generates B'.

A *geometric resolution* of $\mathcal{W}$ is defined by:

- an invertible $n \times n$ square matrix M with entries in k such that the new coordinates $y = M^{-1}x$ are in Noether position with respect to $\mathcal{W}$;
- a primitive element $u = \lambda_{r+1}y_{r+1} + \cdots + \lambda_n y_n$ of $\mathcal{W}$, with the λ_j in k;
- the minimal polynomial $q(T) \in R[T]$ of u in B', monic in T, with degree in T equal to the dimension of B' and
- the *parametrization* of $\mathcal{W}$ by the zeros of q, given by polynomials

  $$\rho y_{r+1} - v_{r+1}(T), \ldots, \rho y_n - v_n(T),$$

  where ρ is in R and the v_j are in $R[T]$ with degree in T strictly less than the one of q and such that B' is isomorphic to $K[T]/(q(T), \rho y_{r+1} - v_{r+1}(T), \ldots, \rho y_n - v_n(T))$.

The total degree of q is bounded by $\deg(\mathcal{W})$ and those of ρ and the v_j can be bounded by $\deg(\mathcal{W})^3$ [GLS01, KPS01].

If $\mathcal{W}$ is not equidimensional, we call a **minimal geometric resolution** a decomposition of $\mathcal{W}$ in $\mathcal{W}_0 \cup \mathcal{W}_1 \cup \cdots \cup \mathcal{W}_n$ such that $\mathcal{W}_i$ is either empty or an algebraic variety of codimension i and no component of any $\mathcal{W}_j$ is included in another one, each $\mathcal{W}_j$ being described by means of a geometric resolution.

B.3 Preparation Lemmas

The first part of our algorithm is incremental in the number of the g_i's: we give geometric resolutions of $\mathcal{V}_0, \mathcal{V}_1, \dots, \mathcal{V}_{n+1}$ in sequence. The following lemmas give the properties of this sequence when the $t_{i,j}$ are chosen generic enough.

The first lemma shows that the "interesting" successive intersections of the g_i's are regular. Note that this lemma is a local version of [GH93, §3.4.1] of Giusti and Heintz (see also [Hei83, GH91]).

Lemma 13 *For all i, $0 \leq i \leq n$, for any $t_{k,l}$ in k, with $1 \leq k \leq i$ and $1 \leq l \leq s$, there exists a nonzero polynomial P_{i+1} in $\overline{k}[(T_{i+1,l})_{1 \leq l \leq s}]$ of total degree bounded by d^i such that: for all $t_{i+1,l}$, $1 \leq l \leq s$, $P_{i+1}((t_{i+1,l})_{1\leq l \leq s}) \neq 0$ implies that for any irreducible component $\mathcal{M}$ of codimension i of the variety $\mathcal{V}_i$ the following alternative holds: $\mathcal{M}$ is in $\mathcal{V}$ or $\mathcal{V}(g_{i+1})$ intersects $\mathcal{M}$ regularly.*

Proof. Let $\mathcal{M}$ be an irreducible component of $\mathcal{V}_i$ of codimension i not included in $\mathcal{V}$, so that there exists some m such that $\mathcal{M}$ is not included in the hypersurface defined by $f_m = 0$. We take a point $C_{\mathcal{M}}$ in $\mathcal{M}$ not in $\mathcal{V}(f_m)$. Let us associate such a point $C_{\mathcal{M}}$ to each irreducible component of $\mathcal{V}_i$ of codimension i not included in $\mathcal{V}$. Let P_{i+1} be the polynomial defined by

$$\prod_{\mathcal{M}} (T_{i+1,1} f_1(C_{\mathcal{M}}) + \cdots + T_{i+1,s} f_s(C_{\mathcal{M}})).$$

By construction P_{i+1} is not identically zero, has total degree bounded by d^i and has the required property. $\square$

We apply incrementally the lemma: first we choose values for $t_{1,1}, \dots, t_{1,s}$, then for $t_{2,1}, \dots, t_{2,s}$ and so on until $t_{n+1,1}, \dots, t_{n+1,s}$. The system of g_i's we obtain satisfies the following property, denoted by R_1 in the sequel:

$\mathbf{R_1}$: for all i, $0 \leq i \leq n$, an irreducible component of $\mathcal{V}_i$ of codimension i is either included in $\mathcal{V}$ or regularly intersected by g_{i+1}.

We now quantify the probability of success of this construction.

Proposition 54 *Let α be a strictly positive integer, and Ω be a set of points of k of cardinal αd^n. A random choice of all the $t_{i,j}$, $1 \leq i \leq n+1$, $1 \leq j \leq s$, in $\Omega^{(n+1)s}$ leads to polynomials g_i satisfying the property R_1 with a probability at least $1 - (n+1)/\alpha$.*

Proof. Using Zippel-Schwartz' zero test [Sch79, Zip79, Zip93], the probability that a point $(t_{i+1,1}, \dots, t_{i+1,s})$ chosen at random in Ω^s is not a zero of P_{i+1} is at least $1 - 1/\alpha$. Thus the probability that some $t_{i,j}$ chosen at random in $\Omega^{(n+1)s}$ define some g_i satisfying the property R_1 is at least $(1 - 1/\alpha)^{n+1} \geq 1 - (n+1)/\alpha$. $\square$

In the sequel we denote by $\mathcal{V}_i^P$ (Pure codimension) the variety composed of the components of $\mathcal{V}_i$ of codimension i, by $\mathcal{V}_i^R$ (Regularly intersected) the ones of $\mathcal{V}_i^P$ being intersected regularly by g_{i+1} and $\mathcal{V}_i^I$ (Improper) the ones of $\mathcal{V}_i^P$ included in the hypersurface defined by $g_{i+1} = 0$.

Proposition 55 *For choices of $t_{i,j}$ such that properties R_1 holds, the varieties $\mathcal{V}_i$ satisfy the following properties.*

1. *For $i \geq 0$, the minimal equidimensional decomposition of $\mathcal{V}_i$ is given by*
$$\mathcal{V}_0^I \cup \cdots \cup \mathcal{V}_{i-1}^I \cup \mathcal{V}_i^P.$$
2. *The minimal decomposition of $\mathcal{V}$ into equidimensional parts is*
$$\mathcal{V}_0^I \cup \mathcal{V}_1^I \cup \cdots \cup \mathcal{V}_n^I.$$

Proof. We prove the first point by induction on i. The case $i = 0$ is trivial, let us assume that the property is true for a given $i \geq 0$. The variety $\mathcal{V}_{i+1}$ is the intersection of $\mathcal{V}_i$ by the hypersurface defined by g_{i+1}, the property R_1 implies that all the $\mathcal{V}_j^I$, $1 \leq j \leq i$, are included in $\mathcal{V}$, thus in the hypersurface defined by g_{i+1}. Hence $\mathcal{V}_{i+1}$ equals $\mathcal{V}_0^I \cup \cdots \cup \mathcal{V}_i^I \cup \mathcal{V}_i^R \cap \mathcal{V}(g_{i+1})$. The variety $\mathcal{V}_{i+1}^P$ is then the union of the irreducible components of $\mathcal{V}_i^R \cap \mathcal{V}(g_{i+1})$ not included in $\mathcal{V}_0^I \cup \cdots \cup \mathcal{V}_i^I$. The property holds at $i+1$.

For the second point, the inclusion of $\mathcal{V}_0^I \cup \mathcal{V}_1^I \cup \cdots \cup \mathcal{V}_n^I$ in $\mathcal{V}$ is a direct consequence of the property R_1. For the reverse inclusion, let $\mathcal{M}$ be an irreducible component of $\mathcal{V}$ of codimension $i > 0$. The variety $\mathcal{M}$ is included in $\mathcal{V}_0^R$, let j be the greatest index such that $\mathcal{M}$ is included in $\mathcal{V}_j^R$ and not in $\mathcal{V}_{j+1}^R$. Necessarily $\mathcal{M}$ is included in $\mathcal{V}_j^R \cap \mathcal{V}(g_{j+1})$. If $\mathcal{M}$ is included in one of the $\mathcal{V}_k^I$ for a given $k \leq j$ we are done, otherwise $\mathcal{M}$ is included in $\mathcal{V}_{j+1}^P$, and thus in $\mathcal{V}_{j+1}^I$. □

Let us now turn to the question of the multiplicities of the components of the $\mathcal{V}_i$. The next lemma is a weak local version of Bertini's first theorem, the complete proof we give of our assertion has been inspired by Bertini's original one as explained by Kleiman in [Kle97]. This result is also a local version of [KP94, Proposition 37] or [KPS01, Proposition 4.4].

Lemma 14 *For each i, $0 \leq i \leq n$, there exists a non zero polynomial Q_i in the ring $\overline{k}[(T_{k,l})_{1\leq k\leq i, 1\leq l\leq s}]$ of total degree bounded by $3(d+1)^{3i}$ such that: for all $t_{k,l}$ in k, $1 \leq k \leq i$, $1 \leq l \leq s$, $Q_i((t_{i+1,l})_{1\leq l\leq s}) \neq 0$ implies that for any irreducible component $\mathcal{M}$ of codimension i of the variety $\mathcal{V}_i$ the following alternative holds: $\mathcal{M}$ is included in $\mathcal{V}$ or $\mathcal{M}$ is reduced with respect to $g_1, \ldots, g_i$.*

Proof. First we prove that when the $t_{k,l}$ are generic the alternative holds.
In this proof R represents the polynomial ring
$$k[(T_{k,l})_{1\leq k\leq i, 1\leq l\leq s}],$$
K its field of functions, G_i the polynomials $T_{i,1}f_1 + \cdots + T_{i,s}f_s$, for $1 \leq i \leq n$. Let us now fix i in the range $0 \ldots n$, let r be $n - i$ and $\mathcal{M}$ be an irreducible component of the variety defined by $G_1, \ldots, G_i$ in $K[x_1, \ldots, x_n]$. Let us consider a geometric resolution of $\mathcal{M}$: let $y_1, \ldots, y_n$ be the variables in Noether position, $y = M^{-1}x$, u the primitive element, q the minimal polynomial and the parametrization given by $\rho y_j = v_j(u)$, $r+1 \leq j \leq n$. Let us assume that the component is multiple with respect to $G_1, \ldots, G_i$. This means that

there exist polynomials $H_1, \ldots, H_i$ in $K(y_1, \ldots, y_r)[u]$, such that q is prime with one of them, say H_k, and such that

$$\sum_{j=1}^{i} H_j \nabla (G_j \circ M)(y_1, \ldots, y_r, v_{r+1}(u)/\rho, \ldots, v_n(u)/\rho)$$

has all its entries in the ideal (q), the gradient being considered with respect to the variables $y_{r+1}, \ldots, y_n$. This implies that

$$\sum_{j=1}^{i} (H_j(G_j \circ M))(y_1, \ldots, y_r, v_{r+1}(u)/\rho, \ldots, v_n(u)/\rho)$$

belongs to the ideal generated by q^2. Let Y represent the vector $(v_{r+1}(u)/\rho, \ldots, v_n(u)/\rho)$. Differentiating the last expression with respect to $T_{k,l}$ we get that

$$\sum_{j=1}^{i} (\frac{\partial (H_j(y_1, \ldots, y_r, Y))}{\partial T_{k,l}})(G_j \circ M)(y_1, \ldots, y_r, Y)$$

$$+\sum_{j=1}^{i} H_j(y_1, \ldots, y_r, Y) \nabla (G_j \circ M)(y_1, \ldots, y_r, Y) \cdot \frac{\partial Y}{\partial T_{k,l}}$$

$$+\sum_{j=1}^{i} H_j(y_1, \ldots, y_r, Y)(\frac{\partial (G_j \circ M)}{\partial T_{k,l}})(y_1, \ldots, y_r, Y),$$

which is in (q). Since the two first terms are in (q), the last one is also in (q). But the last term A equals

$$H_k(y_1, \ldots, y_r, Y)(\frac{\partial G_k \circ M}{\partial T_{k,l}})(y_1, \ldots, y_r, Y)$$

and thus equals $H_k(y_1, \ldots, y_r, Y)(f_l \circ M)(y_1, \ldots, y_r, Y)$.
Since $H_k(y_1, \ldots, y_r, Y)$ is invertible modulo q, this implies that $f_l(y_1, \ldots, y_r, Y)$ belongs to (q). This property is true for all l, we deduce that $\mathcal{M}$ is included in $\mathcal{V}$, which achieves the first part of the proof.

We now prove constructively the existence of the desired polynomial Q_i. Let $\mathfrak{I}$ be the ideal generated by $(G_1, \ldots, G_i)$ in $k[(T_{k,l})_{1 \le k \le i, 1 \le l \le s}, x_1, \ldots, x_n]$ and $\mathcal{W}$ be the corresponding variety.

Let $\mathcal{M}$ be an irreducible component of $\mathcal{W}$ and $\mathfrak{J}$ be its annihilating ideal.

First we consider the case $\mathfrak{J} \cap R \neq (0)$: if π denotes the canonical projection $\overline{k}^{is} \times \overline{k}^n \to \overline{k}^{is}$, $\overline{\pi(\mathcal{W})}$ is contained in a hypersurface of degree bounded by the degree of $\mathcal{M}$. Thus there exists a non zero polynomial F in $\mathfrak{J} \cap R$ of total degree bounded by the degree of $\mathcal{M}$. If $t_{k,l}$ are such that $F(t_{k,l}) \neq 0$ then $\mathfrak{J} + ((T_{k,l} - t_{k,l})_{1 \le k \le i, 1 \le l \le s}) = (1)$.

Now consider the case $\mathfrak{J} \cap R = (0)$. There exists a geometric resolution of $\mathfrak{J}$ in $K[x_1, \ldots, x_n]$: let y_j, $1 \le j \le n$, be the variables in Noether position, u the primitive element, q its minimal polynomial and ρ, v_i be the parametrization such that $\rho y_j = v_j$ for $j \geq n-i+1$ and

$$\mathfrak{I}_{\phi\psi\rho} = (q(u), \rho y_{n-i+1} - v_{n-i+1}(u), \ldots, \rho y_n - v_n(u))_{\phi\psi\rho},$$

where ψ is the discriminant of q with respect to u and ϕ the leading coefficient of q. Let $t_{k,l}$ be such that $\phi\psi\rho(t) \neq 0$ then

$$\mathfrak{I} + ((T_{k,l} - t_{k,l})_{1\leq k\leq i, 1\leq l\leq s})$$

$$= (q_t(u), \rho_t y_{n-i+1} - v_{t,n-i+1}(u), \ldots, \rho_t y_n - v_{t,n}(u)),$$

where q_t, ρ_t and $v_{t,j}$ represent respectively the values of q, ρ and the v_j, when the $T_{k,l}$ are specialized at the $t_{k,l}$, for $1 \leq k \leq i$, $1 \leq l \leq s$. Since q_t is square free, the ideal $\mathfrak{I} + ((T_{k,l} - t_{k,l})_{1\leq k\leq i, 1\leq l\leq s})$ is reduced. Hence we let Q_i be one of the coefficients of $\phi\psi\rho$ with respect to the free variables $y_1, \ldots, y_{n-i}$.

It remains to give a bound on the degrees of ϕ, ψ and ρ. According to the bound given by Schost in [Sch00b, Proposition 1], the polynomial q can be chosen to be a polynomial in $R[y_1, \ldots, y_r][u]$ of total degree bounded by the degree of $\mathcal{M}$, and the total degree of ρ bounded by $\deg(\mathcal{M})^3$. Since the degree of $\mathcal{M}$ is bounded by $(d+1)^i$, the total degree of $\phi\psi\rho$ is bounded by $3(d+1)^{3i}$. □

We denote by R_2 the following property:

$\mathbf{R_2}$: for any irreducible component $\mathcal{M}$ of codimension i of the variety $\mathcal{V}_i$ the following alternative holds: $\mathcal{M}$ is included in $\mathcal{V}$ or $\mathcal{M}$ is reduced with respect to $g_1, \ldots, g_i$.

Proposition 56 *Let α be a strictly positive integer, and Ω be a set of points of k of cardinal $3\alpha(d+1)^{3n}$. A random choice of all the $t_{i,j}$, $1 \leq i \leq n+1$, $1 \leq j \leq s$, in $\Omega^{(n+1)s}$ leads to polynomials g_i satisfying the property R_2 with a probability at least $1-(n+1)/\alpha$.*

Proof. Zippel-Schwartz' zero test implies that a random choice of the $t_{i,j}$ in the set $\Omega^{(n+1)s}$ is not a root of the product of all the Q_i, for $1 \leq i \leq n$, with a probability at least $1-(n+1)/\alpha$. □

B.4 Algorithm

In this section we assume that we have chosen values of the $t_{i,j}$ such that the g_i satisfy the properties R_1 and R_2. In this situation we adapt the algorithm given in [GHM+98] in order to give a complete solution of the input system $f_1 = \cdots = f_s = 0$.

The algorithm is almost incremental. Indeed we would expect to compute resolutions of $\mathcal{V}^I_{i+1}$ and $\mathcal{V}^R_{i+1}$ from the one of $\mathcal{V}^R_i$. But doing this way yields a recursive dependency between the $\mathcal{V}^I_i$ and thus an exponential factor in the complexity. Our strategy consists in computing first a resolution of the system, not necessarily minimal, and at the end we deduce a minimal one. The first task is called the *incremental resolution* and the second one the *minimization*.

For each i, $0 \leq i \leq n-1$, we define the variety $\mathcal{V}_{i+1}^J$ composed of the components of $\mathcal{V}_i^R \cap \mathcal{V}(g_{i+1})$ which are included in the hypersurface defined by $g_{i+2} = 0$; in other words $\mathcal{V}_i^R \cap \mathcal{V}(g_{i+1})$ is the (minimal) union of the components of $\mathcal{V}_{i+1}^J$ and the ones of $\mathcal{V}_{i+1}^R$; moreover the variety $\mathcal{V}_{i+1}^I$ is included in $\mathcal{V}_{i+1}^J$. By convention we define $\mathcal{V}_0^J$ to be $\mathcal{V}_0^I$, so that the following set equalities hold:

$$\mathcal{V}_i = \mathcal{V}_0^J \cup \cdots \cup \mathcal{V}_i^J \cup \mathcal{V}_i^R,\ 0 \leq i \leq n,$$

$$\mathcal{V} = \mathcal{V}_0^J \cup \mathcal{V}_1^J \cup \cdots \cup \mathcal{V}_n^J.$$

Here is the outlook of the algorithm:
I. Incremental Resolution We compute geometric resolutions of each $\mathcal{V}_i^J$, it is incremental in i: at the ith step we compute resolutions of $\mathcal{V}_{i+1}^J$ and $\mathcal{V}_{i+1}^R$ from the one of $\mathcal{V}_i^R$. Each step divides into three parts:

1. *Compression* of the SLP of the geometric resolution of $\mathcal{V}_i^R$.

2. Computation of the *intersection* of $\mathcal{V}_i^R$ by g_{i+1}.

3. *Splitting* $\mathcal{V}_i^R \cap \mathcal{V}(g_{i+1})$ into $\mathcal{V}_{i+1}^J$ and $\mathcal{V}_{i+1}^R$.

II. Minimization We compute the $\mathcal{V}_i^I$ using the equality

$$(*) \qquad \mathcal{V}_i^I = \overline{\mathcal{V}_i^J \setminus \cup_{0 \leq j < i} \mathcal{V}_j^J}, \qquad 0 \leq i \leq n.$$

Note that this formula for the $\mathcal{V}_i^I$ is not recursive.

The next subsections are devoted to the presentation of the complexity model we use, the description of each elementary block of the algorithm and a proof of Theorem 9.

B.4.1 Complexity Model

We assume that k is an effective field for which each elementary operation (addition, substraction, multiplication, division and equality test) has cost 1. The multivariate polynomials we use in our algorithm are stored by means of Straight-Line Programs (SLP) without test nor division, we refer to Strassen [Str72], von zur Gathen [Gat86] or Heintz [Hei89] for precise definitions.

The only non-trivial point when dealing with this data structure is equality checking (or zero testing). One can perform this task by evaluating the SLP at sufficiently many points. This is formalized in terms of *correct test sequences* of points with coordinates from k according to a theorem due to Heintz and Schnorr [HS82b], which we recall for completeness.

Let D and L be two positive integers, and let us define the subset $W(D, n, L)$ of polynomials of $k[x_1, \ldots, x_n]$, of degree at most D, which can be coded by a SLP with at most L arithmetic operations. Furthermore given a subset Γ of k, a family $\gamma := \{\gamma_1, \ldots, \gamma_m\}$

(with $\gamma_i \in \Gamma^n$) of m points in k^n is called a *correct test sequence* for $W(D,n,L)$ if every polynomial in the latter vanishing on the points of γ is actually identically zero.

Theorem 10 *[HS82b] Given a subset Γ of k of cardinality $\#\Gamma = 2L(D+1)^2$, and a cardinality $m := 6(L+n)(L+n+1)$, the subset $\tau(D,n,L,\Gamma) \in \Gamma^{nm}$ of correct test sequences for $W(D,n,L)$ satisfies:*

$$\#\tau(D,n,L,\Gamma) \geq (\#\Gamma)^{nm}(1-(\#\Gamma)^{-\frac{m}{6}}).$$

In other words, if we have precomputed all the correct test sequences needed for a fixed range of applications we have a polynomial time algorithm for testing the nullity of our SLP. In this sense we say that our complexity model is *non-uniform.* Finally, we say that *all the operations (addition, multiplication, zero test) on SLP have a non-uniform complexity polynomial in their size and their number of variables.*

All our results remain valid in a uniform complexity model if we replace correct test sequences by the probabilistic test given in [FGS95] asserting that a SLP of size L can be tested to represent zero or not in time polynomial in L and n with a probability of failure uniformly bounded by $1/262144$.

B.4.2 Incremental Resolution

Let us now turn to the description of the elementary blocks constituting the core of the loop of the first part of the algorithm.

Compression

The compression technique is fundamental: it avoids the growth of the SLP representing the geometric resolution of the successive $\mathcal{V}_i^R$. It is based on a Newton-Hensel iterator. We adapt the result [GHM+98, Lemma 5] to our situation:

Lemma 15 *Let β be the size of a SLP encoding a geometric resolution of $\mathcal{V}_i^R$, for $i \leq n$, there exists a deterministic algorithm with a non-uniform complexity polynomial in $\beta Lsn\deg(\mathcal{V}_i^R)$ returning a SLP encoding a geometric resolution of $\mathcal{V}_i^R$ of size polynomial in $Lsn\deg(\mathcal{V}_i^R)$.*

Note that the size of the output is independent of β.

Intersection

The second step is also taken from [GHM+98, Proposition 15]. It consists in computing a resolution of the intersection of $\mathcal{V}_i^R$ by the hypersurface defined by the equation $g_{i+1} = 0$.

Lemma 16 *Let β be the size of a SLP of a geometric resolution of $\mathcal{V}_i^R$, there exists a deterministic algorithm with a non-uniform complexity polynomial in $\beta Lsnd\deg(\mathcal{V}_i^R)$ which computes a geometric resolution of $\mathcal{V}_i^R \cap \mathcal{V}(g_{i+1})$ encoded by a SLP of size polynomial in $\beta Lsnd\deg(\mathcal{V}_i^R)$.*

Splitting

From a resolution of $\mathcal{V}_i^R \cap \mathcal{V}(g_{i+1})$ we want to compute its decomposition into $\mathcal{V}_{i+1}^R$ and $\mathcal{V}_{i+1}^J$.

Lemma 17 *Let β be the size of the SLP encoding a geometric resolution of $\mathcal{V}_i^R \cap \mathcal{V}(g_{i+1})$, there exists a deterministic algorithm of non-uniform complexity polynomial in $\beta Lnsd\deg(\mathcal{V}_i^R \cap \mathcal{V}(g_{i+1}))$ which computes geometric resolutions of $\mathcal{V}_{i+1}^R$ and $\mathcal{V}_{i+1}^J$ encoded by a SLP of size polynomial in $\beta Lnsd\deg(\mathcal{V}_i^R \cap \mathcal{V}(g_{i+1}))$.*

Proof. We apply directly Proposition 57 of the Section B.5 with the equation $g_{i+2} = 0$.□

Let us introduce C to represent the expression $Lnsd\delta$. First we note that the degrees of all the varieties $\mathcal{V}_i^J$ appearing during the resolution process are bounded by $d\delta$. At each step of the loop, the compression produces a SLP of size bounded by $C^{\mathcal{O}(1)}$, after the intersection we also get a SLP bounded by $C^{\mathcal{O}(1)}$, and the same holds for the splitting. This induces that each step of the loop has a complexity at most polynomial in C and that the computed SLP of the $\mathcal{V}_i^J$ have size polynomial in C.

B.4.3 Minimization

Let us turn to the second part of the algorithm: from the resolutions of the $\mathcal{V}_i^J$ computed above, we want to compute resolutions of the $\mathcal{V}_i^I$. Using equality $(*)$, it is enough to remove the components of $\mathcal{V}_i^J$ which are included in $\mathcal{V}_0^J \cup \cdots \cup \mathcal{V}_{i-1}^J$. First we show that testing if a point belongs to an algebraic variety is equivalent to testing the nullity of the Chow form of the variety at the given point, then we give an algorithm for computing the Chow form of an equidimensional variety from a geometric resolution.

Let $\mathcal{W}$ be an r-equidimensional variety given by a geometric resolution: y denotes the variables in Noether position, $y = M^{-1}x$, u the primitive element, q the minimal polynomial, ρ and v_j the parametrizations, such that $\rho y_j = v_j(u)$, $j > r$. Let B represent $k[\mathcal{W}]$ and B' be $k(y_1, \dots, y_r) \otimes B$.

Let $\Lambda_{r+1}, \dots, \Lambda_n$ be new generic parameters, and denote by $\chi(y_1, \dots, y_r, T)$ the characteristic polynomial of the endomorphism of multiplication by the generic linear form $U = \Lambda_{r+1} y_{r+1} + \cdots + \Lambda_n y_n$, in B'. The polynomial χ is called the Chow form of $\mathcal{W}$, it is polynomial in the y_i and the Λ_i, monic in T, square free, its total degree in the y_i does not exceed the degree of $\mathcal{W}$ and its total degree in the y_i and Λ_i does not exceed $2\deg(\mathcal{W})$; moreover $\chi(u) = 0$ holds in B [GLS01, Corollary 2].

Lemma 18 *A point $P = (P_1, \dots, P_n)$ of $\overline{k}^n$ belongs to $\mathcal{W}$ (in the coordinates y) if and only if $\chi(P_1, \dots, P_r, U(P))$ is zero.*

Proof. First, if P belongs to $\mathcal{W}$ the Chow form vanishes at P. Reciprocally, let P be a point such that $\chi(P_1, \dots, P_r, U(P))$ is zero. Let π denote the canonical projection onto the free variables, using the theorem concerning the multiplicity of a specialization [Sam67, p.89] the specialization of χ leads to the formula

$$\chi(P_1, \dots, P_r, T) = \prod_{i=1}^{N} (T - Z_i)^{m_i},$$

where the Z_i are the point of $\pi^{-1}(P_1, \ldots, P_r)$ of multiplicity m_i. This implies that there exists i such that $U(P) = Z_i$; since the Z_i are independent of the Λ_j, necessarily P equals Z_i. □

Let β be the size of the SLP encoding the given geometric resolution of $\mathcal{W}$, we are able to compute χ quite easily as a SLP depending on the Λ_i and the free variables.

Lemma 19 *There exists a deterministic algorithm which computes a SLP of the Chow form χ of $\mathcal{W}$ of size polynomial in $n\beta\deg(\mathcal{W})$, the non-uniform complexity is polynomial in the same quantity.*

Proof. Let D be the dimension of B' and M_i be the multiplication matrices of the ρx_i for $r+1 \leq i \leq n$, in the basis $1, u, \ldots, u^{D-1}$. The matrix $\Lambda_{r+1}M_{r+1} + \cdots + \Lambda_n M_n$ is the multiplication matrix by ρU, let $\Psi(T)$ be its characteristic polynomial, then $\chi(T)$ equals $\Psi(\rho T)/\rho^D$, this division is exact. Since the matrices have size $D \times D$ and D is at most the degree of $\mathcal{W}$, their construction can be done with a complexity polynomial in $n\beta\deg(\mathcal{W})$. We use Berkowitz' algorithm [Ber84, Abd97] to compute the characteristic polynomial and "Vermeidung von Divisionen" [Str73] to compute the exact division, this yields the claimed complexity. □

We are now ready to explain the minimization method. Let $\mathcal{M}$ be a component of one the $\mathcal{V}_i^J$ for $i > 0$, testing whether $\mathcal{M}$ is included in one of the $\mathcal{V}_j^J$ with $j < i$ is equivalent to testing if a Zariski dense subset of $\mathcal{M}$ is included or not. Let y, u, q, ρ, v be the components of a geometric resolution of $\mathcal{M}$ and χ be the Chow form of $\mathcal{V}_j^J$ expressed in the variables y. The condition of the above lemma can be restated as: $\mathcal{M}$ is included in $\mathcal{V}_j^J$ if and only if $\chi(y_1, \ldots, y_{n-i}, v_{n-i+1}(u)/\rho, \ldots, v_n(u)/\rho)$ reduces to zero modulo q or not (in $k(y_1, \ldots, y_{n-i})[T]$).

Lemma 20 *Let β_j be the size of the SLP encoding a geometric resolution of $\mathcal{V}_j^J$, for $0 \leq j \leq n$, and D_j be the degree of $\mathcal{V}_j^J$, there exists a deterministic algorithm with non-uniform complexity polynomial in $n(\beta_1 D_1 + \cdots + \beta_i D_i)$ which computes a geometric resolution of $\mathcal{V}_i^I$ of size polynomial in the same quantity.*

Proof. With the above notations, χ_j, the Chow form of $\mathcal{V}_j^J$, can be computed in time polynomial in $n\beta_j D_j$. Proposition 57 applied on the product of the χ_j, $j < i$, yields the claimed complexity. □

Proof of Theorem 9 Let C still be $Lnsd\delta$. Since the output of the first part of the algorithm is polynomial in C and the degrees of all the $\mathcal{V}_i^J$ are bounded by $d\delta$, the minimization part is also polynomial in C. This proves the first part of Theorem 9. For the second part, it suffices to put together the probabilities that the election of the $t_{i,j}$ leads to a sequence of g_i satisfying the properties R_1 and R_2 of Propositions 54 and 56. □

B.5 Splitting a Resolution

Let $\mathcal{W}$ be an equidimensional variety of codimension i, given by a geometric resolution encoded by a SLP of size β. Let h be a polynomial in $k[x_1, \ldots, x_n]$, we explain in this section how one can split the given geometric resolution into two parts, one for the components of $\mathcal{W}$ included in the hypersurface defined by $h = 0$ and the other for the rest.

Let $r = n - i$ and the notations of the geometric resolution be:

- M represents an invertible matrix with entries in k, the coordinates in Noether position are $y = M^{-1}x$: the variables $y_1, \ldots, y_r$ are free.
- $u = \lambda_{r+1}y_{r+1} + \cdots + \lambda_n y_n$ is the primitive element, the λ_i are in k.
- q is the minimal polynomial of u, monic in u.
- $\rho y_j = v_j(u)$, for $r + 1 \leq j \leq n$ are the parametrizations, ρ is polynomial in the free variables and the v_j are polynomial in the free variables and u.

Let $A(y_1, \ldots, y_r, u)$ denote

$$h \circ M(y_1, \ldots, y_r, v_{r+1}(u)/\rho, \ldots, v_n(u)/\rho),$$

$q_1(u)$ be the greatest common divisor of A and q computed in $k(y_1, \ldots, y_r)[u]$, and let $q_2 = q/q_1$. Since q is monic and polynomial in the free variables, q_1 and q_2 are also monic and polynomial in the free variables. We set GR_i the geometric resolution defined by M, u, q_i, ρ and the remainders of the v_j modulo q_i, for $i = 1, 2$. The resolution GR_1 (respectively GR_2) is a geometric resolution of the components $\mathcal{W}_1$ (respectively $\mathcal{W}_2$) of $\mathcal{W}$ included (respectively not included) in the hypersurface defined by $h = 0$. Our aim is now to give a bound on the size of the SLP of the GR_i in function of β.

Proposition 57 *Let L_h be the size of the SLP encoding h, d_h a bound on the total degree of h, booth GR_1 and GR_2 can be computed by a deterministic algorithm in non-uniform time polynomial in $nL_hd_h\beta\deg(\mathcal{W})$. The sizes of the resulting SLP are polynomial in $nL_hd_h\beta\deg(\mathcal{W})$.*

Proof. Let H be the homogenized polynomial with respect to the new variable y_0 of $h \circ M$ can be computed in time polynomial in nL_hd_h, the resulting SLP having size also polynomial in nL_hd_h. Now let

$$B = H(\rho, \rho y_1, \ldots, \rho y_r, v_{r+1}(u), \ldots, v_n(u)),$$

B is proportional to A up to a ρ^{d_h} and its SLP is polynomial in $nL_hd_h\beta$. Using [GHM+98, Lemma 10] the greatest common divisor of B and q can be computed in time polynomial in $nL_h\beta d_h\beta\deg(\mathcal{W})$, this gives q_1 and q_2. $\square$

Annexe C

Fast Multivariate Power Series Multiplication in Characteristic Zero

Let k be a field of characteristic zero. We present a fast algorithm for multiplying multivariate power series over k truncated in total degree. Up to logarithmic factors, its complexity is optimal, i.e. *linear* in the number of coefficients of the series.

Joint work with Schost [LS01].

C.1 Introduction

Let k be a field of characteristic zero. We denote by S the multivariate power series ring in n variables $k[[x_1, \ldots, x_n]]$ and by $\mathfrak{m}$ its maximal ideal $(x_1, \ldots, x_n)$. For any positive integer d we write $\deg(\mathfrak{m}^{d+1})$ for the *degree* of the ideal $\mathfrak{m}^{d+1}$, that is the number of monomials in S which are not in $\mathfrak{m}^{d+1}$. It is well-known that

$$\deg(\mathfrak{m}^{d+1}) = \binom{d+n}{n}.$$

We view a power series f in S at precision $\mathfrak{m}^{d+1}$ as a vector in the k-algebra $S/\mathfrak{m}^{d+1}$; this algebra has dimension $\deg(\mathfrak{m}^{d+1})$.

In this article we give an asymptotically fast algorithm for multiplying two power series at precision $\mathfrak{m}^{d+1}$: the cost of one multiplication is linear in $\deg(\mathfrak{m}^{d+1})$ up to logarithmic factors. As for many other algorithms dedicated to fast multiplication, our method relies on multipoint evaluation and interpolation: this bring back the problem to univariate power series multiplication in $k[[t]]$ at precision t^{d+1} for which fast algorithms are known.

Applications. Our interest for power series multiplication originates from the field of polynomial system solving. In the second author's PhD thesis [Sch00b], the situation is as follows. We consider some polynomials $f_1, \ldots, f_m$ in $k(x_1, \ldots, x_n)[y_1, \ldots, y_m]$, and want to solve the system $f_1 = \cdots = f_m = 0$. The variables x_i play the role of parameters, and we are looking for formulas expressing the y_i in terms of these parameters. If the system is zero dimensional in the algebraic closure of $k(x_1, \ldots, x_n)$, we proceed the following way: we pick up a point $p_1, \ldots, p_n$ at random in k^n, we substitute the variables x_i by p_i in the system, solve this specialized system. We then lift the dependency of the solutions in the parameters in the formal power series ring $k[[x_1 - p_1, \ldots, x_n - p_n]]$, using a refined version of Newton's iterator, as proposed in [GLS01, Lec01]. Once we have reached a sufficient precision we recover the solutions thanks to a multivariate version of Padé's approximants [Sak90, FF92, Sch00b]. The lifting step is the bottleneck of this method and this is where we really need fast multivariate power series multiplication.

In a similar spirit, multiplication routines modulo $\deg(\mathfrak{m}^{d+1})$ are useful to treat systems of partial differential equations: roughly speaking, once a characteristic set of the system is known, the Taylor series expansions of non-singular solutions can be computed by successive approximations, which require arithmetic operations on valuated power series. This idea is presented for instance in [BLOP95] and [Pél97].

PREVIOUS WORK. To the best of our knowledge, this is the first time that the question of a fast multiplication algorithm is addressed in this context. Apart from the naive algorithm, with complexity quadratic in $\deg(\mathfrak{m}^{d+1})$, the best algorithm known up to now is hinted at in [BK77]: it relies on Kronecker's substitution [Kro82]. This method requires to compute modulo $(x_1^{d+1}, \ldots, x_n^{d+1})$ instead of $\mathfrak{m}^{d+1}$ and amounts to multiplying two univariate polynomials in degree $(2d)^n$. With respect to the size of the series $\deg(\mathfrak{m}^{d+1})$, the overhead is at least $c2^n n!$, for fixed n, $d \gg n$ and a positive constant c.

MAIN RESULT. All along this paper, our model of computation is the *computation tree*, that is the *arithmetic circuit* over k, with operations $(+, \times, \div)$ and branching. In short, we count the operations in k and the tests at unit cost; see [BCS97, Chapter 4.4] for a precise definition. For a positive integer d, the elements of $S/\mathfrak{m}^{d+1}$ will be represented by the vector of their coefficients in the monomial basis.

With these conventions, our main result is the following:

Theorem 11 *Let d be a positive integer, and let D denote $\deg(\mathfrak{m}^{d+1})$. Let f and g be two elements of $S/\mathfrak{m}^{d+1}$. In terms of operations in k, the product fg has complexity*

$$\mathcal{O}(D \log(D)^3 \log(\log(D))).$$

Our result is closely related to the algorithms for sparse [BOT88, Zip90] or dense [CKL89] multivariate polynomial multiplication, which achieve a linear complexity in the number of monomials in the output. All these results rely on a fast multipoint evaluation and interpolation scheme for multivariate polynomials, for a specific choice of sample points, namely power of prime numbers.

This idea was introduced in [GK87, Tiw87]. We recall this fundamental result in Lemma 21, following the presentation of [CKL89].

C.2 Proof of the main result

We first introduce some notation. In the following, C (resp. D) denotes the number of monomials in $n-1$ (resp. n) variables of degree at most d:

$$C := \binom{d+n-1}{n-1}, \qquad D := \binom{d+n}{n}.$$

The log function is the Neperian logarithm ($\log(e) = 1$).

Our proof is divided in three lemmas, the first of which is taken from [CKL89]. Its result is stated in terms of the function $\mathcal{U}(\delta)$, which denotes the complexity of the multiplication of two polynomials of degree δ in $k[t]$. Schönhage and Strassen proved in [SS71, Sch77] that $\mathcal{U}(\delta)$ belongs to $\mathcal{O}(\delta \log(\delta) \log(\log(\delta)))$.

Lemma 21 *Let f be a polynomial in $k[x_2, \ldots, x_n]$ of degree at most d. Let $(p_2, \ldots, p_n)$ be distinct prime numbers. For $i = 0, \ldots, C-1$, we denote by P_i the point $(p_2^i, \ldots, p_n^i)$. There exist computation trees of sizes $\mathcal{O}(\mathcal{U}(C)\log(C))$ which perform the following tasks:*

- **Multipoint evaluation:** *compute the values $f(P_0), \ldots, f(P_{C-1})$;*
- **Interpolation:** *recover f from the values $f(P_0), \ldots, f(P_{C-1})$.*

The choice of the points P_i reduces both problems to "transposed" versions of univariate multipoint evaluation and interpolation, for which asymptotically fast solutions exist. These solutions require to invert a Vandermonde matrix built upon all the monomials of degree at most d in $(p_2, \ldots, p_n)$. The key point is that in characteristic zero, all these monomials have pairwise distinct values.

On the basis of this result, the following lemma gives a first upper bound on the complexity of multivariate series multiplication, stated in terms of the function $\mathcal{U}$. In a second time, we will show that Schönhage-Strassen's multiplication scheme yields the claim of Theorem 11.

Lemma 22 *Let f and g be two elements of $S/\mathfrak{m}^{d+1}$. In terms of operations in k, the product fg has complexity*

$$\mathcal{O}\Big(d\,\mathcal{U}(C)\log(C) + \mathcal{U}(d)C\Big).$$

Proof. Let h be the product fg, and t a new variable. We denote by F, G, H the evaluations $F = f(t, x_2 t, \ldots, x_n t)$, $G = g(t, x_2 t, \ldots, x_n t)$ and $H = h(t, x_2 t, \ldots, x_n t)$. F, G and H belong to $k[x_2, \ldots, x_n][[t]]$, and can be written

$$\begin{aligned} F &= f_0 + f_1 t + \cdots + f_d t^d, \\ G &= g_0 + g_1 t + \cdots + g_d t^d, \\ H &= h_0 + h_1 t + \cdots + h_d t^d, \end{aligned}$$

where for all i, f_i, g_i and h_i belong to $k[x_2, \ldots, x_n]$ and have degrees at most i. The series F, G and H satisfy the equality $H = FG \bmod (t^{d+1})$.

The polynomial h can be recovered from H the following way: $h(x_1 t, \ldots, x_n t)$ is obtained by homogenizing each h_i in degree i with respect to the variable x_1, and the evaluation at $t = 1$ yields h.

Consequently, we focus on a fast way to compute H. For any $P = (p_2, \ldots, p_n)$ in k^{n-1}, we write F_P for the series $f_0(P) + f_1(P)t + \ldots + f_d(P)t^d$ in $k[[t]]/(t^{d+1})$, and similarly define G_P and H_P, so that the equality $H_P = F_P G_P \bmod (t^{d+1})$ holds. This leads to the following evaluation-interpolation scheme.

Algorithm. Given f and g in $S/\mathfrak{m}^{d+1}$, to compute $h = fg$ in $S/\mathfrak{m}^{d+1}$.

1. Compute F and G as defined above.
2. Compute (F_{P_i}, G_{P_i}) for C points $(P_0, \ldots, P_{C-1})$, with $P_i \in k^{n-1}$ for all i.
3. Compute the C products $H_{P_i} = F_{P_i} G_{P_i}$ in $k[[t]]/(t^{d+1})$.
4. Compute H by interpolating the polynomials $h_0, \ldots, h_d$.
5. Recover h.

Steps 1 and 5 can be performed by an arithmetic circuit of size linear in C. Following Lemma 21, we choose $P_i = (p_2^i, \ldots, p_n^i)$, for distinct prime numbers $(p_2, \ldots, p_n)$ and examine the cost of steps 2, 3 and 4 for this specific choice.

- For each j in $0, \ldots, d$, the values $f_j(P_0), \ldots, f_j(P_{C-1})$ and $g_j(P_0), \ldots, g_j(P_{C-1})$ can be computed within $\mathcal{O}(\mathcal{U}(C)\log(C))$ operations in k using the algorithm for fast multipoint evaluation given in Lemma 21. This yields an overall bound of $\mathcal{O}(d\,\mathcal{U}(C)\log(C))$ operations for step 2.
- For each i in $0, \ldots, C-1$, H_{P_i} is obtained by an univariate series product in $k[[t]]/(t^{d+1})$, which takes $\mathcal{O}(\mathcal{U}(d))$ operations; step 3 then requires $\mathcal{O}(\mathcal{U}(d)C)$ operations.
- For each j in $0, \ldots, d$, the interpolation of h_j requires $\mathcal{O}(\mathcal{U}(C)\log(C))$ operations, using the second result given in Lemma 21. The interpolation of all h_j then takes $\mathcal{O}(d\,\mathcal{U}(C)\log(C))$ operations.

□

The result of Lemma 22 is given in terms of the number of monomials C, whereas our objective is a bound in terms of the quantity D. The last lemma answers this question.

Lemma 23 *For all $d \geq 0, n \geq 1$, the following inequality holds:*

$$\frac{dC}{D} = \frac{nd}{n+d} \leq \log(D).$$

Proof. We rewrite D into $\frac{(n+1)\dots(n+d)}{d!}$. Then we fix d, and introduce the functions $u : n \mapsto \frac{nd}{n+d}$ and $v : n \mapsto \log\frac{(n+1)\dots(n+d)}{d!}$. For $n = 1$, the inequality to prove reads $\frac{d}{1+d} \leq \log(1+d)$, which is true. It is immediate to check that $u'(n) \leq v'(n)$ for all n, which implies that $u(n) \leq v(n)$ for all $n \geq 1$. $\square$

We are now ready to prove Theorem 11. We replace $\mathcal{U}(C)$ by $\mathcal{O}(C\log(C)\log(\log(C)))$ in the complexity of Lemma 22; then, according to the above lemma we bound C by $D\log(D)/d$. This yields a complexity in:

$$\mathcal{O}\Big(D\log(D)\Big(\log(D)+\log(\log(D))\Big)^2\log\Big(\log(D)+\log(\log(D))\Big)$$
$$+D\log(D)\log(d)\log(\log(d))\Big).$$

As for the second term we use $d \leq D$, therefore:

$$d\,\mathcal{U}(C)\log(C)+\mathcal{U}(d)C \in \mathcal{O}(D\log(D)^3\log(\log(D))).$$

This concludes the proof of Theorem 11.

C.3 Conclusion

The problem of fast computation with multivariate power series modulo any ideal $\mathfrak{I}$, not necessarily a power of the maximal ideal, is still open. A first question in this direction is the cost of the multiplication modulo the ideal $(x_1^{d+1}, \dots, x_n^{d+1})$. In this situation, our algorithm requires precision $\mathfrak{m}^{nd+1}$; this yields a complexity in $\mathcal{O}((ed)^n)$, up to logarithmic factors. We do not improve the best complexity result, which is in $\mathcal{O}(\mathcal{U}(2d)^n)$ using Kronecker's substitution.

Moreover our result is stated in terms of arithmetic complexity. It is not immediate to design an efficient implementation of this algorithm. For instance, if we were on $k = \mathbb{Q}$, we would certainly want to use multimodular and Chinese remainder techniques in order to avoid the growth of the integers in the intermediate computations; this would require to extend our result to finite fields.

This naturally opens the more general question of fast computation with multivariate power series over any ring. The point is to extend Lemma 21; the main difficulty is to choose points in the base ring such that distinct monomials take distinct values on these points. A first result in this direction, based on combinatorial arguments, is given in [Zip90].

Bibliographie

[Abd97] J. Abdeljaoued. *Algorithmes rapides pour le calcul du polynôme caractéristique.* PhD thesis, Université de Franche-Comté, Besançon, France, 1997.

[ABRW96] M. E. Alonso, E. Becker, M.-F. Roy, and T. Wörmann. Zeroes, multiplicities and idempotents for zerodimensional systems. In *Algorithms in Algebraic Geometry and Applications. Proceedings of MEGA'94*, volume 142 of *Progress in Mathematics*, pages 1–15. Birkhäuser, 1996.

[ADS98] M. Almeida, L. D'Alfonso, and P. Solernó. On the degrees of bases of free modules over a polynomial ring. *Mathematische Zeitschrift*, pages 1–24, 1998.

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley, 1974.

[ALM99] Ph. Aubry, D. Lazard, and M. Moreno Maza. On the theories of triangular sets. *Journal of Symbolic Computation*, 28(1,2):45–124, 1999.

[AS95] I. Armendáriz and P. Solernó. On the computation of the radical of polynomial complete intersection ideals. In G. Cohen, M. Giusti, and T. Mora, editors, *Applied Algebra, Algebraic Algorithms and Error Correcting Codes. Proceedings of AAECC-11*, volume 948 of *Lecture Notes in Computer Science*, pages 106–119. Springer, 1995.

[Ass94] A. Assi. On flatness of generic projections. *Journal of Symbolic Computation*, 18(5):447–462, 1994.

[BB01] A. Waissbein B. Bank, L. Lehmann. Kronecker goes to waveletland. http://www-pool.mathematik.hu-berlin.de/~llehmann/, March 2001.

[BC89] G. Butler and J. Cannon. Cayley version 4: The user language. In *Proceedings of ISSAC'88*, volume 358 of *Lecture Notes in Computer Science*, pages 456–466. New York: Springer, July 1989.

[BC90] G. Butler and J. Cannon. The design of Cayley, a language for modern algebra. In A. Miola, editor, *Design and Implementation of Symbolic Computation Systems*, volume 429 of *Lecture Notes in Computer Science*, pages 10–19. New York: Springer, July 1990.

[BC95] W. Bosma and J. Cannon. Handbook of Magma functions. Sydney: School of Mathematics and Statistics, University of Sydney, 1995.

[BCM94] W. Bosma, J. Cannon, and J. Matthews. Programming with algebraic structures: design of the Magma language. In M. Giesbrecht, editor, *Proceedings of ISSAC'94*. ACM, 1994.

[BCP97] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system I: The user language. *Journal of Symbolic Computation*, 24, 1997.

[BCS97] P. Bürgisser, M. Clausen, and M. A. Shokrolahi. *Algebraic Complexity Theory*. Springer, 1997.

[BDG95] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural complexity. I.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, second edition, 1995.

[Ber75] D. N. Bernshtein. The number of roots of a system of equations. *Funktional Analalysis and its Applications*, 9(3), 1975.

[Ber84] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18:147–150, 1984.

[BF91] G. Björck and G. Froberg. A faster way to count the solution of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots. *Journal of Symbolic Computation*, 12(3):329–336, 1991.

[BGW95] P. A. Broadbery, T. Gómez Díaz, and S. M. Watt. On the implementation of dynamic evaluation. In *ISSAC'95*, pages 77–84, 1995.

[BGY80] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun. Fast solution of Toeplitz systems of equations and computation of Padé approximants. *Journal of Algorithms*, 1(3):259–295, September 1980.

[BH74] J. Bunch and J. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computations*, 28:231–236, 1974.

[Bjö90] G. Björck. Functions of modulus 1 on Z_n whose Fourier transforms have constant modulus, and "cyclic n-roots". In *Recent advances in Fourier analysis and its applications (Il Ciocco, 1989)*, volume 315 of *NATO Advance Science Institutes Series C: Mathematical and Physical Sciences*, pages 131–140. Kluwer Academic Publishers, Dordrecht, 1990.

[BK77] R. P. Brent and H. T. Kung. Fast algorithms for composition and reversion of multivariate power series. In *Proceedings of the Conference on Theoretical Computer Science, University of Waterloo, Ontario, Canada*, pages 149–158, 1977.

[BL94] B. Beckermann and G. Labahn. A uniform approach for the fast computation of matrix-type Padé approximants. *SIAM Journal on Matrix Analysis and Applications*, 15(3):804–823, July 1994.

[BLOP95] F. Boulier, D. Lazard, F. Ollivier, and M. Petitot. Representation for the radical of a finitely generated differential ideal. In *Proceedings of ISSAC'95*, pages 158–166. ACM, 1995.

[BM72] A. Borodin and J. Munro. *The Computational Complexity of Algebraic and Numeric Problems*. Elsevier, 1972.

[BMP98] D. Bondyfalat, B. Mourrain, and V. Y. Pan. Controlled iterative methods for solving polynomial systems. In *Proceedings of ISSAC'98*, pages 252–259. ACM, 1998.

[BOT88] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *20th Annual ACM Symposium Theory of Computing*, 1988.

[BP94] D. Bini and V. Pan. *Polynomial and matrix computations.* Progress in theoretical computer science. Birkhäuser Boston-Basel-Berlin, 1994.

[BS83] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.

[BS88] D. Bayer and M. Stillman. On the complexity of computing syzygies. *Journal of Symbolic Computation*, 6(2–3), October–December 1988.

[BSS01] A. Bostan, B. Salvy, and É. Schost. Fast algorithms for zero-dimensional polynomial systems using duality. Research Report 4291, Institut National de Recherche en Informatique et en Automatique, 2001. 24 pages.

[BW93] T. Becker and V. Weispfenning. *Groebner bases: a computational approach to commutative algebra*, volume 141 of *Graduate Texts in Mathematics: readings in mathematics.* Springer, 1993.

[Can88] J. Canny. Some algebraic and geometric problems in PSPACE. In *Proceedings 20 ACM STOC*, pages 460–467, 1988.

[Cas01] D. Castro. *Sobre la complejidad de la representación de variedades algebraicas.* PhD thesis, Universidad de Cantabria, Departemento de Matemáticas, Estadística y Computación, July 2001.

[CDPR98] L. Correnson, E. Duris, D. Parigot, and G. Roussel. Composition symbolique. In *Journées Francophones des Langages Applicatifs*, 1998.

[CE93] J. Canny and I. Emiris. An efficient algorithm for the sparse mixed resultant. In *Applied Algebra, Algebraic Algorithms and Error Correcting Codes. Proceedings of AAECC-10*, volume 673 of *Lecture Notes in Computer Science*, pages 89–104. Springer-Verlag, 1993.

[CG82] A. L. Chistov and D. Y. Grigoriev. Polynomial-time factoring of multivariable polynomials over a global field. LOMI preprint E-5-82, Steklov Institute, Leningrad, 1982.

[CG83] A. L. Chistov and D. Y. Grigoriev. Subexponential time solving systems of algebraic equations. LOMI preprint E-9-83, E-10-83, Steklov Institute, Leningrad, 1983.

[CGH88] L. Caniglia, A. Galligo, and J. Heintz. Borne simple exponentielle pour les degrés dans le théorème des zéros sur un corps de caractéristique quelconque. *Comptes Rendus l'Académie des Sciences de Paris*, 307:255–258, 1988.

[Chi96] A. L. Chistov. Polynomial-time computation of the dimension of algebraic varieties in zero-characteristic. *Journal of Symbolic Computation*, 22(1):1–25, July 1996.

[Chi97] A. L. Chistov. Polynomial-time computation of the dimensions of components of algebraic varieties in zero-characteristic. In A. M. Cohen and M.-F. Roy, editors, *Proceedings of MEGA'96*, volume 117–118 of *Progress in Mathematics*, pages 145–175. Birkhäuser, 1997.

[CHMP01] D. Castro, K. Hägele, J. E. Morais, and L. M. Pardo. Kronecker's and newton's approaches to solving: A first comparison. *Journal of Complexity*, 17(1):212–303, March 2001.

[CKL89] J. Canny, E. Kaltofen, and Y. Lakshman. Solving systems of non-linear polynomial equations faster. In *Proceedings of ISSAC'89*, pages 121–128. ACM, 1989.

[CKM97] S. Collart, M. Kalkbrener, and D. Mall. Converting bases with the Gröbner walk. *Journal of Symbolic Computation*, 24:465–469, 1997.

[CLO97] D. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms.* Undergraduate Texts in Mathematics. Springer-Verlag, New York, second edition, 1997. An introduction to computational algebraic geometry and commutative algebra.

[CMPS01] D. Castro, J. L. Montaña, L. M. Pardo, and J. San Martín. On the uniform distribution of rational inputs with respect to condition numbers of numerical analysis. Manuscript, Universidad de Cantabria, Santander, Spain, March 2001.

[CP96] J. Cannon and C. Playoust. Magma: A new computer algebra system. *Euromath Bulletin*, 2(1):113–144, 1996.

[Csa76] L. Csanky. Fast parallel matrix inversion algorithms. *SIAM Journal of Computing*, 5(4):618–623, 1976.

[CW90] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, March 1990.

[Dav86] J. Davenport. Looking at a set of equations. Technical Report 87-06, Bath Computer Science, 1986.

[DD89] C. Dicrescenzo and D. Duval. Symbolic and algebraic computation. *Lecture Note in Computer science, Springer Verlag*, 358:440–446, 1989.

[DDD85] J. Della Dora, C. Dicrescenzo, and D. Duval. About a new method for computing in algebraic number fields. In *EuroCal'85*, volume 204 of *Lecture Notes in Computer Science*, pages 289–290, 1985.

[Del99] S. Dellière. *Triangularisation de systèmes constructibles — Application à l'évaluation dynamique.* PhD thesis, Université de Limoges, 1999.

[Día94] T. Gómez Díaz. *Quelques applications de l'évaluation dynamique.* PhD thesis, Université de Limoges, 1994.
ftp://medicis.polytechnique.fr/pub/src/dynamic_evaluation.

[Dix82] J. Dixon. Exact solution of linear equations using p-adic expansions. *Numerische Mathematik*, 40:137–141, 1982.

[DR94a] D. Duval and J.-C. Reynaud. Sketches and computation - I : Basic definition and static evaluation. In *Mathematical Structures in Computer Science*, volume 4. Cambridge University Press, 1994.

[DR94b] D. Duval and J.-C. Reynaud. Sketches and computation - II : Dynamic evaluation and applications. In *Mathematical Structures in Computer Science*, volume 4. Cambridge University Press, 1994.

[Dre77] F. J. Drexler. Eine Methode zur Berechnung sämtlicher lösungen von Polynomgleichungssystemen. *Numerische Mathematik*, 29(1):45–58, 1977.

[DST93] J. H. Davenport, Y. Siret, and E. Tournier. *Calcul formel.* Masson, Paris, second edition, 1993.

[Duv87] D. Duval. *Diverses questions relatives au calcul formel avec des nombres algebriques.* PhD thesis, Université de Grenoble 1, 1987.

[Duv89] D. Duval. Simultaneous computation in fields of arbitrary characteristic. In *Computer and mathematics 89*, pages 321–326. Springer-Verlag, 1989.

[Duv94] D. Duval. Algebraic numbers: an example of dynamic evaluation. *Journal of Symbolic Computation*, 18:429–445, 1994.

[Duv95] D. Duval. Évaluation dynamique et clôture algébrique en Axiom. *Journal of pure and Applied Algebra*, 99:267–295, 1995.

[EC95a] I. Emiris and J. Canny. Efficient incremental algorithms for the sparse resultant and mixed volume. *Journal of Symbolic Computation*, 2:117–149, 1995.

[EC95b] I. Emiris and J. F. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. *Journal of Symbolic Computation*, 20(2), August 1995.

[Eis95] David Eisenbud. *Commutative algebra with a view toward algebraic geometry*, volume 150 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, 1995.

[EM99a] M. Elkadi and B. Mourrain. A new algorithm for the geometric decomposition of a variety. In *Proceedings of ISSAC'99.* ACM, 1999.

[EM99b] I. Emiris and B. Mourrain. Matrices in elimination theory. *Journal of Symbolic Computation*, 28:3–44, 1999.

[Emi94] I. Emiris. *Sparse Elimination and Applications in Kinematics.* PhD thesis, Computer Science Division, U. C. Berkley,
http:www.inria.fr/saga/emiris, 1994.

[EP98] I. Emiris and Y. V. Pan. Symbolic and numeric methods for exploiting structure in constructing resultant matrices. *Journal of Symbolic Computation*, 1998. Special Issue on Symbolic-Numeric Algebra for Polynomials.

[Fau] J.-C Faugère. FGb.
https://calfor.lip6.fr/.

[Fau94] J.-C. Faugère. *Solution des systèmes d'équations polynomiales.* PhD thesis, Université Paris 6, 1994.

[Fau95] J.-C. Faugère. *GB Reference Manual.* LIP6, Laboratoire CalFor, 1995.
http://www-calfor.lip6.fr/~jcf.

[Fau97] J.-C. Faugère. GB: State of GB + tutorial. LITP, 1997.

[Fau99] J.-C. Faugère. A new efficient algorithm for computing Groebner bases (F_4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.

[Fau00] J.-C. Faugère. How my computer find all the solutions of cyclic 9. Technical report, LIP6,
http://www.lip6.fr/reports/lip6.2000.007.html, 2000.

[Fau01] J.-C. Faugère. Nouveaux algorithmes pour la résolution des systèmes algébriques et applications. Séminaire EUCLIDE, March 2001.

[FF92] P. Fitzpatrick and J. Flynn. A Gröbner basis technique for Padé approximation. *Journal of Symbolic Computation*, 13:133–138, 1992.

[FGLM93] J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.

[FGS95] N. Fitchas, M. Giusti, and F. Smietanski. Sur la complexité du théorème des zéros. In J. Guddat, editor, *Approximation and Optimization in the Caribbean II, Proceedings 2nd Int. Conf. on Non-Linear Optimization and Approximation*, volume 8 of *Approximation and Optimization*, pages 247–329. Peter Lange Verlag, Frankfurt am Main, 1995.

[Fid72] C. M. Fiduccia. Polynomial evaluation via the division algorithm: The fast Fourier transform revisited. In *Conference Record, Fourth Annual ACM Symposium on Theory of Computing*, pages 88–93, Denver, Colorado, 1–3 May 1972.

[Fid87] C. M. Fiduccia. A rational view of the fast Fourier transform. In *25th Allerton Conference on Communication, Control and Computing*, 1987.

[FR98] J.-C. Faugère and F. Rouillier. Mupad interface for gb/realsolving. http://www.loria.fr/~rouillie/RSDoc/mupdoc/mupdoc.html, 1998.

[Ful84] W. Fulton. *Intersection Theory.* Number 3 in Ergebnisse der Mathematik. Springer, second edition, 1984.

[Gat86] J. von zur Gathen. Parallel arithmetic computations: a survey. In B. Rovan J. Gruska and J. Wiedermann, editors, *Proceedings of the 12th Symposium on Mathematical Foundations of Computer Science*, volume 233 of *Lecture Notes in Computer Science*, pages 93–112, Bratislava, Czechoslovakia, August 1986. Springer.

[Gau00] P. Gaudry. *Algorithmique des courbes hyperelliptiques et applications à la cryptologie.* PhD thesis, École polytechnique, 2000.

[GCL94] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for computer algebra.* Kluwer Academic Publishers, 1994.

[GFT00] P. Gianni, E. Fortuna, and B. Trager. Degree reduction under specialization. Exposé à MEGA 2000, 2000.

[GG99] J. von zur Gathen and J. Gerhard. *Modern computer algebra.* Cambridge University Press, 1999.

[GH91] M. Giusti and J. Heintz. Algorithmes - disons rapides - pour la décomposition d'une variété algébrique en composantes irréductibles et équidimensionelles. In T. Mora and C. Traverso, editors, *Proceedings of MEGA'90*, volume 94 of *Progress in Mathematics*, pages 169–194. Birkhäuser, 1991.

[GH93] M. Giusti and J. Heintz. La détermination des points isolés et de la dimension d'une variété algébrique peut se faire en temps polynomial. In D. Eisenbud and L. Robbiano, editors, *Computational Algebraic Geometry and Commuta-*

tive Algebra, volume XXXIV of *Symposia Matematica*, pages 216–256. Cambridge University Press, 1993.

[GHH+97] M. Giusti, K. Hägele, J. Heintz, J. E. Morais, J. L. Montaña, and L. M. Pardo. Lower bounds for Diophantine approximation. In *Proceedings of MEGA'96*, volume 117,118, pages 277–317. Journal of Pure and Applied Algebra, 1997.

[GHL+00] M. Giusti, K. Hägele, G. Lecerf, J. Marchand, and B. Salvy. Computing the dimension of a projective variety: the projective Noether Maple package. *Journal of Symbolic Computation*, 30(3):291–307, September 2000.

[GHM+98] M. Giusti, J. Heintz, J. E. Morais, J. Morgenstern, and L. M. Pardo. Straight-line programs in geometric elimination theory. *Journal of Pure and Applied Algebra*, 124:101–146, 1998.

[GHMP95] M. Giusti, J. Heintz, J. E. Morais, and L. M. Pardo. When polynomial equation systems can be solved fast? In G. Cohen, M. Giusti, and T. Mora, editors, *Applied Algebra, Algebraic Algorithms and Error Correcting Codes, Proceedings AAECC-11*, volume 948 of *Lecture Notes in Computer Science*, pages 205–231. Springer, 1995.

[GHMP97] M. Giusti, J. Heintz, J. E. Morais, and L. M. Pardo. Le rôle des structures de données dans les problèmes d'élimination. *Comptes Rendus de l'Académie des Sciences de Paris*, 325:1223–1228, 1997.

[GHS93] M. Giusti, J. Heintz, and J. Sabia. On the efficiency of effective Nullstellensätze. *Computational Complexity*, 3:56–95, 1993.

[Gia87] P. Gianni. Properties of Gröbner bases under specializations. In *European Conference on Computer Algebra*, number 378 in Lecture Notes in Computer Science, pages 293–297, 1987.

[Giu88] M. Giusti. Combinatorial dimension theory of algebraic varieties. *Journal of Symbolic Computation*, 6(2-3):249–265, October-December 1988. Special issue on computational aspects of commutative algebra.

[GK87] D. Y. Grigoriev and M. Karpinski. The matching problem for bipartite graphs with polynomially bounded permanents is in NC. In *Proceedings of the 28th IEEE Symposium on the Foundations of Computer Science*, pages 166–172, 1987.

[GKZ94] I. M. Gel'fand, M. M. Kapranov, and A. V. Zelevinski. *Discriminants, Resultants and Multidimensional Determinants*. Birkhäuser, Boston, 1994.

[GLS01] M. Giusti, G. Lecerf, and B. Salvy. A Gröbner free alternative for polynomial system solving. *Journal of Complexity*, 17(1):154–211, 2001.

[GM89] P. Gianni and T. Mora. Algebraic solution of systems of polynomial equations using Gröbner bases. In *Applied Algebra, Algebraic Algorithms and Error Correcting Codes, Proceedings of AAECC-5*, volume 356 of *Lecture Notes in Computer Science*, pages 247–257. Springer, 1989.

[GPS01] G.-M. Greuel, G. Pfister, and H. Schönemann. Singular 2.0. A Computer Algebra System for Polynomial Computations, Centre for Computer Algebra, University of Kaiserslautern, 2001.
http://www.singular.uni-kl.de.

[GR01] A. Galligo and D. Rupprecht. Semi-numerical determination of irreducible branches of a reduced space curve. In *Proceedings of ISSAC'2001*. ACM, 2001.

[Grä93] H.-G. Gräbe. On lucky primes. *Journal of Symbolic Computation*, 15:199–209, 1993.

[GS00] P. Gaudry and É. Schost. Invariants des quotients de la Jacobienne d'une courbe de genre 2. Technical report, École polytechnique, http://www.gage.polytechnique.fr/schost.html, 2000.

[GSZ95] C. Gomez, B. Salvy, and P. Zimmermann. *Calcul formel : mode d'emploi. Exemples en Maple*. Masson, 1995. ISBN 2-225-84780-0. 328 pages.

[GZ79] C. Garcia and W. Zangwill. Finding all solutions to polynomial systems and other systems of equations. *Mathematic Programming*, 16:159–176, 1979.

[Haa97] U. Haagerup. Orthogonal maximal abelian $*$-subalgebras of the $n \times n$ matrices and cyclic n-roots. In S. Doplicher et al., editor, *Operator Algebras and Quantum Field Theory*, pages 296–322. International Press, 1997.

[Häg98] K. Hägele. *Intrinsic height estimates for the Nullstellensatz*. PhD thesis, Universidad de Cantabria, Santander, 1998.

[Hei83] J. Heintz. Definability and fast quantifier elimination in algebraically closed fields. *Theoretical Computer Science*, 24(3):239–277, 1983.

[Hei89] J. Heintz. On the computational complexity of polynomials and bilinear mappings. A survey. In *Applied Algebra, Algebraic Algorithms and Error Correcting Codes, Proceedings of AAECC-5*, volume 356 of *Lecture Notes in Computer Science*, pages 269–300. Springer, 1989.

[Hil92] D. Hilbert. Ueber die Irreducibilität ganzer rationaler Functionen mit ganzzahligen Coefficienten. *Journal für die Reine und Angewandte Mathematik*, 110:104–129, 1892.

[HKP+00] J. Heintz, T. Krick, S. Puddu, J. Sabia, and A. Waissbein. Deformation techniques for efficient polynomial equation solving. *Journal of Complexity*, 16(1), 2000.

[HM97] K. Hägele and J. L. Montaña. Polynomial random test for the equivalence problem of integers given by arithmetic circuits. Preprint 4/97 Depto. Matemáticas, Universidad de Cantabria, Santander, Spain, January 1997.

[HMPS97] K. Hägele, J. E. Morais, L. M. Pardo, and M. Sombra. On the intrinsic complexity of the arithmetic Nullstellensatz. In J. Heintz, G. Matera, and R. Wachenschauzer, editors, *Proceedings of TERA'97*, Córdoba, Argentina, September 1997. Universidad de Córdoba, Fa.M.A.F, http://tera.medicis.polytechnique.fr/.

[HMPS00] K. Hägele, J. E. Morais, L. M. Pardo, and M. Sombra. On the intrinsic complexity of the arithmetic Nullstellensatz. *Journal of Pure And Applied Algebra*, 146(2):103–183, 2000.

[HMW01] J. Heintz, G. Matera, and A. Waissbein. On the time-space complexity of geometric elimination procedures. *Applicable Algebra in Engineering, Communication and Computing*, 11(4):239–296, 2001.

[Hoe00] M. van Hoeij. Factoring polynomials and the knapsack problem. To appear in Journal of Number Theory, August 2000.

[HRS90] J. Heintz, M.-F. Roy, and P. Solernó. Sur la complexité du principe de Tarski-Seidenberg. *Bulletin de la Société Mathématique de France*, 118:101–126, 1990.

[HS82a] J. Heintz and C. P. Schnorr. Testing polynomials which are easy to compute. In *Logic and Algorithmic (Zürich, 1980)*, volume 30 of *Monographie de l'Enseignement Mathématique*, pages 237–254, 1982.

[HS82b] J. Heintz and C. P. Schnorr. Testing polynomials which are easy to compute. In *Logic and Algorithmic*, volume 30 of *Monographie de l'Enseignement Mathématique*, pages 237–254, 1982.

[HSS00] J. Hubbard, D. Schleicher, and S. Sutherland. How to find all roots of complex polynomials by newton's method, March 2000. http://www.math.sunysb.edu/~scott/.

[JS00] G. Jeronimo and J. Sabia. Probabilistic equidimensional decomposition. *Comptes rendus de l'Académie des sciences de Paris*, 331(1), 2000.

[Kal87] M. Kalkbrener. Solving systems of algebraic equations by using Gröbner bases. In *European Conference on Computer Algebra*, number 378 in Lecture Notes in Computer Science, pages 282–292, 1987.

[Kal91] M. Kalkbrener. *Three contributions to elimination theory*. PhD thesis, Kepler University, Linz, 1991.

[Kal97] M. Kalkbrener. On stability of Gröbner bases under specializations. *Journal of Symbolic Computation*, 24(1):51–58, 1997.

[KFF88] H. Kobayashi, T. Fujise, and A. Furukawa. Solving systems of algebraic equations by general elimination method. *Journal of Symbolic Computation*, 5:303–320, 1988.

[Kle97] S. L. Kleiman. Bertini and his two fundamental theorems. In *Rendiconti del circolo matematico di Palmero*, 1997.

[Knu73] D. E. Knuth. *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*. Addison-Wesley, Reading, MA, second edition, 1973.

[Koi97] P. Koiran. Randomized and deterministic algorithms for the dimension of algebraic varieties. In *Proceedings of the 38th Annual IEEE Computer Society Conference on Foundations of Computer Science (FOCS)*. ACM, 1997.

[Kön03] J. König. *Einleitung in die allgemeine Theorie der algebraischen Gröszen*. Druck und Verlag von B. G. Teubner, Leipzig, 1903.

[KP94] T. Krick and L. M. Pardo. Une approche informatique pour l'approximation diophantienne. *Comptes rendus de l'Académie des sciences de Paris*, 318(1):407–412, 1994.

[KP96] T. Krick and L. M. Pardo. A computational method for Diophantine approximation. In L. González-Vega and T. Recio, editors, *Algorithms in Algebraic Geometry and Applications. Proceedings of MEGA'94*, volume 143 of *Progress in Mathematics*, pages 193–254. Birkhäuser Verlag, 1996.

[KPS01] T. Krick, L. M. Pardo, and M. Sombra. Sharp estimates for the arithmetic Nullstellensatz. *Duke Mathematical Journal*, 109(3), 2001.

[Kro81] L. Kronecker. Zur Theorie der Elimination einer Variabeln aus zwei algebraischen Gleichungen. *Monatsberichte der Königlich Preussischen Akademie der Wissenschaften, Berlin*, pages 535–600, 1881.

[Kro82] L. Kronecker. Grundzüge einer arithmetischen Theorie der algebraischen Grössen. *Journal für die Reine und Angewandte Mathematik*, 92:1–122, 1882.

[Kru13] E. Kruppa. Zur Ermittlung eines Objektes aus zwei Perpesktiven mitinnerer Orientierung. *Sitz.-Ber. Akad. Wiss., Wien. Math. Naturw. Kl.*, 1913.

[KS94] M.-H. Kim and S. Sutherland. Polynomial root-finding algorithms and branched covers. *SIAM Journal on Computing*, 23(2):415–436, April 1994.

[KS01a] H. Koy and C. P. Schnorr. Segment LLL-reduction of lattice bases. In *Cryptography and Lattices Conference (CalC) 2001*, http://www.mi.informatik.uni-frankfurt.de/research/papers.html, 2001.

[KS01b] H. Koy and C. P. Schnorr. Segment lll-reduction with floating point orthogonalization. In *Cryptography and Lattices Conference (CalC) 2001*, http://www.mi.informatik.uni-frankfurt.de/research/papers.html, 2001.

[Kun85] E. Kunz. *Introduction to Commutative Algebra and Algebraic Geometry.* Birkhäuser Verlag, 1985.

[Lan93] S. Lang. *Algebra.* Addison Wesley, 1993.

[Laz91] D. Lazard. A new method for solving algebraic systems of positive dimension. *Discrete Applied Mathematics*, 33:147–160, 1991.

[Laz92] D. Lazard. Solving zero-dimensional algebraic systems. *Journal of symbolic computation*, 13:117–133, 1992.

[Lec97] G. Lecerf. *The Projective Noether Package, User's Manual.* Laboratoire GAGE, École polytechnique, Palaiseau, France, 1997.

[Lec99a] G. Lecerf. *Getting Started with Kronecker.* http://kronecker.medicis.polytechnique.fr, from 1999.

[Lec99b] G. Lecerf. *Kronecker. Reference Manual.* http://kronecker.medicis.polytechnique.fr, from 1999.

[Lec99c] G. Lecerf. Kronecker, a Magma package for polynomial system solving, from 1999. http://kronecker.medicis.polytechnique.fr.

[Lec00] G. Lecerf. Computing an equidimensional decomposition of an algebraic variety by means of geometric resolutions. In *Proceedings of ISSAC'2000*, pages 209–216. ACM, 2000.

[Lec01] G. Lecerf. Quadratic Newton iteration for systems with multiplicity. Manuscript, Laboratoire GAGE, École polytechnique, France. To appear in Journal of FoCM, April 2001.

[Lev40] U. J. J. Leverrier. Sur les variations séculaires des éléments elliptiques des sept planètes principales : Mercure, Vénus, la terre, Mars, Jupiter, Saturne et Uranus. *Journal de Mathématiques Pures et Appliquées*, 4:220–254, 1840.

[LL91] Y. N. Lakshman and D. Lazard. On the complexity of zero-dimensional algebraic systems. In *Effective methods in algebraic geometry*, volume 94 of *Progress in Mathematics*, pages 217–225. Birkhäuser, 1991.

[LL01] T. Y. Li and X. Li. Finding mixed cells in the mixed volume computation. *Foundation of Computational Mathematics*, 2001. Online publication, DOI: 10.1007/s102080010005.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:513–534, 1982.

[LR96] T. Lickteig and M.-F. Roy. Sylvester-Habicht sequences and fast Cauchy index computation. In *Calcolo 33*, pages 337–371, 1996.

[LS97] G. Lecerf and E. Schost. *Maple Package: GB link.* Laboratoire GAGE, École polytechnique, Palaiseau, France, 1997.
http://tera.medicis.polytechnique.fr/tera/soft.html.

[LS01] G. Lecerf and É. Schost. Fast multivariate power series multiplication in characteristic zero. Manuscrit, Laboratoire GAGE, École polytechnique, France, April 2001.

[Mac16] F. S. Macaulay. *The Algebraic Theory of Modular Systems.* Cambridge University Press, 1916.

[Mag] Magma.
http://www.maths.usyd.edu.au:8000/u/magma/.

[Mat86] H. Matsumura. *Commutative Ring Theory.* Cambridge University Press, 1986.

[Mat97] G. Matera. *Sobre la complejidad en espacio y tiempo de la eliminación geométrica.* PhD thesis, Universidad de Buenos Aires, Argentina, 1997.

[MB73] R. Moenck and A. B. Borodin. Fast computation of GCD's. In *5th Annual ACM Symposium on Theory of Computing*, pages 142–151, 1973.

[Med] MEDICIS, unité mixte de service CNRS/Polytechnique en calcul formel.
http://www.medicis.polytechnique.fr/.

[MF92] S. Maybank and O. Faugeras. A theory of self-calibration of a moving camera. *International Journal of Computer Vision*, 8(2):123–151, 1992.

[Möl98] H. M. Möller. Gröbner bases and numerical analysis. In B.Buchberger and F. Winkler, editors, *Gröbner Bases and Applications*, volume 251 of *Lecture Notes Series.* London Mathematical Society, Cambridge University Press, 1998.

[MNU97] K. Mehlhorn, S. Näher, and C. Uhrig. *Library for Efficient Datastructures and Algorithms.* Max Planck Institute for Computer Science, Saarbrücken, 1997.
http://www.mpi-sb.mpg.de/LEDA/leda.html.

[Mon92] P. Montgomery. *An FFT Extension of the Elliptic Curve Method of Factorization.* PhD thesis, University of California, Los Angeles, 1992.

[Mor97] J. E. Morais. *Resolución eficaz de sistemas de ecuaciones polinomiales.* PhD thesis, Universidad de Cantabria, Santander, Spain, 1997.

[MP00] B. Mourrain and V. Pan. Multivariate polynomials, duality, and structured matrices. *Journal of Complexity*, 16, 2000.

[MS95] H. M. Möller and H. J. Stetter. Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. *Numerische Mathematik*, 70:311–329, 1995.

[MT00] B. Mourrain and P. Trebuchet. Solving projective complete intersection faster. In Carlo Traverso, editor, *Proceedings of ISSAC'2000*, pages 234–241. ACM, August 7–9 2000.

[Mul87] K. Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7(1):101–104, 1987.

[MuP96] The MuPAD Group (B. Fuchssteiner et al.). *MuPAD User's Manual - MuPAD Version 1.2.2*. John Wiley and sons, Chichester, New York, www.mupad.de, first edition, march 1996.

[Nau98] R. Nauheim. Systems of algebraic equations with bad reduction. *Journal of Symbolic Computation*, 25(5):619–641, 1998.

[Nus80] H. J. Nussbaumer. Fast polynomial transform algorithms for digital convolutions. *IEEE Transactions on Acoustic, Speech and Signal Processing*, 28:205–215, 1980.

[Oji82] T. Ojika. Deflation algorithm for the multiple roots of simultaneous nonlinear equations. *Memoirs of Osaka Kyoiku University. III. Natural Science and Applied Science*, 30:197–209, 1982.

[Oji87] T. Ojika. Modified deflation algorithm for the the solution of singular problems. I. a system of nonlinear algebraic equations. *Journal of Mathematical Analysis and Applications*, 123:199–221, 1987.

[OWM83] T. Ojika, S. Watanabe, and T. Mitsui. Deflation algorithm for the multiple roots of a system of nonlinear equations. *Journal of Mathematical Analysis and Applications*, 96:463–479, 1983.

[Pad92] H. Padé. Sur la représentation approchée d'une fonction par des fractions rationnelles. *Annales scientifiques de l'École Normale Supérieure*, 9, 1892.

[Pan95] V. Y. Pan. Optimal (up to polylog factors) sequential and parallel algorithms for approximating complex polynomial zeros. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 741–750, Las Vegas, Nevada, 29 May–1 June 1995.

[Pan96] V. Pan. Optimal and nearly optimal algorithms for approximating polynomial zeros. *Computers and Mathematics (with Applications)*, 31:97–138, 1996.

[Par95] L. M. Pardo. How lower and upper complexity bounds meet in elimination theory. In G. Cohen, M. Giusti, and T. Mora, editors, *Applied Algebra, Algebraic Algorithms and Error Correcting Codes, Proceedings of AAECC-5*, volume 948 of *Lecture Notes in Computer Science*, pages 33–69. Springer, Berlin, 1995.

[Pau92] F. Pauer. On lucky ideals for Gröbner basis computation. *Journal of Symbolic Computation*, 14(5):471–482, 1992.

[Pél97] A. Péladan. *Tests effectifs de nullité dans des extensions d'anneaux différentiels.* PhD thesis, École polytechnique, 1997.

[PS78] F. P. Preparata and D. V. Sarwate. An improved parallel processor bound in fast matrix inversion. *Information Processing Letters*, 7(3):148–150, April 1978.

[Ren92] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part I. *Journal of Symbolic Computation*, 13(3):255–299, March 1992.

[Rit32] J. F. Ritt. Differential equations from an algebraic standpoint. *American Mathematical Society Colloquium Publications*, 14, 1932.

[Rit66] J. F. Ritt. *Differential Algebra.* Dover Publications, 1966.

[Roj94] J. M. Rojas. A convex geometric approach to counting the roots of a polynomial system. *Theoretical Computer Science*, 133(1):105–140, October 1994.

[Roj99] J. M. Rojas. Solving degenerate sparse polynomial systems faster. *Journal of Symbolic Computation*, 27, 1999. special issue on Elimination Theory.

[Roj00] M. Rojas. Computing complex dimension faster and deterministically. Manuscript of City University of Hong Kong, 2000.

[Rou] F. Rouillier. Realsoving.
http://www.loria.fr/~rouillie.

[Rou96] F. Rouillier. *Algorithmes efficaces pour l'étude des zéros réels des systèmes polynomiaux.* PhD thesis, Université de Rennes I, may 1996.

[Rou99] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Applicable Algebra in Engineering, Communication and Computing*, 6:353–376, 1999.

[RSS01] F. Rouillier, M. Safey el Din, and É. Schost. Solving the Birkhoff interpolation problem via the critical point method: an experimental study. In *Proceedings ADG 2000 (LNAI 2061).* Springer Verlag., 2001.

[Rup01] D. Rupprecht. Semi-numerical absolute factorization of polynomials with integer coefficients. *Journal of Symbolic Computation*, 2001. to appear.

[Saf01] Mohab Safey El Din. *Résolution réelle des systèmes polynomiaux en dimension positive.* PhD thesis, Université Paris VI,
http://calfor.lip6.fr/ safey/, 2001.

[Sak90] S. Sakata. Extension of the Berlekamp-Massey algorithm to n dimensions. *Information and Computation*, 84:207–239, 1990.

[Sam67] P. Samuel. *Méthodes d'algèbre abstraite en géométrie algébrique.* Springer-Verlag, 2nd edition, 1967.

[Sch71] A. Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Informatica*, 1:139–144, 1971.

[Sch77] A. Schönhage. Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. *Acta Informatica*, 7:395–398, 1977.

[Sch79] J. T. Schwartz. Probabilistic algorithms for verification of polynomial identities. In W. Ng. Edward, editor, *Symbolic and Algebraic Computation (EURO-*

SAM'79), volume 72 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, October 1980.

[Sch82] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Manuscript, University of Tübingen, Germany, 1982.

[Sch87] A. Schönhage. Equation solving in terms of computational complexity. In *Proceeding of the International Congress of Mathematicans*. American Mathematical Society, 1987.

[Sch00a] É. Schost. *Computing Parametric Geometric Resolutions*. PhD thesis, École polytechnique,
http://www.gage.polytechnique.fr/schost.html, 2000.

[Sch00b] É. Schost. *Sur la résolution des systèmes polynomiaux à paramètres*. PhD thesis, École polytechnique, 2000.
http://www.gage.polytechnique.fr/schost.html.

[Sie72] M. Sieveking. An algorithm for division of power series. *Computing*, 10:153–156, 1972.

[Sma81] S. Smale. The fundamental theorem of algebra and complexity theory. *Bulletin of the American Mathematical Society*, 4:1–36, 1981.

[Sma86] S. Smale. Algorithms for solving equations. In *Proceedings of the International Congress of Mathematicians*, pages 172–195, Berkeley, California, USA, 1986.

[SS71] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.

[SS96] J. Sabia and P. Solernó. Bounds for traces in complete intersections and degrees in the Nullstellensatz. *Applicable Algebra in Engineering, Communication and Computing*, 6:353–376, 1996.

[Ste96] H. J. Stetter. Analysis of zero clusters in multivariate polynomial systems. In *International Symposium on Symbolic and Algebraic Computation*, pages 127–136, 1996.

[Sto89] H.-J. Stoß. On the representation of rational functions of bounded complexity. *Theoretical Computer Science*, 64:1–13, 1989.

[Str69] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[Str72] V. Strassen. Berechnung und Programm. I, II. *Acta Informatica*, 1(4):320–355; *ibid.* 2(1), 64–79 (1973), 1972.

[Str73] V. Strassen. Die Berechnungskomplexität von elementarysymmetrischen Funktionen un von Interpolationskoeffizienten. *Numerische Mathematik*, 20:238–251, 1973.

[Stu94] B. Sturmfels. On the Newton polytope of the resultant. *Journal of Algebraic Combinatorics*, 3:207–236, 1994.

[Sut92] J. Sutor. *Axiom The Scientific Computation System*. Springer-Verlag, 1992.

[SV00] A. J. Sommese and J. Verschelde. Numerical homotopies to compute generic points on positive dimensional algebraic sets. *Journal of Complexity*, 16(3):572–602, 2000.

[SVW01a] A. J. Sommese, J. Verschelde, and C. W. Wampler. Numerical decomposition of the solution sets of polynomial systems into irreducible components. *SIAM Journal of Numerical Analysis*, 38(6):2022–2046, 2001.

[SVW01b] A. J. Sommese, J. Verschelde, and C. W. Wampler. Using monodromy to decompose solution sets of polynomial systems into irreducible components. In C. Ciliberto, F. Hirzebruch, R. Miranda, and M. Teicher, editors, *Application of Algebraic Geometry to Coding Theory, Physics, and Computation, Proceedings of NATO Conference 2001.* Kluwer, 2001. http://www.math.uic.edu/~jan/.

[TER98] TERA Development Group. A (hopefully) efficient polynomial equation system solver. Manuscript 57 pages, gmatera@mate.dm.uba.ar, 1998.

[Tiw87] P. Tiwari. Parallel algorithms for instance of the linear matroid parity with a small number of solutions. Technical Report IBM Reseach Report RC 12766, IBM Watson Research Center, Yorktown Heights, NY, 1987.

[Tri85] W. Trinks. On improving approximate results of Buchberger's algorithm by Newton's method. In B. Caviness, editor, *Proceedings of EUROCAL'85*, number 204 in Lecture Notes in Computer Science, pages 608–611. Springer-Verlag, 1985.

[VC93] J. Verschelde and R. Cools. Symbolic homotopy construction. *Applicable Algebra in Engineering, Communication and Computing*, 4(3):169–183, 1993.

[Ver96] J. Verschelde. *Homotopy Continuation Methods for Solving Polynomial Systems.* PhD thesis, K.U. Leuven, 1996.

[Ver99] J. Verschelde. Algorithm 795: PHCPACK: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software*, 25(2):251–276, June 1999.

[Vog84] W. Vogel. *Results on Bézout's Theorem.* Tata Institute of Fundamental Research. Springer, 1984.

[Vor99] N. Vorobjov. Complexity of computing the local dimension of semialgebraic set. *Journal of Symbolic Computation*, 27(6):565–579, 1999.

[Wad90] P. Wadler. Deforestation: transforming programs to eliminate trees. *Theoretical Computer Science*, 73:231–248, 1990. Special issue of selected papers from 2nd ESOP.

[Wan93] D. Wang. An elimination method for polynomial systems. *Journal of Symbolic Computation*, 16:83–114, 1993.

[Win88] F. Winkler. A p-adic approach to the computation of Gröbner bases. *Journal of Symbolic Computation*, 6:287–304, 1988.

[Wu86] W. T. Wu. On zeros of algebraic equations — an application of Ritt principle. *Kexue Tongbao*, 31:1–5, 1986.

[Yak95] J.-C. Yakoubsohn. A universal constant for the convergence of Newton's method and an application to the classical homotopy method. *Numerical Algorithms*, 9(3–4):223–244, 1995.

[Yak00] J.-C. Yakoubsohn. Finding a cluster of zeros of univariate polynomials. *Journal of Complexity*, 16, 2000.

[Zas69] H. Zassenhaus. Hensel factorization I. *Journal of Number Theory*, 1:291–311, 1969.

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings EUROSAM'79*, number 72 in Lecture Notes in Computer Science, pages 216–226. Springer, 1979.

[Zip90] R. Zippel. Interpolating polynomials from their values. *Journal of Symbolic Computation*, 9(3):375–403, 1990.

[Zip93] R. Zippel. *Effective Polynomial Computation*. ECS 241. Kluwer Academic Publishers, 1993.

243 références.

Table des algorithmes

Index

Résumé.

Introduite par H. Hironaka au milieu des années 1960, la notion de base standard d'un idéal dans un anneau de polynômes connaît depuis les travaux de B. Buchberger un engouement particulier en mathématiques et informatique. La construction effective d'une telle base est désormais une fonctionnalité essentielle dans tous les logiciels de calcul formel. Les algorithmes sous-jacents sont sans cesse améliorés et permettent de traiter des problèmes concrets inaccessibles aux méthodes numériques. Et pourtant la complexité de ces algorithmes est doublement exponentielle dans le pire des cas. Dans les années 1990, M. Giusti et J. Heintz montrent que les problèmes d'élimination peuvent être ramenés dans une classe polynomiale en représentant les polynômes éliminant par des calculs d'évaluation. Sur la base de leurs travaux, ma thèse aboutit à un algorithme probabiliste de calcul de la décomposition en composantes équidimensionnelles de l'ensemble des solutions d'un système d'équations et d'inéquations polynomiales. Sa complexité est polynomiale en un degré intrinsèque, de nature géométrique. Implanté dans le système de calcul formel Magma et appelé Kronecker en hommage à l'illustre mathématicien pour ses travaux sur l'élimination, notre paquetage affiche des performances qui sont à la hauteur des meilleures implantations dans le langage C de calcul de bases standard.

Abstract.

Introduced by H. Hironaka in the middle of the 60s, the concept of a standard basis of an ideal in a polynomial ring has become a topic of particular interest in mathematics and computer science since B. Buchberger's work. Nowadays the effective construction of such a basis is an essential functionality in all computer algebra systems. The subjacent algorithms are unceasingly improved and make it possible to deal with concrete problems inaccessible to numerical methods. And yet the complexity of these algorithms is doubly exponential in the worst case. In the 90s, M. Giusti and J. Heintz showed that elimination problems can be brought back in a polynomial complexity class by representing the eliminating polynomials by straight-line programs. On the basis of their work, my thesis leads to a probabilistic algorithm to compute the decomposition into equidimensional components of the solution set of a system of polynomial equations and inequations. Its complexity is polynomial in an intrinsic degree, of geometrical nature. Implemented in the Magma computer algebra system and called Kronecker in homage to the famous mathematician who worked on elimination, our package is competitive with the best implementations in the C language of standard bases.