Gröbner basis algorithms in service of algebraic set decomposition

Rafael Mohr

GRÖBNER BASIS ALGORITHMS IN SERVICE OF ALGEBRAIC SET DECOMPOSITION

RAFAEL MOHR

Thesis presented to obtain the degree of
DOCTEUR de SORBONNE UNIVERSITÉ
and
Doctor rerum naturalium (Dr. rer. nat.)

at
École Doctorale Informatique, Télécommunications et Électronique Paris
and
Graduate School of the Department of Mathematics of RPTU Kaiserslautern-Landau

supervised by
Christian Eder
Max Horn
Pierre Lairez
Mohab Safey El Din

Defended on the **25th of October 2024**
with the jury composed of:
Anna M. Bigatti (Examiner, Università degli Studi di Genova)
Claus Fieker (Examiner, RPTU Kaiserslautern-Landau)
Stef Graillat (Président du Jury, Examiner, Sorbonne Université)
Anton Leykin (Examiner, Georgia Tech University)
Michael Stillman (Reviewer, Cornell University)
Gilles Villard (Reviewer, CNRS, ENS Lyon)

ABSTRACT

Algebraic sets, i.e. solution sets of polynomial systems of equations, model a wide variety of nonlinear problems, both in applied and pure mathematics. One of the most foundational results in algebraic geometry sais that every algebraic set has a unique decomposition into *irreducible* algebraic sets and a frequent task is to decompose such an algebraic set into its irreducible components, or to produce some kind of coarser decomposition of the algebraic set in question. This task comes up for example in certain problems in robotics on the applied side and enumerative geometry on the pure side.

In this thesis, we present a series of algorithms solving such decomposition problems for algebraic sets. All of these algorithms use *Gröbner bases* for polynomial ideals at their core. Gröbner bases are an ubiquitous tool in symbolic computation. They form the core of many higher level algorithms and are implemented in all prominent computer algebra systems.

Three of these algorithms produce so-called *equidimensional* decompositions of an algebraic set, i.e. they partition a given algebraic set dimension-by-dimension. They are designed to avoid potentially costly elimination operations and, partially, use features of so-called *signature-based* Gröbner basis algorithms. Our software implementations of these algorithms showcase their practical efficiency compared to state-of-the-art computer algebra systems on examples of interest.

Another set of algorithms (respectively based on the F_4 *algorithm* and the FGLM *algorithm* use Hensel lifting methods in a novel way to compute Gröbner bases for *generic fibers* of polynomial ideals. We outline how these algorithms can be used as a core piece in known algorithms for equidimensional or irreducible decomposition of an algebraic set and exhibit their quasi-linear complexity in the precision up to which certain power series are computed.

Finally, we give an extension of generic fiber techniques for equidimensional decomposition of algebraic sets to compute *Whitney stratifications* of singular algebraic sets, improving on the state of the art. A Whitney stratification partitions a singular algebraic set into smooth pieces in a desirable way and has applications, for example, in physics. We, in addition, give an algorithm to minimize a given Whitney stratification.

Einen Prozeß, mit dem man überhaupt nie fertig werden könnte, wie das Zusammenzählen einer unendlichen Reihe, ermöglicht die Mathematik unter günstigen Umständen in wenigen Augenblicken zu vollziehen. Bis zu komplizierten Logarithmenrechnungen, ja selbst Integrationen macht sie es überhaupt schon mit der Maschine; die Arbeit des Heutigen beschränkt sich auf das Einstellen der Ziffern seiner Frage und auf das Drehen an einer Kurbel oder ähnliches. Der Amtsdiener einer Lehrkanzel kann damit Probleme aus der Welt schaffen, zu deren Auflösung sein Professor noch vor zweihundert Jahren zu den Herren Newton in London oder Leibniz in Hannover hätte reisen müssen.

Robert Musil, Der Mathematische Mensch, 1913

ACKNOWLEDGMENTS

There are numerous people which have accompanied me during the period of my life dedicated to the research presented in this thesis and who have helped it come to fruition in one way or another.
First of all, I want to mention my supervisors Christian Eder, Pierre Lairez and Mohab Safey El Din and thank them for their unending dedication in seeing me succeed as a scientist both during the past four years and in my academic future to come. I have learned a great deal from them, both mathematically and about navigating the world of science. The same goes for the other people with whom the work presented in this thesis was conducted, Jérémy Berthomieu and Martin Helmer.
Next I want to thank the members of the defense jury of this thesis for taking a significant amount of time out of their busy schedules to evaluate my work. I want to mention in particular the two reviewers, Mike Stillman and Gilles Villard, for reading my thesis in great detail and making numerous helpful suggestions for improvement.
Next, I would like to thank all the current and past members of the PolSys group in Paris and the algebra, geometry and computer algebra group in Kaiserslautern. Both have welcomed me with open arms and succeeded in creating a very comfortable, friendly and inspiring work environment. Special thanks go out to the friends I made in both of these teams: Lorenzo Baldi, Jorge García Fontán, Hadrien Notarantonio, Tobias Metzlaff, Rémi Prebet and Georgy Scholten for their help both in my personal and professional life and for making every conference a memorable event.
Outside of the world of academia, the material and mental support of my parents and my brother was invaluable during the past four years. I will be eternally grateful for their presence in my life. I also want to make particular mention of Katrin Kurz who has done more

for me than I could ever repay, not least of which was helping me in my difficult relocation to Paris.
Finally, I want to thank Yulia Mukhina from the bottom of my heart. Her presence in my life and the countless wonderful memories she gave me during the past year remind me why everything presented here was worth the effort.

CONTENTS

1

INTRODUCTION

SCIENTIFIC CONTEXT

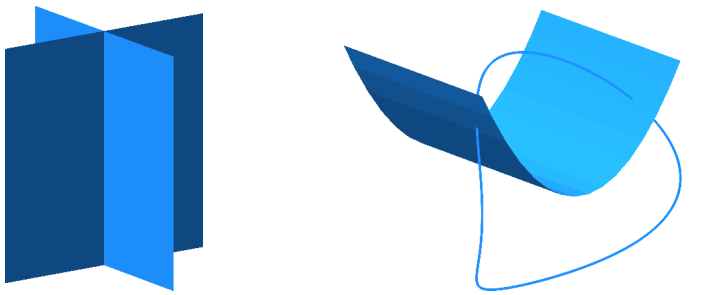
Polynomial systems, and their solution sets, arise in many applications to model non-linear problems. To name just a few example applications, they can be used to design cryptography schemes (e.g. Kipnis, Patarin, and Goubin, 1999) and to model problems in robotics (e.g. Hills, Baskar, Plecnik, and Hauenstein, 2024) and biology (e.g. Feinberg, 2019). If the polynomial system coming from some such application has finitely many solutions, one may then wish to *solve* the polynomial system which can either mean to list all of the solutions, or only a subset, either exactly (e.g. if the sought after solutions have entries in a finite field) or approximately (e.g. if the sought after solutions are real numbers) or approximately with some bound on the error of the approximations. If the polynomial system has infinitely many solutions one may wish to extract only the isolated solutions or to algorithmically describe the solution set of the polynomial system in such a way that further algebro-geometric information about the solution set can be extracted.

Very broadly, if one is interested in computing solutions to polynomial systems of equations, then, depending on the context, two classes of techniques and algorithms exist.

The first broad class of algorithms use *numerical* techniques, nowadays mostly based on the concept of *homotopy continuation*, see e.g. Sommese, Verschelde, and Wampler (2005), which can be of interest when solutions over the real or complex numbers are to be computed. When implemented on modern computers, these algorithms offer great efficiency, e.g. in the form of the `Julia`-package `HomotopyContinuation.jl`, see Breiding and Timme (2018), `Bertini`, see Bates, Hauenstein, Sommese, and Wampler (2013) or `PHCpack`, see Verschelde (2014), however, due to their numerical nature, they can suffer from issues of accuracy and inexhaustivity. We refer for example to Colotti et al. (2024) where one wishes to extract approximations of all real solutions of a polynomial system and numerical techniques fail to find all such solutions.

The second class of algorithms use *symbolic* techniques. In the context of polynomial system solving, this means that they perform arithmetic over algebraic structures whose elements allow for an exact, finite representation on a computer. A common use case is the computation of solutions over finite fields, which is a typical problem in cryptography. They can however also be used to compute real or complex solutions

to polynomial systems. There, they are usually slower than numerical algorithms, but *can* offer garantuees of accuracy and exhaustivity and modern implementations, e.g. in the form of the Gröbner basis library `msolve` (Berthomieu, Eder, and Safey El Din, 2021), are competitive enough to attack problems of real interest, which can, on some occasions, be intractable with numerical techniques, we refer again to Colotti et al. (2024) for examples.
The contributions of this thesis fall into the second class, i.e. we design symbolic algorithms to solve certain problems involving polynomial systems and their solution sets (which will henceforth be called *algebraic sets* in this thesis). To describe the problem, recall that one of the most foundational theorems in algebraic geometry says that each algebraic set may be written as a union of so-called *irreducible* algebraic sets (called irreducible components), each of which has associated to it a *dimension* invariant (see e.g. Part II in Eisenbud (1995) for a detailed introduction to the dimension theory of ideals and algebraic sets). An algebraic set is irreducible if it cannot be written (non-trivially) as a union of other algebraic sets.



(a) The surface $xy = 0$ in 3-space with the two 2-dimensional components $x = 0$ and $y = 0$.

(b) An algebraic set with one component of dimension one and one of dimension two.

Figure 1.1: Two algebraic sets over $\mathbb{R}$ plotted by the computer algebra system `Maple`.

Given a polynomial system in some application, often one is interested in only computing solutions that are part of a component of a certain dimension and one wishes to discard all other solutions or one would like to analyze the solution set dimension by dimension. To illustrate this we now give two examples.

Example 1.0.1. In robotics, *image-based visual servoing* is the task of controlling a robot's position and motion via a set of image features observed through a camera. In García Fontán, Nayak, Briot, and Safey El Din (2022), the authors deal with the *PnL problem*: A camera observes n lines in 3-space via their projection to a camera plane and the task is to estimate the camera's position and orientation (given as a vector in $\mathbb{R}^6$) from these projected lines.

Each observed line may be parametrized by *Plücker coordinates* and relating the time derivatives of these coordinates to the (angular and linear) velocity of the camera yields

$$s' = M_n \tau_c.$$

Here, τ_c describes the linear and angular velocity of the camera, s' is the time derivative of the lines given in Plücker coordinates projected on the camera plane and M_n is a matrix in three variables x, y, z (representing the spatial coordinates) and 11 parameters (corresponding to the choice of lines).
At points where M_n is singular (i.e. not of maximal rank), the pose estimation of the camera through the position of the observed lines faces accuracy and controllability issues. For actual design questions it is therefore important to exactly describe the geometry of the algebraic set described by the vanishing of the maximal minors of M_n.
In García Fontán, Nayak, Briot, and Safey El Din (2022) the authors now observe, through extensive analysis and ad hoc methods, that the algebraic set defined by the vanishing of the maximal minors of M_n can have both 1-dimensional components and isolated points, i.e. components of dimension zero.
This analysis can then inform design decision: The 1-dimensional components can be avoided by arranging the observed lines in a particular way.

Example 1.0.2. A *plane conic* is the solution set in $\mathbb{R}^2$ of an irreducible quadratic polynomial equation in two variables. The *Steiner problem* asks: Given five general (i.e. "random") conics A, B, C, D, E in $\mathbb{R}^2$, how many conics are tangent to A, B, C, D, E? The coefficients of the sought

Figure 1.2: Figure taken from www.juliahomotopycontinuation.org/3264/. A conic in red, together with five tangent conics in blue.

conics tangent to $A, \ldots, E$ are the isolated solutions to a polynomial system S in 5 variables with 5 equations, each of degree 6. This polynomial system is determined from the *Jacobian criterion* (e.g. Theorem

16.19 in Eisenbud, 1995). The algebraic set defined by S, denoted $\mathbf{V}(S)$, contains a surface (i.e. a 2-dimensional irreducible component) whose points correspond to conics that are squares of linear forms. It thus turns out that to properly count the number of conics tangent to $A, \dots, E$ requires to compute some sort of description of just the *isolated* solutions of S, i.e. the zero-dimensional irreducible components of $\mathbf{V}(S)$. We refer to Breiding, Sturmfels, and Timme (2020) for more details on the problem and a numerical approach to solve it. We will treat the polynomial system S as a benchmark for our algorithms throughout this thesis and show that the algorithms we design are able to compute such a description of the isolated solutions where other computer algebra software, using state-of-the-art symbolic techniques, fail.

While both problems in Example 1.0.1 and Example 1.0.2 can be dealt with by using (non-trivial) ad hoc methods, they nonetheless show the interest in having an automatized approach to partition a given algebraic set dimension by dimension.
To now more precisely introduce the problem(s) that we will be working on, let us introduce some notation. Let $\mathbb{K}$ be a field and let $R := \mathbb{K}[x_1, \dots, x_n]$ be the polynomial ring in n variables over this field. Let $\overline{\mathbb{K}}$ be an algebraic closure of $\mathbb{K}$ and let $\mathbb{A}^n$ be the affine space $\overline{\mathbb{K}}^n$. Given polynomials $f_1, \dots, f_r \in R$, consider the algebraic set

$$X := \mathbf{V}(f_1, \dots, f_r) := \{p \in \mathbb{A}^n \mid f_1(p) = \cdots = f_r(p) = 0\}.$$

These sets define the closed sets of a topology on $\mathbb{A}^n$, the *Zariski topology*. As mentioned above, X may be written, uniquely up to reordering, as a union of finitely many $\mathbb{K}$-*irreducible* algebraic sets (see Definition 2.1.1 and Theorem 2.1.1):

$$X = \bigcup_{i=1}^{s} X_i.$$

Here, an algebraic set is $\mathbb{K}$-irreducible if it cannot be written, non-trivially, as the union of two algebraic sets defined by polynomials in R. Each such X_i has associated to it a notion of dimension (see Definition 2.1.2) and we say that X is *equidimensional* if each of the X_i has the same dimension. If X is not equidimensional (as is the case in Example 1.0.2 above) then the largest part of this thesis deals, directly or indirectly, with the computation of an *equidimensional decomposition* of X, i.e. we will want to compute polynomials cutting out algebraic sets $Y_1, \dots, Y_t$ such that

$$X = \bigcup_{i=1}^{t} Y_i$$

and such that each Y_i is equidimensional. Before we outline our contributions exactly let us give an overview of related works.

RELATED WORKS

In the world of symbolic computation, a vast family of algorithms exists to compute (equidimensional or irreducible) decompositions of algebraic sets. We distinguish them first by the data structure they use to encode algebraic sets.

Geometric Resolutions

Geometric Resolutions form a "lazy" encoding of equidimensional algebraic sets. This means that they describe a given algebraic set on a Zariski open, dense subset of itself. The basic mathematical idea underlying geometric resolutions is the following: Suppose that $X \subset \mathbb{A}^n_{\mathbf{x}}$ is equidimensional of dimension d, with coordinates $\mathbf{x} = \{x_1, \ldots, x_n\}$ on $\mathbb{A}^n_{\mathbf{x}}$. Choose a generic affine subspace $\mathbb{A}^d_{\mathbf{z}} \subset \mathbb{A}^n$ of dimension d in new coordinates $\mathbf{z} := \{z_1, \ldots, z_d\}$. Choose another generic linear combination y of the variables $\mathbf{x}$ with corresponding affine space $\mathbb{A}^1_y$. Then projecting X to $\mathbb{A}^d_{\mathbf{z}} \times \mathbb{A}^1_y$ will yield a hypersurface X' in $\mathbb{A}^d_{\mathbf{z}} \times \mathbb{A}^1_y$, i.e. X' is cut out by a single polynomial g in the $d+1$ variables $\mathbf{z} \cup \{y\}$: $X' = \mathbf{V}(g)$. Furthermore, X will be *birational* to X', i.e. for almost every specilization of the $\mathbf{z}$-variables (i.e. for every choice of specialization outside of a Zariski open subset of $\mathbb{A}^d_{\mathbf{z}}$) we obtain the corresponding points of X as parametrizations over the corresponding zeros of the univariate specialization of the polynomial g. A geometric resolution for X consists then, essentially, of the variables $\mathbf{z}$ and y, the polynomial g and the parametrizations of the variables $\mathbf{x}$ over the vanishing locus of g.

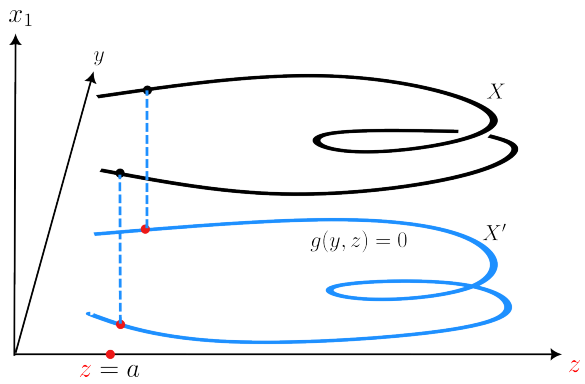


Figure 1.3: The curve in black projects birationally to the curve in blue, lying in the subspace given by the y- and z-coordinate. For almost every choice of specialization of $z = a$, we obtain the corresponding points on the black curve via a parametrization over the blue curve.

Algorithms for decomposing algebraic sets into equidimensional ones where each output component is encoded by a geometric resolution over an affine subspace with respect to which the component is in

Noether position have been developed in Lecerf (2000, 2003). Noether position means here, that the fiber of X over each point in $\mathbb{A}^d_y$ is zero-dimensional, i.e. consists only of finitely many points. The decompositions produced by these algorithms allow to work with the algebraic set in question only inside the Zariski open sets which are attached to the geometric resolution of each output component.
In Schost (2003) the author develops a decomposition algorithm which avoids the Noether position assumption, i.e. here the subspace $\mathbb{A}^d_z$ is given and one asks only that the projection of X to $\mathbb{A}^d_z$ is *dominant*, i.e. the closure of the image of this projection is the entire $\mathbb{A}^d_z$. A restriction for this algorithm is that the input algebraic set X be then given by exactly $n - d$ equations. The algorithm then computes a geometric resolution of X outside of the locus where the Jacobian matrix of our $n - d$ equations is rank deficient.
These algorithms utilize so-called *straight line programs* for fast, division-free evaluation of polynomials and a formal Newton iteration algorithm, which relies on assuring certain non-degeneracy assumptions. This formal Newton iteration is used to lift a geometric resolution with the variables in $\mathbf{z}$ specialized to some value to a one-dimensional representation, after which the intersection of the algebraic set encoded by the geometric resolution with a hypersurface can be again given in terms of a "specialized" geometric resolution.
These algorithms obtain the best known complexity bounds for equidimensional decomposition, polynomial in a quantity derived from the *degree* of the algebraic set cut out by the input system. Recall that the degree of an algebraic set X of dimension d is the number of points in the intersection of X with d general hyperplanes.
We also mention Jeronimo and Sabia (2002) for an approach using similar techniques.
The algorithms mentioned above are not implemented in any of the more widely used computer algebra systems such as `Maple`, `Singular` or `Macaulay2`.

Triangular Sets and Regular Chains

Triangular sets also encode equidimensional algebraic sets "lazily" by describing them on a Zariski open dense subset of themselves. The basic idea is that an equidimensional algebraic set of codimension c should be cut out, almost everywhere, by exactly c equations. We then enforce these c equations to have in addition a triangular structure. More precisely, let $\mathbf{y} := \{y_1, \ldots, y_c\} \subset \mathbf{x}$ and let $\mathbf{z} := \mathbf{x} \setminus \mathbf{y}$. A triangular set T then consists of c polynomials

$$T := \{f_1(\mathbf{z}, y_1), f_2(\mathbf{z}, y_1, y_2), \ldots, f_c(\mathbf{z}, y_1, \ldots, y_c)\}.$$

Regarding each f_i as a univariate polynomial in y_i, each f_i has a leading coefficient $h_i \in \mathbb{K}[\mathbf{z}, y_1, \ldots, y_{i-1}]$. Let $h := \operatorname{lcm}(h_1, \ldots, h_c)$. Then each triangular set has attached to it the *locally closed set*

$$\mathbf{W}(T) := \mathbf{V}(f_1, \ldots, f_c) \setminus \mathbf{V}(h) \subset \mathbb{A}^n.$$

Recall that a locally closed set is the set difference of two algebraic sets. Therefore, a triangular set T as above describes an algebraic set away from the degenerate points given by $\mathbf{V}(h)$.
Such triangular sets have their origin in so-called Wu-Ritt characteristic sets (see e.g. Chou and Gao, 1990; Gallo and Mishra, 1991; Ritt, 1950; Wang, 1993; Wu, 1986) which have proven particularly effective in the context of differential algebra.
Of particular importance, especially in the realm of equidimensional decomposition, are certain special triangular sets called *regular chains*, introduced by Kalkbrener (1993) and Lu and Jingzhong (1994). A regular chain is a triangular set T as above where, essentially, every solution of the first k equations of T can be lifted to a solution of the first $k+1$ equations. A regular chain T has good algorithmic properties with respect to the attached locally closed set $\mathbf{W}(T)$: In particular we can test whether a given polynomial $f \in \mathbb{K}[\mathbf{x}]$ vanishes on the Zariski closure $\overline{\mathbf{W}(T)}$ of $\mathbf{W}(T)$, where the Zariski closure of a given set in $\mathbb{A}^n$ is the smallest algebraic set containing it. Moreover, every triangular set T as above has attached to it an ideal $\mathbf{I}(T)$ s.t. $\mathbf{V}(\mathbf{I}(T)) = \overline{\mathbf{W}(T)}$. Then the ring $\mathbb{K}(\mathbf{z})[\mathbf{y}]/\mathbf{I}(T)$ is a product of fields and when T is a regular chain, one may use suitable generalizations of algorithms for univariate polynomials over fields to univariate polynomials over *products* of fields to work with regular chains, in particular the Euclidean algorithm, through the so-called D5 principle (Della Dora, Dicrescenzo, and Duval, 1985).
Two kinds of decompositions may be computed using regular chains: Given an algebraic set $X \subset \mathbb{A}^n$ one may either compute regular chains $T_1, \ldots, T_r$ s.t.

$$X = \bigcup_{i=1}^{r} \mathbf{W}(T_i),$$

henceforth called a *Lazard decomposition*, see in particular Lazard (1991), or regular chains $T_1, \ldots, T_s$ s.t.

$$X = \bigcup_{i=1}^{s} \overline{\mathbf{W}(T_i)},$$

henceforth called a *Kalkbrener decomposition*, see again Kalkbrener (1993) and Lu and Jingzhong (1994). While a Lazard decomposition yields a complete description of X, it may be slower to compute than a Kalkbrener decomposition, especially on more difficult examples of positive dimension, see e.g. Aubry and Maza (1999). A Kalkbrener decomposition, however, allows one only to work with the points of X

which lie outside of the locus of degeneracy of the produced triangular sets.
Algorithms using regular chains are prominently part of the computer algebra system `Maple` (Chen and Moreno Maza, 2012; Chen et al., 2007). We further refer to Hubert (2003) and Wang (2001) for comprehensive introductions to the subject and to Aubry, Lazard, and Maza (1999) for a theoretical account as to how certain different notions of triangular sets relate to each other.

Techniques based on Gröbner Bases

The notion of a *Gröbner basis* differs, philosophically, from geometric resolutions and triangular sets in that an ideal-theoretic, instead of a geometric, point of view is taken. A Gröbner basis depends on two parameters: A polynomial ideal $I \subset \mathbb{K}[\mathbf{x}]$ and a *monomial order* $\prec$, which is a total order on the set of monomials in $\mathbb{K}[\mathbf{x}]$ satisfying certain additional conditions.
A Gröbner basis G of an ideal $I \subset \mathbb{K}[\mathbf{x}]$ w.r.t. to $\prec$ is then a certain finite generating set of I which allows for well-behaved multivariate polynomial long division with terms ordered w.r.t. $\prec$. More precisely, a polynomial $f \in \mathbb{K}[\mathbf{x}]$ is contained in I if and only if multivariate polynomial long division of f by G returns zero.
Through this property, in contrast to geometric resolutions and triangular sets, G represents the algebraic set $\mathbf{V}(I)$ everywhere, i.e. there is no locus of degeneracy in $\mathbf{V}(I)$ that cannot be described by G. Gröbner bases can also be used to compute with any polynomial ideal (and thus any algebraic set) and not just equidimensional ones as is the case for geometric resolutions and triangular sets. This, together with the fact that Gröbner bases can represent algebraic sets everywhere, means that, from G, various algebro-geometric properties about I or $\mathbf{V}(I)$ may be extracted, such as dimension or degree, and most ideal-theoretic and geometric operations of interest, such as computing the intersection of ideals or taking the image of algebraic sets under algebraic maps, become computable. This gives them a great deal of flexibility, in particular in how they can be used to design algorithms for various tasks involving algebraic sets or polynomial ideals, making them part of the backbone of many popular computer algebra systems.
If one wants to use Gröbner basis techniques to compute an (equidimensional or irreducible) decomposition of an algebraic set then there are two popular approaches:
The first uses projections of algebraic sets, or *elimination theory*. Recall that a Gröbner basis depends on a chosen monomial order and projections become computationally available using so-called *block orders*. Given an algebraic set X via a polynomial ideal I, i.e. $X = \mathbf{V}(I)$, we choose a partition $\mathbf{x} = \mathbf{z} \sqcup \mathbf{y}$ s.t. X projects *dominantly* to the $\mathbf{z}$-space $\mathbb{A}^d_{\mathbf{z}}$, i.e. the closure of the image of X in $\mathbb{A}^d_{\mathbf{z}}$ is $\mathbb{A}^d_{\mathbf{z}}$ itself. Here,

$d = \dim(X)$. Then, we compute a Gröbner basis of I w.r.t. a suitable block order which then gives a Gröbner basis of the *generic fiber* of I w.r.t. $\mathbf{z}$, denoted in this thesis $\operatorname{gen}(I, \mathbf{z}) := I\mathbb{K}(\mathbf{z})[\mathbf{y}]$.

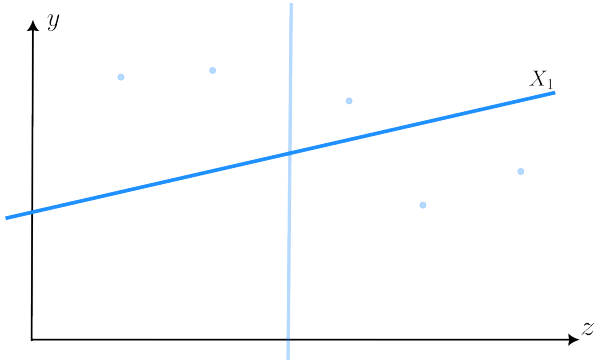


Figure 1.4: An algebraic set $X = \mathbf{V}(I)$ consisting of two lines and several isolated points. When projecting to the z-axis, the generic fiber $\operatorname{gen}(I, z)$ "sees" only those components which project dominantly to the z-axis, i.e. only the horizontal line X_1.

Now, one can compute the ideal $J := \operatorname{gen}(I, \mathbf{z}) \cap \mathbb{K}[\mathbf{x}]$ and $\mathbf{V}(J)$ will consist only of those components of X which project dominantly to $\mathbb{A}^d_{\mathbf{z}}$ themselves, in particular $\mathbf{V}(J)$ will be equidimensional of dimension d. This gives the basic approach of computing an equidimensional decomposition using this approach.
When an *irreducible* decomposition is desired instead, one replaces the first variable in $\mathbf{y}$ by a generic linear combination of the variables in $\mathbf{y}$ and projects, again, using block orders, to the space $\mathbb{A}^d_{\mathbf{z}} \times \mathbb{A}^1_y$. Under some additional assumptions, the resulting Gröbner basis will then in fact give a geometric resolution of those components of X which project dominantly to the $\mathbf{z}$-axis, i.e. of $\mathbf{V}(J)$. Then, a factorization of the univariate polynomial g in this geometric resolution yields an irreducible decomposition of $\mathbf{V}(J)$ and thus a subset of the irreducible components of X.
On any given example, a potential difference compared to working with geometric resolutions directly is that no Noether position assumptions or restrictions on the input are required. This potentially lowers the degrees of the algebraic sets with which we are computing, making computations easier and producing a finer decomposition with more output components than there are dimensions.
We should again emphasize that in contrast to algorithms using geometric resolutions or triangular sets, the above algorithms using Gröbner bases work ideal-theoretically as opposed to geometrically. This means that they can also be adapted to obtain ideal-theoretic decompositions, such as a *primary decomposition* of a polynomial ideal. In Chapter 8 we present an application of these kinds of techniques where this turns out to be crucial.

We refer to the classical references Caboara, Conti, and Traverso (1997), Gianni, Trager, and Zacharias (1988), and Krick and Logar (1991), to the survey given by Decker, Greuel, and Pfister (1999) and to Section 3.2 for more details on the kind of algorithms sketched above.
The second approach using Gröbner bases relies instead on a certain homological characterization of dimension based on the Auslander-Buchsbaum formula, see Theorem 19.9 in Eisenbud (1995). This approach is first presented in Eisenbud, Huneke, and Vasconcelos (1992): An exact "homological formula" for the components of highest dimension of a polynomial ideal is given and can be made computationally explicit via the computation of syzygies and free resolutions of polynomial ideals. Eisenbud, Huneke, and Vasconcelos (1992) also present an elimination-free algorithm for primary decomposition, based on *normalization* of polynomial ideals. We refer to Decker, Greuel, and Pfister (1999), Greuel and Pfister (2007), and Vasconcelos (1998) for further details.
For computation over the rational numbers, the techniques mentioned above can be combined with multi-modular techniques to avoid the well known problem of "coefficient swell" when computing directly over the rational numbers. These can then be combined with dedicated techniques and implementations for finite fields. We refer e.g. to Ishihara (2022), Noro and Yokoyama (2004), and Yokoyama (2002).
Finally, we mention the ad hoc algorithm given in Moroz (2008) which shares many ideas with the algorithm we develop in Chapter 5. We will comment on the differences below.
Algorithms using Gröbner bases are prominently implemented in all modern computer algebra systems. Both the techniques above using elimination and the ones using homological techniques can be found, for example, in the computer algebra systems `Maple`, `OSCAR`, `Singular`, `Macaulay2` and `Magma`.

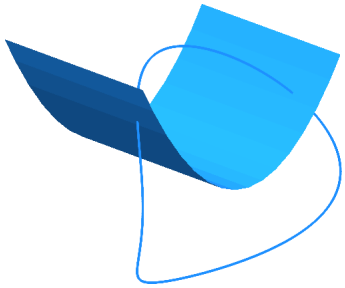
OVERVIEW OF CONTRIBUTIONS

Chapters 4, 5 and 6 deal with the problem of equidimensional decomposition of algebraic sets using Gröbner basis computations. While we give no complexity results therein, we show experimentally that these algorithms consistently outperform implementations of some of the algorithms introduce above in state-of-the-art computer algebra systems on a wide range of examples, sometimes by several orders of magnitude.
In Chapter 7, we develop dedicated algorithms to compute Gröbner bases of generic fibers of polynomial ideals, defined as above. As hinted at before, these algorithms can be used as cornerpieces for equidimensional and irreducible decomposition as well. This chapter contains a complexity analysis of the algorithms presented therein.

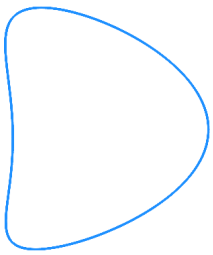
As a potential application of the algorithms presented in Chapter 7 and as an extension to algebraic set decomposition using generic fibers of ideals as introduced above, in Chapter 8, we show how Gröbner bases of generic fibers can be used to compute so-called *Whitney stratifications* of singular algebraic sets. In addition, we give an algorithm to minimize a given Whitney stratification.
Finally, in Chapter 9 we outline some research perspectives following from the results presented in this thesis.
Let us now give a more detailed overview of our contributions.

Computation of Nondegenerate Loci

The first chapter in Part ii, Chapter 4, based on the results given by Eder, Lairez, Mohr, and Safey El Din (2023b), published in the Journal of Symbolic Computation, deals with the computation of the *nondegenerate locus* of a sequence of polynomials $F := (f_1, \ldots, f_c)$. The nondegenerate locus is the set of points p in $\mathbf{V}(F)$ where F forms a *regular sequence*. F is a regular sequence if for every i and every $g \in \mathbb{K}[\mathbf{x}]$ with $gf_i \in \langle f_1, \ldots, f_{i-1}\rangle$ we have $g \in \langle f_1, \ldots, f_{i-1}\rangle$. Geometrically, regular sequences define *complete intersections*, which are algebraic sets such that the codimension of every irreducible component matches the number of defining equations.
Equivalently, the nondegenerate locus of F is the set of points p in $\mathbf{V}(F)$ where every irreducible component of $\mathbf{V}(F)$ passing through p has codimension c, so that the closure of the nondegenerate locus gives us precisely the components of codimension c. Note that in Example 1.0.2 this set encodes precisely the sought after isolated solutions.



(a) $\mathbf{V}(x^2z^2 + x^2y - z^3 - x^2 - yz + z,$ $x^4z - x^2y - x^2z + yz)$



(b) $\mathbf{V}(x^2 + z^2 - 1, x^2 - y)$

Figure 1.5: The algebraic set on the left-hand side is cut out by two equations. The closure of the nondegenerate locus on the right thus gives the components of codimension two, i.e. dimension one.

The key idea of the algorithm presented in Chapter 4 is to utilize a certain feature of so-called *signature-based* Gröbner basis algorithms, the first of which was the F_5 *algorithm* (Faugère, 2002). Very roughly, these algorithms can be thought of as algorithms which build and echelonize a series of matrices, called *Macaulay matrices*. A Macaulay

matrix associated to F organizes several monomial multiples of F as rows into a matrix, with columns labeled by the union of the supports of these monomial multiples. Their special feature, compared to other Gröbner basis algorithms which follow a similar strategy, is that no row in these matrices is reduced to zero during echelonization if and only if F is a regular sequence. By contraposition, this means that, when a reduction to zero occurs, F is not a regular sequence. More precisely, when we meet a reduction to zero, then it corresponds exactly to an element $g \in \mathbb{K}[\mathbf{x}]$ which satisfies $gf_i \in \langle f_1, \ldots, f_{i-1}\rangle$ but $g \notin \langle f_1, \ldots, f_{i-1}\rangle$ for some i. We can then immediately use the knowledge of this g to change the polynomial system F on the fly, during a Gröbner basis computation, so that at the end we obtain a Gröbner basis of an ideal cutting out the closure of the nondegenerate locus of F.
A frequent strategy employed by a number of the algorithms mentioned in the last section is that they compute a decomposition incrementally, equation-by-equation, i.e. first for $\mathbf{V}(f_1)$, then for $\mathbf{V}(f_1, f_2)$ and so on until all equations have been processed. This structurally simplifies the problem of equidimensional decomposition. This incremental structure is a feature shared with a certain class of signature-based Gröbner basis algorithms, i.e. they compute a Gröbner basis for the ideal generated by F equation-by-equation. This, together with the feature above, makes them suitable for the task of computing an ideal cutting out the closure of the nondegenerate locus of F.
The algorithm presented in this chapter works in stark contrast to the previously known decomposition algorithms using Gröbner bases introduced above: In these algorithms Gröbner basis computations are used as a *black box*, i.e. they ask only for certain Gröbner bases of certain ideals, no matter the algorithm which computes them. In contrast to that, our algorithm is a *dedicated* Gröbner basis algorithm for the task at hand, i.e. we change the process of computing a Gröbner basis itself to obtain the sought after nondegenerate locus at the end of the computation. This potentially means that the polynomials involved have lower degree: We discover equations cutting out the nondegenerate locus before a Gröbner basis of $\langle F\rangle$ is computed. We give, in Chapter 4, some experimental data which shows the validity of this approach independent of implementational considerations.

Incremental Computation of Equidimensional Decompositions

The next chapter, Chapter 5, based on the results given by Eder, Lairez, Mohr, and Safey El Din (2023a), published and presented at ISSAC 2023, presents an algorithm to compute a full equidimensional decomposition of an algebraic set. More precisely, given a sequence $F := (f_1, \ldots, f_r) \subset R$ as before, we compute a set of pairwise disjoint

locally closed sets (see Definition 2.2.1) $Y_1, \ldots, Y_s$ of the shape $Y_i = X_i \setminus \mathbf{V}(h_i)$, each with equidimensional Zariski closure, such that

$$\mathbf{V}(F) = \bigcup_{i=1}^{s} Y_i.$$

As in Chapter 4, the algorithm proceeds incrementally, i.e. by computing such a decomposition first for $\mathbf{V}(f_1, f_2)$, then intersecting and decomposing the resulting locally closed sets with f_3, then with f_4 and so on, until all f_i have been processed. We should note again, that this incremental idea is used frequently in decomposition algorithms using geometric resolutions or regular chains present in the literature and also for the algorihm given by Moroz (2008) using Gröbner bases. Essentially, this incremental structure reduces the problem of equidimensional decomposition to the following: Given an equidimensional algebraic set X and $f \in R$, compute an equidimensional decomposition of $X \cap \mathbf{V}(f)$.
To perform this operation, the algorithm presented in this chapter again uses Gröbner bases in a core way. Our algorithm does not rely on elimination theory, giving us the freedom to compute our Gröbner bases using the so-called degree reverse lexicographic (DRL) monomial order instead of being forced to use block orders. It has been observed many times that Gröbner basis computations tend to behave better when using the DRL order instead of block orders, this observation has been justified from a complexity theoretic view by Lazard (1983). The incremental nature of our algorithm also allows to avoid extensive syzygy computations (as is required for the algorithm by Eisenbud, Huneke, and Vasconcelos, 1992), essentially we only use partial information about the polynomial relations of F.
While we do use Gröbner bases, and thus an ideal-theoretic instrument, we do design our algorithm from a geometric perspective, working explicitly with a data structure encoding a locally closed set, derived from a Gröbner basis of an ideal cutting out the closure of this locally closed set. Working with locally closed sets explicitly is a property shared with the algorithms using regular chains. In the incremental approach, this turns out to have the following advantage: It naturally removes from the output sets of our iterative algorithm certain superflous irreducible components that appear during the decomposition. To illustrate this consider the following example:

Example 1.0.3. Let $R := \mathbb{Q}[x, y, z]$, $X := \mathbf{V}(xy)$, $f := xz$. To decompose $X \cap \mathbf{V}(f)$ into equidimensional components one may start by decomposing $X = \mathbf{V}(x) \cup \mathbf{V}(y)$. Then one intersects these two components with $\mathbf{V}(f)$ to obtain the equidimensional decomposition $X \cap \mathbf{V}(f) = \mathbf{V}(x) \cup \mathbf{V}(y, xz)$. The latter set has the irreducible component $\mathbf{V}(y, x)$ which is embedded in $\mathbf{V}(x)$ and thus superflous in the decomposition. If one instead splits into a *disjoint* union $X = \mathbf{V}(x) \cup [\mathbf{V}(y) \setminus \mathbf{V}(x)]$ and again intersects both components with $\mathbf{V}(f)$ one obtains $X =$

$\mathbf{V}(x) \cup (\mathbf{V}(y,z) \setminus \mathbf{V}(x))$ and the closure of the latter component no longer has the irreducible component $\mathbf{V}(y,x)$.

Second, when an iterative equidimensional decomposition algorithm produces redundant components, then, if they are not deduplicated, one may suffer from an exponential blow-up in the number of components: if one has decomposed $X = \bigcup_i X_i$ with the X_i sharing a large number of irreducible components then decomposing each $X_i \cap \mathbf{V}(f)$ to obtain a decomposition of $X \cap \mathbf{V}(f)$ results in an even more redundant decomposition. Because we use locally closed sets to model our equidimensional sets we are enabled to carefully design a decomposition procedure which avoids the production of such redundancies. This comprises a central difference with the algorithm by Moroz (2008): While this algorithm obtains a slightly stronger decomposition than our algorithm (it decomposes into *local complete intersections* and not just into equidimensional locally closed sets) it does introduce redundancies in every decomposition step [1]. On the polynomial system Sing(2,10) (see Appendix A.1 for an explanation of this polynomial system) our algorithm finishes computation in 2.9 seconds while the redundancies introduced by the algorithm by Moroz (2008) cause it to not finish within several minutes of computation.
Borrowing further from the design approach of algorithms using triangular sets we also adopt the heuristic that it is a good idea to decompose given algebraic sets as often and as finely as possible when working with them. This philosophy is baked into a certain recursive structure of our algorithms which exists so as to decompose a given locally closed set as much as possible given generating sets for certain ideals. Coupled with the fact that we try to decompose irredundantly and avoid elimination theory as much as possible this tends to yield locally closed sets whose underlying ideals have low degree which speeds up necessary Gröbner basis computations. Supporting this, an implementation of our algorithm in the computer algebra system `OSCAR` (Decker et al., 2025) is demonstrated to outperform existing similar algorithms in state-of-the-art computer algebra systems by several orders of magnitude.

Non-Incremental Computation of Equidimensional Decompositions

The algorithm last presented, treated in detail in Chapter 5, has potentially two issues:

(1) Recall that this algorithm computes the desired equidimensional decomposition by processing the equations $f_1, \ldots, f_r$ one-by-one. This can cause issues because, in Chapter 5, we are representing

1 In Example 1.0.3 our algorithm obtains the decomposition $V(xy,xz) = \mathbf{V}(x) \cup (\mathbf{V}(y,z) \setminus \mathbf{V}(x))$ while the algorithm by Moroz (2008) obtains the decomposition $X \cap \mathbf{V}(f) = \mathbf{V}(x) \cup \mathbf{V}(y,xz)$.

our locally closed sets using Gröbner bases of certain associated ideals. Experimentally, the computations of these Gröbner bases tend to be hardest "in the middle", i.e. when about half of the equations have been processed, in particular when the algorithm presented in Chapter 5 does not produce any decomposition after having processed the first few equations. This behaviour can be somewhat explained by complexity statements for Gröbner basis computations under some regularity assumptions: Indeed, for a so-called *regular sequence in strong Noether position*, the cost of computing a Gröbner basis equation-by-equation by echelonizing Macaulay matrices is higher than in the final steps (Bardet, Faugère, and Salvy, 2015). Dimension dependent complexity bounds provide another confirmation of this behaviour, see Hashemi and Seiler (2017).

(2) While the locally closed output sets $Y_1, \ldots, Y_s$ of this algorithm are pairwise disjoint, some of them might be redundant when taking Zariski closures, i.e. we may have $\overline{Y_i} \subseteq \overline{Y_j}$ for certain $i \neq j$. This is a property shared with Lazard decompositions into triangular sets. Since we, however, work with Gröbner bases instead of regular chains, we can explicitly work with the Zariski closures of our locally closed sets. Thus, one would like to get rid of those locally closed sets which become superflous once we take their Zariski closures.

The algorithm presented in Chapter 6, developed independently by the author of this thesis, tackles both issues while having many structural similarities and similar design goals as the algorithm presented in Chapter 5: It relies, in a key way on a non-incremental characterization of regular sequences by Vasconcelos (1967) (see Theorem 2.5.1) in terms of their syzygies, i.e. in terms of their polynomial relations. This theorem again allows us to decompose without having to rely on elimination theory or heavy homological algebra and it allows us to design an algorithm which considers the entire input sequence F at once which, in turn, enables us to adapt certain subroutines from Chapter 5 in order to produce, again, a set of pairwise disjoint locally closed sets $Y_1, \ldots, Y_s$ such that, this time

$$\mathbf{V}(F) = \bigcup_{i=1}^{s} \overline{Y_i},$$

i.e. we produce something analogous to a Kalkbrener decomposition into regular chains.
Using Gröbner bases gives us the flexibility of designing an algorithm whose output satisfies *complete irredundancy*: The output components Y_i do not share irreducible components among each other and the union of the irreducible components of each $\overline{Y_i}$ gives precisely the irreducible components of $\mathbf{V}(F)$. The decomposition produced by the

algorithm in Chapter 6 thus satisfies a certain strong minimality condition. In recognition of the algorithm presented in Kalkbrener (1993), and in analogy to Kalkbrener decompositions into regular chains, we call a decomposition as produced by the algorithm presented in Chapter 6 a *Kalkbrener partition*.
In order to algorithmically use Vasconcelos' Theorem 2.5.1 mentioned above, we rely again, as in Chapter 4, on signature-based Gröbner basis algorithms to facilitate the necessary syzygy computations in an efficient way.
The proposed algorithm is shown to outperform the algorithm of Chapter 5 on a range of examples.

Computing Gröbner Bases of Generic Fibers

The object under study in Chapter 7, based on the results given by Berthomieu and Mohr (2024), published and presented at ISSAC 2024, is the following: Fix a polynomial ideal $I \subset \mathbb{K}[\mathbf{x}, \mathbf{z}]$ in two finite sets of variables $\mathbf{x}$ and $\mathbf{z}$ such that the map $\mathbb{K}[\mathbf{z}] \to \mathbb{K}[\mathbf{x}, \mathbf{z}]/I$ is injective and such that the *generic fiber*

$$\operatorname{gen}(I, \mathbf{z}) := I\mathbb{K}(\mathbf{z})[\mathbf{x}]$$

of I is a zero-dimensional ideal. Geometrically, this means that the projection of $\mathbf{V}(I)$ to the affine space in the $\mathbf{z}$-coordinates is dominant with almost everywhere zero-dimensional (i.e. finite) fiber given by specialization of $\operatorname{gen}(I, \mathbf{z})$, as in Figure 1.4.
As mentioned above, if one is able to compute Gröbner bases of such generic fibers (w.r.t. a suitable block order, depending on the task at hand) then one obtains algorithms both for equidimensional decomposition and for irreducible decomposition, this serves as the primary motivation for Chapter 7.
In Chapter 7 we present two algorithms for computing Gröbner bases of generic fibers by relying on classical Hensel lifting techniques: If we let $\mathfrak{m} := \langle \mathbf{z} \rangle$, then, under the (probabilistic) assumption that the origin in the $\mathbf{z}$-space lies outside of a certain hypersurface, we will compute the desired Gröbner basis G of $\operatorname{gen}(I, \mathbf{z})$ by computing its image in $(\mathbb{K}[\mathbf{z}]/\mathfrak{m})[\mathbf{x}] \simeq \mathbb{K}[\mathbf{x}]$ and then lifting it modulo increasing powers of $\mathfrak{m}$. When this power is large enough, we can then recover G using the classical technique of Padé approximation. With this approach, we also expect that our algorithm can be transported without much difficulty to the setting where a Gröbner basis G of a zero-dimensional ideal in $\mathbb{Q}[\mathbf{x}]$ is required: Given p a well-chosen prime number and $k \in \mathbb{N}^*$ sufficiently large, our algorithm would extract G from its image in $(\mathbb{Z}/p\mathbb{Z})[\mathbf{x}]$ and then lifting it modulo p^k for sufficiently large k.
The first algorithm combines Hensel lifting techniques with the FGLM algorithm given in Faugère, Gianni, Lazard, and Mora (1993). The FGLM algorithm converts the Gröbner basis w.r.t. one monomial order

to a reduced Gröbner basis w.r.t. another monomial order when I is zero-dimensional (equivalently if $\mathbf{V}(I)$ consists of finitely many points). Recall that this is the case if and only if $\mathbb{K}[\mathbf{x}]/I$ is a finite-dimensional $\mathbb{K}$-vector space. Then, having a Gröbner basis of I w.r.t. one monomial order enables us to perform linear algebra in this vector space which, in turn, enables us to find a Gröbner basis of I w.r.t. any other monomial order. As mentioned before, depending on the monomial order for one wishes to compute a Gröbner basis, this can be easier than computing it with a "direct" approach such as with the famous Buchberger algorithm (Buchberger, 1965) or F_4 (Faugère, 1999).

Because our modular lifting approach reduces the problem to computing just with the zero-dimensional ideals $I + \mathfrak{m}^k$, we can then adapt this basic strategy of the FGLM algorithm to our setting. Our algorithm thus enables potentially efficient computation of Gröbner bases of generic fibers w.r.t. block orders, which is in particular needed for the computation of irreducible decompositions using Gröbner bases outlined above. Our algorithm computes a Gröbner basis of $\mathrm{gen}(I, \mathbf{z})$ for a monomial order of interest using Gröbner bases of the ideals $I + \mathfrak{m}^k$ for another monomial order.

The description and discussion of this algorithm takes up most of Chapter 7. When a "quadratic lifting strategy" is chosen, we show that this algorithm runs in a complexity quasi-linear in the number of terms in $\mathbf{z}$ whose partial degrees are large enough so as to recover the desired Gröbner basis of $\mathrm{gen}(I, \mathbf{z})$ using Padé approximation, see Theorem 7.3.2. This complexity statement would then also likely transport to a p-adic setting as outlined above.

The second algorithm is a combination of the same Hensel lifting technique with the classical F_4 algorithm introduced by Faugère (1999) to compute Gröbner bases. It adapts the technique of so-called *Gröbner tracers* introduced by Traverso (1989), which are used for multi-modular Gröbner basis computations, to the setting of modular lifting. The F_4 algorithm works again by echelonizing a series of Macaulay matrices associated to the input polynomials. Only those rows which do not reduce to zero during this echelonization contribute to the output Gröbner basis and a Gröbner tracer is essentially a data structure which "remembers" which the choice of reducers and which rows did *not* reduce to zero during the computation. When e.g. a Gröbner basis over $\mathbb{Q}$ is required, the idea is then, in a multi-modular setting, to run the F_4 algorithm module one prime p, extract a tracer out of this computation, and apply the computational steps of this tracer modulo other primes $p_1, \ldots, p_r$. This is more efficient then rerunning the F_4 algorithm for every new prime. With probability one, this will give the image of the desired Gröbner basis G over $\mathbb{Q}$ modulo p and $p_1, \ldots, p_r$. G is then recovered using the Chinese Remainder Theorem.

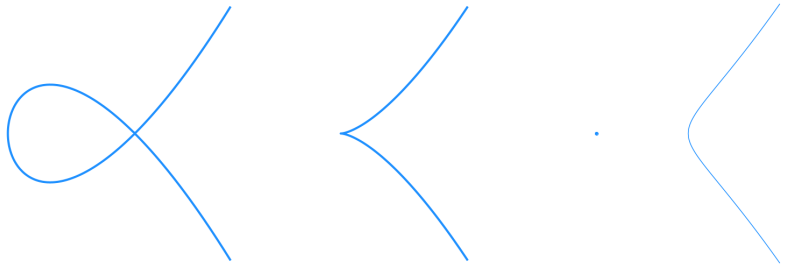
Our adaptation of Gröbner tracers to a modular lifting strategy then poses the task of lifting the echelonizations computed by the F_4 algorithm running on the image of I in $\mathbb{K}[\mathbf{x},\mathbf{z}]/\mathfrak{m} \cong \mathbb{K}[\mathbf{x}]$ modulo larger and larger powers of $\mathfrak{m}$. In other words, we have to solve the problem of lifing an LU-factorization of a given matrix with coefficients in $\mathbb{K}[\mathbf{z}]/\mathfrak{m}^k$ to $\mathbb{K}[\mathbf{z}]/\mathfrak{m}^{k+1}$. For the case where $\mathbf{z} = \{z\}$ is a single variable, we show how to do this in Theorem 7.4.1 which then yields a complexity statement for this second algorithm in line with the complexity of other modular lifting algorithms, i.e. it is the complexity of computing the image of G in $(\mathbb{K}[\mathbf{z}]/\mathfrak{m})[\mathbf{x}]$ times the square of the number of terms up to degree δ in $\mathbf{z}$ where δ is chosen sufficiently large so as to extract G out of its image in $(\mathbb{K}[\mathbf{z}]/\mathfrak{m}^k)[\mathbf{x}]$, see Corollary 7.4.1.
We also briefly point out how both algorithms given in Chapter 7 can potentially be profitably combined.
Let us also comment on how the algorithms we design in Chapter 7 relate to other methods of computing Gröbner basis of generic fibers. As mentioned above, a Gröbner basis of $\mathrm{gen}(I,\mathbf{z})$ can be computed by computing a Gröbner basis for I w.r.t. a suitable block order. Besides the aforementioned issues when it comes to Gröbner basis computations w.r.t. block orders, another problem is potentially that this computation involves the entire I while the output encodes only a subset of the components of $\mathbf{V}(I)$, as in Figure 1.4. Thus would one would like to have algorithms whose computations reflect the fact that $\mathrm{gen}(I,\mathbf{z})$ is potentially simpler than I itself, this is accomplished by using Hensel lifting as sketched above. Other algorithms combining Gröbner basis computations with Hensel lifting are given by Arnold (2003) and Winkler (1988) which combine Buchberger's algorithm with Hensel lifting. These algorithms require the computation of so-called *representation matrices* for Gröbner bases which are polynomial matrices describing how each element in a computed Gröbner basis relates to the input polynomials. Computing these matrices in addition to computing a Gröbner basis comprises a significant overhead. In addition, no complexity analysis for these algorithms is provided by Arnold (2003) and Winkler (1988). The bivariate case is treated by Schost and St-Pierre (2023), including a complexity analysis. Besides this, multi-modular techniques (such as the ones developed by Ebert, 1983; Pauer, 1992) can certainly be used as well. While they are certainly simpler than the techniques we present, they require the evaluation of the variables in $\mathbf{z}$ at multiple points, each of which is required to lie outside a certain hypersurface which is probabilistically assumed. This is in contrast to Hensel lifting techniques which just require one such evaluation.

Computation of Whitney Stratifications

As another way to use Gröbner bases of generic fibers, and thus as potential application of the ideas developed in Chapter 7, we give in Chapter 8, based on the results given by Helmer and Mohr (2024), an algorithm to compute a so-called *Whitney stratification* of a singular algebraic set. In addition, we give an algorithm to *minimize* a given Whitney stratification.
A Whitney stratification provides the basic structure needed to decompose singular algebraic sets into smooth pieces which join together in a desirable way. This has applications for example in physics, see Helmer, Papathanasiou, and Tellander (2024).
Let us start by motivating the concept of a Whitney stratification. For this example we will work over the real numbers $\mathbb{R}$. For a set of polynomials $F \subset \mathbb{Q}[\mathbf{x}]$, we denote $\mathbf{V}_{\mathbb{R}}(F) := \mathbf{V}(F) \cap \mathbb{R}^n$.
Consider now the curve in $\mathbb{R}^2$ defined by the parametric polynomial

$$f_z(x, y) = (y-1)^2 - (x-z)(x-1)^2 \tag{1.1}$$

in variables x, y with parameter z. For $z < 1$ the real algebraic set $\mathbf{V}_{\mathbb{R}}(f_z)$ is a (connected) nodal cubic, for $z > 1$ the real algebraic set $\mathbf{V}_{\mathbb{R}}(f_z)$ has two connected components and is smooth (i.e. is a real manifold), and at $z = 0$ the curve is a cubic cusp, see Figure 1.6.



(a) When $z < 1$ we obtain a nodal cubic with one loop.

(b) When $z = 1$ we obtain a cusp cubic.

(c) When $z > 1$, over the reals, we get a curve with two connected components.

Figure 1.6: Plots of the curve defined by (1.1) for different parameter values z; the topology of the curve changes at $z = 1$. While the rightmost curve is smooth and has two connected components (of different dimensions) in $\mathbb{R}^2$ it is connected and singular, with singularity at $(1, 1)$, in $\mathbb{C}^2$.

To understand what is happening here consider now the algebraic set $X = \mathbf{V}_{\mathbb{R}}(f)$ in $\mathbb{R}^3$ defined by the same polynomial $f(x, y, z) = (y-1)^2 - (x-z)(x-1)^2$, plotted in Figure 1.7. The surface X is singular along the entire line $\mathrm{Sing}(X) = \mathbf{V}_{\mathbb{R}}(x-1, y-1)$ and the projection map $\pi : \mathbb{R}^3 \to \mathbb{R}$ onto the last coordinate, $(x, y, z) \to z$, restricted to either the manifold $X - \mathbf{V}_{\mathbb{R}}(x-1, y-1)$ or to the manifold $\mathbf{V}_{\mathbb{R}}(x-1, y-1)$

is a submersion, however the behavior of the fibers of π restricted to $X - \mathbf{V}_{\mathbb{R}}(x-1, y-1)$ depends on whether z is greater than or less than 1, while the behavior of the fibers of the restriction to $\mathbf{V}_{\mathbb{R}}(x-1, y-1)$ is independent of z.

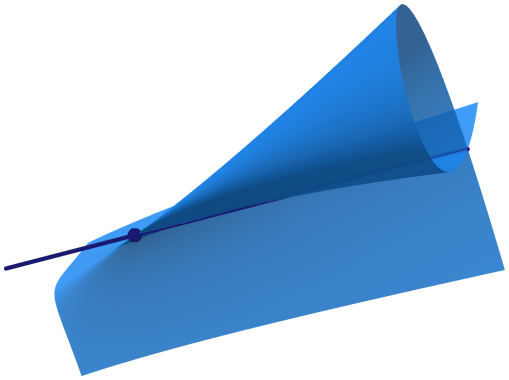


Figure 1.7: The surface $X = \mathbf{V}\left((y-1)^2 - (x-z)(x-1)^2\right)$ in $\mathbb{R}^3$. Its singular locus is $Y = \mathbf{V}(x-1, y-1)$ and its Whitney stratification arises from the flag $\{(1,1,1)\} \subset Y \subset X$. The dimension 2 stratum is $X - Y$, the two dimension 1 strata are the connected components of $Y - \{(1,1,1)\}$, and the dimension 0 stratum is $(1,1,1)$.

Whitney's Condition (B) fails for the pair of manifolds $X - \mathbf{V}(x-1, y-1)$ and $\mathbf{V}(x-1, y-1)$ at the singular point $(1,1,1)$, this condition (B), given in terms of limits of tangent directions at singular points, captures the fact that the point $(1,1,1)$ is "more singular" in X than the rest of the singular locus. As it turns out, this in particular allows us to know exactly when the topology of the fibers of the map π change. More precisely a classical result known as *Thom's Isotopy Lemma* (see e.g. Proposition 11.1 in Mather, 2012) allows us to partition the codomain of (proper) maps into regions of constant topology; Whitney's condition is a critical component of this construction as illustrated by the example above.

The algorithm to compute a Whitney stratification we present in Chapter 8 is not based on a new geometric insight on the concept of Whitney stratifications but rather utilizes an algebraic criterion to check Whitney's condition (B) given in Helmer and Nanda (2023). This criterion uses *primary decomposition* of polynomial ideals. A primary decomposition is essentially an ideal-theoretic analogue of irreducible decompositions of algebraic sets, which keeps track of additional embedded and multiplicity structure encoded in I which is lost when looking only at $\mathbf{V}(I)$.

In Helmer and Nanda (2023) this criterion was applied by directly computing a primary decomposition of some involved ideals, using Gröbner basis computations, in order to obtain an algorithm computing a Whitney stratification. We show in Chapter 8 that the same criterion may be applied by computing a single Gröbner basis of a

generic fiber, thus avoiding a potentially difficult primary decomposition.
This modification is critical: The computation of a Whitney stratification of a given algebraic set X using the above mentioned algebraic criterion is done by computing with ideals associated to the so-called *conormal space*, encoding hyperplanes which contain the tangent spaces of X. These ideals can be complicated (i.e. have many primary components of high degree) even when X itself is relatively simple:

Example 1.0.4. Consider the algebraic set $X := \mathbf{V}(x_1^6 + x_2^6 + x_1^4x_3x_4 + x_3^3) \subset \mathbb{A}^4$ defined over $\mathbb{Q}$. The singular locus of X is just the line $\mathbf{V}(x_1, x_2, x_3)$. The fiber of the singular locus in the conormal space of X is a non-radical ideal in twice as many variables as X with one primary component of degree 11, one of degree 3 and seven of degree 1.

We note that, to apply this algebraic criterion, it is essential to have access also to the embedded structure given by a primary decomposition. This requires an ideal-theoretic perspective, "purely geometric" techniques such as geometric resolutions or triangular sets do not suffice.
The algorithm we present to compute a Whitney stratification may produce more superflous algebraic sets compared to the algorithm presented by Helmer and Nanda (2023). Motivated by this, it is coupled in Chapter 8 with an algorithm to minimize a given Whitney stratification, based on a multiplicity criterion by Teissier (1982).

Software

Let us briefly remark on the software that has been written for the purpose of this thesis. All contributions in this thesis consist of the design of algorithms and are accompanied by software implementations, written in `Julia`, with the exception of the implementation accompanying Chapter 8, which was written by Martin Helmer in `Macaulay2`. Links to these software implementations can be found in the "Benchmarks" section of each chapter.
We want to highlight in particular the `Julia`-package `AlgebraicSolving.jl`, designed originally as a `msolve`-wrapper for `Julia`. This package, publicly available in the `Julia` package repositories, contains an implementation of the author of a signature-based Gröbner basis algorithm, using fast linear algebra, some new ideas by Lairez (2024) and standard techniques for optimizing Gröbner basis computations, see e.g. Monagan and Pearce (2015), as well as an implementation of the algorithm presented in Chapter 6 which makes use of the aforementioned signature-based Gröbner basis algorithm.

OUTLINE OF PRELIMINARIES

Let us close this introduction by briefly outling the preliminaries of this thesis, i.e. Part i.
Chapter 2 focuses mostly on the needed results from commutative algebra and algebraic geometry to design and show correctness of the algorithms in Chapter 4, Chapter 5 and Chapter 6. We also recall the properties of generic fibers of ideals with regard to the decomposition of algebraic sets and ideals in Section 2.4 and briefly introduce the concept of Whitney stratifications in Section 2.5.
Chapter 3 gives an introduction to the concept and computation of Gröbner bases. We recall one of the most classical algorithm to compute them, i.e. Buchberger's algorithm (Section 3.3) as well as the F_4 algorithm (Section 3.4) and FGLM (Section 3.5). A particular focus is given towards signature-based algorithms in Section 3.6. In addition we introduce decomposition algorithms based on Gröbner bases of generic fibers in Section 3.2.1 and Section 3.2.2.

Part I

PRELIMINARIES

2

PREREQUISITES FROM COMMUTATIVE ALGEBRA AND ALGEBRAIC GEOMETRY

Throughout this chapter, we denote by $R := \mathbb{K}[x_1, \ldots, x_n]$ a polynomial ring in n variables over a field $\mathbb{K}$. We will often use the shorthand notation $\mathbf{x} := \{x_1, \ldots, x_n\}$. For an ideal $I \subset R$ we will denote the radical of I as

$$\sqrt{I} := \left\{ f \in R \;\middle|\; f^k \in I \text{ for some } k \in \mathbb{N} \right\}.$$

Fixing an algebraic closure $\overline{\mathbb{K}}$ of $\mathbb{K}$, we denote by $\mathbb{A}^n$ the space $\overline{\mathbb{K}}^n$ equipped with the Zariski topology, whose closed sets are precisely the algebraic sets defined below.
For any set $S \subset R$ we denote

$$\mathbf{V}(S) := \{p \in \mathbb{A}^n \mid f(p) = 0 \; \forall f \in S\}.$$

Definition 2.0.1 (Algebraic Set)**.** We call a set of the form $\mathbf{V}(S)$ an *algebraic set* in $\mathbb{A}^n$.

Note that $\mathbf{V}(S) = \mathbf{V}(\langle S \rangle) = \mathbf{V}(\sqrt{\langle S \rangle})$ for any set $S \subset R$.
Conversely, for any set $T \subset \mathbb{A}^n$ we denote by $\mathbf{I}(T)$ the radical ideal

$$\mathbf{I}(T) := \{f \in R \mid f(p) = 0 \; \forall p \in T\}.$$

The closure of a set $T \subset \mathbb{A}^n$ in the Zariski topology is denoted $\overline{T}$. Recall that $\overline{T} = \mathbf{V}(\mathbf{I}(T))$ by the Nullstellensatz, see e.g. Theorem 1.6 in Eisenbud (1995). Throughout this thesis all considered algebraic sets are assumed to be defined over $\mathbb{K}$. This means that for every algebraic set X appearing in this thesis we suppose that $\mathbf{I}(X)$ can be generated by polynomials lying in R.
The biggest part of this thesis is devoted to the presentation of algorithms which either compute, or can be used for the computation of, various *decompositions* of algebraic sets, i.e. the task is to write a given algebraic set as a union of other algebraic sets. This chapter gives the necessary preliminaries from commutative algebra and algebraic geometry for these algorithms.
All material in this chapter can be found in standard textbooks on commutative algebra and algebraic geometry. We refer to Greuel and Pfister (2007) and Cox, Little, and O'Shea (2015) for textbooks taking a computational perspective, the first more from the perspective of commutative algebra and the latter more from the perspective of algebraic geometry.
Standard theoretical references of commutative algebra include Eisenbud (1995), Matsumura (1987) and Atiyah and MacDonald (1994). The first gives a more extensive account of the subject while the latter two focus mostly on the essentials of the subject.

2.1 DECOMPOSITION OF ALGEBRAIC SETS AND POLYNOMIAL IDEALS

The theoretical underpinning of our decomposition algorithms is the well known decomposition of an algebraic set into *irreducible* components which we recall here.

Definition 2.1.1 (Prime Ideals, Irreducibility)**.** An ideal $I \subset R$ is called *prime* if for $f, g \in R$ with $fg \in I$ we have $f \in I$ or $g \in I$. An algebraic set $X \subset \mathbb{A}^n$ is called *irreducible* if $\mathbf{I}(X)$ is a prime ideal. Equivalently, again by the Nullstellensatz, X cannot be written as a union of two proper algebraic subsets defined over $\mathbb{K}$.

Now we have the well-known

Theorem 2.1.1 (Theorem 2 in Chapter 6 of Cox, Little, and O'Shea, 2015)**.** *Any algebraic set $X \subset \mathbb{A}^n$ can be written as a finite union*

$$X = \bigcup_{i=1}^{r} X_i$$

with each X_i irreducible. This decomposition is unique up to reordering the X_i. We call each X_i an irreducible component *of X and write* $\mathrm{Irred}(X) := \{X_1, \ldots, X_r\}$*. Equivalently, by the Nullstellensatz, every radical ideal $I \subset R$ can be written, uniquely up to order, as an intersection*

$$I = \bigcap_{i=1}^{r} P_i$$

where each of the P_i is a prime ideal. The prime ideals P_i are the prime ideals minimal with the property that they contain I and are called the minimal primes *over I and are denoted* $\mathrm{MinAss}(I)$*.*
The operation $\mathbf{I}(\bullet)$ establishes a one-to-one correspondence between the irreducible components of X and the minimal primes over $\mathbf{I}(X)$ and the operation $\mathbf{V}(\bullet)$ establishes a one-to-one correspondence between the minimal primes over I and the irreducible components of $\mathbf{V}(I)$.

By the dimension of an algebraic set or a polynomial ideal we always mean its Krull dimension, following e.g. Chapter 8 in Eisenbud (1995):

Definition 2.1.2 (Krull Dimension)**.** For a prime ideal $I \subset R$ we define the *Krull dimension* of I, denoted $\dim(I)$, as the length of any maximal chain of prime ideals in R/I. For an algebraic set $X \subset \mathbb{A}^n$ we define $\dim(X) := \dim(\mathbf{I}(X))$. By the codimension of an algebraic set X (or a polynomial ideal I) we mean $n - \dim(X)$ (or $n - \dim(I)$).

Most of our algorithms try to decompose a given algebraic set into *equidimensional* algebraic sets:

Definition 2.1.3 (Equidimensionality)**.** An algebraic set in $\mathbb{A}^n$ is called *equidimensional* if all of its irreducible components have the same dimension.

Remark 2.1.1. Note that, while primeness of ideals is not preserved under field extensions of $\mathbb{K}$, equidimensionality of the corresponding algebraic sets is, essentially because dimension is preserved under integral extensions of rings by "Lying Over" and "Going Up", see e.g. Proposition 4.15 in Eisenbud (1995).

From a purely ideal-theoretic viewpoint, a notion closely corresponding to an irreducible algebraic set is that of a *primary ideal*:

Definition 2.1.4 (Primary Ideal)**.** An ideal $Q \subset R$ is called *primary* if for $f, g \in R$ with $fg \in Q$ we have $f \in Q$, $g \in Q$ or $f, g \in \sqrt{Q}$.

Definition 2.1.5 (Associated Prime)**.** A prime ideal $P \subset R$ is called an *associated prime* of an ideal I if there exists $f \in R/I$ with

$$\operatorname{ann}_I(f) := \{g \in R \mid gf = 0 \mod I\} = P.$$

The set of all associated primes is denoted $\operatorname{Ass}(I)$.

Just as an algebraic set can be decomposed into irreducible components, any polynomial ideal can be decomposed (possibly non-uniquely) into primary components:

Theorem 2.1.2 (Theorem 3.10 in Eisenbud, 1995)**.** *For any ideal $I \subset R$ the set $\operatorname{Ass}(I)$ is finite and we have $\operatorname{MinAss}(I) \subseteq \operatorname{Ass}(I)$. For every $P \in \operatorname{Ass}(I)$ there exists a primary ideal Q_P with $\sqrt{Q_P} = P$ and*

$$I = \bigcap_{P \in \operatorname{Ass}(I)} Q_P.$$

The choice of Q_P is unique if $P \in \operatorname{MinAss}(I)$. Such a decomposition of I is called a primary decomposition *of I. The primes in $\operatorname{Ass}(I) \setminus \operatorname{MinAss}(I)$ are called the* embedded primes *of I.*

In line with Definition 2.1.3, we say that

Definition 2.1.6. An ideal $I \subset R$ is *equidimensional* if all of its associated primes have the same dimension.

2.2 COLON IDEALS, SATURATION IDEALS AND LOCALLY CLOSED SETS

In our algorithms we will more often work with *locally closed sets* instead of algebraic sets:

Definition 2.2.1 (Locally Closed Set)**.** A set $X \subset \mathbb{A}^n$ is called *locally closed* if it can be written in the form $X = Y \setminus Z$ with Y and Z algebraic.

By the irreducible components of a locally closed set X we mean the set

$$\left\{ Y \cap X \;\middle|\; Y \in \operatorname{Irred}(\overline{X}) \right\}.$$

We say that X is *equidimensional* if $\overline{X}$ is.
Ideal-theoretically, locally closed sets are closely linked to colon and saturation ideals:

Definition 2.2.2 (Colon and Saturation Ideal)**.** For two ideals $I, J \subset R$ we define the *colon ideal* of I by J as

$$(I : J) := \{f \in R \mid fJ \subset I\}$$

and the *saturation* of I by J as

$$(I : J^\infty) := \left\{f \in R \;\middle|\; fJ^k \subset I \text{ for some } k \in \mathbb{N} \setminus \{0\}\right\}.$$

For a single element $f \in R$ we write $(I : f)$ and $(I : f^\infty)$ for $(I : \langle f \rangle)$ and $(I : \langle f \rangle^\infty)$ respectively.

Remark 2.2.1. Throughout this thesis we will frequently write

$$(I + J : K^\infty) \text{ or } (I + J : K)$$

for ideals $I, J, K \subset R$. This is always understood to refer to the ideals $((I + J) : K^\infty)$ or $((I + J) : K)$.

First, one concludes from the definitions of colon and saturation ideals that

Proposition 2.2.1. *For three ideals $I, J, K \subset R$ we have*

$$(I \cap J : K) = (I : K) \cap (J : K)$$

and

$$(I \cap J : K^\infty) = (I : K^\infty) \cap (J : K^\infty)$$

From this, and the definition of a primary ideal one now concludes

Proposition 2.2.2. *Let $I, J \subset R$ be two ideals and let $I = \bigcap_{i=1}^{s} Q_i$ be a primary decomposition. Then*

$$(I : J^\infty) = \bigcap_{J \not\subset \sqrt{Q_i}} Q_i$$

is a primary decomposition of $(I : J^\infty)$.

This proposition will be used repeatedly in Section 2.3. It also implies, using the correspondence between minimal primes over an ideal and the irreducible components of the corresponding algebraic set given in Theorem 2.1.1

Proposition 2.2.3. *For any two ideals I and J we have*

$$\overline{\mathbf{V}(I) \setminus \mathbf{V}(J)} = \mathbf{V}((I : J^\infty)).$$

Remark 2.2.2. Geometrically, Proposition 2.2.2 now translates to

$$\operatorname{Irred}(\mathbf{V}(I) \setminus \mathbf{V}(J)) = \{Y \cap \mathbf{V}(I) \setminus \mathbf{V}(J) \mid Y \in \operatorname{Irred}(\mathbf{V}(I)), Y \not\subset \mathbf{V}(J)\}.$$

In particular, the irreducible components of $\mathbf{V}(I) \setminus \mathbf{V}(f)$ for some $f \in R$ correspond to those components of $\mathbf{V}(I)$ on which f is not identically zero.

In Chapter 4 we will use the following

Proposition 2.2.4. *Let* $I, J \subset R$ *be two ideals. If* I *is radical then* $(I : J) = (I : J^\infty)$.

Proof. Using Proposition 2.2.1, we may suppose that I is a prime ideal. Let $p \in (I : J^\infty)$, i.e. pJ^k in I for some $k \in \mathbb{N}$. Then either $p \in I$ (and so $p \in (I : J)$) or $p \notin I$ and so $J^k \subset I$ which implies $J \subset I$. This then implies $1 = (I : J) \subseteq (I : J^\infty) \subseteq 1$, establishing the proposition. □

In Chapter 5, we will work with locally closed sets by looking at their intersection with an appropriate linear space. This is based on the following

Lemma 2.2.1. *Let* X *be an equidimensional locally closed set of dimension* d *and let* $f \in R$. *For a generic linear linear subspace* $L \subset \mathbb{A}^n$ *of codimensional* d *we have*

$$f \in \mathbf{I}(X) \quad \Leftrightarrow \quad f \in \mathbf{I}(X \cap L)$$

Proof. We always have $\mathbf{I}(X) \subseteq \mathbf{I}(X \cap L)$. Conversely, assume that $f \notin \mathbf{I}(X)$. Let $U := \{p \in X \mid f(p) \neq 0\}$. U is a Zariski-open subset of X and it is not empty, by hypothesis. Since X is equidimensional, the Zariski closure of U has dimension d and the intersection $U \cap L$ is nonempty (because L is generic). Therefore f is nonzero on a nonempty subset of $X \cap L$. In particular, $f \notin \mathbf{I}(X)$. □

2.2.1 *Further Properties of Colon and Saturation Ideals*

Here, we gather various further properties of colon and saturation ideals that will be used in later chapters.

Proposition 2.2.5. *Let* I, J *be two ideals in* R. *Then*

$$(I : (I : J^\infty)^\infty) = (I : (I : J^\infty)) .$$

Proof. By Proposition 2.2.1 we may suppose that I is a primary ideal. In this case

$$(I : J^\infty) = \begin{cases} 1 & \text{if } J \subset \sqrt{I} \\ I & \text{otherwise.} \end{cases}$$

and the statement follows. □

In the following we write $I \overset{\text{rad}}{=} J$ for two ideals $I, J \subset R$ if $\sqrt{I} = \sqrt{J}$. We record the following variant of the prime avoidance lemma:

Proposition 2.2.6. *Let* $P_1, \ldots, P_s \subset R$ *be a collection of prime ideals and let* $g_1, \ldots, g_u \in R$ *s.t.* $\langle g_1, \ldots, g_u \rangle \not\subset P_i$ *for any* i. *Then there exists a non-empty Zariski-open subset* $D \subset \mathbb{A}^u$ *such that*

$$(a_1, \ldots, a_u) \in D \Rightarrow \sum_{j=1}^{u} a_j g_j \notin P_i \quad \forall i.$$

Proof. Let V be the $\overline{\mathbb{K}}$-span of $g_1, \ldots, g_r$ and let W_i be the $\mathbb{K}$-vector space $W_i := V \cap P_i$. If $V = W_i$ for some i then $\langle g_1, \ldots, g_r \rangle \subset P_i$, a contradiction. Therefore W_i is properly contained in V. The union of all W_is corresponds to a Zariski closed subset of $\mathbb{A}^r$, which is proper, since $\overline{\mathbb{K}}$ is infinite. This proves the statement. See also the solution of Exercise 3.19 on p. 717 of Eisenbud (1995). □

Lemma 2.2.2. *Let* $I, J \subseteq R$ *be two ideals with* $J = \langle g_1, \ldots, g_u \rangle$.

(1) If $S = R[t_1, \ldots, t_u]$ *then* $(I : J^\infty) = \left(IS : (\sum_{j=1}^u t_j g_j)^\infty\right) \cap R$.

(2) There exists a Zarisiki-open subset $D \subset \mathbb{A}^u$ *such that*

$$(a_1, \ldots, a_u) \in D \Rightarrow (I : J^\infty) = \left(I : (\sum_{j=1}^u a_j g_j)^\infty\right).$$

(3) If $K \overset{\text{rad}}{=} J$ *then* $(I : K^\infty) \overset{\text{rad}}{=} (I : J^\infty)$.

Proof. Proof of (1): The left-to-right inclusion follows from the definition of saturation ideals. For the right-to-left inclusion note that $S/IS \cong R/I[t_1, \ldots, t_u]$. Therefore, if $h \in \left(IS : (\sum_{j=1}^u t_j g_j)^\infty\right) \cap R$ then, by expanding $h(\sum_{j=1}^u t_j g_j)^k$ for suitable k, we find a polynomial $p \in R[t_1, \ldots, t_u]$ which is zero in $R/I[t_1, \ldots, t_u]$. This can only happen if $h \in (I : J^\infty)$.
Proof of (2): The ideal $(I : J^\infty)$ is the intersection of those primary components of I whose radical does not contain J (Proposition 2.2.2). Now we may appeal to Proposition 2.2.6: There is a Zariski-open subset $D \subset \mathbb{A}^u$ such that for any $(a_1, \ldots, a_u) \in D$ we have that $\sum_{i=1}^u a_i g_i$ is not contained in any of the associated primes of I which do not contain J. Then the statement follows again from Proposition 2.2.2.
Proof of (3): If $p \in R$ such that $p^k J^l \subset I$ for $k, l \in \mathbb{N}$ then for a suitably large $m \in \mathbb{N}$ we have $K^m \subseteq J^l$ so $p^k K^m \subset I$ and hence $p \in \sqrt{(I : K^\infty)}$. □

Lemma 2.2.3. *Let* $I, J \subseteq R$ *be two ideals and let* $J = \langle g_1, \ldots, g_t \rangle$. *Then*

$$(I : J) \overset{\text{rad}}{=} (I : g_1) \cap (I + \langle g_1 \rangle : g_2) \cap \cdots \cap (I + \langle g_1, \ldots, g_{t-1} \rangle : g_t).$$

Proof. The inclusion "$\subseteq$" is obvious. Now, let $p \in R$ be such that

$$p^m \in (I : g_1) \cap (I + \langle g_1 \rangle : g_2) \cap \cdots \cap (I + \langle g_1, \ldots, g_{t-1} \rangle : g_t).$$

for some $m \in \mathbb{N}$. Then we have in particular $p^m g_1 \in I$. Now let $i > 1$. By induction, if for some $k \in \mathbb{N}$ we have $p^k g_j \in I$ for all $j \leqslant i$ then

$$p^{km} g_{i+1} = p^k f + p^k a_1 g_1 + \cdots + p^k a_i g_i \in I$$

for a suitable $f \in I$, $a_1, \ldots, a_i \in R$ and so $p^{km} \in (I : g_{i+1})$. We deduce that a power of p actually lies in $(I : J)$ which ends the proof. □

2.3 REGULAR INTERSECTION & REGULAR SEQUENCES

A large part of our algorithms is based on the notion of *regular intersection* and *regular sequences* which we treat here.
We first recall the concept of *localization* of a ring S and an S-module M:

Definition 2.3.1 (Localization)**.** Let S be any ring and let $C \subset S$ be multiplicatively closed, i.e. the product of two elements in C lies again in C. The *localization* of S at C, denoted $\mathrm{loc}(S, C)$, is the ring consisting of the formal fractions s/c, $s \in S$, $c \in C$ with addition, multiplication and equality defined as for $\mathbb{Q}$.
The localization of M at C is the $\mathrm{loc}(S, C)$-module consisting of the formal fractions a/c with $a \in M$, $c \in C$ with addition, multiplication by scalars and equality defined just as for $\mathrm{loc}(S, C)$.

For any ring S, and S-module M, any $f \in S$ and any prime ideal $P \subset S$ we use the following notation:

$$\begin{aligned} S_f &:= \mathrm{loc}(S, \{f^k \mid k \in \mathbb{N}\}) \\ M_f &:= \mathrm{loc}(M, \{f^k \mid k \in \mathbb{N}\}) \\ S_P &:= \mathrm{loc}(S, S \setminus P) \\ M_P &:= \mathrm{loc}(M, S \setminus P) \end{aligned}$$

If we use a point $p \in \mathbb{A}^n$ as a localization index in the following, we mean the localization at the maximal ideal $\mathbf{I}(p)$ considered in the appropriate ring.

Definition 2.3.2 (Regular Intersection, Regular Sequences)**.** Let $X \subset \mathbb{A}^n$ be a locally closed set and let $f \in R$. We say that f *intersects* X *regularly at a point* $p \in X$ if f is not a zero divisor in $R_p/\mathbf{I}(X)_p$. It *intersects* X *regularly* if it is not a zero divisor in $R/\mathbf{I}(X)$. A sequence $f_1, \ldots, f_c$ in R is called a *regular sequence* or *complete intersection* if f_i intersects $\mathbf{V}(f_1, \ldots, f_{i-1})$ regularly for every $i = 2, \ldots, c$.

Regular intersection can be described ideal-theoretically as follows:

Lemma 2.3.1. *Let I be any ideal in R and let $f \in R$. Then f does not regularly intersect $\mathbf{V}(I)$ if and only if there exists a minimal prime P over I such that $f \in P$. In particular, in this situation, by Proposition 2.2.2, we have $I \subsetneq (I : f^\infty)$.*

Proof. If $P_1, \ldots, P_s$ are the minimal primes over I then we can write $\sqrt{I} = \mathbf{I}(\mathbf{V}(I)) = \bigcap_{i=1}^{s} P_i$. Suppose that f is contained in none of the P_i and that $g \in (I : f^\infty)$. Then, by the definition of a prime ideal, we have $g \in P_i$ for all i and so $g \in \sqrt{I}$. Hence f regularly intersects $\mathbf{V}(I)$, a contradiction. Conversely, if f is contained in one of the P_i but not all of them then we may choose any $g \in \bigcap_{j \neq i} P_j \neq \sqrt{I}$ and then $gf \in \sqrt{I}$ which shows that f does not regularly intersect X. □

Recall that the minimal primes over I correspond to the irreducible components of $\mathbf{V}(I)$. Now we want to point out how regular intersection relates to equidimensionality:

Proposition 2.3.1. *Suppose $X \subset \mathbb{A}^n$ is an equidimensional locally closed set of codimension c and suppose some $f \in R$ intersects X regularly. Then $X \cap \mathbf{V}(f)$ is either empty or equidimensional of codimension $c+1$. In particular, any regular sequence of length c defines an equidimensional algebraic set of codimension c.*

Proof. By Krull's principal ideal theorem (e.g. Theorem 2.10 in Eisenbud (1995)), any irreducible component of $X \cap \mathbf{V}(f)$ has has codimension at most $c+1$. Suppose that $Y = \mathbf{V}(P)$ is an irreducible component of X of codimension c. Then P is a minimal prime over $\mathbf{I}(X)$ and $f \in P$ which implies that f is a zero divisor in S by Lemma 2.3.1, a contradiction. □

We further need

Lemma 2.3.2. *Let X be an equidimensional locally closed set of dimension d. For a generic degree one polynomial $\ell \in R$, $X \cap \mathbf{V}(\ell)$ is equidimensional of dimension $\dim(X) - 1$.*

Proof. We may assume that X is irreducible. Then we need to prove that for a generic choice of ℓ, X is not contained in $\mathbf{V}(\ell)$ or in other words that ℓ is not contained in $P := \mathbf{I}X$. But this follows from Proposition 2.2.6 applied to the set $\{1\} \cup \mathbf{x}$. □

Given a locally closed set X and $f \in R$ we now want to describe the set of points $p \in X$ at which f regularly intersects X. For this, suppose that f is a zero divisor in $R/\mathbf{I}(X)$, i.e. that there exists $g \in R \setminus \mathbf{I}(X)$ such that $gf \in \mathbf{I}(X)$. If now, f intersects X regularly at some point $p \in X$ then a necessary condition is certainly that $g \in \mathbf{I}(X)_p$. If we now replace X by the locally closed set

$$Y := X \setminus \left[\overline{X \setminus \mathbf{V}(g)}\right]$$

then g vanishes at every point of Y and f will be "more regular" over Y than over X. We may summarize this paragraph in the following

Proposition 2.3.2. *Suppose $X \subset \mathbb{A}^n$ is a locally closed set and let $f \in R$. Let $J := (\mathbf{I}(X) : f)$. Let*

$$\operatorname{reg}(X, f) := \{p \in X \mid f \textit{ regularly intersects } X \textit{ at } p\}.$$

Then

$$\operatorname{reg}(X, f) = X \setminus \left[\overline{X \setminus \mathbf{V}(J)}\right].$$

Proof. First note that

$$\overline{X \setminus \mathbf{V}(f)} = \left\{p \in \overline{X} \;\middle|\; f \notin \mathbf{I}(X)_p\right\}.$$

Indeed, by Theorem 3.1 in Eisenbud (1995)

$$\mathrm{MinAss}(I_p) = \{P_p \mid P \in \mathrm{MinAss}(I), p \in \mathbf{V}(P)\}$$

and the statement follows from Proposition 2.2.2. Hence we have

$$X \setminus \mathbf{V}(J) = \{p \in X \mid f \in \mathbf{I}(X)_p\}.$$

Now let $p \in X$ such that f is a zero divisor modulo $\mathbf{I}(X)_p$. Then there is an irreducible component Y of X with $p \in Y$ such that $f \in \mathbf{I}(Y)$, again by Proposition 2.2.2. Hence $p \in \overline{X \setminus \mathbf{V}(J)}$. This proves the desired statement. □

Example 2.3.1. To give a simple example, let $R := \mathbb{Q}[x, y, z]$, $X = \mathbf{V}(xy)$ and $f = xz$. Note that xy, xz defines a regular sequence at a point $p \in X$ exactly when x is a unit in R_p. Correspondingly we find with $J := (\mathbf{I}(X) : f) = \langle y \rangle$

$$X \setminus \overline{[X \setminus \mathbf{V}(J)]} = X \setminus \overline{X \setminus \mathbf{V}(y)} = X \setminus \mathbf{V}(x).$$

Note that the definition of a regular sequence depends on the ordering of the sequence elements. Using this definition forces our algorithms that use it to process the equations of our input one-by-one. As an alternative we will use the following theorem by Vasconcelos (1967):

Theorem 2.3.1. *Let* $f_1, \ldots, f_c \in R$, $I := \langle f_1, \ldots, f_r \rangle$ *and* $h \in R$. *Then* $f_1, \ldots, f_c$ *is a regular sequence at every point* $p \in \mathbf{V}(I) \setminus \mathbf{V}(h)$ *if and only if* $(I/I^2)_h$ *is a free* $(R/I)_h$*-module, with basis given by the images of* $f_1, \ldots, f_c$ *in* I/I^2.

Example 2.3.2. Again let $R := \mathbb{Q}[x, y, z]$ and consider the sequence xy, xz in R. Then, in I/I^2, we have the relation

$$z \cdot xy - y \cdot xz = 0$$

but neither y nor z lie in I so that xy, xz does not form a local regular sequence.

2.4 GENERIC FIBERS OF POLYNOMIAL IDEALS

One concept that can be used for the decomposition of a polynomial ideal $I \subset R$ or the algebraic set $\mathbf{V}(I) \subset \mathbb{A}^n$ is that of a *generic fiber* of I, which we introduce here. Recall that we write $R = \mathbb{K}[\mathbf{x}]$ where $\mathbf{x}$ is a finite set of variables.

Definition 2.4.1 (Independent Subsets)**.** Let $I \subset R$ be a polynomial ideal.

(1) An *independent subset* of I is a subset $\mathbf{y} \subset \mathbf{x}$ such that $I \cap \mathbb{K}[\mathbf{y}] = 0$. It is a *maximally independent subset* (MIS) if it is an independent subset of I of maximal cardinality.

(2) For any MIS $\mathbf{y}$ of I we call

$$\operatorname{gen}(I, \mathbf{y}) := I\mathbb{K}(\mathbf{y})[\mathbf{x} \setminus \mathbf{y}]$$

a *generic fiber* of I. Here $\mathbb{K}(\mathbf{y})$ denotes the field of rational fractions in $\mathbf{y}$ with coefficients in $\mathbb{K}$.

First we have

Proposition 2.4.1 (see e.g. p. 4 in Decker, Greuel, and Pfister, 1999)**.** *Any MIS of a polynomial ideal* $I \subset R$ *has cardinality equal to* $\dim(I)$.

Let us now point out how generic fibers relate to ideal decomposition. Generic fibers of an ideal I can be used to isolate components of dimension equal to $\dim(I)$:

Proposition 2.4.2. *Let* $I \subset R$ *be a polynomial ideal with generic fiber* $\operatorname{gen}(I, \mathbf{y})$ *for some subset* $\mathbf{y} \subset \mathbf{x}$*. Define* $J := \operatorname{gen}(I, \mathbf{y}) \cap R$*. Then*

$$\operatorname{Ass}(J) = \{P\mathbb{K}(\mathbf{y})[\mathbf{x} \setminus \mathbf{y}] \mid P \in \operatorname{Ass}(I) \text{ and } P \cap \mathbb{K}[\mathbf{y}] = 0\}.$$

In particular J *is equidimensional of dimension equal to* $\dim(I)$.

Proof. For the first claim in the proposition, note that $\mathbb{K}(\mathbf{y})[\mathbf{x} \setminus \mathbf{y}]$ is the localization of R at the multiplicatively closed set $\mathbb{K}[\mathbf{y}] \setminus \{0\}$. Now we may appeal to the behaviour of associated primes under localization see e.g. Theorem 3.10 in Eisenbud (1995). Note that if P is a minimal prime over I with $P \cap \mathbb{K}[\mathbf{y}] = 0$ then Proposition 2.4.1 implies that $\dim(P) \geqslant \dim(I)$. But then $\dim(P) \leqslant \dim(I)$, since P is minimal over I, proving the proposition. □

Later we will give two algorithms to compute Gröbner bases (see Chapter 3) of generic fibers based on Hensel lifting ideas.

2.5 WHITNEY STRATIFICATIONS OF SINGULAR VARIETIES

Now, throughout this section fix $\mathbb{K} = \mathbb{Q}$ and $R := \mathbb{K}[\mathbf{x}]$ with $\mathbf{x} := \{x_1, \ldots, x_n\}$. Note that now $\mathbb{A}^n = \mathbb{C}^n$ equipped with the Zariski topology. Let us recall

Definition 2.5.1 (Jacobian Matrix, Singular Locus)**.** Suppose $X := \mathbf{V}(F)$ is equidimensional where $F = (f_1, \ldots, f_r) \subset R$ defines a radical ideal and that $\operatorname{codim}(X) = c$. The *Jacobian matrix* of F is defined as

$$\mathbf{J}_F(x) := \begin{bmatrix} \partial_{x_1} f_1 & \ldots & \partial_{x_n} f_1 \\ \vdots & & \vdots \\ \partial_{x_1} f_r & \ldots & \partial_{x_n} f_r \end{bmatrix}$$

Denote by M the set of $c \times c$-minors of $\mathbf{J}_F$. The *singular locus* of X is defined as

$$\operatorname{Sing}(X) := X \cap \mathbf{V}(M).$$

If $\mathrm{Sing}(X) \neq \emptyset$ then X will be called *singular*, otherwise *smooth*. The *tangent space* of X at a point $p \in X \setminus \mathrm{Sing}(X)$ is defined as the $\dim(X)$-dimensional vector space

$$T_pX = \ker(\mathbf{J}_F(x)).$$

Smooth varieties are in particular manifolds and so inherit their local (euclidean) topology from that of $\mathbb{C}^n$. We introduce here briefly the concept of *Whitney stratifications* which is a certain well-behaved way of partitioning a given singular X into smooth pieces. In Chapter 8 we give an algorithm to compute such Whitney stratifications, which in line with the rest of this thesis, uses techniques for equidimensional decomposition, in particular based on the concept of generic fibers.
We denote by $\mathbb{P}^k$ the projective space of dimension k over $\mathbb{C}$ and by $\mathrm{Gr}(k, n)$ (*Grassmannian*) the projective algebraic set whose points correspond to vector spaces of dimension k in $\mathbb{C}^n$. Note that for every $p \in X \setminus \mathrm{Sing}(X)$, T_pX defines a point in $\mathrm{Gr}(\dim(X), n)$.
We start with

Definition 2.5.2 (Whitney's Condition (B))**.** Let $X, Y \subset \mathbb{A}^n$ be smooth with $\dim(Y) < \dim(X)$. The pair satisfies *Whitney's condition (B)* if for any $p \in Y$

- and any sequences $(x_i) \subset X$ and $(y_i) \subset Y$ converging to p,
- if the secant lines $\ell_i = [x_i, y_i] \in \mathbb{P}^{n-1}$ converge to some $\ell \in \mathbb{P}^{n-1}$,
- and if the tangent spaces $T_{x_i}X \in \mathrm{Gr}(\dim(X), n)$ converge to some $T \in \mathrm{Gr}(\dim(X), X)$,

then $\ell \subset T$.

Example 2.5.1. Consider the *Whitney umbrella* $X := \mathbf{V}(x^2 - y^2z) \subset \mathbb{R}^3$. The singular locus of X is the z-axis $\mathbf{V}(x, z)$, the vertical line in Figure 2.1. Whitney's condition is not satisfied at the origin $(0, 0, 0)$: If we choose a sequence (x_i) in $X \setminus \mathbf{V}(x, z)$ converging to the origin then we obtain the limiting tangent plane $\mathbf{V}(z)$. However, choosing another sequence $(y_i) \in \mathbf{V}(y, z)$ approaching the origin from below, the limiting secant line of the sequence $([x_i, y_i])$ is the line $\mathbf{V}(x, y)$ which is not contained in the limiting tangent plane.

This now allows to formally define a Whitney stratification of an algebraic set $X \subset \mathbb{A}^n$:

Definition 2.5.3 (Whitney Stratification)**.** A *Whitney stratification* of an algebraic set $X \subset \mathbb{A}^n$ is a flag

$$\emptyset = W_{-1} \subset W_0 \subset \cdots \subset W_{k-1} \subset W_k = X$$

where each difference $M_i := W_i \setminus W_{i-1}$ is a smooth locally closed set. The connected (in the euclidean topology on $\mathbb{C}^n$) components of each

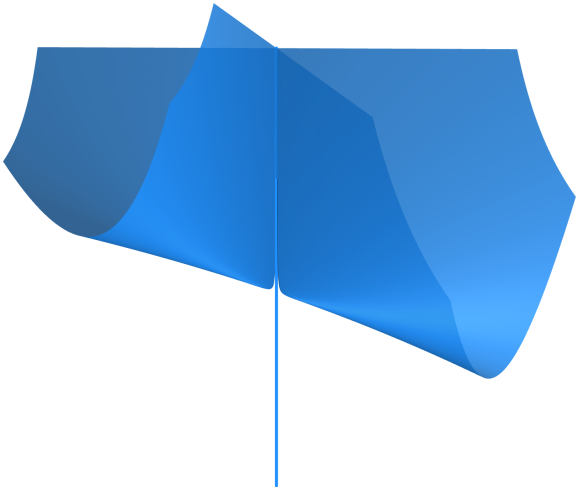


Figure 2.1: The Whitney umbrella $\mathbf{V}(x^2 - y^2 z)$. plotted by the computer algebra system `SageMath`.

M_i are called *strata* and each pair of strata must satisfy Whitney's condition (B). A Whitney stratification is called *minimal* if, after removing any stratum, the resulting flag fails to be a Whitney stratification.

Remark 2.5.1. The existence and uniqueness of a minimal Whitney stratification is established in Teissier (1982).

Example 2.5.2. In Example 2.5.1, Whitney's condition (B) for X and $\mathbf{V}(x, z)$ is only violated at the origin. Correspondingly, a Whitney stratification is given by

$$\mathbf{V}(x, y, z) \subset \mathbf{V}(x, z) \subset X = \mathbf{V}(x^2 - y^2 z)$$

In Helmer and Nanda (2023) an algebraic criterion based on primary decomposition of polynomial ideals is given that allows to check computationally if a given pair $X, Y \subset \mathbb{A}^n$ satisfies condition (B), and as such, to compute a Whitney stratification of a given variety by successively computing singular loci and then checking condition (B) for all pairs of resuling varieties. The contribution in Chapter 8 consists in still applying the same criterion but avoiding the potentially difficult problem of primary decomposition.
In order to introduce the aforementioned algebraic criterion we have to start with

Definition 2.5.4 (Conormal Space)**.** Denote by $(\mathbb{P}^{n-1})^*$ the set of all linear functionals on $\mathbb{C}^n$ modulo scalar multiplication. This is naturally a projective space of dimension $n - 1$. The *conormal space* $\mathrm{Con}(X) \subset \mathbb{A}^n \times (\mathbb{P}^{n-1})^*$ of an algebraic set $X \subset \mathbb{A}^n$ is the Zariski closure of the set

$$\{(p, \zeta) \mid p \in X \setminus \mathrm{Sing}(X) \text{ and } T_p X \subset \zeta\}.$$

The canonical projection $\kappa_X : \mathrm{Con}(X) \to X$ is called the *conormal map*.

Now we can give the aforementioned algebraic criterion. This criterion itself is based on the statement in Remark 4.2 in Flores and Teissier (2017).

Theorem 2.5.1 (Theorem 4.3 in Helmer and Nanda, 2023)**.** *Let X be an algebraic set, let $\emptyset \neq Y \subset \mathrm{Sing}(X)$ be equidimensional and let I_Y be any ideal with $\mathbf{V}(I_Y) = Y$. Let*

$$I_Y + \mathbf{I}(\mathrm{Con}(X)) = \bigcap_{i \in I} Q_i$$

be a primary decomposition. Let

$$J := \{i \in I \mid \dim(\kappa_X(\mathbf{V}(Q_i))) < \dim(Y)\}.$$

Further, let

$$A := \left[\bigcup_{j \in J} \kappa_X(\mathbf{V}(Q_i))\right] \cup \mathrm{Sing}(Y)$$

Then the pair $(X \setminus \mathrm{Sing}(X), Y \setminus A)$ satisfies Whitney's condition (B).

Our algorithm for Whitney stratifications will in general produce a "less minimal" Whitney stratification than the algorithm by Helmer and Nanda (2023). For this reason, we will also give an algorithm to minimize a given Whitney stratification. This algorithm is based on an equivalent characterization of Whitney's condition (B) based on certain local multiplicities, given by Teissier (1982).
We define

Definition 2.5.5 (Local Polar Variety)**.** Let $X \subset \mathbb{A}^n$ be an equidimensional algebraic set of dimension d. For fixed i with $1 \leqslant i \leqslant d-1$, consider a dimension $d+i-1$ linear space in $\mathbb{A}^n$ through a given point $p \in X$ with dual $L_i \subset (\mathbb{P}^{n-1})^*$.
A *codimension i local polar variety through p* is the algebraic set

$$\delta_i(X, p) := \kappa_X(\mathrm{Con}(X) \cap L_i).$$

If our linear space L_i is chosen sufficiently generic, then, using a dimension count, some of the local polar varities $\delta_i(X, p)$ will contain p and some will not (see Remark 3.1 in Flores and Teissier, 2017). We will want to compute the *multiplicity* of points p in local polar varieties $\delta_i(X, p)$. Let us define the notion of multiplicity used here.

Definition 2.5.6 (Length)**.** Let S be any ring and let M be a S-module. The *length* of M is the supremum of the lengths of chains of the form

$$M_0 \subsetneq M_1 \subsetneq \cdots \subsetneq M_n$$

with M_i submodules of M.

Definition 2.5.7 (Hilbert-Samuel Function, Multiplicity)**.** Let S be a local ring with maximal ideal $\mathfrak{m}$ of Krull dimension d. The *Hilbert-Samuel function* of S is defined as

$$\mathrm{HS}(t) := \mathrm{length}(S/\mathfrak{m}^t),\ t \in \mathbb{N}.$$

There exists $N \in \mathbb{N}$ and a polynomial $\mathrm{HP}(t)$ whose leading coefficient is divisible by $d!$ s.t. $\mathrm{HS}(t) = \mathrm{HP}(t)$ for $t \geqslant N$. The *multiplicity* of S is defined as the leading coefficient of $\mathrm{HP}(t)$ divided by $d!$.
Let $X \subset \mathbb{A}^n$ be an equidimensional algebraic set of dimension d and let $p \in \mathbb{A}^n$. The *multiplicity* of X at p, denoted $m_p(X)$, is the multiplicity of the local ring $(\mathbb{C}[\mathbf{x}]/\mathbf{I}(X))_p$.

We refer to Chapter 12 in Eisenbud (1995) for general results about Hilbert-Samuel functions, in particular for the existence of a polynomial $\mathrm{HP}(t)$ as in Definition 2.5.7.
One may now characterize Whitney's condition (B) in terms of the multiplicites of local polar varieties at their defining points as follows, this theorem will be the basis of our minimization algorithm:

Theorem 2.5.2 (see p. 69 and Proposition 3.6 in Flores and Teissier, 2017)**.** *Let X be an equidimensional algebraic set, let $Y \subset X$ be an algebraic subset and let $p \in Y$. Then the sequence*

$$m_\bullet(X,p) := (m_p(X), m_p(\delta_1(X,p)), \ldots, m_p(\delta_{\dim(X)-1}(X,p)))$$

is independent of the linear subspaces chosen to construct the local polar varieties if they are sufficiently general.
In addition, Whitney's condition (B) is satisfied at p for X and Y if and only if the sequence takes the same value for every $q \in X$ in a euclidean neighborhood of p in Y.

3

GRÖBNER BASES OF POLYNOMIAL IDEALS

Again we fix for this chapter a polynomial ring $R := \mathbb{K}[\mathbf{x}]$ over a field $\mathbb{K}$ in a finite set of variables $\mathbf{x} := \{x_1, \ldots, x_n\}$. We denote by $\mathrm{Mon}(\mathbf{x})$ the set of monomials in the variables $\mathbf{x}$. The key computational tool for the algorithms in Part ii is the notion of a *Gröbner basis* of a polynomial ideal $I \subset R$. Gröbner bases can be used to make various ideal-theoretic operations in R computable. We give here a self-contained introduction to their properties and computation.
For a comprehensive treatment of the subject of Gröbner bases we refer to Becker and Weispfenning (1993), containing in particular a chapter on algorithms for computing radicals and primary decompositions of polynomial ideals. The essentials of the subject can again be found in Cox, Little, and O'Shea (2015) and Greuel and Pfister (2007). While we do introduce the properties of Gröbner bases we need, our focus in this chapter lies more on explaining various algorithms for computing Gröbner bases that will be needed in the following chapters.

3.1 BASIC DEFINITIONS & CONCEPTS

The basic idea behind Gröbner bases is to provide a generating set G for a polynomial ideal I for which a multivariate version of polynomial long division yields a *membership test* for I: Given $f \in R$ this polynomial long division w.r.t. G should return zero if and only if $f \in I$. It turns out that this property is not satisfied for *any* generating set of I, this furnishes the need for Gröbner bases.
To perform multivariate polynomial long division we need a way to order the terms of a given polynomial. This ordering is given by a *monomial order*:

Definition 3.1.1 (Monomial Order)**.** A *monomial order* $\prec$ on $\mathrm{Mon}(\mathbf{x})$ is a total order on $\mathrm{Mon}(\mathbf{x})$ which

1. extends the partial order on $\mathrm{Mon}(\mathbf{x})$ given by divisibility and
2. is compatible with multiplication i.e. we have

$$u \prec v \;\Rightarrow\; wu \prec wv \quad \forall u, v, w \in \mathrm{Mon}(\mathbf{x}).$$

When the context is clear, we will just speak of a monomial order without specifying the set of monomials it is defined on. We introduce immediately three important monomial orders:

Definition 3.1.2 (Degree Reverse Lexicographic Order)**.** The *degree reverse lexicographic* order on $\mathrm{Mon}(\mathbf{x})$ is defined as follows for $u, v \in$

$\mathrm{Mon}(\mathbf{x})$: $u \prec_{\mathrm{drl}} v$ iff $\deg u < \deg v$ or $\deg u = \deg v$ and the last nonzero exponent of u/v is positive.

Definition 3.1.3 (Lexicographic Order)**.** The *lexicographic* order on $\mathrm{Mon}(\mathbf{x})$ is defined as follows for $u, v \in \mathrm{Mon}(\mathbf{x})$: $u \prec_{\mathrm{lex}} v$ iff the first nonzero exponent of u/v is negative.

Definition 3.1.4 (Block Order)**.** Let $\mathbf{x}$ and $\mathbf{z}$ be two finite sets of variables. Write each monomial $u \in \mathrm{Mon}(\mathbf{x} \cup \mathbf{z})$ uniquely as a product $u = u_{\mathbf{x}} u_{\mathbf{z}}$ with $u_{\mathbf{x}} \in \mathrm{Mon}(\mathbf{x})$ and $u_{\mathbf{z}} \in \mathrm{Mon}(\mathbf{z})$. Fix a monomial order $\prec_{\mathbf{x}}$ on $\mathrm{Mon}(\mathbf{x})$ and a monomial order $\prec_{\mathbf{z}}$ on $\mathrm{Mon}(\mathbf{z})$. The corresponding *block order eliminating* $\mathbf{x}$ is defined as follows: $u \prec v$ iff $u_{\mathbf{x}} \prec_{\mathbf{x}} v_{\mathbf{x}}$ or $u_{\mathbf{x}} = v_{\mathbf{x}}$ and $u_{\mathbf{z}} \prec_{\mathbf{z}} v_{\mathbf{z}}$ for $u, v \in \mathrm{Mon}(\mathbf{x} \cup \mathbf{z})$.

A monomial order on $\mathbf{x}$ yields a notion of *leading monomial* in R:

Definition 3.1.5 (Leading Monomial, Coefficient and Term)**.** Let $\prec$ be a monomial order on $\mathrm{Mon}(\mathbf{x})$.

(1) For a nonzero element $f \in R$ the *leading monomial* of f w.r.t. $\prec$, denoted $\mathrm{lm}_{\prec}(f)$, is the $\prec$-largest monomial in the support of f. If $f = a\,\mathrm{lm}_{\prec}(f) + g$ for some $a \in \mathbb{K}$ and $g \in R$ with $\mathrm{lm}_{\prec}(g) \prec \mathrm{lm}_{\prec}(f)$, then we define *leading coefficient* of f w.r.t. $\prec$ as $\mathrm{lc}_{\prec}(f) := a$. The *leading term* is $\mathrm{lt}_{\prec}(f) := \mathrm{lc}_{\prec}(f)\,\mathrm{lm}_{\prec}(f)$.

(2) For a finite set F in R we define $\mathrm{lm}_{\prec}(F) := \{\mathrm{lm}_{\prec}(f) \mid f \in F \setminus \{0\}\}$. For an ideal I in R we define the *leading monomial ideal* of I as $\mathrm{lm}_{\prec}(I) := \langle \mathrm{lm}_{\prec}(f) \mid f \in I \setminus \{0\} \rangle$.

Once we have a monomial order we can give a multivariate version of polynomial long division:

Algorithm 1 Multivariate polynomial long division

Input: A finite set G, a monomial order $\prec$, an element $f \in R \setminus \{0\}$
Output: An element r such that $f = r \mod \langle G \rangle$ and such that $\mathrm{lm}_{\prec}(r)$ is not divisible by any $\mathrm{lm}_{\prec}(g)$, $g \in G$.

```
1: procedure REDUCE(f, G, ≺)
2:     r ← f
3:     while r ≠ 0 and there exists g ∈ G and u ∈ Mon(x) with lm_≺(r) = lm_≺(ug)
4:         r ← r − (lc_≺(r) / lc_≺(g)) ug
5:     return r
```

Fixing a monomial order gives *normal forms* for images of elements in quotient rings of R:

Definition 3.1.6 (Staircase, Normal Form)**.** Let I be an ideal in R and let $\prec$ be a monomial order $\mathrm{Mon}(\mathbf{x})$.

1. The set $S_{I,\prec} := \{u \in \mathrm{Mon}(\mathbf{x}) \mid u \notin \mathrm{lm}_\prec(I)\}$ is called the *staircase* of I w.r.t. $\prec$. It naturally forms a $\mathbb{K}$-vector space basis of the quotient ring R/I.

2. The image of every element $f \in R$ in R/I can be uniquely written as a $\mathbb{K}$-linear combination of elements in $S_{I,\prec}$. This linear combination of elements in $S_{I,\prec}$ is called the *normal form* of f w.r.t. I and $\prec$ and denoted $\mathrm{NF}_\prec(f, I)$. The corresponding vector of coefficients of this linear combination, with the elements in $S_{I,\prec}$ ordered by $\prec$, will be denoted $c_{I,\prec}(f)$.

Note that, w.r.t. any monomial order $\prec$, we have $\mathrm{NF}_\prec(f, I) = 0$ iff $f \in I$.
We finally define the notion of Gröbner bases.

Definition 3.1.7 (Gröbner Basis)**.** A *Gröbner basis* of an ideal $I \subset R$ w.r.t. a monomial order $\prec$ is a finite set $G \subset I$ such that $\langle \mathrm{lm}_\prec(G)\rangle = \mathrm{lm}_\prec(I)$. It is called *minimal* if the leading monomial of any $g \in G$ is not divisible by any leading monomial of $G \setminus \{g\}$. It is called *reduced* if no monomial of any $g \in G$ is divisible by any leading monomial of $G \setminus \{g\}$.

For any monomial order $\prec$ and any ideal $I \subset R$, there is a unique reduced $\prec$-Gröbner basis of I.
The desired ideal membership test is now given by

Lemma 3.1.1. *If G is a Gröbner basis for a monomial order $\prec$, then* $\textsc{Reduce}(f, G, \prec) = \mathrm{NF}_\prec(f, \langle G\rangle)$ *for any* $f \in R$.

Proof. Because $\langle \mathrm{lm}_\prec(G)\rangle = \mathrm{lm}_\prec(I)$, the result of $\textsc{Reduce}(f, g, \prec)$ certainly is a linear combination of the elements in $S_{I,\prec}$. This proves the statement. □

One of the most important properties, with numerous applications, of Gröbner basis is the *elimination property*:

Theorem 3.1.1 (Theorem 2 in Chapter 3 of Cox, Little, and O'Shea, 2015)**.** *Let $I \subset \mathbb{K}[\mathbf{x}, \mathbf{z}]$ where $\mathbf{x}$ and $\mathbf{z}$ are two finite sets of variables. Let $\prec$ be a block order eliminating $\mathbf{x}$ and let G be a $\prec$-Gröbner basis of I. Then $G \cap \mathbb{K}[\mathbf{z}]$ is a Gröbner basis of $I \cap \mathbb{K}[\mathbf{z}]$ w.r.t. $\prec$ restricted to* $\mathrm{Mon}(\mathbf{z})$.

For us, we will mainly use this theorem to compute saturation ideals (Definition 2.2.2) via *Rabinowitsch's trick*:

Lemma 3.1.2. *Let $I \subset R$ be an ideal and $f \in R$. Introduce a new variable t and let $J := I + \langle tf - 1\rangle \subset R[t]$. Then*

$$J \cap R = (I : f^\infty)\,.$$

Proof. Let us first show that $IR_f \cap R = (I : f^\infty)$. Suppose $g \in IR_f \cap R$. Then we can write $g = p/f^k$ for suitable $k \in \mathbb{N}$ and $p \in I$. Hence

$gf^k \in I$ and so $g \in (I : f^\infty)$. Suppose on the other hand that $g \in (I : f^\infty)$ so that there exists $k \in \mathbb{N}$ with $gf^k \in I$. Then, in R_f, we have

$$g = \frac{gf^k}{f^k} \in IR_f$$

proving the claim. To prove the lemma, note finally that

$$R_f \cong R[t]/\langle tf - 1\rangle.$$

□

3.2 GRÖBNER BASES OF GENERIC FIBERS AND ALGEBRAIC SET DECOMPOSITION

As an application of Gröbner basis we give here a brief overview of how Gröbner bases of generic fibers (Definition 2.4.1) of polynomial ideals can be used for the purpose of decomposing algebraic sets and polynomial ideals. This serves as a motivation for Chapter 7, in which we will give two algorithms to compute Gröbner bases of generic fibers and for Chapter 8, where we will present a similar strategy as demonstrated here to compute so-called *Whitney stratifications* of algebraic sets. We refer to Decker, Greuel, and Pfister (1999) for more details on the contents of this section.

3.2.1 *Equidimensional Decomposition*

Let $I \subset R$ be a polynomial ideal and suppose that we want to compute an *equidimensional decomposition* of $\mathbf{V}(I)$, i.e. we want to compute a finite set of ideals $\mathcal{D}$ such that for each $J \in \mathcal{D}$, $\mathbf{V}(J)$ is equidimensional (see Definition 2.1.3) and such that

$$\mathbf{V}(I) = \bigcup_{J \in \mathcal{D}} \mathbf{V}(J).$$

Suppose now that $\mathbf{y} \subset \mathbf{x}$ is a MIS of I. If we then define

$$J := \mathrm{gen}(I, \mathbf{y}) \cap R$$

then, by Proposition 2.4.2, the irreducible components of J are some of the irreducible components of I of dimension $\dim(I)$, in particular $\mathbf{V}(J)$ is equidimensional.
Let us now discuss how to compute such ideals J. Classically, the computation is based on the following propositions:

Proposition 3.2.1. *(1) Let $M \subset R$ be a finite set of monomials. Then a subset $\mathbf{y} \subset \mathbf{x}$ is a MIS of $\langle M\rangle$ if and only if there is no monomial in M that depends only on the variables in $\mathbf{y}$ and $\mathbf{y}$ is maximal with this property.*

(2) *For any monomial order* $\prec$ *and for any ideal* $I \subset R$, *any MIS of* $\mathrm{lm}_\prec(I)$ *is a MIS of* I.

Proof. Proof of (1): Suppose $\mathbf{y} \subset \mathbf{x}$ is a MIS of $\langle M \rangle$ so that $\langle M \rangle \cap \mathbb{K}[\mathbf{y}] = 0$. If there were a monomial in $\mathbf{y}$ contained in $\langle M \rangle$ then clearly $\langle M \rangle \cap \mathbb{K}[\mathbf{y}] \neq 0$, a contradiction. On the other hand, suppose that $\mathbf{y}$ is a subset of $\mathbf{x}$ such that no monomial in M depends only on $\mathbf{y}$. Note that M is a Gröbner basis for $\langle M \rangle$ w.r.t. any monomial order. If we now assume there exists $f \neq 0$ with $f \in \langle M \rangle \cap \mathbb{K}[\mathbf{y}]$ then one of the elements of M must divide the leading monomial of f and therefore lie in $\mathbb{K}[\mathbf{y}]$, a contradiction.
Proof of (2): If $\mathbf{y}$ is a MIS of $\mathrm{lm}_\prec(I)$ and we have $f \in I \cap \mathbb{K}[\mathbf{y}]$ with $f \neq 0$ then there must exist $u \in \mathrm{lm}_\prec(I)$ which divides $\mathrm{lm}_\prec(f)$. Therefore $u \in \mathbb{K}[\mathbf{y}]$, a contradiction to the fact that $\mathbf{y}$ is a MIS of $\mathrm{lm}_\prec(I)$. This shows that $I \cap \mathbb{K}[\mathbf{y}] = 0$. Let us now show that $\mathbf{y}$ is maximal with this property. Because $\mathbf{y}$ is a MIS of $\mathrm{lm}_\prec(I)$, for any $x \in \mathbf{x} \setminus \mathbf{y}$ there exists a monomial $u \in \mathrm{Mon}(\mathbf{y})$ and $k \in \mathbb{N}$ s.t. $x^k u \in \mathrm{lm}_\prec(I)$. In particular $x^k \in \mathrm{lm}_\prec(\mathrm{gen}(I, \mathbf{y}))$, so that $\mathrm{gen}(I, \mathbf{y})$ is of dimension zero. Hence there exists a polynomial

$$f \in \mathrm{gen}(I, \mathbf{y}) \cap \mathbb{K}(\mathbf{y})[x].$$

Multiplying f by a suitable element in $\mathbb{K}[\mathbf{y}]$, we find an element $p \in I \cap \mathbb{K}[\mathbf{y}, x]$, showing the maximality of $\mathbf{y}$. □

This proposition thus gives us the ability to compute MIS of polynomial ideals I: We compute a Gröbner basis of I, w.r.t. some monomial order $\prec$, and then extract a MIS of $\mathrm{lm}_\prec(I)$ by applying Proposition 3.2.1. Now we recall

Proposition 3.2.2 (Lemma 4 in Decker, Greuel, and Pfister, 1999)**.** *Let* I *be an ideal in a polynomial ring* $\mathbb{K}[\mathbf{x}, \mathbf{z}]$. *Let* $\prec$ *be a block order eliminating* $\mathbf{x}$ *and let* G *be a Gröbner basis of* I *w.r.t.* $\prec$. *Then* G *is also a Gröbner basis of* $I\mathbb{K}(\mathbf{z})[\mathbf{x}]$.

This proposition gives us one way to compute Gröbner bases of generic fibers. Next, once we have a Gröbner basis of a generic fiber $\mathrm{gen}(I, \mathbf{y})$, we have to show how to compute generators for the ideal $\mathrm{gen}(I, \mathbf{y}) \cap R$. For this we have

Proposition 3.2.3. *Let* $\mathbf{y} \subset \mathbf{x}$ *and let* $G \subset R$ *be any Gröbner basis for a monomial order* $\prec$ *on* $\mathrm{Mon}(\mathbf{x} \setminus \mathbf{y})$ *of an ideal* $K \subset \mathbb{K}(\mathbf{y})[\mathbf{x} \setminus \mathbf{y}]$. *Let*

$$h := \mathrm{lcm}\{\mathrm{lc}_\prec(g) \mid g \in G\}.$$

Then

$$K \cap R = (\langle G \rangle : h^\infty).$$

Now we are ready to give an equidimensional decomposition algorithm, see Algorithm 5 in Decker, Greuel, and Pfister (1999) for proofs of correctness and termination:

Algorithm 2 Equidimensional Decomposition Using Generic Fibers

Input: A finite sequence of polynomials $F \subset R$
Output: A set of Gröbner bases s.t. the corresponding ideals define an equidimensional decomposition of $\mathbf{V}(F)$

```
1: procedure GenFibEqui(F)
2:     Determine a MIS y of ⟨F⟩    Proposition 3.2.1
3:     ≺ ← any monomial order
4:     G ← a ≺-Gröbner basis in R of gen(⟨F⟩, y)    Chapter 7
5:     h ← lcm{lc_≺(g) | g ∈ G}
6:     H ← a Gröbner basis of (⟨F⟩ : h^∞)    Lemma 3.1.2
7:     if reduce(h, G, ≺) = 0 for all h ∈ H
8:         return {H}
9:     else
10:        return {H} ∪ GenFibEqui(F ∪ {h})
```

3.2.2 *Irreducible Decomposition*

Similar ideas as in the last subsection can be used to find the irreducible components of $\mathbf{V}(I)$ for a given polynomial ideal I, we informally sketch this method here. A similar strategy works to compute a primary decomposition of I, we refer again to Sections 8.6 and 8.7 in Becker and Weispfenning (1993) for further details.
For now let us assume that I is zero-dimensional, i.e. that $\mathbf{V}(I)$ is a finite set. Now let $\mathbf{x} := \{x_1, \ldots, x_n\}$. We define

Definition 3.2.1 (Shape position)**.** Let g_I be the unique, monic generator of $I \cap \mathbb{K}[x_n]$. The ideal I is said to be in *shape position* if there exist $g_1, \ldots, g_{n-1} \in \mathbb{K}[x_n]$ s.t.

$$I = \langle g_I, x_1 - g_1, \ldots, x_{n-1} - g_{n-1} \rangle.$$

Note that the polynomial g_I defined as above can be computed using any block order eliminating $x_1, \ldots, x_{n-1}$. Once this polynomial is computed we have

Proposition 3.2.4 (Proposition 8.69 in Becker and Weispfenning, 1993)**.** *Suppose I is zero-dimensional and $\sqrt{I}$ is in shape position. Let g_I be as in Definition 3.2.1. Suppose that g_I has $\mathbb{K}$-irreducible factorization*

$$g_I = \prod_{i=1}^{s} f_i^{k_i}$$

with the f_i pairwise distinct. Then an irreducible decomposition of $\mathbf{V}(I)$ is given by

$$\mathbf{V}(I) = \bigcup_{i=1}^{s} \mathbf{V}(I + \langle f_i \rangle).$$

Supposing that we can factor polynomials over $\mathbb{K}$ into irreducible factors, this proposition immediately furnishes an algorithm to compute the irreducible components of $\mathbf{V}(I)$ if $\sqrt{I}$ is in shape position. The question remains what to do if $\sqrt{I}$ is not in shape position. Probabilistically, we can put any zero-dimensional radical ideal in shape position by introducing an extra variable:

Proposition 3.2.5 (Lemma 8.72 in Becker and Weispfenning, 1993)**.** *Let I be zero-dimensional. There exists a Zariski-open subset U of $\mathbb{A}^n$ such that for any $\mathbf{c} := (c_1, \ldots, c_n) \in U$ the radical of the ideal*

$$I_{\mathbf{c}} := I + \langle z - c_1 x_1 - c_2 x_2 - \cdots - c_n x_n \rangle \subset \mathbb{K}[\mathbf{x}, z]$$

is in shape position.

Supposing that we have a $\mathbf{c}$ as in this proposition we can compute the irreducible components of $\mathbf{V}(I_{\mathbf{c}})$ and compute their projection to the original affine space using a block order eliminating z.
Hence we have the ability to probabilistically compute irreducible decompositions of arbitrary zero-dimensional ideals. Note that it can be checked whether $I_{\mathbf{c}}$ is in shape position, thus the resulting algorithm is Las Vegas, see again Sections 8.6 and 8.7 in Becker and Weispfenning (1993) for further details. Finally, again using Proposition 2.4.2, we can compute irreducible decompositions of arbitrary algebraic sets by applying zero-dimensional irreducible decomposition to their generic fibers.

3.3 BUCHBERGER'S ALGORITHM

The obvious question is now how to compute a Gröbner basis for a given polynomial ideal $\langle f_1, \ldots, f_r \rangle \subset R$ w.r.t. some monomial order $\prec$. Historically, the first algorithm to compute Gröbner bases is given by Buchberger's algorithm, see Buchberger (1965). The closely related concept of *standard bases* was developed by Hironaka (1964). The algorithm by Buchberger is based on *Buchberger's criterion* which gives a criterion for a given set $G \subset R$ to be a Gröbner basis by checking the result of $\textsc{Reduce}(\bullet, G, \prec)$ for finitely many polynomials, called *S-pairs*:

Definition 3.3.1 (S-Pair)**.** Let $f, g \in R$ be two polynomials and $\prec$ be a monomial order. Let $u = \mathrm{lm}_\prec(f), v = \mathrm{lm}_\prec(g)$ The *S-pair* between f, g (w.r.t. $\prec$) is given by

$$\mathrm{sp}_\prec(f, g) = \frac{\mathrm{lcm}(u, v)}{\mathrm{lt}_\prec(f)} f - \frac{\mathrm{lcm}(u, v)}{\mathrm{lt}_\prec(g)} g.$$

Now we have:

Theorem 3.3.1 (Buchberger's Criterion, e.g. Theorem 5.48 in Becker and Weispfenning, 1993)**.** *Let $G \subset R$ be a finite set and $\prec$ be a monomial order. Then G is a $\prec$-Gröbner basis if and only if*

$$\textsc{Reduce}(\mathrm{sp}_\prec(f, g), G, \prec) = 0$$

for all $f, g \in G$ *with* $f \neq g$.

This immediately furnishes

Algorithm 3 Buchberger's Algorithm

Input: A finite set $\{f_1, \ldots, f_r\} \subset R$, a monomial order $\prec$
Output: A $\prec$-Gröbner basis of $\langle F \rangle$

1: $G \leftarrow \{f_1, \ldots, f_r\}$
2: $P \leftarrow \left\{ sp_\prec(f_i, f_j) \mid 1 \leqslant i < j \leqslant r \right\}$
3: **while** $P \neq \emptyset$
4: Select some $p \in P$, $P \leftarrow P \setminus \{p\}$
5: $r \leftarrow \textsc{Reduce}(f, G, \prec)$
6: **if** $r \neq 0$
7: $P \leftarrow P \cup \left\{ sp_\prec(p, g) \mid g \in G \right\}$
8: $G \leftarrow G \cup \{p\}$
9: **return** G

Note that termination of this algorithm follows from the Noetherianity of R: After Line 8, the ideal $\langle lm_\prec(G) \rangle$ has gotten larger. The correctness follows from Buchberger's criterion, Theorem 3.3.1.
There are two practical questions that this algorithm yields: First, which S-pair should be selected in Line 4? Second, any S-pair which reduces to zero in Line 5 does not contribute to the output. How can such S-pairs be detected in advance? In the next section, Section 3.4, we present the F_4 algorithm, given by Faugère (1999), which circumvents the selection issue by selecting a large number of S-pairs at once. In Section 3.6.1 we introduce *signature-based Gröbner basis algorithms*, initialized by the F_5 algorithm given by Faugère (2002), which introduce criteria to detect S-pairs reducing to zero in Buchberger's algorithm in advance, i.e. before reduction is performed.

3.4 THE F_4 ALGORITHM

The F_4 algorithm given by Faugère (2002) builds upon the framework of Buchberger's algorithm. The modification made to Buchberger's algorithm is the following: Instead of selecting and reducing one S-pair at a time we select several S-pairs at once and organize them, together with all of their reducers in a matrix. The rows of this matrix will correspond to certain polynomials. This matrix is then echelonized and the rows whose leftmost entry has changed during the echelonization then correspond to polynomials which need to be added to our eventual output. In Section 7.4 we will show how to adapt the F_4 algorithm to compute Gröbner bases of generic fibers.
Let us start by defining the kind of matrix we will be building in the F_4 algorithm:

Definition 3.4.1 (Macaulay Matrix)**.** For any $f \in R$ denote by $\operatorname{supp}(f) \subset \operatorname{Mon}(\mathbf{x})$ the support of f. Now, let $F = (f_1, \ldots, f_r)$ be a finite set of polynomials in R and let $\prec$ be a monomial order. The *Macaulay matrix* $M_{F,\prec}$ determined by F and $\prec$ is the matrix with the ith row indexed by f_i and the columns indexed in descending order (w.r.t. $\prec$) by the monomials $\bigcup_{i=1}^{r} \operatorname{supp}(f_i)$. The entry in the row indexed by f_i and the column indexed by $u \in \operatorname{Mon}(\mathbf{x})$ is $c \in \mathbb{K}$ where we write uniquely $f = cu + g$ with $u \notin \operatorname{supp}(g)$.

For a nonzero row r of a matrix M, let us refer to its leftmost nonzero entry by $\operatorname{pivot}(r)$. Now we can give the F_4 algorithm and elaborate on its various parts:

Algorithm 4 The F_4 Algorithm

Input: A finite set $\{f_1, \ldots, f_r\} \subset R$, a monomial order $\prec$
Output: A $\prec$-Gröbner basis of $\langle F \rangle$

```
1:  G ← {f_1, ..., f_r}
2:  P ← {(uf_i, vf_j) | i < j, sp_≺(f_i, f_j) = uf_i − vf_j}
3:  while P ≠ ∅
4:      S ← Select(P)
5:      S ← SymbolicPreprocessing(S, G, ≺)
6:      M ← M_{S,≺}
7:      M' ← Echelonize(M)
8:      for every row r' of M'
9:          r ← the corresponding row of M
10:         if r' ≠ 0 and pivot(r') ≠ pivot(r)
11:             p ← the polynomial corresponding to r'
12:             P ← P ∪ {(up, vg) | g ∈ G, sp_≺(p, g) = up − vg}
13:             G ← G ∪ {p}
14: return G
```

Compared to Buchberger's algorithm we store the two constitual components of each S-pair in the set P instead of storing the S-pairs directly. In Line 4, we then select several (i.e. more than one) pairs out of P. For each selected pair (p, q) out of P, we store p and q in the set S and remove (p, q) from P. We have freedom in which S-pairs we want to select, a typical choice is to select all S-pairs of minimal degree in P.
Then, in Line 5, we look for a reducer in G for each monomial appearing in one of the elements in S with the exception of the leading monomials: By the definition of S-pairs each leading monomial already appears at least twice in S, so they will reduce during the echelonization step. In pseudocode this looks as follows:

Algorithm 5 Symbolic Preprocessing for the F_4 Algorithm

Input: A finite set of polynomials S, a finite set of polynomials G, a monomial order $\prec$
Output: A finite set of polynomials T

```
1: procedure SymbolicPreprocessing(S, G, ≺)
2:     U ← ⋃_{s∈S} supp(s) \ {lm_≺(s)}
3:     T ← S
4:     while U ≠ ∅
5:         u ← some element in U, U ← U \ {u}
6:         if there exists g ∈ G, v ∈ Mon(x) with lm_≺(vg) = u
7:             T ← T ∪ {vg}
8:             U ← U ∪ supp(vg) \ lm_≺(vg)
9:     return T
```

Finally, in line Line 7, we echelonize the Macaulay matrix built out of the set S. Here, we are not allowed to swap columns, they need to be kept ordered by $\prec$. This echelonization has the effect of reducing all previously selected S-pairs in one single linear algebra computation. This is the key point where F_4 gains efficiency over Buchberger's algorithm also because the Macaulay matrices built by F_4 have a particular structure that can be utilized for efficient echelonization. This is done, for example, by the Gröbner basis library `msolve` (Berthomieu, Eder, and Safey El Din, 2021).
For proofs of correctness and termination of Algorithm 4 and Algorithm 5 we refer again to Faugère (1999).

3.4.1 *Gröbner Tracers*

Gröbner tracers were designed in Traverso (1989) for the purpose of efficient multi-modular Gröbner basis computation over the rational numbers. While originally designed to be used in conjunction with Buchberger's algorithm (Algorithm 3), we present them here briefly and somewhat informally as an extension of the F_4 algorithm. This is because in Section 7.4 we will adapt a similar idea to a modular lifting strategy combined with F_4 to compute Gröbner bases of generic fibers. These Gröbner tracers, as presented in Traverso (1989), can also be easily combined with *signature-based* Gröbner basis algorithms, see Section 3.6 below for an introduction to these algorithms. The basic observation of multi-modular Gröbner basis computation over the rational numbers is the following

Theorem 3.4.1 (e.g. Arnold, 2003; Ebert, 1983; Pauer, 1992; Traverso, 1989)**.** *Let A be an integral domain with field of fractions Q and let $F \subset A[\mathbf{x}]$ be a finite sequence of polynomials. Let G be a $\prec$-Gröbner basis of $\langle F \rangle$ in $Q[\mathbf{x}]$. There exists an ideal $J \subset A$ s.t. for all prime ideals $P \subset A$ with $J \not\subset P$ (called*

henceforth a good prime for $\langle F\rangle$ *and* $\prec$*) we have* $G \subset A_P[\mathbf{x}]$ *and* $G \mod P$ *is a* $\prec$*-Gröbner basis of the ideal* $\langle F\rangle A_P/P_P[\mathbf{x}]$.

Note that if $A = \mathbb{Z}$, then the prime ideals P which do contain a given ideal $J \subset A$ as in Theorem 3.4.1 constitute a finite set.
Moreover, based on this theorem, if we run the F_4 algorithm on an input system $F \subset \mathbb{Q}[\mathbf{x}]$ and get output G then, for all primes p except finitely many, all coefficients appearing in this computation will lie in $\mathbb{Z}_p$, i.e. the localization of $\mathbb{Z}$ at $\langle p\rangle$. Thus we can repeat the exact same computational steps over the finite field $\mathbb{F}_p := \mathbb{Z}/p\mathbb{Z}$ to obtain the Gröbner basis $G \mod p$. Since this works for almost every prime p, we may now do the following:

(1) Choose a random prime p and run F_4 on the system $F \mod p$.

(2) "Remember" the computational steps we performed except the ones that yielded reductions to zero.

(3) Apply the exact same computational steps modulo other randomly chosen primes $p_1, p_2, \ldots$.

(4) Recombine the resulting Gröbner bases using the Chinese Remainder Theorem.

Remembering the computational steps of the F_4 algorithm in Item (2) for a given input can now be done using a *tracer*:

Definition 3.4.2 (Tracer)**.** A *tracer* T for a system of polynomials $f_1, \ldots, f_r$ consists of the following data:

(1) A monomial order $\prec$.

(2) A finite sequence of finite sets of integers $I_1, \ldots, I_s$

(3) For each I_i and each $j \in I_i$ a finite set of monomials U_{ij}.

With this data structure, Item (3) above now takes the following shape:

Algorithm 6 Following a Tracer

Input: A finite sequence of polynomials $F \subset R$, a tracer T for F
Output: A finite sequence of polynomials $G \subset R$

```
1: G ← F
2: ≺ ← the monomial order underlying T
3: for I_i the integer sequences of T
4:     R ← ⋃_{j∈I_i} {uG[j] | u ∈ U_ij, U_ij as in Definition 3.4.2}
5:     M ← the Macaulay matrix M_{R,≺}
6:     M′ ← echelonization of M
7:     Append all rows of M′ with changed leading terms to G
8: return G
```

On a computer, we would store the set G in Algorithm 6 as an array. In Line 4 by $G[j]$ we mean the jth element in such an array.

Definition 3.4.3 (Good Tracer)**.** A tracer T for $F := \{f_1, \ldots, f_r\}$ and $\prec$ is called *good* if running Algorithm 6 is well-defined (i.e. there are no out-of-bounds accesses in Line 4) and if its output G is a $\prec$-Gröbner basis of $\langle F \rangle$.

As detailed above, by Theorem 3.4.1, a tracer T extracted out of an F_4 run modulo a random prime p will be good when applied modulo almost all other primes q. During a specific computation, assuming that p is a good prime, q will be not a good prime if applying T yields a different leading monomial ideal modulo q than the one obtained modulo p.

3.5 THE FGLM ALGORITHM

For practical purposes, Gröbner bases for the lexicographic (LEX) order (Definition 3.1.3, can be used to solve polynomial systems via Definition 3.2.1) or block orders (Definition 3.1.4, when we want to eliminate variables via Theorem 3.1.1) are frequently of interest. It has been observed, however, that Buchberger's algorithm and F_4 can be badly behaved when giving such an order as input and are, in general, much better behaved in practice when using the degree reverse lexicographic (DRL) order (Definition 3.1.2).
This motivates the design of "change-of-order" algorithms: Given a Gröbner basis of a polynomial ideal $I \subset R$ w.r.t. one ordering $\prec_{in}$ (e.g. DRL), convert it to a Gröbner basis w.r.t. another ordering (e.g. LEX). If I is zero-dimensional then the FGLM algorithm, originally given in Faugère, Gianni, Lazard, and Mora (1993), and in particular its optimized variants under certain genericity assumptions on the ideal I (see Berthomieu, Neiger, and Safey El Din, 2022; Faugère, Gaudry, Huot, and Renault, 2014; Faugère and Mou, 2017; Neiger and Schost, 2020) have turned out to be very practical for example as part of the software library msolve (Berthomieu, Eder, and Safey El Din, 2021) where these optimized variants of FGLM form a critical component of the polynomial system solving machinery provided by msolve. Besides the FGLM algorithm we mention the *Gröbner walk* algorithm which performes a change-of-order even when I is positive dimensional, see Collart, Kalkbrener, and Mall (1997).
We describe here a simplified version of the original FGLM algorithm. In Chapter 7, we adapt this algorithm, in a certain sense, to the positive-dimensional setting by computing Gröbner bases of generic fibers of polynomial ideals, again via a modular lifting strategy.
Recall that an ideal $I \subset R$ is zero-dimensional if and only if R/I is finite-dimensional as a $\mathbb{K}$-vector space. If this is the case, then we can use normal form computations to perform finite-dimensional linear algebra in R/I. This lies at the core of the FGLM algorithm which, given a Gröbner basis of a zero-dimensional ideal I w.r.t. a monomial order $\prec_{in}$, obtains a Gröbner basis of I w.r.t. another monomial order

$\prec_{out}$. The basic idea is that each normal form of an element in R can be thought of as an element in the finite-dimensional vector space R/I. A linear dependence between normal forms then corresponds to an element in I.

Algorithm 7 The FGLM Algorithm

Input: A Gröbner basis H w.r.t. a monomial order $\prec_{in}$, another monomial order $\prec_{out}$

Output: The reduced Gröbner basis G of $\langle H\rangle$ w.r.t. $\prec_{out}$, the $\prec_{out}$-staircase of $\langle H\rangle$

1: **procedure** FGLM(H, $\prec_{out}$)
2: $L \leftarrow \emptyset$, $C \leftarrow \emptyset$, $S \leftarrow \emptyset$, $G \leftarrow \emptyset$
3: **while** there exists a monomial not in $S \cup \langle L\rangle$
4: $u \leftarrow$ the $\prec_{out}$-minimal such monomial
5: $c_u \leftarrow c_{\langle H\rangle,\prec_{in}}(u)$ *computed via* REDUCE$(u, H, \prec_{in})$
6: **if** there exist $\alpha_v \in \mathbb{K}$ for each $v \in S$ s.t. $c_u := \sum_{c_v \in C} \alpha_v v$
7: $L \leftarrow L \cup \{u\}$
8: $G \leftarrow G \cup \{u - \sum_{v\in S} \alpha_v v\}$
9: **else**
10: $C \leftarrow C \cup \{c_u\}$
11: $S \leftarrow S \cup \{u\}$
12: **return** G, S

Remark 3.5.1. The FGLM algorithm, as presented in Faugère, Gianni, Lazard, and Mora (1993), has arithmetic complexity $O\left(nD^3\right)$, where D is the degree of the input ideal I, i.e. the $\mathbb{K}$-dimension of R/I. This complexity is achieved by not computing the required normal forms naively in Line 5 but by using *multiplication tensors* associated to I and $\prec_{in}$ instead. In Section 7.3 we will present a similar complexity analysis for our modular lifting version of FGLM, so we forego it here.

3.6 SIGNATURE-BASED GRÖBNER BASIS COMPUTATIONS

Let $f_1, \ldots, f_r \in R$ be a sequence in R. Just as in the previous subsections, our goal here is again to compute a Gröbner basis for the ideal generated by $f_1, \ldots, f_r$ for a given monomial order $\prec$ on R. The class of algorithms we introduce here are called *signature-based* Gröbner basis algorithms and modify the basic structure of Buchberger's algorithm in so far as they attach an eponymous *signature* to each polynomial to be reduced in the course of Buchberger's algorithm with the goal of using this data to detect S-pairs that will reduce to zero and do thus not contribute to the output. These signatures lead to certain structural properties of signature-based Gröbner basis algorithms which we will use in Chapter 4 and Chapter 6.

The field of signature-based Gröbner basis algorithms was started with the F_5 algorithm (Faugère, 2002) and extended in many different

ways. Our exposition in this chapter follows closely the survey paper given by Eder and Faugère (2017) in which various signature-based algorithms are introduced and compared via a common framework. To introduce the ideas behind signature-based algorithms let us start with a description of *Lazard's algorithm* given by Lazard (1983). Suppose that $f_1, \dots, f_r$ are homogeneous.

Definition 3.6.1 (Truncated Gröbner Basis)**.** For $d \in \mathbb{N}$ and a monomial order $\prec$ a set G is a d-*truncated Gröbner basis of* $I := \langle f_1, \dots, f_r \rangle$ if for any $u \in \mathrm{lm}_\prec(I)$ with $\deg(u) \leqslant d$ there exists $g \in G$ with $\mathrm{lm}_\prec(g) \mid u$.

Lazard's algorithm now computes a d-truncated Gröbner basis of $\langle f_1, \dots, f_r \rangle$ by echelonizing a series of suitable Macaulay matrices. To this end define

Definition 3.6.2 (Macaulay Matrix in Degree d)**.** The *Macaulay matrix* M_d in degree d of homogeneous $f_1, \dots, f_r$ w.r.t. a monomial is the Macaulay matrix $M_{F_d, \prec}$ where

$$F_d := \bigcup_{i=1}^{r} \{u f_i \mid u \in \mathrm{Mon}(\mathbf{x}), \deg(u f_i) = d\}.$$

Lazard (1983) now shows that a d-truncated Gröbner basis (w.r.t. some monomial order) of $\langle f_1, \dots, f_r \rangle$ can be computed by echelonizing each Macaulay matrix $M_1, \dots, M_d$. The polynomials corresponding to the rows of these matrices then form the desired d-truncated Gröbner basis.

The basic idea of signature-based Gröbner basis computations is now best explained in the context of Lazard's algorithm. Let us illustrate this idea with the following example, coming from Eder and Faugère (2017): In $\mathbb{F}_5[x, y, z]$, let

$$\begin{aligned} f_1 &= y^2 + 4yz \\ f_2 &= 2x^2 + 3xy + 3z^2 \\ f_3 &= 3x^2 + 4xy + 2y^2 \end{aligned}$$

and suppose we want to compute a (truncated) Gröbner basis of the ideal $I := \langle f_1, f_2, f_3 \rangle$ w.r.t. the DRL order using Lazard's algorithm. We start by constructing the Macaulay matrix in degree 2:

$$M_2 = \begin{array}{c} \begin{matrix} x^2 & xy & y^2 & xz & yz & z^2 \end{matrix} \\ \begin{bmatrix} 3 & 4 & 2 & 0 & 0 & 0 \\ 2 & 3 & 4 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 4 & 0 \end{bmatrix} \end{array} \begin{matrix} f_3 \\ f_2 \\ f_1 \end{matrix}$$

After echolonisation this matrix takes the shape

$$\begin{array}{c} \begin{matrix} x^2 & xy & y^2 & xz & yz & z^2 \end{matrix} \\ \begin{bmatrix} 0 & 2 & 0 & 1 & 3 & 3 \\ 2 & 3 & 0 & 0 & 4 & 3 \\ 0 & 0 & 1 & 0 & 4 & 0 \end{bmatrix} \end{array} \begin{matrix} p_3 \\ p_2 \\ f_1 \end{matrix} \begin{matrix} \leadsto f_5 = f_3 + f_2 + 4f_1 \\ \leadsto f_4 = f_2 + f_1 \\ \\ \end{matrix}$$

Hence, a degree 2 truncated Gröbner basis of I is given by $\{f_1, f_4, f_5\}$. Now we make the key modification: Throughout our computation we "remember" where each new polynomial came from by remembering the row label of the row from which it was obtained. Such a row label is of the form uf_i, $i = 1, 2, 3$ for a monomial u.
Let $\epsilon_1, \epsilon_2, \epsilon_3$ be the standard basis of unit vectors of R^3. We have a map $R^3 \to R$, mapping ϵ_i to f_i. Now we label each element obtained so far by the row label of the row from which it came, replacing f_i by ϵ_i and store this data in our degree 2 truncated Gröbner basis. Such a label, of the form $u\epsilon_i$, $i = 1, 2, 3$, for a monomial u is called a *signature*. With this modification, our degree 2 truncated Gröbner basis now takes the shape

$$G_2 := \{(\epsilon_1, f_1), (\epsilon_2, f_4), (\epsilon_3, f_5)\}.$$

To extend G_2 into a degree 3 truncated Gröbner basis we would now build the Macaulay matrix M_3 in degree three, whose rows are given by xf_i, yf_i, zf_i, $i = 1, 2, 3$, or, in our new terminology, whose rows have signatures $x\epsilon_i, y\epsilon_i, z\epsilon_i$, $i = 1, 2, 3$. The idea behind remembering these signatures is now the following:

Throughout our computation, if we disallow certain row operations during echelonization, then the result of reducing the row with signature $u\epsilon_i$ depends only on this signature.

In our example, this means that before doing any reduction, we can replace the rows in M_3 corresponding to xf_2, yf_2, zf_2 with xf_4, yf_4, zf_4 because p_2 and p_4 have the same signature, and similarly for f_5. This has the effect of putting M_3 into a "more reduced" shape before any arithmetic has been performed. Indeed, the Macaulay matrix M_3 with all rows coming from f_1, f_2, f_3 is given by

$$M_3 = \begin{array}{c} \begin{matrix} x^3 & x^2y & xy^2 & y^3 & x^2z & xyz & y^2z & xz^2 & yz^2 & z^3 \end{matrix} \\ \begin{bmatrix} 3 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 4 & 2 & 0 & 0 & 0 \\ 2 & 3 & 4 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 2 & 3 & 4 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 & 3 & 4 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 4 & 0 \end{bmatrix} \end{array} \begin{matrix} x\epsilon_3 \\ y\epsilon_3 \\ z\epsilon_3 \\ x\epsilon_2 \\ y\epsilon_2 \\ z\epsilon_2 \\ x\epsilon_1 \\ y\epsilon_1 \\ z\epsilon_1 \end{matrix}$$

After the replacement rule above, replacing xf_2, yf_2, zf_2 by xf_4, yf_4, zf_4, and similarly for f_3 and f_5, we get

$$\begin{array}{c} \begin{matrix} x^3 & x^2y & xy^2 & y^3 & x^2z & xyz & y^2z & xz^2 & yz^2 & z^3 \end{matrix} \\ \begin{bmatrix} 0 & 2 & 0 & 0 & 0 & 1 & 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 3 \\ 2 & 3 & 0 & 0 & 0 & 4 & 0 & 3 & 0 & 0 \\ 0 & 2 & 3 & 0 & 0 & 0 & 4 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 & 3 & 0 & 0 & 4 & 3 \\ 0 & 0 & 1 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 4 & 0 \end{bmatrix} \begin{matrix} xe_3 \\ ye_3 \\ ze_3 \\ xe_2 \\ ye_2 \\ ze_2 \\ xe_1 \\ ye_1 \\ ze_1 \end{matrix} \end{array}$$

which is already closer to having echelon form than M_3 above!
At the beginning of this section we mentioned that the signatures also serve to avoid reductions to zero during Gröbner basis computations. This also is an effect of the replacement rule above. Let us denote by $\mathrm{RR}(ue_i, f)$ the polynomial obtained from reducing the row corresponding to the polynomial f with signature ue_i. Suppose that previously some row g with signature ve_i has reduced to zero and that $u = wv$ for some monomial w. Then, by the replacement rule above

$$\mathrm{RR}(ue_i, f) = \mathrm{RR}(ue_i, wg) = \mathrm{RR}(ue_i, \mathrm{RR}(ve_i, g)) = \mathrm{RR}(ue_i, 0) = 0,$$

so any row whose signature is divisible by the signature of a row that has previously reduced to zero can be avoided!

3.6.1 *Basic Concepts & Definitions*

Let us now give the appropriate technical framework to the idea of signatures: Let R^r be the free module over R of rank r with standard basis $e_1, \ldots, e_r$. We define a map $R^r \to R$ by R-linear extension of

$$e_i \mapsto \overline{e_i} := f_i.$$

Fix a monomial order $\prec$ on $\mathrm{Mon}(\mathbf{x})$. Abusing notation, we extend it to a *module monomial order* $\prec$ on R^r:

Definition 3.6.3 (Module Monomial Order)**.** A *module monomial* is a term of the form ue_i where $u \in \mathrm{Mon}(\mathbf{x})$. A *module monomial order* extending a monomial order $\prec$ on R is defined analogously to a monomial order on $\mathrm{Mon}(\mathbf{x})$ with the additional constraint that $u \prec v$ implies $ue_i \prec ve_i$ for all $u, v \in \mathrm{Mon}(\mathbf{x})$ and $1 \leqslant i \leqslant r$.

Whether we mean a monomial order or a module monomial order will be clear from the context.
In the context of signature-based algorithms, the typical choices for module monomial orders are

Definition 3.6.4 (Position Over Term, Term Over Position, Schreyer Order)**.** Let $\prec$ be a monomial order on R and let $e_1, \dots, e_r$ be the standard basis of unit vectors of R^r.

(1) The *position-over-term* (POT) order on R^r is defined as $ue_i \prec_{\text{pot}} ve_j$ iff $i < j$ or $i = j$ and $u \prec v$.

(2) The *term-over-position* (TOP) order on R^r is defined as $ue_i \prec ve_j$ iff $u \prec v$ or $u = v$ and $i < j$.

(3) The *Schreyer ordering* on R^r is defined as follows: Fix $f_1, \dots, f_r \in R$. Then $ue_i \prec ve_j$ iff $u\,\mathrm{lm}_\prec(f_i) \prec v\,\mathrm{lm}_\prec(f_j)$ or $u\,\mathrm{lm}_\prec(f_i) = v\,\mathrm{lm}_\prec(f_j)$ and $ue_i \prec_{\text{pot}} ve_j$.

In this thesis, the POT order will be of particular importance hence we give it its own notation above.

Definition 3.6.5 (Signature)**.** The *signature* of an element $\alpha \in R^r$, denoted $\mathfrak{s}(\alpha)$, is the $\prec$-leading monomial of α.

Remark 3.6.1. As described at at the beginning of the section, for the purpose of signature-based Gröbner basis computations, we will just need the data $(\mathfrak{s}(\alpha), \overline{\alpha})$. This data is enough to implement the replacement rule from the beginning of this section as long as one disallows reductions that would change the signature (in the following indicated by the keyword *regular*). We work here in the module R^r to utilize signature-based algorithms for syzygy computations of the sequence $f_1, \dots, f_r$, which we will use in later chapters. For this, the choice of module monomial order also plays a crucial role.

For two elements $\alpha, \beta \in R^r$ we construct a signature-theoretic analogue of the S-pairs in Buchberger's algorithm as follows:

Definition 3.6.6 (S-Pair, Signature Version)**.** The S-pair between $\alpha, \beta \in R^r$ is denoted by $\mathfrak{sp}(\alpha, \beta)$ and defined as follows: Let

$$\begin{aligned} c &:= \mathrm{lcm}(\mathrm{lm}(\overline{\alpha}, \overline{\beta}) \\ a &:= c/\mathrm{lm}(\overline{\alpha}) \\ b &:= c/\mathrm{lm}(\overline{\beta}). \end{aligned}$$

Then

$$\mathfrak{sp}(\alpha, \beta) := a\alpha - b\beta.$$

This S-pair is called *regular* if $\mathfrak{s}(a\alpha) \neq \mathfrak{s}(b\beta)$ and *singular* otherwise.

The *regular reduction* of an element $\alpha \in R^r$ with respect to a finite set $G \subset R^r$ of sig-poly pairs is defined to be the output of Algorithm 8. The procedure tries to reduce the leading term of $\overline{\alpha}$ using some multiple $b\beta$ of an element $\beta \in G$ such that $b\mathfrak{s}(\beta) \prec \mathfrak{s}(\alpha)$. The procedure stops when there is no such reducer. Compared to the usual division algorithm in polynomial rings, only reductions by lower signature elements are allowed.

Algorithm 8 Regular reduction

procedure RegularReduction(α, G, $\prec$)
  $\gamma \leftarrow \alpha$
  while $\mathcal{R} := \left\{(b, \beta) \in R \times G \;\middle|\; b\,\mathrm{lt}(\overline{\beta}) = \mathrm{lt}(\overline{\gamma}), b\mathfrak{s}(\beta) \prec \mathfrak{s}(\gamma)\right\} \neq \varnothing$
    $(b, \beta) \leftarrow$ some element in $\mathcal{R}$
    $\gamma \leftarrow \gamma - b\beta$
  return γ

We may now describe a variant of Buchberger's algorithm using signatures and only regular reductions and S-pairs, see Algorithm 9.

Algorithm 9 Buchberger with signatures

Input: $f_1, \ldots, f_r \in R$, a module monomial order $\prec$
Output: A Gröbner basis of $\langle f_1, \ldots, f_r \rangle$, a $\prec$-Gröbner basis of the syzygy module of $f_1, \ldots, f_r$.

procedure Buchberger($f_1, \ldots, f_r$, $\prec$)
  $G \leftarrow \{e_i \mid 1 \leqslant i \leqslant r\}$
  $S \leftarrow \emptyset$
  $P \leftarrow \{(\alpha, \beta) \mid \alpha, \beta \in G \text{ form a regular S-pair}\}$
  while $P \neq \varnothing$
    $(\alpha, \beta) \leftarrow$ the pair in P with $\mathfrak{s}(\mathfrak{sp}(\alpha, \beta))$ minimal w.r.t. $\prec$
    $P \leftarrow P \setminus \{\alpha, \beta\}$
    $\gamma \leftarrow$ RegularReduction($\mathfrak{sp}(\alpha, \beta)$, G)
    if $\overline{\gamma} \neq 0$
      $G \leftarrow G \cup \{\gamma\}$
      $P \leftarrow P \cup \{(\gamma, \beta) \mid \beta \in G \text{ forms a regular S-pair with } \gamma\}$
    else *(record a syzygy)*
      $S \leftarrow S \cup \{\gamma\}$
  return $\left\{\overline{\beta} \;\middle|\; \beta \in G\right\}, S$

Remark 3.6.2. Algorithm 9 computes a Gröbner basis of the syzygy module of $f_1, \ldots, f_r$, i.e. of the kernel of the map $\overline{\bullet} : R^r \to R$. While we have not formally defined this notion, using the concept of a module monomial order it is defined exactly as a Gröbner basis of an ideal in R.

For proofs of correctness and termination we refer to Theorem 5.1 in Eder and Faugère (2017).

3.6.2 *From Buchberger's Algorithm to sGB-Computations*

Now we can put the replacement rule from the beginning of this section into a formal statement. Recall that the overarching principle is the following: *at most one element in* R^r *has to be regular-reduced at each signature*. This is made precise by the following statement.

Lemma 3.6.1 (Lemma 6.2 in Eder and Faugère, 2017)**.** *In the course of Algorithm 9, assume that only S-pairs in signature $\succeq \sigma$ are left in P. Let G be the current intermediate state of the Gröbner basis of $I = \langle f_1, \dots, f_r \rangle$ computed by the algorithm at the beginning of its while-loop. Then for any $\gamma, \gamma' \in R^r$ with $\mathfrak{s}(\gamma) = \mathfrak{s}(\gamma') = \sigma$,*

$$\overline{\text{RegularReduction}(\gamma, G)} = \overline{\text{RegularReduction}(\gamma', G)}$$

This now furnishes several criteria to get rid of S-pairs in Algorithm 9 without affecting correctness. Before making them precise, define

Definition 3.6.7 (Koszul Syzygy)**.** Let $\alpha, \beta \in R^r$. The *Koszul syzygy* between α and β is defined as

$$\text{ksyz}(\alpha, \beta) = \overline{\beta}\alpha - \overline{\alpha}\beta.$$

Note that for any $\alpha \in R^r$ and any monomial u we have $\mathfrak{s}(u\alpha) = u\mathfrak{s}(\alpha)$. Now we can scratch any S-pair $\mathfrak{sp}(\alpha, \beta) = a\alpha - b\beta$ if either (a, α) or (b, β) are *rewritable*:

Definition 3.6.8 (Rewritability)**.** During a run of Algorithm 9 let G be the current intermediate state at the beginning of the algorithm's while-lopp of the output Gröbner basis of $\langle f_1, \dots, f_r \rangle$ and let S be the current intermediate state of the output Gröbner basis of the syzygy module of $f_1, \dots, f_c$. Let $u \in \text{Mon}(R)$ be a monomial and $\alpha \in G$. Then (u, α) is called *rewritable* if either one of the following are true:

SYZYGY CRITERION There exists $\sigma \in S$ with $\sigma \mid u\mathfrak{s}(\alpha)$.

KOSZUL CRITERION There exists $\beta_1, \beta_2 \in G$ with

$$\mathfrak{s}(\text{ksyz}(\beta_1, \beta_2)) \mid u\mathfrak{s}(\alpha).$$

SINGULAR CRITERION There exists $\beta \in G$ added later to G than α with $\mathfrak{s}(\beta) \mid u\mathfrak{s}(\alpha)$.

Applying these criteria we get

Algorithm 10 sGB Algorithm with rewrite checks

Input: $f_1, \ldots, f_r \in R$, a module monomial order $\prec$
Output: A Gröbner basis G of $\langle f_1, \ldots, f_r \rangle$, a set S such that $S \cup \{\mathrm{ksyz}(\alpha, \beta) \mid \alpha, \beta \in G\}$ is Gröbner basis of the syzygy module of $f_1, \ldots, f_r$.

```
1:  procedure sGB(f_1, ..., f_r, ≺)
2:      G ← {ε_i | 1 ⩽ i ⩽ r}
3:      S ← ∅
4:      P ← {(α, β) | α, β ∈ G form a regular S-pair}
5:      while P ≠ ∅
6:          (α, β) ← the pair in P with 𝔰(𝔰p(α, β)) minimal w.r.t. ≺
7:          P ← P \ {α, β}
8:          a, b ← the monomials s.t. 𝔰p(α, β) = aα − bβ
9:          if (a, α) and (b, β) are not rewritable w.r.t. G and S
10:             γ ← RegularReduction(𝔰p(α, β), G)
11:             if γ̄ ≠ 0
12:                 G ← G ∪ {γ}
13:                 P ← P ∪ {(γ, β) | β ∈ G forms a regular S-pair with γ}
14:             else  (record a syzygy)
15:                 S ← S ∪ {γ}
16:     return {β̄ | β ∈ G}, S
```

Correctness and termination of this algorithm is given in Theorem 7.1 in Eder and Faugère (2017).

Remark 3.6.3. Morally, in line with the beginning of this section, it is useful to think of an S-pair $u\alpha - b\beta$ with $\mathfrak{s}(b\beta) \prec \mathfrak{s}(u\alpha)$ as a row in Lazard's algorithm with signature $\mathfrak{s}(u\alpha)$. Then the singular criterion in Definition 3.6.8 implements the replacement rule from the beginning of this section. If some $\gamma \in G$ was added later to G than α and we have $\mathfrak{s}(u\alpha) = \mathfrak{s}(w\gamma)$ for some monomial γ then we expect $w\gamma$ to be "easier to reduce" than $u\alpha$. The syzygy criterion reflects the fact that we can scratch any row whose signature is divisible by the signature of a row that previously reduced to zero. The Koszul criterion is just a special case of the syzygy criterion but we mention it for later applications based on Proposition 3.6.1 given below. The question remains why we are allowed to check $b\beta$ for rewritability as well: This is based on the fact that, if some element $u\alpha$ has a regular reducer in G, then we can always find a non-rewritable reducer, see Lemma 7.3 in Eder and Faugère (2017).

Example 3.6.1. To illustrate Algorithm 10 let us illustrate it on the example given by $f_1 := x^2$ and $f_2 := xy + y^2$ in $\mathbb{Q}[x, y]$. We use the DRL order as our monomial order and the POT order as our module order. We start by setting

$$G := \{\epsilon_1, \epsilon_2\}$$

at the beginning of the algorithm. The pairset P contains then only the regular pair

$$\mathfrak{sp}(e_2, e_1) = xe_2 - ye_1$$

which has image $-xy^2$ in R. We then regular reduce this S-pair using ye_2 (the leading monomial of the image of which is precisely xy^2), which yields $\gamma := (x+y)e_2 - ye_1$. We add γ to G and build the S-pairs with the existing elements in G, i.e. e_1 and e_2:

$$\mathfrak{sp}(\gamma, e_1) = x^2\gamma - y^3e_1$$
$$\mathfrak{sp}(\gamma, e_2) = x\gamma - y^2e_2.$$

The first S-pair has signature x^3e_2 and so is rewriteable by the Koszul criterion, since this signature is divisible by $\mathrm{lm}(\overline{e_1}) = x^2$. The second S-pair has signature x^2e_2 and so is eliminated by the same rewrite check. We finally return G and $S = \emptyset$, i.e. there are no non-Koszul syzygies between f_1 and f_2.

Remark 3.6.4. As already remarked at the beginning of this section, the data of $(\mathfrak{s}(\alpha), \overline{\alpha})$ suffices to compute a Gröbner basis with Algorithm 10. In this case we of course do not compute a full $\prec$-Gröbner basis of the syzygy module of $f_1, \ldots, f_r$ but just the syzygy signatures, i.e. the leading monomials of such a Gröbner basis which are then only applied to make use of the Syzygy criterion. In fact, even if we want to use Algorithm 10 to compute syzygies, tracking the full module representation of the considered polynomials is computationally impractical. In the chapters where we require more data about the syzygies, other than just the signatures, we will point out more efficient ways to compute this data rather than directly tracking the module representations as in the pseudocode here.

3.6.3 *sGB's and Regular Sequences*

We will now point out how sGB computations relate to the computation of quotient and saturation ideals and regular sequences. We will use these properties in Chapter 4. Recall that Algorithm 10 depends on choosing a monomial order on R^r.
Note that Algorithm 10 processes S-pairs by increasing order on R^c. If this order is the POT order, then the consequence of this is that Algorithm 10 computes a Gröbner basis for $\langle f_1, \ldots, f_r\rangle$ incrementally: First for $\langle f_1, f_2\rangle$ then for $\langle f_1, f_2, f_3\rangle$ and so on. Indeed, all S-pairs with signature of the form ue_i lie in $\langle f_1, \ldots, f_i\rangle$ and will be processed before any S-pair with signature of the form ve_j if $i < j$.
Further, with this choice of ordering, a byproduct of running Algorithm 10 on $f_1, \ldots, f_r$ is the following: If we let $I_i := \langle f_1, \ldots, f_i\rangle$, then the output set S of Algorithm 10 consists of generators of all colon ideals $(I_{i-1} : f_i)$ for $2 \leqslant i \leqslant r$. More precisely, let $\pi_i : R^r \to R$ be the

projection to the ith coordinate. Further let S be the set of syzygies of $f_1, \dots, f_r$ returned by running Algorithm 10 on S. Then

Proposition 3.6.1. *Let*

$$S_i := \{\pi_i(\gamma) \mid \gamma \in S, \mathfrak{s}(\gamma) = u\epsilon_i \text{ for some } u \in \mathrm{Mon}(\mathbf{x})\}.$$

Then $(I_{i-1} : f_i) = I_{i-1} + \langle S_i \rangle$, $S_i \cap I_{i-1} = 0$ *and all elements in* S_i *have distinct leading monomial not lying in* $\mathrm{lm}(I_{i-1})$. *In particular,* $f_1, \dots, f_r$ *is a regular sequence if and only if the output set* S *of running Algorithm 10 on* $f_1, \dots, f_r$ *is empty, i.e. no reduction to zero is computed.*

Proof. Fix some $i = 2, \dots, r$ and let $\gamma \in S$ with $\mathfrak{s}(\gamma) = u\epsilon_i$. By definition of the POT order this means that

$$\gamma = \sum_{j=1}^{i} \pi_j(\gamma)\epsilon_j$$

and

$$0 = \overline{\gamma} = \sum_{j=1}^{i} \pi_j(\gamma) f_j.$$

In particular $\pi_i(\gamma) \in (I_{i-1} : f_i)$. Suppose that $u := \mathrm{lm}(\pi_i(\gamma)) \in \mathrm{lm}(I_{i-1})$, i.e. $\mathfrak{s}(\gamma) = u\epsilon_i$. By definition of the POT order, Algorithm 10 has computed a Gröbner basis for I_{i-1} before considering any element in signature $\mathfrak{s}(\gamma)$. Hence there must exist $\alpha \in G$ with $\mathfrak{s}(\alpha) = v\epsilon_j$ and $j < i$, added to G before γ has been considered, with $\mathrm{lm}(\overline{\alpha}) \mid u$. Consider the Koszul syzygy $\kappa := \overline{\alpha}\epsilon_i - f_i\alpha$. Then

$$\mathfrak{s}(\kappa) = \mathrm{lm}(\overline{\alpha})\epsilon_i \mid u\epsilon_i = \mathfrak{s}(\gamma),$$

and so no element in $\mathfrak{s}(\gamma)$ is computed by the Koszul criterion, a contradiction. The property that all elements in S_i have distinct leading monomials is proven by a similar argument using the Syzygy criterion. □

Part II

CONTRIBUTIONS

4

COMPUTING THE NONDEGENERATE LOCUS

As usual in this thesis, let $R := \mathbb{K}[\mathbf{x}]$ be a polynomial ring over a field $\mathbb{K}$ in a finite set of variables $\mathbf{x}$. Denote by $\mathbb{A}^n$ the corresponding affine space over $\overline{\mathbb{K}}$. Let $F := (f_1, \ldots, f_c) \in R$ be a finite sequence in R. We define

Definition 4.0.1 (Nondegenerate Locus)**.** The *nondegenerate locus* of a finite sequence $F \subset R$ is defined as

$$\operatorname{ndeg}(F) := \{p \in \mathbf{V}(F) \mid F \text{ is a regular sequence in } R_p\}.$$

Our goal in this chapter is to design an algorithm which computes a Gröbner basis of an ideal K such that $\mathbf{V}(K) = \overline{\operatorname{ndeg}(F)}$. This algorithm will owe its efficiency to the careful exploitation of certain inherent features of signature-based Gröbner basis algorithms, which were introduced in Section 3.6.
In Section 4.1 we first give a basic description of our algorithm, without explicit usage of signature-based techniques. Despite of its geometric goal we give it purely in terms of ideal-theoretic operations, to make the connection to signature-based Gröbner basis algorithms easier in Section 4.2.
The content of this chapter is based on Eder, Lairez, Mohr, and Safey El Din (2023b).

4.1 THE BASIC ALGORITHM

The computation of the nondegenerate locus of a given sequence $F := (f_1, \ldots, f_c)$ is based on the following simple observation:

Proposition 4.1.1. *Let* $F' := (f_1, \ldots, f_{c-1})$. *Then*

$$\operatorname{ndeg}(F) = \operatorname{reg}(\operatorname{ndeg}(F'), f_c).$$

Proof. This follows immediately from the definition of a regular sequence: F defines a regular sequence at a point $p \in \mathbf{V}(F)$ iff F' defines a regular sequence at p and if f_c regularly intersects $\mathbf{V}(F')$ at p. □

Again we denote for two ideals $J, K \subset R$ $J \overset{\text{rad}}{=} K$ if $\sqrt{J} = \sqrt{K}$.
Recall now from Proposition 2.3.2 that for $X := \operatorname{ndeg}(F')$ we have $\operatorname{reg}(X, f) = X \setminus \left[\overline{X \setminus \mathbf{V}(J)}\right]$ where $J := (\mathbf{I}(X) : f)$. Being given an ideal I with $\mathbf{V}(I) = \overline{X}$ we then have to perform the following ideal-theoretic operations to obtain an ideal K with $\mathbf{V}(K) = \overline{\operatorname{ndeg}(F)} = \overline{\operatorname{reg}(X, f)}$:

(1) Compute the ideal

$$J := (I : f^\infty)\,.$$

Note that $J \overset{\text{rad}}{=} (\mathbf{I}(X) : f)$ (Proposition 2.2.2) and therefore $\mathbf{V}(J) = \overline{X \setminus \mathbf{V}(f)}$.

(2) Compute the ideal

$$H = (I : J) \overset{\text{Proposition 2.2.5}}{=} (I : J^\infty)\,.$$

Note that $H \overset{\text{rad}}{=} \mathbf{I}(X \setminus \mathbf{V}(J))$ (Proposition 2.2.3) and therefore $\mathbf{V}(H) = \overline{X \setminus \mathbf{V}(J)}$.

(3) For an ideal I_f with $I_f \overset{\text{rad}}{=} \mathbf{I}(X \cap \mathbf{V}(f))$ compute the ideal

$$K := (I_f : H^\infty)\,.$$

Now we have $\mathbf{V}(K) = \overline{\operatorname{reg}(\overline{X}, f)}$. Recall that X is a locally closed set and may be written as $\mathbf{V}(I) \setminus \mathbf{V}(N)$ where N is a suitable polynomial ideal. An ideal I_f as above may then be obtained as $(I + \langle f \rangle : N^\infty)$.

In purely ideal-theoretic terms this now furnishes the following informal algorithm to compute the nondegenerate locus of F:

Algorithm 11 Computation of the nondegenerate locus

Input: A finite sequence $f_1, \ldots, f_c$ in R where $c \leqslant n$
Output: An ideal K with $K \overset{\text{rad}}{=} \mathbf{I}(\operatorname{ndeg}(f_1, \ldots, f_c))$

```
K ← 0, as an ideal of R
𝓗 ← ∅
for k from 1 to c
    J ← (K : f_k^∞)        Item (1) above
    𝓗 ← 𝓗 ∪ {(K : J)}      Item (2) above
    K ← K + ⟨f_k⟩          Item (3) above
    for H ∈ 𝓗
        K ← (K : H^∞)      Item (3) above
return K
```

Theorem 4.1.1. *Algorithm 11 is correct and terminates.*

Proof. The termination of the algorithm is clear. To prove correctness, assume inductively that Algorithm 11 has correctly computed the nondegenerate locus of the partial sequence $f_1, \ldots, f_k$ as an ideal K. Note that this means that

$$\mathbf{V}(K) = \overline{\mathbf{V}(f_1, \ldots, f_k) \setminus \bigcup_{H \in \mathcal{H}} \mathbf{V}(H)}$$

with $\mathcal{H}$ in its state before we start the $k+1$st iteration of the for-loop of Algorithm 11. The claimed correctness follows because Algorithm 11 performs exactly the steps given in Item (1) through Item (3) above. □

Remark 4.1.1. A straightforward implementation of Algorithm 11 can be given via Gröbner basis computations: To perform the necessary saturations we can use Lemma 3.1.2 and Lemma 2.2.2. To compute the colon ideals in Line 5 we can use e.g. Proposition 6.33 in Becker and Weispfenning (1993).

We will now explain how to instantiate Algorithm 11 into a concrete algorithm using signature-based computations.

4.2 INTEGRATION WITH SIGNATURE-BASED GRÖBNER BASIS COMPUTATIONS

In order to instantiate Algorithm 11 efficiently using sGB computations, we will use the properties laid out in Section 3.6.3, i.e. the incremental structure coming performing sGB computations using the POT order (Definition 3.6.4) and their properties laid out in Proposition 3.6.1.

4.3 THE SGB TREE DATASTRUCTURE

We first specify a data structure, called *sGB tree*. It is meant to extend Algorithm 10 in two ways: by offering the possibility to add new input equations during the computation; and to then facilitate the saturation and colon ideal computations needed for Algorithm 11.

4.3.1 *Specification*

An sGB tree represents a rooted tree $\mathcal{T}$ where each node holds an element of the polynomial ring R. The nodes are partially ordered by the ancestor-descendant relation: $\nu \prec_{\mathcal{T}} \mu$ if ν is on the unique path from μ to the root of $\mathcal{T}$ (or, equivalently, if μ is in the subtree rooted at ν). For a node ν, the polynomial contained in ν is denoted $\overline{\nu}$, and the ideal generated by the polynomials contained by the ancestors of ν (not including ν) is denoted $I_{<\nu}$. An sGB tree offers the following three operations. How we implement them is the matter of the next section.

NODE INSERTION Insert a new node, containing a given polynomial f, anywhere in the tree, as a new leaf or on an existing edge. Denoted InsertNode($\mathcal{T}$, f, position).

GRÖBNER BASIS Given a node ν, outputs a Gröbner basis of the ideal generated by the polynomials contained in the nodes $\preceq_{\mathcal{T}} \nu$. Denoted Basis($\mathcal{T}$, ν).

GET A SYZYGY Given a node ν, outputs an element of $I_{<\nu} : \overline{\nu}$. Denoted GetSyzygy($\mathcal{T}$, ν).

If GetSyzygy($\mathcal{T}$, ν) outputs zero, then $I_{<\nu} + J = I_{<\nu} : \overline{\nu}$, where J is the ideal generated by all previous invocations of GetSyzygy($\mathcal{T}$, ν).

GetSyzygy will modify some state of $\mathcal{T}$ and we demand that this causes GetSyzygy($\mathcal{T}$, ν) to eventually output zero after sufficiently many invocations.

4.3.2 *Implementation*

An sGB tree $\mathcal{T}$ can now be implemented using sGB computations. To this end, recall that running Algorithm 10 on a sequence $f_1, \ldots, f_r$ of polynomials performs operations on elements in the free module R^r while considering the images of these elements in R^r under the map mapping the ith unit vector to f_i. This transports to working with an sGB tree $\mathcal{T}$ as follows: For any node ν consider the ordered sequence of all $\overline{\mu}$ where $\mu \preceq_{\mathcal{T}} \nu$. Just as above, we may perform operations on elements in the free module corresponding to this sequence and consider their images under the map $\overline{\bullet}$, defined as at the beginning of Section 3.6.1. The unit vectors in such a free module now correspond to some nodes in $\mathcal{T}$. For a node ν in $\mathcal{T}$ we denote by ϵ_ν the corresponding unit vector.
For an element α in any such defined free module we can now define the signature $\mathfrak{s}(\alpha)$ w.r.t. the POT ordering (induced by the ordering $\prec_{\mathcal{T}}$). We additionally define the *node* of α, denoted $\mathrm{node}(\alpha)$, via $\mathrm{node}(\alpha) := \nu$ if $\mathfrak{s}(\alpha) = u\epsilon_\nu$ for some monomial $u \in \mathrm{Mon}(\mathbf{x})$ and a node ν in $\mathcal{T}$.
Two elements α, β in potentially two different thusly defined free modules are said to be *comparable* if $\mathrm{node}(\alpha)$ and $\mathrm{node}(\beta)$ are comparable by $\prec_{\mathcal{T}}$ (i.e. if $\mathrm{node}(\alpha)$ and $\mathrm{node}(\beta)$ lie on the same path in $\mathcal{T}$), in which case α, β may be considered as part of the same free module in a canonical manner. We redefine the notion of an S-pair to be built only between comparable elements.
Now we can adapt Algorithm 10 to implement the sGB tree data structure, yielding Algorithm 12 below.

Remark 4.3.1. In Algorithm 12, the sets G, P and S are considered as part of the inherent state of $\mathcal{T}$. They are modified by the different procedures in Algorithm 12 as side effects. More precisely, we assume that the inherent state of a sGB tree always results from a sequence of calls to InsertNode, Basis or GetSyzygy applied to an initially empty tree.

Proposition 4.3.1. *Let $\mathcal{T}$ be a sGB tree and let ν be a node of $\mathcal{T}$.* Basis($\mathcal{T}$, ν) *given in Algorithm 12 terminates and outputs a Gröbner basis of the ideal $I_{\leqslant\nu}$ generated by all $\overline{\mu}$ with $\mu \preceq_{\mathcal{T}} \nu$.*

Proof. This algorithm considers only S-pairs whose signatures are above a given node ν. After this restriction, the signatures are totally

Algorithm 12 Implementation of the sGB tree data structure

Input: An sGB tree $\mathcal{T}$ and a node ν of $\mathcal{T}$
Output: Process the pair in P with node above ν with smallest signature

procedure ProcessSPair($\mathcal{T}$, ν)
  (restrict to S-pairs whose nodes are above ν)
  $P' \leftarrow \{(\alpha,\beta) \mid \alpha,\beta \in G, \max\{\mathrm{node}(\alpha),\mathrm{node}(\beta)\} \prec_{\mathcal{T}} \nu\}$
  if $P' \neq \varnothing$
    $(\alpha,\beta) \leftarrow$ the pair in P' with $\mathfrak{s}(\mathfrak{sp}(\alpha,\beta))$ minimal
    $P \leftarrow P \setminus \{(\alpha,\beta)\}$
    $a, b \leftarrow$ the monomials such that $a\alpha - b\beta = \mathfrak{sp}(\alpha,\beta))$
    if (a,α) and (b,β) are not rewriteable w.r.t. G and S
      $\gamma \leftarrow$ RegularReduction($\mathfrak{sp}(\alpha,\beta)$, G)
      if $\overline{(\gamma)} \neq 0$
        $G \leftarrow G \cup \{\gamma\}$
        $P \leftarrow P \cup \{(\gamma,\beta) \mid \beta \in G$ forms a regular S-pair with $\gamma\}$
      else *(record a syzygy)*
        $S \leftarrow S \cup \{\gamma\}$

Input: A sGB tree $\mathcal{T}$ and a node ν of $\mathcal{T}$
Output: A Gröbner basis of $I_{<\nu}$

procedure Basis($\mathcal{T}$, ν)
  while there is a pair in P with node $\preceq_{\mathcal{T}} \nu$
    ProcessSPair($\mathcal{T}$, ν)
  return $\{\overline{\alpha} \mid \alpha \in G$ and $\mathrm{node}(\alpha) \preceq_{\mathcal{T}} \nu\}$

Input: A sGB tree $\mathcal{T}$ and a node ν of $\mathcal{T}$
Output: An element of the quotient ideal $(I_{<\nu} : \overline{\nu})$ not contained in $I_{<\nu}$

procedure GetSyzygy($\mathcal{T}$, ν)
  while there is a pair in P with node $\preceq_{\mathcal{T}}$ ν **and** $S_\nu := \{\pi_{\epsilon_\nu}(\gamma) \mid \gamma \in S, \mathrm{node}(\gamma) = \nu\} = \emptyset$
    ProcessSPair($\mathcal{T}$, ν)
  if $S_\nu \neq \varnothing$
    pick and remove some h in S_ν
    return h
  else
    return 0

Input: A sGB tree $\mathcal{T}$, a polynomial f and a description of the position of the new node in $\mathcal{T}$
Output: The new node in $\mathcal{T}$

procedure InsertNode($\mathcal{T}$, f, position)
  insert a node ν with $\overline{\nu} = f$ in $\mathcal{T}$, as described by "position"
  $S_\nu \leftarrow \varnothing$
  $P \leftarrow \{(\epsilon_\nu,\beta) \mid \beta \in G$ and (ϵ_ν,β) forms a regular S-pair$\}$
  $G \leftarrow G \cup \{\epsilon_\nu\}$
  return ν

ordered, so Basis behaves exacly like Algorithm 10. We note that, contrary to Algorithm 10, Basis may start in a state where several S-pairs have already been processed, in an unspecified order, by earlier calls to Basis or GetSyzygy on different nodes. This does not invalidate either the termination and correctness proof for Algorithm 10 given in Theorem 7 in Eder and Roune (2013), see also Theorem 7.1 in Eder and Faugère (2017) and Algorithmic Property 7.1 (c) therein, where the case of handling S-pairs in any order is explicitly treated. □

Proposition 4.3.2. *Let $\mathcal{T}$ be a sGB tree and let ν be a node of $\mathcal{T}$.* GetSyzygy($\mathcal{T}$, ν) *(Algorithm 12) terminates and outputs some $f \in R$ such that:*

(1) $f \in (I_{<\nu} : \overline{\nu})$;

(2) if $f \neq 0$, then $\mathrm{lm}(f)$ is not divisible by the leading monomial of any other polynomial previously output by GetSyzygy($\mathcal{T}$, ν)*, or any polynomial in $I_{<\nu}$;*

(3) if $f = 0$, then $(I_{<\nu} : \overline{\nu})$ is generated by $I_{<\nu}$ and the polynomials previously output by GetSyzygy($\mathcal{T}$, ν)*.*

Proof. Termination follows from the termination of Basis since the main loop is similar, but with the possibility of earlier termination. Correctness follows from Proposition 3.6.1 after again restricting to nodes above ν. □

Remark 4.3.2. As pointed out in Remark 3.6.4, it is computationally infeasible to track full module representations in a sGB computation. For the purposes of GetSyzygy it suffices to track just the largest nonzero entry of each module element (i.e. the full entry of which the signature is the leading monomial): The output returned in Line 5 of GetSyzygy is $\pi_{\epsilon_\nu}(\gamma)$ of an element γ with $\mathrm{node}(\gamma) = \nu$. Tracking just these highest entries turns out to not produce a large overhead in computations.

4.4 COMPUTATION OF THE NONDEGENERATE LOCUS

The sGB tree data structure will now be used to implement an efficient variant of Algorithm 11 for computing an ideal representing the nondegenerate locus. We use an sGB tree to efficiently compute saturations of the form $(I : f^\infty)$ for an ideal $I \subset R$ and $f \in R$, and also double quotients of the form $(I : (I : f^\infty))$, with the idea to exploit newly discovered relations as soon as possible to simplify further computations. This leads to Algorithm 13, which we describe informally as follows.

We use the POT ordering in the following but explicitly introduce the equations $f_1, \ldots, f_r$ one after the other, as Algorithm 10 would do implicitly in this case, to make the relation to Algorithm 11 clearer.

We maintain a sGB tree which, at the beginning of the kth iteration, that is after having processed $f_1, \ldots, f_{k-1}$, has the following shape:

$$\mathbf{g}_1 \leftarrow f_1 \leftarrow \mathbf{p}_1 \leftarrow \cdots \leftarrow \mathbf{g}_{k-1} \leftarrow f_{k-1} \leftarrow \mathbf{p}_{k-1} \leftarrow \underbrace{0}_{\nu} \begin{array}{l} \swarrow\, h_1 \\ \leftarrow h_2, \\ \vdots \end{array}$$

where bold letters represent a sequence of zero, one or several nodes. The tree grows from the node labeled ν, by adding new leaf nodes, or inserting nodes just above ν. Using the notations of Algorithm 11, the nodes $\mathbf{g}_i$ are contained in the saturations by the f_i in Line 4, the leaf nodes h_i are generic elements of the ideals in the set $\mathcal{H}$, and the nodes $\mathbf{p}_i$ are contained in the saturations $(G : K^\infty)$ in Line 8. We added $\mathbf{g}_1$ for the consistency of the above picture although it will always be empty and so f_1 is the root of the tree.

Remark 4.4.1 (Deterministic variant). The leaf nodes h_i are generic in the sense that they are linear combinations of generators of the ideals in $\mathcal{H}$ where all coefficients are either random scalars or new variables. Algorithm 13 chooses either a random scalar or a new variable in Line 11. For a randomized algorithm, favoring speed over deterministic correctness, choose t to be a random scalar, this choice is justified by Lemma 2.2.2. For a deterministic algorithm, choose t to be a slack variable, unused in the input equation. It is guaranteed that such a t is generic enough, again by Lemma 2.2.2. In this case the monomial order on R has to be extended to $R[t]$ by a monomial order that eliminates t. The implementation discussed in the next section exclusively chooses t to be a random scalar.

The kth iteration proceeds as follows. Firstly, a new node μ containing f_k is created just above ν:

$$\cdots \leftarrow \underbrace{f_k}_{\mu} \leftarrow \underbrace{0}_{\nu} \leftarrow \cdots.$$

As long as GetSyzygy$(\mathcal{T}, \mu)$ returns nonzero elements $(g_1, g_2, \ldots)$, we insert them above μ:

$$\cdots \leftarrow g_1 \leftarrow g_2 \leftarrow \cdots \leftarrow \underbrace{f_k}_{\mu} \leftarrow \underbrace{0}_{\nu} \leftarrow \cdots.$$

This saturation has the effect of completing $I_{<\mu}$ into $\left(I_{<\mu} : f_k^\infty\right)$. Each time we insert a polynomial g_i in a node, say γ, we also record the syzygies GetSyzygy$(\mathcal{T}, \gamma)$, take a generic linear combination of them and insert it as a new leaf node. These syzygies come from the double quotients $\left(I_{<\mu} : \left(I_{<\mu} : f_k^\infty\right)\right)$ computed in Line 5 of Algorithm 11. Before going to the next iteration, insert above ν all the syzygies obtained from the children of ν, this again has the effect of saturating $I_{<\nu}$ by the polynomials contained in these nodes, and performs the ideal-theoretic operations in Line 8 of Algorithm 11.

After all the input equations have been processed, the ideal $I_{<\nu}$ represents the closure of the nondegenerate locus of the input, which we will prove by comparing with Algorithm 11. Before this proof, let us illustrate Algorithm 13 using a concrete example:

Example 4.4.1. To illustrate Algorithm 13 let us illustrate its behaviour on the sequence $F = (xy, xz) \subset \mathbb{Q}[x, y, z]$. During the first loop starting in Line 5, the sGB tree $\mathcal{T}$ has the shape

$$xy \leftarrow \underbrace{0}_{\nu}.$$

No actual computations happen in this loop because we are just working with a single element. In the next iteration of the loop, we insert the second element of the sequence, the sGB tree $\mathcal{T}$ has the shape

$$xy \leftarrow xz \leftarrow \underbrace{0}_{\nu}.$$

Now, if μ is the node which contains xz, then $\textsc{GetSyzygy}(\mathcal{T}, \mu)$ returns $g = y$. Then, after Line 14, $\mathcal{T}$ has the shape

$$xy \leftarrow y \leftarrow xz \leftarrow \underbrace{0}_{\nu}.$$

Next, we run through the loop starting in Line 12. If γ is node containing y then we first find $\textsc{GetSyzygy}(\mathcal{T}, \gamma) = x$ in Line 13. The next call $\textsc{GetSyzygy}(\mathcal{T}, \gamma)$ already returns zero, so in Line 17 we insert $h := x$ and $\mathcal{T}$ now has the shape

$$xy \leftarrow y \leftarrow xz \leftarrow \underbrace{0}_{\nu} \leftarrow x.$$

Finally, in the loop starting in Line 18, we only find $b = z$, so after Line 23, $\mathcal{T}$ takes the shape

$$xy \leftarrow y \leftarrow xz \leftarrow z \leftarrow \underbrace{0}_{\nu} \leftarrow x.$$

Finally we terminate with the basis $\{xy, y, xz, z\}$.

Theorem 4.4.1. *Algorithm 13 terminates. Algorithm 13 is correct, provided that one chooses t as a new variable in line 12 or that t is chosen as a scalar in a suitable Zariski open subset of $\mathbb{K}$.*

Proof. Termination follows from the assumption that for any node ν of a sGB tree $\mathcal{T}$, $\textsc{GetSyzygy}(\mathcal{T}, \nu)$ eventually returns 0 after sufficiently many calls.
To prove correctness, we show that Algorithm 13 computes the same ideal as Algorithm 11. Let J_{k-1} be the value of I_ν at the beginning of the kth iteration. After Line 4, we also have $I_{<\mu} = J_{k-1}$, while $I_{<\nu} = I_{<\mu} + \langle f_k \rangle$.

Algorithm 13 Computation of the nondegenerate locus with an sGB tree

Input: $f_1, \ldots, f_c \in R$
Output: A Gröbner basis G of an ideal representing nondegenerate locus of $(f_1, \ldots, f_c)$

$\mathcal{T} \leftarrow$ an empty sGB tree
$\nu \leftarrow$ InsertNode($\mathcal{T}$, 0)
for k from 1 to c
    $\mu \leftarrow$ InsertNode($\mathcal{T}$, f_k, just above ν)
    loop
        $g \leftarrow$ GetSyzygy($\mathcal{T}$, μ)
        if $g = 0$
            break
        $\gamma \leftarrow$ InsertNode($\mathcal{T}$, g, just above μ)
        $h \leftarrow 0$
        $t \leftarrow$ a random scalar (or the slack variable, see Remark 4.4.1)
        loop
            $h' \leftarrow$ GetSyzygy($\mathcal{T}$, γ)
            if $h' = 0$
                break
            $h \leftarrow th + h'$
        InsertNode($\mathcal{T}$, h, as a child of ν)
    for all child β of ν
        loop
            $b \leftarrow$ GetSyzygy($\mathcal{T}$, β)
            if $b = 0$
                break
            InsertNode($\mathcal{T}$, b, just above ν)
return Basis(ν)

We first examine the loop on Line 5. It inserts above the node μ all the polynomials obtained from GetSyzygy($\mathcal{T}$, μ). Every node inserted on Line 14 is in $(I_{<\mu} : f_k)$. No other node is inserted above μ. So by induction, it follows that all along the loop, we have $J_{k-1} \subseteq I_{<\mu} \subseteq \left(J_{k-1} : f_k^\infty\right)$. Moreover, after the loop terminates, we have $(I_\mu : f_k) = I_{<\mu}$, due to the specification of GetSyzygy (Proposition 4.3.2). If now $g \in \left(J_{k-1} : f_k^\infty\right)$ then $g \in \left(I_\mu : f_k^\infty\right) = I_{<\mu}$ and so all in all it follows that before Line 18, we have

$$I_{<\mu} = (J_{k-1} : f_k^\infty) \quad \text{and} \quad I_{<\nu} = (J_{k-1} : f_k^\infty) + \langle f_k \rangle. \tag{4.1}$$

Next, we examine the loop on Line 18 and its inner loop on Line 19. By the same argument as above, the inner loop has the effect of saturating $I_{<\nu}$ by $\overline{\beta}$. So after the loop on Line 18, we have

$$I_{<\nu} = J_k = ((J_{k-1} : f_k^\infty) + \langle f_k \rangle) : \left(\prod_{\beta \text{ child of } \nu} \overline{\beta} \right)^\infty. \tag{4.2}$$

It remains to understand the nature of the children of ν. They all come from the insertion of h on Line 17. And h is simply a generic linear combination of the return values of GetSyzygy($\mathcal{T}$, γ). So h is a generic linear combination of some $h_1, \ldots, h_r$ such that $I_{<\gamma} + \langle h_1, \ldots, h_r \rangle = (I_{<\gamma} : \overline{\gamma})$ (by Proposition 4.3.2). For each node γ inserted on Line 14, let L_γ denote the ideal $(I_{<\gamma} : \overline{\gamma})$. If $g_1, \ldots, g_s$ are the successive return values of GetSyzygy($\mathcal{T}$, μ) on Line 6, and $\gamma_1, \ldots, \gamma_r$ the corresponding nodes, we have $L_{<\gamma_i} = (I_{<\gamma_i} : g_i)$ and $I_{<\gamma_i} = J_{k-1} + \langle g_1, \ldots, g_i \rangle$. By Lemma 2.2.3, it follows that

$$L_{\gamma_1} \cap \cdots \cap L_{\gamma_r} \overset{\text{rad}}{=} (J_{k-1} : \langle g_1, \ldots, g_r \rangle^\infty). \tag{4.3}$$

Moreover, by Equation (5.4), we obtain that before Line 18

$$I_{<\mu} = J_{k-1} + \langle g_1, \ldots, g_r \rangle = (J_{k-1} : f_k^\infty), \tag{4.4}$$

so, combining with (4.3),

$$\begin{aligned} L_{\gamma_1} \cap \cdots \cap L_{\gamma_r} &\overset{\text{rad}}{=} (J : \langle g_1, \ldots, g_r \rangle^\infty) && (4.5)\\ &= J_{k-1} : \left(J_{k-1} + \langle g_1, \ldots, g_r \rangle\right)^\infty && (4.6)\\ &\overset{\text{rad}}{=} (J_{k-1} : (J_{k-1} : f_k^\infty)). && (4.7) \end{aligned}$$

As remarked above, the loop on Line 19 has the effect of saturating $I_{<\nu}$ by $\overline{\beta}$. By the analysis above, $\overline{\beta}$ is actually a generic linear combination of some $h_1, \ldots, h_r$ such that $I_{<\gamma} + \langle h_1, \ldots, h_r \rangle = L_\gamma$, for some node γ above ν. By Lemma 2.2.2, saturating by $\overline{\beta}$ is the same as saturating by $\langle h_1, \ldots, h_r \rangle$ (assuming that $\overline{\beta}$ is sufficiently generically in the case where Algorithm 13 chooses random scalars in line 12). Besides, $I_{<\nu}$ contains $I_{<\gamma}$, so saturating $I_{<\nu}$ by $\langle h_1, \ldots, h_r \rangle$ is the same as saturating by L_γ. Back to Equation (4.2), we conclude from Equation (4.7)

that saturating $I_{<\nu}$ by all the $\overline{\beta}$ is the same as saturating by all the ideals $(J_{i-1} : (J_{i-1} : f_i^\infty))$, for $i \leqslant k$.
Therefore J_k satisfies the same recurrence relation

$$J_k = \left((J_{k-1} : f_k^\infty) + \langle f_k \rangle \right) : \left(\bigcap_{i \leqslant k} (J_{i-1} : (J_{i-1} : f_i^\infty)) \right)^\infty. \tag{4.8}$$

This proves that Algorithm 13 and Algorithm 11 compute the same ideal. □

Remark 4.4.2. As mentioned in Remark 3.6.4, tracking the full module representation of each element during a signature-based Gröbner basis computation is computationally impractical. Note however, that GetSyzygy just requires the largest non-zero entry of any given syzygy, not the full syzygy itself. In the context of Algorithm 13, the leading monomial of this entry, together with its index, gives us the signature of any given element. So, in an actual implementation, we can just track the non-zero entry of highest index of any given element, this turns out to be not a computational burden compared to the actual polynomial arithmetic in the image of $\tilde{\cdot}$.

4.5 BENCHMARKS

The benchmarks for this chapter were recorded using an implementation of the author of both Algorithm 10 and Algorithm 13 in the programming language `Julia` (Bezanson, Edelman, Karpinski, and Shah, 2017) with an interface to the `Julia`-package `Singular.jl` which itself is an interface to the computer algebra system `Singular` (Decker, Greuel, Pfister, and Schönemann, 2022). This implementation is available at

`https://github.com/RafaelDavidMohr/SignatureGB.jl`

Since the recording of these benchmarks, the author has written a newer, more optimized, implementation of Algorithm 10 as part of the `Julia` package `AlgebraicSolving.jl`. The implementation used for the benchmarks here is not competitive with optimized implementations of Gröbner basis algorithms such as in the computer algebra system `Maple` or `msolve`. Despite this, the implementation used here incorporates linear algebra techniques as in the F_4 algorithm, following Section 13 in Eder and Faugère (2017) and uses certain standard techniques for optimizing Gröbner basis computations in a concrete implementation such as *monomial hashtables* as described by Monagan and Pearce (2015).
All benchmarks for this chapter were recorded with $\mathbb{K} = \mathbb{F}_{65521}$. We computed all examples on a single Intel Xeon Gold 6244 CPU @ 3.60GHz with a limit of 200G memory.
In Table A.1 we compare Algorithm 13 and a straightforward implementation of ours of Algorithm 11 in Maple on some of the polynomial

systems listed in Appendix A.1. In this `Maple` implementation, we saturated an ideal J by an ideal K by picking a random linear combination p of generators of K and saturating J by p using the internal saturation routine of `Maple`, this, probabilistically, gives the ideal $(J : K^{\infty})$ by Lemma 2.2.2.
In Table A.2 we compare Algorithm 13 to other ideal decomposition methods available in the computer algebra systems `Singular`, `Maple` and `Macaulay2` (Grayson and Stillman, n.d.). We refer to Appendix A.2 for an explanation of the symbols used in Table A.2.
In `Macaulay2` one is able to compute the intersection of all components of non-minimal dimension again with the method presented in Eisenbud, Huneke, and Vasconcelos (1992). We then saturated the original ideal by the result to obtain the nondegenerate locus. On a high level, our algorithm works similarly, incrementally obtaining information about the component of higher dimension and then removing it via saturation. One should keep in mind that all of these methods, compared to Algorithm 13, work more generally: Except for what we tried in `Macaulay2` they are all able to obtain a full equidimensional decomposition of the input ideal or respectively the algebraic set defined by this input ideal.

4.5.1 *Discussion of Experimental Results*

Despite the non-optimized nature of the implementation used to create these benchmarks, Table A.1 shows the improvement of Algorithm 13 over Algorithm 11 independent of implementational considerations: While the optimized Gröbner basis engine of `Maple` beats the implementation of Algorithm 10 mentioned above by a wide margin, the ratio between the timings of our implementation of Algorithm 10 and our implementation of Algorithm 13 is much better than the ratio between the time it took to compute a Gröbner basis in `Maple` and our `Maple` implementation of Algorithm 11. This indicates that the overhead over a Gröbner basis computation incurred by Algorithm 11 is significantly reduced by bringing in signature-based techniques as in Algorithm 13. This can be seen by looking at the two respect "ratio"-columns of Table A.1. To additionally show the overhead of Algorithm 13 over Algorithm 10 we noted the number of arithmetic operations in $\mathbb{K}$ when running each of the two algorithms on the polynomial system in question. Our implementation of Algorithm 13 never takes more than 10 times the number of arithmetic operations Algorithm 10 takes, on certain examples we compare very favorably in terms of arithmetic operations to Algorithm 10.

5

COMPUTING AN EQUIDIMENSIONAL DECOMPOSITION OF AN ALGEBRAIC SET

We reuse the notation from Chapter 4. Our goal here is to, instead of computing the nondegenerate locus of a finite sequence $F \subset R$, to present an algorithm to compute an *equidimensional partition* of $\mathbf{V}(F)$:

Definition 5.0.1 (Equidimensional Partition)**.** An *equidimensional partition* of a locally closed set $X \subset \mathbb{A}^n$ is a finite set of locally closed sets $\mathcal{D}$ such that each $Y \in \mathcal{D}$ is equidimensional, the elements of $\mathcal{D}$ are pairwise disjoint and such that

$$X = \bigcup_{Y \in \mathcal{D}} Y.$$

The content of this chapter is based on Eder, Lairez, Mohr, and Safey El Din (2023a).

5.1 THE BASIC PRINCIPLE

Following a similar outline as Chapter 4, in this section we describe the geometric principles of the algorithm presented in this chapter first without any concern for particular data structures.
We will also use similar principles as in Chapter 4. More precisely, we will iterate through the equations $F = (f_1, \dots, f_c)$ one by one, at each step producing an equidimensional partition of $\mathbf{V}(f_1, \dots, f_i)$ for each $1 \leqslant i \leqslant c$. At each incremental step we compute an equidimensional partition of $X \cap \mathbf{V}(f_i)$ where X is one of the output sets of the previous step. This computation, in turn, will rely again on the concept of regular intersection, as in Chapter 4.
To explain this computation suppose now that $X \subset \mathbb{A}^n$ is an equidimensional locally closed set and that $f \in R$. Thanks to Proposition 2.3.1 we now have the equidimensional partition

$$X \cap \mathbf{V}(f) = \overline{X \setminus \operatorname{reg}(X, f)} \sqcup [\operatorname{reg}(X, f) \cap \mathbf{V}(f)]\,. \tag{5.1}$$

Example 5.1.1. Let $R := \mathbb{Q}[x, y, z]$, let $X := \mathbf{V}(xy, xz, yz)$, the union of the three coordinate lines, and let $f := x(x-1)$. $\mathbf{V}(f)$ contains the two irreducible components $\mathbf{V}(x, y)$ and $\mathbf{V}(x, z)$ of X. Note that therefore

$$\operatorname{reg}(X, f) = X \setminus \mathbf{V}(x) = \mathbf{V}(y, z)$$

and

$$\overline{X \setminus \operatorname{reg}(X, f)} = \mathbf{V}(x, z) \cup \mathbf{V}(x, y) = \mathbf{V}(x, yz).$$

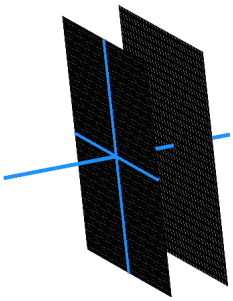


Figure 5.1: X plotted in blue and $\mathbf{V}(f)$ in black using `Maple`.

We apply Equation (5.1) here, but without computing these two locally closed sets directly. Suppose that f does not intersect X regularly, i.e. there exists $g \in R$ such that $gf \in \mathbf{I}(X)$ but $g \notin \mathbf{I}(X)$. Defining

$$X_1 = \overline{X \setminus \mathbf{V}(g)}$$
$$X_2 = X \setminus X_1$$

we can write $X = X_1 \sqcup X_2$. By Proposition 2.3.2 we now have $X_2 \supset \mathrm{reg}(X, f)$ and therefore also $X_1 \subset \overline{X \setminus \mathrm{reg}(X, f)}$ which in particular implies $f \in \mathbf{I}(X_1)$. Hence

$$X \cap \mathbf{V}(f) = X_1 \sqcup (X_2 \cap \mathbf{V}(f)) \tag{5.2}$$

with X_1 equidimensional. Having computed X_1 and X_2, we can then start the same procedure again with X replaced by X_2, i.e. we look for $g' \in R$ such that $g'f \in \mathbf{I}(X_2)$ but $g' \notin \mathbf{I}(X_2)$. If no such g' exists then f intersects X_2 regularly and Equation (5.2) gives an equidimensional partition of $X \cap \mathbf{V}(f)$.
In instantiating this algorithmic idea into an algorithm, with the idea of possibly inducing a finer decomposition of X, we will force every locally closed set X we encounter to have the shape $X = Y \setminus \mathbf{V}(h)$ where Y is Zariski-closed and $h \in R$. Such locally closed sets will be called *affine cells* in the following. If, in the notation above, X is an affine cell then so is $X_1 = \overline{X \setminus \mathbf{V}(g)}$ but we will need to represent $X \setminus X_1$ as a union of affine cells as well.
This is accomplished as follows: Suppose $H \subset R$ is any finite set such that

$$\langle H \rangle \overset{\mathrm{rad}}{=} \mathbf{I}(X_1).$$

Take $h \in H$ and let $H' := H \setminus \{h\}$. Then for any affine cell X

$$X \setminus \mathbf{V}(H) = [X \setminus \mathbf{V}(h)] \sqcup \left[(X \setminus \mathbf{V}(H')) \cap \mathbf{V}(h)\right]. \tag{5.3}$$

$X \setminus \mathbf{V}(h)$ is now an affine cell and, applying this formula recursively, we can decompose $X \setminus \mathbf{V}(H)$ as a disjoint union of affine cells. The equidimensionality of every output affine cell can therein be insured by again applying the above equidimensional decomposition idea when intersecting with $\mathbf{V}(h)$ on the right hand side of Equation (5.3).

5.2 PRIMITIVE OPERATIONS & DATA STRUCTURES

In the previous section we gave the underlying geometric ideas to give an equidimensional partition of any algebraically closed set into special locally closed sets, called affine cells. Here, we specify precisely which operations we need to be able to perform with any data structure describing an affine cell to bake the ideas from the previous sections into precise algorithms. We then give two different data structures with which all necessary operations can be performed.
First of all, for both data structures, we assume every affine cell X to be explicitly given in terms of equations $f_1, \dots, f_r$ and an inequation h such that $X = \mathbf{V}(f_1, \dots, f_r) \setminus \mathbf{V}(h)$. This equips each affine cell with the distinguished, not necessarily radical, ideal

$$I_X := (\langle f_1, \dots, f_r \rangle : h^\infty)$$

such that $\overline{X} = \mathbf{V}(I_X)$.
The operations that our data structures need to supply are more precisely the following:

(1) Given $f \in R$, compute a data structure representing the affine cell $X \cap V(f)$;

(2) Given $f \in R$, compute a data structure representing the affine cell $X \setminus V(f)$;

(3) Given $f \in R$, decide if $f \in I_X$, or if $f \in \sqrt{I_X}$, both will yield a correct and terminating algorithm;

(4) Compute a generating set of I_X, denoted $\textsc{basis}(X)$.

For our first data structure, we may simply represent an affine cell X by a pair (G, h), where G is a Gröbner basis of I_X, for some monomial ordering, and h a polynomial such that $X = \mathbf{V}(G) \setminus \mathbf{V}(h)$. We denote $X = \mathbf{V}(G; h)$. For a set $G \subseteq R$ and an element $h \in R$, let $\textsc{sat}(G, h)$ denote a Gröbner basis of the saturation ideal $(\langle G \rangle : h^\infty)$ (computed e.g. via Lemma 3.1.2). The operations above then take the following shape:

(1) $\mathbf{V}(G; h) \cap V(f) = \mathbf{V}(\textsc{sat}(G \cup \{f\}, h); h)$;

(2) $\mathbf{V}(G; h) \setminus V(f) = \mathbf{V}(\textsc{sat}(G, f); fh)$;

(3) $f \in I_X$ if and only if the normal form of f w.r.t. G is zero;

(4) $\textsc{basis}(\mathbf{V}(G; h)) = G$.

For our second data structure we rely on some randimization. This randomization relies on intersecting with random linear subspaces of appropriate dimension to reduce to the zero-dimensional case. This idea is used in the computation of geometric resolutions under the

name *lifting fiber* (Giusti, Lecerf, and Salvy, 2001; Lecerf, 2003) and numerical algebraic geometry (e.g. Bates, Hauenstein, Sommese, and Wampler, 2013) wherein these intersections of algebraic sets with random suitable random linear subspaces are known under the name *witness sets*.

Proposition 5.2.1. *Let X be an equidimensional affine cell of dimension d and let $f \in R$. Then, for a generic linear subspace $L \subset \mathbb{A}^n$ of codimension d the following statements hold:*

(1) $f \in \sqrt{I_X}$ *if and only if* $f \in \sqrt{I_{X\cap L}}$.

(2) $I_{X\setminus \mathbf{V}(f)} \subseteq \sqrt{I_X}$ *if and only if* $X \cap L \cap \mathbf{V}(f) = \varnothing$ *where*

$$I_{X\setminus \mathbf{V}(f)} = (I_X : f^\infty).$$

Proof. The first point follows from Lemma 2.2.1.
For the second point note that $I_{X\setminus \mathbf{V}(f)} \subseteq \sqrt{I_X}$ if and only if f intersects X regularly (Lemma 2.3.1) , that is $X \cap \mathbf{V}(f)$ is equidimensional of dimension $d-1$. The intersection of $X \cap \mathbf{V}(f)$ with the codimension d generic space L is empty if and only if the dimension of $X \cap \mathbf{V}(f)$ is less than d by Lemma 2.3.2. This proves the second point. □

Equipped with Proposition 5.2.1 we can now represent an equidimensional affine cell X with a tuple (F, h, W, d) where F and W are subsets of R, $\dim(X) = d$ and $X = \mathbf{V}(F) \setminus \mathbf{V}(h)$. We denote such a tuple by $\mathbf{V}(F, h, W, d)$. Here, W is a Gröbner basis of $I_{X\cap L}$ where L is randomly chosen linear subspace of $\mathbb{K}^n$ of codimension d, we call W a *witness set* of X. Here, $I_{X\cap L}$ is the ideal $(I_X + J : h^\infty)$ where J is given with $\mathbf{V}(J) = L$. The operation of computing W is denoted $\textsc{witness}(F, h, d)$. Now the four operations above take the following shape:

(1) *[Regular intersection]* Given $f \in R$ such that X intersects $\mathbf{V}(f)$ regularly,

$$X \cap \mathbf{V}(f) = \mathbf{V}\big(F'; h, \textsc{witness}(F', h, d-1), d-1\big),$$

with $F' = F \cup \{f\}$;

(1′) *[Purely irregular intersection]* Given $H \subset R$ such that $X \cap \mathbf{V}(H)$ is a union of components of X,

$$X \cap \mathbf{V}(H) = \mathbf{V}\big(F \cup H; h, \textsc{gb}(W \cup H), d\big),$$

where $\textsc{gb}(W \cup H)$ denotes a Gröbner basis of the ideal generated by $W \cup H$;

(2) for $f \in R$, $X \setminus \mathbf{V}(f) = \mathbf{V}(F; hf, \textsc{sat}(W, f), d)$;

(3) $f \in \sqrt{I_X}$ if and only if $1 \in (W : f^\infty)$;

(4) $\textsc{basis}(X) = \textsc{sat}(F, h)$.

In addition we obtain a fifth operation: a probabilistic algorithm to check if $X \cap \mathbf{V}(f)$ is empty or equidimensional of dimension one less than X (or, equivalently if f intersects X regularly). This is given by Algorithm 14.

Algorithm 14 Regular intersection check

Input: An equidimensional affine cell X, an element $f \in R$
Output: TRUE if f intersects X regularly, FALSE otherwise

1: **procedure** isRegular(X, f)
2: $W \leftarrow$ the witness set of X
3: $W' \leftarrow \textsc{gb}(W \cup \{f\})$
4: **return** $1 \in W'$

Example 5.2.1. Let again $R := \mathbb{Q}[x, y, z]$, $X := \mathbf{V}(xy, xz, yz)$ and let $f := x - 1$. f intersects X regularly and we have $\dim(X) = 1$. Let $\ell := x + 2y + 4z + 5$, a witness set for X is then given by a Gröbner basis of the zero-dimensional ideal $W := \langle xy, xz, yz, \ell \rangle$. Furthermore, a Gröbner basis computation in `OSCAR` reveals that

$$-\left(\frac{1}{6}x + 1\right) f = 1 \mod W$$

so that $\textsc{isRegular}(X, f)$ returns true with this choice of W.

5.3 THE ALGORITHMS

With the primitives from Section 5.2 we can now bake the ideas from Section 5.1 into precise pseudocode, see Algorithm 15 below. We again give the algorithm in purely geometric terms, to allow for the flexibility of choosing either datastructure for affine cells from Section 5.2. A slight subtlety is the following: To intersect an affine cell with an algebraic set using the witness set data structure we need to know whether we need to perform a regular or purely irregular intersection. This is always known *a priori* for Algorithm 15. The intersection on Line 4 of SPLIT (given in Algorithm 15) is regular, the intersection on Line 8 is purely irregular. The one on Line 9 is a bit more subtle. Indeed, the decomposition algorithm may produce here a nonequidimensional cell when considering $X \cap \mathbf{V}(g)$. With the notations of this algorithm, the cell $X' = X \cap \mathbf{V}(g)$ is only equidimensional outside of $\mathbf{V}(H)$ (of dimension $\dim(X)$). This nonequidimensional cell will go through only one operation among the four primitives: $X' \setminus \mathbf{V}(h)$ for some $h \in H$. This operation restores equidimensionality. So we can ignore this issue and compute the intersection $X \cap \mathbf{V}(g)$ as a purely improper intersection, pretending that $X \cap \mathbf{V}(g)$ is equidimensional.

Note also that with the witness set data structure, we can replace the if-condition in Line 3 of SPLIT with ISREGULAR(X, f). Only if this is not satisfied do we proceed to compute a Gröbner basis for $I_{X\setminus \mathbf{V}(f)}$.

Algorithm 15 Equidimensional partition

Input: An equidimensional affine cell X, an element $f \in R$
Output: A partition of $X \cap \mathbf{V}(f)$ into equidimensional affine cells

1: **procedure** SPLIT(X, f)
2: $G \leftarrow$ BASIS$(X \setminus \mathbf{V}(f))$
3: **if** $G \subseteq I_X$ *can be replaced by* $G \subseteq \sqrt{I_X}$
4: **return** $\{X \cap \mathbf{V}(f)\}$
5: **else**
6: $g \leftarrow$ any element of $G \setminus I_X$
7: $H \leftarrow$ BASIS$(X \setminus \mathbf{V}(g))$
8: $\mathcal{D} \leftarrow \{X \cap \mathbf{V}(H)\}$
9: **for** $Y \in$ REMOVE$(X \cap \mathbf{V}(g), H)$
10: $\mathcal{D} \leftarrow \mathcal{D} \cup$ SPLIT(Y, f)
11: **return** $\mathcal{D}$

Input: An affine cell X, a finite set $H \subset R$ with $X \setminus \mathbf{V}(H)$ equidimensional
Output: A partition of $X \setminus \mathbf{V}(H)$ into equidimensional affine cells

1: **procedure** REMOVE(X, H)
2: **if** $H = \emptyset$
3: **return** $\emptyset$
4: **else**
5: $h \leftarrow$ any element in H
6: $\mathcal{D} \leftarrow \{X \setminus \mathbf{V}(h)\}$
7: **for** $Y \in$ REMOVE$(X, H \setminus \{h\})$
8: $\mathcal{D} \leftarrow \mathcal{D} \cup$ SPLIT(Y, h)
9: **return** $\mathcal{D}$

Input: A finite set $F \subset R$
Output: An equidimensional partition of $\mathbf{V}(F)$ into affine cells

1: **procedure** EQUIDIM(F)
2: $\mathcal{D} \leftarrow \{\mathbf{V}(\varnothing; 1)\}$ *the full affine space*
3: **for** f in F
4: $\mathcal{D} \leftarrow \bigcup_{X \in \mathcal{D}}$ SPLIT(X, f)
5: **return** $\mathcal{D}$

Example 5.3.1. To illustrate SPLIT we spell out how it behaves on the input $X := \mathbf{V}(xy, zw)$ and $f := xz$. Using the notation of SPLIT we find $G = \{y, w\}$ in Line 2. This is not contained in I_X, so we may choose

$g := y$ in Line 6 of SPLIT. Then we find $H = \{x, zw\}$ in Line 7. Note that $X \setminus \mathbf{V}(zw) = \emptyset$ and so SPLIT returns

$$\begin{aligned} &X \cap \mathbf{V}(H) \text{ and } \textsc{split}(\textsc{remove}(X, H), f) \\ &= \mathbf{V}(zw, x) \text{ and } \textsc{split}(\mathbf{V}(y, zw) \setminus \mathbf{V}(x), xz). \end{aligned}$$

This second call to SPLIT finds $G = \{y, w\}$, again this set is not contained in $I_{\mathbf{V}(y,zw)\setminus\mathbf{V}(x)}$, and so we can choose $g := w$ in Line 6. Then we find $H = \{z\}$ which this time yields

$$\begin{aligned} \textsc{split}(\mathbf{V}(y, zw) \setminus \mathbf{V}(x), xz) =& \mathbf{V}(y, z) \setminus \mathbf{V}(x) \\ &\text{and } \textsc{split}(\mathbf{V}(y, w) \setminus \mathbf{V}(xz), xz) \end{aligned}$$

The last call to split simply finds the empty set and so all in all we have obtained the equidimensional partition

$$\mathbf{V}(xy, zw, xz) = \mathbf{V}(x, zw) \sqcup \mathbf{V}(y, z) \setminus \mathbf{V}(x).$$

5.4 TERMINATION & CORRECTNESS

In this section we prove the termination and correctness of Algorithm 15. We start with the following

Lemma 5.4.1. *Let X be an equidimensional affine cell. Let $f \in R$, let $g \in (I_X : f^\infty)$ and let $I_g = (I_X : g^\infty)$. Let $X_1 = X \cap \mathbf{V}(I_g)$ and $X_2 = (X \cap \mathbf{V}(g)) \setminus \mathbf{V}(I_g)$. Then:*

(i) $X = X_1 \sqcup X_2$;

(ii) $X \cap \mathbf{V}(f) = X_1 \sqcup (X_2 \cap \mathbf{V}(f))$;

(iii) X_1 *is empty or equidimensional with* $\dim X_1 = \dim X$;

(iv) X_2 *is empty or equidimensional with* $\dim X_2 = \dim X$;

Proof. Item (i) follows from the definition of X_1 and X_2, note that g vanishes on $X \setminus \mathbf{V}(I_g)$. Item (ii) follows from the fact that $gf \in I_X$, therefore $f \in \left(\mathbf{I}(X) : I_g^\infty\right) = \mathbf{I}(X_1)$. Item (iii) follows from Proposition 2.2.2. Item (iv) also follows from Proposition 2.2.2: Indeed note that $X_2 = X \setminus \mathbf{V}(I_g) = X \setminus \overline{X_1}$ because $g \in \left(I_X : I_g^\infty\right)$. □

We now prove correctness and termination of SPLIT and REMOVE with a mutual induction. On Line 3, the test $G \subseteq I(X)$ can be replaced by $G \subseteq \sqrt{I_X}$, or any condition which holds when $G \subseteq \sqrt{I_X}$ and does not hold when $G \not\subseteq \sqrt{I_X}$, this does not affect correctness or termination. This is important when we use the witness set data structure from Section 5.2.

Theorem 5.4.1. *For any affine cell X:*

(i) *If* X *is equidimensional, then for any* $f \in R$, SPLIT *terminates on input* X *and* f *and outputs an equidimensional partition of* $X \cap \mathbf{V}(f)$ *into affine cells* Y *with* $I_X \subseteq I_X$.

(ii) *For any finite set* $H \subset R$ *such that* $X \setminus \mathbf{V}(H)$ *is equidimensional, the procedure* REMOVE *terminates on input* X *and* H *and outputs an equidimensional partition of* $X \cap \mathbf{V}(H)$ *into affine cells* Y *with* $I_X \subseteq I_X$.

Proof. We proceed by Noetherian induction on I_X and assume that both statements hold for any affine cell X' with $I_X \subsetneq I_{X'}$.
We begin with SPLIT. Let $f \in R$ and let $I_f = (I_X : f^\infty)$. If $I_f \subseteq I_X$, then Proposition 2.3.1 applies and $X \cap \mathbf{V}(f)$ is equidimensional. So SPLIT(X, f) terminates and is correct in this case.
Assume now that there is some $g \in I_f \setminus I_X$. Let $I_g = (I_X : g^\infty)$. Lemma 5.4.1 applies: an equidimensional decomposition of $X \cap \mathbf{V}(f)$ is given by $X \cap \mathbf{V}(I_g)$ and an equidimensional decomposition of

$$((X \cap \mathbf{V}(g)) \setminus \mathbf{V}(I_g)) \cap \mathbf{V}(f).$$

Moreover $(X \cap \mathbf{V}(g)) \setminus \mathbf{V}(I_g) = X \setminus \mathbf{V}(I_g)$ is equidimensional since X is. Since $g \notin I_X$, we have $I_X \subsetneq I_{X \cap \mathbf{V}(g)}$ so REMOVE$(X \cap \mathbf{V}(g), H)$ (using the notations of SPLIT, where H is a generating set of I_g) is correct and terminates, by induction hypothesis. Moreover, it outputs affine cells Y such that $I_X \subsetneq I_{X \cap \mathbf{V}(g)} \subseteq I_Y$. So the recursive calls SPLIT(Y, f) are correct and terminate.
As for REMOVE, let $H \subset R$ be finite such that $X \setminus \mathbf{V}(H)$ is equidimensional. If $H = \varnothing$, then Item (ii) holds trivially. As for the case $H \neq \varnothing$, let $h \in H$ and $H' = H \setminus \{h\}$. Since $\mathbf{V}(H) = \mathbf{V}(h) \cap \mathbf{V}(H')$, we have

$$X \setminus \mathbf{V}(H) = (X \setminus \mathbf{V}(h)) \sqcup \left((X \setminus \mathbf{V}(H')) \cap \mathbf{V}(h)\right). \tag{5.4}$$

The set $X \setminus \mathbf{V}(h)$ and $X \setminus \mathbf{V}(H')$ are locally closed subset of X and so equidimensional (or empty). By induction on the cardinal of H, we assume that REMOVE(X, H') is a partition of $X \setminus \mathbf{V}(H')$ into equidimensional affine cells, and that every cell Y of this partition satisfies $I_X \subseteq I_Y$. By Item (i), the calls SPLIT(Y, h) terminates and yield a partition of $(X \setminus \mathbf{V}(H')) \cap \mathbf{V}(h)$ into cells Y with $I_X \subseteq I_Y$. Moreover the affine cell $Y = X \setminus \mathbf{V}(h)$ also satisfies $I_X \subseteq I_Y$. By Equation (5.4), REMOVE(X, H) terminates too and is a partition of $X \setminus \mathbf{V}(H)$ into cells Y with $I_X \subset I_Y$. □

This implies immediately

Corollary 5.4.1. EQUIDIM *is correct and terminates.*

5.5 BENCHMARKS

The benchmarks for this chapter we recorded using an implementation of the author of Algorithm 15 using the computer algebra system `OSCAR` (Decker et al., 2025), written in `Julia`. The source code of this implementation is available at

https://github.com/RafaelDavidMohr/Decomp.jl

All necessary Gröbner basis computations are done using `msolve` for which `OSCAR` offers an interface.
The benchmarks can be found in Table A.3. All benchmarks for this chapter were recorded with $\mathbb{K} = \mathbb{F}_{65521}$. We computed all examples on a single Intel Xeon Gold 6244 CPU @ 3.60GHz with a limit of 200G memory, except for the `Magma` computations which were done on a single core of an Intel Xeon E5-2690 @ 2.90GHz. Again we refer to Appendix A.1 for brief explanations of the polynomial systems on which we tested our implementation and to the beginning of Appendix A.2 for an explanation of the symbols used in Table A.3.
To obtain the timings in Table A.3, we almost exclusively used the data structure for affine cell based on witness sets, described in Section 5.2. In the second column of Table A.3, we additionally provide the number of affine cells that Algorithm 15 decomposed the respective system into.

5.5.1 *Discussion of Experimental Results*

Our algorithm, i.e. Algorithm 15, seems to behave best in comparison to the other implementations when the input system is dense in the sense that each of the input equations of the system in question involves most, or all, of the variables. This is the case for Cyclic(8), the class of the PS(•) systems, the class of the Sing(•) systems, the class of the SOS(•, •) systems and the Steiner polynomial system.
On certain polynomial systems, where each input equation involves only a small subset of the variables, we were able to improve our timings by foregoing the witness set based data structure and instead using the first data structure described in Section 5.2. The improvement we thusly obtained can be explained by the fact that intersecting very sparse systems with random hyperplanes can "destroy their sparsity" and make certain Gröbner basis computations much harder. This was the case for the example Leykin-1: Here running the deterministic version improved our timing from 15.2 seconds to 2.6 seconds.
The Gonnet and dgp6 polynomial systems demonstrate that Algorithm 15 is highly sensitive to the ordering of the input equations: By default we ran our implementation by iterating over the input equations degree by degree in Algorithm 15. With this ordering, Algorithm 15 did not terminate within several hours of computation. When we changed this ordering on these two examples and sorted the input equations instead by length of support, Algorithm 15 terminated in less than one second on these two examples. Algorithm 15's practical efficiency depends highly on the difficulty of the intermediate polynomial systems which it encounters, these in turn depend on the order of the input equations. On the Gonnet polynomial system, the new ordering resulted in only a subset of the variables being involved

in the first few intermediate systems, thus making Gröbner basis for them more tractable. On the dgp6 example the new ordering resulted in an intermediate polynomial system consisting of monomials and binomials which Algorithm 15 decomposes very finely, making the treatment of the remaining equations substantially easier.
The system sys2874 can be attacked by both changing the order of the input equations to be ordered by length of support and by using the deterministic version of Algorithm 15: Doing this, the timing improved by several orders of magnitude to 0.26 seconds.
We also remark that `OSCAR`'s timings improved significantly on the examples sys2449, sys2297 and Leykin-1 (each to less than one second) if one decomposes the radicals of these systems instead of the systems themselves.
For the examples KdV and sys2882 we seem to be bottlenecked by very difficult Gröbner basis computations and less by the inherent structure of Algorithm 15. Informal experiments where we tried to compute just a Gröbner basis for these systems using msolve suggest that even this is a highly non-trivial computation. For these two systems, techniques involving regular chains seem to be vastly superior over anything that involves Gröbner basis computations.
All in all, these experiments illustrate that on a wide range of examples, Algorithm 15 performs on average better than state-of-the-art implementations and can tackle some problems which were previously unreachable. We refer to the next chapter for an algorithm which, in contrast to Algorithm 13 and Algorithm 15 does not use an incremental structure and thus potentially circumvents issues stemming from the incremental structure of Algorithm 13 and Algorithm 15.

6

COMPUTING A KALKBRENER DECOMPOSITION OF AN ALGEBRAIC SET

We reuse the notation from Chapter 5. Again let $F := (f_1, \ldots, f_r) \subset R$ be a finite sequence. For the purposes of this chapter we denote by $\mathrm{Irred}(X)$ the set of irreducible components of an algebraic set $X \subset \mathbb{A}^n$. Let us first define

Definition 6.0.1 (Closure Partition)**.** Let $X \subset \mathbb{A}^n$ be a locally closed set. For another locally closed set $Y \subset \mathbb{A}^n$ we write

$$X \overset{\text{clo}}{=} Y$$

if $\overline{X} = \overline{Y}$. A *closure partition* of X is a finite set of pairwise disjoint locally closed sets $\mathcal{D}$ such that

$$X \overset{\text{clo}}{=} \bigcup_{Y \in \mathcal{D}} Y$$

and such that in addition $\mathrm{Irred}(\overline{Y_1}) \cap \mathrm{Irred}(\overline{Y_2}) = \emptyset$ for $Y_1 \neq Y_2 \in \mathcal{D}$ and

$$\mathrm{Irred}(\overline{X}) = \bigcup_{Y \in \mathcal{D}} \mathrm{Irred}(\overline{Y}).$$

Given our sequence $F := (f_1, \ldots, f_r)$ we will compute

Definition 6.0.2 (Irredundant Kalkbrener Partition)**.** An *irredundant Kalkbrener partition* of a locally closed set $X \subset \mathbb{A}^n$ is a finite set of locally closed sets $\mathcal{D}$ such that $\mathcal{D}$ forms a closure partition of X and such that each $Y \in \mathcal{D}$ is equidimensional.

The requirement of our output to be a closure partition can be understood as a strong minimality condition: It ensures that scratching any of the $Y \in \mathcal{D}$, or even removing an irreducible component from one of the $Y \in \mathcal{D}$ destroys the property $\mathbf{V}(F) \overset{\text{clo}}{=} \bigcup_{Y \in \mathcal{D}} Y$.

Remark 6.0.1. The name "Kalkbrener partition" comes from the algorithm presented by Kalkbrener (1993) which computes a similar partition using triangular sets.

Example 6.0.1. With $R := \mathbb{Q}[x, y]$, a Kalkbrener partition of $X := \mathbf{V}(xy, xz)$ into affine cells is given by $\{\mathbf{V}(x), \mathbf{V}(y, z) \setminus \mathbf{V}(x)\}$.

6.1 THE BASIC PRINCIPLE

While we will reuse the concept of an affine cell from Chapter 5 we first describe the geometric principles of the algorithm presented here without concern for this particular data structure.

The theoretical underpinning of this chapter is Theorem 2.3.1. We start by making this theorem a bit more computationally explicit:

Proposition 6.1.1. *Let* $f_1, \ldots, f_c \in R$ *and let* $h \in R$. *Suppose that for every* $g \in R$ *with* $gf_i \in \langle f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_c \rangle$ *for some* i *we have* $g \in (I : h^\infty)$. *Then* $(I/I^2)_h$ *is free of rank* c *over* $(R/I)_h$, *freely generated by the images of* $f_1, \ldots, f_c$. *In other words, at every point* $p \in \mathbf{V}(f_1, \ldots, f_c) \setminus \mathbf{V}(h)$, $f_1, \ldots, f_c$ *forms a local regular sequence.*

Proof. Suppose that in R_h we have a relation

$$\sum_i g_i f_i = \sum_i \sum_{j \leqslant i} g_{ij} f_j f_i$$

i.e. $\sum_i g_i f_i = 0$ in $(I/I^2)_h$. Then we can write

$$\sum_i \left(g_i - \sum_{j \leqslant i} g_{ij} f_j \right) f_i = 0.$$

By assumption we then have

$$(g_i - \sum_{j \leqslant i} g_{ij} f_j) h^k \in I$$

for every i and some $k \in \mathbb{N}$. This implies $g_i \in I_h$ for every i or in other words $g_i = 0$ in $(R/I)_h$ for every i. This proves that $(I/I^2)_h$ is freely generated by the images of $f_1, \ldots, f_c$ over $(R/I)_h$. □

Based on this proposition, we will now follow a similar strategy as in Chapter 5: With input $f_1, \ldots, f_r$, let $I := \langle f_1, \ldots, f_r \rangle$. We will start by identifying a polynomial $g \in R$ with $gf_i \in \langle f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_r \rangle$ and $g \notin I$. This means that $f_1, \ldots, f_r$ is not a local complete intersection. We then compute a closure partition of $X := \mathbf{V}(F)$ into two locally closed sets X_1 and X_2 which will satisfy

$$\overline{X_1} = \bigcup_{\substack{Y \in \mathrm{Irred}(X) \\ g \notin \mathbf{I}(Y)}} Y,$$

$$\overline{X_2} = \bigcup_{\substack{Y \in \mathrm{Irred}(X) \\ g \in \mathbf{I}(Y)}} Y.$$

By Proposition 2.2.2, for X_1, we may simply choose

$$X_1 := \mathbf{V}(f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_r) \setminus \mathbf{V}(g)$$

which is an affine cell and can thus be represented computationally by a suitable modification of first data structure presented in Section 5.2. This modification will be detailed in Section 6.2. Note that we may no longer use the witness set data structure from Section 5.2, since the affine cells we will be working with are no longer necessarily

equidimensional, essentially because we are no longer following the strategy of decomposing X equation-by-equation.
It remains to represent X_2 via a closure partition into affine cells. To compute a suitable X_2 we define for an affine cell Y and a polynomial $p \in R$

$$\operatorname{hull}(Y, p) := Y \setminus \left[\overline{Y \setminus \mathbf{V}(p)}\right].$$

Note that the closure of $\operatorname{hull}(Y, p)$ consists precisely of the components of the closure of Y on which p vanishes, by Proposition 2.2.2 and Proposition 2.2.3.
Suppose now that we have computed some finite generating set $P \subset R$ for I_{X_1}. Choose $p \in P$ and let $P' := P \setminus \{p\}$. Then we compute

$$\begin{aligned} X_2 := \operatorname{hull}(X, g) &= X \setminus \mathbf{V}(P) \\ &\overset{\text{clo}}{=} X \setminus \mathbf{V}(p) \sqcup \operatorname{hull}(X \setminus \mathbf{V}(P'), p). \end{aligned}$$

So we are decomposing $X \setminus \mathbf{V}(P)$ into the union of the components of X where p does not vanish and the union of the components where p vanishes but one of the elements in P' does not. This formula gives us now a recursive algorithm to compute a closure partition of the chosen X_2, which will terminate, similarly to the procedure REMOVE in Algorithm 15 because R is Noetherian.

Example 6.1.1. To briefly illustrate this recursive strategy for computing $\operatorname{hull}(X, g)$ let $X = \mathbf{V}(xy, xz, yz)$ and $g = y$. First note that $(I_X : g^\infty) = \langle x, z\rangle$. Therefore

$$\begin{aligned} \operatorname{hull}(X, g) &= X \setminus \mathbf{V}(x, z) \\ &\overset{\text{clo}}{=} X \setminus \mathbf{V}(x) \sqcup \operatorname{hull}(X \setminus \mathbf{V}(z), x) \\ &= \mathbf{V}(y, z) \setminus \mathbf{V}(x) \sqcup \operatorname{hull}(\mathbf{V}(x, y), x) \\ &= \mathbf{V}(y, z) \setminus \mathbf{V}(x) \sqcup \mathbf{V}(x, y). \end{aligned}$$

Once this recursive algorithm terminates with a list of affine cells $Y_1, \ldots, Y_s$ of the form

$$Y_i = \mathbf{V}(f_1, \ldots, f_r) \setminus \mathbf{V}(h_i)$$

we will replace $f_1, \ldots, f_r$ by $f_1, \ldots, f_r, g$ in each of these affine cells and recursively go through the same process with X_1 and all the Y_i.

Remark 6.1.1. Perhaps a more obvious idea than the algorithmic strategy lined out above is to, upon finding g as above, split X into

$$X = [X \setminus \mathbf{V}(g)] \sqcup [X \cap \mathbf{V}(g)].$$

Note that this is not a closure partition of X, we introduce components that X did not have.
For example, taking $X = \mathbf{V}(xy, xz) \subset \mathbb{A}^3$ as above, we have for $g := y$

$$g \cdot xz \in \langle xy \rangle$$

and if we factor

$$X = \mathbf{V}(xy) \setminus \mathbf{V}(x) \sqcup \mathbf{V}(xy, xz, y)$$

then the algebraic set $\mathbf{V}(xy, xz, y) = \mathbf{V}(y, xz)$ has the irreducible component $\mathbf{V}(x, y)$ which X did not have.
One may accept this, because it yields a simpler algorithm, but we found experimentally that this behaves worse than decomposing X into a closure partition as above. The superflous components introduced cause an exponential blow up in the number of output affine cells produced on certain examples.

6.2 THE DATA STRUCTURE

Note that, given an affine cell $X = \mathbf{V}(f_1, \ldots, f_r) \setminus \mathbf{V}(h)$, we now require an algorithm that returns, if it exists, a polynomial $g \in R$ such that $gf_i \in \langle f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_r \rangle$ and $g \notin I_X$. This means we have to compute syzygies of the sequence $f_1, \ldots, f_r$. A natural choice for this is to use sGB computations similarly to how Algorithm 13 uses syzygies of some input sequence. With this in mind, we will now represent the affine cell X by a data structure containing the following fields:

(1) A sGB tree $\mathcal{T}_X$ whose nodes correspond to the polynomials $f_1, \ldots, f_r$ in order.

(2) The polynomial $h \in R$.

(3) A Gröbner basis G_X of I_X w.r.t. some monomial order $\prec$.

(4) An upper bound c_X on the maximal codimension of the components of X.

Knowledge of G_X and h permits us to add inequations (i.e. to replace a given X by $X \setminus \mathbf{V}(p)$ for some polynomial p) similarly to how this was done in Section 5.2, by replacing the Gröbner basis G_X underlying X with $\text{SAT}(G_X, p)$ (again, computed e.g. via Lemma 3.1.2).
To find syzygy cofactors g as above we can now slightly adapt the GETSYZYGY algorithm from Section 4.3.2 using the notation from this section, this yields Algorithm 16.
Let us explain a few differences to the situation in Chapter 4. Now, the sGB tree $\mathcal{T}$ just encodes a sequence of polynomials, so the setting from Section 3.6 applies. We will just use $\mathcal{T}$ to conveniently insert and remove elements from our sequences while still correctly managing our sGB computations. Since we no longer rely on equation-by-equation processing either, we can use any module order we like other than position-over-term. In practice, we restrict to homogeneous input $f_1, \ldots, f_r$ and choose the *degree-position-over-term* order:

Definition 6.2.1 (Degree Position Over Term order)**.** Using the notation from Section 3.6, the *degree-position-over-term* (DPOT) order on R^r is

Algorithm 16 Adapted version of GETSYZYGY

Input: An affine cell X with sGB tree $\mathcal{T}_X$
Output: A syzygy cofactor g of the sequence underlying $\mathcal{T}_X$ not contained in G_X

```
1: procedure GetSyzygy2(X)
2:     while there is a pair in P, the underlying pairset of T_X
3:         ProcessSPair(T_X)
4:         while S ≠ ∅
5:             pick and remove some γ in S
6:             if There exists an entry g of γ with Reduce(g, G_x, ≺) ≠ 0
7:                 return g and the node ν corresponding to the entry g
8:     return (0, 0)
```

defined as $u\epsilon_i \prec_{pot} v\epsilon_j$ iff $\deg(uf_i) < \deg(vf_j)$ or $\deg(uf_i) = \deg(vf_j)$ and $u\epsilon_i \prec_{pot} v\epsilon_j$.

Choosing this module order has the advantage of giving our algorithm a non-incremental structure but still preserving Proposition 3.6.1: If $f_1, \ldots, f_r$ are homogeneous and form a regular sequence already then Algorithm 16 returns $(0, 0)$ and we can simply return $\mathbf{V}(f_1, \ldots, f_r)$ as our desired irredundant Kalkbrener partition. This fact can be proven exactly as Proposition 3.6.1 where the same is proven for the POT order, see also Corollary 7.2 in Eder and Faugère (2017).
Being able to choose a different module order also has the effect of simplifying the ProcessSpair procedure in Algorithm 16: Compared to the ProcessSpair procedure in Algorithm 12 we just process all S-pairs left from $\mathcal{T}$ without restricting to the ones below a given node. In addition to the operations specified in Section 4.3.1 provided by $\mathcal{T}$ we will also need the option to remove a node from $\mathcal{T}$. This is provided by Algorithm 17.

Algorithm 17 Removing a node from a sGB tree

Input: A sGB tree $\mathcal{T}$, a node ν
Output: The sGB tree $\mathcal{T}$ with ν removed, with G, P and S updated accordingly

```
1: procedure RemoveNode(T, ν)
2:     Delete all elements in signature ⪰_T ν from G
3:     Delete all elements in signature ⪰_T ν from P
4:     Delete all elements in signature ⪰_T ν from S
5:     for all nodes μ ⪰_T ν in T
6:         ε ←the unit vector correspondig to μ
7:         G ← G ∪ {ε}
8:         P ← P ∪ {(ε, β) | β ∈ G and (ε, β) forms a regular S-pair}
9:     Delete the node ν from T
```

In line with Remark 4.3.1, we consider G, P and S as part of the inherent state of $\mathcal{T}$. They then need to be adjusted accordingly in Algorithm 17 when we remove a node ν from $\mathcal{T}$. Essentially, all elements of G, P and S which have signature $\succeq_{\mathcal{T}}$ than ν have to be deleted.

6.3 THE ALGORITHMS

Combining Section 6.1 and Section 6.2 now allows us to give exact algorithms for computing an irredundant Kalkbrener partition of $\mathbf{V}(F)$ where F is a given input sequence, see Algorithm 18.
Note again that all necessary operations on affine cells in the procedures Hull and CloRemove are covered by the operations Item (1) through Item (4) in Section 5.2 which our data structure for affine cells provides using Gröbner bases, again as in Section 5.2.

Theorem 6.3.1. *The procedures* Hull *and* CloRemove *terminate and are correct in that they satisfy their output specifications.*

Proof. The correctness and termination of Hull are immediate once the correctness and termination of CloRemove is established. The correctness of CloRemove is proven once we can prove that

$$X \setminus \mathbf{V}(p) \sqcup \operatorname{hull}(X \setminus \mathbf{V}(P'), p)$$

gives a closure partition of $X \setminus \mathbf{V}(P)$. Denote $X_p := X \setminus \mathbf{V}(p)$ and $X_{P'} := \operatorname{hull}(X \setminus \mathbf{V}(P'), p)$. For this, note that

$$X_{P'} = \left[X \setminus \mathbf{V}(P')\right] \setminus \overline{X_p}.$$

Hence we have $X \setminus \mathbf{V}(P) \overset{\text{clo}}{=} X_p \sqcup X_{P'}$ and X_p and $X_{P'}$ do not share any irreducible components by Proposition 2.2.2 and Proposition 2.2.3.
Now we just have to prove that CloRemove terminates. Note that Line 11 is only called if $I_X \subsetneq I_Y$. Hence, by Noetherian induction, during any run of CloRemove, we can only call Line 11 finitely many times. But then termination follows, since the input set P is finite. □

Corollary 6.3.1. *The procedure* KalkPart *terminates and is correct in that it satisfies its output specification.*

Proof. To establish correctness, let us first look at Line 3. By definition, c_X is an upper bound on the maximum codimension of all components of X. So, if $\operatorname{codim}(X) = c_X$, then X is equidimensional of codimension c. To fully establish the correctness of this line, we now have to show, that, in Line 11, c_{X_1} is really an upper bound on the maximum codimension of all components of X_1. The closures of the components of X_1 are a subset of the closures of the components of X, so the maximum codimension of all components of X_1 is certainly bounded by c_X. With r defined as in Line 10, the codimension of all components of X_1 is bounded by r by Krull's principal ideal theorem (Theorem 10.2 in

Algorithm 18 Computing an irredundant Kalkbrener partition

Input: An affine cell X
Output: An irredundant Kalkbrener partition of X

procedure KALKPART(X)
  if $\operatorname{codim}(X) = c_X$, defined as in Section 6.2
    return $\{X\}$
  $g, \nu \leftarrow$GETSYZYGY2(X)
  if $g = 0$
    return $\{X\}$
  else
    $X_1 \leftarrow X \setminus \mathbf{V}(g)$
    REMOVENODE($\mathcal{T}_{X_1}, \nu$)
    $r \leftarrow$number of nodes in $\mathcal{T}_{X_1}$
    $c_{X_1} \leftarrow \min\{c_X, r\}$
    $\mathcal{X}_2 \leftarrow$HULL(X, g)
    for X_2 in $\mathcal{X}_2$
      INSERTNODE($\mathcal{T}_{X_2}$, g, any position)
    return KALKPART(X_1) $\cup \bigcup_{X_2 \in \mathcal{X}_2}$ KALKPART(X_2)

Input: An affine cell X, an element $g \in R$
Output: A closure partition of $\operatorname{hull}(X, g)$

procedure HULL(X, g)
  $H \leftarrow \operatorname{sat}(G_X, g)$
  return CLOREMOVE(X, H)

Input: An affine cell X, a finite set $P \subset R$
Output: A closure partition of $X \setminus \mathbf{V}(P)$

procedure CLOREMOVE(X, P)
  $p \leftarrow$any element in P
  $P' \leftarrow P \setminus \{p\}$
  $\mathcal{D} \leftarrow$CLOREMOVE(X, P')
  $Y \leftarrow X \setminus \mathbf{V}(p)$
  if $I_X = I_Y$
    return Y
  else if $Y = \emptyset$
    return $\mathcal{D}$
  else
    return $\{Y\} \cup \bigcup_{Z \in \mathcal{D}}$ CLOREMOVE(Z, G_Y)

Eisenbud, 1995). This shows that c_{X_1} is really an upper bound on the codimension of all components of X_1.
Next, in line Line 6, the equidimensionality of X follows from Theorem 2.3.1 and Proposition 6.1.1.
Finally note that $\{X_1\} \cup \mathcal{X}_2$, as defined in the algorithm, form a closure partition of X by Theorem 6.3.1. This is not affected by Line 14 because we have $g \in \sqrt{X_2}$ for all $X_2 \in \mathcal{X}_2$. This finally establishes the correctness of the algorithm.
For termination, note that I_X is strictly contained in the ideal I_{X_1} and the ideals I_{X_2} for $X_2 \in \mathcal{X}_2$ if g does not regularly intersect X_1: In this case X_1 and every $X_2 \in \mathcal{X}_2$ has as irreducible components a strict subset of those of X by Proposition 2.2.2 and Lemma 2.3.1. So by Noetherian induction, after a finite number of steps, every g found in Line 4 regularly intersects X. In this case, $\mathcal{X}_2 = \emptyset$ and $I_X = I_{X_1}$ but the sequence underlying the sGB tree of X_1 is strictly shorter. This establishes the termination of KalkPart. □

Remark 6.3.1. Compared to the procedure GetSyzygy (given in Algorithm 13 in Chapter 4), the procedure GetSyzygy2 needs to potentially look at every entry of a given syzygy, not just the non-zero entry of highest index: In line with Proposition 6.1.1, if, on input $f_1, \dots, f_r$, there is a relation

$$\sum_{i=1}^{r} g_i f_i$$

then we need to check for every i whether $g_i \in \langle f_1, \dots, f_{i-1}, f_{i+1}, \dots, f_r \rangle$ where as, in the context of Chapter 4, we just work with g_r. In Algorithm 13, it is feasible to track enough data about the underlying module representation of every occuring element in the sGB computation to obtain g_r directly, but it is not practically feasible to track the *entire* module representation which is needed to obtain every g_i above directly. We may instead use the algorithm given in Sun and Wang (2011): Here, the module representation (or any entry of it) of any element appearing in a signature-based Gröbner basis computation can be constructed *a posteriori* from just its signature, this is more practical than tracking full module computations throughout the computation.

6.4 BENCHMARKS

The benchmarks for this chapter we recorded using an implementation of the author of Algorithm 18 as part of the `Julia`-package `AlgebraicSolving.jl`. This implementation contains an implementation of Algorithm 10 to furnish the syzygy computations needed for Algorithm 16.
This latter implementation is more optimized and considerably faster than the implementation used for the benchmarks in Chapter 4, incorporating ideas by Lairez (2024) to speed up our rewrite checks

following Definition 3.6.8: Each element α in the computed Gröbner basis G is organized in a tree. The children of α are the results of reducing some S-pairs of the form $a\alpha - b\beta$ with $\mathfrak{s}(b\beta) \prec \mathfrak{s}(a\alpha)$. To check if some element of the form $c\gamma$ with $\gamma \in G$ is rewriteable by the Singular criterion, we then traverse this tree checking at each level if there is an element whose signature divides $\mathfrak{s}(c\gamma)$. If we arrive at γ then $c\gamma$ is not rewriteable, otherwise it is rewriteable. This does not match the rewriteability check of Definition 3.6.8 but still ensures that we choose at most one element per signature to reduce, so correctness and termination are unaffected.
We use `msolve` to compute the required Gröbner bases G_X for the appearing affine cells X, as in Section 6.2. This is done to furnish a comparison between Algorithm 15 and Algorithm 18 independent of implementational considerations. The source code of `AlgebraicSolving.jl` is available at

`https://github.com/algebraic-solving/AlgebraicSolving.jl`.

The benchmarks can be found in Table A.4. In it, we compare the runtimes of Algorithm 15 (implemented as in Chapter 5) and Algorithm 18. The examples used are not homogeneous, so we homogenized our input $f_1, \dots, f_r$ w.r.t. a new variable z, yielding $f_1^{\text{hom}}, \dots, f_r^{\text{hom}}$ and ran Algorithm 18 on $X = \mathbf{V}(f_1^{\text{hom}}, \dots, f_r^{\text{hom}}) \setminus \mathbf{V}(z)$. After dehomogenizing at the end, we obtain a Kalkbrener partition of $\mathbf{V}(f_1, \dots, f_r)$.
All benchmarks for this chapter were recorded with $\mathbb{K} = \mathbb{F}_p$ where p is a randomly chosen prime with less than 32 bits. We computed all examples on a single Intel Xeon Gold 6244 CPU @ 3.60GHz with a limit of 200G memory. Again we refer to Appendix A.1 for brief explanations of the polynomial systems used to create these benchmarks.

6.4.1 *Discussion of Experimental Results*

Despite its completely irredundant output, with the exception of the Cyclic(8) example, Algorithm 18 performs usually better than Algorithm 15, or at least not much worse.
On the examples, where Algorithm 18 performs significantly better than Algorithm 15, the behaviour indicated in the introduction of this thesis is exhibited: Due to its incremental nature Algorithm 15 has to treat difficult "intermediate" systems, i.e. systems where only some of the input equations have been processed. Concretely, for example, Algorithm 15 gets stuck on SOS(7,5) performing certain saturations on the ideal defined by the first five (out of seven) equations of the system. On examples like this it therefore looks to be a big advantage to be able to work with the entire system all at once, as in Algorithm 18.
On Cyclic(8) the bottleneck for Algorithm 18 lies entirely in the syzygy computations performed by Algorithm 16. For this example a significant number of syzygies are computed and then verified to already

lie in the ideal underlying the affine cell in question, they therefore cannot be used by Algorithm 18 to further decompose the affine cell. Despite it usually bringing a performance boost, processing all defining equations at the same time can also be a bottleneck: In particular, with the algorithm as presented here, at the start of the computation, a Gröbner basis of the ideal defined by the input equations is required. This slows down Algorithm 18 compared to Algorithm 15 in those cases where Algorithm 15 performs a decomposition after only a small handful of the input equations have been considered, this is the case for the examples sys2353 and sys2161. We refer to Chapter 9 for ideas how this issue could potentially be circumvented.

7

GRÖBNER BASIS ALGORITHMS FOR COMPUTING GENERIC FIBERS

For this chapter we slightly change our notation: We let $R := \mathbb{K}[\mathbf{z}, \mathbf{x}]$ be a polynomial ring in two finite sets of variables and suppose that $I \subset R$ is an ideal with MIS $\mathbf{z}$ (Definition 2.4.1) so that

$$\operatorname{gen}(I, \mathbf{z}) := I\mathbb{K}(\mathbf{z})[\mathbf{x}]$$

is a generic fiber of I. Note that this implies in particular that $\operatorname{gen}(I, \mathbf{z})$ is a zero-dimensional ideal. Our goal in this chapter is to show how to compute a Gröbner basis G of $\operatorname{gen}(I, \mathbf{z})$ w.r.t. a given monomial order on $\operatorname{Mon}(\mathbf{x})$. We will give two algorithms based on *Hensel lifting* to compute such a Gröbner basis. Roughly, if $\mathbf{z} = \{z\}$ is a single variable, the idea is to pick a random $a \in \mathbb{K}$ and to compute a Gröbner basis G_0 of the image of I in the polynomial ring $\mathbb{K}[z, \mathbf{x}]/\langle z - a\rangle$ for the desired monomial order. Then we lift G_0, in a unique and well-defined way, to the ring $\mathbb{K}[z, \mathbf{x}]/\langle z - a\rangle^2$. Proceeding further like this, i.e. lifting modulo higher and higher powers of $z - a$, we recover the coefficients of the elements in G (which lie in $\mathbb{K}(\mathbf{z})$) as truncated power series. If this is done up to a sufficiently large power of $z - a$, we can then recover G, using Padé approximation.
The primary contribution of this chapter is based on combining this Hensel lifting idea with the FGLM algorithm (Section 3.5). We also show how to use the F_4 algorithm (Section 3.4) with this idea, utilizing a version of Gröbner tracers (Section 3.4.1) adapted to Hensel lifting to obtain an efficient lifting step.
This chapter is based on the results given in Berthomieu and Mohr (2024).

7.1 POINTS OF GOOD SPECIALIZATION

We start by giving the necessary theory in order to show that our lifting steps give correct algorithms. We first introduce some convenient notation.

Definition 7.1.1 (Notation 1)**.** We denote for a monomial $u \in \operatorname{Mon}(\mathbf{z})$

$$m_u := \langle v \in \operatorname{Mon}(\mathbf{z}) \mid v \succ_{\mathrm{drl}} u\rangle \text{ and } I_u := I + m_u,$$

$m := m_1 = \langle \mathbf{z}\rangle$, as well as $\operatorname{next}(u) = \min\{v \in \operatorname{Mon}(\mathbf{z}) \mid v \succ_{\mathrm{drl}} u\}$.

Definition 7.1.2 (Notation 2)**.** Let $g \in \mathbb{K}[\mathbf{z}]_m[\mathbf{x}]$. Write

$$g = \sum_{w \in \operatorname{Mon}(\mathbf{x})} \frac{p_w}{q_w} w$$

with $p_w, q_w \in \mathbb{K}[\mathbf{z}]$ and $q_w(0) \neq 0$ for all $w \in \mathrm{Mon}(\mathbf{x})$ whenever $p_w \neq 0$. Then, each p_w/q_w can be written as a formal power series

$$\frac{p_w}{q_w} = \sum_{v \in \mathrm{Mon}(\mathbf{z})} r_{w,v} v \in \mathbb{K}[\![\mathbf{z}]\!]$$

and for a monomial $u \in \mathrm{Mon}(\mathbf{z})$ we denote

$$g_u := \sum_{w \in \mathrm{Mon}(\mathbf{x})} \sum_{\substack{v \in \mathrm{Mon}(\mathbf{z}) \\ v \preceq_{\mathrm{drl}} u}} r_{w,v} vw = g \bmod \mathfrak{m}_u$$

For a set $G \subset \mathbb{K}[\mathbf{z}]_{\mathfrak{m}}[\mathbf{x}]$ we define $G_u := \{g_u \mid g \in G\}$.

Throughout this section fix $G \subset \mathbb{K}(\mathbf{z})[\mathbf{x}]$ to be the reduced Gröbner basis of $\mathrm{gen}(I, \mathbf{z})$ w.r.t. a monomial order $\prec_{\mathbf{x}}$ on $\mathrm{Mon}(\mathbf{x})$. Our algorithms will work under the assumption that $G \subset \mathbb{K}[\mathbf{z}]_{\mathfrak{m}}[\mathbf{x}]$ and that given the set G_u we can lift G_u uniquely to $G_{\mathrm{next}(u)}$. In fact the condition $G \subset \mathbb{K}[\mathbf{z}]_{\mathfrak{m}}[\mathbf{x}]$ turns out to be sufficient. We capture this in a definition:

Definition 7.1.3 (Point of Good Specialization)**.** We say that $\mathfrak{m}$ is a *point of good specialization* (for $\prec_{\mathbf{x}}$) if $G \subset \mathbb{K}[\mathbf{z}]_{\mathfrak{m}}[\mathbf{x}]$.

Example 7.1.1. Consider the ideal $I \subset \mathbb{Q}[z, x_1, x_2, x_3]$ defined by the Cyclic 4 polynomial system given by

$$\begin{aligned} z + x_1 + x_2 + x_3, \\ zx_1 + x_1x_2 + x_2x_3 + x_3z, \\ zx_1x_2 + x_1x_2x_3 + x_2x_3z + x_3zx_1, \\ zx_1x_2x_3 - 1. \end{aligned}$$

The Gröbner basis $G \subset \mathbb{K}(z)[x_1, x_2, x_3]$ of $\mathrm{gen}(I, z)$ w.r.t. to the order $\prec_{\mathbf{x}} = \prec_{\mathrm{lex}}$ (the lexicographic monomial order, Definition 3.1.3) is given by

$$G = \{x_3^2 - \frac{1}{z^2}, x_2 + z, x_1 + x_3\}.$$

Hence $\mathfrak{m}$ is not a point of good specialization: The first element listed in G does not lie in $\mathbb{K}[z]_{\mathfrak{m}}[x_1, x_2, x_3]$.

Remark 7.1.1. By definition, being a point of good specialization is a Zariski open condition, so that, if $\mathbb{K}$ is infinite, it is ensured with probability 1 after replacing each $z_i \in \mathbf{z}$ by $z_i - a_i$ for randomly chosen $a_i \in \mathbb{K}$. In Remark 7.3.5 we point out a situation in which an upper bound for the probability that $\mathfrak{m}$ is a point of good specialization can be given intrinsically in terms of I if $\mathbb{K}$ is finite.

We now somewhat extend Theorem 3.4.1 to our situation. Throughout this chapter, for some polynomial ideal J and some monomial order $\prec$ on the underlying polynomial ring of J, $S_{J,\prec}$ denotes the $\prec$-staircase of J, as defined in Definition 3.1.6. Now we first show

Theorem 7.1.1. *If the ideal $\mathfrak{m}$ is a point of good specialization, then the $\mathbb{K}[\mathbf{z}]_{\mathfrak{m}}$-module $\mathbb{K}[\mathbf{z}]_{\mathfrak{m}}[\mathbf{x}]/I$ is free of finite rank.*

Proof. Write $A := \mathbb{K}[\mathbf{z}]_{\mathfrak{m}}$, $K = \mathbb{K}(\mathbf{z})$ for the field of fractions of A and $F := A[\mathbf{x}]/I$.
Suppose that $\mathfrak{m}$ is a point of good specialization so that $G \subset A[\mathbf{x}]$. Let $S := S_{\mathrm{gen}(I,\mathbf{z}),\prec_x}$, note that S is finite since $\mathrm{gen}(I,\mathbf{z})$ is 0-dimensional. We first show that S generates F as an A-module. Let $u \in \mathrm{Mon}(\mathbf{x})$ with $u \notin S$ so that $u \in \mathrm{lm}_{\prec_x}(\mathrm{gen}(I,\mathbf{z}))$. Then there exists $g \in G$ and $v \in \mathrm{Mon}(\mathbf{x})$ such that $\mathrm{lm}_{\prec_x}(vg) = u$ and therefore such that $\mathrm{lm}_{\prec_x}(u - vg) \prec_x u$.
Reducing further the expression $u - vg$ by G is done with arithmetic over A only and hence shows that, in F, we can write $u = \sum_{s\in S} r_s s$ with $r_s \in A$. This shows that S generates the A-module F. Now, to prove that F is free over A, it suffices to show that there are no non-trivial A-relations between the elements of S. Suppose that for certain $s_1, \ldots, s_t \in S$, there is a relation

$$\sum_{i=1}^{t} r_i s_i = 0$$

in F with $r_i \in A \setminus \{0\}$ for all i. This gives in particular a relation between the s_i over K. Hence, if j is such that s_j is $\prec_x$-maximal among the s_i, then $s_j \in L$, but $L \cap S = \emptyset$, a contradiction. □

To proceed further, we will use

Proposition 7.1.1. *Let $M \neq 0$ be a finitely generated module over an integral domain S. Suppose $\mathfrak{m} \subset S$ is a maximal ideal, suppose $\mathfrak{m}'$ is an ideal with $\sqrt{\mathfrak{m}} = \mathfrak{m}'$ and that $\mathfrak{m}'M = 0$. Then $M \cong M_{\mathfrak{m}}$.*

Proof. Let $\varphi : M \to M_{\mathfrak{m}}$ be the map sending $M \ni a \mapsto a/1 \in M_{\mathfrak{m}}$. We show that φ is an isomorphism.
Suppose first that $a/1 = 0$ in $M_{\mathfrak{m}}$ for $a \in M$. Then there exists $f \notin \mathfrak{m}$ with $fa = 0$. Hence the ideal

$$\mathrm{ann}(a) := \{s \in S \mid sa = 0\}$$

contains $\mathfrak{m}' + \langle f \rangle = S$. Therefore $a = 0$. This establishes the injectivity of φ.
For the surjectivity, it suffices to show that for any $f \notin \mathfrak{m}$ we have $fM = M$. Indeed, in that case, for any $a/f \in M_{\mathfrak{m}}$ with $a \in M$ we have $b \in M$ with $a = bf$. Therefore $a/f = bf/f = b/1$ which lies in the image of φ. Note now that for such f we have $\mathfrak{m}' + \langle f \rangle = S$. Therefore

$$M = (\mathfrak{m}' + \langle f \rangle)M = \mathfrak{m}'M + fM = fM,$$

finishing the proof. □

In the sequel we will also use

Corollary 7.1.1. *Suppose $\mathfrak{m}$ is a point of good specialization. Then for each $u \in \mathrm{Mon}(\mathbf{z})$, the ideal I_u is zero-dimensional.*

Proof. By Theorem 7.1.1, the $\mathbb{K}[\mathbf{z}]_{\mathfrak{m}}$-module $\mathbb{K}[\mathbf{z}]_{\mathfrak{m}}[\mathbf{x}]/I$ is free of finite rank. Note that this implies that $\mathbb{K}[\mathbf{x},\mathbf{z}]/I_u \cong \mathbb{K}[\mathbf{z}]_{\mathfrak{m}}[\mathbf{x}]/I_u$ (Proposition 7.1.1) is a finitely generated free module over $\mathbb{K}[\mathbf{z}]_{\mathfrak{m}}/\mathfrak{m}_u \cong \mathbb{K}[\mathbf{z}]/\mathfrak{m}_u$ (Proposition 7.1.1) which itself is a finite dimensional vector space over $\mathbb{K}$. Hence $\mathbb{K}[\mathbf{x},\mathbf{z}]/I_u$ is itself a finite-dimensional $\mathbb{K}$-vector space. □

We now give some further properties regarding points of good specializations. Item 1 in the theorem below will be used to show the correctness of our lifting algorithm in Section 7.2 whereas Item 2 will be used for our complexity analysis in Section 7.3.

Theorem 7.1.2. *Suppose that $\mathfrak{m}$ is a point of good specialization.*

1. *For each $u \in \mathrm{Mon}(\mathbf{z})$ and $z_i \in \mathbf{z}$, the multiplication by z_i induces an isomorphism of $\mathbb{K}$-vector spaces:*
$$u\mathbb{K}[\mathbf{z},\mathbf{x}]/I_u \to z_i u\mathbb{K}[\mathbf{z},\mathbf{x}]/I_{z_i u}.$$
2. *Let $\prec$ be the block order eliminating $\mathbf{x}$ with $\prec=\prec_{\mathbf{x}}$ on $\mathrm{Mon}(\mathbf{x})$ and $\prec=\prec_{\mathrm{drl}}$ on $\mathrm{Mon}(\mathbf{z})$. Let M_u be the (unique) minimal generating set of $\mathfrak{m}_u$. Then, the reduced $\prec$-Gröbner basis of I_u is precisely $G_u \cup M_u$.*

Proof. We reuse the notation from the proof of Theorem 7.1.1. By Theorem 7.1.1, $\mathfrak{m}$ being a point of good specialization implies that F is a free R-module of finite rank.
Proof of (1): Note that multiplication by z_i induces a surjective, well-defined map of finite-dimensional (Corollary 7.1.1) $\mathbb{K}$-vector spaces
$$u\mathbb{K}[\mathbf{z},\mathbf{x}]/I_u \to z_i u\mathbb{K}[\mathbf{z},\mathbf{x}]/I_{z_i u}.$$
Note that the structure of $V_u := u\mathbb{K}[\mathbf{z},\mathbf{x}]/I_u$ as a vector space is induced by the canonical A-module structure of F, because $\mathfrak{m}V_u = 0$ and therefore $(V_u)_{\mathfrak{m}} = V_u$ by Proposition 7.1.1. Hence, if $F \cong A^r$, we have,
$$V_u \simeq (uA/\mathfrak{m}_u)^r \simeq \mathbb{K}^r.$$
Therefore multiplication by z_i induces an epimorphism between vector spaces of the same dimension, so it must be an isomorphism.
Proof of (2): Let $S := S_{\mathrm{gen}(I,\mathbf{z}),\prec_{\mathbf{x}}}$. It suffices to show that
$$S_{I_u,\prec} = S_u := \bigcup_{v \preceq_{\mathrm{drl}} u} vS.$$
Note that the set S_u certainly generates $\mathbb{K}[\mathbf{z},\mathbf{x}]/(I+\mathfrak{m}_u)$ as a $\mathbb{K}$-vector space. As $\mathbb{K}[\mathbf{z},\mathbf{x}]/(I+\mathfrak{m}_u) \simeq F/\mathfrak{m}_u$, and since F is free, a $\mathbb{K}$-dimension count shows that the set S_u is $\mathbb{K}$-linearly independent. Now, let $s \in S$ be $\prec_{\mathbf{x}}$-minimal such that there exists some $w \in \mathrm{Mon}(\mathbf{z})$, $w \preceq_{\mathrm{drl}} u$ with $ws \in \mathrm{lm}_{\prec}(I_u)$. By minimality, the $\prec$-normal form of ws w.r.t. I_u has support in $\bigcup_{v \preceq_{\mathrm{drl}} u}\{vt \mid t \prec_{\mathbf{x}} s, t \in S\} \subset S_u$, therefore inducing a linear dependence between the elements of S_u, a contradiction. □

7.2 HENSEL LIFTING AND THE FGLM ALGORITHM

Based on the theory presented in Section 7.1 the goal in this section is to show how to combine the FGLM algorithm (Algorithm 7, presented in Section 3.5) with a Hensel lifting strategy in order to compute the reduced Gröbner basis of $\mathrm{gen}(I, \mathbf{z})$ w.r.t. some monomial order $\prec_{\mathrm{out}}$ on $\mathrm{Mon}(\mathbf{x})$ where I is an ideal in $\mathbb{K}[\mathbf{z}, \mathbf{x}]$. We will assume throughout that $\mathfrak{m}$ is a point of good specialization for $\prec_{\mathrm{out}}$.
Recall that the FGLM algorithm requires a Gröbner basis H (w.r.t. some monomial order) of the ideal of interest as part of its input, this Gröbner basis is to be converted to the reduced Gröbner basis w.r.t. the monomial order of interest. In our setting, the role of H will be played by multiple Gröbner bases: Namely, we fix another monomial order $\prec_{\mathrm{in}}$ on $\mathrm{Mon}(\mathbf{x} \cup \mathbf{z})$ and define H_u to be the reduced $\prec_{\mathrm{in}}$-Gröbner basis of $I_u = I + \mathfrak{m}_u$ (see Definition 7.1.1).

Remark 7.2.1. Let us emphasize that there is a slight abuse of notation here: H_u is the reduced $\prec_{\mathrm{in}}$-Gröbner basis of I_u and should not be read following the notation introduced in Definition 7.1.2.

Using the data of these Gröbner bases H_u, our goal is now to compute the reduced $\prec_{\mathrm{out}}$-Gröbner basis G of $\mathrm{gen}(I, \mathbf{z})$.
Let us sketch our strategy. By the assumption that $\mathfrak{m}$ is a point of good specialization, we have $G \subset \mathbb{K}[\mathbf{z}]_{\mathfrak{m}}[\mathbf{x}]$. Recall that $\mathfrak{m}_1 = \mathfrak{m}$, in the notation of Definition 7.1.2. We start by running the FGLM algorithm (Algorithm 7) with H_1 to obtain the reduced $\prec_{\mathrm{out}}$-Gröbner basis of the image of I in $(\mathbb{K}[\mathbf{z}]/\mathfrak{m})[\mathbf{x}] \simeq \mathbb{K}[\mathbf{x}]$. By Theorem 7.1.2 this Gröbner basis will now precisely be the set G_1 in the notation introduced in Definition 7.1.2.
For a monomial $u \in \mathrm{Mon}(\mathbf{z})$, let $v := \mathrm{next}(u)$. Starting with $u = 1$ and a given $g_1 \in G_1$, we will lift g_u to g_v by performing linear algebra in the finite-dimensional (Corollary 7.1.1) $\mathbb{K}$-vector space $v\mathbb{K}[\mathbf{z}, \mathbf{x}]/I_v$, using the $\prec_{\mathrm{in}}$-Gröbner basis H_v in a role analogous to how the input Gröbner basis is used in the original FGLM algorithm. This will rely on Item 1 in Theorem 7.1.2.

Remark 7.2.2. In this section we assume the required Gröbner bases H_u, $u \in \mathrm{Mon}(\mathbf{z})$, to be given, they can be computed for example by Buchberger's algorithm (Algorithm 3) or F_4 (Algorithm 4) before they are needed in our algorithm. In Section 7.3, we point out that these sets may be obtained free of arithmetic operations from an $\prec_{\mathrm{in}}$-Gröbner basis of I when $\prec_{\mathrm{in}} = \prec_{\mathrm{drl}}$ and I satisfies a certain genericity assumption. Under the assumption that $\prec_{\mathrm{in}}$ is a suitable block order and that $\mathfrak{m}$ is also a point of good specialization for $\prec_{\mathrm{in}}$ restricted to $\mathrm{Mon}(\mathbf{x})$, we give in Section 7.4 a tracer-based (Section 3.4.1) method to compute the sets H_u, this is the second contribution of this chapter.

This lifting step is now given by Algorithm 19.

Algorithm 19 Lifting of Gröbner basis elements

Input: A monomial $u \in \mathrm{Mon}(\mathbf{z})$, $g_u \in G_u$, $v := \mathrm{next}(u)$, the reduced Gröbner basis H_v of I_v w.r.t. $\prec_{\mathrm{in}}$, the staircase $S := S_{I_1,\prec_{\mathrm{out}}}$
Output: The corresponding element $g_v \in G_v$

1: **procedure** LIFT(g_u, H_v, S)
2: $c \leftarrow c_{I_v,\prec_{\mathrm{in}}}(g_u)$ *see Definition 3.1.6, computed via* REDUCE($H_v, \prec_{in}, g_u$)
3: **if** $c = 0$
4: **return** g_u
5: compute α_w s.t. $c = \sum_{w\in S} \alpha_w c_{I_v,\prec_{\mathrm{in}}}(uw)$ *computed via* H_v
6: **return** $g_u - \sum_{w\in S} \alpha_w uw$

Note that in Line 5 we are just solving a linear system.

Theorem 7.2.1. *If $\mathfrak{m}$ is a point of good specialization for $\prec_{out}$, then Algorithm 19 terminates and is correct in that it satisfies its output specification.*

Proof. We use the notation from the pseudocode of the algorithm. The termination of the algorithm is clear. For the correctness of the algorithm, note that the vectors $c_{I_v,\prec_{\mathrm{in}}}(uw)$ in line 5 are linearly independent thanks to Item 1 of Theorem 7.1.2. Thus, there exists at most one choice of coefficients $\alpha_w, w \in S_{I_1,\prec_{\mathrm{out}}}$, such that $c = \sum_{w\in S_{I_1,\prec_{\mathrm{out}}}} \alpha_w c_{I_v,\prec_{\mathrm{in}}}(uw)$. Furthermore, since $\mathfrak{m}$ is a point of good specialization, the element $g \in G$ corresponding to g_u provides such a choice of coefficients, implying that there exists at least one solution to this linear system. This proves the correctness. □

Remark 7.2.3. We want to emphasize that our algorithms never verify deterministically (and cannot verify deterministically) whether $\mathfrak{m}$ is a point of good specialization, this is a probabilistic assumption. Nonetheless, running Algorithm 19 can sometimes detect when $\mathfrak{m}$ is not a point of good specialization, namely if there exists no or more than one solution to the linear system in Line 5 of Algorithm 19. In this case one would apply a random change of coordinates $z_i \leftarrow z_i - a_i$ for each $z_i \in \mathbf{z}$ and restart the computation.

Example 7.2.1. Let us unroll Algorithm 19 by considering Example 7.1.1 over the field $\mathbb{F}_{11}$, denoting the finite field with eleven elements. For the ideal generated by the polynomials in question, $\mathfrak{m}$ is not a point of good specialization. So we replace the variable z by $z+8$ to ensure, probabilistically, that $\mathfrak{m}$ is a point of good specialization. Our ideal I is now generated by

$$\begin{aligned}(z+8)+x_1+x_2+x_3,\\ (z+8)x_1+x_1x_2+x_2x_3+x_3(z+8),\\ (z+8)x_1x_2+x_1x_2x_3+x_2x_3(z+8)+x_3(z+8)x_1,\\ (z+8)x_1x_2x_3-1.\end{aligned}$$

Following Remark 7.2.5, one verifies that $\mathbb{F}_{11}[z] \to \mathbb{F}_{11}[z, x_1, x_2, x_3]/I$ is injective with generically finite fiber.
For our orders we choose $\prec_{in} = \prec_{drl}$ on $\mathrm{Mon}(\mathbf{x} \cup \{z\})$ and $\prec_{out}$ as the lexicographic order on $\mathrm{Mon}(\mathbf{x})$. Now, the set G_1 is given by

$$G_1 := \{x_3^2 + 6, x_2 + 8, x_1 + x_3\}.$$

Hence $S_{I_1, \prec_{out}} = \{1, x_3\}$. Assuming that $g_1 := x_3^2 + 6$ is the image of some element g in the target Gröbner basis $G \subset \mathbb{F}_{11}(z)[\mathbf{x}]$, we now try to lift g_1 to g_z, i.e. the image of g modulo $\mathfrak{m}_z = \langle z^2 \rangle$, so that, in the notation of Algorithm 19, we have $u = 1$ and $v = z$.
If such a g exists, there must now exist, by Item 1 in Theorem 7.1.2, unique scalars $\alpha_1, \alpha_{x_3} \in \mathbb{F}_{11}$ such that

$$g_z = g_1 + \alpha_1 z + \alpha_{x_3} z x_3 = 0 \bmod I_z = I + \langle z^2 \rangle,$$

and Algorithm 19 attempts to compute these scalars by finding a linear relation between the normal forms w.r.t. $\prec_{in}$ of g_1, z and zx_3 modulo I_z. Using an $\prec_{in}$-Gröbner basis of I_z, we find that $S_{I_z, \prec_{in}} = \{1, z, x_3, zx_3\}$ and we compute, using normal form computations

$$\begin{aligned} c_{I_2, \prec_{in}}(g_1) &= (0, 7, 0, 0) \sim 7z \\ c_{I_2, \prec_{in}}(z) &= (0, 1, 0, 0) \sim z \\ c_{I_2, \prec_{in}}(zx_3) &= (0, 0, 0, 1) \sim zx_3 \end{aligned}$$

so that finally, $\alpha_1 = 6$ and $\alpha_{x_3} = 0$ which yields for g_z the unique candidate $g_z = x_3^2 + (4z + 6)$, finishing the example.

Algorithm 19 is only able to compute the set G_u for a monomial $u \in \mathrm{Mon}(\mathbf{z})$, i.e. it "approximates" the set G up to order u. A natural question is then how to extract the actual set G out of G_u. For this, we may use the classical technique of Padé approximants. Having computed the set G_u, we have computed the image g_u of a given element $g \in G$ as

$$g_u = \sum_{w \in \mathrm{Mon}(\mathbf{x})} \sum_{v \preceq_{drl} u} r_{w,v} v w.$$

Now we have for the coefficient $p_w/q_w \in \mathbb{K}(\mathbf{z})$ of w in g

$$p_w - q_w \sum_{v \preceq_{drl} u} r_{w,v} v = 0 \bmod \mathfrak{m}_u, \tag{7.1}$$

which determines a set of linear equations in the unknown coefficients of p_w and q_w. Let $d := \deg u$. Suppose that $\deg(\mathrm{next}(u)) = d + 1$, so that $\mathfrak{m}_u = \mathfrak{m}^{d+1}$. Fix d_1 and d_2 with $d_1 + d_2 = d$ and let n be the cardinality of the set $\mathbf{z}$. If we impose that $\deg p_w \leqslant d_1$ and $\deg q_w \leqslant d_2$, then the linear system (7.1) has a finite set of unknowns and equations. Let us say that any solution to this linear system of equations is a *Padé approximant of order* (d_1, d_2) *of* $\lambda_w := \sum_{\deg v < d+1} r_{w,v} v$. If d_1 and d_2 are large enough then any Padé approximant of order (d_1, d_2) of λ_w is equal to p_w/q_w:

Lemma 7.2.1 (Proposition 2.1 in Guillaume and Huard, 2000)**.** *Let* p/q *be a Padé approximant of order* (d_1, d_2) *of* λ_w*. If* $d_1 \geqslant \deg p_w$ *and* $d_2 \geqslant \deg q_w$ *then* $p/q = p_w/q_w$*.*

By solving this linear system we obtain an algorithm $\textsc{Pade}(g_u, d_1, d_2)$ which computes a candidate $g_{\text{cand}} \in \mathbb{K}(\mathbf{z})[\mathbf{x}]$ whose coefficients are Padé approximants of the coefficients of g_u of order (d_1, d_2) regarded as a polynomial in the variables $\mathbf{x}$. Let us say that g_u has *stable Padé approximation* if for $v := \text{next}(u)$ we have

$$g_{\text{cand}} = g_v \bmod \mathfrak{m}_v.$$

Based on this, we now obtain Algorithm 20 for computing the set G probabilistically. We state this algorithm in an informal way. In Line 6 by "lifting G_{lift} to degree d" we mean that we compute the set G_u where u is the $\prec_{\text{drl}}$-maximal monomial of degree d.

Algorithm 20 Computing a Gröbner basis of the generic fiber

Input: A generating set F of I, a monomial order $\prec_{\text{in}}$, a monomial order $\prec_{\text{out}}$.
Output: A guess for the set G.

1: $H \leftarrow$ reduced $\prec_{\text{in}}$-Gröbner basis of I_1 *(using F)*
2: $G_{\text{lift}}, S \leftarrow \text{FGLM}(H, \prec_{\text{out}})$
3: $G_{\text{result}} \leftarrow \emptyset$
4: $d \leftarrow 2$
5: **while** $G_{\text{lift}} \neq \emptyset$
6: $\quad G_{\text{lift}} \leftarrow$ lift G_{lift} to degree d *(Algorithm 19 with S)*
7: $\quad$ Run $\textsc{Pade}(g, d/2, d/2)$ for all $g \in G_{\text{lift}}$
8: $\quad$ Lift G_{lift} one monomial higher *(Algorithm 19 with S)*
9: $\quad$ Add to G_{result} all elements with stable Padé approx.
10: $\quad$ Remove the corresponding elements from G_{lift}
11: $\quad d \leftarrow 2d$
12: **return** G_{result}

Clearly, by Theorem 7.2.1 and Lemma 7.2.1, this algorithm returns the correct result if the computed Padé approximants are of sufficiently large degree and $\mathfrak{m}$ is a point of good specialization.

Remark 7.2.4. Note that Algorithm 19 works also if we replace v by any monomial larger than u: In this case we just have to write c as a linear combination of all the vectors $c_{I_v, \prec_{\text{in}}}(uv')$ where $u \prec_{\text{drl}} v' \preceq_{\text{drl}} v$.

Example 7.2.2 (Example 7.2.1 continued)**.** Let us try to see how Algorithm 20 recovers the element denoted g in Example 7.2.1. First, Algorithm 20 lifts the element $g_1 = x_3^2 + 6$ to degree $d = 2$, i.e. we compute the image of g modulo z^3 (Line 6 in Algorithm 20). This yields, in our usual notation,

$$g_{z^2} = x_3^2 + (2z^2 + 4z + 6).$$

Now we attempt (Line 7 in Algorithm 20) to find a Padé approximation of order $(1,1)$ for g, i.e., here, $p, q \in \mathbb{F}_{11}[z]$ of degree at most one such that $p/q = 2z^2 + 4z + 6 \bmod z^3$ by solving a linear system as outlined above. This yields the candidate

$$g_{\text{cand}} = x_3^2 + \frac{z+6}{5z+1}$$

which satisfies $g_{\text{cand}} = g_{z^2} \bmod z^3$. Next, in Line 8, we lift g one monomial higher, i.e. modulo z^4. This yields

$$g_{z^3} = x_3^2 + (7z^3 + 2z^2 + 4z + 6),$$

But p/q has now the truncated power series $z^3 + 2z^2 + 4z + 6$, so that g_2 does not have stable Padé approximation. Hence we double d to 4 and lift g_{z^3} to g_{z^4}, i.e. from modulo z^4 to modulo z^5, and attempt another Padé approximation. This time, computing a Padé approximation of order $(2,2)$, this yields the candidate

$$g_{\text{cand}} = x_3^2 + \frac{1}{10z^2 + 6z + 2}.$$

Finally, we lift g_{z^4} to g_{z^5} and find that $g_{z^5} = g_{\text{cand}} \bmod z^6$. So g_{z^4} has stable Padé approximation and we terminate with $g := g_{\text{cand}}$. Computing the $\prec_{\text{out}}$-Gröbner basis G of $\text{gen}(I, \mathbf{z})$ using block orders as in Proposition 3.2.2 shows that g is indeed the correct element.

Remark 7.2.5. Note that we have so far required that the partition of the variables of $\mathbb{K}[\mathbf{z}, \mathbf{x}]$ is given. Recall that, using Proposition 3.2.1, one can also computationally determine such a partition s.t. $\text{gen}(I, \mathbf{z})$ is zero-dimensional from Proposition 3.2.1 from any Gröbner basis of I.

7.3 COMPLEXITY ESTIMATES

In this section, we analyze the arithmetic complexity of a version of the algorithm presented in the last section more akin to the original FGLM algorithm as presented by Faugère, Gianni, Lazard, and Mora (1993). We will reuse the notation from the last section and additionally denote throughout $\mathbf{z} = \{z_1, \ldots, z_r\}$. We will treat r as a constant in our analysis, note that $r = \dim(I)$ by Proposition 2.4.1 and Proposition 3.2.1.
We now add the additional assumption that the order $\prec_{\text{in}}$ used in the last section is a block order eliminating $\mathbf{x}$ with $\prec_{\text{in}} = \prec_{\text{drl}}$ on $\text{Mon}(\mathbf{z})$ and that $\mathfrak{m}$ is a point of good specialization for both $\prec_{\text{in}}$ restricted to $\text{Mon}(\mathbf{x})$ and $\prec_{\text{out}}$. Here, we analyze the number of arithmetic operations in $\mathbb{K}$ required to obtain the sought $\prec_{\text{out}}$-Gröbner basis G using the same strategy as in Algorithm 20, but with a more optimized lifting step. As is standard, we say that an algorithm has arithmetic complexity $\widetilde{O}(g(n))$, for some polynomial g, if the algorithm has arithmetic complexity $O\left(g(n)\log^k(n)\right)$ for some k.

Our cost analysis will require measuring the cost of performing certain linear algebra operations on structured matrices, these will enable us to perform the necessary normal form computations in Algorithm 20 with a well controlled cost, similar to what has been done by Faugère, Gianni, Lazard, and Mora (1993). These structured matrices will be *multiplication matrices*, named so because they will represent the $\mathbb{K}$-linear maps

$$\begin{aligned}\mathbb{K}[\mathbf{z},\mathbf{x}]/I_u &\to \mathbb{K}[\mathbf{z},\mathbf{x}]/I_u\\ f &\mapsto xf\end{aligned}$$

for $x \in \mathbf{x}$ and different $u \in \mathrm{Mon}(\mathbf{z})$.
In our setting these matrices will have a very particular structure, namely they can be identified with polynomials in a certain way. We introduce the necessary theory before we perform our complexity analysis.

7.3.1 *Multi-Block-Toeplitz Band Matrices*

Let S be any ring and let $S^{k\times k}$ be the S-algebra of $k \times k$-matrices with entries in S. Consider the map

$$\begin{aligned}\phi : S[t] &\to S^{k\times k}\\ t &\mapsto (\delta_{i,j-1})_{1\leqslant i,j\leqslant k}\end{aligned}$$

where $\delta_{i,j-1} \in S$ denotes the Kronecker delta.
Suppose that $p \in \mathbb{K}^{D\times D}[z_1,\dots,z_r]$ with degree in z_i bounded by some $k_i \in \mathbb{N}$ then, slightly abusing notation, p can be identified with a matrix $\phi(p) \in \mathbb{K}^{k_1\dots k_r D\times k_1\dots k_r D}$: We first apply the map ϕ defined above with $S := \mathbb{K}^{D\times D}[z_1,\dots,z_{r-1}]$ and $t := z_r$. This will yield a matrix in $\left(\mathbb{K}^{D\times D}[z_1,\dots,z_{r-1}]\right)^{k_r\times k_r}$. We can then apply ϕ to each entry of this matrix, this time with $S := \mathbb{K}^{D\times D}[z_1,\dots,z_{r-2}]$ and $t := z_{r-1}$, this yields a matrix in $\left(\mathbb{K}^{D\times D}[z_1,\dots,z_{r-2}]\right)^{k_r k_{r-1}\times k_r k_{r-1}}$. Continuing in this manner, we obtain $\phi(p) \in \mathbb{K}^{k_1\dots k_r D\times k_1\dots k_r D}$.

Definition 7.3.1 (Multi-Block-Toeplitz Band)**.** A matrix

$$M \in \mathbb{K}^{k_1\dots k_r D\times k_1\dots k_r D}$$

is called *multi-block-Toeplitz band* of type $(k_1,\dots,k_r,D)$ for $k_1,\dots,k_r,D \in \mathbb{N}$ if there exists a polynomial $p \in \mathbb{K}^{D\times D}[\mathbf{z}]$ with partial degrees bounded by $k_1,\dots,k_r$ s.t. $M = \phi(p)$. It is called *block-Toeplitz band* if $r = 1$.

Remark 7.3.1. Note that if $r = D = 1$, then a block-Toeplitz band matrix is a lower-triangular *Toeplitz matrix*, i.e. a matrix with entries repeating along each diagonal.

A vector $v \in \mathbb{K}^{k_1 \dots k_r D}$ may be recursively identified with a polynomial $q_v \in \mathbb{K}^D[\mathbf{z}]$ as follows: Divide v into k_r blocks $v_0, \dots, v_{k_r-1} \in \mathbb{K}^{k_1 \dots k_{r-1} D}$. Having defined $q_{v_0}, \dots, q_{v_{k_r-1}} \in \mathbb{K}^D[z_1, \dots, z_{r-1}]$, let

$$q_v := q_{v_{k_r-1}} z_r^{k_r-1} + \cdots + q_{v_0}.$$

We now want to measure the arithmetic cost of computing a product Mv with v a vector where M is multi-block-Toeplitz band and the cost of inverting such a matrix M. We start with the case $r = 1$.
For the required complexities, we need the concept of *displacement rank* of a matrix.

Definition 7.3.2 (Displacement Rank)**.** Let $Z \in \mathbb{K}^{n \times n}$ be the matrix defined by

$$Z := (\delta_{i-1,j})_{1 \leqslant i,j \leqslant n},$$

where $\delta_{i-1,j}$ is the Kronecker delta and let Z^T be the transpose of Z. The *displacement rank* of a matrix $M \in \mathbb{K}^{n \times n}$ is

$$\alpha(M) := \operatorname{rk}(M - ZMZ^T).$$

Note that the displacement rank of a Toeplitz matrix (and thus in particular of block-Toeplitz band matrix with $r = D = 1$) is upper bounded by 2. The concept of displacement rank can be used as a general method to utilize "Toeplitz-like" structures in algorithmic linear algebra and has been investigated by Bitmead and Anderson (1980), Bostan, Jeannerod, Mouilleron, and Schost (2017), and Morf (1980). We now have:

Proposition 7.3.1. *Let M be block-Toeplitz band of type (k, D) and let $v \in \mathbb{K}^{kD}$. Then*

1. *Mv can be computed in $\widetilde{O}(kD^2)$ arithmetic operations in $\mathbb{K}$,*
2. *If N is another block-Toeplitz band matrix of type (k, D) then the product MN can be computed in $\widetilde{O}(kD^3)$ operations and*
3. *M can be inverted in $\widetilde{O}(kD^3)$.*

Proof. For any matrix $M \in \mathbb{K}^{n \times n}$, according to Bostan, Jeannerod, Mouilleron, and Schost (2017), a matrix-vector product Mv can be computed in time $\widetilde{O}(\alpha(M)n)$ and M can be inverted in time $\widetilde{O}(\alpha(M)^{\omega-1}n)$. Using a series of rows and column swaps, more precisely sending row $pD + i$ to $ik + p$ (resp. column $qD + j$ to $jk + q$), we may transform a block-Toeplitz band matrix M of type (k, D) into a matrix $N = (N_{ij})_{0 \leqslant i,j < D} \in \mathbb{K}^{kD \times kD}$ where each N_{ij} lies in $\mathbb{K}^{k \times k}$ and is Toeplitz. Now, $N - ZNZ^T$ has D dense columns and $(k-1)D$ columns with potentially nonzero coefficients in positions iD for all i. Only D of these latter columns can be linearly independent so that $\alpha(M) \leqslant 2D$, proving the claims in Item 1 and Item 3.

To prove the claim in Item 2, note that MN is again block-Toeplitz band of type (k, D). Therefore, to compute the product MN it suffices to compute the product of M with the first D columns of N. Thanks to Item 1, this is done in time $\widetilde{O}\left(kD^3\right)$. □

Remark 7.3.2. We remark that the algorithm for matrix inversion used in the proof of Item 3 in Proposition 7.3.1 is probabilistic (Las Vegas). This probability depends on the size of a chosen finite subset in $\mathbb{K}$ with probability of failure going to zero as the size of this finite subset increases.

We now go back to the general case $r \geqslant 1$.

Proposition 7.3.2. *Let $M \in \mathbb{K}^{k_1 \ldots k_r D \times k_1 \ldots k_r D}$ be multi-block-Toeplitz band of type $(k_1, \ldots, k_r, D)$ and let $v \in \mathbb{K}^{k_1 \ldots k_r D}$. Then*

(1) Mv can be computed in $\widetilde{O}\left(k_1 \ldots k_r D^2\right)$ arithmetic operations in $\mathbb{K}$ and

(2) M can be inverted in $\widetilde{O}\left(k_1 \ldots k_r D^3\right)$.

Proof. Proof of (1): The case $r = 1$ being settled by Proposition 7.3.1, we work by induction over r. Identify v with an element $q_v \in \mathbb{K}^D[\mathbf{z}]$ as above and let $p \in \mathbb{K}^{D \times D}[\mathbf{z}]$ be s.t. $\phi(p) = M$. We measure the cost of computing the product pq_v. Make the substitution $z_r \leftarrow z_{r-1}^{2k_r}$ and denote by $\tilde{p}$ and $\tilde{q_v}$ the images of p and q_v under this substitution. The degrees in z_{r-1} of $\tilde{p}$ and $\tilde{q_v}$ are bounded by $2k_{r-1}k_r$. By our induction hypothesis we can therefore compute the product $\tilde{p}\tilde{q_v}$ in $\widetilde{O}\left(k_1 \ldots k_r D^2\right)$ arithmetic operations in $\mathbb{K}$. Regarding p and q_v as bivariate polynomials in z_{r-1} and z_r, we can now recover the coefficient of $z_{r-1}^i z_r^j$ in pq_v as the coefficient of $z_{r-1}^i z_{r-1}^{2k_{r-1}j} = z_{r-1}^{2k_{r-1}j+i}$ in the product of $\tilde{p}\tilde{q_v}$ since the degree of p in z_{r-1} is bounded by $2k_{r-1} - 1$. This proves the desired statement.
Proof of (2): Again, we work inductively over r, the case $r = 1$ being settled by Proposition 7.3.1. Let $p \in \mathbb{K}^{D \times D}[\mathbf{z}]$ be such that $\phi(p) = M$. Regarding p as a univariate polynomial in z_r, i.e.

$$p \in \left(\mathbb{K}^{D \times D}[z_1, \ldots, z_{r-1}]\right)[z_r],$$

the inverse of M is then given by the associated power series of $1/p$ up to order k_r. As is well known, this can be done in the same time as multiplying p with another polynomial $q \in \left(\mathbb{K}^{D \times D}[z_1, \ldots, z_{r-1}]\right)[z_r]$ with partial degrees bounded by $k_1, \ldots, k_r$ using Newton iteration, see e.g. Section 2 in Johansson (2015). By the exact same substitution argument as in the proof for Item (1), and using Item 2 in Proposition 7.3.1, this can be done in time $\widetilde{O}\left(k_1 \ldots k_r D^3\right)$ by our induction hypothesis, finishing the proof. □

7.3.2 *The Complexity Analysis*

Now, let us introduce the notion of *multiplication tensors* relevant to this complexity analysis.

Definition 7.3.3 (Multiplication Tensor)**.** Let I be a zero-dimensional ideal in the polynomial ring R and let $\prec$ be a monomial order. Let $S := S_{I,\prec}$. The multiplication tensor of I w.r.t. $\prec$ is defined as the 3-tensor

$$M(I, \prec) = (c_{I,\prec}(x_i u))_{x_i \in \mathbf{x}, u \in S}.$$

Recall that, in the context of this definition, the vectors $c_{I,\prec}(x_i u)$ are defined as in Definition 3.1.6. Note that $M(I, \prec)_{x_i}$ is precisely the matrix representation of the linear map $R/I \to R/I, f \mapsto x_i f$ in terms of the basis S. The column of $M(I, \prec)_{x_i}$ indexed by $u \in S$ is the vector $c_{I,\prec}(x_i u)$.
We now fix D to be the degree of $\mathrm{gen}(I, \mathbf{z})$, i.e. the $\mathbb{K}(\mathbf{z})$-dimension of $\mathbb{K}(\mathbf{z})[\mathbf{x}]/\mathrm{gen}(I, \mathbf{z})$. Note that this degree is upper-bounded by that of I. Denote further $I_k := \langle z_1^k, \ldots, z_r^k \rangle$, H_k as the reduced $\prec_{in}$-Gröbner basis of I_k and G_k as the reduced $\prec_{out}$-Gröbner basis of I_k. We now analyze the complexity of successively converting H_k to G_k (like in Faugère, Gianni, Lazard, and Mora, 1993) for $k = 2, 4, \ldots, 2^\ell$ until ℓ is large enough so as to recover G from G_k by means of Padé approximation as in Algorithm 20. We will closely follow the analysis of Faugère, Gianni, Lazard, and Mora (1993) while using the fact that the occuring multiplication matrices turn out to be multi-block-Toeplitz band in conjunction with the results of Section 7.3.1.

Theorem 7.3.1. *Let $k \in \mathbb{N}$. Let c be the cardinality of $\mathbf{x}$. Suppose that we are given the set H_k. Then the multiplication tensor of I_k w.r.t. $\prec_{in}$ is computed in arithmetic complexity $\widetilde{O}\left(k^r c D^3\right)$.*

Proof. Let E_k be the $\prec_{drl}$-staircase of $\langle z_1^k, \ldots, z_r^k \rangle$ and let S be the $\prec_{in}$-staircase of I_1. E_k consists of all monomials in $\mathrm{Mon}(\mathbf{z})$ with partial degrees bounded by k. By Item 2 of Theorem 7.1.2 we have that the $\prec_{in}$-staircase S_k of I_k is given by

$$S_k = \bigcup_{u \in E_k} uS.$$

For $x_i \in \mathbf{x}$ let $M_{x_i} := M(I_k, \prec_{in})_{x_i}$. We now measure the cost of computing the matrices M_{x_i}. For $u \in \mathbf{z}$, let $M_u \in \mathbb{K}^{D \times D}$ be the matrix whose columns are indexed by $x_i S$, whose rows are indexed by uS and whose entries are the coefficients corresponding to the elements of uS of the $\prec_{in}$-normal forms of the elements in $x_i S$. Note now that for $u \in \mathrm{Mon}(\mathbf{z})$ and $v \in \mathrm{Mon}(\mathbf{x})$ we have

$$\mathrm{NF}_{\prec_{in}}(uv, I_k) = u\,\mathrm{NF}_{\prec_{in}}(v, I_k).$$

In particular, the multiplication matrices corresponding to the variables $z_1, \ldots, z_r$ are obtained free of arithmetic operations. This also implies that if we define

$$p_{x_i} := \sum_{u \in E_k} M_u u$$

then $M_{x_i} = \phi(p_{x_i})$ with ϕ defined as above, so that M_{x_i} is multi-block-Toeplitz band of type $(k, \ldots, k, D)$. In particular, to recover the matrices M_{x_i}, it suffices to compute the $\prec_{in}$-normal forms of the elements in $\mathbf{x}S$ of which there are at most cD. Now, we proceed as follows: Sort the set $\mathbf{x}S$ by the monomial order $\prec_{in}$. Choose $u \in \mathbf{x}S$ and suppose that the normal forms of all elements less than u in $\mathbf{x}S$ are known. Two easy cases can arise:

1. $u \in S$, in which case the normal form of u is computed without any arithmetic operations;
2. $u \in \mathrm{lm}(H_k)$, in which case the normal form of u is computed without any arithmetic operations, it is just given by the tail of the corresponding element in H_k.

Lastly, it can happen that $u \in \mathrm{lm}(I_k)$ but $u \notin \mathrm{lm}(H_k)$. In this case there exists $v \in \mathbf{x}S$ and $x_j \in \mathbf{x}$ with $u = x_j v$. By assumption the normal form of v is known and so is the normal form of each element $x_j b$ with $b \in S$ and $b \prec_{in} v$. Since M_{x_j} is also multi-block-Toeplitz band of type $(k, \ldots, k, D)$, we can now compute the required column of M_{x_i} as the product of M_{x_j} with a vector. This is done in time $\widetilde{O}\left(k^r D^2\right)$ thanks to Proposition 7.3.2. We need to compute at most cD such matrix-vector products concluding the proof. □

From this we conclude

Corollary 7.3.1. *Let $k \in \mathbb{N}$. Let c be the cardinality of $\mathbf{x}$. Given the set H_k, the set G_k is computed in arithmetic complexity $\widetilde{O}\left(k^r c D^3\right)$.*

Proof. By Theorem 7.3.1, the $\prec_{in}$-multiplication tensor of I_k can be computed in arithmetic complexity $\widetilde{O}\left(k^r c D^3\right)$. Having computed this tensor, we proceed as follows: let S be the $\prec_{in}$-staircase of $I_1 = I + \mathfrak{m}$, T be the $\prec_{out}$-staircase of I_1 and L be the set of minimal $\prec_{out}$-leading terms of I_1, with $\mathbf{z}$ removed. Denoting S_k as in the proof of the preceding theorem, and similarly T_k, now we first compute the $\prec_{in}$-normal forms of each element in $T \cup L$ w.r.t. I_{2k}. Writing these normal forms as column vectors indexed by S yields a tableau of the form

$$\begin{array}{c} \\ S \end{array} \begin{array}{c} \begin{array}{cc} T_k & L \end{array} \\ \left[\begin{array}{cc} C & B \end{array}\right] \end{array}.$$

Note that C is again, by the exact some argument as in the proof of the preceeding theorem, multi-block-Toeplitz band of type $(k, \ldots, k, D)$.

The required matrices C and B can be computed by using the $\prec_{in}$-multiplication tensor of I_k , enumerating the monomials in $\mathrm{Mon}(\mathbf{x})$ in order of $\prec_{out}$ and computing their normal forms via matrix-vector multiplications similar to the proof of the preceding theorem. Combining this with the fact that the multiplication matrices of I_k w.r.t. $\prec_{in}$ are multi-block-Toeplitz band of type $(k, \dots, k, D)$, this can again be done in time $\widetilde{O}\left(k^r c D^3\right)$. Finally, to compute the set G_k, we have to write each column in B corresponding to an element in L as a $\mathbb{K}$-linear combination of the columns corresponding to T, Hence this requires

- inverting the submatrix C which, by Proposition 7.3.2, is done in time $\widetilde{O}\left(k^r D^3\right)$;
- computing the product $C^{-1}B$.

Note that C is certainly invertible, since it encodes a change of basis, and that the cardinality of L is upper bounded by cD. Thus, for the second step above, we have to compute at most cD matrix-vector products of the form $C_0^{-1}v$. Again, thanks to Proposition 7.3.2, this is done in time $\widetilde{O}\left(k^r c D^3\right)$. This finally yields the desired complexity. □

Remark 7.3.3. Let us compare Corollary 7.3.1 with Proposition 4.1 in Faugère, Gianni, Lazard, and Mora (1993): Here, it is shown that the reduced Gröbner basis of a zero-dimensional ideal J in n variables w.r.t. one monomial order can be converted to the reduced Gröbner basis w.r.t. another order in $O(nD^3)$ arithmetic operations where D is the degree of J. Our Corollary 7.3.1 preserves the cubic behavior in the degree while being quasi-linear in the number of terms computed in $\mathbb{K}[\mathbf{z}]$.

The following corollary gives the complexity of computing successively the sets G_{2^i} from H_{2^i} until i is large enough to recover G, like in Algorithm 20.

Corollary 7.3.2. *Let c be the cardinality of $\mathbf{x}$. Let $\delta - 1$ be the maximum degree of all numerators and denominators of all coefficients of G. Further, let ℓ be minimal such that $2^\ell \geqslant 2\delta$.*
Given H_{2^ℓ}, computing successively the sets G_{2^i}, for $i = 1, \dots, \ell$, can be done in arithmetic complexity $\widetilde{O}\left(2^{\ell r} c D^3\right) = \widetilde{O}\left(\delta^r c D^3\right)$.

Proof. Note that the sets H_{2^i}, for $i = 1, \dots, \ell$, are obtained from H_{2^ℓ} free of arithmetic operations. By Corollary 7.3.1, the computation of G_{2^i} requires $\widetilde{O}\left(2^{ir} c D^3\right)$ operations. Summing these complexities for i from 1 to ℓ yields the desired complexity. □

We close this section by pointing out a well-known case in which $\prec_{in}$ is the $\prec_{drl}$ order, the required Gröbner bases H_u of $I + \mathfrak{m}_u$ are extracted without any arithmetic operations of the $\prec_{drl}$-Gröbner basis H of I and the $\prec_{drl}$-staircase of $I + \mathfrak{m}_u$ behaves the same as in the above case when $\prec_{in}$ is a block order. We start with

Definition 7.3.4 (Projective Generic Position)**.** Let y be an extra variable and let $I^{hom} \subset \mathbb{K}[\mathbf{z}, \mathbf{x}, y]$ be the homogenization of I w.r.t y. Suppose that I^{hom} is Cohen-Macaulay, i.e. that the length of any maximal regular sequence in $\mathbb{K}[\mathbf{z}, \mathbf{x}, y]$ equals the dimension of I^{hom}, which is the cardinality of $\mathbf{z}$ plus one. We say that I is in *projective generic position* if $\{y\} \cup \mathbf{z}$ is such a maximal homogeneous regular sequence in $\mathbb{K}[\mathbf{z}, \mathbf{x}, y]/I^{hom}$.

Remark 7.3.4. Note that in the homogeneous setting of Definition 7.3.4, $\{y\} \cup \mathbf{z}$ is a maximal homogeneous regular sequence in $\mathbb{K}[\mathbf{z}, \mathbf{x}, y]/I^{hom}$ if and only if $\dim(I^{hom} + \langle y, \mathbf{z}\rangle) = 0$. Using Lemma 2.3.2 this can be ensured after a generic (i.e. chosen out of a suitably defined Zariski-open subset) change of coordinates, since Cohen-Macaulayness implies in particular equidimensionality (Corollary 18.11 in Eisenbud, 1995).

Supposing that I^{hom} is Cohen-Macaulay we now have the following statement. This statement has frequently been used in the complexity analysis of Gröbner basis algorithms.

Lemma 7.3.1 (Theorem 15.13 in Eisenbud, 1995)**.** *Let I be in projective generic position with I^{hom} Cohen-Macaulay. Let H be the reduced $\prec_{drl}$-Gröbner basis of I (with the variables in $\mathbf{z}$ considered smaller as those in $\mathbf{x}$). Then*

$$\mathrm{lm}(H) \subset \mathrm{Mon}(\mathbf{x}).$$

In particular, if S is the $\prec_{drl}$-staircase of $I_1 := I + \mathfrak{m}$, then the $\preceq_{drl}$-staircase of $I + \mathfrak{m}_u$ is given by

$$S_u := \bigcup_{v \preceq_{drl} u} vS.$$

This implies that when I is such that I^{hom} is Cohen-Macaulay and is in projective generic position then we can replace $\prec_{in}$ with $\prec_{drl}$ and H_u with H in the statements of Theorem 7.3.1 and Corollary 7.3.1. This now implies in particular:

Theorem 7.3.2. *Let $f_1, \ldots, f_c$ be a regular sequence in projective generic position of respective degrees $d_1, \ldots, d_c$ in $\mathbb{K}[\mathbf{z}, \mathbf{x}]$. Assume that the $\prec_{drl}$-Gröbner basis of $I = \langle f_1, \ldots, f_c\rangle$ is known and that the $\prec_{out}$-Gröbner basis G of $I \cdot \mathbb{K}(\mathbf{z})[\mathbf{x}]$ has coefficients which are rational functions with degrees at most δ in the numerators and denominators. Then, one can compute G up to precision 2δ using $\widetilde{O}\left(\delta^r c (d_1 \cdots d_c)^3\right)$ operations in $\mathbb{K}$.*

Remark 7.3.5. Let us close this section with a remark on the probability of $\mathfrak{m}$ being a point of good specialization in the situation of Theorem 7.3.2 if $\prec_{out} = \prec_{lex}$. If $\mathrm{gen}(I, \mathbf{z})$ is in shape position (Definition 3.2.1), then the reduced $\prec_{lex}$-Gröbner basis of $\mathrm{gen}(I, \mathbf{z})$ is of the form

$$\{g_c(\mathbf{z}, x_c), x_1 - g_1(\mathbf{z}, x_c), \ldots, x_{c-1} - g_{c-1}(\mathbf{z}, x_c)\}$$

with $g_c(\mathbf{z}, x_c) \in \mathbb{K}[\mathbf{z}, x_c]$ of total degree $D := d_1 \ldots d_c$. One can then show (Section 3.3, Proposition 2, Definition 1 and the following paragraph in Schost, 2003), that the degree of the lcm of the denominators

of the coefficients of $g_1, \ldots, g_{c-1}$ is bounded by D^2. Then, if $\mathbb{K} = \mathbb{F}_q$ for a prime power q, the Schwartz-Zippel lemma (Schwartz, 1980) implies that the probability of $\mathfrak{m}$ not being a point of good specialization is bounded by D^2/q which goes to zero as q increases.

7.4 USING TRACERS FOR GRÖBNER BASIS LIFTING

Again reusing the notation from Section 7.2, recall that Algorithm 20 requires the $\prec_{in}$-Gröbner bases H_u of the ideals $I + \mathfrak{m}_u$. In this section, we will show how these Gröbner bases can be computed non-naively when $\prec_{in}$ is a block order on $\mathrm{Mon}(\mathbf{x} \cup \mathbf{z})$ eliminating $\mathbf{x}$ using a similar idea to the one of Gröbner tracers given in Section 3.4.1. Non-naively means here, that when we want to compute H_u for usage in Algorithm 20, we have previously computed H_v for some $v \prec_{drl} u$. We can then again lift H_v to obtain H_u, which we show how to do here by combining the F_4 algorithm with Hensel lifting via the concept of tracers introduced in Section 3.4.1. Diagrammatically, the algorithm presented in this section can be combined with Algorithm 19 as follows:

$$\begin{array}{ccc} H_v & \xrightarrow{\text{lift via this section}} & H_u \\ \Big\downarrow{\scriptstyle \text{input of Algorithm 19}} & & \Big\downarrow{\scriptstyle \text{input of Algorithm 19}} \\ G_v & \xrightarrow[\text{lift via Algorithm 19}]{} & G_u \end{array}$$

Besides, the algorithm presented in this section can be used to compute Gröbner basis over fraction fields in its own right without performing arithmetic over the same function field or use of elimination orders as in Proposition 3.2.2.

Recall that under the additional assumption that $\mathfrak{m}$ is a point of good specialization for $\prec_{in}$ restricted to $\mathrm{Mon}(\mathbf{x})$, by Theorem 7.1.2, the reduced $\prec_{in}$-Gröbner basis of I_u is given by $H_u \cup M_u$, where H is the reduced $\prec_{in}$-Gröbner basis of $\mathrm{gen}(I, \mathbf{z})$ and M_u is the minimal generating set of $\mathfrak{m}_u$. If, in addition, the origin in the $\mathbf{z}$-space lies outside a suitably chosen Zariski-closed subset in the $\mathbf{z}$-space, then all coefficients that appear when computing H with an algorithm like F_4 lie in $\mathbb{K}[\mathbf{z}]_{\mathfrak{m}}$.

We will now roughly do the following:

(1) Suppose that $I = \langle F \rangle$ for a finite sequence $F \subset R$. Run F_4 on the system F mod $\mathfrak{m}$ and extract a tracer T out of this computation. After reduction, this F_4 run yields the Gröbner basis H_1, by Theorem 7.1.2.

(2) Inductively, suppose that we have computed the set H_u for $u \in \mathrm{Mon}(\mathbf{z})$ and let $v := \mathrm{next}(u)$.

(3) Apply T to the system $F \bmod \mathfrak{m}_v$, this yields the set H_v, again by Theorem 7.1.2. Note that computing the required echelonizations over the ring $\mathbb{K}[\mathbf{z}]/\mathfrak{m}_v$ is well defined: All coefficients in the appearing matrices are invertible modulo $\mathfrak{m}_v$ by the assumption above.

In the situation of Item (3), we previously applied T modulo $\mathfrak{m}_u$, this means that we have echelonized a series of Macaulay matrices with entries in $\mathbb{K}[\mathbf{z}]/\mathfrak{m}_u$. Now we have to echelonize the Macaulay matrices with the same row and column labels but with entries in $\mathbb{K}[\mathbf{z}]/\mathfrak{m}_v$ instead. Instead of recomputing these echelonizations from scratch, we will want to lift the previously obtained echelonizations from $\mathbb{K}[\mathbf{z}]/\mathfrak{m}_u$ to $\mathbb{K}[\mathbf{z}]/\mathfrak{m}_v$. Let us now show how to do this, when $\mathbf{z} = \{z\}$ consists of a single variable.

7.4.1 *Lifting of* LU*-Factorizations*

We assume now that $\mathbf{z} = \{z\}$ is a single variable. Let us denote $R := \mathbb{K}[\mathbf{z}]$ and $Q := \mathbb{K}(\mathbf{z})$. In this section, we want to lift an echelonization of a matrix, or in other words a LU decomposition, from the field $R/\mathfrak{m}$ to $R/\mathfrak{m}^{k+1}$ for some $k \geqslant 0$. As previously, we assume that the steps of the computation of this decomposition over $R/\mathfrak{m}$ are exactly those over Q projected onto $R/\mathfrak{m}$, in particular this implies that no nonzero entry $p/q \in Q$ with $p, q \in R$ is such that $p \notin \mathfrak{m}$ and $q \in \mathfrak{m}$.
Let $A \in Q^{n\times m}$ be a matrix with LU-factorization $A = LU$ with $L \in Q^{n\times n}$ and $U \in Q^{n\times m}$. We assume that A, L and U have coefficients in $R_{\mathfrak{m}}$. In that case we have matrices A_k, L_k and U_k with coefficients in $R/\mathfrak{m}^{k+1}$ and

$$A = A_k \bmod \mathfrak{m}^{k+1}, \quad L = L_k \bmod \mathfrak{m}^{k+1}, \quad U = U_k \bmod \mathfrak{m}^{k+1}.$$

We also have $A_k = L_k U_k \bmod \mathfrak{m}^{k+1}$. Now we want to lift L_k and U_k to $R/\mathfrak{m}^{k+2}$. This can be done as follows:

Theorem 7.4.1. *Given* A, L_0^{-1} *and* L_k, U_k *there is an algorithm which computes* L_{k+1} *and* U_{k+1} *in* $O\left(kmn^2\right)$ *arithmetic operations in* $\mathbb{K}$.

Proof. Recall that L_k is invertible and that $L_k^{-1} = L_0^{-1} \bmod \mathfrak{m}$. Thus,

$$\begin{aligned} A &= L_k U_k + z^{k+1}\delta_A \bmod \mathfrak{m}^{k+2} \\ &= L_k\left(U_k + z^{k+1}L_k^{-1}\delta_A\right) \bmod \mathfrak{m}^{k+2} \\ &= L_k\left(U_k + z^{k+1}L_0^{-1}\delta_A\right) \bmod \mathfrak{m}^{k+2}, \end{aligned}$$

and we define $B := U_k + z^{k+1}L_0^{-1}\delta_A$. The matrix δ_A is computed by computing the term of valuation $k+1$ of the product $L_k U_k$, this is done in $O\left(kmn^2\right)$ arithmetic operations. As $L_0^{-1} \bmod \mathfrak{m}$ has already been computed, $L_0^{-1}\delta_A$ is computed in $O\left(mn^2\right)$ arithmetic operations

in $\mathbb{K}$. It now suffices to prove that we can compute an LU-factorization of B over $R/\mathfrak{m}^{k+2}$ in $O\left(mn^2\right)$ operations in $\mathbb{K}$. Note that we can write

$$B = U_0 + zU_1' + \cdots + z^k U_k' + z^{k+1}V,$$

where

$$U_i' = \frac{1}{z^i}(U_i - U_{i-1})$$

is upper triangular with coefficients in $R/\mathfrak{m}$, for all $1 \leqslant i \leqslant k$, and $V = \frac{1}{z^{k+1}}(B - U_k) \in \mathbb{K}^{n\times n}$. Echelonizing B comes down to reducing the rows of V with the rows of B above. In particular, since $b_{i,1} = z^{k+1}v_{i,1}$ for $i > 1$, it can be reduced using the pivot $b_{1,1}$, which is invertible in $R/\mathfrak{m}^{k+1}$ by assumption. Thus, the row operation

$$\begin{aligned} b_{i,j} &\leftarrow b_{i,j} - z^{k+1}\frac{b_{i,1}}{b_{1,1}}b_{1,j} \bmod \mathfrak{m}^{k+2} \\ &= z^{k+1}(v_{i,j} - \frac{v_{i,1}}{u'_{0,1,1}}u'_{0,1,j} \bmod \mathfrak{m}) \end{aligned}$$

consists only in performing row operations on the layer of valuation $k+1$ in B. In other words, if

$$L' = \begin{bmatrix} 1 & & \\ -v_{2,1}/u'_{0,1,1}z^{k+1} & 1 & \\ \vdots & & \ddots \\ -v_{n,1}/u'_{0,1,1}z^{k+1} & & 1 \end{bmatrix}$$

then $L'B$ can be written

$$L'B = U_0' + zU_1' + \cdots + z^k U_k' + z^{k+1}V'$$

where we have $v'_{i,1} = 0$ for all $i \geqslant 2$. Proceeding with further row operations like this, we thus triangularize B. This clearly has the same cost as echelonizing a matrix in $\mathbb{K}^{n\times m}$, finishing the proof. □

Remark 7.4.1. Theorem 7.4.1 and its proof are also valid for $R = \mathbb{Z}$, $Q = \mathbb{Q}$, $\mathfrak{m} = p$ a good prime and $\mathbb{K} = \mathbb{F}_p$. We refer e.g. to Dixon (1982), Haramoto and Matsumoto (2009), and Pan (2011) for other methods in p-adic linear algebra.

7.4.2 *The Algorithm*

Analogous to Algorithm 6, we can now bake the lifting step sketched at the beginning of this section into an algorithm, this gives Algorithm 21. Instead of explicitly using a tracer, we lift directly the echelonizations of the considered Macaulay matrices, as outlined at the beginning of this section so as to utilize Theorem 7.4.1. We again assume that $\mathbf{z} = \{z\}$ is a single variable.

Algorithm 21 Gröbner Basis Lifting Using a Tracer

Input: A finite sequence of polynomials $F \subset R$, a monomial order $\prec_{in}$ on $\mathrm{Mon}(\mathbf{x})$, $k \in \mathbb{N}$

Output: The set $H_{z^{k+1}}$ as defined in this section

1: Run F4($F \bmod \mathfrak{m}, \prec$)
2: $M_1^{(0)} = L_{1,0}U_{1,0}, \ldots, M_r^{(0)} = L_{r,0}U_{r,0} \leftarrow$ The Macaulay matrices considered by this run together with their LU-factorizations
3: **for** $i = 1, \ldots, k$
4: $\quad G \leftarrow F \bmod \mathfrak{m}^{k+1}$
5: $\quad M_i^{(k)} \leftarrow$ the Macaulay matrix with the same row/column labels as $M_i^{(k-1)}$, built from G
6: $\quad U_{i,k} \leftarrow$ the upper triangular part of a LU-factorization of $M_i^{(k-1)}$ lifted from one of $M_i^{(k)}$ as in Theorem 7.4.1
7: $\quad$ Append all rows of $U_{i,k}$ with leading monomial different from the corresponding row of $M_i^{(k)}$ to G
8: **return** G

The termination and correctness of Algorithm 21 follows from the considerations at the beginning of this section.
Note that Theorem 7.4.1 sais that we can lift a LU-factorization from $\mathbb{K}[z]/\mathfrak{m}^k$ to $\mathbb{K}[z]/\mathfrak{m}^{k+1}$ at k times the same cost as computing the corresponding LU-factorization over $\mathbb{K}[z]/\mathfrak{m} \cong \mathbb{K}$. We obtain:

Corollary 7.4.1. *Suppose that, in the context of Algorithm 21, the Gröbner basis H_z of the image of $I = \langle F \rangle$ in $\mathbb{K}[z, \mathbf{x}]/\mathfrak{m} \cong \mathbb{K}[\mathbf{x}]$ can be computed in $\mathcal{A}$ arithmetic operations by the F_4 algorithm. Then Algorithm 21 computes the Gröbner basis $H_{z^{k+1}}$ in $O\left(k^2\mathcal{A}\right)$ operations.*

Remark 7.4.2. We investigated the complexity of Algorithm 21 only with a linear lifting strategy, i.e. when we lift from modulo $\mathfrak{m}^k$ to modulo $\mathfrak{m}^{k+1}$. A question for future research is whether the complexity in Corollary 7.4.1 becomes quasi-linear in k when a quadratic lifting strategy is chosen, i.e. when we lift from modulo $\mathfrak{m}^k$ to modulo $\mathfrak{m}^{k^2}$ as in Section 7.3.2.

7.5 BENCHMARKS

In this section, we provide benchmarks for a proof-of-concept implementation of Algorithm 20. These benchmarks are recorded in Table A.5.
We first give a brief description of our implementation. This implementation is written using `OSCAR`. All required Gröbner basis computations use `msolve` via its `Julia`-interface `AlgebraicSolving.jl` or `Groebner.jl` (Demin and Gowda, 2023), also written in `Julia`. The implementation is available at

`https://gitlab.lip6.fr/mohr/genfglm`

For the recorded benchmarks, the following computations were performed, keeping the notation Section 7.2 and Section 7.3:

(1) Compute a $\prec_{drl}$-Gröbner basis for the polynomial ideal I in question.

(2) Use this $\prec_{drl}$-Gröbner basis to compute a maximally independent set of variables modulo I (following Proposition 3.2.1), this gives us the partition of the variables into the subsets $\mathbf{x}$ and $\mathbf{z}$ as above.

(3) If $\mathbf{z} = \{z_1, \ldots, z_{n-c}\}$, choose random $a_1, \ldots, a_{n-c} \in \mathbb{K}$ and make the coordinate substitution $z_i \leftarrow z_i - a_i$.

(4) Choose $\prec_{in}$ as the block order on $\mathrm{Mon}(\mathbf{z}, \mathbf{x})$ eliminating $\mathbf{x}$ with $\prec_{drl}$ on both blocks of variables.

(5) If $\mathbf{x} = \{x_1, \ldots, x_c\}$, choose $\prec_{out}$ as a block order on $\mathrm{Mon}(\mathbf{x})$ eliminating $\mathbf{x}' := \{x_1, \ldots, x_{c-1}\}$ with $\prec_{drl}$ on $\mathbf{x}'$ and the total order by degree on $\{x_c\}$.

(6) By the elimination property of block orders (Theorem 3.1.1), the target Gröbner basis G contains a single polynomial g_c in the univariate polynomial ring $\mathbb{K}(\mathbf{z})[x_c]$.

(7) Use Algorithm 20 to compute only the polynomial g_c, ignoring the rest of the set G. Note that this is indeed possible, in Line 6 of Algorithm 20 we may choose which of the elements in G_{lift} to actually keep and lift and ignore the rest.

This computation is motivated by the method outlined in Section 3.2.2: If the considered generic fiber is in shape position w.r.t. x_c (see Definition 3.2.1) then we can use the computed polynomial g_c to compute some of the irreducible components of $\mathbf{V}(I)$ where I is the polynomial ideal under study.
All computations were performed with $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ where p was a randomly chosen prime of 16 bits.
We compared the time this computation took with the computation of the set G using `msolve` and which, in this case, just runs the F_4 algorithm with a suitable block order on $\mathrm{Mon}(\mathbf{z}, \mathbf{x})$, following Proposition 3.2.2. All computations were performed on an Intel Xeon Gold 6244 CPU @ 3.60 GHz with 1.5 TB of memory.
On most small examples in Table A.5, we achieve a comparable timing with `msolve`, with the exception of PS(2,12). On the larger example RD(5) we have a small improvement, while on ED(3,3) we achieve a much better timing. We point out the preliminary nature of our implementation and thus the preliminary nature of the benchmarks recorded here:

(1) The main step in Algorithm 20, Algorithm 19, was implemented naively, close to the provided pseudocode, i.e. without the use

of multiplication tensors to compute normal forms as described in Section 7.3.

(2) Algorithm 21 is not yet implemented, the input Gröbner bases H_u required by Algorithm 20 where computed naively, i.e. simply by running `msolve` on the ideals $I + \mathfrak{m}_u$ where I is the polynomial ideal corresponding to the example in question.

(3) We observed that the linear systems appearing in Algorithm 19 tend to be sparse. Yet, in order to use `OSCAR`'s black box linear system solving facilities, we work with dense matrices in our implementation. This occurs additional overhead due to `OSCAR`'s data structure for finite field elements, which even made the memory allocation for the required matrices a burdensome task.

We observed that the "naive" computation of the Gröbner bases as in Item (2) formed a significant bottleneck in our computatios. We therefore expect the timings of our computation to improve a lot once this step is performed using Algorithm 21.

8

GRÖBNER BASES OF GENERIC FIBERS AND WHITNEY STRATIFICATIONS

Let $R := \mathbb{Q}[\mathbf{x}]$ where $\mathbf{x} := \{x_1, \ldots, x_n\}$. Our goal in this chapter is to give an algorithm which computes a Whitney stratification (Definition 2.5.3) of a given algebraic set $X := \mathbf{V}(F) \subset \mathbb{A}^n$ where $F \subset R$ is a finite set and to show how to computationally minimize a given Whitney stratification. We assume throughout this chapter that X is equidimensional. We assume further that throughout this chapter every occuring algebraic set Y is defined over $\mathbb{Q}$ and computationally represented by a Gröbner of an ideal J s.t. $\mathbf{V}(J) = Y$. We remark, at the relevant points, how the necessary computations can be performed using Gröbner bases but present our pseudocode at a high level, without a focus on concrete data structures. In this chapter, we will in particular have to compute certain Gröbner basis of generic fibers of ideals. Note that these computations can be made with the algorithms presented in Chapter 7.
The contents of this chapter are based on the results given in Helmer and Mohr (2024).

8.1 THE ALGORITHMS

8.1.1 *Computing a Whitney Stratification*

The foundational theorem for the algorithm to compute Whitney stratifications in Helmer and Nanda (2023) is Theorem 2.5.1. When applied immediately, this requires the computation of primary decompositions of fibers in the conormal space (Definition 2.5.4) $\mathrm{Con}(X)$ associated to X. These fibers are often much more complicated than their image, making the computation of the required associated primes difficult even for relatively simple algebraic sets, see Example 1.0.4. In line with this observation, practical experience has shown that the computation of these associated primes is precisely the main bottleneck of this algorithm. We note that, theoretically speaking, for the ideals involved it is absolutely essential that we obtain both the embedded and the minimal primes. In particular, algorithms which compute only the isolated primes of an ideal cannot be used.
In this chapter we adress this challenge by showing how Theorem 2.5.1 can be exploited without having to compute associated primes, by computing Gröbner bases of certain generic fibers instead. This turns out to yield a considerable speedup in practice. To this end, we modify Theorem 2.5.1 as follows, note the similarity to Proposition 3.2.3:

Theorem 8.1.1. *Let* $Y \subset \mathrm{Sing}(X)$. *Choose any MIS of variables* $\mathbf{u} \subset \mathbf{x}$ *of any defining ideal* I_Y *of* Y. *Let* $J := \mathbf{I}(\mathrm{Con}(X)) + I_Y$. *Let* G *be a minimal Gröbner basis of* $J\mathbb{C}(\mathbf{u})[\mathbf{x} \setminus \mathbf{u}, \mathbf{y}]$ *w.r.t. any monomial order* $\prec$ *with* $G \subset \mathbb{C}[\mathbf{x}, \mathbf{y}]$. *Let*

$$h = \mathrm{lcm}\{\mathrm{lc}_\prec(g) \mid g \in G\} \in \mathbb{C}[\mathbf{u}].$$

Then

(1) $Y \setminus \mathbf{V}(h)$ *is equidimensional of dimension* $\dim(Y)$.

(2) The pair $(X, Y \setminus \mathbf{V}(h) \cup \mathrm{Sing}\, Y)$ *satisfies Whitney's condition (B).*

Proof. (*Proof of Item (1)*) By Proposition 3.2.3, we have

$$(J : h^\infty) = J\mathbb{C}(\mathbf{u})[\mathbf{x} \setminus \mathbf{u}, \mathbf{y}] \cap \mathbb{C}[\mathbf{x}, \mathbf{y}]. \tag{8.1}$$

The ring $\mathbb{C}(\mathbf{u})[\mathbf{x} \setminus \mathbf{u}, \mathbf{y}]$ is the localization of $\mathbb{C}[\mathbf{x}, \mathbf{y}]$ at the multiplicative set $\mathbb{C}[\mathbf{u}] \setminus \{0\}$. Hence, using Theorem 3.1 in Eisenbud (1995),

$$\mathrm{Ass}((J : h^\infty)) = \{P \in \mathrm{Ass}(J) \mid P \cap \mathbb{C}[\mathbf{u}] = 0\}.$$

Further, clearly, $(J : h^\infty) \cap \mathbb{C}[\mathbf{x}] = (I_Y : h^\infty)$. For P a minimal prime over $(I_Y : h^\infty)$, choose a minimal prime Q over $(J : h^\infty)$ with $Q \cap \mathbb{C}[\mathbf{x}] = P$. Then we have

$$0 = Q \cap \mathbb{C}[\mathbf{u}] = P \cap \mathbb{C}[\mathbf{u}]$$

and therefore $\dim(P) \geqslant \dim(Y)$, this follows easily from the definition of Krull dimension of an ideal. On the other hand, P is also minimal over I_Y. Therefore $\dim(P) \leqslant \dim(Y)$ and finally $\dim(P) = \dim(Y)$. This shows that $Y \setminus \mathbf{V}(h)$ is equidimensional of dimension equal to $\dim(Y)$.
(*Proof of Item (2)*): Let $P \in \mathrm{Ass}(J)$ such that $\dim(\kappa_X(\mathbf{V}(P))) < \dim(Y)$. This implies again $P \cap \mathbb{C}[\mathbf{u}] \neq 0$, therefore $P \notin \mathrm{Ass}(J\mathbb{C}(\mathbf{u})[\mathbf{x} \setminus \mathbf{u}, \mathbf{y}])$ and therefore also $P \notin \mathrm{Ass}((J : h^\infty))$. Hence necessarily $h \in P$ or, since $h \in \mathbb{C}[\mathbf{u}]$, $\kappa_X(\mathbf{V}(P)) \subset \mathbf{V}(h)$. If then

$$A := \bigcup_{\substack{P \in \mathrm{Ass}(J) \\ \dim(\kappa_X(\mathbf{V}(P)) < \dim(Y)}} \kappa_X(\mathbf{V}(P))$$

then $A \subset \mathbf{V}(h)$. By Theorem 2.5.1, the pair $(X, Y \setminus A \cup \mathrm{Sing}\, Y)$ satisfies Whitney's condition (B) and since $Y \setminus \mathbf{V}(h) \subset Y \setminus A$, so does $(X, Y \setminus \mathbf{V}(h) \cup \mathrm{Sing}\, Y)$. □

To now describe our algorithm based on this theorem, let us briefly recall how to compute an ideal defining the conormal space of a given X (see also Section 4.1 in Helmer and Nanda, 2023). From our sequence $F := (f_1, \ldots, f_r)$ cutting out X, we build the augmented Jacobian matrix

$$\mathbf{J}_\mathbf{y}(F) := \begin{bmatrix} y_0 & y_1 & \ldots & y_{n-1} \\ \partial_1 f_1 & \partial_2 f_1 & \ldots & \partial_n f_n \\ \vdots & & & \vdots \\ \partial_1 f_r & \partial_2 f_r & \ldots & \partial_n f_r \end{bmatrix}$$

with new variables $\mathbf{y} := \{y_0, \ldots, y_{n-1}\}$. Now

Proposition 8.1.1 (see Section 4.1 in Helmer and Nanda, 2023)**.** *Let* $c := \operatorname{codim}(X)$*, let* M *be the set of* c*-minors of* $\mathbf{J}(F)$ *and let* $M_{\mathbf{y}}$ *be the set of* $c+1$*-minors of* $\mathbf{J}_{\mathbf{y}}(F)$*. Then*

$$\operatorname{Con}(X) = \mathbf{V}\left(\left(\langle F\rangle + \langle M_{\mathbf{y}}\rangle : M^{\infty}\right)\right).$$

Remark 8.1.1. Recall that the necessary saturation to compute conormal spaces can be performed using Gröbner basis computations with block orders, see Lemma 3.1.2 and Lemma 2.2.2.

Now we can give the following algorithm WhitPoints (Algorithm 22), which given any pair of algebraic sets X and Y with $Y \subset \operatorname{Sing} X$, returns a polynomial h as in Theorem 8.1.1. This algorithm is the centerpiece of our algorithm for computing Whitney stratifications. As previously mentioned, we avoid the computation of associated primes in the conormal space of X which an immediate application of Theorem 2.5.1 would require.

Algorithm 22 Computation of h as in Theorem 8.1.1

Input: An equidimensional algebraic set $X \subset \mathbb{A}^n$, any closed $Y \subset \operatorname{Sing}(X)$

Output: An element $h \in \mathbb{C}[\mathbf{u}]$ as in Theorem 8.1.1

1: **procedure** WhitPoints(X, Y)
2: $I_X \leftarrow$any ideal defining X, $I_Y \leftarrow$any ideal defining Y
3: $\mathbf{u} \leftarrow$any maximally independent subset of variables of I_Y
4: $\prec \leftarrow$any monomial ordering eliminating $(\mathbf{x} \cup \mathbf{y}) \setminus \mathbf{u}$
5: $G \leftarrow$a $\prec$-Gröbner basis of $\mathbf{I}(\operatorname{Con}(X)) + I_Y$
6: Minimize G over $\mathbb{C}(\mathbf{u})[\mathbf{x} \setminus \mathbf{u}, \mathbf{y}]$
7: **return** $h := \operatorname{lcm}\{\operatorname{lc}(g) \mid g \in G\}$

Remark 8.1.2. In line 2, we use a Gröbner basis of I_Y and Proposition 3.2.1 to compute the desired maximally independent subset of variables of I_Y. Note further that we could also use the algorithm presented in Section 7.4 to compute the required Gröbner basis of our generic fiber.

Theorem 8.1.2. *For a given pair of algebraic sets* X *and* $Y \subset \operatorname{Sing} X$*, the output polynomial* h *of* WhitPoints(X, Y) *is such that the pair* $(X, Y \setminus \mathbf{V}(h))$ *satisfies Whitney's condition (B).*

Proof. Note that, thanks to Proposition 3.2.2, a Gröbner basis of an ideal $I \subset \mathbb{C}[\mathbf{x}]$ w.r.t. a monomial order eliminating $\mathbf{x} \setminus \mathbf{u}$, where $\mathbf{u} \subset \mathbf{x}$, is also a Gröbner basis of $I\mathbb{C}(\mathbf{u})[\mathbf{x} \setminus \mathbf{u}]$. The stated result then follows from Theorem 8.1.1. □

Finally, we can use this routine to give the new algorithm Whitney (Algorithm 23) below to compute a Whitney stratification of an affine equidimensional algebraic set X. In this algorithm, each occurring algebraic set Z is represented as a union its $\mathbb{Q}$-irreducible components $Z = \bigcup_{i=1}^{r} Z_i$. Let us define three subroutines used by this algorithm:

- We define Components(X) to return the set $\{Z_1, \ldots, Z_r\}$ (see e.g. Section 3.2.2 for an algorithm to compute Gröbner bases cutting out the Z_i).
- For two algebraic sets Z, W we define the routine Add(Z, W) to return $Z \cup W$ represented again by $\mathbb{Q}$-irreducible algebraic sets, i.e. by Components$(Z) \cup$ Components(W).
- For a flag $W_\bullet$ and a an irreducible algebraic set Z we then define Update$(W_\bullet, Z)$ to change $W_{\dim(Z)}$ to Add$(W_{\dim(Z)}, Z)$.

Remark 8.1.3. Note that, as Theorem 8.1.1 does not rely on Y being irreducible, it is not strictly needed for the correctness of our algorithm to represent the occuring algebraic sets by their $\mathbb{Q}$-irreducible components. We chose to present the algorithm like this here for ease of reading. It would not be difficult to adapt Algorithm 2 to our situation. However, as remarked above, the key contribution of this work is to avoid the computation of associated primes *in conormal spaces*. Even with our new method, we observed that the required computations in conormal spaces are still the bottleneck of our algorithm, compared to representing algebraic sets by their irreducible components as above.

Algorithm 23 Computation of a Whitney Stratification

Input: An equidimensional algebraic set $X \subset \mathbb{A}^n$
Output: A Whitney stratification of X

```
1:  procedure Whitney(X)
2:      d ← dim(X)
3:      W_0 ← ∅, ..., W_{d-1} ← ∅, W_d ← X
4:      for i from d to 0
5:          Z_1, ..., Z_r ← Components(W_i)
6:          for j, k from 1 to r with j < k
7:              Update(W_•, Z_j ∩ Z_k)
8:          for Z in Components(W_i)
9:              Update(W_•, Sing(Z))
10:             for j from d − 1 to 0
11:                 for Y in Components(W_j) if Y ⊂ Z
12:                     h ← WhitPoints(Z, Y)
13:                     Y' ← Y ∩ V(h)
14:                     Update(W_•, Y')
15:     return W_•
```

Theorem 8.1.3. *For a given equidimensional affine algebraic set $X \subset \mathbb{A}^n$,* Whitney(X) *terminates and outputs a Whitney stratification of X.*

Proof. The termination of the algorithm is clear. Note that the singular locus of any algebraic set Y consists of the the union of the singular loci of its $\mathbb{Q}$-irreducible components together with the intersections of

all its $\mathbb{Q}$-irreducible components: This follows because the local ring $(R/\mathbf{I}(Y))_p$ at a smooth point $p \in Y$ is a regular local ring (Theorem 16.19 in Eisenbud, 1995) and as such in particular an integral domain (Corollary 10.10 in Eisenbud, 1995) and so p can only be contained in one irreducible component of Y. Thus Line 7 and Line 9 in WHITNEY garantuee the smoothness of all strata of the output of WHITNEY(X). Further, the fact that Whitney's condition (B) is satisfied for all pair of the strata of the output of WHITNEY(X) follows immediately from Theorem 8.1.2. □

8.1.2 *Minimizing a Whitney Stratification*

Our first goal here is to show how to compute, for a given $X \subset \mathbb{A}^n$ and a generic point p in an algebraic subset $Y \subset X$, the sequence $m_\bullet(X, p)$ without being given p explicitly. Since, computationally, we are given Y only by a Gröbner basis of an ideal cutting out Y we cannot explicitly construct points in Y using symbolic methods.
Next, we will show that the sequence $m_\bullet(X, \bullet)$ is constant on a Zariski-open subset of Y, if Y is $\mathbb{Q}$-irreducible. These two things combined allow us to compute, probabilistically, the "generic" multiplicities $m_\bullet(X, p)$, $p \in Y$. Combining this with Theorem 2.5.2 we obtain a procedure to produce the unique minimal Whitney stratification of a complex algebraic set given a Whitney stratification computed by Algorithm 23.
Suppose for the moment that $Y = \mathbf{V}(f_1, \ldots, f_c)$ where $c \leqslant n$ and $f_1, \ldots, f_c$ is such that $F := (f_1, \ldots, f_c, \ell_{c+1}, \ldots, \ell_n)$ is a *reduced regular sequence*, where the ℓ_i are generic degree one polynomials in $\mathbf{x}$, i.e. the sequence F is a regular sequence and the determinant of $\mathbf{J}(F)$ is non-zero at every point of Y.
Let now

$$\begin{aligned} \phi : \mathbb{A}^n &\to \mathbb{A}^n \\ p := (p_1, \ldots, p_n) &\mapsto (f_1(p), \ldots, f_c(p), \ell_{c+1}(p), \ldots, \ell_n(p)), \end{aligned}$$

and let $\tilde{X}$ and $\tilde{Y}$ be the closures of the images under ϕ of X and Y respectively. Denote by $\mathbf{z} := \{z_1, \ldots, z_n\}$ the coordinates on the codomain of ϕ. Note that the origin $0 \in \mathbb{A}^n$ lies in $\tilde{Y}$. Now we have

Proposition 8.1.2. *Using the notation above, for any $y \in \phi^{-1}(0)$ we have*

$$m_\bullet(X, y) = m_\bullet(\tilde{X}, 0).$$

Proof. Since F is a reduced regular sequence, ϕ defines a local (analytic) isomorphism near any $p \in Y$ by the inverse function theorem. For a finite set of n variables $\mathbf{x}$ denote by $\mathbb{C}\{\mathbf{x}\}_p$ the ring of formal power series in $\mathbf{x}$ centered at p with coefficients in $\mathbb{C}$. Now let

$$\begin{aligned} \phi_p^* : \mathbb{C}\{\mathbf{z}\}_{\phi(p)}/\mathbf{I}(\tilde{X}) &\to \mathbb{C}\{\mathbf{x}\}_p/\mathbf{I}(X) \\ g &\mapsto g \circ \phi. \end{aligned}$$

Because the local inverse of ϕ is also analytic, ϕ_p^* is an isomorphism. Now, the multiplicity of a local ring $(S, \mathfrak{m})$ depends only on the *associated graded ring* $\mathrm{gr}_{\mathfrak{m}}(R)$ (see Section 5.1 in Eisenbud, 1995 for a definition of associated graded rings and p. 274 therein). Now recall also that $\mathrm{gr}_{\mathfrak{m}}(S) = \mathrm{gr}_{\mathfrak{m}}(\widehat{S})$, where $\widehat{S}$ is the $\mathfrak{m}$-*adic completion* of S (see p. 181 in Eisenbud, 1995 for a definition of completions and Corollary 7.13 therein).
We finish the proof by noting that for an ideal $J \subset \mathbb{C}[\mathbf{x}]$, $\mathbb{C}\{\mathbf{x}\}_p/I$ is the $\mathbf{I}(p)$-adic completion of $\mathbb{C}[\mathbf{x}]_p$. □

Next suppose that $Y \subset X$ is equidimensional of codimension c but no longer necessarily cut out by a reduced regular sequence. Choose, at random, a sequence $f_1, \ldots, f_c \in \mathbf{I}(Y)$, for example as random linear combinations of given generators of $\mathbf{I}(Y)$. Defining $Z := \mathbf{V}(f_1, \ldots, f_c)$ we then have the following

Proposition 8.1.3. *Use the notations above. If $f_1, \ldots, f_c$ are sufficiently generic, there exists a Zariski-open subset $U \subset \mathbb{A}^n$ such that*

$$Y \cap U = Z \cap U$$

and $Y \cap U$ is Zariski dense in Y. Moreover, for sufficiently generic degree one polynomials $\ell_{c+1}, \ldots, \ell_n$, the sequence

$$F := (f_1, \ldots, f_c, \ell_{c+1}, \ldots, \ell_n)$$

is a reduced regular sequence on Y.

Proof. Since $f_1, \ldots, f_c$ is sufficiently generic, $f_1, \ldots, f_c$ defines a local regular sequence in $\mathbf{I}(Y)$, of maximal length, this follows e.g. from Lemma 3 in Jeronimo and Sabia (2002). Then, by Corollary 8 in Decker, Greuel, and Pfister (1999) we have for $Z := \mathbf{V}(f_1, \ldots, f_c)$ that

$$Y = \overline{Z \setminus \overline{Z \setminus Y}}.$$

Hence, choosing $U := \mathbb{A}^n \setminus \overline{Z \setminus Y}$ proves the first part of the proposition.
Now, at every point $p \in U$, we have $\mathbf{I}(Y)\mathbb{C}[\mathbf{x}]_p = \langle f_1, \ldots, f_c \rangle \mathbb{C}[\mathbf{x}]_p$. This shows that $f_1, \ldots, f_c$ defines a radical ideal at every point $p \in U$. By Exercise 12.11 in Eisenbud (1995), the same remains true for the sequence $F := (f_1, \ldots, f_c, \ell_{c+1}, \ldots, \ell_n)$ where the ℓ_i are sufficiently generic degree one polynomials in $\mathbf{x}$. This means that the Jacobian determinant of F is nonzero at every $p \in U$, by Corollary 7 in Decker, Greuel, and Pfister (1999), proving the proposition. □

From this we finally obtain the following corollary.

Corollary 8.1.1. *Let $m_\bullet(\tilde{X}, 0)$ be the multiplicity sequence constructed as above, with Y replaced by Z. Then $m_\bullet(\tilde{X}, 0) = m_\bullet(X, p)$ for every $p \in \phi^{-1}(0)$.*

Proof. Choose $U \subset \mathbb{C}^n$ as in Proposition 8.1.3. By the same proposition, with probability 1, the intersection $Y \cap \mathbf{V}(\ell_{c+1}, \ldots, \ell_n) \cap U \subset \phi^{-1}(0)$ is not empty. For any choice of $p \in Y \cap \mathbf{V}(\ell_{c+1}, \ldots, \ell_n) \cap U$ we then have $m_\bullet(\tilde{X}, 0) = m_\bullet(X, p)$ by Proposition 8.1.2. □

Now, given an algebraic set X and an algebraic subset Y, we can compute the sequence $m_\bullet(X, p)$ for a random point $p \in Y$ as follows:

Algorithm 24 Computing the sequence $m_\bullet(X, p)$

Input: An algebraic set $X \subset \mathbb{A}^n$, any equidimensional closed $Y \subset X$.
Output: $m_\bullet(X, p)$ for a random point p.

1: **procedure** MULT(X, Y)
2: $g_1, \ldots, g_r \leftarrow$ generators of $\mathbf{I}(Y)$
3: $c \leftarrow \operatorname{codim}(Y)$
4: Choose $f_1, \ldots, f_c$ as random linear combinations of the g_i
5: Choose $\ell_{c+1}, \ldots, \ell_n$ as random degree one polynomials
6: $\phi \leftarrow$ the map given by $\mathbf{x} \mapsto (f_1, \ldots, f_c, \ell_{c+1}, \ldots, \ell_n)$
7: Compute ideals defining $\delta_0(\phi(X), 0), \ldots, \delta_{\dim(X)-1}(\phi(X), 0)$
8: **return** $(m_0(\delta_0(\phi(X), 0)), \ldots, m_0(\delta_{\dim(X)-1}(\phi(X), 0)))$

Remark 8.1.4. Let us make the computational steps in Algorithm 24 a bit more explicit: Given generators for an ideal defining Y, we can compute generators of the radical ideal $\mathbf{I}(Y)$ in Line 2 using any of the algorithms in Section 2.1 of Decker, Greuel, and Pfister (1999).
In Line 7, an ideal defining $\phi(X)$ is again computed using Gröbner basis computations with block orders, see e.g. Proposition 15.30 in Eisenbud (1995). Using the notation of Proposition 8.1.1, ideals defining the necessary local polar varieties can then be computed as ideals of the form

$$\left(I(\phi(X)) + \langle M_y \rangle + L : M^\infty\right) \cap \mathbb{C}[\mathbf{x}]$$

where M_y are the suitable minors of the augmented Jacobian matrix associated to generators of $I(\phi(X))$, M are the suitable minors of the Jacobian matrix of generators of $I(\phi(X))$ and L is a collection of randomly chosen degree one forms, defining a linear space of suitable dimension in $\mathbb{P}^{n-1}$.
In Line 8, to compute the required multiplicities we use degree computations via Gröbner bases as described in Harris and Helmer (2019) and implemented in the `SegreClasses` `Macaulay2` package, see in particular Theorem 5.3 in Harris and Helmer (2019). Alternatively one can use standard basis computations, see e.g. Sayrafi (2017) for details. Finally, we remark that, to compute the multiplicities in Line 8 correctly, we require the ideals cutting out our local polar varieties to be radical. This is generically ensured once we work with the radical ideal $\mathbf{I}(\tilde{X})$: Then the ideal defining $\mathrm{Con}(\tilde{X})$ following Proposition 8.1.1 is also radical (see Proposition 2.9 in Flores and Teissier, 2017), and then the ideals defining our local polar varieties are also radical for

generic choices of linear subspace, see Remark 3.10 in Flores and Teissier (2017).

Now we have the ability to (probabilistically) minimize a given Whitney stratification. For a given algebraic set $X \subset \mathbb{C}^n$, the output of WHITNEY(X) consists of a flag

$$W_0 \subset W_1 \subset \cdots \subset W_d = X$$

on X. Each W_i is given by its $\mathbb{Q}$-irreducible components $W_{i1}, \ldots, W_{ir_i}$. Let us define $W_{00} := X$. These W_{ij} may now be organized in a tree as follows: Each node of this tree contains one of the W_{ij}. The children of a node containing W_{ij} are given by those $W_{i+1,k}$ with $W_{i+1,k} \subset W_{ij}$. Let us call such a data structure a *Whitney tree* of X. Note that such a tree may be extracted from the output of WHITNEY(X) by ideal containment checks using Gröbner bases and normal form computations, here we suppose that every occuring irreducible algebraic set is given by a Gröbner basis of the radical ideal cutting it out. From such a tree, we may now minimize a given Whitney stratification as follows:

Algorithm 25 Minimizing a Whitney Stratification

Input: A Whitney tree $\mathcal{T}_X$ on an algebraic set X.
Output: A minimal Whitney stratification of X.

1: **for** each node W in $\mathcal{T}_X$
2: $\quad P \leftarrow \{(W, Z, Y) \mid Z, Y$ are nodes of $\mathcal{T}_X$ with $W \subset Z \subset Y\}$
3: $\quad$ **if** MULT$(Y, W) =$ MULT(Y, Z) for all $(W, Z, Y) \in P$
4: $\quad\quad$ Delete the node W from $\mathcal{T}_X$
5: **for** each level i of $\mathcal{T}_X$
6: $\quad W_{d-i} \leftarrow$ union of all components in level i
7: **return** $W_\bullet$

Lemma 8.1.1. *Algorithm 25 is correct and terminates.*

Proof. The termination is clear. Suppose that we have a triple of algebraic sets $W \subset Z \subset Y$ with W and Z irreducible and that for generic $w \in W$ and $z \in Z$ we have $m_\bullet(Y, w) = m_\bullet(Y, z)$. Then there exists a Zariski-dense subset U of Z such that $U \cap W \neq \emptyset$ and such that $m_\bullet(Y, \bullet)$ is constant on U. Hence the points in W at which the pair (Y, Z) does not satisfy Whitney's condition (B) have at least codimension one in W by Theorem 2.5.2. In the situation of WHITNEYMINIMIZE, W can thus be removed without destroying the property of the output flag $W_\bullet$ being a Whitney stratification. This proves the correctness. □

Remark 8.1.5. Note that we may also combine WHITNEYMINIMIZE directly with WHITPOINTS: For given X and irreducible Y let h be the output of WHITPOINTS(X, Y) and let $A := Y \cap \mathbf{V}(h)$. We know that the set of points in Y where Whitney's condition (B) fails to hold with respect to X and Y is contained in A. We may then compute the

irreducible components $A_1, \ldots, A_r$ of A and compare for each A_i the output of $\text{MULT}(X, Y)$ with $\text{MULT}(X, A_i)$. If these sequences are equal we recursively go through the same procedure with X and A_i. If they are not equal we append A_i to a list of output components which we return in the end. This is potentially more optimal then minimizing after the stratification is finished because it prevents a build up of unneeded algebraic sets.

8.2 BENCHMARKS

In this section we collect some runtime comparisons of the new Whitney stratification algorithm described in this chapter with the algorithm of Helmer and Nanda (2023) for a variety of examples. These runtimes are collected in Table A.6, we record in addition whether the algorithms produced a minimal Whitney stratification and the time it took to minimize the Whitney stratification produced by Algorithm 23. The implementation of our Algorithm used to create these benchmarks is available in version 2.11, and above, of the `WhitneyStratifications` Macaulay2 package, which is available on the website of the first author of the paper corresponding to this chapter at:

`http://martin-helmer.com/Software/WhitStrat/WhitneyStratifications.m2`.

With this Macaulay2 package loaded and given a polynomial ideal `I` the implementation of Algorithm 23 is called with `whitneyStratify(I, AssocPrimes=>false)`, while the algorithm of Helmer and Nanda (2023) is called with `whitneyStratify(I, AssocPrimes=>true)`.
The computations of the Gröbner bases of generic fibers needed to apply Theorem 8.1.1 are performed as in the pseudocode of Algorithm 22, i.e. without using the algorithms presented in Chapter 7.

9

PERSPECTIVES

We close this thesis by outlining some research perspectives following from the content of Chapter 5, Chapter 6 and Chapter 7.

FURTHER INCORPORATION OF WITNESS SETS IN ALGORITHM 15

In the context of Chapter 5, where Algorithm 15 is developed, recall that we frequently perform the following operation: Given an equidimensional affine cell X and $h \in R$, compute a Gröbner basis of the ideal $(I_X : h^\infty)$ where I_X was such that $\overline{X} = \mathbf{V}(I_X)$.
Experimentally, when $\dim(X) > 0$, this can be significantly harder than computing a Gröbner basis of I_X. To better understand this behavior, let us briefly explain how a saturation computation proceeds. Suppose $I_X = \langle f_1, \ldots, f_r \rangle$. Then, during a Gröbner basis computation for $I_X + \langle f \rangle$, every occuring polynomial p is of the form

$$p = a_1 f_1 + \cdots + a_r f_r + af.$$

Algorithm 10 gives us efficient means to track just the element a for each p as above during such a computation. Whenever $p = 0$, we may then insert a into the current intermediate Gröbner basis and proceed, at the end we obtain a Gröbner basis of the ideal $(I_X : f^\infty) + \langle f \rangle$. Even when we use Lemma 3.1.2 to compute $(I_X : f^\infty)$, i.e. when we compute a Gröbner basis of $I_X + \langle tf - 1 \rangle$ for a new variable t w.r.t. a monomial order eliminating t, the computation will behave similarly: an element p as above will correspond to the element $tp - a$ in this computation. Additional overhead is therefore incurred, because we are computing a Gröbner basis of $(I_X : f^\infty) + \langle f \rangle$ and not only for $(I_X : f^\infty)$. We refer also to Berthomieu, Eder, and Safey El Din (2023) where a new saturation algorithm based on F_4 is designed which faces similar issues when X is positive-dimensional.
This overhead tends however not to be a problem when $\dim(X) = 0$: Then $(I_X : h^\infty) + \langle h \rangle = 1$, which means that our Gröbner basis computation is expected to terminate as soon as a Gröbner basis of $(I_X : h^\infty)$ is identified.
This is precisely the reason for the introduction of Witness sets in Chapter 5, which reduces the necessity of computing positive dimensional saturation ideals: Instead of working with the affine cell X we work with the zero-dimensional affine cell $X \cap L$ where L is a generic linear space of codimension equal to $\dim(X)$. However, in Chapter 5, this did not eliminate completely the necessity to compute Gröbner basis of ideals of the form $(I_X : h^\infty)$.

We could design a similar algorithm to Algorithm 15 using *only* witness sets of affine cells (i.e. eliminating the necessity to work directly with ideals of the form I_X) if one is able to answer the following

Question 1. Given:

(1) a finite set of polynomials F,

(2) d degree one polynomials L and a Gröbner basis of $(\langle F\rangle : h^\infty) + \langle L\rangle$ s.t. $X := \mathbf{V}(F) \setminus \mathbf{V}(h)$ is equidimensional of dimension d (note that h is *not* given) and

(3) another polynomial f regularly intersecting X,

compute a Gröbner basis of the zero-dimensional ideal $(\langle F\rangle + \langle f\rangle : h^\infty) + \langle L \setminus \{\ell\}\rangle$ where $\ell \in L$ is chosen at will.

We refer to Lecerf (2003) where a similar question is treated in the context of geometric resolutions and to Duff, Leykin, and Rodriguez (2022) where a similar question is treated in the context of numerical algebraic geometry.
A potential answer to this question lies in the methods developed in Chapter 7: One could imagine that a potential strategy lies in successively computing Gröbner bases of the ideals

$$(\langle F\rangle : h^\infty) + \langle \ell^k\rangle + \langle L \setminus \{\ell\}\rangle$$

for increasing values of $k \in \mathbb{N}$ until a Gröbner basis G of the generic fiber of this ideal over $\mathbb{K}(\ell)$ can be extracted.
Then let G' be the set obtained from G by multiplying each element $g \in G$ by the denominators of its coefficients and let $h \in \mathbb{K}[\ell]$ be the least common multiple of the leading coefficients of the elements of G', regarded as a set with coefficients in $\mathbb{K}(\ell)$. Then, following Proposition 3.2.3, we have

$$\left(G' + \langle f\rangle : h^\infty\right) = (\langle F\rangle + \langle f\rangle : h^\infty) + \langle L \setminus \{\ell\}\rangle.$$

If this question can be solved a natural follow-up question would be, given that we are then essentially only computing with zero-dimensional ideals, for which the complexity of computing Gröbner bases is well controlled (see e.g. Lazard, 1983), whether an interesting complexity statement could be derived for the resulting algorithm.

Question 2. In Jeronimo and Sabia (2002) and Lecerf (2000), mild genericity assumptions are used to obtain complexity statements for incremental decomposition algorithms, see e.g. Lemma 3 in Jeronimo and Sabia (2002). These are used to control the number of components introduced by each incremental decomposition step. What would be the complexity of our algorithm based on witness sets using the same genericity assumptions?

FURTHER INCORPORATION OF ALGORITHM 18 WITH SGB COMPUTATIONS

Recall that Algorithm 18 utilizes syzygy computations of finite sequences of polynomials to decompose algebraic sets. Essentially, given a finite sequence of polynomials $F := (f_1, \ldots, f_r) \subset R$ and a polynomial $h \in R$, we

(1) Compute a polynomial $g \in R$ s.t. $gf_k \in \langle f_1, \ldots, f_{k-1}, f_{k+1}, \ldots, f_r \rangle$.

(2) Decide if $g \in (\langle F \rangle : h^\infty)$.

The first step was accomplished with sGB computations, see Algorithm 16. The second step was then accomplished by knowing a Gröbner basis of the ideal $(\langle F \rangle : h^\infty)$.
Suppose now that $h = 1$. Then, assume that $f_1, \ldots, f_r$, are homogeneous, that Algorithm 16 works with $\prec = \prec_{drl}$ and returns g, k with $\deg(g) = d$. Then Algorithm 16 has in particular computed a $d + \deg(f_k)$-truncated Gröbner basis G of $\langle F \rangle$, which has the consequence that $\text{REDUCE}(G, \prec_{drl}, g)$ returns zero if and only if $g \in \langle F \rangle$ because $\deg(g) < d + \deg(f_k)$. Hence, if $h = 1$ we can avoid precomputing a Gröbner basis of $\langle F \rangle$!
Avoiding the precomputation of a Gröbner basis of $(\langle F \rangle : h^\infty)$ when $h \neq 1$ is more difficult: It is not possible, even in the homogeneous setting, to compute a d-truncated Gröbner basis of $(\langle F \rangle : h^\infty)$, essentially because the localized ring R_h does not have a natural grading by $\mathbb{N}$. We therefore ask

Question 3. Design a signature-based algorithm which computes a Gröbner basis of $(\langle F \rangle : h^\infty)$ and satisfies the following: suppose that it has computed some intermediate Gröbner basis G and that, using this algorithm, one computes g, k as above. Ensure that Algorithm 18 terminates when we replace the membership check of g in $(\langle F \rangle : h^\infty)$ by a check whether $\text{REDUCE}(G, \prec_{drl}, g)$ is zero or not.

Essentially, we would want to replace knowing full Gröbner bases of the ideals of the affine cells involved in Algorithm 18 by knowing only partial Gröbner bases, up to some degree in the homogeneous setting, and decomposing at natural points during their computation, i.e. when we discover syzygy cofactors g as above. The algorithm designed in Chapter 4 showed that this design philosophy can signficantly cut down on computational overhead.
Another, more vague, question is then

Question 4. If one follows this design principle, how do the procedures HULL and CLOREMOVE in Algorithm 18 need to change?

More precisely, recall that for a given X and a polynomial g, $\text{HULL}(X, g)$ proceeds by first computing a Gröbner basis H for $(I_X : g^\infty)$ and

then calling CloRemove(X, H). But, following the new design philosophy, we would know H only up to some degree so we cannot call CloRemove directly. Hence, more asynchronous methods are required for decomposing a given affine cell.

SPECIALIZED VARIANTS OF FGLM FOR GENERIC FIBER COMPUTATION

In Chapter 7, we adapted the FGLM algorithm to compute Gröbner bases of generic fibers of polynomial ideals. Roughly, the task was the following: Given an ideal $I \subset \mathbb{K}[\mathbf{z}, \mathbf{x}]$ s.t. $\mathbf{z}$ is an MIS of I, and a Gröbner basis of I w.r.t. some monomial order $\prec_{in}$, compute the reduced Gröbner basis G of $\mathrm{gen}(I, \mathbf{z})$ w.r.t. another monomial order $\prec_{out}$.
To this end, we used Hensel lifting techniques and normal form computations together with linear algebra in the finite-dimensional $\mathbb{K}$-vector spaces $\mathbb{K}[\mathbf{x}, \mathbf{z}]/I + \mathfrak{m}^k$ where $\mathfrak{m} := \langle \mathbf{z} \rangle$.
For the original FGLM algorithm given in Faugère, Gianni, Lazard, and Mora (1993), which converts a given Gröbner basis of a zero-dimensional ideal to a Gröbner basis of the same ideal w.r.t. another monomial order, the required normal form and linear algebra computations are done using multiplication tensors as we did in Section 7.3. When the zero-dimensional ideal of interest is in shape position (Definition 3.2.1) and the target monomial order is the lexicographic order, a specialized variant of FGLM, called *sparse* FGLM has been developed and analyzed, see Faugère, Gaudry, Huot, and Renault (2014) and Faugère and Mou (2017) and additonal efficiency is gained under certain stability assumptions, see Neiger and Schost (2020). In the notation of Definition 3.2.1, this algorithm computes the univariate polynomial g_I as the minimal polynomial of the multiplication matrix associated to the last variable using Wiedemann's algorithm. This improves the cubic complexity of the original FGLM algorithm to the exponent of matrix multiplication. We ask

Question 5. With the particular application of irreducible decomposition as in Section 3.2.2 in mind, how can the sparse FGLM algorithm be combined with Hensel lifting techniques to compute lexicographic Gröbner bases of generic fibers in shape position? Do we preserve a complexity quasi-linear in the precision in $\mathbf{z}$ up to which we compute, as in Section 7.3?

In Berthomieu, Eder, and Safey El Din (2023) a variant of FGLM is also given to compute saturation ideals. This algorithm identifies elements in a saturation of the form $(I : f^\infty)$ for a polynomial ideal I and a polynomial f as elements in the kernel of the multiplication matrix associated to f in some staircase of I. An algorithm for irreducible decomposition as in Section 3.2.2 would also require computing generic fibers of saturation ideals. We therefore ask

Question 6. How can the FGLM variant given by Berthomieu, Eder, and Safey El Din (2023) be combined with Hensel lifting techniques to compute generic fibers of saturation ideals?

Part III

APPENDIX

A

APPENDIX

A.1 POLYNOMIAL SYSTEMS USED IN BENCHMARKS

We gather here the polynomial systems used in the benchmarks for our algorithms together with references and brief explanations.

- C1, C2 and C3 are certain jacobian ideals of single multivariate polynomials which define singular hypersurfaces.
- Cyclic(n), encoding the standard cyclic benchmarks in computer algebra in n variables.
- ED(d, n) encodes the parametric *euclidean distance* problem for a hypersurface of degree d in n variables, see Draisma et al. (2014).
- PS(n) encodes the singular points of an algebraic set cut out by polynomials
$$f_1, \ldots, f_{n-1}, g_1, \ldots, g_{n-1}$$
with $f_i \in \mathbb{K}[x_1, \ldots, x_{n-2}, z_1, z_2]$, $g_i \in \mathbb{K}[y_1, \ldots, y_{n-2}, z_1, z_2]$, the f_i being chosen as random dense quadrics, and g_i chosen such that $g_i(x_1, \ldots, x_{n-2}, z_1, z_2) = f$, i.e. as a copy of f_i in the variables $y_1, \ldots, y_{n-2}, z_1, z_2$.
- The examples R1, R2 and R3 come from Example 1.0.1.
- The RD(d) systems are randomly generated sequences of 3 polynomials of degree d in 4 variables.
- Sing(n) encodes the critical points of the restriction of the projection on the first coordinate to a (generically singular) hypersurface which is defined by the resultant in x_{n+1} of two
- SOS(s, n) encodes the critical points of the restriction of the projection on the first coordinate to a hypersurface which is a sum of s random dense quadrics in $\mathbb{K}[x_1, \ldots, x_n]$, random dense quadrics A, B in $\mathbb{K}[x_1, \ldots, x_{n+1}]$.
- The Steiner polynomial system, coming from Breiding, Sturmfels, and Timme (2020).
- All remaining examples are part of the BPAS library (Asadi et al., 2021) for triangular decomposition.

For the benchmarks of Algorithm 23, displayed in Table A.6, the following examples were used:
The Whitney Cusp is defined as $\mathbf{V}(y^2 + z^3 - x^2z^2)$. The other examples are given by

$$X_1 = \mathbf{V}(x_1^7 - 2x_1^5x_4 + x_1^3x_4^2 + x_1^2x_2^2 - x_1x_2^2x_3 + x_2^3) \subset \mathbb{C}^4$$

$$X_2 = \mathbf{V}\left(x_1^2x_3 - x_2^2,\ x_2^4 - x_1x_2^2 - x_3x_4^2,\ x_1^2x_2^2 - x_1^3 - x_4^2\right) \subset \mathbb{C}^4$$

$$X_3 = \mathbf{V}\left(x_1^6 + x_2^6 + x_1^4x_3x_4 + x_3^3\right) \subset \mathbb{C}^4$$

$$\begin{aligned} X_4 = \mathbf{V}(&-p_1y_1y_3y_4 - p_1y_2y_3y_4 - sy_1y_3y_5 - \\ &p_2y_1y_4y_5 - ty_2y_4y_5) \subset \mathbb{C}^9 \end{aligned}$$

$$X_5 = \mathbf{V}(x_1^3 - 2x_1^2x_4 + x_1^3 + x_5^2x_4 + x_2^2x_1 - x_2x_1x_3 + x_5^3) \subset \mathbb{C}^5$$

$$X_6 = \mathbf{V}\left(x_2x_5^2 - x_1^2,\ x_2x_4x_5 - x_3^2 - x_2,\ x_1^2x_4 - x_3^2x_5 - x_2x_5\right) \subset \mathbb{C}^5$$

$$\begin{aligned} X_7 = \mathbf{V}(&-p_1x_1x_3x_4 - p_1x_2x_3x_4 - sx_1x_3x_5 - p_2x_1x_4x_5 - tx_2x_4x_5 - \\ &p_1x_3x_4x_5) \subset \mathbb{C}^9 \end{aligned}$$

$$\begin{aligned} X_8 = \mathbf{V}(&x_1x_2^3 - 2x_1x_2^2x_4 + x_1x_2x_3x_5 + x_1x_2x_4^2 + x_1x_2x_5 - x_1x_3x_4x_5 - \\ &x_1x_4x_5 - x_2^2x_3x_5 - x_2^2x_4 - x_2^2x_5 + x_2x_3x_4x_5 - x_2x_3x_5 + \\ &2x_2x_4^2 + x_2x_4x_5 - x_2x_5 - x_3^2x_5^2 + x_3x_4x_5 - 2x_3x_5^2 - \\ &x_4^3 + x_4x_5 - x_5^2) \subset \mathbb{C}^5 \end{aligned}$$

$$\begin{aligned} X_9 = \mathbf{V}(&-x_1^2x_3x_5 - x_1^2x_4x_5 + x_1^2x_5 + x_1x_2x_3x_4 + x_1x_2x_3x_5 + \\ &x_1x_2x_4x_5 - x_1x_2x_4 - x_1x_2x_5 - x_1x_3^2x_4 + x_1x_3^2x_5 - x_1x_3x_4^2 + \\ &x_1x_3x_4x_5 + 2x_1x_3x_4 - x_1x_3x_5^2 - 2x_1x_3x_5 + x_1x_4^2x_5 + x_1x_4^2 - \\ &x_1x_4x_5^2 - x_1x_4x_5 - x_1x_4 + x_1x_5^2 + x_1x_5 - x_2^2x_3x_4 + x_2^2x_4 + \\ &x_2x_3^2x_4 - x_2x_3^2x_5 + x_2x_3x_4^2 - 2x_2x_3x_4 + 2x_2x_3x_5 - x_2x_4^2x_5 - \\ &x_2x_4^2 + x_2x_4 - x_2x_5 - x_3^2x_4^2 + x_3^2x_4x_5 + x_3x_4^2x_5 + \\ &2x_3x_4^2 - x_3x_4x_5^2 - 2x_3x_4x_5 - x_4^2x_5 - x_4^2 + x_4x_5^2 + \\ &x_4x_5) \subset \mathbb{C}^5 \end{aligned}$$

In the tables, the different algorithms to which we compare ourselves are indicated by symbols as follows:

- †: Homological method (see Eisenbud, Huneke, and Vasconcelos, 1992).
- ‡: Elimination method (see Gianni, Trager, and Zacharias, 1988).
- $: Regular chains (see Lemaire, Maza, and Xie, 2005).
- £: Irreducible decomposition.

In the tables an ∞ sign means that the corresponding algorithm took more than 50 times the time of the fastest algorithm recorded in the same row or that an error occured during the computation, for example due to a memory limit we set. In Table A.6, in the "Minimization" column it means that the computation took more than 8 hours. All timings are given in seconds unless otherwise indicated.

Table A.1: Comparing Algorithm 11 and Algorithm 13

	Alg. 10 arith. op.	Alg. 13 arith. op.	Alg. 10	Alg. 13	Ratio	`Maple`: GB	`Maple`: Alg. 11	Ratio
Cyclic(8)	$1.2 \cdot 10^{10}$	$1.3 \cdot 10^{11}$	4m	40m	10	1.2	154m	7700
PS(12)	$5.3 \cdot 10^{7}$	$3.1 \cdot 10^{8}$	1.16	5.2	4.5	0.268	3.44	13
Sing(10)	$5.6 \cdot 10^{7}$	$6.5 \cdot 10^{7}$	1.9	2.9	1.5	0.11	1.642	14.5
Sing(9)	$2.5 \cdot 10^{7}$	$2.9 \cdot 10^{7}$	1.1	1.4	1.27	0.06	0.788	13.1
SOS(5,4)	$1.3 \cdot 10^{8}$	$1.1 \cdot 10^{8}$	8.5	7.3	0.85	0.022	0.479	21.3
SOS(6,3)	$2.1 \cdot 10^{7}$	$2.1 \cdot 10^{7}$	1.11	1.4	1.26	0.021	0.261	12.4
SOS(6,4)	$4.8 \cdot 10^{9}$	$3.8 \cdot 10^{9}$	148	169	1.14	0.172	22.7	132
SOS(6,5)	$4.2 \cdot 10^{9}$	$2.0 \cdot 10^{9}$	75	43	0.57	0.458	10.38	22.7
SOS(7,3)	$1.3 \cdot 10^{8}$	$6.7 \cdot 10^{8}$	5.2	41	7.9	0.047	7.162	152.4
SOS(7,4)	$6.5 \cdot 10^{9}$	$4.5 \cdot 10^{10}$	3m	32m	10.7	0.433	1h	8314
SOS(7,5)	$7.2 \cdot 10^{10}$	$3.5 \cdot 10^{11}$	25m	20h	48	2.294	∞	∞
SOS(7,6)	$1.7 \cdot 10^{12}$	$3.0 \cdot 10^{12}$	31h	73h	2.4	14.348	5.5h	23
Steiner	$3.1 \cdot 10^{10}$	$2.3 \cdot 10^{11}$	4.2m	42m	10	27	13m	28.9

Table A.2: Comparing Algorithm 13 with other Decomposition Methods

	Alg. 13	`Singular`: †	`Singular`: ‡	`Maple`: $	`Macaulay2`: ‡
Cyclic(8)	40m	segfault	>35h	∞	∞
PS(10)	0.3	40	∞	∞	∞
PS(12)	5.2	∞	∞	∞	∞
PS(6)	0.008	<1	<1	0.29	0.07
PS(8)	0.03	<1	23m	5.82	13.78
Sing(10)	2.9	∞	∞	∞	∞
Sing(4)	0.02	1	∞	91.32	0.42
Sing(5)	0.07	4	∞	∞	1.94
Sing(6)	0.15	56	∞	∞	16.64
Sing(7)	0.35	8m	∞	∞	289
Sing(8)	0.68	23m	∞	∞	∞
Sing(9)	1.4	∞	∞	∞	∞
SOS(4,2)	0.03	<1	<1	19.4	0.16
SOS(4,3)	0.03	1	3m	14m	0.63
SOS(5,2)	0.02	<1	∞	∞	0.37
SOS(5,3)	0.34	∞	∞	∞	9.35
SOS(5,4)	7.3	∞	∞	∞	183
SOS(6,2)	0.17	<1	∞	∞	0.7
SOS(6,3)	1.4	∞	∞	∞	107
SOS(6,4)	169	∞	∞	∞	∞
SOS(6,5)	43	∞	∞	∞	∞
SOS(7,2)	2.91	<1	∞	2.94	0.18
SOS(7,3)	41	∞	∞	∞	∞
SOS(7,4)	32m	∞	segfault	∞	∞
SOS(7,5)	20h	segfault	segfault	∞	∞
SOS(7,6)	73h	segfault	segfault	∞	∞
Steiner	42m	∞	segfault	∞	∞

Table A.3: Comparing Algorithm 15 with other Decomposition Methods

	nb. comps.	Algorithm 15	`Maple`: $	`Oscar`: ‡	`Magma`: †	`Magma`: £
8-3-config-Li	23	1.61	16.07	∞	∞	64.62
Cyclic(8)	6	381.20	∞	∞	∞	∞
dgp6	3	+0.25	53.26	2.19	∞	1.21
Gonnet	3	0.19	2.13	2.78	∞	1.35
KdV		∞	352.61	∞	∞	7108.56
Leykin-1	13	2.63	4.38	640.52	∞	1.37
C1	4	128.84	∞	∞	∞	∞
C2	4	0.27	100.29	151.88	∞	1.96
C3	13	9.67	54.51	6.83	0.35	1.49
MontesS16	6	1.92	2.66	2.02	1.40	1.49
PS(10)	2	1.70	∞	29.72	∞	5.66
PS(12)	2	51.21	∞	∞	∞	2060.03
PS(6)	2	0.03	0.25	1.67	0.50	0.34
PS(8)	2	0.09	4.46	1.72	1.19	0.81
Sing(10)	2	0.36	∞	∞	∞	∞
Sing(8)	2	0.13	∞	995.12	∞	∞
Sing(9)	2	0.22	∞	∞	∞	∞
SOS(6,3)	2	0.10	∞	∞	∞	∞
SOS(6,4)	2	4.83	∞	∞	∞	∞
SOS(6,5)	2	13.72	∞	∞	∞	∞
Steiner	2	869.79	∞	∞	∞	∞
sys2161	33	7.54	28.76	∞	∞	7.57
sys2874	5	0.26	201.84	1.87	8.42	9.72
sys2880	50	4.27	144.30	1.80	3.44	4.00
sys2882		∞	38.78	∞	∞	∞

Table A.4: Comparing Algorithm 18 with Algorithm 15

	Algorithm 18	Algorithm 15
Cyclic(8)	∞	381.2
C1	15.0	129.0
C3	0.8	10.0
PS(10)	0.2	1.7
PS(12)	6.3	51.0
PS(14)	248.2	3128.3
PS(16)	13666.2	∞
Sing(10)	3.8	1.0
SOS(6,4)	6.0	5.0
SOS(6,5)	24.9	14.0
SOS(7,4)	14.2	24.4
SOS(7,5)	112.2	∞
Steiner	404.9	870.0
sys2353	10.9	1.6
sys2161	26.2	7.54
ED(3,4)	30.7	294.1
ED(3,5)	828.1	∞

Table A.5: Benchmarks for Algorithm 20

	Algorithm 20	`msolve` with $\prec_{\text{out}}$
ED(3,3)	237.9	43521.43
R1	0.01	0.01
R2	0.01	0.01
R3	0.01	0.01
C2	2.75	0.03
C3	0.19	0.01
PS(2,10)	0.8	0.3
PS(2,12)	44.82	7.3
Sing(2,10)	0.2	0.1
SOS(5,4)	1.2	0.3
SOS(6,4)	11.97	30.35
SOS(6,5)	22.19	26.61
RD(3)	4.29	0.11
RD(4)	33.42	13.43
RD(5)	729.51	780.92

Table A.6: Benchmarks for Algorithm 23

	Algorithm in Helmer and Nanda (2023)	Minimal	Algorithm 23	Minimal	Minimization
Whitney Cusp	0.12	Yes	0.09	Yes	0.3
X_1	∞	-	7.3	No	140.5
X_2	85.3	Yes	82.8	No	1.1
X_3	4.8	No	4.6	No	68.1
X_4	182.3	-	192.7	-	∞
X_5	2.3	Yes	1.5	Yes	0.5
X_6	1.3	Yes	0.9	Yes	2.4
X_7	∞	-	6080.8	-	∞
X_8	∞	-	12	-	∞
X_9	∞	-	1200	-	∞

BIBLIOGRAPHY

Arnold, E. A. (Apr. 2003). "Modular Algorithms for Computing Gröbner Bases." In: *Journal of Symbolic Computation* 35.4, pp. 403–419. ISSN: 0747-7171. DOI: 10.1016/S0747-7171(02)00140-2.

Asadi, M. et al. (2021). *Basic Polynomial Algebra Subprograms (BPAS) (Version 1.791)*.

Atiyah, M. F. and I. G. MacDonald (Feb. 1994). *Introduction To Commutative Algebra*. Avalon Publishing. ISBN: 978-0-8133-4544-4.

Aubry, P., D. Lazard, and M. M. Maza (July 1999). "On the Theories of Triangular Sets." In: *Journal of Symbolic Computation* 28.1, pp. 105–124. ISSN: 0747-7171. DOI: 10.1006/jsco.1999.0269.

Aubry, P. and M. M. Maza (July 1999). "Triangular Sets for Solving Polynomial Systems: A Comparative Implementation of Four Methods." In: *Journal of Symbolic Computation* 28.1, pp. 125–154. ISSN: 0747-7171. DOI: 10.1006/jsco.1999.0270.

Bardet, M., J.-C. Faugère, and B. Salvy (Sept. 2015). "On the Complexity of the F5 Gröbner Basis Algorithm." In: *Journal of Symbolic Computation* 70, pp. 49–70. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2014.09.025.

Bates, D. J., J. D. Hauenstein, A. J. Sommese, and C. W. Wampler (Nov. 2013). *Numerically Solving Polynomial Systems with Bertini*. SIAM. ISBN: 978-1-61197-269-6.

Becker, T. and V. Weispfenning (1993). *Gröbner Bases*. Vol. 141. Graduate Texts in Mathematics. Springer-Verlag, New York. DOI: 10.1007/978-1-4612-0913-3.

Berthomieu, J., C. Eder, and M. S. Safey El Din (July 2023). *New Efficient Algorithms for Computing Gröbner Bases of Saturation Ideals (F4SAT) and Colon Ideals (Sparse-FGLM-colon)*. DOI: 10.48550/arXiv.2202.13387. arXiv: 2202.13387 [cs, math].

Berthomieu, J., C. Eder, and M. Safey El Din (2021). "Msolve: A Library for Solving Polynomial Systems." In: *ISSAC'21*. Saint Petersburg, Russia. DOI: 10.1145/3452143.3465545.

Berthomieu, J. and R. Mohr (July 2024). "Computing Generic Fibers of Polynomial Ideals with FGLM and Hensel Lifting." In: *Proceedings of the 2024 International Symposium on Symbolic and Algebraic Computation*. ISSAC '24. New York, NY, USA: Association for Com-

puting Machinery, pp. 307–315. ISBN: 9798400706967. DOI: 10.1145/3666000.3669703.

Berthomieu, J., V. Neiger, and M. Safey El Din (July 2022). "Faster Change of Order Algorithm for Gröbner Bases under Shape and Stability Assumptions." In: *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation*. ISSAC '22. New York, NY, USA: Association for Computing Machinery, pp. 409–418. ISBN: 978-1-4503-8688-3. DOI: 10.1145/3476446.3535484.

Bezanson, J., A. Edelman, S. Karpinski, and V. B. Shah (2017). "Julia: A Fresh Approach to Numerical Computing." In: *SIAM review* 59.1, pp. 65–98.

Bitmead, R. R. and B. D. O. Anderson (Dec. 1980). "Asymptotically Fast Solution of Toeplitz and Related Systems of Linear Equations." In: *Linear Algebra and its Applications* 34, pp. 103–116. ISSN: 0024-3795. DOI: 10.1016/0024-3795(80)90161-5.

Bostan, A., C.-P. Jeannerod, C. Mouilleron, and É. Schost (Jan. 2017). "On Matrices With Displacement Structure: Generalized Operators and Faster Algorithms." In: *SIAM Journal on Matrix Analysis and Applications* 38.3, pp. 733–775. ISSN: 0895-4798. DOI: 10.1137/16M1062855.

Breiding, P., B. Sturmfels, and S. Timme (2020). "3264 Conics in a Second." In: *Notices of the American Mathematical Society* 67.1, pp. 30–37. ISSN: 0002-9920.

Breiding, P. and S. Timme (May 2018). *HomotopyContinuation.Jl: A Package for Homotopy Continuation in Julia*. DOI: 10.48550/arXiv.1711.10911. arXiv: 1711.10911 [cs, math].

Buchberger, B. (1965). "Ein Algorithmus Zum Auffinden Der Basiselemente Des Restklassenringes Nach Einem Nulldimensionalen Polynomideal." PhD thesis. Universität Innsbruck.

Caboara, M., P. Conti, and C. Traverso (1997). "Yet Another Ideal Decomposition Algorithm." In: *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*. Lecture Notes in Computer Science. Springer, pp. 39–54. DOI: 10/d6b9gb.

Chen, C. and M. Moreno Maza (2012). "Algorithms for Computing Triangular Decomposition of Polynomial Systems." In: *Journal of Symbolic Computation* 47.6, pp. 610–642. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2011.12.023.

Chen, C. et al. (2007). "Efficient Computations of Irredundant Triangular Decompositions with the Regularchains Library." In: *International Conference on Computational Science*. Springer, pp. 268–271.

Chou, S.-C. and X.-S. Gao (1990). "Ritt-Wu's Decomposition Algorithm and Geometry Theorem Proving." In: *10th International Conference on Automated Deduction*. Lecture Notes in Computer Science. Springer, pp. 207–220. ISBN: 978-3-540-47171-4. DOI: 10/bftnht.

Collart, S., M. Kalkbrener, and D. Mall (Sept. 1997). "Converting Bases with the Gröbner Walk." In: *Journal of Symbolic Computation* 24.3, pp. 465–469. ISSN: 0747-7171. DOI: 10.1006/jsco.1996.0145.

Colotti, A. et al. (2024). "Determination of All Stable and Unstable Equilibria for Image-Point-Based Visual Servoing." In: *IEEE Transactions on Robotics*, p. 1.

Cox, D. A., J. Little, and D. O'Shea (2015). *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Cham: Springer International Publishing. ISBN: 978-3-319-16720-6 978-3-319-16721-3. DOI: 10.1007/978-3-319-16721-3.

Decker, W., G.-M. Greuel, and G. Pfister (1999). "Primary Decomposition: Algorithms and Comparisons." In: *Algorithmic Algebra and Number Theory*. Springer, pp. 187–220. ISBN: 978-3-642-59932-3. DOI: 10/b6bvwp.

Decker, W., G.-M. Greuel, G. Pfister, and H. Schönemann (2022). *Singular 4-3-0 — A Computer Algebra System for Polynomial Computations*.

Decker, W. et al., eds. (Jan. 2025). *The Computer Algebra System OSCAR: Algorithms and Examples*. 1st ed. Vol. 32. Algorithms and Computation in Mathematics. Springer.

Della Dora, J., C. Dicrescenzo, and D. Duval (Apr. 1985). "About a New Method for Computing in Algebraic Number Fields." In: *Research Contributions from the European Conference on Computer Algebra-Volume 2*. EUROCAL '85. Springer-Verlag, pp. 289–290. ISBN: 978-3-540-15984-1.

Demin, A. and S. Gowda (Sept. 2023). *Groebner.Jl: A Package for Gröbner Bases Computations in Julia*. DOI: 10.48550/arXiv.2304.06935. arXiv: 2304.06935 [cs, math].

Dixon, J. D. (Feb. 1982). "Exact Solution of Linear Equations Using P-adic Expansions." In: *Numerische Mathematik* 40.1, pp. 137–141. ISSN: 0945-3245. DOI: 10.1007/BF01459082.

Draisma, J. et al. (Nov. 2014). *The Euclidean Distance Degree of an Algebraic Variety*. DOI: 10.48550/arXiv.1309.0049. arXiv: 1309.0049 [math]. (Visited on 09/14/2023).

Duff, T., A. Leykin, and J. I. Rodriguez (June 2022). *u-Generation: Solving Systems of Polynomials Equation-by-Equation*. DOI: 10.48550/arXiv.2206.02869. arXiv: 2206.02869 [cs, math].

Ebert, G. L. (May 1983). "Some Comments on the Modular Approach to Gröbner-bases." In: *ACM SIGSAM Bulletin* 17.2, pp. 28–32. ISSN: 0163-5824. DOI: 10.1145/1089330.1089336.

Eder, C. and J.-C. Faugère (May 2017). "A Survey on Signature-Based Algorithms for Computing Gröbner Bases." In: *Journal of Symbolic Computation* 80, pp. 719–784. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2016.07.031.

Eder, C., P. Lairez, R. Mohr, and M. Safey El Din (July 2023a). "A Direttissimo Algorithm for Equidimensional Decomposition." In:

Proceedings of the 2023 International Symposium on Symbolic and Algebraic Computation. ISSAC '23. New York, NY, USA: Association for Computing Machinery, pp. 260–269. ISBN: 9798400700392. DOI: 10.1145/3597066.3597069.

Eder, C., P. Lairez, R. Mohr, and M. Safey El Din (Nov. 2023b). "A Signature-Based Algorithm for Computing the Nondegenerate Locus of a Polynomial System." In: *Journal of Symbolic Computation* 119, pp. 1–21. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2023.02.001.

Eder, C. and B. H. Roune (June 2013). "Signature Rewriting in Gröbner Basis Computation." In: *Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation*. ISSAC '13. New York, NY, USA: Association for Computing Machinery, pp. 331–338. ISBN: 978-1-4503-2059-7. DOI: 10.1145/2465506.2465522.

Eisenbud, D. (1995). *Commutative Algebra: With a View toward Algebraic Geometry*. New York, NY: Springer New York. ISBN: 978-1-4612-5350-1. DOI: 10.1007/978-1-4612-5350-1.

Eisenbud, D., C. Huneke, and W. Vasconcelos (Dec. 1992). "Direct Methods for Primary Decomposition." In: *Inventiones Mathematicae* 110.1, pp. 207–235. ISSN: 1432-1297. DOI: 10/cjc7vg.

Faugère, J. C., P. Gianni, D. Lazard, and T. Mora (Oct. 1993). "Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering." In: *Journal of Symbolic Computation* 16.4, pp. 329–344. ISSN: 0747-7171. DOI: 10.1006/jsco.1993.1051.

Faugère, J.-C. (1999). "A New Efficient Algorithm for Computing Gröbner Bases (F4)." In: *Journal of Pure and Applied Algebra* 139.1, pp. 61–88. ISSN: 0022-4049. DOI: 10.1016/S0022-4049(99)00005-5.

Faugère, J. C. (July 2002). "A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5)." In: *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*. ISSAC '02. New York, NY, USA: Association for Computing Machinery, pp. 75–83. ISBN: 978-1-58113-484-1. DOI: 10.1145/780506.780516.

Faugère, J.-C., P. Gaudry, L. Huot, and G. Renault (July 2014). "Sub-Cubic Change of Ordering for Gröbner Basis: A Probabilistic Approach." In: *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*. ISSAC '14. New York, NY, USA: Association for Computing Machinery, pp. 170–177. ISBN: 978-1-4503-2501-1. DOI: 10.1145/2608628.2608669.

Faugère, J.-C. and C. Mou (May 2017). "Sparse FGLM Algorithms." In: *Journal of Symbolic Computation* 80, pp. 538–569. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2016.07.025.

Feinberg, M. (2019). *Foundations of Chemical Reaction Network Theory*. Vol. 202. Applied Mathematical Sciences. Cham: Springer International Publishing. ISBN: 978-3-030-03857-1 978-3-030-03858-8. DOI: 10.1007/978-3-030-03858-8.

Flores, A. G. and B. Teissier (Dec. 2017). *Local Polar Varieties in the Geometric Study of Singularities*. DOI: 10.48550/arXiv.1607.07979. arXiv: 1607.07979 [math].

Gallo, G. and B. Mishra (1991). "Efficient Algorithms and Bounds for Wu-Ritt Characteristic Sets." In: *Effective Methods in Algebraic Geometry*. Ed. by T. Mora and C. Traverso. Boston, MA: Birkhäuser Boston, pp. 119–142. ISBN: 978-1-4612-0441-1. DOI: 10.1007/978-1-4612-0441-1_8.

García Fontán, J., A. Nayak, S. Briot, and M. Safey El Din (Apr. 2022). "Singularity Analysis for the Perspective-Four and Five-Line Problems." In: *International Journal of Computer Vision* 130.4, pp. 909–932. ISSN: 1573-1405. DOI: 10.1007/s11263-021-01567-4.

Gianni, P., B. Trager, and G. Zacharias (Oct. 1988). "Gröbner Bases and Primary Decomposition of Polynomial Ideals." In: *Journal of Symbolic Computation* 6.2, pp. 149–167. ISSN: 0747-7171. DOI: 10.1016/S0747-7171(88)80040-3.

Giusti, M., G. Lecerf, and B. Salvy (Mar. 2001). "A Gröbner Free Alternative for Polynomial System Solving." In: *Journal of Complexity* 17.1, pp. 154–211. ISSN: 0885-064X. DOI: 10.1006/jcom.2000.0571.

Grayson, D. R. and M. E. Stillman (n.d.). *Macaulay2, a Software System for Research in Algebraic Geometry*. URL: https://www.macaulay2.com.

Greuel, G.-M. and G. Pfister (2007). *A Singular Introduction to Commutative Algebra*. 2nd ed. Springer Berlin Heidelberg. ISBN: 978-3-540-73541-0. DOI: 10.1007/978-3-540-73542-7.

Guillaume, P. and A. Huard (Sept. 2000). "Multivariate Padé Approximation." In: *Journal of Computational and Applied Mathematics* 121.1, pp. 197–219. ISSN: 0377-0427. DOI: 10.1016/S0377-0427(00)00337-X.

Haramoto, H. and M. Matsumoto (Mar. 2009). "A *P*-Adic Algorithm for Computing the Inverse of Integer Matrices." In: *Journal of Computational and Applied Mathematics* 225.1, pp. 320–322. ISSN: 0377-0427. DOI: 10.1016/j.cam.2008.07.044.

Harris, C. and M. Helmer (May 2019). "Segre Class Computation and Practical Applications." In: *Mathematics of Computation* 89.321, pp. 465–491. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/mcom/3448. arXiv: 1806.07408 [cs, math].

Hashemi, A. and W. M. Seiler (July 2017). "Dimension-Dependent Upper Bounds for Gröbner Bases." In: *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*. ISSAC '17. New York, NY, USA: Association for Computing Machinery, pp. 189–196. ISBN: 978-1-4503-5064-8. DOI: 10.1145/3087604.3087624.

Helmer, M. and R. Mohr (June 2024). *A New Algorithm for Whitney Stratification of Varieties*. DOI: 10.48550/arXiv.2406.17122. arXiv: 2406.17122 [cs, math].

Helmer, M. and V. Nanda (Oct. 2023). "Conormal Spaces and Whitney Stratifications." In: *Foundations of Computational Mathematics* 23.5, pp. 1745–1780. ISSN: 1615-3383. DOI: 10.1007/s10208-022-09574-8.

Helmer, M., G. Papathanasiou, and F. Tellander (Feb. 2024). *Landau Singularities from Whitney Stratifications*. DOI: 10.48550/arXiv.2402.14787. arXiv: 2402.14787 [hep-th, physics:math-ph].

Hills, C., A. Baskar, M. Plecnik, and J. D. Hauenstein (June 2024). "Computing Complete Solution Sets for Approximate Four-Bar Path Synthesis." In: *Mechanism and Machine Theory* 196, p. 105628. ISSN: 0094-114X. DOI: 10.1016/j.mechmachtheory.2024.105628.

Hironaka, H. (1964). "Resolution of Singularities of an Algebraic Variety Over a Field of Characteristic Zero: I." In: *Annals of Mathematics* 79.1, pp. 109–203. ISSN: 0003-486X. DOI: 10.2307/1970486. JSTOR: 1970486.

Hubert, E. (2003). "Notes on Triangular Sets and Triangulation-Decomposition Algorithms I." In: *Symbolic and Numerical Scientific Computation*. Lecture Notes in Computer Science. Springer, pp. 1–39. ISBN: 978-3-540-45084-9. DOI: 10/fqqz59.

Ishihara, Y. (July 2022). "Modular Techniques for Intermediate Primary Decomposition." In: *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation*. ISSAC '22. New York, NY, USA: Association for Computing Machinery, pp. 479–487. ISBN: 978-1-4503-8688-3. DOI: 10.1145/3476446.3535488.

Jeronimo, G. and J. Sabia (Apr. 2002). "Effective Equidimensional Decomposition of Affine Varieties." In: *Journal of Pure and Applied Algebra* 169.2, pp. 229–248. ISSN: 0022-4049. DOI: 10/bmd645.

Johansson, F. (Jan. 2015). "A Fast Algorithm for Reversion of Power Series." In: *Mathematics of Computation* 84.291, pp. 475–484. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-2014-02857-3.

Kalkbrener, M. (1993). "A Generalized Euclidean Algorithm for Computing Triangular Representations of Algebraic Varieties." In: *Journal of Symbolic Computation* 15.2, pp. 143–167.

Kipnis, A., J. Patarin, and L. Goubin (1999). "Unbalanced Oil and Vinegar Signature Schemes." In: *Advances in Cryptology — EUROCRYPT '99*. Ed. by J. Stern. Berlin, Heidelberg: Springer, pp. 206–222. ISBN: 978-3-540-48910-8. DOI: 10.1007/3-540-48910-X_15.

Krick, T. and A. Logar (1991). "An Algorithm for the Computation of the Radical of an Ideal in the Ring of Polynomials." In: *AAECC 1991*. Springer-Verlag, pp. 195–205. ISBN: 978-3-540-54522-4.

Lairez, P. (July 2024). "Axioms for a Theory of Signature Bases." In: *Journal of Symbolic Computation* 123, p. 102275. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2023.102275.

Lazard, D. (1983). "Gröbner Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations." In: *Computer Algebra*. Ed. by J. A. van Hulzen. Lecture Notes in Computer Science. Berlin,

Heidelberg: Springer, pp. 146–156. ISBN: 978-3-540-38756-5. DOI: 10.1007/3-540-12868-9_99.

Lazard, D. (Nov. 1991). "A New Method for Solving Algebraic Systems of Positive Dimension." In: *Discrete Applied Mathematics* 33.1, pp. 147–160. ISSN: 0166-218X. DOI: 10.1016/0166-218X(91)90113-B.

Lecerf, G. (2000). "Computing an Equidimensional Decomposition of an Algebraic Variety by Means of Geometric Resolutions." In: *ISSAC'00*, pp. 209–216.

Lecerf, G. (Aug. 2003). "Computing the Equidimensional Decomposition of an Algebraic Closed Set by Means of Lifting Fibers." In: *Journal of Complexity* 19.4, pp. 564–596. ISSN: 0885-064X. DOI: 10/dv98cb.

Lemaire, F., M. M. Maza, and Y. Xie (Sept. 2005). "The RegularChains Library in MAPLE." In: *ACM SIGSAM Bulletin* 39.3, pp. 96–97. ISSN: 0163-5824. DOI: 10.1145/1113439.1113456. (Visited on 01/12/2023).

Lu, Y. and Z. Jingzhong (Aug. 1994). "Searching Dependency between Algebraic Equations: An Algorithm Applied to Automated Reasoning." In.

Mather, J. (Oct. 2012). "Notes on Topological Stability." In: *Bulletin of the American Mathematical Society* 49.4, pp. 475–506. ISSN: 0273-0979, 1088-9485. DOI: 10.1090/S0273-0979-2012-01383-6.

Matsumura, H. (1987). *Commutative Ring Theory*. Trans. by M. Reid. Cambridge Studies in Advanced Mathematics. Cambridge: Cambridge University Press. ISBN: 978-0-521-36764-6. DOI: 10.1017/CBO9781139171762.

Monagan, M. and R. Pearce (2015). "A Compact Parallel Implementation of F4." In: *Proceedings of the 2015 International Workshop on Parallel Symbolic Computation*, pp. 95–100.

Morf, M. (Apr. 1980). "Doubling Algorithms for Toeplitz and Related Equations." In: *ICASSP '80. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 5, pp. 954–959. DOI: 10.1109/ICASSP.1980.1171074.

Moroz, G. (2008). "Regular Decompositions." In: *Computer Mathematics*. Ed. by D. Kapur. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 263–277. ISBN: 978-3-540-87827-8. DOI: 10.1007/978-3-540-87827-8_22.

Neiger, V. and É. Schost (Oct. 2020). "Computing Syzygies in Finite Dimension Using Fast Linear Algebra." In: *Journal of Complexity* 60, p. 101502. ISSN: 0885-064X. DOI: 10.1016/j.jco.2020.101502.

Noro, M. and K. Yokoyama (Oct. 2004). "Implementation of Prime Decomposition of Polynomial Ideals over Small Finite Fields." In: *Journal of Symbolic Computation*. Symbolic Computation in Algebra and Geometry 38.4, pp. 1227–1246. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2003.08.004.

Pan, V. Y. (Aug. 2011). "Nearly Optimal Solution of Rational Linear Systems of Equations with Symbolic Lifting and Numerical Initial-

ization." In: *Computers & Mathematics with Applications* 62.4, pp. 1685–1706. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2011.06.006.

Pauer, F. (Nov. 1992). "On Lucky Ideals for Gröbner Basis Computations." In: *Journal of Symbolic Computation* 14.5, pp. 471–482. ISSN: 0747-7171. DOI: 10.1016/0747-7171(92)90018-Y.

Ritt, J. F. (Dec. 1950). *Differential Algebra*. American Mathematical Soc. ISBN: 978-0-8218-4638-4.

Sayrafi, M. (Oct. 2017). *Computations over Local Rings in Macaulay2*. DOI: 10.48550/arXiv.1710.09830. arXiv: 1710.09830 [math].

Schost, É. (Feb. 2003). "Computing Parametric Geometric Resolutions." In: *Applicable Algebra in Engineering, Communication and Computing* 13.5, pp. 349–393. ISSN: 1432-0622. DOI: 10.1007/s00200-002-0109-x.

Schost, É. and C. St-Pierre (July 2023). "P-Adic Algorithm for Bivariate Gröbner Bases." In: *Proceedings of the 2023 International Symposium on Symbolic and Algebraic Computation*. ISSAC '23. New York, NY, USA: Association for Computing Machinery, pp. 508–516. ISBN: 9798400700392. DOI: 10.1145/3597066.3597086.

Schwartz, J. T. (Oct. 1980). "Fast Probabilistic Algorithms for Verification of Polynomial Identities." In: *Journal of the ACM* 27.4, pp. 701–717. ISSN: 0004-5411. DOI: 10.1145/322217.322225.

Sommese, A. J., J. Verschelde, and C. W. Wampler (2005). "Introduction to Numerical Algebraic Geometry." In: *Solving Polynomial Equations: Foundations, Algorithms, and Applications*. Ed. by M. Bronstein et al. Algorithms and Computation in Mathematics. Berlin, Heidelberg: Springer, pp. 301–337. ISBN: 978-3-540-27357-8. DOI: 10.1007/3-540-27357-3_8.

Sun, Y. and D. Wang (Aug. 2011). *Solving Detachability Problem for the Polynomial Ring by Signature-based Groebner Basis Algorithms*. DOI: 10.48550/arXiv.1108.1301. arXiv: 1108.1301 [cs].

Teissier, B. (1982). "Varietes Polaires II Multiplicites Polaires, Sections Planes, et Conditions de Whitney." In: *Algebraic Geometry*. Ed. by J. M. Aroca, R. Buchweitz, M. Giusti, and M. Merle. Berlin, Heidelberg: Springer, pp. 314–491. ISBN: 978-3-540-39367-2. DOI: 10.1007/BFb0071291.

Traverso, C. (1989). "Gröbner Trace Algorithms." In: *Symbolic and Algebraic Computation*. Ed. by P. Gianni. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 125–138. ISBN: 978-3-540-46153-1. DOI: 10.1007/3-540-51084-2_12.

Vasconcelos, W. V. (Aug. 1967). "Ideals Generated by R-sequences." In: *Journal of Algebra* 6.3, pp. 309–316. ISSN: 0021-8693. DOI: 10.1016/0021-8693(67)90086-5.

Vasconcelos, W. (1998). *Computational Methods in Commutative Algebra and Algebraic Geometry*. Algorithms and Computation in Mathematics. Springer-Verlag. ISBN: 978-3-540-21311-6.

Verschelde, J. (Apr. 2014). *Modernizing PHCpack through Phcpy*. DOI: 10.48550/arXiv.1310.0056. arXiv: 1310.0056 [cs, math].

Wang, D. (Aug. 1993). "An Elimination Method for Polynomial Systems." In: *Journal of Symbolic Computation* 16.2, pp. 83–114. ISSN: 0747-7171. DOI: 10/cpnt72.

Wang, D. (2001). *Elimination Methods*. Texts and Monographs in Symbolic Computation. Springer Vienna. DOI: 10.1007/978-3-7091-6202-6.

Winkler, F. (Oct. 1988). "A P-Adic Approach to the Computation of Gröbner Bases." In: *Journal of Symbolic Computation* 6.2, pp. 287–304. ISSN: 0747-7171. DOI: 10.1016/S0747-7171(88)80049-X.

Wu, W.-T. (Sept. 1986). "Basic Principles of Mechanical Theorem Proving in Elementary Geometries." In: 2.3, pp. 221–252. ISSN: 1573-0670. DOI: 10/cds7ng.

Yokoyama, K. (July 2002). "Prime Decomposition of Polynomial Ideals over Finite Fields." In: *Mathematical Software*. WORLD SCIENTIFIC, pp. 217–227. ISBN: 978-981-238-048-7. DOI: 10.1142/9789812777171_0022.