REGENERATION, LOCAL DIMENSION, AND APPLICATIONS IN NUMERICAL ALGEBRAIC GEOMETRY

A Dissertation

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Jonathan David Hauenstein

Andrew J. Sommese, Director

Graduate Program in Mathematics
Notre Dame, Indiana
April 2009

REGENERATION, LOCAL DIMENSION, AND APPLICATIONS IN NUMERICAL ALGEBRAIC GEOMETRY

Abstract

by

Jonathan David Hauenstein

Algorithms in the field of numerical algebraic geometry provide numerical methods for computing and manipulating solution sets of polynomial systems. One of the main algorithms in this field is the computation of the numerical irreducible decomposition. This algorithm has three main parts: computing a witness superset, filtering out the junk points to create a witness set, and decomposing the witness set into irreducible components. New and efficient algorithms are presented in this thesis to address the first two parts, namely regeneration and a local dimension test. Regeneration is an equation-by-equation solving method that can be used to efficiently compute a witness superset for a polynomial system. The local dimension test algorithm presented in this thesis is a numerical-symbolic method that can be used to compute the local dimension at an approximated solution to a polynomial system. This test is used to create an efficient algorithm that filters out the junk points. The algorithms presented in this thesis are applied to problems arising in kinematics and partial differential equations.

To my wife, Julie.

CONTENTS

FIGURES

TABLES

ACKNOWLEDGMENTS

This dissertation would not be possible without the guidance and support of many people who have helped throughout my educational process, and I am forever grateful for this.

This dissertation is built upon my mathematics, computer science, and engineering education that started at The University of Findlay. I would like to thank David Wallach and Janet Roll in the Department of Mathematics and Richard Corner and Craig Gunnett in the Department of Computer Science at Findlay for helping to develop my interests in these fields. The Department of Mathematics and Statistics at Miami University further cultivated my mathematical interests. I would like to thank Douglas Ward, Stephen Wright, and Olga Brezhneva for allowing me to see the application of mathematics to real-world problems and Dennis Keeler for introducing me to algebraic geometry.

During my first semester at the University of Notre Dame, I met my advisor, Andrew Sommese. Through him, I met the following people who have helped me with the research in this dissertation that I would like to thank: Bei Hu, Chris Peterson, and Charles Wampler for reading my dissertation and providing many suggestions for improvements; Dan Bates for our many thoughtful discussions regarding Bertini and numerical algebraic geometry; Wenrui Hao, Yuan Liu, Juan Migliore, and Yong-Tao Zhang for expanding the use of numerical algebraic geometry to other areas of research; and Gian Mario Besana, Wolfram Decker,

Sandra Di Rocco, T.Y. Li, Gerhard Pfister, Frank-Olaf Schreyer, Jan Verschelde, and Zhonggang Zeng for the many conversations regarding algebraic geometry and numerical algebraic geometry. Additionally, I would like to thank the Department of Mathematics and the Center for Applied Mathematics at Notre Dame for their support.

It is impossible for me to describe the admiration and respect that I have for my advisor, Andrew Sommese. A few lines in this acknowledgment section can never be enough to describe everything that he has done for me over the past few years. His guidance, knowledge, experience, time, and resources made this dissertation possible and I am looking forward to our continued collaboration and friendship in the years to come.

I would be remised if I did not thank my family for making my college experience possible. I would like to thank my parents, Russell and Virginia Hauenstein, for teaching me the value of an education and sharing their knowledge with me, my sister, Nicole Busey, and brother, Nathan Hauenstein, for demonstrating to me how to be successful, and my grandparents, aunts, uncles, in-laws, other relatives, and friends for their love and support.

Finally, I would like to thank my wife, Julie. Her love, encouragement, and support helped me to turn long hours of research into this dissertation.

CHAPTER 1

INTRODUCTION

Classically, algebraic geometry is the branch of mathematics that studies the solution sets of systems of polynomial equations. Since algebraic geometry can be studied from different points of view, the subject developed with different schools introducing new ideas used to solve open problems. The subject developed due to many accomplished mathematicians including Riemann, Max and Emmy Noether, Castelnuovo, Severi, Poincaré, Lefschetz, Weil, Zariski, Serre, and Grothendieck. To find more details regarding the foundation of algebraic geometry, please see [8, 9, 15, 16, 33].

Modern computational algebraic geometric began in the middle of the twentieth century with the introduction of modern computers. In Buchberger's 1965 Ph.D. thesis [7], he introduced Gröbner basis techniques that today form the foundation of modern symbolic computational algebraic geometry. The method underlying modern numeric computational algebraic geometry was started in 1953 when Davidenko realized that numerical methods for solving ordinary differential equations could be applied to solving systems of nonlinear equations [10, 11]. This method, called homotopy continuation, was later used to compute isolated solutions of polynomial systems. With the ability to compute isolated solutions and using the classical notion of linear space sections and generic points, Sommese and Wampler developed an approach for describing all solutions for a given polynomial

system and coined the phrase *Numerical Algebraic Geometry* [43]. For extensive details on homotopy continuation and numerical algebraic geometry, please see [44].

One of the fundamental algorithms in the field of numerical algebraic geometry is the computation of the numerical irreducible decomposition for the solution set of a polynomial system. This algorithm consists of three main parts: computing a witness superset, filtering out the junk points to create a witness set, and decomposing the witness set into irreducible components. This was done in a sequence of papers by Sommese and Wampler [43], Sommese and Verschelde [38], and Sommese, Verschelde, and Wampler [37, 39–41]. Chapter 2 provides the necessary background information and details regarding this algorithm. This thesis presents new and efficient algorithms to compute a witness superset and to filter junk points.

The cascade algorithm [38] and dimension-by-dimension slicing [43] are the two standard algorithms used to compute a witness superset. The two main disadvantages of the cascade algorithm is the number of paths to track the inability to reduce the number of variables in the system by using intrinsic slicing. The main disadvantage of dimension-by-dimension slicing is that valuable information regarding all larger dimensions is not utilized when solving the current dimension, which generally leads to a larger number of junk points in the witness superset. An equation-by-equation algorithm based on regeneration, called the regenerative cascade, is presented in Chapter 3 to overcome these disadvantages to compute witness supersets efficiently. Regeneration and the regenerative cascade is joint work with Sommese and Wampler.

A dimension k junk point is a point that lies on a component of dimension

larger than k. The standard algorithm for filtering junk points is the homotopy membership test [40]. This test checks each point in the witness superset for a given dimension to determine if it is a member of a component of higher dimension. The main disadvantage of this test is that all of the higher dimensional components need to be known to properly determine the junk points for a given dimension. The local dimension test presented in Chapter 4 uses local information to determine the maximum dimension of the components through the given point. This provides a junk point filtering method by identifying the points in a witness superset whose local dimension is larger than expected. The local dimension test is joint work with Bates, Peterson, and Sommese.

The numerical irreducible decomposition is implemented in Bertini [2, 5] with the ability to use either the cascade algorithm, dimension-by-dimension slicing, or the regenerative cascade for computing a witness superset and either the membership test or local dimension test for junk point filtering. Computational results from computing the numerical irreducible decomposition for a variety of examples using the various algorithms are presented in Chapter 5.

CHAPTER 2

BACKGROUND MATERIAL

The following sections provides a brief overview of commutative algebra and the theory and computation of the numerical irreducible decomposition. This information, in expanded details, can be found in [8, 14, 16, 44].

2.1 Commutative algebra

Let $\mathbb{C}$ denote the field of complex numbers and consider the ring of polynomials $R = \mathbb{C}[x_1, \ldots, x_N]$. For $f_1, \ldots, f_n \in R$, $f = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$ is a polynomial system in the variables $x_1, \ldots, x_N$ with complex coefficients. Define the *ideal generated by* f, denoted $I(f) = I(f_1, \ldots, f_n)$, as

$$I(f) = \left\{ \sum_{i=1}^{n} g_i f_i : g_i \in R \right\}.$$

The ideal $I(f)$ is the algebraic object associated with a polynomial system f. Moreover, for any ideal I, the Hilbert Basis Theorem yields that there is a polynomial system f such that $I = I(f)$.

Each polynomial system can also be viewed geometrically as an algebraic set, which is the set of points where the polynomial system vanishes. That is, for a

polynomial system f, define the *algebraic set associated with* f, denoted $V(f) = V(f_1, \ldots, f_n)$, as

$$V(f) = \{y \in \mathbb{C}^N : f(y) = 0\}.$$

If f and g are polynomial systems such that $I(f) = I(g)$, then $V(f) = V(g)$. However, if $I(f) \neq I(g)$, it could still happen that $V(f) = V(g)$, e.g., $V(x^2) = V(x)$.

An algebraic set V is said to be *reducible* if V can be written as $V = V_1 \cup V_2$ for proper algebraic subsets $V_1, V_2 \subset V$. An algebraic set that is not reducible is called *irreducible*. An irreducible algebraic set, called a *variety*, has the property that its set of smooth points is connected. The following proposition describes the irreducible decomposition for algebraic sets.

Proposition 2.1.1 (Irreducible decomposition of algebraic sets). *For an algebraic set V, there is a unique collection of varieties $V_1, \ldots, V_k$ such that $V = V_1 \cup \cdots \cup V_k$ and $V_i \not\subset V_j$ when $i \neq j$.*

The following definition introduces common concepts related to algebraic sets and varieties.

Definition 2.1.2. *Let V be a variety and A be an algebraic set with varieties $A_1, \ldots, A_k$ forming its irreducible decomposition.*

- *The* **dimension** *of V, denoted* $\dim(V)$*, is the dimension of the tangent space at a smooth point on V.*
- *The* **degree** *of V, denoted* $\deg(V)$*, is the number of points in the intersection of V with a generic linear space of codimension* $\dim(V)$*.*
- $\dim(A) = \max\{\dim(A_i) : 1 \leq i \leq k\}$.

- *A is* **pure-dimensional** *if* $\dim(A) = \dim(A_i)$*, for* $1 \leq i \leq k$*.*

The irreducible decomposition for an algebraic set can be rewritten using pure-dimensional algebraic sets. For an algebraic set V of dimension d, there is a unique collection of pure-dimensional algebraic sets $V_0, \ldots, V_d$, with $\dim(V_i) = i$, such that $V = \bigcup_{i=0}^{d} V_i$. For each i, let $V_{i,1}, \ldots, V_{i,k_i}$ be an irreducible decomposition of V_i. Up to reordering, V can be uniquely written as

$$V = \bigcup_{i=0}^{d} V_i = \bigcup_{i=0}^{d} \bigcup_{j=1}^{k_i} V_{ij}. \tag{2.1.1}$$

Since a variety is a set, it contains no multiplicity information. By relating a variety V to a polynomial system f with $V \subset V(f)$, we can assign a multiplicity to V with respect to f.

Definition 2.1.3. *Let f be a polynomial system and $V \subset V(f)$ be a variety. The* **multiplicity of V with respect to** *f is the multiplicity of a smooth point of V as a solution of $f = 0$.*

The multiplicity assigned to a variety is dependent upon the associated polynomial system. For example, $V = \{0\}$ has multiplicity 1 with respect to $f(x) = x$ and has multiplicity 2 with respect to $f(x) = x^2$.

With multiplicity information, we can say that a variety with respect to a polynomial system is either generically reduced or generically nonreduced.

Definition 2.1.4. *Let f be a polynomial system and $V \subset V(f)$ be a variety. The variety V is called* **generically reduced with respect to** *f if it has multiplicity 1 with respect to f and is called* **generically nonreduced with respect to** *f if it has multiplicity $k > 1$ with respect to f.*

Continuing with the example above, $V = \{0\}$ is generically reduced with respect to $f(x) = x$ and generically nonreduced with respect to $f(x) = x^2$.

For a matrix A, let $\text{null}(A)$ denote the dimension of the null space of A. The following proposition describes the nullity of the Jacobian matrix of a polynomial system f for generically reduced and generically nonreduced varieties with respect to f.

Proposition 2.1.5. *Let f be a polynomial system, $J(x)$ be the Jacobian of f at x, and $V \subset V(f)$ be a variety of dimension d.*

1. *V is generically reduced with respect to f if and only if $\text{null}(J(x)) = d$ for generic $x \in V$.*

2. *V is generically nonreduced with respect to f if and only if $\text{null}(J(x)) > d$ for every $x \in V$.*

2.2 Numerical irreducible decomposition

The numerical irreducible decomposition is a numerical representation that is analogous to Eq. 2.1.1 first presented in [39]. This section provides the underlying theory of the decomposition with Section 2.4 describing its computation. The numerical representation used throughout this thesis for a variety is based on the classical idea of generic linear space sections summarized in the following theorem.

Theorem 2.2.1. *Let V be an algebraic set, $V = \bigcup_{i=0}^{d} V_i = \bigcup_{i=0}^{d} \bigcup_{j=1}^{k_i} V_{i,j}$ be its irreducible decomposition, and L_ℓ be a generic linear space of codimension ℓ.*

1. *$L_\ell \cap V_i = \emptyset$ for $0 \leq i < \ell$.*

2. *For $1 \leq j \leq k_\ell$, $L_\ell \cap V_{\ell,j}$ consists of $\deg(V_{\ell,j})$ isolated points which do not lie on any other variety.*

3. *$L_\ell \cap V_{i,j}$ is a degree* $\deg(V_{i,j})$ *variety of dimension* $i - \ell$ *for* $\ell < i \leq d$ *and* $1 \leq j \leq k_i$.

Let V be a pure-dimensional algebraic set of dimension d with irreducible decomposition $V = \bigcup_{j=1}^{k} V_j$, W be a variety of dimension d and degree r, and L_d be a generic linear space of codimension d. An *irreducible witness point set* for W is the set $W \cap L_d$ consisting of r isolated points. A *witness point set* for V is the set $V \cap L_d$, which consists of isolated points each lying in exactly one irreducible witness point set $V_j \cap L_d$. The witness point set $V \cap L_d$ can be written uniquely as the union of irreducible witness points sets, namely

$$V \cap L_d = \left(\bigcup_{j=1}^{k} V_j \right) \cap L_d = \bigcup_{j=1}^{k} (V_j \cap L_d).$$

The witness point sets provide information regarding the pure-dimensional algebraic sets, but they do not uniquely identify it. That is, for $d > 0$, given L_d and the set $V \cap L_d$, there are many possibilities for V. Along with a witness point set, additional information is needed to uniquely identify V.

Let f be a polynomial system, $V \subset V(f)$ be a variety of dimension d and degree r, and L_d be a generic linear space of codimension d. A *witness set* for V is the collection $\{f, L_d, V \cap L_d\}$, which defines V uniquely.

The numerical irreducible decomposition utilizes the union of witness sets. To understand this union, let f be a polynomial system with $V_1, V_2 \subset V(f)$ varieties and let W_{V_i} be the witness set for V_i. If $\dim(V_1) \neq \dim(V_2)$, the witness set for $V_1 \cup V_2$ is a formal union. That is, $W_{V_1 \cup V_2} = W_{V_1} \cup W_{V_2} = \{W_{V_1}, W_{V_2}\}$. If $\dim(V_1) = \dim(V_2) = d$ and L_d is a generic linear space of codimension d, the witness set for $V_1 \cup V_2$ uses a union of witness point sets. That is, $W_{V_1 \cup V_2} = W_{V_1} \cup W_{V_2} = \{f, L_d, (V_1 \cup V_2) \cap L_d\}$ A witness set W_V corresponding to an algebraic

set V is said to be *irreducible* if V is a variety, i.e. irreducible algebraic set.

With the concept of witness sets, we can define the numerical irreducible decomposition for $V(f)$.

Definition 2.2.2. *For a polynomial system f with $d = \dim(V(f))$, let $\bigcup_{i=0}^{d} V_i = \bigcup_{i=0}^{d} \bigcup_{j=1}^{k_i} V_{i,j}$ be the irreducible decomposition of $V(f)$. A* **numerical irreducible decomposition** *of $V(f)$ is*

$$W = \bigcup_{i=0}^{d} W_i = \bigcup_{i=0}^{d} \bigcup_{j=1}^{k_i} W_{i,j} \tag{2.2.1}$$

where $W_i = \bigcup_{j=1}^{k_i} W_{i,j}$ is a witness set for the i-dimensional algebraic set V_i and $W_{i,j}$ is an irreducible witness set for the variety $V_{i,j}$.

Even though an irreducible witness set provides the information needed to perform computations on the associated variety, generically nonreduced varieties require additional structures for its witness set to be numerically useful. Numerical difficulties arise because of Prop. 2.1.5, namely the rank of the Jacobian at each point is smaller than expected. One way to overcome this is to the use the process of deflation, which was introduced by Ojika, Watanabe, and Mitsui [36] and improved by Ojika [35]. Leykin, Verschelde, and Zhao [26] refined the deflation process for isolated roots (see also [24]), and Sommese and Wampler [44] observed that the deflation procedure may be done for a variety as a whole.

Let f be a polynomial system and $V \subset V(f)$ be a generically nonreduced variety. Deflation constructs a projection π and a polynomial system g, which has a generically reduced variety $\widehat{V} \subset V(g)$, such that π is a generic one-to-one projection from $\widehat{V}$ onto V. The key properties of π are summarized in the following proposition.

Proposition 2.2.3. *Let $\widehat{V}$ and V be varieties with $x \in V$ and let $\pi : \widehat{V} \to V$ be generically one-to-one and onto. Then, there exists $\widehat{x} \in \widehat{V}$ with $\pi(\widehat{x}) = x$, and, if x is generic, then $\widehat{x}$ is unique.*

Due to Prop. 2.2.3, all computations in this thesis involving a generically nonreduced variety V can be completed using a deflated variety $\widehat{V}$. If $\{f, L_d, V \cap L_d\}$ is a witness set for V, we write the witness set for the deflated variety $\widehat{V}$ as $\{g, \pi, L_d, W\}$ where W is the set of points on $\widehat{V}$ such that $\pi(W) = V \cap L_d$. By genericity, $|W| = |V \cap L_d|$.

See [44] for an extensive discussion of witness sets in full generality.

2.3 Homotopy continuation

Homotopy continuation is the main computational tool in numerical algebraic geometry. A homotopy is a map $H(x, q) : \mathbb{C}^N \times \mathbb{C}^M \to \mathbb{C}^n$, and, in this thesis, we require that H is polynomial in x and complex analytic in q. If $n = N$, the homotopy is called *square*.

In this thesis, a linear homotopy between polynomial systems $f, g : \mathbb{C}^n \to \mathbb{C}^n$ is of the form

$$H(x, t) = (1 - t)f(x) + \gamma t g(x)$$

where $\gamma \in \mathbb{C}$ is generic. The system $f(x) = H(x, 0)$ is called the *target system* and $g(x) = H(x, 1)$ is called the *start system*.

The start system g is chosen with structure related to f, e.g., $\deg(g_i) = \deg(f_i)$ or g_i and f_i have the same m-homogeneous structure. See [44] for more details on different types of start systems. Consider solution paths $x(t)$ such that

$$H(x(t), t) \equiv 0. \tag{2.3.1}$$

By denoting $H_x(x,t)$ and $H_t(x,t)$ to be the partial derivatives of $H(x,t)$ with respect to x and t, respectively, Davidenko [10, 11] observed that $x(t)$ satisfies the ordinary differential equation

$$0 \equiv \frac{dH(x(t),t)}{dt} = H_x(x(t),t)\frac{dx(t)}{dt} + H_t(x(t),t). \tag{2.3.2}$$

Homotopy continuation computes the limit points $x(0) = \lim_{t\to 0} x(t)$, where $x(t)$ solves Eqs. 2.3.1 and 2.3.2 and $x(1)$ is an isolated solution of g. The isolated solutions of f are contained in this set of limit points.

The path $x(t)$ is numerically tracked using a predictor/corrector scheme, such as Euler prediction with Newton correction. If (x_0, t_0) approximately lies on a path $x(t)$, e.g., $H(x_0, t_0) \approx 0$, using Eq. 2.3.2, the Euler prediction at $t_1 = t_0 + \Delta t$ is $x_1 = x_0 + \Delta x$ where Δx solves $H_x(x_0, t_0)\Delta x = -H_t(x_0, t_0)\Delta t$. Using Eq. 2.3.1, the Newton correction at t_1 is $x_1 + \Delta x$ where Δx solves $H_x(x_1, t_1)\Delta x = -H(x_1, t_1)$.

A step consists of a prediction along with a few successive Newton corrections. If the Newton corrections converge to a predetermined tolerance, the step is considered successful. If the step is not successful, the stepsize Δt is decreased and the step is attempted again. Conversely, if a few successive steps are successful, the stepsize Δt is increased. This process is known as the adaptive stepsize method and is described in detail in [44]. Adaptive precision tracking [3] adjusts the precision that is used for the computations based on the local conditioning along the path. The methods of adaptive stepsize and adaptive precision tracking are combined in [4].

For completeness, we define the algorithm *homotopy_solve* that, given a square polynomial system f, constructs a finite set of points X that contains the isolated solutions of f.

Algorithm 2.3.1. *homotopy_solve*$(f; X)$

Input:

• f*: a system of* n *polynomials in* $\mathbb{C}[x_1, \ldots, x_n]$.

Output:

• X*: a finite set of points in* $\mathbb{C}^n$ *that contains the isolated solutions of* f.

Algorithm:

1. *Construct a start system* g *related to* f *with known isolated solutions which are all nonsingular. For a summary of ways to construct such a start system* g*, see [27, 44].*

2. *Construct the linear homotopy* $H(x,t) = (1-t)f(x) + \gamma t g(x)$*, for random* $\gamma \in \mathbb{C}$.

3. *Let* X *be the set of limit points* $x(0)$ *that lie in* $\mathbb{C}^n$ *for the paths* $x(t)$ *where* $x(1)$ *is an isolated solution of* g.

2.3.1 Total degree of a polynomial system

Let $f : \mathbb{C}^n \to \mathbb{C}^n$ be a polynomial system and $d_i = \deg f_i$. The *total degree* of f is $d_1 d_2 \cdots d_n$. Bézout's Theorem [44] states that the number of isolated solutions of f, counting multiplicity, is at most the total degree of f.

2.3.2 Endpoints at infinity

For a homotopy $H(x,t)$ and a path $x(t)$, it often happens that the limit point $x(0) = \lim_{t \to 0} x(t)$ diverges to infinity. Such paths are numerically difficult to track and are infinitely long. One way to handle this is to homogenize the system and use a generic patch [30]. That is, the polynomials on $\mathbb{C}^n$ are homogenized

to obtain polynomials on $\mathbb{P}^n$. The computations are then performed on a generic patch of $\mathbb{P}^n$ by restricting to a generic hyperplane in $\mathbb{C}^{n+1}$. This transforms the infinitely long paths that diverge to infinity into finite length paths that converge in $\mathbb{C}^{n+1}$. See [44] for more information.

2.3.3 Complete homotopy

The notions of *trackable path* and *complete homotopy* are theoretical constructs introduced in [17] corresponding to the numerical homotopy method.

Definition 2.3.2 (Trackable path). *Let $H(x,t) : \mathbb{C}^n \times \mathbb{C} \to \mathbb{C}^n$ be polynomial in x and complex analytic in t and let $\hat{x}$ be an isolated solution of $H(x,1) = 0$. We say that $\hat{x}$ is* **trackable** *(or equivalently we say that we can* **track** *$\hat{x}$) for $t \in (0,1]$ from $t = 1$ to $t = 0$ using $H(x,t)$ if*

1. *when $\hat{x}$ is nonsingular, there is a smooth map $\psi_{\hat{x}} : (0,1] \to \mathbb{C}^n$ such that $\psi_{\hat{x}}(1) = \hat{x}$ and $\psi_{\hat{x}}(t)$ is a nonsingular isolated solution of $H(x,t) = 0$ for $t \in (0,1]$; and*

2. *when $\hat{x}$ is singular, letting $\widehat{H}(x,z,t) = 0$ denote the system that arises through deflation, and letting $(\hat{x},\hat{z})$ denote the nonsingular isolated solution of $\widehat{H}(x,z,1) = 0$ over $\hat{x}$, we can track the nonsingular solution $(\hat{x},\hat{z})$ of $\widehat{H}(x,z,1)$ for $t \in (0,1]$ from $t = 1$ to $t = 0$, i.e., there is a smooth map $\psi_{\hat{x}} : (0,1] \to \mathbb{C}^n \times \mathbb{C}^{n'}$ such that $\psi_{\hat{x}}(1) = (\hat{x},\hat{z})$ and $\psi_{\hat{x}}(t)$ is a nonsingular isolated solution of $H(x,z,t) = 0$ for $t \in (0,1]$*

By the **limit of the tracking** *using $H(x,t) = 0$ of the point $\hat{x}$ as t goes to 0, we mean $\lim_{t\to 0} \psi_{\hat{x}}(t)$ in case (1) and the x coordinates of $\lim_{t\to 0} \psi_{\hat{x}}(t)$ in case (2).*

With the formal definition of trackable paths, we can define a complete homotopy.

Definition 2.3.3 (Complete homotopy). *Let $H(x,t) : \mathbb{C}^n \times \mathbb{C} \to \mathbb{C}^n$ be polynomial in x and complex analytic in t. Let S be a finite set of points in $V(H(x,1))$. Then, $H(x,t)$ with S is a* **complete homotopy** *for an algebraic set $Y \subset \mathbb{C}^n$ if*

1. *every point in S is trackable; and*
2. *every isolated point in Y is the limit of at least one such path.*

2.3.4 Parameter continuation

Let $f(x,q) : \mathbb{C}^n \times \mathbb{C}^M \to \mathbb{C}^n$ be polynomial in x and complex analytic in q and let S be the set of isolated (respectively, nonsingular) points in $V(f(x,q_1))$ for a generic $q_1 \in \mathbb{C}^M$. For any $q_0 \in \mathbb{C}^M$, the theory of parameter continuation states that the homotopy $H(x,t) = f(x, tq_1 + (1-t)q_0)$ with start points S is a complete homotopy for finding the isolated [44] (respectively, nonsingular [31]) points in $V(f(x,q_0))$.

Let $Y \subset \mathbb{C}^n$ be an irreducible algebraic set of dimension k and Y^* be a proper algebraic subset of Y. The set $X = Y \setminus Y^*$ is called a *quasiprojective algebraic set*. The following theorem presents a slightly stronger statement of parameter continuation that follows from [44, § A.14].

Theorem 2.3.4 (Parameter continuation). *Let X_k be a quasiprojective algebraic set of dimension k. Let $f(x,q) : \mathbb{C}^n \times \mathbb{C}^M \to \mathbb{C}^k$ be polynomial in x and complex analytic in q, $q_1 \in \mathbb{C}^M$ be generic, and S be the set of isolated (respectively, nonsingular) points in $V(f(x,q_1)) \cap X_k$. Then, $H(x,t) = f(x, tq_1 + (1-t)q_0)$*

with start points S is a complete homotopy for finding the isolated (respectively, nonsingular) points in $V(f(x, q_0)) \cap X_k$.

2.3.5 Product decomposition

Regeneration, as presented in Chapter 3, depends upon the construction of a product decomposition, which was introduced in [32] with related ideas in [49]. Let V_1 and V_2 be finite dimensional $\mathbb{C}$-vector spaces of polynomials on $\mathbb{C}^n$. That is, for each V_i, $i = 1, 2$, there is a set of basis polynomials $\{\alpha_{i,1}, \ldots, \alpha_{i,k_i}\}$ such that each polynomial in V_i is a $\mathbb{C}$-linear combination of the basis polynomials, denoted as $V_i = \langle \alpha_{i,1}, \ldots, \alpha_{i,k_i} \rangle$. The image of $V_1 \otimes V_2$ in the space of polynomials is a vector space of polynomials whose basis is all products $\alpha_{1,j}\alpha_{2,\ell}$. A product decomposition of a polynomial f on $\mathbb{C}^n$ is a list $\mathbb{V} := \{V, V_1, \ldots, V_m\}$ of $\mathbb{C}$-vector spaces $V, V_1, \ldots, V_m$ of polynomials on $\mathbb{C}^n$ such that f is in the image V of $V_1 \otimes \cdots \otimes V_m$ in the space of polynomials. By selecting a generic polynomial from each V_i and setting g as their product, it is clear that g is in the image V. Such a polynomial g is called a *generic product member* of V. Since a generic product member is factored, it is easier to solve than a general member of V, which is a sum of products. Product decomposition methods construct start systems by using generic product members as stated in the following theorem.

Theorem 2.3.5 (Product decomposition [17]). *Suppose that $X_k \subset \mathbb{C}^n$ is a quasiprojective algebraic set of dimension k. Let $f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_k(x) \end{bmatrix}$ be a polynomial system on $\mathbb{C}^n$ and let $\mathbb{V}_i = \{V_i, V_{i,1}, \ldots, V_{i,d_i}\}$ be a product decomposition for each f_i. For $i = 1, \ldots, k$, let g_i be a generic product member of V_i and let S be the set of isolated (respectively, nonsingular isolated) points in $V(g_1, \ldots, g_k) \cap X_k$. Then,*

for a generic $\gamma \in \mathbb{C}$, the homotopy

$$H(x,t) = \begin{bmatrix} (1-t)f_1(x) + \gamma t g_1(x) \\ \vdots \\ (1-t)f_k(x) + \gamma t g_k(x) \end{bmatrix} \tag{2.3.3}$$

with start set S is a complete homotopy for the isolated (respectively, nonsingular isolated) points in $V(f) \cap X_k$.

The regeneration algorithms presented in Chapter 3 use a special case of product decomposition, namely linear product decomposition, where each generic product member is a product of linear functions. Linear products are nearly identical to the set structures described in [49], but the theory presented there only covers nonsingular solutions on $X = \mathbb{C}^N$ and each set must contain 1. The following section defines terms used in linear products.

2.3.6 Linear support

A linear product decomposition for a polynomial can be written in terms of the variables that appear in that polynomial. For polynomials that arise in practice, this can be a much smaller subset that the set of all variables. The following defines the support and support base of a polynomial.

Definition 2.3.6. *Let $g(x_1, \ldots, x_n)$ be a polynomial. The* **support** *of g is the set of all monomials that appear in g and the* **support base** *of g is the union of the subset of variables that appear in g with* 1*.*

For sets of monomials C and D, define $C \otimes D$ as the set consisting of the products of monomials in C and D. If g is a degree d polynomial with support

M and support base B, it is clear that

$$M \subset \underbrace{B \otimes \cdots \otimes B}_{d \text{ times}}.$$

In particular, if $V_i = \langle B \rangle$, $i = 1, \ldots, d$ and $V = V_1 \otimes \cdots \otimes V_d$, then $\{V, V_1, \ldots, V_d\}$ is a (linear) product decomposition for g. The following defines terminology for constructing a linear product decomposition for g.

Definition 2.3.7. *Let $g(x_1, \ldots, x_n)$ be a polynomial with support base B. A set $S \subset \{1, x_1, \ldots, x_n\}$ is a* **linear support set** *for g if $B \subset S$. The vector space $V = \langle S \rangle$ is the* **linear support vector space** *associated to linear support set S. A linear function L is a* **support linear** *for g if L is in a linear support vector space, and L is a* **generic support linear** *if it has generic coefficients. The zero set $V(L)$ is called a* **support hyperplane**. *A* **minimal support linear** *for g is a support linear in $\langle B \rangle$, and its zero set is called a* **minimal support hyperplane**.

2.3.7 Randomization

Let $f : \mathbb{C}^N \to \mathbb{C}^n$ be a polynomial system and $X \subset V(f)$ be a variety of dimension d. As described in Section 2.2, the numerical representation of X is the witness set $\{f, L_d, X \cap L_d\}$ where L_d is a generic linear space of codimension d. The linear space L_d is defined by a system of d linear equations $\ell(x) = 0$. The process of randomization creates a system $f_R : \mathbb{C}^N \to \mathbb{C}^{N-d}$ so that the augmented system $\begin{bmatrix} f_R \\ \ell \end{bmatrix}$ is square and each point of $X \cap L_d$ is an isolated solution.

To construct a randomized system f_R, let I_{N-d} be the $(N-d) \times (N-d)$

identity matrix and A be an $(N-d)\times(n-N+d)$ matrix over $\mathbb{C}$. Consider

$$[I_{N-d}\ A]\,f = \begin{bmatrix} f_1 \\ \vdots \\ f_{N-d} \end{bmatrix} + A \begin{bmatrix} f_{N-d+1} \\ \vdots \\ f_n \end{bmatrix}. \tag{2.3.4}$$

As justified by the following theorem, for a generic A, we use the randomized system $f_R = [I_{N-d}\ A]f$. To minimize the total degree of f_R, we will always reorder the f_i so that $\deg f_1 \geq \cdots \geq \deg f_n$. Since randomization is used throughout this thesis, we shall, following the notation of [44], denote the randomization f_R as $\mathfrak{R}(f; N-d)$.

Theorem 2.3.8. *Let $f : \mathbb{C}^N \to \mathbb{C}^n$ be a polynomial system and $X \subset \mathbb{C}^N$ be a variety of dimension d. For generic $A \in \mathbb{C}^{k\times(n-k)}$,*

1. *if $d > N-k$, then $X \subset V(f)$ if and only if $X \subset V(\mathfrak{R}(f;k))$,*
2. *if $d = N-k$, then $X \subset V(f)$ implies that $X \subset V(\mathfrak{R}(f;k))$, and*
3. *if $X \subset V(f)$ is of multiplicity m with respect to f, then $X \subset V(\mathfrak{R}(f;k))$ is of multiplicity $\widetilde{m} \geq m$ with repect to $\mathfrak{R}(f;k)$, and $m = 1$ implies $\widetilde{m} = 1$.*

2.3.8 Extrinsic and intrinsic homotopies

Homotopies of the form

$$H(x,t) = \begin{bmatrix} H_1(x,t) \\ \ell(x) \end{bmatrix},$$

where $H_1 : \mathbb{C}^n \times \mathbb{C} \to \mathbb{C}^k$ is polynomial in x and complex analytic in t and $\ell : \mathbb{C}^n \to \mathbb{C}^{n-k}$ defines a k-dimensional linear space, arise often, e.g., in *dim_slicing_k* in

Section 2.4.1 and in *regenerate* and *regen_cascade* in Chapter 3. This homotopy $H(x,t)$ is called an *extrinsic* homotopy.

The k-dimensional linear space $V(\ell)$ can be written intrinsically using linear algebra. That is, there is a rank k matrix $A \in \mathbb{C}^{n\times k}$ and vector $b \in \mathbb{C}^n$ such that $V(\ell) = \{Au + b \ : \ u \in \mathbb{C}^k\}$, i.e., $\ell(Au+b) = 0$, for all $u \in \mathbb{C}^k$. The homotopy $\widehat{H}(u,t) : \mathbb{C}^k \times \mathbb{C} \to \mathbb{C}^k$, defined by $\widehat{H}(u,t) = H_1(Au+b,t)$, is the *intrinsic* homotopy corresponding to H. The intrinsic homotopy $\widehat{H}$ can be evaluated efficiently in a straight-line fashion, that is, given u, compute $x = Au + b$ and then evaluate $H_1(x,t)$. Additionally,

$$\frac{\partial \widehat{H}(u,t)}{\partial u} = \frac{\partial H_1(Au+b,t)}{\partial x} \cdot A.$$

When $n \gg k$, the intrinsic homotopy is more efficient to use since it reduces the number of variables to k from n. As k increases, the advantage of tracking using k variables instead of n variables is canceled out by the extra cost of evaluating $Au+b$ and $\frac{\partial \widehat{H}}{\partial u}$. For the implementation of the algorithms *dim_slicing_k*, *regenerate*, and *regen_cascade* in the software package Bertini [2, 5], the intrinsic formulation is automatically used when it is advantageous.

2.4 Computing a numerical irreducible decomposition

The foundation of numerical algebraic geometry is the computation of the numerical irreducible decomposition. This computation, presented below as *numerical_irreducible_decomposition*, depends upon the three algorithms *witness_superset*, *junk_removal*, and *irreducible_decomp*. Section 2.4.1 presents two *witness_superset* algorithms for computing a witness superset. Section 2.4.2 presents a *junk_removal* algorithm for creating a witness set by removing the junk points from the witness

superset using a membership test. Section 2.4.3 presents an *irreducible_decomp* algorithm for decomposing the witness set into the irreducible components.

Algorithm 2.4.1. *numerical_irreducible_decomposition*$(f; W)$

Input:

- f*: a system of* n *polynomials in* $\mathbb{C}[x_1, \ldots, x_N]$.

Output:

- W*: witness set for* $V(f)$ *decomposed as in Eq. 2.2.1.*

Algorithm:

1. $[\widehat{W}] :=$ *witness_superset*(f).
2. $[W_p] :=$ *junk_removal*$(f, \widehat{W})$.
3. $[W] :=$ *irreducible_decomp*(f, W_p).

2.4.1 Computing a witness superset

The first step in computing the numerical irreducible decomposition for a polynomial system is to compute a *witness superset* $\widehat{W}$.

Definition 2.4.2 (Witness superset). *Let* f *be a polynomial system and* $\bigcup_{i=0}^{d} V_i = \bigcup_{i=0}^{d} \bigcup_{j=0}^{k_i} V_{i,j}$ *be the irreducible decomposition of* $V(f)$. $\widetilde{W}_i$ *is a* **witness point superset** *for* V_i *if, for a generic linear space* L_i *of codimension* i*,*

1. $|\widetilde{W}_i| < \infty$ *and*
2. $V_i \cap L_i \subset \widetilde{W}_i \subset V(f) \cap L_i$.

A **witness superset** *for* $V(f)$ *is* $\widehat{W} = \bigcup_{i=0}^{d} \{f, L_i, \widetilde{W}_i\}$.

Using the notation above and letting $W_i = V_i \cap L_i$ be a witness point set for the algebraic set V_i, we can write $\widetilde{W_i} = W_i \cup J_i$. The points in J_i lie on algebraic sets of dimension larger than i, i.e., $J_i \subset \bigcup_{j=i+1}^{d} V_j$. The set J_i is called the set of *junk points* for $\widetilde{W_i}$ and are filtered out of $\widetilde{W_i}$ by *junk_removal*. It should be noted that $J_d = \emptyset$ meaning that the witness point superset for the top dimension is a witness point set.

There are two algorithms for computing a witness superset, namely the dimension-by-dimension slicing approach presented in [43] and the cascade algorithm presented in [38], denoted *dim_slicing* and *cascade*, respectively. Before presenting these algorithms, we need to consider the rank of a polynomial system and a probabilistic null test.

Theorem 2.4.3 (Rank of a polynomial system). *Let* $f : \mathbb{C}^N \to \mathbb{C}^n$ *and* $\widehat{x} \in C^N$ *be generic. The* **rank** *of* f *denoted* $\operatorname{rank}(f)$*, is* $\operatorname{rank}\left(\frac{\partial f}{\partial x}(\widehat{x})\right)$*. If* $V \subset V(f)$ *is an algebraic set, then* $\dim(V) \geq N - \operatorname{rank}(f)$*.*

Theorem 2.4.4 (Probabilistic null test). *Let* $f : \mathbb{C}^N \to \mathbb{C}^n$ *be a polynomial system and* $X \subset \mathbb{C}^N$ *be a variety. With probability* 1*, if* $\widehat{x} \in X$ *is random, then* $X \subset V(f)$ *if and only if* $f(\widehat{x}) = 0$*.*

The slicing approach computes a witness superset by solving independently the systems setup by slicing at each possible dimension. We first describe the algorithm for each dimension, namely *dim_slicing_k*, and then present the dimension-by-dimension slicing algorithm *dim_slicing*.

Algorithm 2.4.5. *dim_slicing_k*$(f, k; \widehat{W_k})$

Input:

- f*: a system of* n *polynomials in* $\mathbb{C}[x_1, \ldots, x_n]$*.*

• *k: an integer between $n - \operatorname{rank}(f)$ and n, inclusive.*

Output:

• *$\widehat{W}_k$: a witness superset for the k-dimensional algebraic subset of $V(f)$.*

Algorithm:

Case $k = n$.

1. *Choose a random $\widehat{x} \in \mathbb{C}^n$.*
2. *If $f(\widehat{x}) = 0$, then $\widehat{W}_k := \{f, \{\widehat{x}\}, \{\widehat{x}\}\}$. Otherwise, $\widehat{W}_k := \{f, \{\widehat{x}\}, \emptyset\}$.*

Otherwise.

1. *Let L_k be a generic linear space of codimension k defined by k linear functions $\mathfrak{L}(x)$.*
2. *Compute $X :=$ homotopy_solve$(\{\mathfrak{R}(f, n-k), \mathfrak{L}(x)\})$.*
3. *Let $\widetilde{W}_k := \{x \in X : f(x) = 0\}$.*
4. *Set $\widehat{W}_k := \{f, L_k, \widetilde{W}_k\}$.*

Algorithm 2.4.6. *dim_slicing$(f; \widehat{W})$*

Input:

• *f: a system of n polynomials in $\mathbb{C}[x_1, \ldots, x_n]$ of rank $r > 0$.*

Output:

• *$\widehat{W}$: a witness superset for $V(f)$.*

Algorithm:

1. *Define $d_i = \deg f_i$ and reorder the polynomials so that $d_1 \geq \cdots \geq d_n$.*
2. *Initialize $\widehat{W} := \emptyset$.*
3. *For $k := n, \ldots, n - r$, do the following:*

(a) $\widehat{W}_k := dim\_slicing\_k(f, k)$.

(b) $\widehat{W} := \widehat{W} \cup \widehat{W}_k$.

The reordering in Step 1 of *dim_slicing* is used to minimize the total number of paths to track. As mentioned in Section 2.3.8, the implementation of Step 2 of *dim_slicing_k* in Bertini [2, 5] uses an intrinsic homotopy when it is advantageous over the extrinsic homotopy.

The cascade algorithm computes a witness superset by using a sequence of homotopies to compute a witness point superset for each possible dimension, starting at the top dimension. The discussion here follows [44], which is a simplification of the original presentation in [38].

Let $f : \mathbb{C}^N \to \mathbb{C}^n$ and $r := \text{rank}(f)$. We first show that we can reduce to the case that $N = n = r$. By Thm. 2.4.3, we know that each algebraic subset of $V(f)$ has dimension at least $N - r$. Accordingly, by Thm. 2.3.8, we can replace f with $\mathfrak{R}(f, r)$ since all algebraic subsets of $V(f)$ are algebraic subsets of $V(\mathfrak{R}(f, r))$. That is, we can assume that $n = r$. Further, since the dimension of each variety of $V(f)$ is at least $N - r$, we will be slicing with a generic linear space that has codimension at least $N - r$. Let L_{N-r} be a generic linear space of codimension $N - r$. There is a matrix $B \in \mathbb{C}^{N\times r}$ and vector $b \in \mathbb{C}^N$ such that $L_{N-r} = \{By + b \ : \ y \in \mathbb{C}^r\}$. Accordingly, we can replace $f : \mathbb{C}^N \to \mathbb{C}^r$ with $g : \mathbb{C}^r \to \mathbb{C}^r$ where $g(y) = f(By + b)$. So, without loss of generality, we shall assume that $f : \mathbb{C}^N \to \mathbb{C}^N$ of rank N.

Before providing the justification for the cascade algorithm, we first need some notation and a definition. Let $1^{[i]} = (\underbrace{1, \ldots, 1}_{i}, \underbrace{0, \ldots, 0}_{N-i})$ and for $t = (t_1, \ldots, t_N) \in$

$\mathbb{C}^N$, let

$$T(t) = \begin{bmatrix} t_1 & & \\ & \ddots & \\ & & t_N \end{bmatrix}.$$

For $a, x \in \mathbb{C}^N$ and $A, \Lambda \in \mathbb{C}^{N\times N}$, let

$$L(a, A, x) = a + Ax,$$

$$\mathcal{E}(\Lambda, a, A, x, t) = f(x) + \Lambda \cdot T(t) \cdot L(a, A, x),$$

and

$$\mathcal{E}_i(\Lambda, a, A, x) = \mathcal{E}(\Lambda, a, A, x, 1^{[i]}).$$

The point x is called a *level i nonsolution* if $\mathcal{E}_i(\Lambda, a, A, x) = 0$ and $f(x) \neq 0$, with the set of level i nonsolutions denoted as $\mathcal{N}_i$.

The following two theorems provide the justification for the cascade algorithm.

Theorem 2.4.7. *Let $f : \mathbb{C}^N \to \mathbb{C}^N$ be a polynomial system of rank N. For generic $a \in \mathbb{C}^N$ and $A, \Lambda \in \mathbb{C}^{N\times N}$, an integer $0 \leq i \leq N$, and $\widehat{x} \in \mathbb{C}^N$ such that $\mathcal{E}_i(\Lambda, a, A, \widehat{x}) = 0$, then either*

1. *$\widehat{x} \in \mathcal{N}_i$, or*
2. *$\widehat{x}$ lies on variety of $V(f)$ of dimension at least i.*

Bertini's Theorem [44] provides that each point in $\mathcal{N}_i$ is a nonsingular isolated solution of $\mathcal{E}_i(\Lambda, a, A, x) = 0$. Using the points in $\mathcal{N}_i$ as start points for the homotopy

$$H_{i-1}(x, t) = \mathcal{E}_{i-1}(\Lambda, a, A, x)(1-t)+t\mathcal{E}_i(\Lambda, a, A, x) = \mathcal{E}(\Lambda, a, A, x, (1-t)1^{[i-1]}+t1^{[i]}),$$

the following theorem shows that the endpoints consist of $\mathcal{N}_{i-1}$ and a witness point superset for the $(i-1)$–dimensional varieties in $V(f)$.

Theorem 2.4.8. *Let $f : \mathbb{C}^N \to \mathbb{C}^N$ be a polynomial system of rank N. For generic $a \in \mathbb{C}^N$ and $A, \Lambda \in \mathbb{C}^{N\times N}$, and integer $1 \leq i \leq N$, there are nonsingular solution paths $t \in \mathbb{C} \to (\phi(t), t) \in \mathbb{C}^N \times \mathbb{C}$ such that $\phi(1) \in \mathcal{N}_i$ and $H_{i-1}(\phi(t), t) \equiv 0$. Let S consists of the limit points $\phi(0)$ in $\mathbb{C}^N$. Then,*

1. *$\mathcal{N}_{i-1} = \{x \in S \ : \ f(x) \neq 0\}$ and*
2. *$\widetilde{W}_{i-1} = \{x \in S \ : \ f(x) = 0\}$ is a witness point superset for the $(i-1)$–dimensional varieties in $V(f)$.*

To avoid triviality, *cascade* assumes that the input polynomial system f has $r = \text{rank}(f) > 0$, i.e., we can start the algorithm by solving for dimension $r-1$. The cascade algorithm is started by using *homotopy_solve* to compute the solutions of $\mathcal{E}_{r-1}$. When using randomization, Thm. 2.3.8 provides that the bottom dimension could contain extraneous points that can be identified by evaluating the original system f.

Algorithm 2.4.9. *cascade*$(f; W)$

Input:

- *f: a system of n polynomials in $\mathbb{C}[x_1, \ldots, x_N]$ of rank $r > 0$.*

Output:

- *$\widehat{W}$: a witness point superset for $V(f)$.*

Algorithm:

1. *Initialize $\widehat{W} := \emptyset$ and $d := N - r$.*
2. *Define $g(y) := \mathfrak{R}(f, r)(By + b)$ for random $B \in \mathbb{C}^{N\times r}$ and $b \in \mathbb{C}^N$.*

3. *Let* $a \in \mathbb{C}^r$ *and* $A, \Lambda \in \mathbb{C}^{r \times r}$ *be random and form* $\mathcal{E}(\Lambda, a, A, y, t) := g(y) + \Lambda \cdot T(t) \cdot L(a, A, y)$.

4. *Compute* $Y :=$ *homotopy_solve*$(\mathcal{E}_{r-1}(\Lambda, a, A, y))$.

5. *Let* $W := \{y \in Y \ : \ g(y) = 0\}$ *and* $\mathcal{N} := \{y \in Y \ : \ g(y) \neq 0\}$.

6. *For* $j := r - 1, \ldots, 1$, *do the following:*

   (a) *Append* $\widehat{W}_{d+j} := B \cdot W + b$ *to* $\widehat{W}$.

   (b) *Track solution paths for* H_{j-1} *starting from* $\mathcal{N}$. *Let* S *be the set of limit points in* $\mathbb{C}^N$.

   (c) *Let* $W := \{y \in Y \ : \ g(y) = 0\}$ *and* $\mathcal{N} := \{y \in Y \ : \ g(y) \neq 0\}$.

7. *Append* $\widehat{W}_d := \{x \ : \ x = By + b, f(x) = 0, y \in W\}$ *to* $\widehat{W}$.

2.4.2 Junk removal via a membership test

The second step in computing the numerical irreducible decomposition for a polynomial system is to remove the junk points from a witness superset to create a witness set. This is accomplished using the homotopy membership test [40, 44].

Let $V \subset \mathbb{C}^N$ be a variety of dimension d and V_{reg} be the set of smooth points in V. The homotopy membership test depends upon the fact that V and V_{reg} are both path connected. In particular, let $y \in \mathbb{C}^N$, L_0 be a codimension d generic linear space passing through y, and L_1 be a codimension d generic linear space. If $y \in V$, y is a limit point of a path defined by $V \cap (tL_1 + (1-t)L_0)$ that starts at a point in $W = V \cap L_1$. For a union of varieties of dimension d, we can apply this test to each variety to determine if y lies on at least one of the varieties. The homotopy membership test *membership* is as follows.

Algorithm 2.4.10. *membership*$(y, W; is\_member)$

Input:

- y: *a point in* $\mathbb{C}^N$.
- W: *a witness set for a pure* d*-dimensional algebraic set* X.

Output:

- is_member: *True, if* $y \in X$, *otherwise False.*

Algorithm:

1. *Let* L_W *be* d *linear equations such that* $W = X \cap V(L_W)$.
2. *Let* $A \in \mathbb{C}^{d \times N}$ *be a random matrix and define* $L_y(x) := A(x - y)$.
3. *Track solution paths defined by* $X \cap V(tL_W + (1-t)L_y)$ *starting at the points in* W *to create* W_0.
4. *If* $y \in W_0$ *then* $is\_member := True$, *otherwise* $is\_member := False$.

With *membership*, and using the observation in Section 2.4.1 that the witness superset for the top dimension is a witness set, we can formulate the junk removal algorithm *junk_removal_mem*.

Algorithm 2.4.11. *junk_removal_mem*$(\widehat{W}; W)$

Input:

- $\widehat{W}$: *a witness superset for an algebraic set* X.

Output:

- W: *a witness set for* X.

Algorithm:

1. *Let* $\widehat{W} = \{\widehat{W}_0, \ldots, \widehat{W}_d\}$.
2. *Initialize* $W_0, \ldots, W_{d-1} := \emptyset$ *and* $W_d := \widehat{W}_d$.

3. For $k := d-1, \ldots, 0$, do the following:

(a) For each $y \in \widehat{W}_k$, do the following:

i. If False = membership(y, W_j) for all $k < j \leq d$, append y to W_k.

4. Set $W := \{W_0, \ldots, W_d\}$.

2.4.3 Decomposing witness sets into irreducible components

The third step in computing the numerical irreducible decomposition for a polynomial system is to decompose the witness set for each dimension into irreducible witness sets. The *membership* algorithm presented in Section 2.4.2 relies on the path connectedness of a variety and its set of smooth points. Monodromy [41, 44] uses this fact to form a partition of the witness set into points that must lie on the same variety. A trace test [41, 44] is then used to certify the decomposition.

Let $X \subset \mathbb{C}^N$ be a pure d-dimensional algebraic set and let $L(t)$ be a one-real-dimensional closed loop of generic linear spaces of codimension d. That is, for each t, $L(t)$ is a codimension d generic linear space with $L(1) = L(0)$. Let $W(t) = X \cap L(t)$. As sets, $W(1) = W(0)$, but it could happen that a path starting at $x \in W(1)$ ends at $y \in W(0)$ with $x \neq y$. If this occurs, x and y must lie on the same variety.

Using an ordering of the points in a witness set W, *monodromy* performs a monodromy loop on each point in W and computes an ordered set W' such that the ith point of W' is on the same variety as the ith point of W.

Algorithm 2.4.12. *monodromy*$(W; W')$

Input:

• *W: a witness set for a pure d-dimensional algebraic set X.*

Output:

• *W': a witness set for X where the ith point of W and W' lie on the same variety.*

Algorithm:

1. *Let L_W be d linear equations such that $W = X \cap V(L_W)$.*
2. *Let $a \in \mathbb{C}^N$ and $A \in \mathbb{C}^{d \times N}$ be random and define $L'(x) := a + Ax$.*
3. *Construct $L(t) := L_W + t(1-t)L'$.*
4. *Track solution paths defined by $X \cap V(L(t))$ starting at the points in W to create W'.*

From a set $Y \subset W$, the trace test algorithm *trace_test* decides if Y is a union of irreducible witness sets. Moreover, if Y was constructed so that every nonempty proper subset of Y is known to not form an irreducible witness set, then *trace_test* decides if Y is an irreducible witness set. The following theorem describes how to use linear traces to perform this test.

Theorem 2.4.13. *Let Y be a nonempty subset of a witness point set W for a pure d-dimensional algebraic set X, L_d be a system of d linear equations such that $W = X \cap V(L_d)$, $v \in \mathbb{C}^N$ be a generic vector, and $\lambda : \mathbb{C}^N \to \mathbb{C}$ be a general linear function. For each $y \in Y$, define $y(t) = X \cap V(L_d + tv)$ where $y(0) = y$. Then, Y is union of irreducible witness sets if and only if $\phi_Y(t) = \sum_{y \in Y} \lambda(y(t))$ is linear in t.*

The genericity of λ and v provides that $\phi(t)$ is linear if and only if $\phi(0)$, $\phi_Y(t_1)$, and $\phi_Y(t_2)$ lie on a line for distinct nonzero $t_1, t_2 \in \mathbb{R}$, which is equivalent to

$$\frac{\phi_Y(t_1) - \phi_Y(0)}{t_1} = \frac{\phi_Y(t_2) - \phi_Y(0)}{t_2}.$$

In particular, ϕ_Y is linear if and only if

$$\sum_{y \in Y} \left(\frac{\phi_{\{y\}}(t_1) - \phi_{\{y\}}(0)}{t_1} - \frac{\phi_{\{y\}}(t_2) - \phi_{\{y\}}(0)}{t_2} \right) = 0.$$

The algorithm *compute_trace* computes, for each y, the value of the summand, called the *linear trace*, and the algorithm *trace_test* computes the summation and determines if it is (approximately) 0.

Algorithm 2.4.14. *compute_trace*$(Y; tr)$

Input:

• Y*: a subset of a witness set for a pure* d*-dimensional algebraic set* X*.*

Output:

• tr*: an array of values containing the linear trace for each point in* Y*.*

Algorithm:

1. *Let* L_W *be* d *linear equations such that* $Y \subset X \cap V(L_W)$*.*
2. *Let* $v, a \in \mathbb{C}^N$ *be generic and* $t_1, t_2 \in \mathbb{R} \backslash \{0\}$ *distinct.*
3. *For each* $y \in Y$*, do the following:*
   (a) *Track the solution path defined by* $X \cap V(L_W + tv)$ *starting with* y *at* $t = 0$ *to compute* y_1 *at* $t = t_1$ *and* y_2 *at* $t = t_2$*.*
   (b) *Append* $a \cdot ((y_1 - y)/t_1 + (y_2 - y)/t_2)$ *to* tr*.*

Algorithm 2.4.15. *trace_test*$(tr, \epsilon; is\_complete)$

Input:

• tr*: an array of values containing the linear trace for each point in a subset of*

a witness set.

- ϵ*: a positive number.*

Output:

- *is_complete: True, if the sum of the values in tr has modulus less than* ϵ*, otherwise False.*

Algorithm:

1. *Let* $t := \sum tr$.
2. *If* $|t| < \epsilon$, *is_complete* $:= True$, *otherwise is_complete* $:= False$.

The algorithm *irreducible_decomp* decomposes a witness set W for a pure-dimensional algebraic set by using a combination of *monodromy*, *compute_trace*, and *trace_test*. Initially, W is partitioned into point sets and *compute_trace* is used to compute the linear traces. Monodromy loops are performed using *monodromy* to identify points that must lie on the same variety with such points being grouped together. Monodromy loops are computed until either it keeps failing to find new connections or the number of groups remaining are small enough to use an exhaustive trace test method. Nonnegative integers M and K control this transition.

Algorithm 2.4.16. *irreducible_decomp*$(W, M, K, \epsilon; W')$

Input:

- W*: a witness set for a pure-dimensional algebraic set* X.
- M, K*: nonnegative integers that control when to stop using monodromy loops.*
- ϵ*: a positive number.*

Output:

- W'*: a list of irreducible witness sets corresponding to the varieties in* X.

Algorithm:

1. *Let* $\{y_1, \ldots, y_j\}$ *be the witness point set associated with* W.
2. *Initialize* $Y = \{Y_1, \ldots, Y_j\}$ *where* $Y_i = \{y_i\}$.
3. *Compute the array of linear traces* tr *where* $tr_i :=$*compute_trace*(Y_i).
4. *If trace_test*(tr_i, ϵ), *move* Y_i *from* Y *to* W'.
5. *Initialize* $k := 0$.
6. *While* $j > M$ *and* $k \leq K$, *do the following:*
   (a) $\{Y'_1, \ldots, Y'_j\} :=$*monodromy*$(\{Y_1, \ldots, Y_j\})$.
   (b) *Update* Y *and* tr *by merging* Y_i *and* Y_l *if* $Y'_i \cap Y_l \neq \emptyset$.
   (c) *If* $j \neq |Y|$, *do the following:*
      i. *For each* i, *if trace_test*(tr_i, ϵ), *move* Y_i *from* Y *to* W'.
      ii. *Update* $j := |Y|$ *and set* $k := 0$.
   (d) *Otherwise,* $k := k + 1$.
7. *While* $|Y| > 0$, *do the following:*
   (a) *Compute the set* $A \subset \{2, \ldots, |Y|\}$ *such that True =* *trace_test*(tr_Z, ϵ) *and* $|Z|$ *is minimized, where* $Z = Y_1 \cup (\cup_{j \in A} Y_j)$.
   (b) *Merge this combination and move it to* W'.

CHAPTER 3

REGENERATION

Regeneration is a technique based on homotopy continuation and product decomposition that solves a system of polynomials by introducing the equations one-by-one or in a group. At each stage of this process, the solution set can be extended to the next stage until the solution set for the polynomial system is obtained. The general procedure of regeneration is similar to the approach in [42] where a diagonal homotopy was used to compute the solutions at the next stage. The main disadvantage to using a diagonal homotopy is the doubling of the number of extrinsic variables for path tracking. Section 5.1 compares the diagonal homotopy approach with regeneration.

The method in [54] uses a two-stage approach to solving mixed polynomial-trigonometric systems with the final stage using a product decomposition homotopy that is solved by using multiple polyhedral homotopies. This also has some resemblance to the general method of regeneration that is presented in the following sections. These sections describe using regeneration for computing isolated solutions and a witness superset of a polynomial system, with Section 3.2 mostly following [17].

3.1 Problem statement

Regeneration will be applied to two basic problems in numerical algebraic geometry.

Problem 3.1.1 (Isolated roots). *Let $f : \mathbb{C}^n \to \mathbb{C}^n$ be a square polynomial system, Y be a proper algebraic subset of $\mathbb{C}^n$, and Z be the set of isolated points in $V(f)\backslash Y$. Given f and a membership test for Y, compute Z.*

For nonsquare systems $f : \mathbb{C}^N \to \mathbb{C}^n$, we can apply Thm. 2.3.8. The case when $n > N$ can be treated by replacing f with $\mathfrak{R}(f; N)$. If $n < N$, f has no isolated solutions.

Problem 3.1.2 (Witness superset). *Let $f : \mathbb{C}^n \to \mathbb{C}^n$ be a square polynomial system and Y be a proper algebraic subset of $\mathbb{C}^n$. Given f and a membership test for Y, compute a witness superset for $V(f)\backslash Y$.*

As above, for nonsquare systems $f : \mathbb{C}^N \to \mathbb{C}^n$ with $n > N$, we can replace f with $\mathfrak{R}(f; N)$. If $n < N$, we know that $\dim V(f) \geq N - n$. Since all varieties in $V(f)$ will be sliced by using at least a $N - n$ dimensional general linear space, we can append $N - n$ general linear equations creating a square system.

Computing a witness superset for $V(f)$ is a special case of Problem 3.1.2 with $Y = \emptyset$.

3.2 Regeneration for isolated roots

This section addresses Problem 3.1.1, in which we seek the isolated roots of a square system $f : \mathbb{C}^n \to \mathbb{C}^n$.

3.2.1 Incremental regeneration

Our strategy for computing the isolated solutions of a square polynomial system will consist of several stages of regeneration, starting with a subset of the polynomials and bringing in new ones at each subsequent stage until finally we have the isolated solutions to the full system. Each regeneration stage has two main steps: use a parameter continuation to get the start points of a product decomposition homotopy that completes the stage. The parameter continuation step regenerates a linear product form related to the new polynomials to be introduced at that stage. This regeneration step is summarized in the following lemma. It should be understood that a sequence $w_i, \ldots, w_j$ is empty if $i > j$.

Lemma 3.2.1 (Regeneration of a linear product). *Let $X_k \subset \mathbb{C}^n$ be a quasiprojective algebraic set of dimension k, $f_1, \ldots, f_m$ be polynomials on $\mathbb{C}^n$, and suppose that for $i = m+1, \ldots, \widehat{m}$, $m < \widehat{m} \leq n$, $g_i = \prod_{j=1}^{d_i} \ell_{i,j}$, where each $\ell_{i,j}$ is a linear function on $\mathbb{C}^n$. Further, let S_m be the isolated (resp., nonsingular isolated) points of*

$$V(f_1, \ldots, f_m, h_{m+1}, \ldots, h_n) \cap X_k,$$

where, for $i = m+1, \ldots, \widehat{m}$, h_i is a generic supporting linear for $\ell_{i,1}, \ldots, \ell_{i,d_i}$ and $h_{\widehat{m}+1}, \ldots, h_n$ are linear functions. Let $T_{m,\widehat{m}}$ be the isolated (resp., nonsingular isolated) points of

$$V(f_1, \ldots, f_m, g_{m+1}, \ldots, g_{\widehat{m}}, h_{\widehat{m}+1}, \ldots, h_n) \cap X_k.$$

Finally, let $\mathcal{I}_{m,\widehat{m}} \in \mathbb{N}^{\widehat{m}-m+1}$ be the index set $[1, d_{m+1}] \times \cdots \times [1, d_{\widehat{m}}]$. Then, for any particular $a = (a_{m+1}, \ldots, a_{\widehat{m}}) \in \mathcal{I}_{m,\widehat{m}}$, the start points S_m, and the homotopy

function

$$H^{\text{parm}}_{m,\widehat{m},a}(x,t) = \{f_1,\ldots,f_m,$$
$$(1-t)\ell_{m+1,a_{m+1}} + th_{m+1},\ldots,(1-t)\ell_{\widehat{m},a_{\widehat{m}}} + th_{\widehat{m}},$$
$$h_{\widehat{m}+1},\ldots,h_n\} = 0 \quad (3.2.1)$$

form a complete homotopy for $T_{m,\widehat{m},a}$, the isolated (resp., nonsingular isolated) points of

$$V(f_1,\ldots,f_m,\ell_{m+1,a_{m+1}},\ldots,\ell_{\widehat{m},a_{\widehat{m}}},h_{\widehat{m}+1},\ldots,h_n)\cap X_k.$$

Furthermore, $T_{m,\widehat{m}}$ is contained in $\cup_{a\in\mathcal{I}_{m,\widehat{m}}} T_{m,\widehat{m},a}$.

The proof follows immediately from Thm. 2.3.4, since each homotopy at Eq. 3.2.1 is a parameter homotopy in the coefficients of the linear functions $h_{m+1},\ldots,h_{\widehat{m}}$. When applying Thm. 2.3.4 to the above situation, the k dimensional space X_k appearing in it is the $\widehat{m}-m$ dimensional component of $V(f_1,\ldots,f_m,h_{\widehat{m}+1},\ldots,h_n)\cap X_k$.

The procedure implied by Lemma 3.2.1 allows us to extend a solution for $f_1,\ldots,f_m$ into one for $f_1,\ldots,f_{\widehat{m}}$, $\widehat{m} > m$. The following lemma establishes the secondary step of regeneration that accomplishes this.

Lemma 3.2.2 (Incremental product decomposition). *Adopt all the notations of Lemma 3.2.1. Further, let $\mathbb{V}_i := \{V_i, V_{i,1},\ldots,V_{i,d_i}\}$ be a linear product decomposition for f_i, $i = m+1,\ldots,\widehat{m}$, and assume that each g_i, $i = m+1,\ldots,\widehat{m}$, is a generic product member of V_i. Then, the start set $T_{m,\widehat{m}}$ with the homotopy*

function

$$H^{\text{prod}}_{m,\widehat{m}}(x,t) = \{f_1,\ldots,f_m,$$
$$(1-t)f_{m+1}+tg_{m+1},\ldots,(1-t)f_{\widehat{m}}+tg_{\widehat{m}},$$
$$h_{\widehat{m}+1},\ldots,h_n\} = 0 \quad (3.2.2)$$

is a complete homotopy for $S_{\widehat{m}}$.

This lemma follows immediately from Theorem 2.3.5. To apply the theorem, the k dimensional space X_k appearing in it is the $\widehat{m}-m$ dimensional component of $V(f_1,\ldots,f_m,h_{\widehat{m}+1},\ldots,h_n)\cap X_n$.

To apply Lemma 3.2.2, we need a linear product decomposition $V_{i,1}\otimes\cdots\otimes V_{i,d_i}$ for each f_i where $d_i=\deg f_i$, $i=m+1,\ldots,\widehat{m}$. We know that it is sufficient to choose each $V_{i,j}$ as the vector space whose elements are the support base f_i, but often some of the $V_{i,j}$ may omit some variables that appear in f_i and still suffice. For example, the polynomial $xy+1$ admits the linear product decomposition $\langle x,1\rangle\otimes\langle y,1\rangle$, whereas its support base is $\{1,x,y\}$.

3.2.2 Extrinsic vs. intrinsic

In both Eq. 3.2.1 and Eq. 3.2.2, there are linear functions $h_{\widehat{m}+1},\ldots,h_n$ that do not change during the path tracking. This provides the opportunity to use an intrinsic formulation as described in Section 2.3.8. When $\widehat{m}$ is small enough for the intrinsic formulation to be advantageous, the software package Bertini [2, 5] automatically invokes it.

3.2.3 Full regeneration

Using Lemmas 3.2.1 and 3.2.2, it is straightforward to solve Problem 3.1.1. One merely specifies any set of strictly increasing integers ending at n, say $0 = m_0 < m_1 < \cdots < m_r = n$. Then, one solves r incremental problems for $(m, \widehat{m}) = (0, m_1), (m_1, m_2), \ldots, (m_{r-1}, n)$, using the isolated (or nonsingular) solutions of one incremental problem as the start points for the next incremental problem. To be clear, we summarize the steps in the regeneration algorithm *regenerate*.

Theorem 3.2.3 (Regeneration of isolated roots). *Subject to genericity, the algorithm regenerate below solves Problem 3.1.1.*

The validity of each homotopy step in *regenerate* is established by Lemmas 3.2.1 and 3.2.2. It is possible that some of the endpoints of the homotopies defined by Eq. 3.2.1 and Eq. 3.2.2 might lie on higher dimensional sets, so these must be cast out to obtain just the set of isolated solutions needed for the subsequent homotopy. When it is needed, Chapter 4 gives a local dimension test that can differentiate between the isolated and nonisolated solutions. Without a local dimension test, we can only solve the more limited, but highly relevant, case of finding just the nonsingular solutions at each stage. The nonsingularity condition is easily checked by computing the rank of the Jacobian matrix of partial derivatives for each point. "Subject to genericity" acknowledges that the algorithm must make generic choices of coefficients in the linear functions $h_1, \ldots, h_n$, the linear functions that form the linear products $g_1, \ldots, g_n$, and generic choices required in any homotopy membership test.

Algorithm 3.2.4. *regenerate*$(f, Y, \sigma; S)$

Input:

- *f: a system of n polynomials in $\mathbb{C}[x_1, \ldots, x_n]$.*
- *Y: a proper subset of $\mathbb{C}^n$ in a form suitable for membership test.*
- *σ: either True or False.*

Output:

- *S: when σ is True (resp. when σ is False), the set of all isolated (resp., nonsingular isolated) points in $V(f) \cap X$, where $X = \mathbb{C}^n \setminus Y$.*

Algorithm:

1. *Reorder the polynomials $f_1, \ldots, f_n$ in any advantageous order (see Section 3.2.4).*
2. *Pick a set of $r+1$ strictly increasing integers starting at 0 and ending at N, say $0 = m_0 < m_1 < \cdots < m_r = n$.*
3. *Specify a linear product decomposition $V_{i,1} \otimes \cdots \otimes V_{i,d_i}$ for each f_i with $d_i = \deg f_i$, $i = 1, \ldots, n$. One alternative that always suffices is each $V_{i,j}$ is generated by the support base of f_i.*
4. *Choose a generic product member $\ell_{i,j}$ in each $V_{i,j}$, $i = 1, \ldots, N$, $j = 1, \ldots, d_i$, i.e., $\ell_{i,j}$ is a linear function with generic coefficients. Let $g_i = \prod_{j=1}^{d_i} \ell_{i,j}$.*
5. *For $i = 1, \ldots, n$, choose a generic linear h_i that supports all $\ell_{i,j}$, $j = 1, \ldots, d_i$.*
6. *For $i = 1, \ldots, r$, let $(m, \widehat{m}) = (m_{i-1}, m_i)$, let*

$$\begin{aligned} G_{m,\widehat{m}} &= \{f_1, \ldots, f_m, g_{m+1}, \ldots, g_{\widehat{m}}, h_{\widehat{m}+1}, \ldots, h_n\}, \\ F_{\widehat{m}} &= \{f_1, \ldots, f_m, f_{m+1}, \ldots, f_{\widehat{m}}, h_{\widehat{m}+1}, \ldots, h_n\}, \end{aligned}$$

and do the following:

(a) *Solve for $T_{m,\widehat{m}}$, a superset of the set of isolated (resp., nonsingular isolated) points of $V(G_{m,\widehat{m}}) \cap X$. There are two cases, as follows.*

Case $m = 0$**.** *Use numerical linear algebra to solve the initial system $\{g_1, \ldots, g_{\widehat{m}}, h_{\widehat{m}+1}, \ldots, h_n\}$. Since each g_i is a product of d_i linear factors, there are at most $D_{1,m_1} = \prod_{i=1}^{m_1} d_i$ solutions, all of which can be found by linear algebra. Since the linear factors may be sparse, there may be fewer than D_{1,m_1} solutions. The solution set is called T_{0,m_1}.*

Otherwise. *Use the homotopies $H^{\text{parm}}_{m,\widehat{m},a}$ from Eq. 3.2.1 with start set S_m.*

(b) *Use a membership test to expunge any points of $T_{m,\widehat{m}}$ that are in Y.*

(c) *If σ is True, use a local dimension test to expunge any singular points that are not isolated.*

(d) *If σ is False, eliminate any singular points from $T_{m,\widehat{m}}$.*

(e) *Solve for $S_{\widehat{m}}$, a superset of the set of all isolated (resp., nonsingular isolated) points of $V(\{f_1, \ldots, f_{\widehat{m}}, h_{\widehat{m}+1}, \ldots, h_n\}) \cap X$ using the product homotopy $H^{\text{prod}}_{m,\widehat{m}}$ from Eq. 3.2.2 with start solutions $T_{m,\widehat{m}}$.*

(f) *Use a membership test to expunge any points of $S_{\widehat{m}}$ that are in Y.*

(g) *If σ is True, use a local dimension test to expunge any singular points of $S_{\widehat{m}}$ that are not isolated.*

(h) *If σ is False, eliminate any singular points from $S_{\widehat{m}}$.*

7. *Set $S := S_n$.*

3.2.4 Ordering of the functions

At Step 1, one may choose to reorder the polynomials. In general, this changes the number of paths that need to be tracked. One way to attempt to minimize the number of paths is to minimize the maximum number of possible paths to track. Suppose we are working equation by equation (that is, $r = n$) and that the linear product decompositions have $d_i = \deg f_i$ factors. Then, the maximum number of paths to track is $p = d_1 + d_1d_2 + \ldots + d_1d_2 \cdots d_n$. By reordering the functions so that $d_1 \leq d_2 \leq \ldots \leq d_n$, the maximum number of paths p is minimized.

It is common that some endpoints at intermediate stages are cast out for lying on positive dimensional components or on the excluded set Y. In fact, it is to our advantage to arrange for this to happen as early and as often as possible. This goal may sometimes conflict with an ordering having monotonically increasing degrees. It is generally impossible to know ahead of time how the number of paths depends on the ordering, but one simple observation seems to help. When the functions are sparse, often only a subset of the variables appear in some equations. A good strategy is to order the functions so that cumulative number of distinct variables that have been introduced at any stage is minimal.

When these two strategies are compatible, as in Section 5.1, a good ordering of the polynomials is easily decided. (There may be more than one equally good ordering.) Unfortunately, we do not yet have good rules for picking an ordering when the strategies conflict. We suggest first ordering by degree, and if some polynomials have the same degree, order them to minimize the rate of accumulation of new variables. When neither of these criteria decides the ordering of some subgroup of the polynomials, our early experience indicates that the ordering within such a group has a minimal effect.

3.2.5 Equation grouping

At Step 2, one may choose how many polynomials to introduce at each stage. One far extreme is to choose $r = 1$, in which case we introduce all of the polynomials at once, resulting in only one stage of homotopy that is effectively a traditional linear product homotopy on the whole system. At the other extreme, one may choose $r = n$, which means $m_0, \ldots, m_r = 0, 1, 2, \ldots, n-1, n$. We call this "solving equation by equation," because only one new polynomial from f is introduced at each pass through the main loop. We often prefer to take this extreme, but sometimes equations appear in related subgroups that we elect to introduce group by group. The example in Section 5.2 has this character: the polynomials arise naturally as subsystems, each consisting of 2 polynomials. For that problem, introducing the equations two at a time results in fewer paths to track than an equation-by-equation approach.

Another consideration comes into play in an implementation on multiple parallel processors. The number of paths to track usually increases at each stage (often dramatically so), and if there are many processors available, it could happen that some of them sit idle in the early stages. To put this resource to best use, it may also be advantageous to introduce groups of equations in the early stages to make enough paths to keep all processors busy, then drop back to working equation by equation as the solution set increases in size.

3.2.6 Choosing linear products

The freedom to choose a linear product decomposition at Step 3 can have a noticeable effect. One may take advantage of multilinearity and other forms of sparseness here.

3.3 Regeneration for witness supersets

Our strategy for computing a witness superset is to use randomization along with regeneration using generic linear equations and going equation by equation.

Let $f : \mathbb{C}^n \to \mathbb{C}^n$ be a square polynomial system with $\deg f_1 \geq \deg f_2 \geq \cdots \geq \deg f_n$. Let $a_{i,j} \in \mathbb{C}$, $1 \leq i < j \leq n$, be random. Define

$$A = \begin{bmatrix} 1 & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ & 1 & a_{2,3} & \cdots & a_{2,n} \\ & & \ddots & & \vdots \\ & & & 1 & a_{n-1,n} \\ & & & & 1 \end{bmatrix} \tag{3.3.1}$$

and let

$$\widehat{f} = A \cdot f. \tag{3.3.2}$$

The remainder of this section uses regeneration on $\widehat{f}$ to compute a witness superset for f.

3.3.1 Regenerative cascade

Let $h_2, \ldots, h_n$ be generic linear functions on $\mathbb{C}^n$ and, for $k = 1, \ldots, n$, define

$$H_k = \begin{bmatrix} \widehat{f}_1 \\ \vdots \\ \widehat{f}_k \\ h_{k+1} \\ \vdots \\ h_n \end{bmatrix}.$$

The following theorem describes the solutions found after the kth stage of regeneration using $\widehat{f}$.

Theorem 3.3.1. *Let $k \in \{1, \ldots, n\}$ and S_k be a witness superset for the isolated solutions of H_k. Let $\widehat{W}_{n-k} = \{x \in S_k \ : \ f(x) = 0\}$ and $\widehat{S}_k = S_k \setminus \widehat{W}_{n-k}$. Then, $\widehat{W}_{n-k}$ is a witness superset for the dimension $n-k$ varieties of $V(f)$ and each point in $\widehat{S}_k$ is a nonsingular isolated solution of H_k.*

Proof. Let V_k be the union of the dimension $n-k$ varieties in $V(f)$ and $L_k = V(h_{k+1}, \ldots, h_n)$. Then, the points in $V_k \cap L_k$ are isolated solutions of H_k. That is, $V_k \cap L_k \subset \widehat{W}_{n-k}$. Since $\widehat{W}_{n-k}$ is a finite set with $\widehat{W}_{n-k} \subset V(f) \cap L_k$, $\widehat{W}_{n-k}$ is a witness superset for the dimension $n-k$ varieties of $V(f)$. Bertini's Theorem [44] provides that each point in $\widehat{S}_k$ is a nonsingular isolated solution of H_k. □

The set $\widehat{S}_k$ is called the set of *stage k nonsolutions*. For $k < n$, the theory of regeneration presented in Section 3.2 computes a superset of the isolated solutions of H_{k+1} using a set of isolated solutions of H_k that can lead to isolated solutions of H_{k+1}. Since the points in $\widehat{W}_{n-k}$ lie on a component of $V(f)$ of dimension at least $n-k$, these points cannot lead to isolated solutions of H_{k+1}. In particular, the nonsolutions at stage k are the only ones that can lead to isolated solutions of H_{k+1}. This provides the justification for following theorem for the regenerative cascade algorithm *regen_cascade*.

Theorem 3.3.2 (Regeneration for witness supersets). *Subject to genericity and using the simplified homotopies*

$$H_{i,k}^{\text{parm}}(x,t) = \{\widehat{f}_1, \ldots, \widehat{f}_i, (1-t)\ell_{i+1,k} + th_{i+1}, h_{i+2}, \ldots, h_n\} = 0 \qquad (3.3.3)$$

and

$$H_i^{\text{prod}}(x,t) = \{\widehat{f}_1, \ldots, \widehat{f}_i, (1-t)\widehat{f}_{i+1} + tg_{i+1}, h_{i+2}, \ldots, h_n\} = 0, \tag{3.3.4}$$

the algorithm regen_cascade below solves Problem 3.1.2.

Algorithm 3.3.3. *regen_cascade*$(f, Y; \widehat{W})$
Input:

- f*: a system of* n *polynomials in* $\mathbb{C}[x_1, \ldots, x_n]$ *of rank* $r > 0$.
- Y*: a proper subset of* $\mathbb{C}^n$ *in a form suitable for membership test.*

Output:

- $\widehat{W}$*: a witness superset for* $V(f) \setminus Y$.

Algorithm:

1. *Define* $d_i = \deg f_i$ *and reorder the polynomials so that* $d_1 \geq \cdots \geq d_n$.
2. *Let* $a_{i,j} \in \mathbb{C}$, $1 \leq i < j \leq n$ *be random. Define* A *as in Eq. 3.3.1 and* $\widehat{f}$ *as in Eq. 3.3.2.*
3. *For* $i = 1, \ldots, n$ *and* $j = 1, \ldots, d_i$, *let* $\ell_{i,j}$ *be a generic linear function on* $\mathbb{C}^n$, $g = \prod_{j=1}^{d_i} \ell_{i,j}$, *and* $h_i = \ell_{i,1}$.
4. *For* $i = 0, \ldots, r-1$, *let*

$$\begin{aligned} G_i &= \{\widehat{f}_1, \ldots, \widehat{f}_i, g_{i+1}, h_{i+2}, \ldots, h_n\}, \\ F_i &= \{\widehat{f}_1, \ldots, \widehat{f}_i, \widehat{f}_{i+1}, h_{i+2}, \ldots, h_n\}, \end{aligned}$$

   and do the following:

   (a) *Solve for* T_i, *the set of isolated points of* $V(G_i) \cap X$ *that are not solutions of* f. *There are two cases, as follows.*

Case $i = 0$**.** *Use numerical linear algebra to solve the initial system* $\{g_1, h_2, \ldots, h_n\}$*. Since each* $\ell_{i,j}$ *is a generic linear, there are* d_1 *solutions that can be found by linear algebra.*

Otherwise. *Use the homotopies* $H_{i,k}^{\text{parm}}$ *for* $k = 2, \ldots, d_{i+1}$ *from Eq. 3.3.3 with start set* $\widehat{S}_i$*.*

(b) Use a membership test to expunge any points of T_i *that are in* Y*.*

(c) Solve for S_{i+1}*, a superset of the set of all isolated points of* $V(F_i) \cap X$ *using the product homotopy* H_i^{prod} *from Eq. 3.3.4 with start solutions* T_i*.*

(d) Use a membership test to expunge any points of S_{i+1} *that are in* Y*.*

(e) Let $\widehat{W}_{n-i-1} = \{x \in S_{i+1} \ : \ f(x) = 0\}$ *and* $\widehat{S}_{i+1} = S_{i+1} \setminus \widehat{W}_{n-i-1}$*.*

5. Set $\widehat{W} := \{\widehat{W}_{n-r}, \ldots, \widehat{W}_{n-1}\}$*.*

As with Eq. 3.2.1 and Eq. 3.2.2, the homotopies described in Eq. 3.3.3 and Eq. 3.3.4 can utilize an intrinsic formulation, as described in Section 2.3.8. When i is small enough for the intrinsic formulation to be advantageous, the software package Bertini [2, 5] automatically invokes it.

3.3.2 Simplification of the regenerative cascade

It often happens that $\widehat{W}_i = \emptyset$ for all large i, i.e., there are no solution components of large dimension. In this situation, we can replace the path tracking in Step 4a with linear algebra. Upon this reduction, these steps of the regenerative cascade are identical to dimension-by-dimension slicing using a linear product start system.

Algorithm 3.3.4. *regen_cascade_simplified*$(f, Y; \widehat{W})$

Input:

- *f: a system of n polynomials in $\mathbb{C}[x_1, \ldots, x_n]$ of rank $r > 0$.*
- *Y: a proper subset of $\mathbb{C}^n$ in a form suitable for membership test.*

Output:

- *$\widehat{W}$: a witness superset for $V(f) \cap X$, where $X = \mathbb{C}^n \setminus Y$.*

Algorithm:

1. *Define $d_i = \deg f_i$ and reorder the polynomials so that $d_1 \geq \cdots \geq d_n$.*
2. *Let $a_{i,j} \in \mathbb{C}$, $1 \leq i < j \leq n$ be random. Define A as in Eq. 3.3.1 and $\widehat{f}$ as in Eq. 3.3.2.*
3. *For $i = 1, \ldots, n$ and $j = 1, \ldots, d_i$, let $\ell_{i,j}$ be a generic linear function on $\mathbb{C}^n$, $g = \prod_{j=1}^{d_i} \ell_{i,j}$, and $h_i = \ell_{i,1}$.*
4. *For $i = 0, \ldots, r-1$, do the following:*
   (a) *Compute T_i, as follows:*

   Case $i = 0$ or $\widehat{W}_j = \emptyset$ for $j \geq n - i$. *Let*

$$\begin{aligned} G_i &= \{g_1, \ldots, g_i, g_{i+1}, h_{i+2}, \ldots, h_n\}, \\ F_i &= \{\widehat{f}_1, \ldots, \widehat{f}_i, \widehat{f}_{i+1}, h_{i+2}, \ldots, h_n\}, \end{aligned}$$

   and use numerical linear algebra to solve to compute $T_i = V(G_i)$. Since each $\ell_{i,j}$ is a generic linear, there are $d_1 \cdots d_{i+1}$ solutions of G_i.

Otherwise. *Let*

$$\begin{aligned} G_i &= \{\widehat{f}_1, \ldots, \widehat{f}_i, g_{i+1}, h_{i+2}, \ldots, h_n\}, \\ F_i &= \{\widehat{f}_1, \ldots, \widehat{f}_i, \widehat{f}_{i+1}, h_{i+2}, \ldots, h_n\}, \end{aligned}$$

and compute T_i, the set of isolated points of $V(G_i) \cap X$ that are not solutions of f, by using the homotopies $H_{i,k}^{\text{parm}}$ for $k = 2, \ldots, d_{i+1}$ from Eq. 3.3.3 with start set $\widehat{S}_i$.

(b) Use a membership test to expunge any points of T_i that are in Y.

(c) Solve for S_{i+1}, a superset of the set of all isolated points of $V(F_i) \cap X$ using the linear homotopy $H(x,t) = F_i(x)(1-t) + tG_i(x)$ with start solutions T_i.

(d) Use a membership test to expunge any points of S_{i+1} that are in Y.

(e) Let $\widehat{W}_{n-i-1} = \{x \in S_{i+1} \ : \ f(x) = 0\}$ and $\widehat{S}_{i+1} = S_{i+1} \setminus \widehat{W}_{n-i-1}$.

5. Set $\widehat{W} := \{\widehat{W}_{n-r}, \ldots, \widehat{W}_{n-1}\}$.

3.3.3 Advantages of the regenerative cascade

The regenerative cascade algorithm *regen_cascade* has many advantages over the cascade algorithm *cascade* and the dimension-by-dimension slicing algorithm *dim_slicing*. This section describes the theoretical advantages with the computational evidence presented in Chapter 5.

Let $f : \mathbb{C}^n \to \mathbb{C}^n$ be a square polynomial system of rank n with $d_1 \geq \cdots \geq d_n$, where $d_i = \deg f_i$.

The main advantage of the cascade algorithm is that it creates a cascade of homotopies which utilize all of the information from previous stages on the current

stage. This creates a witness superset which generally has many fewer junk points than the dimension-by-dimension slicing algorithm. The two main disadvantages of this algorithm is that to start the cascade algorithm, the total degree number of paths, namely $d_1 \cdots d_n$, need to be tracked, and each stage of the cascade algorithm tracks using n variables. For large systems, these disadvantages make the cascade algorithm prohibitively expensive.

The main advantage of the dimension-by-dimension slicing algorithm is the ability to use intrinsic slicing to reduce the number of tracking variables. The main disadvantage of this algorithm is that each dimension is handled independently, which means that valuable information regarding all larger dimensional components is not utilized. This generally leads to a witness superset that contains more junk points than the cascade algorithm.

The regenerative cascade algorithm combines the advantages and overcomes the disadvantages described above for the cascade and dimension-by-dimension slicing algorithms. First, the regenerative cascade algorithm can utilize intrinsic slicing to reduce the number of tracking variables. Additionally, this algorithm creates a sequence of homotopies that utilize all of the information from previous stages on the current stage by only regenerating the nonsolutions at each stage. This creates a witness superset that generally has many fewer junk points than a witness superset created by dimension-by-dimension slicing, and computational evidence, such as will be shown later in Table 5.6, suggests that the number of junk points is the same as that created by the cascade algorithm. This is summarized in the following conjecture.

Conjecture 3.3.5. *The algorithms cascade and regen_cascade produce witness supersets that have the same number of junk points for each dimension.*

CHAPTER 4

LOCAL DIMENSION TEST

For a polynomial system f and a solution $\hat{x} \in V(f)$, the local dimension test computes the maximum dimension of the varieties $V \subset V(f)$ with $\hat{x} \in V$. This algorithm is valuable in many circumstances, including

1. determine whether a given solution is isolated (and computing the multiplicity if it is);
2. computing the local dimension for a given solution;
3. finding all varieties that contain a given solution;
4. filtering junk points from witness supersets;
5. computing the varieties of $V(f)$ of a prescribed dimension.

Computational evidence presented in Chapter 5 indicates that the efficiency of this method will have a significant impact on the structure of many of the algorithms in numerical algebraic geometry, most importantly the numerical irreducible decomposition. The following sections mostly follow [1].

4.1 Introduction

The following definition describes the local dimension at a solution.

Definition 4.1.1 (Local dimension). *Let $f : \mathbb{C}^N \to \mathbb{C}^n$ be a polynomial system and $\hat{x} \in V(f)$. The* **local dimension** *of $\hat{x}$ with respect to f, denoted $\dim_{\hat{x}}(V(f))$, is*

$$\dim_{\hat{x}}(V(f)) = \max\{\dim(V) \ : \ \hat{x} \in V \subset V(f) \text{ where } V \text{ is a variety}\}.$$

The algorithms presented below to compute the local dimension at a solution $\hat{x}$ with respect to a polynomial system f use the theory of Macaulay [29] and, more specifically, the numerical approach of Dayton and Zeng [12]. The basic idea in calculus terms is that the dimensions T_k of the space of Taylor series expansions of degree at most k of algebraic functions on $V(f)$ at the point $\hat{x}$ eventually grow like $O(k^{\dim_{\hat{x}}(V(f))})$. If $\dim_{\hat{x}}(V(f)) = 0$, the dimensions T_k steadily grow until they reach the multiplicity $\mu_{\hat{x}}$ of the point $\hat{x}$ and are constant from that point on. If $\dim_{\hat{x}}(V(f)) > 0$, these dimensions steadily grow without bound. If $\nu_{\hat{x}}$ is an upper bound on the multiplicity $\mu_{\hat{x}}$, we have a simple test to check whether $\hat{x}$ is isolated:

1. Compute the dimensions T_k until the minimum $k = \widehat{k}$, where

$$\widehat{k} := \min\left\{k \geq 1 \ : \ T_k = T_{k-1} \text{ or } T_k > \nu_{\hat{x}}\right\}.$$

2. Then, $\hat{x}$ is isolated if and only if $T_{\widehat{k}} \leq \nu_{\hat{x}}$.

One way to compute an upper bound on $\mu_{\hat{x}}$ is to count the number of paths which converge to $\hat{x}$ for standard homotopies.

If $V(f) \subset \mathbb{C}^N$ is k-dimensional at a point $\hat{x}$, then, for $0 \leq \ell \leq k$, a general linear space L through $\hat{x}$ of dimension equal to $N - k + \ell$ will meet $V(f)$ in an algebraic subset of dimension ℓ at $\hat{x}$. Using this fact, we turn the above algorithm for whether a point is isolated into an algorithm for computing the maximum dimension of the varieties of $V(f)$ containing $\hat{x}$.

Though the local dimension test in this thesis is the first rigorous numerical local dimension algorithm that applies to arbitrary polynomial systems, it is not the first numerical method proposed to compute local dimensions. Using the facts about slicing V with linear spaces L:

- If $V(f) \subset \mathbb{C}^N$ is k-dimensional at a point $\hat{x}$, then a general linear space L through that point of dimension less than or equal to $N-k$ will meet $V(f)$ in no other points in a neighborhood of $\hat{x}$; and
- given a general linear space L of dimension greater than $N-k$, $L \cap V(f)$ will contain points in a neighborhood of $\hat{x}$,

Sommese and Wampler [43, § 3.3] showed that the local dimension $\dim_{\hat{x}}(V(f))$ could be determined by choosing an appropriate family L_t of linear spaces with $L_0 = L$ and then deciding whether the point $\hat{x}$ deforms in $V \cap L_t$. They did not present any numerical method to make this decision. In [22], Kuo and Li present an interesting heuristical method to make this decision. The method works well for many examples, but it does not seem possible to turn it into a rigorous local-dimension algorithm for a point on the solution set of polynomial systems. For instance, since the method is based on the presentation of the system and not on intrinsic properties of the solution set, it is not likely that any result covering general systems can be proved. Indeed, as the simple system in Section 5.3 (consisting of two cubic equations in two variables) shows: solution sets with multiple branches going through a point may lead the method to give false answers.

4.2 Theory behind the algorithms

For a polynomial system $f : \mathbb{C}^N \to \mathbb{C}^n$ and isolated solution $\hat{x} \in V(f)$ of multiplicity $\mu_{\hat{x}}$, there are known methods for computing its multiplicity [6, 12, 55].

In particular, these methods compute a sequence $\{\mu_k\}_{k=0}^{\infty}$ such that $\mu_0 = 1$ and there exists $d \geq 0$ such that $\mu_k < \mu_{k+1}$ for $0 \leq k < d$ and $\mu_k = \mu_{\hat{x}}$ for $k \geq d$. The rest of this section focuses on the theory of Macaulay [29], which is used in the methods of [12, 55], to describe the behavior of the sequence μ_k at solutions which may not be isolated.

Following the notation of [12], for $j \in (\mathbb{Z}_{\geq 0})^N$, define $j! = j_1! \cdots j_N!$, $|j| = j_1 + \cdots + j_N$ and

$$\partial_j = \frac{1}{j!} \frac{\partial^{|j|}}{\partial x_1^{j_1} \cdots \partial x_N^{j_N}}.$$

Define the differential functional $\partial_j[\hat{x}] : \mathbb{C}[x_1, \ldots, x_N] \to \mathbb{C}$ at $\hat{x} \in \mathbb{C}^N$ as

$$\partial_j[\hat{x}](p) = (\partial_j p)(\hat{x}).$$

The set of differential functionals at $\hat{x}$ that vanish on the ideal generated by f is called the *dual space* of f at $\hat{x}$. The dual space, denoted $\mathcal{D}_{\hat{x}}(f)$, which can be written

$$\mathcal{D}_{\hat{x}}(f) = \Big\{ \partial = \sum_{j \in (\mathbb{Z}_{\geq 0})^N} d_j \partial_j[\hat{x}] \ : \ d_j \in \mathbb{C}, \partial\,(p f_i) = 0, \\ \forall p \in \mathbb{C}[x_1, \ldots, x_N], i = 1, \ldots, n \Big\}, \quad (4.2.1)$$

is a $\mathbb{C}$-vector space. The following theorem provides the relationship between the dimension of the dual space and the multiplicity.

Theorem 4.2.1. *Let f be a polynomial system and $\hat{x} \in V(f)$. Then, $\hat{x}$ is an isolated solution of multiplicity m if and only if $\dim_{\mathbb{C}}(\mathcal{D}_{\hat{x}}(f)) = m$. In particular, $\hat{x}$ is isolated if and only if $\dim_{\mathbb{C}}(\mathcal{D}_{\hat{x}}(f)) < \infty$.*

Stetter and Thallinger [45, 47] provided an equivalent definition of the dual

space as follows. For $\sigma \in \{1, \ldots, N\}$, let e_σ be the σth column of the $N \times N$ identity matrix and define the linear anti-differentiation operator Φ_σ as

$$\Phi_\sigma(\partial_j[\hat{x}]) = \begin{cases} \partial_{j-e_\sigma}[\hat{x}] & \text{if } j_\sigma > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Then, $\partial \in \mathcal{D}_{\hat{x}}(f)$ if and only if for each $i = 1, \ldots, n$, $\partial(f_i) = 0$ and, for each $\sigma = 1, \ldots, N$, $\Phi_\sigma(\partial) \in \mathcal{D}_{\hat{x}}(f)$.

For each $k \geq 0$, define

$$\mathcal{D}^k_{\hat{x}}(f) = \left\{ \partial = \sum_{|j| \leq k} d_j \partial_j[\hat{x}] \ : \ \partial \in \mathcal{D}_{\hat{x}}(f) \right\}.$$

Clearly, $\mathcal{D}^k_{\hat{x}}(f) \subset D^{k+1}_{\hat{x}}(f) \subset D_{\hat{x}}(f)$. The Stetter and Thallinger formulation yields the following lemma.

Lemma 4.2.2. *Let $k \geq 2$. If $\partial \in \mathcal{D}^k_{\hat{x}}(f) \backslash \mathcal{D}^{k-1}_{\hat{x}}(f)$, there is a $\sigma \in \{1, \ldots, N\}$ such that $\Phi_\sigma(\partial) \in \mathcal{D}^{k-1}_{\hat{x}}(f) \backslash \mathcal{D}^{k-2}_{\hat{x}}(f)$.*

Proof. Write $\partial = \sum_{|j| \leq k} d_j \partial_j[\hat{x}]$. By assumption, there is $\widehat{j}$ with $|\widehat{j}| = k > 0$ and $d_{\widehat{j}} \neq 0$. For σ such that $\widehat{j}_\sigma > 0$, we have $\Phi_\sigma(\partial) \in \mathcal{D}^{k-1}_{\hat{x}}(f) \backslash \mathcal{D}^{k-2}_{\hat{x}}(f)$. □

The following theorem summarizes the properties of the sequence $\{\mu_k\}_{k=0}^{\infty}$ defined by

$$\mu_k = \dim_{\mathbb{C}} \left(\mathcal{D}^k_{\hat{x}}(f) \right). \tag{4.2.2}$$

Theorem 4.2.3. *Let f be a polynomial system, $\hat{x} \in V(f)$, and μ_k as in Eq. 4.2.2. Then, $\mu_0 = 1$ and, for each $k \geq 0$, either*

1. *$\mu_k < \mu_{k+1}$, or*

2. $\mu_k = \mu_{k+i}$*, for all* $i \geq 1$*, and* $\hat{x}$ *is isolated of multiplicity* μ_k*.*

Proof. Trivially, for $\mathbf{0} = (0, \ldots, 0)$, $\partial_{\mathbf{0}}[\hat{x}](pf_i) = p(\hat{x})f_i(\hat{x}) = 0$ for all $p \in \mathbb{C}[x_1, \ldots, x_N]$ since $\hat{x} \in V(f)$. That is, $\mathcal{D}^0_{\hat{x}}(f) = \{d\, \partial_{\mathbf{0}} \ : \ d \in \mathbb{C}\}$ is a $\mathbb{C}$-vector space of dimension 1. Now, for $k \geq 0$, we have $\mu_k \leq \mu_{k+1}$ by definition. Assume that $\mu_k \not< \mu_{k+1}$, i.e., $\mu_k = \mu_{k+1}$. In particular, $\mathcal{D}^{k+1}_{\hat{x}}(f) = \mathcal{D}^k_{\hat{x}}(f)$. If $\mu_{k+2} > \mu_{k+1}$, then there is $\partial \in \mathcal{D}^{k+2}_{\hat{x}}(f) \backslash \mathcal{D}^{k+1}_{\hat{x}}(f)$. By Lemma 4.2.2, there exists σ such $\Phi_\sigma(\partial) \in \mathcal{D}^{k+1}_{\hat{x}}(f) \backslash \mathcal{D}^k_{\hat{x}}(f) = \emptyset$, yielding a contradiction. That is, $\mu_{k+2} = \mu_{k+1} = \mu_k$ and, by induction, $\mu_{k+i} = \mu_k$ for all $i \geq 1$. Theorem 2 of [12] yields that $\hat{x}$ is isolated of multiplicity μ_k. □

4.3 Algorithms

With the theoretical foundation presented in Section 4.2, we present the following five algorithms:

1. determine whether a given solution is isolated (and computing the multiplicity if it is);
2. compute the local dimension for a given solution;
3. find all varieties that contain a given solution;
4. filter junk points efficiently;
5. compute the varieties of $V(f)$ of a prescribed dimension.

These algorithms depend upon two operations, namely *multiplicity* and *find_bound*. The operation *multiplicity*$(f, \hat{x}, k)$ returns μ_k defined by Eq. 4.2.2, in which the methods of [12, 55] can be used. The operation *find_bound*$(f, \hat{x})$ returns a bound on the multiplicity of $\hat{x}$ as a solution of f. An upper bound can be determined by

using homotopy continuation and simply counting the number of solution paths converging to $\hat{x}$. This upper bound along with Thm. 4.2.3 provides the stopping criterion if $\hat{x}$ is not isolated.

The first algorithm, called *is_isolated*, determines if $\hat{x}$ is isolated. It computes μ_k until either $\mu_k = \mu_{k+1}$ (isolated of multiplicity μ_k) or μ_k is larger than the bound provided by *find_bound* (not isolated).

Algorithm 4.3.1. *is_isolated*$(f, \hat{x}, B; is\_isolated_{\hat{x}}, \mu_{\hat{x}})$

Input:

- f*: a system of* n *polynomials in* $\mathbb{C}[x_1, \ldots, x_N]$.
- $\hat{x}$*: a point on* $V(f) \subset \mathbb{C}^N$.
- B*: an upper bound on the multiplicity of* $\hat{x}$ *if it is isolated.*

Output:

- $is\_isolated_{\hat{x}}$*: True, if* $\hat{x}$ *is isolated, otherwise False.*
- $\mu_{\hat{x}}$*: the multiplicity if* $\hat{x}$ *is isolated.*

Algorithm:

1. *Initialize* $k := 1$, $\mu_0 := 1$, *and* $\mu_1 := multiplicity(f, \hat{x}, 1)$.
2. *While* $\mu_{k-1} < \mu_k \leq B$, *do the following:*
   (a) *Increment* $k := k + 1$.
   (b) $\mu_k := multiplicity(f, \hat{x}, k)$.
3. *If* $\mu_k \leq B$, *then* $is\_isolated_{\hat{x}} :=$ *True and* $\mu_{\hat{x}} := \mu_k$, *otherwise* $is\_isolated_{\hat{x}} :=$ *False.*

The second algorithm, called *local_dimension*, uses *is_isolated* along with linear slicing to determine the local dimension.

Algorithm 4.3.2. *local_dimension*$(f, \hat{x}; \dim_{\hat{x}})$

Input:

- f*: a system of* n *polynomials in* $\mathbb{C}[x_1, \ldots, x_N]$.
- $\hat{x}$*: a point on* $V(f) \subset \mathbb{C}^N$.

Output:

- $\dim_{\hat{x}}$*: the local dimension at* $\hat{x}$.

Algorithm:

1. *Choose* $L_1, \ldots, L_N$ *random linear forms through* $\hat{x}$ *on* $\mathbb{C}^N$*, and set* $k := -1$.
2. *Initialize* $is\_isolated_k :=$ *False.*
3. *While* $is\_isolated_k =$ *False, do the following:*
   (a) *Increment* $k := k + 1$.
   (b) $B_k :=$ *find_bound*$(\{f, L_1, \ldots, L_k\}, \hat{x})$.
   (c) $(is\_isolated_k, \mu_k) :=$ *is_isolated*$(\{f, L_1, \ldots, L_k\}, \hat{x}, B_k)$.
4. *Set* $\dim_{\hat{x}} := k$.

The third algorithm, called *irreducible_components*, uses an irreducible decomposition of $V(f)$ and *membership*, as described in Sections 2.2 and 2.4.2, respectively, to decide which varieties the given solution lies on.

Algorithm 4.3.3. *irreducible_components*$(f, W, \hat{x}; J)$

Input:

- f*: a system of* n *polynomials in* $\mathbb{C}[x_1, \ldots, x_N]$.
- W*: an irreducible decomposition of* f.
- $\hat{x}$*: a point on* $V(f) \subset \mathbb{C}^N$.

Output:

• *J: a set of pairs of numbers, each representing the dimension and component number of an irreducible component of which $\hat{x}$ is a member.*

Algorithm:

1. *Initialize* $J = \{\}$.
2. *Compute* $\dim_{\hat{x}} :=$ *local_dimension*$(f, \hat{x})$.
3. *For* $j := 0, \ldots, \dim_{\hat{x}}$, *do the following:*
   (a) *Set* $m :=$ *number of irreducible components of dimension* j *in* W_j.
   (b) *For* $k := 1, \ldots, m$, *if membership*$(\hat{x}, W_{jk})$, $J := J \cup \{(j, k)\}$.

The fourth algorithm, called *junk_removal_ldt*, uses *is_isolated* to remove the junk points from a witness superset. The witness superset algorithms *cascade*, *dim_slicing*, and *regen_cascade* provide an upper bound on the multiplicity, which is the number of paths that limit to the solution.

Algorithm 4.3.4. *junk_removal_ldt*$(\widehat{W}; W)$

Input:

• $\widehat{W}$*: a witness superset for an algebraic set* X.

Output:

• W*: a witness set for* X.

Algorithm:

1. *Let* f *be the polynomial system associated with* $\widehat{W}$ *and let* $\widehat{W} = \{\widehat{W}_0, \ldots, \widehat{W}_d\}$.
2. *Initialize* $W_0, \ldots, W_{d-1} := \emptyset$ *and* $W_d := \widehat{W}_d$.
3. *For* $k := d-1, \ldots, 0$, *do the following:*
   (a) *Let* L_k *be the system of linear functions associated with* $\widehat{W}_k$.

(b) For each $y \in \widehat{W}_k$, do the following:

i. Let B be the bound on the multiplicity of y, if y is an isolated solution of $\{f, L_k\}$.

ii. If is_isolated$(\{f, L_k\}, y, B)$, append y to W_k.

4. Set $W := \{W_0, \ldots, W_d\}$.

The final algorithm, called *decompose_dim_k*, uses *junk_removal_ldt* to compute the varieties of $V(f)$ of a given dimension. When using *junk_removal_mem* (Algorithm 2.4.11), a witness set is needed for all higher dimensions in order to identify the junk points for a given dimension, which is not the case when using the local dimension test.

Algorithm 4.3.5. *decompose_dim_k$(f, k; W)$*

Input:

• *f: a system of n polynomials in $\mathbb{C}[x_1, \ldots, x_N]$.*

Output:

• *W: witness set for the dimension k varieties of $V(f)$ decomposed as in Eq. 2.2.1.*

Algorithm:

1. $[\widehat{W}] :=$ *dim_slicing_k(f, k).*

2. $[W_p] :=$ *junk_removal_ldt$(f, \widehat{W})$.*

3. $[W] :=$ *irreducible_decomp(f, W_p).*

CHAPTER 5

COMPUTATIONAL RESULTS

The following sections provide computational evidence of the advantages of using regeneration, regenerative cascade and the local dimension test. The serial computations were run on an Opteron 250 processor with 64-bit Linux and the parallel computations were run on a cluster consisting of a manager that uses one core of a Xeon 5410 processor and an additional 8 computing nodes each containing two Xeon 5410 processors running 64-bit Linux. The version 1.1.1. of the Bertini software package [2, 5] was used, with this version utilizing the multiplicity method of [12] in the local dimension computations.

Section 5.1 presents an example that compares regeneration and the diagonal homotopy approach of [42] for computing the isolated solutions of a polynomial system arising in kinematics.

Section 5.2 compares regeneration with the polyhedral homotopy method [20, 27, 50], currently considered the most efficient non-incremental way of solving sparse polynomial systems, on a collection of sparse polynomial systems arising from the discretization of nonlinear partial differential equations. Polyhedral homotopies are implemented in PHCpack [48] and HOM4PS-2.0 [25] with the computations using PHCpack v2.3.39 and HOM4PS-2.0.15.

Section 5.3 presents an example to illustrate potential difficulties arising in the heuristic local dimension approach of [22].

Section 5.4 presents a collection of examples to demonstrate the application of the regenerative cascade and the local dimension test for computing the numerical irreducible decomposition.

Finally, Section 5.5 presents computational results for the decomposition of permanental ideals.

5.1 A comparison of the equation-by-equation methods

To compare the equation-by-equation approaches using the diagonal homotopy [42] and regeneration, consider a polynomial system arising from the inverse kinematics problem of general six-revolute, serial-link robots described in [44, 53]. The polynomial system, available at [2], consists of 2 linear and 10 quadratic polynomials in 12 variables with total degree $1,024$. The system was constructed using random parameter values and is known to have 16 nonsingular finite isolated solutions.

As mentioned in Section 3.2.4, the ordering of the quadratic polynomials that minimizes the total number of paths that need to be tracked is suggested by the construction of the polynomials in the system. In this problem, the 12 variables correspond to the entries of 4 vectors in $\mathbb{C}^3$. Four of the quadratics correspond to normalizing each of these vectors to unit length. The other six quadratics provide conditions on the interaction between two or more vectors. This suggests that the four normalizing quadratics should be placed last. We shall call this optimal setup "order A" and, for comparison, consider another ordering, called "order B", that places the four normalizing quadratics ahead of the other six quadratics. The total number of paths tracked by regeneration was 628 and 928 and the total number of slices moved was 313 and 463 for "order A" and "order B", respectively.

TABLE 5.1

COMPARISON FOR SOLVING THE GENERAL 6R, SERIAL-LINK ROBOT SYSTEM SECURELY, WITH TIME IN SECONDS

	total degree		diagonal		regeneration		
setup	paths tracked	time	paths tracked	time	paths tracked	slices moved	time
order A	1,024	103.46	649	84.67	628	313	19.77
order B	1,024	103.46	949	121.46	928	463	29.96

To compare the methods, these two orderings of the polynomial system were solved using a generic total degree homotopy, the diagonal approach, and the regenerative approach. Table 5.1 compares the 3 methods which were run using adaptive precision tracking [3, 4], a tracking tolerance of 10^{-6}, a final tolerance of 10^{-10}, and utilizing the secure path tracking option in Bertini. Table 5.2 compares the 3 methods utilizing the same setup except that Bertini utilized nonsecure path tracking by truncating paths that correspond to points with a magnitude larger than 10^8.

With secure path tracking and optimal setup "order A," both the diagonal and regeneration approaches are faster than the standard total degree homotopy by reducing the number of paths to track and, more importantly, the number of paths that diverge to infinity. In "order B," the diagonal and regeneration approaches track more paths than "order A" with more diverging to infinity. When the infinite paths are truncated, the difference between "order A" and "order B" is reduced

TABLE 5.2

COMPARISON FOR SOLVING THE GENERAL 6R, SERIAL-LINK ROBOT SYSTEM ALLOWING PATH TRUNCATION, WITH TIME IN SECONDS

	total degree		diagonal		regeneration		
setup	paths tracked	time	paths tracked	time	paths tracked	slices moved	time
order A	1,024	31.16	649	43.09	628	313	11.76
order B	1,024	31.16	949	43.64	928	463	13.32

dramatically. All comparisons clearly demonstrate the disadvantage of doubling the number of extrinsic variables in the diagonal approach.

5.2 A large sparse polynomial system

Regeneration can be used to solve large polynomial systems when other methods are impractical. To illustrate this, consider a sparse polynomial system arising from ongoing research by the author with Andrew Sommese and Bei Hu (University of Notre Dame) related to the discretization of the stationary Lotka-Volterra population model with diffusion [28, 52].

Let $n \in \mathbb{N}$. For $1 \leq i \leq n$ and $1 \leq j \leq 4$, define

$$\begin{aligned} f_{ij} &= \frac{1}{25}\left(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}\right) \\ &\quad + \frac{1}{(n+1)^2}\left(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right) + \frac{1}{25(n+1)^2}u_{i,j}\left(1 - v_{i,j}\right), \\ g_{ij} &= \frac{1}{25}\left(v_{i+1,j} - 2v_{i,j} + v_{i-1,j}\right) \\ &\quad + \frac{1}{(n+1)^2}\left(v_{i,j+1} - 2v_{i,j} + v_{i,j-1}\right) + \frac{1}{25(n+1)^2}v_{i,j}\left(u_{i,j} - 1\right) \end{aligned} \tag{5.2.1}$$

with $u_{0,j} = v_{0,j} = u_{n+1,j} = v_{n+1,j} = u_{i,0} = v_{i,0} = u_{i,5} = v_{i,5} = 0$.

These systems consist of $8n$ quadratic polynomials in $8n$ variables and have 2^{4n} nonsingular isolated solutions. Even though the system has a natural 2-homogeneous structure, with each polynomial being of type $(1,1)$, it is not practical to use this as n increases. Also, even though the mixed volume of the system is the same as the number of solutions, 2^{4n}, current implementations of the polyhedral method failed to solve the system in less than 45 days for $n = 5$.

To solve the system using regeneration, we used the natural ordering of the equations and introduced the equations two at a time as suggested by Eq. 5.2.1. The linear product decomposition of the polynomials used was:

$$\begin{aligned} f_{ij} &\in \langle \{1, u_{i+1,j}, u_{i,j}, u_{i-1,j}, u_{i,j+1}, u_{i,j-1}, v_{i,j}\} \times \{1, v_{i,j}\} \rangle, \\ g_{ij} &\in \langle \{1, v_{i+1,j}, v_{i,j}, v_{i-1,j}, v_{i,j+1}, v_{i,j-1}, u_{i,j}\} \times \{1, u_{i,j}\} \rangle. \end{aligned} \tag{5.2.2}$$

In this decomposition, a generic member of the first set of the product decomposition is a generic linear that supports the second linear. In particular, in *regenerate*, the generic linear selected in Step 5 was taken as the first linear selected in Step 4.

With the above choices, regeneration tracks roughly 4 times as many paths as the number of solutions. Table 5.3 compares the number of paths for various methods and Table 5.4 contains timings for the various software packages. For

TABLE 5.3

COMPARISON FOR SOLVING SYSTEMS RELATED TO THE LOTKA-VOLTERRA POPULATION MODEL

	total degree	2-homogeneous	polyhedral	regeneration	
n	paths	paths	paths	paths	slices moved
1	256	70	16	60	42
2	65,536	12,870	256	1,020	762
3	16,777,216	2,704,156	4,096	16,380	12,282
4	4,294,967,296	601,080,390	65,536	262,140	196,602
5	1,099,511,627,776	137,846,528,820	1,048,576	4,194,300	3,145,722

$n \leq 4$, regeneration can solve the system using only double precision, and for $n = 5$, regeneration utilized adaptive precision tracking [3, 4] to track the paths since double precision was not adequate for some of the paths.

Table 5.4 also presents timings for the parallel implementation of regeneration in Bertini. A parallel version of PHCpack provides a parallel implementation of certain aspects of the polyhedral method [51], namely the solving of the start system and the tracking of the solutions paths. For $n = 3$, using serial processing, this part of the computation took approximately 59 minutes, or 0.22% of the computation. The author was unable to obtain a parallel version of HOM4PS-2.0.

TABLE 5.4

COMPARISON OF POLYHEDRAL METHOD AND REGENERATION FOR SOLVING SYSTEMS RELATED TO THE LOTKA-VOLTERRA POPULATION MODEL

	PHC	HOM4PS-2.0	Bertini	
n	polyhedral	polyhedral	regeneration	parallel regeneration
1	0.6s	0.1s	0.3s	
2	4m57s	7.3s	15.6s	
3	18d10h18m56s	9m32s	9m43s	
4	-	3d8h28m30s	5h22m15s	7m32s
5	-	-	6d16h27m3s	3h41m24s

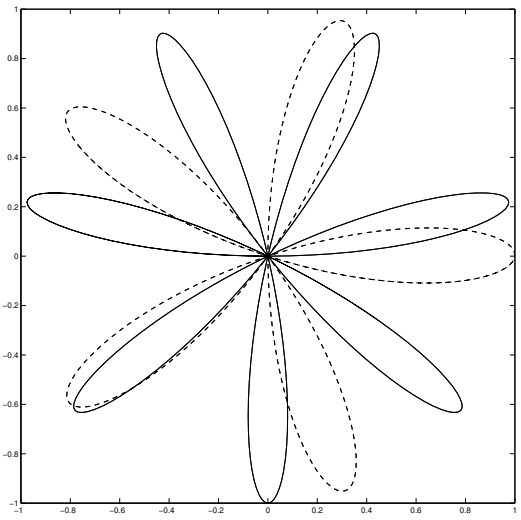

Figure 5.1. Rhodonea curves S_7 and $\widehat{S}_5$.

5.3 A local dimension example

Rhodonea curves, such as those displayed in Figure 5.1, are defined by polar equations of the form $r = \sin(k\theta)$. Denote the curve defined by the polar equation $r = \sin(k\theta)$ by S_k and let $\widehat{S}_k$ denote the curve obtained by applying a random rotation about the origin to S_k. For k even, S_k has $2k$ "petals" and for k odd, S_k has k "petals." For pairs of odd integers m and n, the origin is an isolated solution of the pair of equations defining the curves S_m and $\widehat{S}_n$. Due to the random rotation about the origin, the petals of S_m and $\widehat{S}_n$ do not share a common tangent direction at the origin. As a result, the multiplicity at the origin is mn.

The Rhodonea curve S_1 is the solution set of the polynomial $g(x, y) = x^2 + y^2 - y$. That is, S_1 is the circle of radius $\frac{1}{2}$ centered at $\left(0, \frac{1}{2}\right)$. For a 2×2 random

real orthogonal matrix A, define $(\widehat{x}, \widehat{y})$ by

$$\begin{bmatrix} \widehat{x} \\ \widehat{y} \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix}.$$

The solution set $\widehat{S}_1$ of $\widehat{g}(x, y) = g(\widehat{x}, \widehat{y})$ is a rotation of S_1 about the origin. The set $S_1 \cap \widehat{S}_1$ is the solution set of the system $g(x, y) = \widehat{g}(x, y) = 0$ which consists of the 2 intersection points of the corresponding two circles, namely the origin and the point $(\bar{x}, \bar{y})$ away from the origin.

Let $f_1(x, y) = xg(x, y)$ and $f_2(x, y) = x\widehat{g}(x, y)$. The algebraic set $V(f_1, f_2)$ has 2 irreducible components: the isolated point $p = (\bar{x}, \bar{y})$ and the line $L = \{x = 0\}$. It can be shown that the origin is an embedded point of the line L. Bertini used a total degree homotopy and found 2 paths that lead to the origin. Since the Jacobian of the system is the zero matrix at the origin, the multiplicity must be at least 3 in order for the origin to be an isolated solution. The algorithm *is_isolated* identified that the origin is not isolated in 0.005 seconds. The full numerical irreducible decomposition was completed by Bertini in 0.11 seconds.

The Matlab module described in [22] used a polyhedral homotopy which also had 2 paths leading to the origin. However, it was unable to identify that the origin lies on a one-dimensional component. The module did correctly identify $(\bar{x}, \bar{y})$ as isolated and that another point of the form $(0, y')$ (with $y' \neq 0$) was a point lying on a one-dimensional component.

5.4 A collection of high-dimensional examples

Computing a numerical irreducible decomposition for a given polynomial system is a fundamental algorithm in numerical algebraic geometry. For systems

with components in many different dimensions, the dimension-by-dimension slicing algorithm *dim_slicing* generally produces a witness superset that contains more junk points than the cascade algorithm *cascade* and the regenerative cascade algorithm *regen_cascade*. No matter which algorithm is used to construct a witness superset, filtering out the junk points using the membership test junk removal algorithm *junk_removal_mem* can result in a bottleneck. By using the local dimension test junk removal algorithm *junk_removal_ldt*, the filtering of junk points is very efficient.

Consider the homogeneous polynomial system $F_{k,m,n}$ constructed by taking the determinants of the $k \times k$ adjacent minors of an $m \times n$ matrix of indeterminants [13, 18, 19, 46]. In [13], it was shown that $V(F_{2,2,n})$ has codimension $n-1$, degree 2^{n-1}, and decomposes into $f(n)$ varieties, where $f(n) = f(n-1) + f(n-2)$ is the nth Fibonacci number with starting values $f(0) = 0$ and $f(1) = 1$. This result was generalized to $F_{m,m,n}$ in [19], namely $V(F_{m,m,n})$ has codimension $n-m+1$, degree m^{n-m+1}, and decomposes into $f_m(n)$ varieties, where $f_m(n) = f_m(n-1) + \cdots + f_m(n-m)$ with starting values $f_m(0) = \cdots = f_m(m-2) = 0$ and $f_m(m-1) = 1$.

The system $F_{2,m,n}$ was studied in both [18] and [19]. A combinatorial description for the generators of the varieties in $V(F_{2,m,n})$ was presented in [19]. In [18], it was shown that $V(F_{2,3,n})$ consists of $\phi(n)$ varieties, where $\phi(n) = \phi(n-1) + \phi(n-2) + \sum_{j=0}^{n-5} \phi(j)$ with starting values $\phi(0) = \phi(1) = 1,\ \phi(2) = 2,\ \phi(3) = 3,$ and $\phi(4) = 6$. Table 5.5 summarizes the total number of varieties for $V(F_{2,3,n})$, $3 \leq n \leq 9$, as well as list the codimension of the varieties.

To compare the algorithms, we computed the numerical irreducible decomposition for $V(F_{2,3,n})$ with $3 \leq n \leq 9$. Table 5.6 compares the total number of paths that are tracked and the number of junk points for the three witness superset

TABLE 5.5

SUMMARY OF THE VARIETIES IN $V(F_{2,3,n})$, $3 \leq n \leq 9$

n	#varieties	codimension of varieties
3	3	3,4
4	6	4,5,6
5	10	5,6,7,8
6	18	6,7,8,9,10
7	32	7,8,9,10,11,12
8	57	8,9,10,11,12,13,14
9	102	9,10,11,12,13,14,15,16

algorithms. The total number of slices that were moved using *regen_cascade* is also included in this table. Table 5.7 compares the time needed to compute the numerical irreducible decomposition for $3 \leq n \leq 8$ using the different algorithms with serial processing and Table 5.8 compares the time needed for $n = 8, 9$ using parallel processing. For these parallel runs, Bertini used a dynamic distribution of the points for junk point filtering since each point can be handled independently.

When using *junk_removal_mem*, as n increases, a growing majority of the computational cost is the filtering of the junk points. In this case, the added cost of computing a witness superset with less junk points using *cascade* is advantageous over using *dim_slicing*. Since *regen_cascade* computes a witness superset with the same number of junk points as *cascade* with less cost, it is advantageous over both *cascade* and *dim_slicing*.

TABLE 5.6

COMPARISON FOR COMPUTING A NUMERICAL IRREDUCIBLE DECOMPOSITION FOR $V(F_{2,3,n})$, $3 \leq n \leq 9$

	paths tracked (slices moved)			number of junk points		
n	*regen_cascade*	*dim_slicing*	*cascade*	*regen_cascade*	*dim_slicing*	*cascade*
3	26 (12)	30	56	6	10	6
4	96 (47)	126	295	30	60	30
5	340 (169)	510	1,380	125	295	125
6	1,190 (594)	2,046	6,050	486	1,342	486
7	4,150 (2,074)	8,190	25,465	1,813	5,853	1,813
8	14,456 (7,227)	32,766	104,247	6,600	24,910	6,600
9	50,336 (25,167)	131,070	418,289	23,665	104,399	23,665

TABLE 5.7

TIMING COMPARISON FOR COMPUTING A NUMERICAL IRREDUCIBLE DECOMPOSITION FOR $V(F_{2,3,n})$, $3 \leq n \leq 8$

	junk_removal_ldt			*junk_removal_mem*		
n	*regen_cascade*	*dim_slicing*	*cascade*	*regen_cascade*	*dim_slicing*	*cascade*
3	0.1s	0.1s	0.2s	0.1s	0.1s	0.2s
4	0.6s	0.8s	1.1s	0.8s	1.1s	1.3s
5	3.1s	4.6s	7.4s	6.2s	11.9	11.2s
6	15.6s	29.0s	48.4s	1m1s	2m14s	1m34s
7	1m16s	3m8s	5m23s	10m36s	25m39s	14m54s
8	6m33s	19m45s	29m22s	2h12m54s	5h21m48s	2h33m5s

TABLE 5.8

TIMING COMPARISON FOR COMPUTING A NUMERICAL IRREDUCIBLE DECOMPOSITION IN PARALLEL FOR $V(F_{2,3,n})$, $n = 8, 9$

	junk_removal_ldt			*junk_removal_mem*		
n	*regen_cascade*	*dim_slicing*	*cascade*	*regen_cascade*	*dim_slicing*	*cascade*
8	27.2s	34.1s	49.4s	2m14s	5m20s	2m41s
9	1m11s	2m48s	3m21s	18m42s	52m57s	20m21s

When using *junk_removal_ldt*, as n increases, a growing majority of the computational cost is the computation of a witness superset. In this case, the reduced cost of computing a witness superset using *dim_slicing* is advantageous over using *cascade*. Since *regen_cascade* computes a witness superset that is computationally cheaper than *dim_slicing*, it is advantageous over both *dim_slicing* and *cascade*.

It should be noted that since, for each codimension i, the degree of the algebraic set consisting of the i-codimensional varieties is considerably smaller than the total degree of the polynomial system $\mathfrak{R}(F_{2,3,n}, i)$, the computational cost of using algorithm *irreducible_decomp* to decompose the witness set into irreducible witness sets is small compared to the cost of computing the witness set. For example, using serial processing with $n = 8$, it took about 57 seconds to decompose the witness set. For comparison, with *junk_removal_ldt*, it took about 336 seconds using *regen_cascade*, $1,128$ seconds using *dim_slicing*, and $1,705$ seconds using *cascade* to compute the witness set.

5.5 Computing the numerical irreducible decomposition for permanental ideals

The system $F_{k,m,n}$ of Section 5.4 was constructed by taking the determinant of the $k \times k$ adjacent minors of an $m \times n$ matrix of indeterminants. Related to $F_{k,m,n}$, consider the homogeneous system $P_{k,m,n}$ constructed by taking the permanent of the $k \times k$ adjacent minors of an $m \times n$ matrix of indeterminants. Previous work, e.g., [21, 23, 34], has focused on proving primary decomposition results for $P_{2,m,n}$ and $P_{3,m,n}$.

The system $P_{m,m,n}$ consists of $n - m + 1$ homogeneous polynomials of degree m in mn variables. For $2 \leq m \leq 5$ and various values of n, we used Bertini to compute the numerical irreducible decomposition of $P_{m,m,n}$. Tables 5.9, 5.10, 5.11,

and 5.12 summarize the results of these computations. Based on these results, we formulate the following conjecture.

Conjecture 5.5.1. *The algebraic set* $V(P_{m,m,n})$ *has codimension* $n-m+1$*, degree* m^{n-m+1}*, and decomposes into* $\psi_m(n)$ *varieties, where*

$$\psi_m(n) = \psi_m(n-1) + \psi_m(n-m)$$

with starting values $\psi_m(0) = \cdots = \psi_m(m-2) = 0$ *and* $\psi_m(m-1) = 1$.

TABLE 5.9

SUMMARY OF $V(P_{2,2,n})$ FOR $2 \leq n \leq 17$

n	codimension	degree	#varieties
2	1	2	1
3	2	4	2
4	3	8	3
5	4	16	5
6	5	32	8
7	6	64	13
8	7	128	21
9	8	256	34
10	9	512	55
11	10	1,024	89
12	11	2,048	144
13	12	4,096	233
14	13	8,192	377
15	14	16,384	610
16	15	32,768	987
17	16	65,536	1597

TABLE 5.10

SUMMARY OF $V(P_{3,3,n})$ FOR $3 \leq n \leq 13$

n	codimension	degree	#varieties
3	1	3	1
4	2	9	1
5	3	27	2
6	4	81	3
7	5	243	4
8	6	729	6
9	7	2,187	9
10	8	6,561	13
11	9	19,683	19
12	10	59,049	28
13	11	177,147	41

TABLE 5.11

SUMMARY OF $V(P_{4,4,n})$ FOR $4 \leq n \leq 12$

n	codimension	degree	#varieties
4	1	4	1
5	2	16	1
6	3	64	1
7	4	256	2
8	5	1,024	3
9	6	4,096	4
10	7	16,384	5
11	8	65,536	7
12	9	262,144	10

TABLE 5.12

SUMMARY OF $V(P_{5,5,n})$ FOR $5 \leq n \leq 12$

n	codimension	degree	#varieties
5	1	5	1
6	2	25	1
7	3	125	1
8	4	625	1
9	5	3,125	2
10	6	15,625	3
11	7	78,125	4
12	8	390,625	5

REFERENCES

1. D.J. Bates, J.D. Hauenstein, C. Peterson, and A.J. Sommese. A local dimension test for numerically approximated points on algebraic sets. Preprint, 2009.

2. D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. Bertini: Software for numerical algebraic geometry. Available at www.nd.edu/∼sommese/bertini.

3. D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. Adaptive multiprecision path tracking. *SIAM J. Numer. Anal.*, 46(2):722–746, 2008.

4. D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. Stepsize control for adaptive multiprecision path tracking. To appear in *Interactions of Classical and Numerical Algebraic Geometry*, D. Bates, G. Besana, S. Di Rocco, and C. Wampler (eds.), *Contemporary Mathematics*, 2009.

5. D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler, II. Software for numerical algebraic geometry: a paradigm and progress towards its implementation. In *Software for algebraic geometry*, volume 148 of *IMA Vol. Math. Appl.*, pages 1–14. Springer, New York, 2008.

6. D.J. Bates, C. Peterson, and A.J. Sommese. A numerical-symbolic algorithm for computing the multiplicity of a component of an algebraic set. *J. Complexity*, 22(4):475–489, 2006.

7. B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselmente des Restklassenrings nach einem nulldimensionalen Polynomideal*. PhD thesis, University of Innsbruck, Innsbruck, Austria, 1965.

8. D. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms: An introduction to computational algebraic geometry and commutative algebra*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, second edition, 1997.

9. D. Cox, J. Little, and D. O'Shea. *Using algebraic geometry*, volume 185 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1998.

10. D.F. Davidenko. On a new method of numerical solution of systems of nonlinear equations. *Doklady Akad. Nauk SSSR (N.S.)*, 88:601–602, 1953.

11. D.F. Davidenko. On approximate solution of systems of nonlinear equations. *Ukrain. Mat. Žurnal*, 5:196–206, 1953.

12. B.H. Dayton and Z. Zeng. Computing the multiplicity structure in solving polynomial systems. In *ISSAC'05*, pages 116–123 (electronic). ACM, New York, 2005.

13. P. Diaconis, D. Eisenbud, and B. Sturmfels. Lattice walks and primary decomposition. In *Mathematical essays in honor of Gian-Carlo Rota (Cambridge, MA, 1996)*, volume 161 of *Progr. Math.*, pages 173–193. Birkhäuser Boston, Boston, MA, 1998.

14. W. Fulton. *Algebraic curves. An introduction to algebraic geometry*. W. A. Benjamin, Inc., New York-Amsterdam, 1969. Notes written with the collaboration of Richard Weiss, Mathematics Lecture Notes Series.

15. P. Griffiths and J. Harris. *Principles of algebraic geometry*. Wiley Classics Library. John Wiley & Sons Inc., New York, 1994. Reprint of the 1978 original.

16. R. Hartshorne. *Algebraic geometry*. Graduate Texts in Mathematics, No. 52. Springer-Verlag, New York, 1977.

17. J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. Regeneration homotopies for solving systems of polynomials. Preprint, 2009.

18. S. Hoşten and J. Shapiro. Primary decomposition of lattice basis ideals. *J. Symbolic Comput.*, 29(4-5):625–639, 2000. Symbolic computation in algebra, analysis, and geometry (Berkeley, CA, 1998).

19. S. Hoşten and S. Sullivant. Ideals of adjacent minors. *J. Algebra*, 277:615–642, 2004.

20. B. Huber and B. Sturmfels. A polyhedral method for solving sparse polynomial systems. *Math. Comp.*, 64(212):1541–1555, 1995.

21. G.A. Kirkup. Minimal primes over permanental ideals. *Trans. Amer. Math. Soc.*, 360(7):3751–3770, 2008.

22. Y.C. Kuo and T.Y. Li. Determining dimension of the solution component that contains a computed zero of a polynomial system. *J. Math. Anal. Appl.*, 338(2):840–851, 2008.

23. R.C. Laubenbacher and I. Swanson. Permanental ideals. *J. Symbolic Comput.*, 30(2):195–205, 2000.

24. G. Lecerf. Quadratic Newton iteration for systems with multiplicity. *Found. Comput. Math.*, 2(3):247–293, 2002.

25. T.L. Lee, T.Y. Li, and C.H. Tsai. HOM4PS-2.0: a software package for solving polynomial systems by the polyhedral homotopy continuation method. *Computing*, 83(2-3):109–133, 2008.

26. A. Leykin, J. Verschelde, and A. Zhao. Newton's method with deflation for isolated singularities of polynomial systems. *Theoret. Comput. Sci.*, 359(1-3):111–122, 2006.

27. T.Y. Li. Numerical solution of polynomial systems by homotopy continuation methods. In *Handbook of numerical analysis, Vol. XI*, Handb. Numer. Anal., XI, pages 209–304. North-Holland, Amsterdam, 2003.

28. A.J. Lotka. Undamped oscillations derived from the laws of mass action. *J. Amer. Chem. Soc.*, 42(8):1595–1599, 1920.

29. F.S. Macaulay. *The algebraic theory of modular systems*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, 1994. Revised reprint of the 1916 original, with an introduction by Paul Roberts.

30. A.P. Morgan. A transformation to avoid solutions at infinity for polynomial systems. *Appl. Math. Comput.*, 18(1):77–86, 1986.

31. A.P. Morgan and A.J. Sommese. Coefficient-parameter polynomial continuation. *Appl. Math. Comput.*, 29(2, part II):123–160, 1989.

32. A.P. Morgan, A.J. Sommese, and C.W. Wampler. A product-decomposition bound for Bezout numbers. *SIAM J. Numer. Anal.*, 32(4):1308–1325, 1995.

33. D. Mumford. *Algebraic geometry. I.* Classics in Mathematics. Springer-Verlag, Berlin, 1995. Complex projective varieties, Reprint of the 1976 edition.

34. H. Niermann. *Beiträge zur konstruktiven idealtheorie*. PhD thesis, University of Dortmund, Dortmund, Germany, 1997.

35. T. Ojika. Modified deflation algorithm for the solution of singular problems. I. A system of nonlinear algebraic equations. *J. Math. Anal. Appl.*, 123(1):199–221, 1987.

36. T. Ojika, S. Watanabe, and T. Mitsui. Deflation algorithm for the multiple roots of a system of nonlinear equations. *J. Math. Anal. Appl.*, 96(2):463–479, 1983.

37. A.J. Sommese, J. Verschelde, and C.W. Wampler. Using monodromy to decompose solution sets of polynomial systems into irreducible components. In *Applications of algebraic geometry to coding theory, physics and computation (Eilat, 2001)*, volume 36 of *NATO Sci. Ser. II Math. Phys. Chem.*, pages 297–315. Kluwer Acad. Publ., Dordrecht, 2001.

38. A.J. Sommese and J. Verschelde. Numerical homotopies to compute generic points on positive dimensional algebraic sets. *J. Complexity*, 16(3):572–602, 2000. Complexity theory, real machines, and homotopy (Oxford, 1999).

39. A.J. Sommese, J. Verschelde, and C.W. Wampler. Numerical decomposition of the solution sets of polynomial systems into irreducible components. *SIAM J. Numer. Anal.*, 38(6):2022–2046 (electronic), 2001.

40. A.J. Sommese, J. Verschelde, and C.W. Wampler. Numerical irreducible decomposition using projections from points on the components. In *Symbolic computation: solving equations in algebra, geometry, and engineering (South Hadley, MA, 2000)*, volume 286 of *Contemp. Math.*, pages 37–51. Amer. Math. Soc., Providence, RI, 2001.

41. A.J. Sommese, J. Verschelde, and C.W. Wampler. Symmetric functions applied to decomposing solution sets of polynomial systems. *SIAM J. Numer. Anal.*, 40(6):2026–2046 (electronic) (2003), 2002.

42. A.J. Sommese, J. Verschelde, and C.W. Wampler. Solving polynomial systems equation by equation. In *Algorithms in algebraic geometry*, volume 146 of *IMA Vol. Math. Appl.*, pages 133–152. Springer, New York, 2008.

43. A.J. Sommese and C.W. Wampler. Numerical algebraic geometry. In *The mathematics of numerical analysis (Park City, UT, 1995)*, volume 32 of *Lectures in Appl. Math.*, pages 749–763. Amer. Math. Soc., Providence, RI, 1996.

44. A.J. Sommese and C.W. Wampler, II. *The numerical solution of systems of polynomials arising in engineering and science*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2005.

45. H.J. Stetter. *Numerical polynomial algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2004.

46. B. Sturmfels. *Solving systems of polynomial equations*, volume 97 of *CBMS Regional Conference Series in Mathematics*. Published for the Conference Board of the Mathematical Sciences, Washington, DC, 2002.

47. G.H. Thallinger. Analysis of zero clusters in multivariate polynomial systems. Diploma Thesis, Tech. Univ. Vienna, 1996.

48. J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM T. Math. Software*, 25(2):251–276, 1999.

49. J. Verschelde and R. Cools. Symbolic homotopy construction. *Appl. Algebra Engrg. Comm. Comput.*, 4(3):169–183, 1993.

50. J. Verschelde, P. Verlinden, and R. Cools. Homotopies exploiting Newton polytopes for solving sparse polynomial systems. *SIAM J. Numer. Anal.*, 31(3):915–930, 1994.

51. J. Verschelde and Y. Zhuang. Parallel implementation of the polyhedral homotopy method. In *International Conference on Parallel Processing Workshops*, pages 481–488. IEEE Computer Society, Los Alamitos, CA, USA, 2006.

52. V. Volterra. Variazionie fluttuazioni del numero d'individui in specie animali conviventi. *Mem. Acad. Lincei.*, 2:31–113, 1926.

53. C.W. Wampler and A.P. Morgan. Solving the kinematics of general 6R manipulators using polynomial continuation. In *Robotics: applied mathematics and computational aspects (Loughborough, 1989)*, volume 41 of *Inst. Math. Appl. Conf. Ser. New Ser.*, pages 57–69. Oxford Univ. Press, New York, 1993.

54. B. Yu and B. Dong. A hybrid polynomial system solving method for mixed trigonometric polynomial systems. *SIAM J. Numer. Anal.*, 46(3):1503–1518, 2008.

55. Z. Zeng. The closedness subspace method for computing the multiplicity structure of a polynomial system. To appear in *Interactions of Classical and Numerical Algebraic Geometry*, D. Bates, G. Besana, S. Di Rocco, and C. Wampler (eds.), *Contemporary Mathematics*, 2009.