Dissertation

Algorithms in Numerical Algebraic Geometry and Applications

Submitted by

Eric M. Hanson

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2015

Doctoral Committee:

Advisor: Daniel J. Bates

Chris Peterson
Renzo Cavalieri
Anthony Maciejewski

Abstract

Algorithms in Numerical Algebraic Geometry and Applications

The topics in this dissertation, while independent, are unified under the field of numerical algebraic geometry. With ties to some of the oldest areas in mathematics, numerical algebraic geometry is relatively young as a field of study in its own right. The field is concerned with the numerical approximation of the solution sets of systems of polynomial equations and the manipulation of these sets. Given a polynomial system $f : \mathbb{C}^N \to \mathbb{C}^n$, the methods of numerical algebraic geometry produce numerical approximations of the isolated solutions of $f(z) = 0$, as well as points on any positive-dimensional components of the solution set, $\mathbf{V}(f)$. In a short time, the work done in numerical algebraic geometry has significantly pushed the boundary of what is computable. This dissertation aims to further this work by contributing new algorithms to the field and using cutting edge techniques of the field to expand the scope of problems that can be addressed using numerical methods. We begin with an introduction to numerical algebraic geometry and subsequent chapters address independent topics: perturbed homotopies, exceptional sets and fiber products, and a numerical approach to finding unit distance embeddings of finite simple graphs.

One of the most recent advances in numerical algebraic geometry is regeneration, an equation-by-equation homotopy method that is often more efficient then other approaches. However, the basic form of regeneration will not necessarily find all isolated singular solutions of a polynomial system without the additional cost of using deflation. In the second chapter, we present an alternative to deflation in the form of perturbed homotopies for solving polynomial systems. In particular, we propose first solving a perturbed version of the polynomial system, followed by a parameter homotopy to remove the perturbation. The aim

of this chapter is two-fold. First, such perturbed homotopies are sometimes more efficient than regular homotopies, though they can also be less efficient. Second, a useful consequence is that the application of this perturbation to regeneration will yield all isolated solutions, including all singular isolated solutions.

The third chapter considers families of polynomial systems which depend on parameters. There is a typical dimension for the variety defined by a system in the family; however, this dimension may jump for parameters in algebraic subsets of the parameter space. Sommese and Wampler exploited fiber products to give a numerical method for identifying these special parameter values. In this chapter, we propose a refined numerical approach to fiber products, which uses recent advancements in numerical algebraic geometry, such as regeneration extension. We show that this method is sometimes more efficient then known techniques. This gain in efficiency is due to the fact that regeneration extension allows the construction of the fiber product to be restricted to specified irreducible components. This work is motivated by applications in Kinematics - the study of mechanisms. As such we use an algebraic model of a two link arm to illustrate the algorithms developed in this chapter.

The topic of the last chapter is the identification of unit distance embeddings of finite simple graphs. Given a graph $G(V, E)$, a unit distance embedding is a map ϕ from the vertex set V into a metric space M such that if $\{v_i, v_j\} \in E$ then the distance between $\phi(v_i)$ and $\phi(v_j)$ in M is one. Given G, we cast the question of the existence of a unit distance embedding in $\mathbb{R}^n$ as the question of the existence of a real solution to a system of polynomial equations. As a consequence, we are able to develop theoretic algorithms for determining the existence of a unit distance embedding and for determining the smallest dimension of $\mathbb{R}^n$ for which a unit distance embedding of G exists (that is, we determine the minimal embedding dimension of G). We put these algorithms into practice using the methods of numerical

algebraic geometry. In particular, we consider unit distance embeddings of the Heawood Graph. This is the smallest example of a point-line incidence graph of a finite projective plan. In 1972, Chvátal conjectured that point-line incidence graphs of finite projective planes do not have unit-distance embeddings into $\mathbb{R}^2$. In other words, Chvátal conjectured that the minimal embedding dimension of any point-line incidence graph of a finite projective plane is at least 3. We disprove this conjecture, adding hundreds of counterexamples to the 11 known counterexamples found by Gerbracht.

Acknowledgements

During my years as a student working on this dissertation, I have been blessed with the support, guidance, mentorship, and friendship of many. Dan Bates was by far one of the most influential people during this time. As my advisor, one might think that this is expected; however, Dan's role in my development as a mathematician goes above and beyond that of most advisors. I have no hope of expressing the impact of Dan's selfless devotion to his students and others at Colorado State University. Through countless research meetings, 3AM emails, discussions about jobs, and the like, Dan has helped shape the future of many students. I consider myself lucky to be among Dan's advisees, value his friendship, and hope to live up to his example in my future interactions with students.

There were many other collaborators that contributed to the work in this dissertation, including Chris Peterson, Jonathan Hauenstein, Charles Wampler, Brent Davis, and David Eklund. In particular, Chris Peterson's guidance, ideas, and willingness to discuss our work had an immense impact on this dissertation. Chris has also been exceptionally influential in my development as a mathematician, second only to my advisor. I am thankful for all of our interactions, for his mentoring, and friendship.

While their contributions were not directly related to this work, I would like to acknowledge a number of my peers. I am thankful to Francis Motta, Lori Ziegelmeier, and Tim Hodges for the countless hours we spent collaborating and consider each of you a friend. I am also thankful for the friendship and support of Drew Schwickerath, Steven Ihde, Michael Mikucki, and Sofya Chepushtanova. It has been a privilege to share my time at CSU with each of you and so many others.

There are certainly many other people that contributed to this dissertation and have influenced my development as a mathematician and educator. I am thankful for all of you, but especially for the many faculty and staff members of the mathematics department. You have created a very special place for one to study and grow, a place where many, including me, have experienced tremendous change in our lives.

Finally, I cannot say enough how blessed I am by my family and especially my wife. For the continued support of my academic pursuits I am grateful to my parents, Tom and Deb Hanson. To my wife, Erin Hanson, I owe so much. I will not try to list all the ways you have supported my efforts as a student, but simply wish to say that I love you and I am grateful to walk through life with such a wonderful person.

Table of Contents

CHAPTER 1

An Introduction to Numerical Algebraic Geometry

The field of *numerical algebraic geometry* is primarily concerned with the study of algorithms for computing numerical approximations to solutions of systems of polynomial equations and manipulating the resulting solution sets. The basic computational tool of numerical algebraic geometry is the homotopy. This chapter begins by describing three homotopies for approximating the isolated solutions of a system of polynomial equations, the total degree homotopy, the basic regeneration homotopy, and parameter homotopies. The chapter then concludes with a discussion of the dimension by dimension approach to computing the numerical irreducible decomposition of a solution set of a polynomial system with positive dimensional components.

1.1. Homotopies for Zero Dimensional Solution Sets

Let $f := (f_1, \ldots, f_n) : \mathbb{C}^N \to \mathbb{C}^n$ be a polynomial system with solution set

$$\mathbf{V}(f) := \{z \in \mathbb{C}^N : f_i(z) = 0, \text{ for } i = 1, \ldots, n\}.$$

For this section we assume that $n = N$ (a *square* system) and focus on methods for computing approximations to the isolated solutions of $\mathcal{V}(f)$. The case where $n \neq N$ is addressed at the end of this chapter.

Recall that a point $p \in \mathbb{C}^N$ is an *isolated solution* of $f(z)$ if $f(p) = 0$ and if there is some small ϵ such that $f(w) \neq 0$ for all points w in the punctured neighborhood

$$B_p(\epsilon) = \left\{ z \in \mathbb{C}^N : 0 < ||z - p|| < \epsilon \right\}.$$

Alternatively, p is an isolated solution if the local dimension of $V(f)$ at p is zero. An isolated solution p of $f(z)$ is said to be *singular* if the Jacobian matrix of $f(z)$ is not full rank when evaluated at p. All isolated solutions have associated to them a positive integer, the *multiplicity* of the solution, which is greater than 1 for singular solutions [4].

1.1.1. Basic Homotopy. Given a polynomial system $f : \mathbb{C}^N \to \mathbb{C}^N$, homotopy continuation is a three-step method for approximating all isolated solutions of $f = 0$:

(1) Choose a polynomial system $g : \mathbb{C}^N \to \mathbb{C}^N$ that is in some way "similar" to f but that is easier to solve.

(2) Solve g and form the *homotopy function*

$$H : \mathbb{C}^N \times \mathbb{C} \to \mathbb{C}^N,$$

given by

$$H(z;t) = (1-t)f(z) + tg(z),$$

so that $H(z;0) = f(z)$ and $H(z;1) = g(z)$.

(3) As t varies from 1 to 0, track the solutions of $H(z;t)$, starting with the known solutions of $g(z)$ and leading to the solutions of $f(z)$. This can be accomplished via numerical predictor-corrector methods.

Remark 1.1.1. *When the specific type of homotopy used is not important, the notation $f \to g$ is used to denote the homotopy from f to g as in (2) above.*

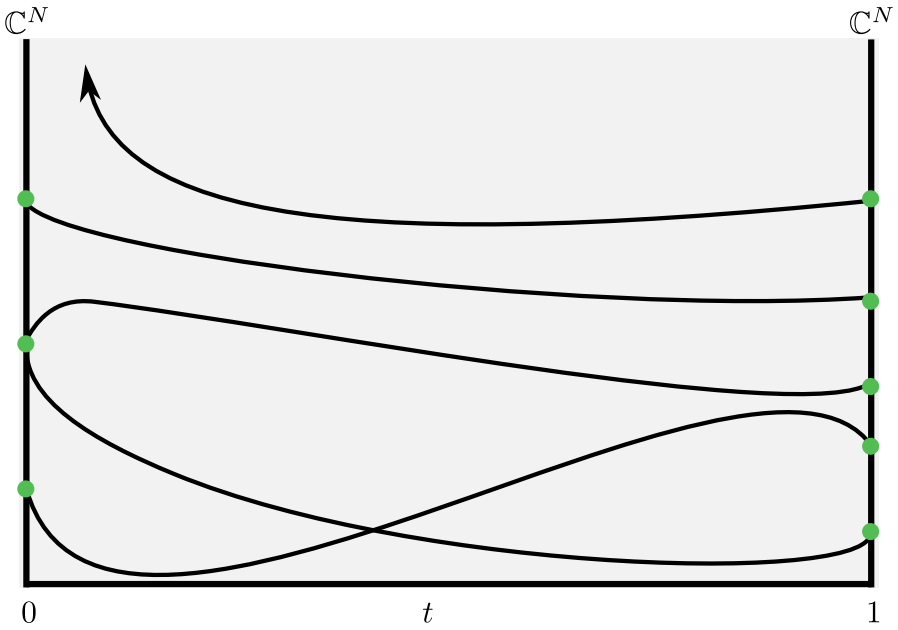


FIGURE 1.1. Homotopy Paths

There are many ways to choose a start system g when building the basic homotopy, including the total degree system (outlined in Section 1.1.2), the mutlihomogenous system, and others (see [4, 30]). In this dissertation, the details behind these other start systems are left to the references. The predictor corrector methods used in the third step of the basic homotopy are outlined in Section 1.1.3.

1.1.2. TOTAL DEGREE HOMOTOPY. In the description of the basic homotopy, Section 1.1.1, the first step is to cast f into a family of systems by choosing a similar system g. One such way to do this is to choose g to have the same total degree as f. The total degree of a polynomial system is the product, $\prod_{i-1}^{N} d_i$ where d_i is the degree of equation i in the polynomial system. The total degree provides a bound on the number of solutions of the polynomial system. This bound is called the Bézout bound.

Once such a g is identified, a homotopy can be constructed and used to approximate the solutions to f using the steps of the basic homotopy. A homotopy constructed in this way is called a total degree homotopy [4, 30].

The key to this set up is choosing a g that is easy to solve. A simple choice for g is to pick the system:

$$g = \begin{bmatrix} z^{d_1} - 1 \\ \vdots \\ z^{d_N} - 1 \end{bmatrix} = 0$$

where d_i is the degree of f_i. Clearly the number of solutions of g is the total degree of f and the solutions are simply the N-tuples of the cyclic roots of unity.

Certainly $g = 0$ can have many more solutions then $f = 0$, such extraneous solution paths in the homotopy diverge, unless one chooses to work over projective space [4, 30]. Tracking paths to infinity in projective space is very costly, so in practice a threshold is used to determine divergence. Multiple paths can converge to the same solution of $f = 0$, which indicates the multiplicity of the solution.

1.1.3. Predictor-Corrector Methods. Numerical predictor-corrector methods are a standard tool of numerical analysis. In the setting of path following, the basic idea of a predictor corrector method is to approximate the next point of a discretization of the path with a predictor and then refine the approximation with a corrector. A common combination of predictor and corrector is Euler's Method and Newton's Method, respectively. Suppose we have a point on the path (z_i, t_i). The procedure to find the next path point goes roughly as follows:

(1) Locally approximate the homotopy function $H(z;t)$ at (z_i, t_i). Using a taylor series we would have

$$H(z_i + \Delta z, t_i + \Delta t) = H(z_i, t_i) + H_z(z_i, t_i)\Delta z + H_t(z_i, t_i,)\Delta t + \text{H.O.T.}$$

(2) Incrementing t_i by Δt, a prediction for the solution to $H(z, t_i + \Delta t)$ is obtained by solving the first order terms of $H(z_i + \Delta z, t_i + \Delta t) = 0$ for Δz:

$$\Delta z = -H_z(z_i, t_i)^{-1} H_t(z_i, t_i)\Delta t$$

(3) Then the approximate solution $(\hat{z}_{i+1}, \hat{t}_{i+1}) = (z_i + \Delta z, t_i + \Delta t)$ can be refined using Newton's method. That is, t can be held constant and a change to $\hat{z}_{i+1}$ can be computed by

$$\Delta \hat{z}_{i+1} = -H_z^{-1}(\hat{z}_{i+1}, \hat{t}_{i+1}) H(\hat{z}_{i+1}, \hat{t}_{i+1})$$

So our new approximate solution is $(z_{i+1}, t_{i+1}) = (\hat{z}_{i+1} + \Delta \hat{z}_{i+1}, \hat{t}_{i+1})$

In the basic homotopy, this procedure is repeated until $t = 0$. However, more robust methods use *endgames* to approximate the end of the path [4, 30]. It is also worth noting that other predictors and correctors can be used and [4, 30] are reference for these details.

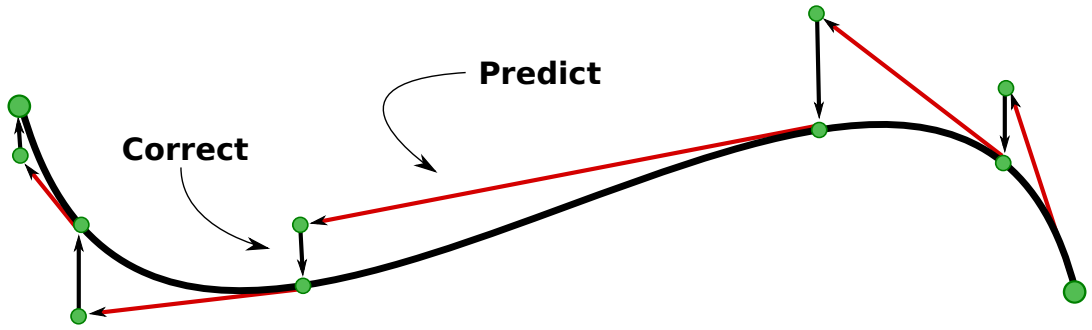


FIGURE 1.2. Path Tracking

The basic homotopy and predictor-corrector methods are the primary engine of modern homotopy continuation. However, a number of other tools exist that make the process more feasible. The current, cutting edge approaches to homotopy continuation include the use of adaptive precision, adaptive step size, endgames, the "gamma trick," and other fundamental tools. We point the reader to [4, 30] which include discussions of many of these topics and [18, 22, 33] which contain details on the related polyhedral homotopy approach to solving polynomial systems.

1.2. Regeneration Homotopies

If we think of each equation in a polynomial system as imposing a condition on the solution set, then a total degree homotopy approximates solutions by imposing all of these conditions simultaneously. A regenerative approach deviates from this in that systems are built up equation by equation, so the conditions are imposed one at at time; thus regeneration is often referred to as an equation by equation method. Here we will outline the basic regeneration algorithm for finding the nonsingular isolated solutions of a polynomial system. In practice, regeneration is typically quite efficient, often producing the nonsingular isolated solutions of a polynomial system much faster than standard homotopy methods, in part by automatically taking advantage of any symmetries or structure in $f(z)$ (see §9.3 of [14]). This basic form of regeneration first appeared in [14] and was extended to the case of positive-dimensional solution sets in [15].

Let d_i denote the degree of polynomial f_i for $i = 1, \ldots, N$. For each $1 < i < N$ and $0 < j < d_i$ let $L_i^{(j)}(z)$ be a linear polynomial with randomly-chosen coefficients. We hereafter suppress the argument z to simplify notation.

Consider the following sequence of homotopies

$$\begin{bmatrix} L_1^{(1)} \\ L_2^{(1)} \\ \vdots \\ L_N^{(1)} \end{bmatrix} \rightarrow \begin{bmatrix} L_1^{(2)} \\ L_2^{(1)} \\ \vdots \\ L_N^{(1)} \end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix} L_1^{(d_1)} \\ L_2^{(1)} \\ \vdots \\ L_N^{(1)} \end{bmatrix}$$

This use of homotopies for solving systems of linear equations is unnecessary; direct solving is certainly adequate. We chose to use homotopies here to illustrate the approach from the very first equation. Also, note that the homotopy function need only depend on t in the first line as the others do not change.

Let $S_1, \ldots S_{d_1}$ be the resulting sets of *nonsingular* solutions to each system in the series of homotopies above. Notice that only the nonsingular solutions are included in $S_1 \ldots S_{d_1}$ later in this section we describe the reason.

Let $S = \bigcup_{j=1}^{d_1} S_j$. This is clearly the set of all isolated solutions of the system

$$\begin{bmatrix} \prod_{j=1}^{d_1} L_1^{(j)} \\ L_2^{(1)} \\ \vdots \\ L_N^{(1)} \end{bmatrix},$$

for which we note that the first equation now has the same degree as $f_1(z)$.

The *regeneration* of f_1 is completed with the following homotopy:

$$\begin{bmatrix} \prod_{j=1}^{d_1} L_j^{(1)} \\ L_2^{(1)} \\ \vdots \\ L_N^{(1)} \end{bmatrix} \rightarrow \begin{bmatrix} f_1 \\ L_2^{(1)} \\ \vdots \\ L_N^{(1)} \end{bmatrix}.$$

It is important to note that homotopy paths starting at singular points cannot be tracked with simple predictor-corrector methods. It is precisely for this reason that S_i, defined above, may contain only *nonsingular* points. Deflation could be used to desingularize any singular start points, but this is a costly approach. See [19, 20] for more information on deflation in the setting of polynomials.

Once $f_1(z)$ has been regenerated, we regenerate $f_2(z)$ similarly, and so on. For more details see [14]. The final result is the set of all nonsingular isolated solutions in $\mathbf{V}(f)$, along with any singular solutions that come from paths that stay nonsingular until the final homotopy to $f_N(z)$. In chapter 2, an adaption to regeneration is presented that ensures that all isolated (both singular and nonsingular) solutions of a system are found.

1.3. The Parameter Homotopy

There are many different ways to construct a homotopy that exploits some sort of special structure in the equations, including multihomogenous homotopies [4] and regeneration homotopies (see §1.2). If one desires to solve multiple systems that differ only in the parameters that define them, another approach to increasing efficiency is the parameter homotopy.

Given a family of polynomial systems $f(z;q)$ in the variables $z \in \mathbb{C}^N$, which depends on the parameters $q \in \mathbb{C}^k$, the idea of a parameter homotopy is to solve $f(z;\hat{q}) = 0$ where

$\hat{q} \in \mathbb{C}^k$ is chosen at random. This exploits the fact that for a generic choice of parameters the number of nonsingular isolated solutions is constant. Since the set of nongeneric parameters is a proper algebraic subset of $\mathbb{C}^k$, a random choice of $\hat{q} \in \mathbb{C}^k$ is generic with probability one. If we know the solutions to $f(z; \hat{q}) = 0$, they can be used as start points in the basic homotopy between $f(z; \hat{q}) = 0$ and $f(z; q) = 0$ for any q. The start system is valid because the number of isolated solutions can only decrease for a nongeneric choice of parameters, so for any choice of q the start system $f(z; \hat{q}) = 0$ has a sufficient number of solutions.

The primary benefit of a parameter homotopy is that the system $f(z; \hat{q}) = 0$ often has fewer solutions then a total degree start system. So at the cost of computing solutions to $f(z; \hat{q}) = 0$, subsequent homotopies for approximation solutions at other points in parameter space require following fewer paths then a total degree homotopy. Additionally, there are other advanced refinements to the parameter homotopy in [4] that are useful when inequalities, nonpolynomial coefficients, or non-Euclidean parameter spaces are involved.

1.4. Positive Dimensional Solutions Sets

Solution components of dimension greater than zero arise in two ways. First, a system could be underdetermined, that is, there are more variables than equations. Second, a system could be square (same number of equations and variables) or overdetermined, but some equations are algebraically dependent on the others. An example of this is a cubic curve given by the generators:

$$f = \begin{bmatrix} xz - y^2 \\ y - z^2 \\ x - yz \end{bmatrix} = 0$$

More is said in Chapter 3 about the occurrence of these types of systems, but for now we focus on computing the *numerical irreducible decomposition* of a polynomial system with positive dimensional components.

1.4.1. Irreducible Decomposition. Recall that the *irreducible decomposition* of an algebraic set (solution set of a polynomial system) is the breaking up of the set into irreducible components by their dimension, that is

$$V(f) = \bigcup_{i=0}^{\dim V(f)} X_i = \bigcup_{i=0}^{\dim V(f)} \bigcup_{j \in \mathcal{J}_i} X_{i,j}$$

where $\mathcal{J}_i$ is an index set for the irreducible components in the pure i-dimensional set X_i.

Recalling that over the complex numbers a general linear subspace of codimension d will intersect a d-dimensional irreducible algebraic set in points (0-dimensional algebraic set), we define the numerical representation of an irreducible component, a *witness set* [30, 4]. A *witness set* for an irreducible component Z of dimension d is the triple

$$W = \{f, L, \mathcal{W}\}$$

where f is the set of generating equations for the algebraic set containing Z, L is the set of generating equations for a generic linear subspace of codimension d, and $\mathcal{W}$ is the set of points $\mathcal{V}(L) \cap Z$.

With the definition of a witness set in hand, we can then define the numerical analog of the irreducible decomposition. Let $W_i = \mathcal{L}_i \cap \mathcal{V}(f)$ where $\mathcal{L}_i$ is an i codimensional generic linear subspace defined by linear polynomials L_i, the the *numerical irreducible decomposition (NID)* is given by

$$V(f) = \bigcup_{i=0}^{\dim V(f)} W_i = \bigcup_{i=0}^{\dim V(f)} \bigcup_{j \in \mathcal{J}_i} W_{i,j}$$

where $W_{i,j} = \{f, L_i, \mathcal{W}_{i,j}\}$ is the witness set corresponding to irreducible component $Z_{i,j}$ [30, 4]. W_i is sometimes called the pure i dimensional witness set and is a subset of the *witness superset* for the i^{th} dimension, which is the topic of subsection 1.4.3. Before addressing witness supersets we review some important operations in numerical algebraic geometry.

1.4.2. Linear Slicing and Slice Moving. Linear systems play an importation role in the setting of positive dimensional solutions sets. When a linear system L is appended to a system f to form the new system:

$$g = \begin{bmatrix} f \\ L \end{bmatrix}$$

it is often said that $\mathcal{V}(f)$ has been *sliced* by $\mathcal{V}(L)$. Alternatively, the construction of g is sometimes referred to as *linear slicing*. Many times it is necessary to move one linear slice L to another linear slice L', which is called *slice moving*. The following homotopy accomplishes this task:

$$H(z,t) = \begin{bmatrix} f \\ Lt + L'(1-t) \end{bmatrix}$$

Linear slicing and slice moving are important operations in numerical algebraic geometry. One of the primary applications of slice moving is the *membership test* [30, 4]. Suppose that $W = \{f, L, \mathcal{W}\}$ is a witness set for component Z, then the membership test can be used to determine if a point p is on component Z using slice moving. First one writes down a linear system that includes p as a solution, such as $A(z-p)$, where A is a random matrix such that the dimension of the linear subspace $A(z-p) = 0$ is equal to $\dim(\mathcal{V}(L))$. Then the homotopy above can be used to move the linear subspace $\mathcal{V}(L)$ to the linear subspace $A(z-p) = 0$:

$$H(z,t) = \begin{bmatrix} f(z) \\ L(z)t + A(z-p)(1-t) \end{bmatrix}$$

The witness points $\mathcal{W}$ are the start points for the numerical homotopy. If the point p is among the endpoints of the homotopy then $p \in Z$ [4]. The membership test will play an important role in the next section, but now we return to computing witness supersets and witness sets.

1.4.3. Witness Supersets. In the setting of positive dimensional solutions sets, the fundamental goal of numerical algebraic geometry is to compute the numerical irreducible decomposition. Given the definition of the numerical irreducible decomposition, it is natural to start by computing the pure i dimensional witness set. Notice that W_i is a subset of $\mathcal{L}_i \cap \mathcal{V}(f)$, where $\mathcal{L}_i$ is a linear subspace of codimension i and equality fails because $\mathcal{L}_i$ can intersect any components in $\mathcal{V}(f)$ with dimension greater than i. These unwanted intersections create the so called junk points of the witness superset.

Let $\widehat{W}_i := \mathcal{L}_i \cap \mathcal{V}(f)$ be the witness superset for dimension i and notice that for a system $f : \mathbb{C}^N \to \mathbb{C}^n$ the witness point superset $\widehat{W}_{N-1}$ would not contain any junk points, as $N-1$ is the largest possible dimension of a nontrivial irreducible component of $\mathcal{V}(f)$. Additionally, $\widehat{W}_{N-2}$ will only have junk points arising from components of dimension $N-1$, thus if we could identify such points we could eliminate the junk points of $\widehat{W}_{N_2}$. One way to accomplish this is using membership testing. Suppose we already have a witness sets for the $N-1$ dimensional irreducible components, then a membership test can be used to determine if any points in $\widehat{W}_{N-2}$ lie on components in dimension $N-1$, thus identifying the junk points. Another option is a tool called the local dimension test. We refer the reader to [2] for more information on the local dimension test.

In summary, if we have witness sets for all components of dimension greater then i, we can remove the junk from the witness superset to get W_i. The next step is to decompose W_i into witness point sets $\mathcal{W}_{i,j}$ for each i dimensional irreducible component. This is the topic of the next section.

1.4.4. Supersets to Witness Sets. All that remains in producing a numerical irreducible decomposition is to decompose each i dimensional superset into witness sets for the components of dimension i. There are two tools for this process, the trace test and monodromy. We outline each of these here, but point the reader to [30] for the details.

1.4.4.1. *Monodromy Test.* Suppose that we have a closed loop L parameterized by $t \in [0, 1]$, in the space of codimension i linear subspaces that intersect an irreducible component Z. Let $\mathcal{L}(k)$ be the linear system corresponding to $L(k)$, then $Z \cap \mathcal{V}(\mathcal{L}(1)) = Z \cap \mathcal{V}(\mathcal{L}(0))$. If we use a homotopy to move the set of points $Z \cap \mathcal{V}(\mathcal{L}(1))$ around the loop L to $Z \cap \mathcal{V}(\mathcal{L}(0))$ we get the same points, but possibly in a permuted order [30, 4]. This fact can be exploited to create a decomposition.

Consider moving the generic linear subspace $\mathcal{L}_i$ which is used to define the pure i dimensional witness set for the i^{th} dimension around a loop as described in the previous paragraph. The points in $W_i = \mathcal{L}_i \cap \mathcal{V}(f)$ are brought back to points in the set, except possibly in a permuted order. Any points in the same permutation orbit are members of the same component. Thus monodromy loops can be used to decompose each pure i dimensional witness superset into witness sets for each i dimensional component.

1.4.4.2. *Trace Test.* In practice the monodromy test works well, but there is not a guarantee that the monodromy action will be observed for a particular loop and we don't have a stopping criterion from monodromy alone; thus another test is necessary. To each point in W_i a value called the trace can be assigned [30]. Traces have the property that if $z_1, \ldots, z_l$

are witness points for component Z, then $\sum tr(z_i) = 0$, but $tr(z_i) \neq 0$ for all i. Thus one option for decomposing is to look for combinations of the points in W_i whose traces sum to zero. The trouble with this approach is in the combinatorial growth in searching the space of all sets of subsets of W_i. However, unlike the monodromy test, this test is guaranteed to terminate with a solution.

In practice a combination of these methods is usually used. For example monodromy loops can be used to group some of the points in W_i, then the traces can be summed and the remaining points can be searched for traces that when added to the sum make it zero.

1.4.5. Other Approaches. The above description of computing an NID is called the dimension-by-dimension approach. Other approaches to create an NID are more efficient. These include the witness cascade [4, 30] and regeneration cascade [4, 15]. We point the reader to the references for a description of these methods.

1.5. Non-square systems

For a system of polynomial equations $f : \mathbb{C}^n \to \mathbb{C}^N$ we already remarked above that if $n > N$, then there are positive dimensional solution components. Thus the homotopies of the previous section would be a good choice for solving these systems. However we have not dealt with the situation where $N > n$. This poses a problem since all of our homotopies are written for square systems.

If $N > n$, the standard technique in numerical algebraic geometry is to randomize $f(z)$ down to $A \cdot f(z)$, where $A \in \mathbb{C}^{N \times n}$ is a randomly chosen matrix of complex numbers [4, 30]. This yields a square system of n equations and variables, with the property that $\mathbf{V}(f(z)) \subset \mathbf{V}(A \cdot f(z))$. We also have that isolated solutions in $\mathbf{V}(f(z))$ will be isolated solutions in $\mathbf{V}(A \cdot f(z))$, though the multiplicity of solutions having multiplicity greater than one (with

respect to $f(z)$) might increase. It is easy to filter any "new" isolated solutions of $\mathbf{V}(A \cdot f(z))$ that are not actually solutions of $f(z) = 0$, simply by evaluating each isolated solution in the polynomials $f(z)$.

Thus, the homotopy algorithms for square systems can easily be extended to find all isolated solutions of non-square polynomial systems. Of course, with the exception that the computed multiplicity of a solution of a non-square system, that has been randomized down to a square system, might be larger than the multiplicity of that solution with respect to the original system.

CHAPTER 2

Perturbed Homotopies

In this chapter[1] we consider the use of perturbed homotopies for solving polynomial systems. Specifically, we propose first solving a perturbed version of the polynomial system, followed by a parameter homotopy to remove the perturbation. We demonstrate that perturbed homotopies are sometimes more efficient than regular homotopies, though they can also be less efficient. A useful consequence is the application of this perturbation to regeneration. Recall from the previous chapter, that the basic form of regeneration fails to find approximations to singular solutions. However, a perturbed version of regeneration – *perturbed regeneration* – will yield all isolated solutions, including all singular isolated solutions. This method can decrease the efficiency of regeneration, but increases its applicability.

A very simple example illustrates how regeneration can fail to find singular solutions. Consider the following polynomial system of equations:

$$\begin{bmatrix} y(x-2)^2 \\ x(y-3) \end{bmatrix}$$

It is easy to see that this system has isolated solutions $(0,0)$ and $(2,3)$. The solution $(2,3)$ is singular, with multiplicity two.

Consider the system after regenerating the first equation:

$$\begin{bmatrix} y(x-2)^2 \\ r_1 x + r_2 y + r_3 \end{bmatrix}$$

[1] A version of this chapter was open access published as [1]. We reproduce the content here, but omit background details that are covered in other parts of this dissertation.

where $r_1, r_2, r_3 \in \mathbb{C}$ are random. This system has nonsingular solution $(-r_3/r_1, 0)$ and singular solution $(2, -(2r_1+r_3)/r_2))$. In practice, as in the current implementation in Bertini [3], this singular solution is discarded before moving on, leaving us to follow only the path originating from $(0, 0)$.

Proceeding in the regeneration algorithm, we follow the homotopy

$$\begin{bmatrix} y(x-2)^2 \\ r_1x + r_2y + r_3 \end{bmatrix} \rightarrow \begin{bmatrix} y(x-2)^2 \\ s_1x + s_2y + s_3 \end{bmatrix}$$

where $s_1, s_2, s_3 \in \mathbb{C}$ are random. Finally, we complete regeneration via

$$\begin{bmatrix} y(x-2)^2 \\ (r_1x + r_2y + r_3)(s_1x + s_2y + s_3) \end{bmatrix} \rightarrow \begin{bmatrix} y(x-2)^2 \\ x(y-3) \end{bmatrix}$$

to arrive at only the nonsingular solution $(0, 0)$.

In this example, regeneration fails to find the singular solution; this is an unfortunate drawback to regeneration, since it is such an efficient technique. The authors of [14] provide a solution to this problem. Their solution is rooted in deflation [27, 28, 21, 20]. Unfortunately, deflation is costly and involves randomization to a square system, which can destroy the monomial structure of the problem and thus increase run time. Thus, we are led to pose the following:

Fundamental Problem 1: Modify regeneration to find a numerical approximation of each isolated point of $V(f)$, including isolated singular solutions.

In this chapter, we provide a solution to Fundamental Problem 1. There are two simple steps:

(1) Find the isolated solutions of a perturbation $\hat{f}(z)$ of $f(z)$.

(2) Solve $f(z)$ by tracking the solutions as we deform from $\hat{f}(z)$ back to $f(z)$.

This method is the focus of Section 2.1 and is the main contribution of this chapter.

While the principal goal of this chapter is to describe a variation of regeneration, it is worth noting that this sort of perturbation can also be paired with homotopy methods other than regeneration, such as total degree and multihomogeneous homotopies. This leads to the following:

Fundamental Problem 2: What are the costs and benefits of perturbation when paired with various standard homotopy methods?

We will not provide a thorough investigation and complete solution of Fundamental Problem 2. Rather, we present some examples and timings that could provide a starting point for a more thorough investigation of this problem.

There are certainly other approaches to dealing with singular solutions. In §2.4, we describe the connection of our perturbation approach to the deflation approach of [14], the method of regenerative cascade [15], and a very early technique in the field known as the cheater's homotopy [23] in which the authors made use of a perturbation of $\hat{f}(z)$ for somewhat different reasons. It is important to note that our perturbation is virtually the same as the cheater's homotopy, in the case where there are no parameters.

It is observed in [23] and [30] that a perturbation can cause positive-dimensional irreducible components to "break" into a (possibly very large) number of isolated solutions.

In §2.5, we investigate this phenomenon and describe our attempts to extract from it useful information.

2.1. Algorithm

We remedy the problem of basic regeneration failing to find the singular solutions of $f(z) = 0$ by replacing $f(z)$ with polynomial system $\hat{f}(z) = f(z) - y$ for a randomly chosen point $y \in \mathbb{C}^N$. It may seem surprising that this nearly trivial change to $f(z)$ could significantly alter the behavior of the solutions, but the result of this perturbation is that the singular solutions of $f(z)$ each become several isolated nonsingular solutions of $\hat{f}(z)$.

Before we describe the the theory underlying this approach, we present the pseudocode for the main algorithm of this chapter:

We state the following algorithm in more generality than is needed for Fundamental Problem 1 (regeneration only) since this version also works for Fundamental Problem 2.

Algorithm 1 Main Algorithm: Perturbed Homotopies

Input: Polynomial system $f : \mathbb{C}^N \to \mathbb{C}^N$.
Output: Superset $\widehat{V}$ of all isolated solutions V of $f(z)$ or, optionally, V.

1. Choose random $p \in \mathbb{C}^N$.
2. Use a homotopy method (e.g., regeneration, a total degree homotopy, or a multihomogeneous homotopy) to find all isolated nonsingular solutions T of $f_p(z) = f(z) - p$.
3. Follow all paths beginning at points of T through the parameter homotopy $f(z) - tp$, letting t go from 1 to 0, storing all finite endpoints in $\widehat{V}$.
4. (Optional) Remove from $\widehat{V}$ all non-isolated $z \in \widehat{V}$ via a local dimension test to produce V. There are several options for this local dimension test. One standard choice was first described in [2].

Naturally, any technique for optimizing a homotopy method will also reduce run time for the related perturbed homotopy method. For example, ordering of the polynomials by degree has an impact on run time for regeneration, so the same can be said in the perturbed case. This and other optimizations of regeneration are described in [14].

In the third step, one very special type of homotopy is used, the *parameter homotopy* [23, 24]. Given a parametrized polynomial system $f(z;q)$ with variables $z \in \mathbb{C}^N$ and parameters $q \in Q \subset \mathbb{C}^k$ for some parameter space Q, the idea of a parameter homotopy is to solve $f(z;q_0) = 0$ at a random, complex $q_0 \in Q$, then move (via a homotopy only in the parameters) from q_0 to any particular point $q = q' \in Q$ of interest. With probability one, $q = q_0$ will have the maximum number of nonsingular isolated solutions for all $q \in Q$, so there will be at least one path leading to each isolated solution at $q = q'$. In general, the benefit of using a parameter homotopy is that it involves one potentially costly run up front (at $q = q_0$), followed by one or more relatively inexpensive runs (at each $q = q'$ of interest). In this algorithm, we use exactly one parameter homotopy, to remove the perturbation, as described in step three.

In the fourth step a local dimension test is used to remove the non-isolated points of the system, that is points that live on positive dimensional components. There are several options for this local dimension test but, one standard choice was first described in [2].

2.2. Justification

Much of the theory underlying the ideas of this chapter is well known and has since been repeated in various forms, for example in [12, 30]. The main contribution of this chapter is in the application of this theory in the setting of regeneration, not in the theory itself. In this section, we provide justification for the correctness of the algorithm, pointing to appropriate sources for proofs and further background.

Theorem 2.2.1. *For a polynomial system $f : \mathbb{C}^N \to \mathbb{C}^N$ with $rk(f) = N$, with probability one, the procedure described by Algorithm 1 produces as output a superset of numerical approximations to all isolated solutions of $f(z) = 0$.*

Let $\mathrm{rk}(f)$ denote the rank of the polynomial system $f(z)$, i.e., the dimension of the closure of the image of $f(z)$, $\overline{f(\mathbb{C}^N)} \subseteq \mathbb{C}^N$. The rank of $f(z)$ is an upper bound on the codimension of the irreducible components of $\mathbf{V}(f)$ [30]. Thus, $f(z) = 0$ may only have isolated solutions if $\mathrm{rk}(f) = N$.

There are three key facts supporting Thereom 2.2.1:

LEMMA 2.2.2. *Given a polynomial system as in Theorem 2.2.1, there is a Zariski open set of complex numbers $p \in \mathbb{C}$ for which the solution set of $f - p$ consists of only smooth irreducible components of dimension $N - rk(f)$. In the case that $rk(f) = N$, $f - p$ will have only nonsingular isolated solutions.*

LEMMA 2.2.3. *In the case that $rk(f) = N$, as $p \to 0$ with $p \in f(\mathbb{C}^N)$, there is at least one path starting at a solution of $f - p$ leading to each isolated solution of $f(z) = 0$.*

LEMMA 2.2.4. *If $rk(f) = N$, then f is a dominant map, i.e., $\overline{f(\mathbb{C}^N)} = \mathbb{C}^N$.*

Before discussing the justification of these lemmas, we provide a simple proof of the main result, Theorem 2.2.1.

PROOF OF THEOREM 2.2.1. The statement is vacuously true if there are no isolated solutions, so we assume $\mathrm{rk}(f) = N$. According to Lemma 2.2.2, almost all perturbations $p \in \mathbb{C}$ of $f(z)$ will result in a polynomial system having only smooth isolated solutions. For some specific choice $\hat{p} \in \mathbb{C}$, we refer to this set of solutions as $\mathbf{V}(f - \hat{p})$. Regeneration can compute all of these nonsingular solutions [14].

Lemma 2.2.3 then guarantees that, for each isolated solution q of $f(z) = 0$, there is a homotopy path beginning from some $p \in \mathbf{V}(f - \hat{p})$ that ends at q, so long as $p = t\hat{p}$ stays in the image of f as t moves from 1 to 0.

Lemma 3 removes this final condition, so we may move freely from almost any $\hat{p} \in \mathbb{C}$ directly to 0 without worrying about staying in the image. Thus, all isolated solutions of $f(z) = 0$, including singular isolated solutions, will be produced as output by Algorithm 1. □

REMARK 2.2.5. *Note that we find only a superset of the isolated solutions of $f(z) = 0$, not the set itself. This is because points on positive-dimensional components may also be found by Algorithm 1. As mentioned in subsection 1.4.3, there are known methods for removing such points, if desired.*

Generalizations of all three lemmas appear in Appendix A of [30] as consequences of an algebraic version of Sard's Theorem. Indeed, Lemma 2.2.2 is proved as Theorem A.6.1 in [30]. Similarly, Lemma 2.2.3 is proven in more generality as Corollary A.4.19 of [30].

A more general statement than Lemma 2.2.4 is given as an exercise in [11] and a related result for a pure d-dimensional algebraic subset is presented in [12], for $d > 0$. For the specific setting of the lemma, the proof is trivial:

PROOF OF LEMMA 2.2.4. Since $\mathbf{V}(f)$ contains a pure 0-dimensional algebraic subset, we must have that the rank of f is N. So f is full rank and equivalently f is dominant. □

Now that we have completed the justification of Theorem 2.2.1, we may discuss a few extensions.

First, we may trivially compute the multiplicity, $\mu(z_i)$, of each isolated solution z_i of $f(z) = 0$, as defined in [30]:

COROLLARY 2.2.6. *Algorithm 1 produces not only the isolated solutions of $f(z) = 0$ but also the multiplicity $\mu(z_i)$ of each solution z_i.*

This is based on the fact, proved as Theorem A.14.1(3) in [30], that each isolated solution z_i will be the endpoint of $\mu(z_i)$ paths beginning at points in $\mathbf{V}(f - \hat{p})$.

Second, we may consider the case of non-square system $f : \mathbb{C}^N \to \mathbb{C}^n$, with $N \neq n$. Of course, if $n < N$, there are no isolated solutions, so a homotopy for finding witness sets of positive dimension components would be a better approach.

If $N > n$, the standard technique in numerical algebraic geometry is to randomize $f(z)$ down to a square system, as discussed in Section 1.5. Thus, Algorithm 1 can easily be extended to find all isolated solutions of non-square polynomial systems. Of course, as discussed in Section 1.5 the computed multiplicity of a solution of a non-square system that has been randomized down to a square system might be larger than the multiplicity of that solution with respect to the original system.

2.3. Examples

In this section we consider several examples where perturbed regeneration provides some benefit and give timings for other types of perturbed homotopies. All runs made use of Bertini 1.4 [3]. All reported timings except those of the last example come from runs on a 3.2 GHz core of a Dell Precision Workstation with 12 GB of memory. The last example, the nine point problem, used 145 2.67 GHz Xeon 5650 cores (144 workers).

2.3.1. A very simple illustrative example.

Let us first consider the simple example from the introduction of this chapter to illustrate Algorithm 1. Recall the system,

$$f(x, y) = \begin{bmatrix} y(x-2)^2 \\ \\ x(y-3) \end{bmatrix}$$

which had isolated solutions $(0,0)$ and $(2,3)$. Basic regeneration without deflation will not find the solution $(2,3)$ because it is singular with multiplicity 2.

For perturbed regeneration, we first solve the perturbed system,

$$f_p(x,y) = \begin{bmatrix} y(x-2)^2 - p_1 \\ x(y-3) - p_2 \end{bmatrix},$$

using basic regeneration, where $p = (p_1, p_2) \in \mathbb{C}^2$ is chosen randomly. Suppose $p = (-0.521957 + 0.810510i, -0.0312394 - 0.602051i)$, then the perturbed system $f_p(x,y)$ has three solutions, approximated as:

$$(x,y) = (2.2895552262 - 0.48184726973i, 3.0399274146 - 0.25455256993i)$$

$$(x,y) = (1.6965408998 + 0.48949083868i, 2.8884817358 - 0.32269420324i)$$

$$(x,y) = (0.0243170205 + 0.19304012524i, -0.090137930 - 0.22743203050i)$$

Then we use the homotopy

$$h(x,y;t) = \begin{bmatrix} y(x-2)^2 - tp_1 \\ x(y-3) - tp_2 \end{bmatrix}$$

to deform the solutions to $f_p(x,y)$ to solutions of $f(x,y)$. Two solutions above converge to $(2,3)$; the other converges to $(0,0)$. Thus, perturbed regeneration finds the solution missed by basic regeneration (excluding deflation).

2.3.2. An example with several isolated singular solutions. Next, we consider the system cpdm5, from the repository of systems [32] but originally considered in [9]. This system has five equations and five variables, with solutions as described in Table 2.1. The 5 singular solutions each have multiplicity 11.

TABLE 2.1. Basic properties of the cpdm5 solutions.

	Real Solutions	Non-real Solutions	Total
Non-singular	38	120	158
Singular	5	0	5
Total	43	120	163

Here, again, basic regeneration (without deflation) missed all of the singular solutions. Timings for regular and perturbed total degree and regular and perturbed regeneration are provided in Table 2.2. It is perhaps interesting to note that the timings for the perturbed runs (regeneration or total degree) vary much less than those of the unperturbed runs, as indicated by the standard deviation in the table (column 5).

TABLE 2.2. Run times for the cpdm5 problem. Each timing is an average over 100 runs.

	Computation Time (s)				**Paths Tracked**		
Method	**Step1**	**Step 2**	**Total**	**Std Dev**	**Step 1**	**Step 2**	**Total**
perturbed regeneration	2.3	1.2	3.6	0.2	363	213	576
perturbed total degree	0.7	1.2	1.9	0.2	243	213	456
regeneration	-	-	4.3	0.9	-	-	363
total degree	-	-	1.9	0.8	-	-	243

A priori, users may wish to use regeneration. While most examples of this section show that perturbed regeneration should be used instead, this example further shows that total degree (or perturbed total degree) can sometimes be faster.

2.3.3. AN EXAMPLE WITH MANY SINGULAR ISOLATED SOLUTIONS. In the article [26], Morrison and Swinarski study a polynomial system with 13 equations, having 51 isolated solutions. All of these solutions are singular, 30 with multiplicity 2, 20 with multiplicity 8, and one with multiplicity 32.

Basic regeneration failed to find any of the solutions, but perturbed regeneration found all of them. Some timings are provided in Table 2.3.

TABLE 2.3. Run times for the Morrison-Swinarski problem. Each timing is an average over 100 runs.

	Computation Time (s)				Paths Tracked		
Method	**Step1**	**Step 2**	**Total**	**Std Dev**	**Step 1**	**Step 2**	**Total**
perturbed regeneration	22.8	2.6	25.4	2.7	2560	252	2812
perturbed total degree	6.7	2.7	9.4	2.8	1024	252	1276
perturbed 2-hom	2.9	2.9	5.9	1.9	252	252	504
regeneration	-	-	31.2	8.5	-	-	2560
total degree	-	-	12.4	6.5	-	-	1024
2-hom	-	-	12.4	7.7	-	-	252

Here again, while perturbed regeneration finds all the solutions, it is not the most efficient method. As with the previous example, total degree (and perturbed total degree) are more efficient. A more specialized sort of homotopy, the 2-homogeneous homotopy [30], performs even better in this case. Note that basic regeneration both misses all solutions and takes the longest. This is probably due to the fact that singular solutions are discovered (which is costly), then discarded at various regeneration levels. Again, the standard deviations for the perturbed homotopy methods are lower in this case.

2.3.4. THE BUTCHER PROBLEM: POSITIVE-DIMENSIONAL COMPONENTS. Finally, we consider the following system, originally due to C. Butcher [7],

$$f = \begin{bmatrix} zu + yv + tw - w^2 - 1/2w - 1/2 \\ zu^2 + yv^2 - tw^2 + w^3 + w^2 - 1/3t + 4/3w \\ xzv - tw^2 + w^3 - 1/2tw + w^2 - 1/6t + 2/3w \\ zu^3 + yv^3 + tw^3 - w^4 - 3/2w^3 + tw - 5/2w^2 - 1/4w - 1/4 \\ xzuv + tw^3 - w^4 + 1/2tw^2 - 3/2w^3 + 1/2tw - 7/4w^2 - 3/8w - 1/8 \\ xzv^2 + tw^3 - w^4 + tw^2 - 3/2w^3 + 2/3tw - 7/6w^2 - 1/12w - 1/12 \\ -tw^3 + w^4 - tw^2 + 3/2w^3 - 1/3tw + 13/12w^2 + 7/24w + 1/24 \end{bmatrix},$$

which appeared in [32]. Computing the numerical irreducible decomposition [4, 30], the solution set consists of 10 irreducible components of various dimensions, provided in Table 2.4. All isolated solutions are nonsingular.

TABLE 2.4. Summary of the solution set of the Butcher problem.

Dimension	Components	Degree
3	3	1
2	2	1
0	5	1

When basic regeneration is applied to this system, only the five nonsingular points are approximated. Thus, in this case, basic regeneration finds all isolated solutions. If perturbed regeneration is applied, there are eleven nonsingular points in the solution set of the perturbed system. Five points go to nonsingular solutions of the original problem, two points converge to the positive-dimensional components[2], and the remaining paths diverge. Note that a total degree homotopy will also find points on the positive-dimensional components, but computation time increases, since hundreds of points converge to the positive-dimensional components.

TABLE 2.5. Run times for the Butcher problem. Each timing is an average over 100 runs, except perturbed total degree and total degree, which are averages over 50 runs.

	Computation Time (s)				**Paths Tracked**		
Method	**Step1**	**Step 2**	**Total**	**Std Dev**	**Step 1**	**Step 2**	**Total**
perturbed regeneration	32.4	0.5	32.9	7.5	982	11	993
perturbed total degree	663.4	0.5	663.8	113.4	4608	11	4619
regeneration	-	-	41.0	15.3	-	-	838
total degree	-	-	1106.0	158.3	-	-	4608
regenerative cascade	-	-	117.4	70.1	-	-	1414

[2]Having run this several times, it seems clear that these two points always land on one specific 3-dimensional component. These points vary for different runs, meaning they land at generic points on that irreducible component and are therefore not at intersection points between the components as one might expect. As further evidence, TrackType 6 of Bertini, using isosingular deflation [17] show that these two points are in fact smooth points on this irreducible component.

Table 2.5 shows the timings for this problem using a total degree homotopy, regenerative cascade, basic regeneration, and perturbed regeneration. As opposed to the previous examples, perturbed regeneration was faster for this problem, even faster than the shorter basic regeneration method. This is due to the fact that basic regeneration encounters singular solutions on positive-dimensional components throughout the algorithm, which slows down the path tracker; the perturbed system has only nonsingular isolated solutions, which can be handled much more efficiently.

2.3.5. A LARGE EXAMPLE FROM AN APPLICATION. As a final example, we consider the nine-point four-bar design problem, exactly as formulated in Chapter 5 of [4]. This eight polynomial, eight variable system has total degree 7^8 =5,764,901, a 2-homogeneous root count of 4,587,520, and a 4-homogeneous root count of 645,120. There are 8652 nonsingular, isolated solutions and a number of positive-dimensional components.

Using precisely the Bertini settings described on the examples page for [4] for all runs, we find that regeneration is fastest, followed by perturbed regeneration. All other homotopy types (perturbed or not) were cost prohibitive, taking at least twice as long as perturbed regeneration. The timings are summarized in Table 2.6. As discussed in §2.5, the positive-dimensional components are ignored by basic regeneration but result in many more paths to follow for perturbed regeneration, at least partially explaining the difference in timings.

Note that the number of paths is not reported in the table as there were some path failures during the runs. One could change the configurations (independently for each run type) to get all paths to converge, though that would destroy the direct comparison of the various methods as configuration changes affect run times. Generally speaking, there were about 290,000 paths for perturbed regeneration runs and about 175,000 paths for basic regeneration.

TABLE 2.6. Run times for the nine point problem. Each timing is an average over 10 runs.

	Computation Time			
Method	**Step1**	**Step 2**	**Total**	**Std Dev**
perturbed regeneration	$2h18m19s$	$1m19s$	$2h19m38s$	$42m1s$
perturbed total degree	$> 6h$	-	$> 6h$	-
perturbed 2-hom	$> 6h$	-	$> 6h$	-
perturbed 4-hom	$> 6h$	-	$> 6h$	-
regeneration	-	-	46m 53s	24m 12 s
total degree	-	-	$> 6h$	-
2-hom	-	-	$> 6h$	-
4-hom	-	-	$> 6h$	-

In this case, perturbed regeneration does not add any value beyond basic regeneration (since all isolated solutions are known to be nonsingular), but a user with no *a priori* knowledge of the solutions would be best served using perturbed regeneration in case there might be singular isolated solutions.

2.4. Techniques related to perturbed regeneration

As mentioned above, the theory behind perturbed homotopies is not new and other alternatives to perturbed regeneration exist for using regeneration to find isolated singular solutions. Perhaps the earliest reference to this sort of perturbation for homotopy methods was the cheater's homotopy, described briefly in §2.4.3. In this section, we very briefly describe these related techniques and indicate the differences between them and perturbed regeneration.

2.4.1. Regeneration with deflation. As outlined in [14], regeneration techniques can be combined with deflation to find singularities. Deflation is a technique that replaces a polynomial system f on $\mathbb{C}^N$ and an isolated singular solution x^* with a new polynomial system $f(x, \xi)$ on $\mathbb{C}^N \times \mathbb{C}^M$ with an isolated nonsingular soluition (x^*, ξ^*). For more on

deflation in the polynomial setting see [20, 21] and in a more general context [27, 28]. A more recent and more general version of deflation is called isosingular deflation [17]. To find singularities using regeneration, deflation must be applied to any intermediate system that has a singular solution. Each application of deflation increases the number of variables and equations in the system, thus making computation more difficult. Algorithm 1 will find isolated singular solutions while avoiding intermediate deflation steps.

2.4.2. Regenerative cascade. The regenerative cascade of [15] provides an equation-by-equation approach to computing the numerical irreducible decomposition of the solution set of a polynomial system. A consequence of this method is that isolated singular solutions of the system will also be identified. However, if only isolated solutions are of interest, this information comes at a significant cost increase, namely the cost of cascading through a number of dimensions. Perturbed regeneration avoids this cost, but if the complete information provided by a numerical irreducible decomposition is desired, the regenerative cascade is clearly the better choice.

2.4.3. The cheater's homotopy. Parameterized polynomial systems $f(v, p)$ arise with some frequency in applications, so it is sometimes useful to solve the same polynomial system at numerous points $p = p_1, \ldots, p_k$ in some parameter space. Parameter homotopies are the right tool for this job. This idea has been implemented in *Bertini* [3] and *Paramotopy* [6]. Some background may be found in [23] and [24].

The trick to such methods is choosing an intermediate system $f(v, \hat{p})$ which satisfies some necessary properties, including that the solutions are smooth. The cheaters homotopy in [23] addresses this issue by including the same perturbation parameter as in Lemma 2.2.2. In that case, the primary motivation for using such a perturbation is to have smooth solutions

as start points of another homotopy. We require the smooth solutions so that regeneration can be used to compute the approximations. Thus, the methods are quite similar but have different goals.

More explicitly, the cheater's homotopy seeks to solve the parametrized system $f(z;q) = 0$ by first solving $f(z;\widehat{q}) + b = 0$, where $\widehat{q}$ and b are random and complex. The solutions of this first solve are then used as start points for $f(z, t\widehat{q} + (1-t)q') + tb = 0$ as $t \to 0$ to arrive at the solution set at parameter values q'. Thus, the method of this paper is the same as the cheater's homotopy when $f(z;q)$ is just $f(z)$, i.e., when there are no parameters.

2.5. The effect of perturbation on positive-dimensional irreducible components

The focus of this chapter is the extension of basic regeneration to find all isolated solutions. However, it is natural to consider the effect of this method on positive-dimensional solution components. This issue has arisen previously [30], but there has never been a careful, thorough analysis. Unfortunately, there is actually rather little to conclude.

2.5.1. Failure to find the numerical irreducible decomposition.

As described in Chapter 1, there are a number of methods for computing the numerical irreducible decomposition of the solution set of a polynomial system. It is tempting to try to use the perturbation method of this article to compute such a decomposition in just one step. Given system $f(z)$, the idea would be to solve a perturbed system $\hat{f}(z)$ for which all irreducible components have been broken into points, then move from $\hat{f}(z)$ back to $f(z)$ with some number of points landing on each irreducible component. If desired, monodromy and the trace test [4, 29] could be used to find d_Z points on each component Z.

At first, there seems to be some hope for this. As described in [25], such a perturbation will lead to at least one point per connected (complex) component. Of course, with all of this, we must assume that f has full rank, or else the perturbation does not break positive-dimensional components into points.

However, a simple example shows that this method will not work in general. Let

$$f(x, y) = \begin{bmatrix} x^2 \\ \\ xy \end{bmatrix}.$$

The solution set of this polynomial system is just the line $x = 0$, with a singular embedded point at the origin. It is easy to see that the four points from any perturbed system will necessarily all go to the origin, not a generic point on $x = 0$.

This can similarly be seen from the Butcher problem in Section 2.3.4. The solution set for that problem consists of five positive-dimensional components, but the perturbation yields only two points on those five components.

Based on these examples, it seems that there is little hope of directly computing a numerical irreducible decomposition via this sort of perturbation. Perhaps there is some modification of the ideas of this chapter that will yield the numerical irreducible decomposition in a similar manner, but such a modification is not treated in this dissertation.

2.5.2. Extracting useful information. It would be interesting to know exactly how many points come from each component when breaking an algebraic set into points via perturbation. Of course, if $f(z)$ is not full rank, each component will break into some number of positive-dimensional components, possibly of lower dimension.

Unfortunately, given that there is not even a guarantee that there will be even one point per irreducible component, this is a moot point. The solution of Morgan and Sommese [25]

seems to be the best we can hope for – there is at least one point on each connected component – unless perhaps more conditions are added to the polynomial system. However, this is beyond the scope of this dissertation.

2.6. Conclusions

In this chapter, we presented a variation of regeneration that will yield all isolated solutions of a polynomial system, not just those that are nonsingular. We refer to this method as *perturbed regeneration*. We further applied such a perturbation to other basic homotopy methods, giving run times on several examples.

While this sort of perturbation is not new, it is new in the context of regeneration and provides a significant improvement to the output of basic regeneration for a modest increase in computational cost. While it would be interesting to better understand how this perturbation affects positive-dimensional irreducible components, the results of the previous section seem to indicate that there is little that can be said in general.

Although the examples in this article may make it seem that perturbed methods are usually faster than their unperturbed analogues, it is clear that unperturbed methods will sometimes (perhaps often) be more efficient. Indeed, given a problem with the total degree number of nonsingular, isolated solutions, an unperturbed total degree homotopy will follow exactly as many paths as the number of solutions whereas a perturbed total degree homotopy will follow twice that number. Since all paths will be smooth throughout such a run, a perturbed approach will clearly be slower.

One clear topic for future research is the automatic detection of which method to use for a given polynomial system. Of course, if the user knows the solutions already, then it is

easy to make a reasonable educated guess as to which method(s) will be fastest. For new problems, though, we can currently provide only a little guidance.

If the user expects positive-dimensional solution sets (or cannot preclude their presence), regenerative cascade [15] is typically the best bet. If only isolated solutions are of interest (or if it is known that there cannot be positive-dimensional solution sets), then there is no need to search for positive-dimensional solution sets with regenerative cascade and a standard or perturbed standard homotopy is a good choice. If singular isolated solutions are expected and of interest, basic regeneration (without deflation) would not be a good choice and this might be a good place for one of the perturbed methods of this paper. Regeneration (perturbed or not) seems to be a good option if the problem has some special structure, whereas a multihomogeneous homotopy may be a good option if the variables naturally fall into multiple groupings. For sparse problems, polyhedral methods [18, 22, 33] are an especially good option.

All told, these are just suggestions motivated by experience, and there are surely polynomial systems for which this is not optimal advice. Hopefully, software will one day detect which of the many homotopy methods is optimal for a given polynomial system, but we are simply not there yet. For now, the best course of action is to run the problem of interest through several methods simultaneously, killing all remaining processes once one terminates. With the ever-decreasing cost of processors, this approach is becoming increasingly reasonable.

CHAPTER 3

Finding Exceptional Sets via Fiber Products

Consider a family of polynomial systems $f(v;p)$ on $\mathbb{C}^{N+m}$, which depends on the parameters $p \in \mathbb{C}^m$. It is well known that the choice of $p \in \mathbb{C}^m$ can have a dramatic effect on the dimension of $\mathcal{V}(f)$. For example, given:

$$g(v;p) = \begin{bmatrix} p_1 v_2 - p_2 v_1^2 \\ p_3 v_3 - p_4 v_1^3 \\ p_5 v_3 - p_6 v_1 v_2 \end{bmatrix} = 0$$

the solution set $\mathcal{V}(g)$ is zero dimensional for a generic choice of parameters (i.e., for a randomly selected $p \in \mathbb{C}^m$). However for $p = (1,1,1,1,1,1)$, the dimension of $\mathcal{V}(g(v;p))$ is one (the solution set is a cubic curve). In this chapter, we consider the fundamental problem of identifying parameters for which $\dim \mathcal{V}(f(v;p))$ is greater than the $\dim \mathcal{V}(f(v;\hat{p}))$ for a generic $\hat{p} \in \mathbb{C}^m$. We call such sets of parameters *exceptional sets*.

A motivating application is the classification of over constrained mechanisms. The assembly of a mechanism can be modeled by a polynomial system where the parameters correspond to constants of a constructed mechanism (such as link lengths) and variables give the joint displacements. The goal is to identify parameters for which the corresponding mechanism has more degrees of freedom (greater range of motion) then a generic configuration of the mechanism. Algebraically, the question is the same as above, identify p such that $\dim \mathcal{V}(f)$ increases.

A simple example is the two link planar mechanism as depicted in Figure 3.1.
The mechanism is anchored to the plane by its triangular base and has two joints which allow the links (l_1 and l_2) to rotate 360 degrees. Letting θ_1 and θ_2 be the angle between the

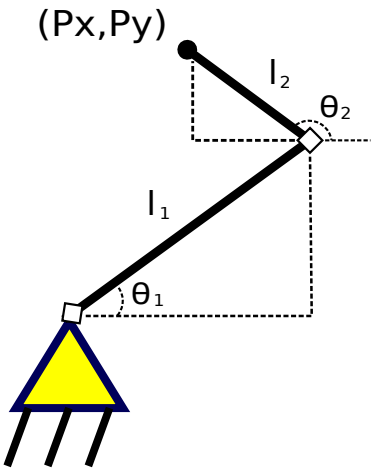


FIGURE 3.1. Two Link Mechanism

corresponding link and the horizontal axis, we can obtain the location of the end effector (black dot in the image above) using trigonometric relationships; that is, these equations solve the *forward kinematics problem* for the two link mechanism:

$$(P_x, P_y) = (\cos\theta_1 l_1 + \cos\theta_2 l_2, \sin\theta_1 l_1 + \sin\theta_2 l_2)$$

The forward kinematics problem is generally easy to solve; one of the strengths of numerical algebraic geometry is the ability to solve the *inverse kinematics*, which asks for the joint angles necessary to reach a given point in space. However, to use numerical algebraic geometry the work space must be modeled with polynomial equations. We can adapt the model of the workspace given above to a system of polynomial equations by thinking of $\cos(\theta_i)$ and $\sin(\theta_i)$ as the variables c_i and s_i and including additional equations that force the trigonometric relations:

$$F = \begin{cases} f_1 = c_1 l_1 + c_2 l_2 - p_x \\ f_2 = s_1 l_1 + s_2 l_2 - p_y \\ f_3 = c_1^2 + s_1^2 - 1 \\ f_4 = c_2^2 + s_2^2 - 1 \end{cases} \tag{1}$$

Consider the mechanism that corresponds with positive length links such that $l_1 \neq l_2$. Notice that the workspace is an annulus and that any point on the interior of the annulus can be reached in two ways. In other words, for the given l_1, l_2, P_x, and P_y, the polynomial system F has isolated solutions (note that it is easy to verify that the solutions are isolated over both $\mathbb{R}$ and $\mathbb{C}$). An example of an exceptional set occurs when $P_x = P_y = 0$ and $l_1 = l_2$. Under these conditions the end effector is placed over the top of the base, and the angle of θ_1 is free, meaning that any choice of θ_1 will still keep the end effector over the top of the base. Thus the solution set of F for this choice of parameters is one dimensional (an exceptional set, since for a generic choice of l_1, l_2, P_x, and P_y, the solution set is zero dimensional).

We address the question of identifying exceptional sets via fiber products and numerical algebraic geometry. This approach was first used by Sommese and Wampler in [31]. Our primary contribution is a new method for constructing fiber products, which relies on a recent advancement in numerically algebraic geometry, regeneration extension (a generalized approach to computing intersections of algebraic sets). Much of the theory that underpins our work comes from [31]. The necessary results are discussed in Sections 3.1 and 3.2. In Section 3.2 we present a new result that allows for an adaption of the main component identification algorithm from [31], which has better numerical stability. Lastly, we present our new approach in Section 3.4 and pose some questions for future work.

3.1. Fiber Products and Exceptional Sets

Let $f(v;p) : \mathbb{C}^N \times \mathbb{C}^m \to \mathbb{C}^n$ be a system of polynomial equations in N variables and m parameters, and $\pi : \mathcal{V}(f(v;p)) \to \mathbb{C}^m$ the projection map $(v;p) \to p$. We are interested in the sets $D_h \subset \mathbb{C}^m$, the closure of the set of $\hat{p} \in \mathbb{C}^m$ such that $\dim(\mathcal{V}(f(v;\hat{p})) = h$. Notice that $\mathcal{V}(f(v;\hat{p})) = \pi^{-1}(\hat{p})$, so we are interested in the dimension of the fibers of the map π. The fundamental problem is that even though the closure of D_h is algebraic, it is not necessarily an irreducible component of $\mathcal{V}(f(v;p))$.

Our tool for discovering the sets $D_h(\pi)$ is the fiber product. Let X and Y be algebraic sets, then the *two-fold fiber product* of the map $\pi : X \to Y$, denoted by $X \times_Y X$, is defined to be the algebraic set $(\pi \times \pi)^{-1}(\Delta)$ where

(1) Δ is the diagonal of $Y \times Y$, and

(2) $\pi \times \pi$ is the induced map $X \times X \to Y \times Y$.

The diagonal, Δ, is identified with Y via the map $(y, y) \to y$ and the composition of this map with $\pi \times \pi$ induces an algebraic map $\Pi : X \times X \to Y$. The k-fold fiber product is defined analogously (and stated in further generality below).

The fiber product can be defined in any setting where a set of maps share a codomain. However, we restrict our attention to the algebraic setting, where the k-fold fiber product is defined for k algebraic maps $\pi_i : X_i \to Y$ from algebraic sets $X_1 \times X_2 \times \ldots \times X_k$ to an algebraic set Y. The *fiber product*,

$$(\prod_{i=1}^{k} X_i)_Y$$

over Y is defined to be the algebraic set $(\pi_1 \times \ldots \times \pi_k)^{-1}(\Delta)$ where

(1) Δ is the diagonal of $Y^k = Y \times \ldots \times Y$ (k times) consisting of all points $(y, \ldots, y) \in Y^k$

(2) $\pi_1 \times \ldots \times \pi_k$ is the induced map $\prod_{i=1}^{k} X_i \to Y^k$.

As in the two-fold case, the composition of the projection $\Delta \to Y$ with $\pi_1 \times \ldots \times \pi_k$ induces an algebraic map $\Phi : \prod_{i=1}^{k} X_i \to Y$.

Much of the theory of fiber products can be cast in even more general terms, where X_i and Y are quasiprojective algebraic sets [31]. However, since our goal is to use fiber products to investigate a single algebraic set, we can restrict our attention to the algebraic sets $X := \mathcal{V}(f(v;p))$, $Y := \mathbb{C}^m$, and only one algebraic map, the projection $\pi : X \to Y$. The notation above can then be simplified by denoting the k-fold fiber product of X with itself over Y by $\prod_{Y}^{k} X$.

3.2. Main Components: How fiber products uncover exceptional sets

The main result of [31] tells us how fiber products can be used to discover exceptional sets. In particular, Corollary 2.14 of [31] guarantees that if Z is an irreducible component of D_h, then there exists Z_π^k an irreducible component of the fiber product $\Pi_Y^k X$ for k sufficiently large, whose projection onto X is Z. The irreducible component Z_π^k is a *main component* of the fiber product and we begin this section with the definition of main components from [31]:

Definition 3.2.1. *Let Z be an irreducible algebraic set and let $\pi : Z \to Y$ be an algebraic map from Z to an algebraic set Y. Note that generically Z is smooth and $\pi_z : Z \to \pi(Z)$ is well behaved. Precisely, by [[30], Theorem A.4.20], there exists a Zariski open dense set $U \subset Z$ such that:*

(1) U consists of smooth points;

(2) $W := \pi(U)$ is a Zariski open dense set of $\overline{\pi(Z)}$ consisting of smooth points;

(3) π_U is of maximal rank;

(4) π_U factors as the composition $\pi_U = s \circ r$, where $r : U \to V$ is an algebraic map onto a quasiprojective manifold V with connected fibers, and $s : V \to W$ is a covering map.

The inclusion map i_U of U into Z induces an inclusion map of $\prod_W^k U$ into $\prod_Y^k Z$. Since $\prod_W^k U$ is a subset of $\prod_W^k U$, we have an embedding $I_U^k : \prod_V^k U \to \prod_Y^k Z$. We define the ***main component*** *Z_π^k of Z in $\prod_Y^k Z$ to be the closure of the image under this inclusion of $\prod_V^k U$ in $\prod_Y^k Z$.*

More important to our work than this definition is the fact that main components are irreducible (Lemma 2.3 from [31]) and the equivalent definition stated in Theorem 3.2.3 (Theorem 2.7 in [31]). We state both of these results here:

LEMMA 3.2.2. *Let Z be an irreducible algebraic subset and let $\pi : Z \to Y$ be an algebraic map from Z to an algebraic set Y. Then the main component Z_π^k is irreducible.*

THEOREM 3.2.3. *Let $\pi : X \to Y$ be an algebraic map between algebraic sets. For a positive integer k, an irreducible algebraic subset $W \subset \prod_Y^k X$ is the main component Z_π^k of some irreducible algebraic subset $Z \subseteq X$ if and only if*

(1) W is taken to itself under the natural action of the symmetric group S_k on $\prod_\pi^k X$;

(2) the dimension of W is $kh + b$ where b is the dimension of the image of W in Y, and $b + h$ is the dimension of the image of W in X under any one of the k induced projections $q_i : \prod_Y^k X \to Y$; and

(3) given a generic point $(w_1, \ldots, w_k)$ of $W \subset \prod_Y^k X$, it follows that $(w_1, \ldots, w_1) \in W$.

If these conditions are satisfied, then $Z = q_i(W)$ for any i.

REMARK 3.2.4. *For a main component Z_π^k and its projection Z under any of the maps q_i, the values b and h in the preceding theorem are given by $b = \pi(Z)$ and $h = \dim Z - b$. We often refer to b and h as the* base *and* fiber *dimension, respectively, and Z as a component of type (b, h).*

The theorem above leads naturally to an algorithm for identifying main components by verifying the three conditions in the theorem. However, many times the point $(w_1, \ldots, w_1) \in W$ in the third condition is highly ill conditioned which can cause issues with numerical algorithms. The following lemma allows us to replace that condition with a more numerically suitable one.

LEMMA 3.2.5. *Let $G = K_1 \cup \ldots \cup K_l$ be a decomposition of an algebraic set into irreducible components. For a general point $(x_1, \ldots, x_k) \in K$ an irreducible component of G^k, let $p_j = (x_1, \ldots, x_{j-1}, x_1, x_{j+1}, \ldots, x_k)$. If $p_1, \ldots, p_k \in K$ then the point $(x_1, x_1, \ldots, x_1) \in K$.*

Proof. The irreducible components of G^k are of the form $K_{i_1} \times \ldots \times K_{i_k}$ for $i_1, \ldots, i_k \in \{i, \ldots, l\}$. Since none of the K_i belong to the union of the remaining K_i, if follows that a general point of anyone of the K_i does not belong to any of the remaining K_i.

So each p_j must be in an irreducible component of the form

$$K_{i_1} \times \ldots \times K_{i_j - 1} \times K_{i_1} \times K_{i_j + 1} \times \ldots \times K_{i_k}$$

However, each $p_j \in K$, thus $K = K_{i_1}^k$. Then if follows that $(x_1, \ldots, x_1) \in K$. □

Using Theorem 3.2.3 and Lemma 3.2.5 we have the following pseudo-code for an algorithm that identifies main components of a fiber product:

Algorithm 2 Main Component Identification Algorithm

Input: A witness set for an irreducible component W of a fiber product
Output: 1 if main component, 0 if not a main component

1. Use a membership test to determine if every point in the set S is on the component, where S is the set of points resulting from the symmetric group action on the coordinates of a witness point.

2. Compute b and h_i for the component, where $h_i = \dim q_i(W)$. Confirm that the h_i are all equal and that $\dim(W) = kb + h$.

3. Use a membership test to confirm that the points $p_j = (x_1, \ldots, x_{j-1}, x_1, x_{j+1}, \ldots, x_k)$ for $j = 1 \ldots k$ are on the component.

4. Return 1 if $1 - 3$ are true, else return 0.

The main component algorithm relies on two tools. First, a membership test is used in steps one and three. Membership tests were outline in Section 1.4.2. The second tool is necessary for computing b and h_i. The following result provides a way to compute the dimension of a projection. The result is a special case of Theorem A.6.1 in [30], but is stated nicely in [13] as follows:

THEOREM 3.2.6. *Let $V \subset \mathbb{C}^j$ be an irreducible algebraic set and $x^* \in V$ a generic point. Denote by $J(x^*)$ the Jacobian matrix of f at x^*. The dimension of the null space of* $\begin{bmatrix} J(x^*) \\ B \end{bmatrix}$, *say p, is the dimension of the fiber over x^* and $\dim \pi(V) = j - p$, where $\pi(x) = Bx$.*

In the opening paragraph of this section, we remarked that an irreducible component $Z \in D_h(\pi)$ of type (b, h) would be promoted to irreducibility in the fiber product for k sufficiently large. A lower bound on the value of such k is given in the following result from [30]:

THEOREM 3.2.7. *Let $\pi : X \to Y$ be a dominant algebraic map between irreducible algebraic subsets and Z an irreducible component of $\mathcal{D}_h(\pi)$ of type (b, h). Then Z_π^k is an irreducible component of $\prod_Y^k X$ for*

$$k \geq \begin{cases} \dim X - \dim Z + 1 & \text{if } h = h(X) + 1 \text{ or } \dim Z = \dim X - 1; \\ \dim X - \dim Z & \text{otherwise} \end{cases}$$

where $h(X)$ is the generic fiber dimension.

REMARK 3.2.8. *Recall that an algebraic map, between algebraic sets, $g : X \to Y$ is called dominant if $Y = \overline{g(X)}$.*

The following corollary (also from [30]) of Theorem 3.2.7 gives a lower bound on k to promote all irreducible components of D_h to irreducibility for all h:

COROLLARY 3.2.9. *Let $\pi : X \to Y$ be a dominant algebraic map between irreducible algebraic sets. Then all irreducible components of $\mathcal{D}_h(\pi)$ that could be contained in X for any h will be promoted to irreducibility in $\prod_Y^k X$ for $k \geq \dim(\pi(X))$.*

The bound in this corollary is useful when a global approach to computing fiber products is used (see the next section). However, if one is only interested in finding exceptional sets of a particular (b, h)-type, this bound is a worst case scenario and many times the bound for a particular (b, h)-type is better (Theorem 3.2.7).

3.3. Overview of the "Doubled System" Approach

There is a straightforward, yet clever, way to construct the fiber product due to [31]. As before, let $X = \mathcal{V}(f(v; p)) \subset \mathbb{C}^N \times \mathbb{C}^m$ and $\pi : X \to Y := \mathbb{C}^m$ denote the projection $(v; p) \to p$. The approach of [31] is to form the fiber product of X with itself over Y, that is

$X \times_Y X$, as the solution set of

$$g(v;p) = \begin{bmatrix} f(v;p) \\ f(v';p) \end{bmatrix} = 0$$

in the variables $(v;v';p) \in \mathbb{C}^{2N+m}$. Higher fiber products ($k \geq 2$) can be formed similarly, by adding additional copies of the equations and variables. Using this system of equations a standard algorithm for computing a numerical irreducible decomposition (see chapter 1) can be used to obtain witness sets, then Algorithm 2 can be used to test for main components (and thus compute witness sets for D_h).

EXAMPLE 3.3.1. *Two Link Mechanism: "Doubled System" Approach*

Recall the polynomial system that models the work space of the two link mechanism with variables c_1, s_1, c_2, s_2 and parameters l_1, l_2, p_x, p_y:

$$F(c_1, s_1, c_2, s_2; l_1, l_2, p_x, p_y) = \begin{cases} f_1 = c_1 l_1 + c_2 l_2 - p_x \\ f_2 = s_1 l_1 + s_2 l_2 - p_y \\ f_3 = c_1^2 + s_1^2 - 1 \\ f_4 = c_2^2 + s_2^2 - 1 \end{cases} \tag{2}$$

The solution set, $X = \mathcal{V}(F)$ has dimension 4 (as verified by the software package Bertini [3]). To investigate the parameter space for exceptional sets, we consider the fibers of the map $\pi : \mathcal{V}(F) \to \mathbb{C}^4$. As discussed in the beginning of this chapter, the generic fiber dimension is zero. This is easy to verify by choosing c_1, s_1, c_2, s_2 as random complex numbers and computing a numerical irreducible decomposition. (The software package Bertini [3] confirms

that the corresponding solution set is zero dimensional). Since, $\dim X = 4$ *and the generic fiber dimension is* 0*,* X *is of type* $(b, h) = (4, 0)$*.*

Based on the bound in Corollary 3.2.9, the 4^{th} *fiber product is necessary to promote all possible irreducible components in* D_h *(for any* h*) to irreducible components in the fiber product. However, it is instructive to consider the bound in Theorem 3.2.7 which tells us when sets in* D_h *of a particular* (b, h)*-type are promoted to irreducibility. The following table illustrates the theorem:*

TABLE 3.1. Two Link Mechanism: Bounds on k for Promotion to Irreducibility

$h\backslash b$	0	1	2	3	4
3	2	-	-	-	-
2	2	2	-	-	-
1	4	3	2	-	-
0	-	-	-	-	1

*In the table, a '*1*' is placed in the* $(b, h) = (4, 0)$ *entry to indicate that* $X = \Pi_Y^1 X$*. Notice that the worst case appears in the table; specifically,* $k = 4$ *is required to promote any sets of type* $(b, h) = (0, 1)$ *to irreducibility.*

Suppose we were only interested in exceptional sets with base dimension $b = 1$ *then we only need to compete the fiber product for* $k = 3$*. Recall from the beginning of this chapter that an example of an exceptional set occurs when* $l_1 = l_2$ *and* $p_x = p_y = 0$*. Under this choice of parameters there is one degree of freedom in the choices of the joint angles (and thus in the variables* c_1, s_1, c_2, s_2*); so, this exceptional set is of type* $(b, h) = (1, 1)$ *and requires the* $k = 3$ *fiber product to promote it to irreducibility. Below is the decomposition of the irreducible components of the* $k = 3$ *fiber product by dimension:*

TABLE 3.2. Two Link Mechanism: $k = 3$ Fiber Product Decomposition by Dimension

Dimension	*Number of Components*
6	1
5	2
4	6

If we apply the main component testing algorithm above we would find three main components, summarized as follows:

TABLE 3.3. Two Link Mechanism: $k = 3$ Fiber Product Main Components

Dimension	*Associated Set*	b	h
5	$l_2 = 0$	2	1
4	*Initial System*	4	0
4	$l_1 = l_2,\ P_x = P_y = 0$	1	1

Notice that the main component that corresponds to the exceptional set of type $(b, h) = (1, 1)$ *has dimension* 4. *There is also a main component that corresponds to* X *(meaning that the projection under any of the* q_i *is* X*); we always expect such a component. Lastly, we also find another main component that corresponds to the exceptional set with type* $(b, h) = (2, 1)$. *We leave the reader to decipher the physical meaning of this exceptional set.*

3.4. Using Advanced Tools in NAG to Find Exceptional Sets

As the preceding example shows, if we restrict our attention to exceptional sets of a particular type, it is not always necessary to construct the fiber product for k such that all irreducible sets in D_h for all h are promoted to irreducibility. This is a great advantage; since constructing fiber products for large k using the approach of [31] (outlined in Section 3.3) requires solving a system with k copies of the equations defining the $k = 1$ system and $k \cdot |V| + |P|$ variables, where $|V|$ and $|P|$ are the number of variables and parameters of

the $k = 1$ polynomial system, respectively. This growth in the size of the system and the observation that a global approach is not always necessary (especially if we only care about a specific (b, h)-type), is what motivates us to seek a new approach to finding exceptional sets.

The primary tool of our new approach is a recent advancement in numerical algebraic geometry, regeneration extension. We explain how to use regeneration extension to construct fiber products in the next section and illustrate how it can be more efficient then the double system approach by considering the polynomial system which models the two link mechanism.

3.4.1. Regeneration Extension and Fiber Products. As before, let $f(v; p) : \mathbb{C}^N \times \mathbb{C}^m \to \mathbb{C}^n$ be a system of polynomial equations in N variables and m parameters, and $\pi : \mathcal{V}(f(v; p)) \to \mathbb{C}^m$ the projection map. We are interested in the sets $D_h \subset \mathbb{C}^m$, the closure of the set of $\hat{p} \in \mathbb{C}^m$ such that $\dim(\mathcal{V}(f(v; \hat{p})) = h$.

Remark 3.4.1. *We assume that $X = \mathcal{V}(f)$ is an algebraic set consisting of one irreducible component. If it consists of multiple irreducible components, the methods of this chapter can simply be applied to each component individually.*

A natural first step to investigating the sets D_h is to determine all possible (b, h)-types of the irreducible components of the sets D_h and the value of k sufficient to promote each to irreducibility. This can be determined using Theorem 3.2.7 and requires computing the following:

(1) $\dim X = \dim \mathcal{V}(f)$ (by computing the NID of X), and

(2) $h(X)$, the generic fiber dimension of π (by computing an NID of $\mathcal{V}(f(v; \hat{p}))$, where $\hat{p} \in \mathbb{C}^m$ is chosen at random).

The information obtained from Theorem 3.2.7 can be nicely summarized in a table; for example here is the (b, h) table for X of type $(b, h) = (8, 0)$:

TABLE 3.4. Bounds on k for Promotion to Irreducibility for X of type $(b, h) = (8, 0)$

$h\backslash b$	0	1	2	3	4	5	6	7	8
7	2	-	-	-	-	-	-	-	-
6	2	2	-	-	-	-	-	-	-
5	3	2	2	-	-	-	-	-	-
4	4	3	2	2	-	-	-	-	-
3	5	4	3	2	2	-	-	-	-
2	6	5	4	3	2	2	-	-	-
1	8	7	6	5	4	3	2	-	-
0	-	-	-	-	-	-	-	-	1

This natural first step to exploring the exceptional sets of a polynomial system, leads one to seek a method for constructing fiber products that leverages the computations that have already been completed. More specifically, we desire a way to extend the solution set of $X = \mathcal{V}(f(v; p))$ to include the additional conditions (equations) necessary for the k^{th} fiber product (for any k that might be of interest). For example, if we want to compute the NID of $\Pi^2_Y X$ we need the additional conditions $f(v', p)$, where v' is a relabeling of the variables v as in the approach of Section 3.3. The method of *regeneration extension* [16] provides exactly this capability.

Given a witness set for the pure dimensional algebraic set $A \subset \mathbb{C}^M$ and a polynomial system $g : \mathbb{C}^{M+N} \to \mathbb{C}^n$, regeneration extension computes a numerical irreducible decomposition of $(A \times \mathbb{C}^n) \cap \mathcal{V}(g)$. This method is an adaption of the regeneration cascade algorithm for computing a numerical irreducible decomposition and we point the reader to [16] for the details. Our interest is in using regeneration extension as a subroutine for moving from the k^{th} to $(k + 1)^{st}$ fiber product. Suppose that A is an irreducible component of the k^{th} fiber

product. Regeneration extension can be used to extend A to the $(k+1)^{st}$ fiber product by letting $g = f(v_{k+1}; p)$. We call this subroutine **increase_k(A)**. By applying the subroutine to each irreducible component of the $\Pi_Y^k X$ we obtain an NID for the $\Pi_Y^{k+1} X$.

Using this subroutine and Algorithm 2 we have the following pseudo-code for a new algorithm that finds witness sets for all main components of the k^{th} fiber product:

Algorithm 3 k^{th} Fiber Product NID Algorithm

Input: Polynomial System $f(v, p)$ and k.
Output: Witness sets for the main components of $\Pi_Y^k X$

1. Compute an NID for $X = \mathcal{V}(f)$. Let $\mathcal{W}$ be the set of witness data sets.
2. For $i = 2$ to k do
3. $n := |\mathcal{W}|$
   a. For $j = 1$ to n do
      1. **increase_K(W_j)** for $W_j \in \mathcal{W}$ and let $\mathcal{W}_j$ be the set of witness data sets.
      End for
   b. Let $\mathcal{W} = \{\mathcal{W}_1 \dots \mathcal{W}_n\}$
   End for.
4. Use Algorithm 2 to test each witness set of $\mathcal{W}$ to see if it is a main component.
5. Return witness sets for each main component.

If we choose k sufficient large (as in Theorem 3.2.7) we can obtain a decomposition of the sets D_h for all h; thus finding all possible exceptional sets. Note it is useful in the algorithm above to retain the (b, h) values competed in step 4 to determine the base and fiber dimension of each main component.

The main advantage of this approach, as compared to the *doubled system approach*, is that it often requires tracking fewer homotopy paths. This fact is due to the use of regeneration extension. We point the reader to [16] for examples where computing an NID via regeneration extension requires tracking fewer paths than other NID algorithms, but present an example in the context of exceptional sets here:

EXAMPLE 3.4.2. *NID of $\Pi^3_Y X$ of the Two Link Polynomial System.*

*In section 3.3 the "double system" approach was applied to the polynomial system that models the Two Link Mechanism to find exceptional sets of type $(b, h) = (1, 1)$. This required computing the $k = 3$ fiber product. Using the "double system" approach, computing the NID of $\Pi^3_Y X$ using the software package Bertini [3] and the default NID algorithm (*regenerative cascade*), required tracking* 4222 *homotopy paths* [3] *To obtain the same data using regeneration extension, we first compute an NID for $X = \mathcal{V}(f)$ then apply regeneration extension to each irreducible component $Z \in X$, using the scheme outlined above to compute an NID for $\Pi^2_{\mathbb{C}^m}$. Lastly, we repeat the process to compute an NID for $\Pi^3_{\mathbb{C}^m} X$. The following table summarizes the total number of paths tracked in this process:*

TABLE 3.5. Path Tracking Summary: Two Link $k = 3$ Fiber Product by Regeneration Extension

Fiber Product	*Paths Tracked to compute NID*
$k = 1$	30
$k = 2$	344
$k = 3$	1980
Total:	2354

Using a regeneration extension approach to computing the NID of $\Pi^3_Y X$ requires tracking 1868 *fewer paths then using the default NID algorithm in Bertini [3].*

3.4.2. FURTHER INCREASES IN EFFICIENCY. Regeneration extension is a special case of Cross-Product Intersections [16]. Given a witness set for the pure-dimensional algebraic sets $A \subset \mathbb{C}^M$ and $B \subset \mathbb{C}^N$ and a polynomial system $g : \mathbb{C}^{M+N} \to \mathbb{C}^n$, the Cross-Product Intersection algorithm of [16] computes a numerical irreducible decomposition of $(A \times B) \cap$

[3]Other options for computing an NID using Bertini [3] include the dimension-by-dimension approach and the dimension-by-dimension cascade; to compute an NID for $\Pi^3_{\mathbb{C}^m} X$ these approachs require following 8190 and 13248, respectively.

$\mathcal{V}(g)$. This gives us another tool for finding exceptional sets. For example, suppose that the solution set of the initial system $X = \mathcal{V}(f)$ consists of multiple irreducible components. Letting $A \subset f(v_1, p_1)$ and $B \subset f(v_2, p_2)$ be two of the components and $g = p_1 - p_2$, we can use the algorithm in [16] to compute witness sets for the irreducible components of $A \times_Y B$, the components that arise from A and B in $\Pi^2_Y X$. Higher fiber products can be constructed analogously.

The Cross-Product Intersection approach to constructing fiber products allows certain irreducible components of the k^{th} fiber product to be entirely excluded from extension to the $(k+1)^{st}$ fiber product. Thus we are let to pose the following question:

Question 1: Is it possible to restrict the construction of the fiber product to specified components and still guarantee that we can find all exceptional sets?

In addition to the Cross-Product Intersection, addition efficiencies can be had if one is only interested in exceptional sets of a particular (b, h)-type. If this is the case, a more refined approach is possible, as remarked in [31]. The idea is to intersect a main component with a special hyperplane that is generated by b generic linear polynomials in the parameters, and h generic linear polynomials in each of the k variable groups of the k-th fiber product. Then any of the approaches to constructing fiber products can be applied to this new system, the *special slices system*, which we denote by $f_{(b,h)}$. The result is that upon promotion to irreducibility, main components of the specified (b, h)-type are represented by isolated points in the solution set. However, the solution set that contains the points on the main component of interest can still contain other isolated points, so the main component algorithm is still necessary. This algorithm requires a witness set for the irreducible component that is suspected of being a main component. We pose the following question:

Question 2: Can the special linear polynomials in the *special slices system* and isolated solution sets obtained by applying fiber products methods to the *special slices system* be transformed into witness sets (generic linear polynomials and generic points) for the corresponding irreducible components in the NID of $\Pi^k_Y X$?

An answer to **Question 2** would allow for an algorithm that finds all exceptional sets by searching for exceptional sets of each possible (b, h)-type individually, where the construction of fiber products could use **increment_k(A)** or an even more efficient approach arising from a solution to **Question 1**. We close this section by outlining such an algorithm.

Algorithm 4 Parameter Space Decomposition

Input: Polynomial System $f(v, p)$
Output: Witness sets for the irreducible components of D_h

1. Compute NID for $\mathcal{V}(f)$.
2. Compute the (b, h) type of $\mathcal{V}(f)$.
3. Determine the possible (b, h) types of exceptional sets and for each:
   a. Create system $f_{(b,h)}$ using special slices to isolate sets of type (b, h).
   b. Increment up to the k^{th} fiber product using regeneration extension (or cross product intersections), at each step:
      1. Test for main components
      2. Exclude components by the criteria developed as an answer to **Question 1**.
4. Return list of main components and corresponding witness sets.

3.5. Conclusions

In this chapter, we consider how fiber products can be used to identify sets of parameters for which the solution set of a system of polynomial equations has greater dimension than the dimension that occurs for a generic (random) choice of parameters. While using fiber products in this context is not new, we present two new algorithms that contribute to the existing theory. In particular, we show how regeneration extension, a recent advancement in numerical algebraic geometry, can be used to compute the NID of a fiber product in a

way that can be more efficient then applying standard NID algorithms, as in the *double system approach* of [31]. We also prove a new result that improves the numerical stability of existing algorithms for identifying the main components of fiber products (the irreducible components of the fiber product that must be identified to find exceptional sets).

We also pose two questions, which if answered have the potential to further increase the efficiency with which the NID of fiber products can be computed using numerical methods. The motivation of this ongoing work is the analysis of the parameter spaces of polynomial systems that arise in modeling the work space of mechanisms. As such, we illustrated our discussion and approach using a polynomial system that models the two link mechanism. As future work is completed on the questions posed in this chapter, we anticipate applying these methods to more complicated mechanisms.

CHAPTER 4

Numerical algebraic geometry and unit distance embeddings of finite simple graphs

A finite simple graph is a graph with a finite number of vertices, no loops, and no multiple edges. Such a graph can be considered as a pair $G = (V, E)$ consisting of a finite vertex set V and a non-redundant edge set E consisting of 2-element subsets of V. Let M be a metric space and let $d(p, q)$ denote the distance between a pair of points $p, q \in M$. A unit distance embedding of G into M is a map $\phi : V \to M$ such that the distance between every pair of vertices in E is 1. In other words, if $\{v_i, v_j\} \in E$ then $d(\phi(v_i), \phi(v_j)) = 1$. We will focus on unit distance embeddings into the Euclidean space $\mathbb{R}^n$ equipped with the usual Euclidean metric. This brings us to the fundamental question to be considered in this chapter:

Question 1: How can we determine whether there is a unit distance embedding of a graph $G = (V, E)$ into $\mathbb{R}^n$?

We first remark that there is a unit distance embedding of a graph $G = (V, E)$ into some $\mathbb{R}^n$ for n sufficiently large. This follows from the fact that there is a unit distance embedding of the complete graph K_n into $\mathbb{R}^n$ as follows. First give an arbitrary ordering of the n vertices of K_n using the integers from 1 to n. Next map the i^{th} vertex of K_n to $\frac{1}{\sqrt{2}}e_i$ (where e_i is the i^{th} standard basis vector of $\mathbb{R}^n$). Note that the n coordinate vectors, as points in $\mathbb{R}^n$ lie on a linear space of dimension $n - 1$. This linear space can be shifted so as to contain the origin. In other words, there is a unit distance embedding of K_n into $\mathbb{R}^{n-1}$. As a consequence, any finite simple graph on n vertices has a unit distance embedding into $\mathbb{R}^{n-1}$ since any finite simple graph on n vertices can be embedded into K_n. Given a finite simple graph G, the

minimal embedding dimension of G is defined to be the smallest k such that there exists an embedding of G into $\mathbb{R}^k$ as a unit distance graph. This brings us to the second question to be considered in this chapter:

Question 2: Given a finite simple graph G what is the minimal embedding dimension of G as a unit distance graph?

The crux of this chapter is that there is a unit distance embedding of G into $\mathbb{R}^k$ if there is a real solution to the system of polynomial equations derived from the distance constraints. Thus, a discrete graph-theoretic problem becomes a polynomial system problem, and there are a number of tools from computational algebraic geometry that can be brought to bear on this latter type of problem. We focus in this chapter primarily on numerical methods from the area of numerical algebraic geometry, though the methods described could largely be adapted to symbolic methods.

In this chapter, we provide theoretical algorithmic solutions to Questions 1 and 2 and apply these techniques to two problems, the complete unit-distance bipartite graph ($K_{2,3}$) and the Heawood graph. Using further techniques from numerical algebraic geometry, we expand on the Gerbracht counterexamples to the 1972 conjecture of Chvátal that the Heawood graph (pictured in Figure 4.1) has minimal unit distance embedding dimension 3.

The chapter is organized as follows. In Section 4.1, we briefly describe some of the methods from real numerical algebraic geometry used to solve polynomial systems arising in this chapter. We present the theory and solutions to Questions 1 and 2 in Section 4.2. Sections 4.3 and 4.4 then focus on our two examples.

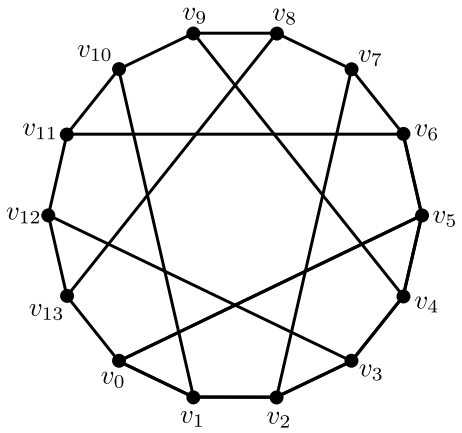


FIGURE 4.1. Heawood Graph

4.1. Real Solutions and Numerical Algebraic Geometry

In this chapter, we concerned with the set of real solutions of a polynomial system, $\mathcal{V}_{\mathbb{R}}(f)$. Computing the real solutions is generally a difficult task. If $\dim \mathcal{V}(f) = 0$, then we can simply use a standard algorithm of NAG and check for solutions with numerically zero complex part. However, this is not always the case. In certain settings, such as the last example in this chapter (embeddings of the Heawood Graph), we can use *a priori* knowledge of the problem to transform the given polynomial system to one that has real isolated solutions. Again, this is not always possible, but all is not lost.

A general approach to computing a solution on each real connected component of $\mathcal{V}_{\mathbb{R}}(f)$ is given in [12]. We make use of this approach to implement the theoretical algorithms presented in this chapter, although other approaches could certainly take the place of this one. The approach of [12] takes as its input a system of polynomial equations $f : \mathbb{R}^N \to \mathbb{R}^{N-d}$ and a witness set for a pure d-dimensional algebraic set and returns a point on each real connected component. Note that if the system has more then $N - d$ equations, one can reduce down to this case using $g = f_1^2 + f_2^2 + \ldots + f_n^2$. We point the reader to [12] for all the

details, but the approach is based on a homotopy that includes an infinitesimal deformation and conditions from optimization:

$$H(x,\lambda,t) = \begin{bmatrix} f(x) - t\gamma z \\ \lambda_0(x-y) + \lambda_1 \nabla f_1(x)^T + \ldots + \lambda_{N-d} \nabla f_{N-d}(x)^T \\ \alpha_0\lambda_0 + \ldots + \alpha_{N-d}\lambda_{N-d} - 1 \end{bmatrix} \tag{3}$$

where $\lambda_0 \ldots \lambda_{N-d}$ are additional variables and $z \in \mathbb{R}^{N-d}$, $\gamma \in \mathbb{C}$, $y \in \mathbb{R}^N \setminus \mathcal{V}_{\mathbb{R}}(f)$, and $\alpha \in \mathbb{C}^{N-d+1}$ are chosen randomly. One can use Bertini [3] to compute the start points $\mathcal{V}(H(x,\lambda,1))$ of the homotopy and then track the paths defined by $H(x,\lambda,t)$ to obtain the desired real solutions. Note that one must be careful to ensure that each solution path $\xi(t)$ and its projection $\pi(\xi(t))$, where $\pi(x,\lambda) = x$, converge. Aside from this description, we will treat this method as a black box, but note that these methods are parallelizable.

4.2. Unit Distance Embeddings as Solutions to Polynomial Systems

Let $G = (V, E)$ be a finite simple graph with vertex set V and edge set E; additionally let $m = |E|$ and $n = |V|$. Then $\phi : V \to \mathbb{R}^N$ defined by $\phi(v_i) := x_i$ is a unit distance embedding of G if $\{v_i, v_j\} \in E$ implies $d(\phi(v_i), \phi(v_j)) = d(x_i, x_j) = 1$. The square of this distance equation is a polynomial and a unit distance embedding of G in $\mathbb{R}^N$ must satisfy the following system of m polynomials:

$$F_p = \left\{ d(\phi(v_i), \phi(v_j))^2 - 1 = 0 \mid \{v_i, v_j\} \in E \right\} \tag{4}$$

Satisfying the system of equations (4) is a necessary condition for a unit distance embedding, but it is not sufficient, since the system does not exclude the possibility that $\phi(v_i) = \phi(v_j)$ for $\{v_i, v_j\} \notin E$. To ensure that the coordinates of each embedded vertex are unique, we can include a semi-algebraic condition for each of the $\frac{n(n-1)}{2} - m$ pairs of vertices $\{v_i, v_j\} \notin E$. This leads to the system of semi-algebraic equations:

$$F_s = \left\{ d(\phi(v_i), \phi(v_j))^2 > 0 \mid \{v_i, v_j\} \notin E \right\} \tag{5}$$

LEMMA 4.2.1. *For a graph $G = (V, E)$, the map $\phi : G \to \mathbb{R}^N$ is a unit distance embedding if and only if the system of semi-algebraic equations $F = \begin{cases} F_p \\ \\ F_s \end{cases}$ is satisfied.*

PROOF. Given a unit distance embedding $\phi : G \to \mathbb{R}^N$, it is clear from the construction above that F is satisfied. To prove the reverse implication, assume that $(x_1; \ldots; x_n) \in \mathbb{R}^{n \times N}$ is a solution to F, where x_i denote the solution coordinates of the N variables associated to the vertex v_i. The solution coordinates x_i define $\phi(v_i)$. □

REMARK 4.2.2. *We consider an embedding where a vertex w_i is mapped to a coordinate on an edge between v_i and v_j (where $w_i \neq v_i, v_j$) a valid embedding. In our examples we did not encounter such an embedding arising from a solution of F; although, it is possible.*

To use the methods of numerical algebraic geometry, we need our system of equations to consist only of polynomials. Recall that the solution set of a system of semi-algebraic equations can be realized by the projection of a solution set of a system of algebraic equations. If we can identify a suitable system of algebraic equations, then we can use the techniques of numerical algebraic geometry. The condition that $d(\phi(v_i), \phi(v_j))^2 > 0$ for $\{v_i, v_j\} \notin$

E is actually much stronger then is necessary to ensure the uniqueness of the embedding coordinates. We actually only need $d(\phi(v_i), \phi(v_j))^2 \neq 0$ for $\{v_i, v_j\} \notin E$. The following polynomial equation imposes this condition by way of the dummy variable q:

$$q \cdot d(\phi(v_i), \phi(v_j))^2 - 1 = 0$$

Thus, the solution set of the polynomial system:

$$\hat{F}_s = \left\{ q_{\{i,j\}} \cdot d(\phi(v_i), \phi(v_j))^2 - 1 = 0 \mid \{v_i, v_j\} \notin E \right\} \tag{6}$$

projects onto the solution set of F_s by the map $\pi(q; x_1; \ldots; x_n) = (x_1; \ldots; x_n)$. This idea generalizes to the following lemma:

LEMMA 4.2.3. *Let $g : \mathbb{R}^n \to \mathbb{R}^N$ be the system of inequalities $\{g_1(x) > 0, \ldots, g_i(x) > 0, \ldots, g_N(x) > 0\}$ such that $g_i(x) \geq 0$ for all $x \in \mathbb{R}^n$, $f : \mathbb{R}^{n+N} \to \mathbb{R}^N$ the system of polynomials defined $f_i(q, x) = q_i \cdot g_i(x) - 1$, and π the projection map defined by $(q, x) \to x$, then $\pi(\mathcal{V}_{\mathbb{R}}(f))$ is equal to the solution set of g.*

This is certainly not a new result and is in fact a variant on the standard trick for representing a semi-algebraic set as the projection of an algebraic set, which can be found in texts on real algebraic geometry such as [5]. However, the proof in this case is simple so we include it here:

PROOF. Let $x^* = (x_1, \ldots, x_j, \ldots, x_n)$ be a solution of g. The system f is satisfied for x^* and $q = (g(x_1)^{-1}, \ldots, g(x_j)^{-1}, \ldots, g(x_n)^{-1})$, so every solution of g has a pre-image in $\mathcal{V}_{\mathbb{R}}(f)$.

Now let (q^*, x^*) be a solution of f, then we have $g_i(x^*) = \dfrac{1}{q_i^*} > 0$. Thus every solution of f projects to a solution of g. □

Lemmas 4.2.3 and 4.2.1 provide a characterization of a unit-distance embedding as a solution to a system of polynomial equations, which we rely on throughout the rest of this chapter and state in the following theorem:

THEOREM 4.2.4. *For a graph $G = (V, E)$, the map $\phi : G \to \mathbb{R}^N$ defined by $\phi(v_i) = x_i$ is a unit distance embedding if and only if $x = (x_1, \ldots, x_i, \ldots, x_{|V|})$ is a solution to the system of algebraic equations* $F = \begin{cases} F_p \\ \\ \hat{F}_s \end{cases}$.

PROOF. The theorem follows directly from Lemmas 4.2.3 and 4.2.1 . □

4.2.1. SOLUTION TO QUESTION 1. The answer to **Question 1** comes as a direct corollary to Theorem 4.2.4:

COROLLARY 4.2.5. *A graph $G = (V, E)$ can be embedded as a unit distance graph in $\mathbb{R}^N$ if $\mathcal{V}_{\mathbb{R}}(F) \neq \emptyset$.*

As a result of this corollary we have the following simple computational test which determines if G has a unit distance embedding into $\mathbb{R}^N$:

Algorithm 5 Unit Distance Embedding Test

Input: A finite simple graph $G = (V, E)$ and N.
Output: 1 if there exists a unit distance embedding of G in $\mathbb{R}^N$, else 0.

1. Construct the system F.
2. Use a method such as [12] to determine if $\mathcal{V}_{\mathbb{R}}(F)$ is non-empty.
3. Return 1 if $\mathcal{V}_{\mathbb{R}} \neq \emptyset$, else return 0

While we use the method of [12] to determine if $\mathcal{V}_{\mathbb{R}}(F)$ is non-empty, there are certainly other ways to ascertain the same information, including symbolic methods in computational algebraic geometry. We use numerical methods because many of the polynomial systems that arise in this context can be rather large. Additionally, the method of [12] finds a solution to F, which could also be returned in Algorithm 5 and can be used to explicitly define a unit distance embedding of G. We also note that an efficient implementation should exploit the fact that unit distance embeddings are invariant under a translation of the coordinate system. Thus, any pair of vertices $\{v_i, v_j\} \in E$ can be embedded as the origin and e_i (where e_i is the i^{th} standard basis vector of $\mathbb{R}^N$). This eliminates $2N$ variables and one equation from the polynomial system.

4.2.2. Solution to Question 2. Our solution to **Question 2** is strictly algorithmic and relies on Algorithm 5.

Algorithm 6 Minimal Unit Distance Embedding Dimension

Input: A finite simple graph $G = (V, E)$
Output: n the minimal embedding dimension of G.

1. n:=2
2. N:=0
3. while N:=0 do
   a. Algorithm 5 with inputs G and n
   b. If Algorithm 5 returns true, $N := 1$. Else $n := n + 1$.
   End While.
3. Return n.

There are two obvious adaptions of this algorithm: (i) as before, we can optionally return a solution of the system used to test for an embedding in dimension n in order to explicitly construct an embedding; and (ii) if, *a priori*, a user knows that G is not unit distance embeddable into $\mathbb{R}^m$ for $m = 1, \ldots k$ the while loop can be set to start at $n = k$. We also know that the algorithm terminates due to the following result:

THEOREM 4.2.6. *The complete graph K_n has a unit distance embedding in $\mathbb{R}^{n-1}$.*

PROOF. An explicit construction was given in the introduction of this chapter. □

The following corollary shows that Algorithm 6 terminates in at most $n = |V|$ iterations of Algorithm 5:

COROLLARY 4.2.7. *A finite simple graph $G = (V, E)$ with $n = |V|$ has a unit distance embedding in $\mathbb{R}^{n-1}$*

PROOF. Any embedding of G into K_n induces an embedding into $\mathbb{R}^{n-1}$ due to Theorem 4.2.6. □

4.3. EXAMPLE: MINIMAL EMBEDDING DIMENSION OF $K_{2,3}$

In this section, we consider the complete bipartite graph $K_{2,3}$ (Figure 4.2) and find its minimal embedding dimension as a unit distance graph.

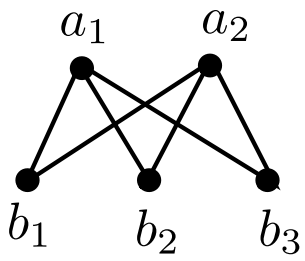


FIGURE 4.2. Complete Bipartite Graph $K_{2,3}$

It is well known that $K_{2,3}$ is not unit distance embeddable into $\mathbb{R}^2$. This can be seen by centering a unit circle at the vertices a_1 and a_2 in Figure 4.2. In order for the vertices b_i for $i \in \{1, 2, 3\}$ to be one unit away from a_1 and a_2, each must be placed at an intersection of the circles. Since two circles with distinct centers can intersect in at most 2 points, the graph is not unit distance embeddable into $\mathbb{R}^2$. This is easily verified using Algorithm 5 and the

software package Bertini [3], which finds $\mathcal{V}_{\mathbb{R}}(F_2) = \emptyset$ (where F_2 is the system of equations for testing if G is unit distance embeddable into $\mathbb{R}^2$).

The next step in Algorithm 6 is to test if G is unit distance embeddable into $\mathbb{R}^3$. The system of equations we need to consider is $\hat{F}_3 : \mathbb{R}^{15} \times \mathbb{R}^4 \to \mathbb{R}^{10}$ defined by:

$$\hat{F}_3 = \begin{cases} (x_{a_i} - x_{b_j})^2 + (y_{a_i} - y_{b_j})^2 + (z_{a_i} - z_{b_j})^2 - 1 & \text{for each pair } \{a_i, b_j\} \in E \\ q_k[(x_{a_i} - x_{a_j})^2 + (y_{a_i} - y_{a_j})^2 + (z_{a_i} - z_{a_j})^2] - 1 & \text{for each pair } \{a_i, a_j\} \notin E \\ q_k[(x_{b_i} - x_{b_j})^2 + (y_{b_i} - y_{b_j})^2 + (z_{b_i} - z_{b_j})^2] - 1 & \text{for each pair } \{b_i, b_j\} \notin E \end{cases} \tag{7}$$

where $(x_{a_i}, y_{a_i}, z_{a_i})$ are variables representing the embedding coordinates of vertex a_i and similarly $(x_{b_i}, y_{b_i}, z_{b_i})$ are variable representing the embedding coordinates of vertex b_i. This is a system in $N \cdot |V| = 3 \cdot 5 = 15$ variables and $\frac{(|V|-1) \cdot |V|}{2} = 10$ equations. However, we can take advantage of the fact that unit distance embeddings are translation invariant, by embedding a pair of vertices in E at the coordinates $(0, 0, 0)$ and $(1, 0, 0)$ (or any other pair of coordinates that are separated by a distance of one). This reduces the system by six variables and one equation, so we work with the system $F_3 : \mathbb{R}^6 \times \mathbb{R}^4 \to \mathbb{R}^9$. Applying the method of [12], we find that $\mathcal{V}_{\mathbb{R}}(F_3) \neq \emptyset$, so $K_{2,3}$ is unit distance embeddable into $\mathbb{R}^3$. One such embedding is given by:

$$\begin{aligned} \phi(a_1) &= (x_{a1}, y_{a1}, z_{a1}) = (0, 0, 0) \\ \phi(a_2) &= (x_{a2}, y_{a2}, z_{a2}) = (-6.31689e^{-03}, 7.29044e^{-01}, 3.15562e^{-01}) \\ \phi(b_1) &= (x_{b1}, y_{b1}, z_{b1}) = (0, 0, 1) \\ \phi(b_2) &= (x_{b2}, y_{b2}, z_{b2}) = (-9.13743e^{-01}, 4.03169e^{-01}, 5.02672e^{-02}) \\ \phi(b_3) &= (x_{b3}, y_{b_3}, z_{b3}) = (6.52283e^{-01}, 6.24457e^{-01}, -4.29626e^{-01}) \end{aligned}$$

Since this embedding was found using the numerical method of [12], the actual solution that gives rise to this embedding was computed to 16 decimal digits. Here we truncated the values to 5 decimal digits and the numerically zero complex parts were omitted.

4.4. Minimal embedding dimension of the Heawood graph

In 1972, Chvátal conjectured that point-line incidence graphs of finite projective planes do not have unit distance embeddings into $\mathbb{R}^2$ [8]. In other words, Chvátal conjectured that the minimal embedding dimension of any point-line incidence graph of a finite projective plane is at least 3. The Heawood graph (Figure 4.1), which we denote by H, is the point-line incidence graph of the Fano plane (the finite projective plane of order 2) and consists of 14 vertices and 21 edges.

Chvátal's conjecture was disproved when Gerbracht provided 11 unit distance embeddings of the Heawood graph in [10]. However, it remains unknown whether these are the only unit distance embeddings or whether there may be more. We seek to answer that question using the methods developed in this chapter.

For this problem, we consider the solution set of the following 21 equations in 28 variables:

$$F = \{(x_i - x_j)^2 + (y_i - y_j)^2 - 1 = 0\} \tag{8}$$

where (x_i, y_i) and (x_j, y_j) are variables representing the embedding coordinates of $\{v_i, v_j\} \in E$. We omit the equations derived from the inequalities that guarantee vertices are assigned unique embedding coordinates, since including these results in a system of 91 equations. In place of these conditions, we simply cull the solutions that result in embeddings with non-unique vertices from the solution set.

Instead of using a method such as [12] to find a real solution of F, we will instead fix a unit distance embedding of a sub-graph of the Heawood graph, which in turn eliminates equations and variables from the system F. If the subgraph is of the right size, the resulting system has the same number of variables and equations, a so called *square system*. We expect that such a system will have isolated solutions (although this is not guaranteed), then any real solutions are easily identified.

To define our new system, we fix a unit distance embedding of a 6-cycle subgraph of H, and one additional vertex that is adjacent to a vertex in the cycle (see Figure 4.3). Assuming such a fixed embedding of the subgraph, we then have the following system:

$$(9)\quad F'(x,y) = \begin{cases} f_1 = (\hat{x}_6 - x_7)^2 + (\hat{y}_6 - y_7)^2 - 1 & f_2 = (x_7 - x_8)^2 + (y_7 - y_8)^2 - 1 \\ f_3 = (x_8 - x_9)^2 + (y_8 - y_9)^2 - 1 & f_4 = (x_9 - x_{10})^2 + (y_9 - y_{10})^2 - 1 \\ f_5 = (x_{10} - x_{11})^2 + (y_{10} - y_{11})^2 - 1 & f_6 = (x_{11} - x_{12})^2 + (y_{11} - y_{12})^2 - 1 \\ f_7 = (x_{12} - x_{13})^2 + (y_{12} - y_{13})^2 - 1 & f_8 = (x_{13} - \hat{x}_0)^2 + (y_{13} - \hat{y}_0)^2 - 1 \\ f_9 = (x_7 - \hat{x}_2)^2 + (y_7 - \hat{y}_2)^2 - 1 & f_{10} = (x_8 - x_{13})^2 + (y_8 - y_{13})^2 - 1 \\ f_{11} = (x_9 - \hat{x}_4)^2 + (y_9 - \hat{y}_4)^2 - 1 & f_{12} = (x_{10} - \hat{x}_1)^2 + (y_{10} - \hat{y}_1)^2 - 1 \\ f_{13} = (x_{11} - \hat{x}_6)^2 + (y_{11} - \hat{y}_6)^2 - 1 & f_{14} = (x_{12} - \hat{x}_3)^2 + (y_{12} - \hat{y}_3)^2 - 1 \end{cases}$$

where $\hat{x}_i, \hat{y}_i$ for $i \in \{0, \ldots, 6\}$ are the embedding coordinates of the vertices in the subgraph. This system defines an algebraic subset of $\mathcal{V}_{\mathbb{R}}(F)$ which may be empty if the particular subgraph cannot be extended to a unit-distance embedding of H.

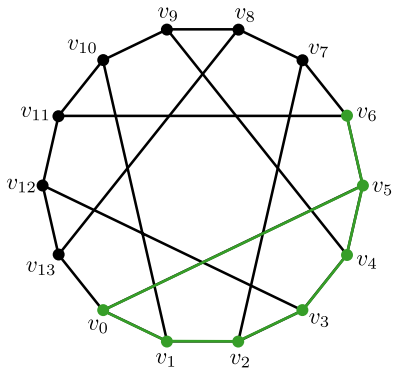


FIGURE 4.3. Heawood Graph and Subgraph (in green).

4.4.1. CHOOSING A UNIT DISTANCE EMBEDDING OF THE SUBGRAPH. There are many ways to choose a unit distance embedding of the subgraph that would result in a polynomial system F' that defines an algebraic subset $\mathcal{V}_{\mathbb{R}}(F') \subset \mathcal{V}_{\mathbb{R}}(F)$. We choose our subgraph embedding randomly in the hope of producing generic points on $\mathcal{V}_{\mathbb{R}}(F)$. Said another way, we assume that it is more likely that a random unit distance embedding of the subgraph can be extended to a unit distance embedding of H.

One way to construct this graph is to choose a random closed polygonal chain of length 6 with unit length edges and a random edge of length one connecting a vertex in the chain to a seventh vertex. This gives us a random unit-distance embedding of the subgraph described above and depicted in Figure 4.3. Algorithmically finding these subgraphs takes some care, but here is an outline of a simple approach:

(1) Let $v_0 = (1, 0)$ and choose 3 random unit vectors, v_1, v_2, and v_3 such that $|v_0 + v_1 + v_2 + v_3| < 2$. Notice that placing these vectors "head to tail" defines an open polygonal chain of length 4 with unit length edges. In addition, unit circles centered at the initial and terminal points must intersect (see Figure 4.4).

(2) Choose one of the intersection points to be the sixth vertex in the desired closed polygonal chain of length six.

(3) Choose a vertex in the closed chain and add a random unit vector to embed the final vertex.

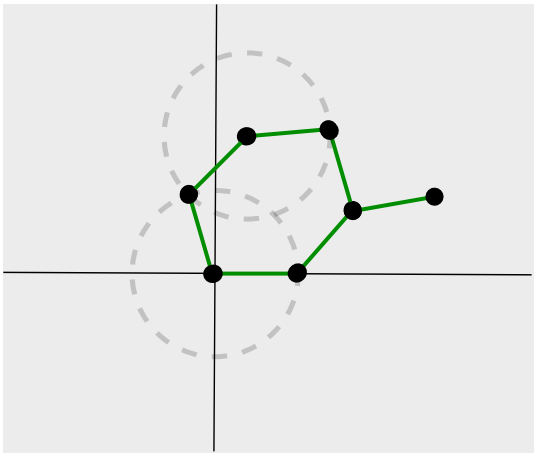
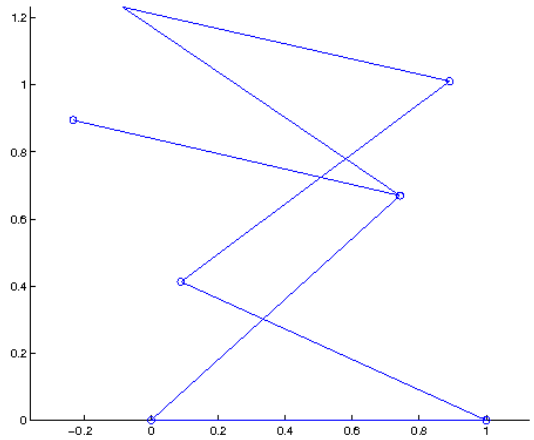


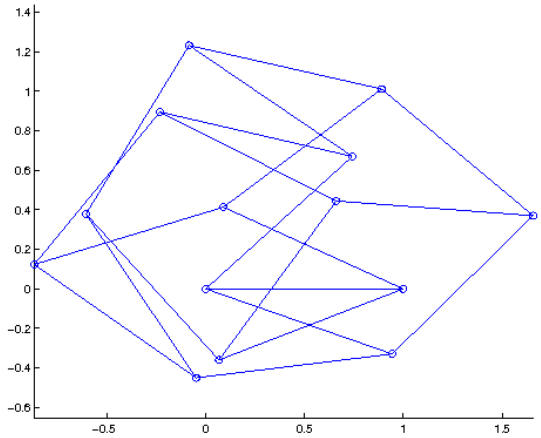
FIGURE 4.4. Random Unit-Distance Subgraph of H

While there are many clever ways to choose the vectors v_1, v_2, and v_3, it is interesting to note that our experiments suggest that for randomly chosen vectors, $|v_0 + v_1 + v_2 + v_3| < 2$ more often then not. If reasonable constraints are put on the angles of the randomly chosen vectors, the probability of success increases significantly. Since ad-hoc methods of fixing a unit distance embedding of the subgraph were so successfully, we did not develop a more sophisticated method.

4.4.2. REAL SOLUTIONS OF THE SUBSYSTEMS. Once we have fixed a unit distance embedding of the subgraph S, we can use the software package Bertini [3] to compute approximate solutions to the system of Equations (9) to arbitrary accuracy. We can think of these solutions as ways to extend the unit distance embedding of the subgraph S to a unit-distance embedding of H. For example, fixing a unit distance embedding as in Figure 4.5a, the software package Bertini finds 94 real solutions (52 of these correspond to embeddings with non-unique vertices). One of the unit-distance embeddings is depicted in Figure 4.5b and the table below lists the coordinates of the vertices.



(A) Unit Distance Embedding of S



(B) Unit Distance Embedding of H

FIGURE 4.5

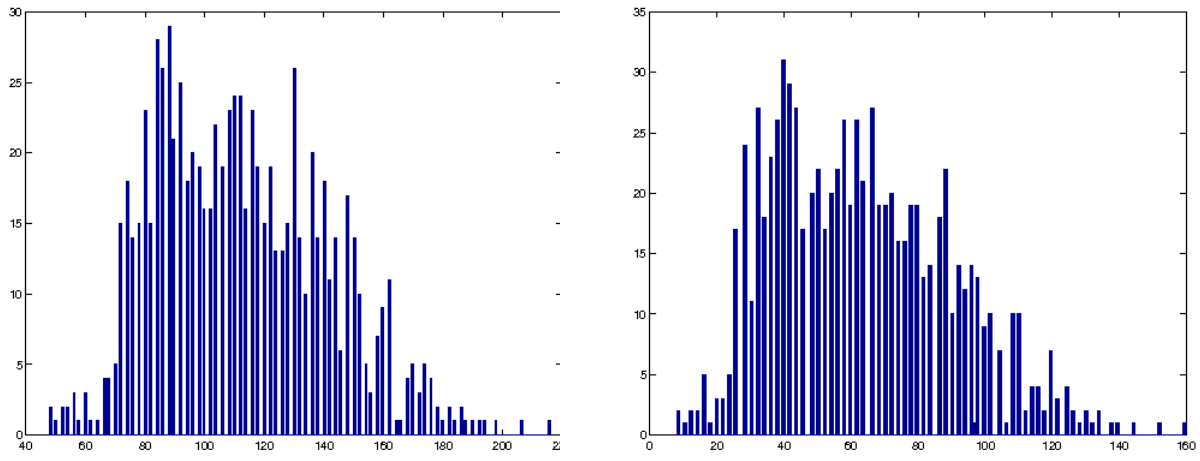
TABLE 4.1. Embedding Vertices

-	x	y
v_0	0	0
v_1	1.0000	0
v_2	0.0893	0.4131
v_3	0.8906	1.0113
v_4	-0.0847	1.2324
v_5	0.7423	0.6701
v_6	-0.2321	0.8950
v_7	-0.8677	0.1229
v_8	-0.0487	-0.4509
v_9	-0.6063	0.3792
v_{10}	0.0671	-0.3601
v_{11}	0.6607	0.4446
v_{12}	1.6580	0.3701
v_{13}	0.9440	-0.3300

4.4.3. SAMPLING $\mathcal{V}_{\mathbb{R}}(F)$. By fixing additional unit distance embeddings of S and computing solutions to the corresponding polynomial systems, we are able to sample $\mathcal{V}_{\mathbb{R}}(F)$. We fix 1000 unit distance embeddings of the subgraph and use Bertini to find all real solutions to the corresponding 1000 systems. For efficiency we use parameter homotopies. In order to use a parameter homotopy, we must first find the solutions to System 9 when the parameters

$\hat{x}_i, \hat{y}_i$ for $i \in \{0, \ldots, 6\}$ are picked at random over the complex numbers. Once these random numbers have been fixed, Bertini can be used to find the solutions. With probability one, the random choice of complex numbers results in a system that has 248 complex solutions. We use the solutions of this system as start points to the parameter homotopy to compute solutions to the systems that arise from the various unit-distance embeddings of S.

Figure 4.6a records the frequency of the size of the real solution sets (excluding the solution sets that were empty). The nonempty solutions sets range in size from 48 solutions to 216 solutions and there are 161 empty solutions sets. However, these solution sets include solutions that correspond to unit distance embeddings with non-unique vertices. To cull these solutions from the solutions sets, we simply say that two vertices v_1 and v_2 are embedded at the same coordinates if $||\phi(v_1) - \phi(v_2)|| < 10^{-8}$ and remove any solutions that contain such pairs of vertices. The figure below shows the frequency of the size of the real solution sets after removing non-embeddings and empty solution sets. The resulting sets range in size from 8 to 160.



(A) Size of Real Solution Sets (B) Size of Embedding Only Sets

FIGURE 4.6. Frequencies over 1000 Trials

4.4.4. Visualizing the Real Solution Counts. As the previous subsection illustrates, changing the parameters that define a polynomial system can have a dramatic effect on the number of real solutions. In this subsection we consider a particular curve through the parameter space corresponding to subgraph S and investigate the real solution counts.

As above, fix a unit distance embedding of the subgraph S. We generate a curve in parameter space defined by rotating this seventh edge around its vertex in the 6-cycle. See Figure 4.7.

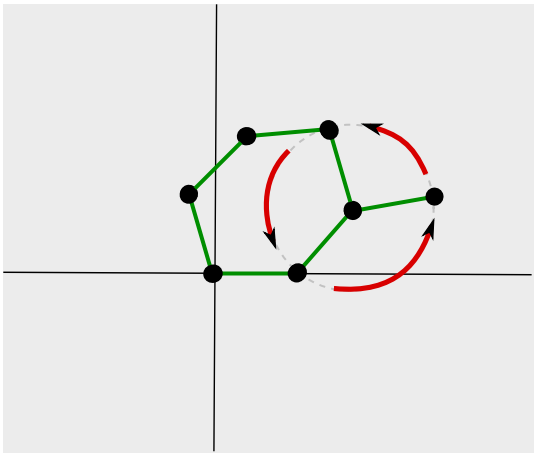
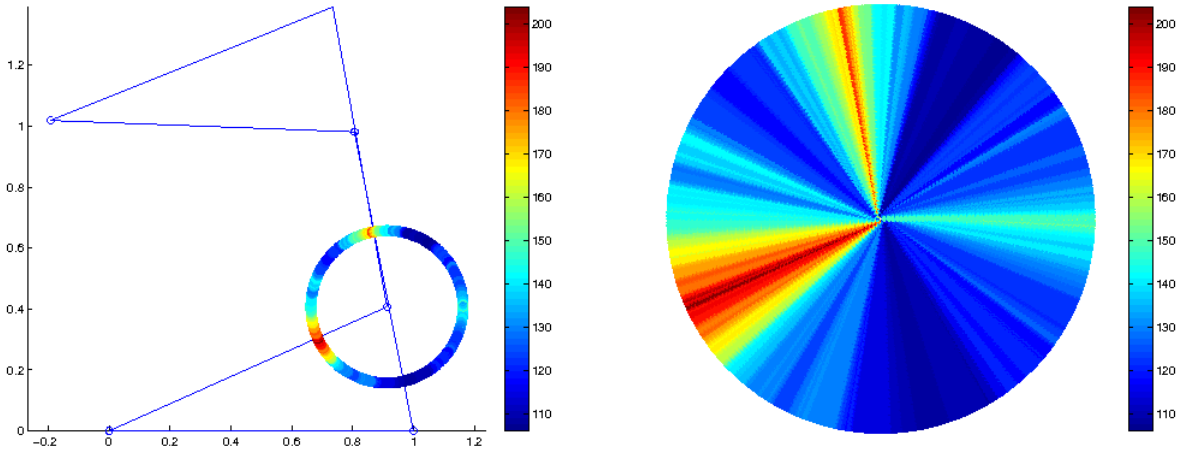


Figure 4.7. Generating A Curve in Parameter Space

We consider a 720 point sampling of this curve that comes from rotating the edge at 0.5 degree intervals. Using the same methods as before, we can obtain approximations for the solutions of the 720 corresponding polynomial systems.

Consider the subgraph S depicted in Figure 4.8a. We generate a heat map corresponding to the solution counts by coloring the edge based on the size of the solution sets. One such heat map appears in Figure 4.8b and more examples appear in Section 4.6.



(A) Subgraph (B) Size of Real Solution Set Heat Map

FIGURE 4.8. Coloring the rotated edge by real solution count generates a heat map.

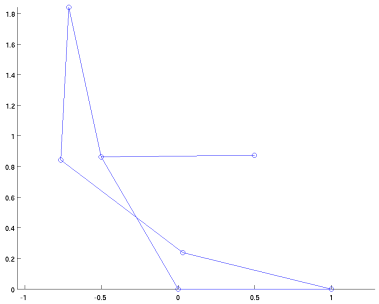
4.5. CONCLUSIONS

In this chapter we present theoretical algorithmic solutions to the problems of determining if there exists a unit distance embedding of a graph in $\mathbb{R}^N$ and determining the minimal embedding dimension of a graph. We demonstrate how the techniques of numerical algebraic geometry can be used to put these algorithms into practice.

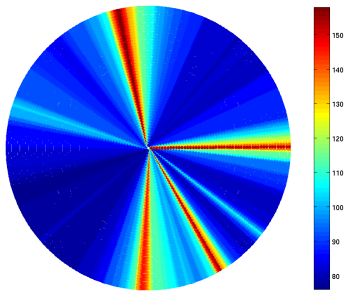
In particular, we consider the Heawood graph and provide further counterexamples to Chvátal's conjecture, that point-line incidence graphs of finite projective planes do not have unit-distance embeddings into $\mathbb{R}^2$ [8]. Our calculations show that there are likely infinitely many counterexamples and possibly non-ridged embeddings of the Heawood graph in $\mathbb{R}^2$.

We close by remarking that there are clear generalizations to the algorithms in this chapter. For instance, it is easy to use exactly these techniques to determine if an embeddings of a graph exists with each pair of vertices $\{v_i, v_j\} \in E$ separated by distance $d_{i,j}$. We hope to investigate these generalizations and others in future work.

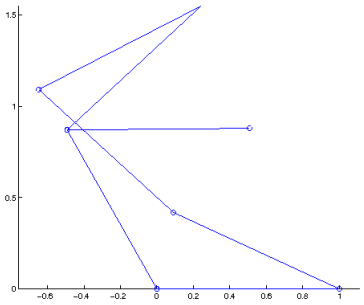
4.6. Appendix of Edge Rotation Solution Counts



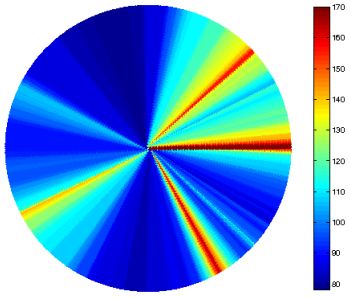
(A) Subgraph S



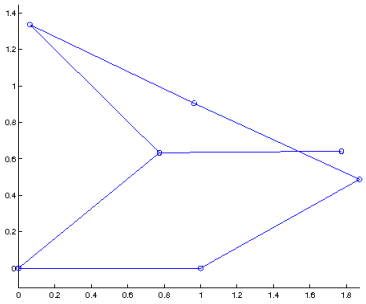
(B) Size of Real Solution Set Heat Map



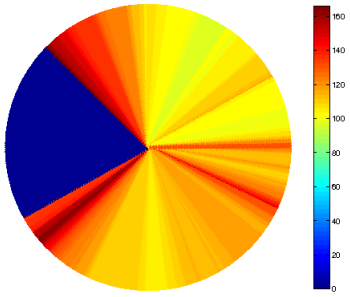
(C) Subgraph S



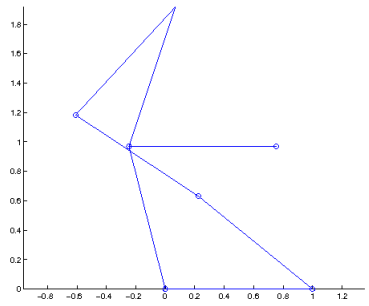
(D) Size of Real Solution Set Heat Map



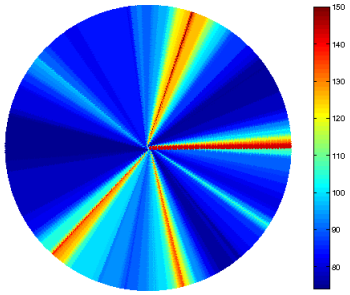
(E) Subgraph S



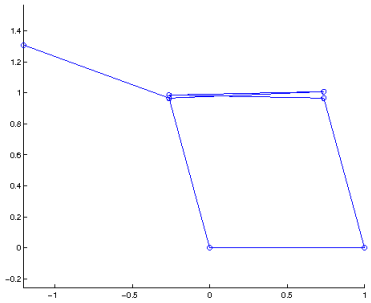
(F) Size of Real Solution Set Heat Map



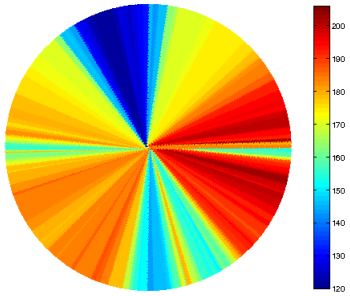
(G) Subgraph S



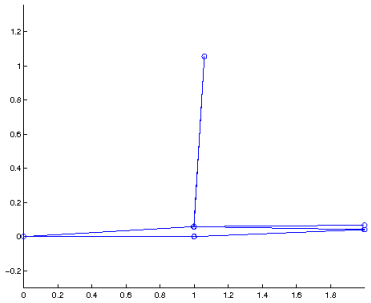
(H) Size of Real Solution Set Heat Map



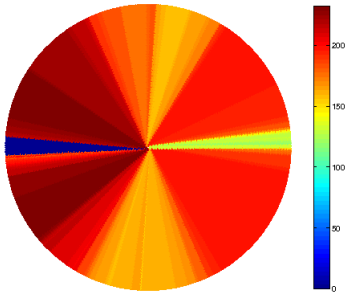
(I) Subgraph S



(J) Size of Real Solution Set Heat Map



(K) Subgraph S



(L) Size of Real Solution Set Heat Map

BIBLIOGRAPHY

[1] D.J. Bates, B. Davis, D. Eklund, E.M. Hanson, C. Peterson. Perturbed homotopies for finding all isolated solution of polynomial systems. *Applied Mathematics and Computation*, 247 (2014), pp. 301-311.

[2] D.J. Bates, J.D. Hauenstein, C. Peterson, and A.J. Sommese, A numerical local dimension test for points on the solution set of a system of polynomial Equations. *SIAM J. Num. Anal.* 47 (2009), pp. 3608–3623.

[3] D.J. Bates, D. Brake, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler, *Bertini: Software for Numerical Algebraic Geometry*, Software available at `www.bertini.nd.edu`, 2013.

[4] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler, *Numerically Solving Polynomial Systems with Bertini*, SIAM, Philadelphia, 2013.

[5] J. Bochnak, M. Coste, M-F. Roy. *Real Algebraic Geometry.* Springer-Verlag Berlin Heidelberg, New York, New York, 1998.

[6] D. Brake, M. Niemerg, and D.J. Bates. Paramotopy: parallel parameter homotopy through Bertini, Software available at `www.paramotopy.com`, 2013.

[7] C. Butcher. An application of the Runge-Kutta space. BIT, 24 (1984), pp. 425-440.

[8] V. Chvátal, D.A. Klarner, D.E. Knuth, *Selected Combinatorial Research Problems, Tech. Report STAN-CS-72-292*, Computer Science Department, School of Humanities and Sciences. Stanford, CA: Stanford University, (1972), pp. 11-13.

[9] K. Gatermann. Symbolic solution of polynomial equation systems with symmetry. *Proceedings of ISSAC-90*, ACM, New York, 1990, pp. 112–119.

[10] E.H.-A Gerbracht, *Eleven Unit Distance Embeddings of the Heawood Graph*, Dec. 30, 2009. http://arxiv.org/abs/0912.5395.

[11] R. Hartshorne, *Algebraic Geometry* Springer-Verlag, New York, 1977.

[12] J.D. Hauenstein, *Numerically computing real points on algebraic sets*, Acta Applicandae Mathematicae, 125 (2012), pp. 115.

[13] J.D. Hauenstein and A.J. Sommese, *Witness sets of projections*, Applied Mathematics and Computation, 217 (2010), pp. 3349-3354.

[14] J.D. Hauenstein, A.J. Sommese, and C.W. Wampler, *Regeneration homotopies for solving systems of polynomials*, Mathematics of Computation, 80 (2011), pp. 345377.

[15] J.D. Hauenstein, A.J. Sommese, and C.W. Wampler, *Regenerative cascade homotopies for solving polynomial systems*, Applied Mathematics and Computation, 218 (2011), pp. 12401246.

[16] J.D. Hauenstein and C.W. Wampler, *Unification and extension of intersection algorithms in numerical algebraic geometry.* Preprint. `http://www3.nd.edu/~jhauenst/preprints/hwGeneralIntersection.pdf`.

[17] J.D. Hauenstein and C.W. Wampler. Isosingular sets and deflation. *Foundations Comp. Math.* 13 (2013), pp. 371–403.

[18] B. Huber and B. Sturmfels *A polyhedral method for solving sparse polynomial systems*, Math. Comput., 64 (1995), pp. 1541-1555.

[19] A. Leykin, *Higher-order deflation for polynomial systems with isolated singular solutions*, in Algorithms in Algebraic Geometry, A. Dickenstein, F.-O. Schreyer, and A. J. Sommese, eds., Springer, New York, 2008, pp. 7997.

[20] A. Leykin, J. Verschelde, Ailing Zhao. Newton's method with deflation for isolated singularities of polynomial systems. *Theoretical Computer Science* 359 (2006), pp. 111–122.

[21] A. Leykin, J. Verschelde, A. Zhao. Higher-Order Deflation for Polynomial Systems with Isolated Singular Solutions. *Algorithms in Algebraic Geometry* 147 (2008), pp. 79–97.

[22] T.Y. Li, *Numerical solution of polynomial systems by homotopy continuation methods*, in Handbook of numerical analysis, Vol. XI, North-Holland, Ams- terdam, 2003, pp. 209304.

[23] T.Y. Li, T. Sauer, J.A. Yorke. The Cheaters Homotopy: An Efficient Procedure For Solving Systems Of Polynomial Equations, *SIAM J. Numer. Anal.* 26 (1989), pp. 1241–1251.

[24] A.P. Morgan and A.J. Sommese. Coefficient-parameter polynomial continuation, *Applied Math. Comp.* 29 (1989), pp. 123–160.

[25] A.P. Morgan and A.J. Sommese. Computing all solutions to polynomial systems using homotopy continuation, *Applied Math. Comp.* 24 (1987), pp. 115–138.

[26] I. Morrison and D. Swinarski. Can you play a fair game of craps with a loaded pair of dice? *Experimental Math.* 20 (2011), pp. 34–56.

[27] T. Ojika, S. Watanabe, T. Mitsui. Deflation algorithm for the multiple roots of a system of nonlinear equations, *Journal of Mathematical Analysis and Applications* 96 (1983), pp. 463–479.

[28] T. Ojika. Modified deflation algorithm for the solution of singular problems. I. A system of nonlinear algebraic equations, *Journal of Mathematical Analysis and Applications* 123 (1987), pp. 199–221.

[29] A.J. Sommese, J. Verschelde, and C.W. Wampler, Numerical decomposition of the solution sets of polynomials into irreducible components, *SIAM J. Numer. Anal.* 38 (2001), pp. 2022–2046.

[30] A.J. Sommese and C.W. Wampler, *The Numerical Solution to Systems of Polynomials Arising in Engineering and Science*, World Scientific, Singapore, 2005.

[31] A.J. Sommese and C.W. Wampler, *Exceptional sets and fiber products*, Foundations of Computational Mathematics, 28 (2008), pp. 171196.

[32] J. Verschelde. *Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation* ACM Trans. Math. Softw. 25 (1999), pp. 251–276. Software and example systems available at `http://www.math.uic.edu/~jan`.

[33] J. Verschelde, P. Verlinden, and R. Cools, *Homotopies exploiting Newton polytopes for solving sparse polynomial systems*, SIAM J. Numer. Anal., 31 (1994), pp. 915930.