Numerical Nonlinear Algebra

vorgelegt von
M. Sc.
Sascha Timme

von der Fakultät II - Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
Dr.rer.nat.

vorgelegte Dissertation

Promotionsausschuss

Vorsitzende: Prof. Dr. Gabriele Steidl
Gutachter: Prof. Dr. Michael Joswig
Prof. Dr. Bernd Sturmfels
Prof. Dr. Mohab Safey El Din

Tag der wissenschaftlichen Aussprache: 15. Juni 2021

Berlin 2021

Acknowledgements

First and foremost, I wish to express my gratitude to my advisors Prof. Dr. Michael Joswig and Prof. Dr. Bernd Sturmfels for many helpful discussions and their advice, support, and encouragement. I would also like to thank them for enabling me to go to various workshops and conferences.

I also want to thank Paul Breiding for many discussions, helpful advice, and the close collaboration on `HomotopyContinuation.jl`. I could also collaborate with many other great people. In particular, I want to thank Piotr Zwiernik for hosting me in Barcelona and for Simon Telen and Marc Van Barel for hosting me in Leuven. Also, I thank my other collaborators Laura Brustenga, Alex Heaton, Kemal Rose, and Madeleine Weinstein. Furthermore, I am grateful for the many helpful discussions with Carlos Amendola, Taylor Brysiewicz, Alperen Ergür, Oliver Gäfvert, Marek Kaluba, Marta Pannizut, and the many other people I met at TU Berlin, MPI Leipzig, and various conferences and workshops.

I want to thank especially my wife Katayoun for her continuous support, counsel, and encouragement. I could not wish for a better partner on my side.

Finally, I want to emphasize that this work would not have been possible without the financial support by the Deutsche Forschungsgemeinschaft (German Research Foundation) Graduiertenkolleg *Facets of Complexity* (GRK 2434).

Abstract

Numerical nonlinear algebra is concerned with the development of numerical methods to solve problems in nonlinear algebra. The main computational task is the solution of systems of polynomial equations. In this thesis, we focus on the numerical solution of polynomial systems using homotopy continuation methods.

We apply techniques from numerical analysis to obtain a mixed-precision path tracking algorithm specifically designed for the application in polynomial homotopy continuation methods. This algorithm uses an adaptive step size control that builds on a local understanding of the region of convergence of Newton's method and the distance to the closest singularity.

Important for the use of numerical nonlinear algebra in mathematical proofs is the possibility to certify that the computed approximate solutions of a polynomial system correspond to true (distinct) solutions of the system. We present a new certification routine based on interval arithmetic and Krawczyk's method that outperforms the state of the art by multiple orders of magnitude.

We also demonstrate numerical nonlinear on a range of applications. To illustrate tools and techniques from numerical nonlinear algebra, we consider Steiner's conic problem and give the first fully real instance of Steiner's conic problem using a computer-assisted proof. We study the action of the projective linear group on cubic surfaces. In particular, we compute the degree of the projective variety given by the Zariski closure of the orbit of a general cubic surface. We also consider the maximum likelihood estimation problem for Gaussian models whose covariance matrices lie in a given linear space. Using numerical nonlinear algebra, we compute the ML degree and dual ML degree for various models of linear covariance matrices. Another application is the study from tensegrity frameworks made from rigid bars and elastic cables, depending on many parameters. We use numerical nonlinear algebra to sample the semi-algebraic *catastrophe set* which characterizes a region of the parameter space that can trigger sudden large-scale shape changes.

Finally, we present the software package `HomotopyContinuation.jl` for the numerical solution of polynomial systems. We describe its functionality, share some of its design and implementation details and demonstrate its impact on the broader research community.

Zusammenfassung

Numerische nichtlineare Algebra beschäftigt sich mit der Entwicklung von numerischen Methoden zur Lösung von Problemen in der nichtlinearen Algebra. In der nichtlinearen Algebra ist die zentrale Berechnungsaufgabe das Lösen von System von polynomiellen Gleichungen ist. In dieser Dissertation fokussieren wir uns auf das Lösen von Polynomsystem mittels Homotopie-Fortsetzungsverfahren.

Wir wenden Techniken aus der numerischen Analysis an, um einen gemischte Präzision Pfadverfolgungsalgorithmus zu erhalten, der speziell für die Anforderungen von polynomiellen Homotopie-Fortsetzungsverfahren gestaltet ist. Dieser Algorithmus nutzt eine adaptive Schrittweitensteuerung, welche auf einem lokalen Verständnis der Konvergenzregion vom Newtonverfahren und dem Abstand zur nächsten Singularität basiert.

Wichtig für die Anwendung von Methoden der numerischen nichtlinearen Algebra in mathematischen Beweisen ist die Möglichkeit zu zertifizieren, dass die berechneten approximativen Lösungen eines Polynomsystems zu echten (unterschiedlichen) Lösungen des Systems korrespondieren. Wir implementieren eine neue Zertifizierungsmethode basierend auf Intervallarithmetik und dem Krawczyk-Verfahren, welche den aktuellen Stand der Technik um mehrere Größenordnungen schlägt.

Wir demonstrieren außerdem numerische nichtlineare Algebra an einer Reihe von Anwendung. Um die Werkzeuge und Techniken der numerischen nichtlinearen Algebra zu demonstrieren, betrachten wir Steiners Kegelschnittproblem und geben die erste komplett reelle Instanz mittels eines computergestützten Beweises an. Wir betrachten die Wirkung der projektiven linearen Gruppe auf kubischen Flächen und berechnen den Grad der projektiven Varietät, die durch den Zariskiabschluss des Orbits einer allgemeinen kubischen Fläche gegeben ist. Wir betrachten zudem das Problem der Maximum-Likelihood Schätzung für Modelle von Gaußschen Verteilungen, dessen Kovarianzmatritzen innerhalb eines gegeben linearen Raums liegen. Mittels numerischer nichtlinearer Algebra berechnen wir den ML Grad und den dualen ML Grad für verschiedene Modelle von linearen Kovarianzmatritzen. Eine weitere Anwendung ist die Untersuchung von Tensegrity Frameworks, welche aus starren Stangen und elastischen Kabeln bestehen. Wir benutzen numerische nichtlineare Algebra, um die semi-algebraische *Katastrophenmenge* zu samplen, welche die Region des Parameterraums charakterisiert, die plötzliche große Formveränderungen auslösen kann.

Abschließend präsentieren wir das Softwarepaket `HomotopyContinuation.jl` für die numerische Lösung von Polynomsystemen. Wir beschreiben seine Funktionalität, teilen einige der Design- und Implementierungsdetails und demonstrieren seinen Einfluss auf die breitere Forschungsgemeinschaft.

Contents

1 Introduction

Numerical linear algebra allows us to efficiently and accurately compute approximate answers to problems in linear algebra. We can hardly overstate its importance in applied mathematics, engineering, and the sciences. Every day, engineers, researchers, and scientists solve linear algebra problems using reliable and robust numerical linear algebra software.

In many applications, linear algebra is the result of approximating *nonlinear* equations. A first step to avoid the reduction to linear algebra is *nonlinear algebra* [MS21]. Nonlinear algebra is a generalization of linear algebra where nonlinear polynomial equations and inequalities replace linear systems. At the heart of nonlinear algebra is algebraic geometry but there are also links to many other branches of mathematics. These include combinatorics, algebraic topology, commutative algebra, and discrete geometry. Nonlinear algebra connects these different branches of mathematics with a strong focus on computations and applications.

In linear algebra, one of the main computational tasks is solving a system of linear equations. The analogue in nonlinear algebra is solving a system of polynomial equations. What does it mean to solve such systems, and how should the solutions be represented? For a system of linear equations, the answer is relatively simple since the solution is always a finite-dimensional affine space. In numerical linear algebra, we represent such a space by a floating-point approximation of its basis vectors. For a system of polynomial equations, the description depends on the dimension of its solution set. If the solution set only consists of finitely many points, then computing a floating-point approximation of each solution is sufficient. If the solution set is positive-dimensional, then a description is more complicated. A possibility is to describe it by a collection of witness sets where each witness set contains the isolated solutions of an associated system of polynomial equations together with some additional data.

There are many methods, symbolic and numerical, to compute all isolated solutions of a polynomial system. One of the first researchers to describe a numerical method were Drexler [Dre77], and Garcia and Zangwill [GZ79]. They used *numerical homotopy continuation* to find all isolated solutions of a system of polynomial equations. The general idea is to construct a homotopy that interpolates between a start and a target system using a continuation parameter. Each solution of the start system is then tracked to a solution of the target system as the homotopy interpolates between the start and the target system. At this point, continuation methods were already established as a general method of finding roots of systems of nonlinear equations but with a significant focus on solving differential equations. Allgower and Georg's book [AG90] provides an overview of the theory with the important case of polynomial systems placed in the perspective of much more general systems.

In the years around 1990, the theory for solving polynomial systems with homotopy continuation methods significantly improved by using techniques from algebraic geometry and complex analysis [MS87b, MS89, MSW90, MSW92b]. The polyhedral homotopy method [HS95]

substantially improved the efficiency to solve general polynomial systems. Around the turn of the century the concept of a *witness set* was introduced [SW96] and refined [SV00, SVW01]. Witness sets give a numerical method to describe positive-dimensional varieties and to compute their irreducible components. The concept of a witness set opened up many applications in algebraic geometry and created the area of *numerical algebraic geometry* [SW96]. The article [HS17] by Hauenstein provides an overview of recent advancements in numerical algebraic geometry. These include pseudo-witness sets [HS10], algorithms for intersecting algebraic varieties [SVW04] that led to a new class of algorithms for solving systems equation by equation [HSW11], as well as other notions of witness sets [LRS18]. The theoretical development was accompanied by the emergence of software packages to put the theory into practice. These include the currently actively maintained software packages `Bertini` [BHSW], `Hom4PS-2/3` [LLT08, CLL14], `NAG4M2` [Ley11], and `PHCpack` [Ver99]. Another important advancement was the development of the software `alphaCertified` [HS12] to certify isolated solutions of a polynomial system starting from numerical approximations. This makes it possible to use numerical results in mathematical proofs.

So far, we have associated the numerical solution of polynomial systems with the field of numerical algebraic geometry. Numerical algebraic geometry's primary focus is on applying numerical techniques to algebraic geometry problems. But only limited attention is paid to the basic underlying numerical techniques. This lack of attention limits the robustness and reliability of computations in numerical algebraic geometry. While experts in numerical algebraic geometry software know how to deal with these issues, other researchers often struggle to the point where they give up on these tools. For a wider adoption of nonlinear algebra in the sciences, it is necessary to improve the robustness and reliability of its computational methods. Therefore, it is necessary to develop *numerical nonlinear algebra*. The first step is a rebuilding of the tools and methods from numerical algebraic geometry with a strong focus on applying ideas, methods, and best practices from numerical analysis.

This thesis is concerned with three different aspects of numerical nonlinear algebra: improving the efficiency and robustness of its computational methods, developing easy to use and reliable software, and applying it to problems in mathematics and the sciences. The reason for these three different aspects is as follows. Dissatisfied with the existing software for the numerical solution of polynomial systems, the author and Paul Breiding started developing the software package `HomotopyContinuation.jl` in late 2017. To test and improve the software, we solved a wide range of nonlinear algebra problems in mathematics and the sciences. This experience showed us where it was necessary to apply methods from numerical analysis to increase the robustness and reliability of the software and the underlying methods. We present the results of this process in this thesis. In the following, we give an overview of its content.

To introduce tools and techniques from numerical nonlinear algebra, we consider in Chapter 2 Steiner's conic problem. Steiner's conic problem asks the following question:

How many conics in the plane are tangent to five given conics in general position?

This classic geometric problem, posed by Jakob Steiner in 1848, allows us to touch on many themes present in this thesis. It was also one of the problems that most significantly helped

in the development of `HomotopyContinuation.jl` and the starting point for the results presented in Chapter 4 and 5. The correct answer to Steiner's conic problem is 3264 (counting over the complex numbers). This was first shown by Chasles in 1864. But whether there exists a configuration of five conics with 3264 real tangent conics was answered affirmatively only in 1997 by Ronga, Tognoli and Vust [RTV97] using a non-constructive proof. In Chapter 2, we give the first fully real instance of Steiner's conic problem using a computer-assisted proof.

After this first adventure into the world of numerical nonlinear algebra, we recall in Chapter 3 the foundations for the numerical solution of polynomials systems using homotopy continuation methods.

Inspired by the work of Deuflhard on Newton methods [Deu11], we present in Chapter 4 a new path tracking algorithm specifically designed for the demands of homotopy continuation methods in the context of numerical nonlinear algebra. The algorithm uses an adaptive step size control that builds on a local understanding of the region of convergence of Newton's method and the distance to the closest singularity. To handle numerically challenging situations, the algorithm uses mixed-precision arithmetic. The result is an algorithm that substantially outperforms the path tracking algorithm in the state of the art software package `Bertini` in terms of efficiency and robustness. We demonstrate this in several numerical examples, including Steiner's conic problem.

To prove that we found a fully real instance of Steiner's conic problem in Chapter 2, we needed to certify that a certain polynomial system has 3264 real isolated zeros. The certification of isolated zeros in numerical algebraic geometry was pioneered by Hauenstein and Sottile with their implementation `alphaCertified` [HS12]. To certify our fully real instance of Steiner's conic problem took more than 36 *hours* using `alphaCertified`. This experience motivated us to develop and implement a faster method for certifying isolated zeros of a polynomial system. The result is presented in Chapter 5. One of the key ideas in our approach is the use of interval arithmetic. While interval arithmetic is used throughout the sciences, it was not yet much used in numerical nonlinear algebra. Our contribution outperforms `alphaCertified` by several orders of magnitude. To illustrate this, we consider our fully real instance of Steiner's conic problem. The certification of this instance now just takes around 3 *seconds* compared to the previous 36 hours. This dramatic increase in certification efficiency allows for a paradigm shift in numerical nonlinear algebra where certification is the default and not just an option.

In Chapter 6–8, we demonstrate the application of numerical nonlinear algebra to three different problem domains: classic algebraic geometry, statistics and metric geometry.

In Chapter 6, we apply numerical nonlinear algebra to a problem in classic algebraic geometry. We study the action of the projective linear group $\mathrm{PGL}(\mathbb{C}, 4)$ on cubic surfaces parameterized by points in $\mathbb{P}^{19}$. We compute the degree of the 15-dimensional projective variety given by the orbit closure of a general cubic surface.

In Chapter 7, we demonstrate the application of numerical nonlinear algebra in statistics. We study the problem of maximum likelihood estimation (MLE) for Gaussian models that have their covariance matrix lying in a given linear space. Maximum likelihood estimation for linear covariance models is a nonlinear nonconvex algebraic optimization problem that is,

in general, poorly understood. There is a need for accurate methods that reliably identify all local maxima. Numerical nonlinear algebra provides such a method. The number of complex critical points for a given model is its maximum likelihood degree (ML degree). Understanding the ML degree of statistical models is one of the prominent topics in algebraic statistics. Using numerical nonlinear algebra, we determine the ML degree and dual ML degree for various linear covariance models and discuss cases where the likelihood function has multiple local maxima.

In Chapter 8, we discuss elastic tensegrity frameworks made from rigid bars and elastic cables, depending on many parameters. For any fixed parameter values, the stable equilibrium position of a framework is determined by minimizing an energy function subject to algebraic constraints. As parameters smoothly change, a stable equilibrium can disappear. This loss of equilibrium is called 'catastrophe' since the framework will experience sudden large-scale shape changes despite small changes of parameters. Using nonlinear algebra, we characterize a semialgebraic subset of the parameter space, the catastrophe set, where catastrophes are possible. In particular, the catastrophe set detects the merging of local extrema from this parametrized family of constrained optimization problems. Tools from numerical nonlinear algebra allow the reliable and efficient computation of all stable equilibrium positions and the catastrophe set.

Finally, in Chapter 9, we demonstrate the functionality of `HomotopyContinuation.jl` and share some of its design and implementation details. One of the primary goals in the development of `HomotopyContinuation.jl` was to create a software package that allows mathematicians, engineers and scientists to solve difficult problems in nonlinear algebra without being an expert in numerical nonlinear algebra. We highlight the impact of `HomotopyContinuation.jl` on this goal by demonstrating the wide range of research results already obtained using the software.

2 Steiner's Conic Problem

This chapter is based on the article "3264 Conics in a Second" [BST20] by Paul Breiding, Bernd Sturmfels, and Sascha Timme. The article is published in the January 2020 issue of the Notices of the American Mathematical Society*. Compared to the published article this chapter is updated to consider the new developments described in Chapter 5.*

In this chapter, we demonstrate tools and techniques from numerical nonlinear algebra on *Steiner's conic problem*. This classic problem, posed by Jakob Steiner in 1848, allows us to touch on many themes present in this thesis.

2.1 Introduction

A *conic* in the plane $\mathbb{R}^2$ is the set of solutions to a quadratic equation $A(x, y) = 0$, where

$$A(x, y) = a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6. \tag{2.1}$$

If there is a second conic

$$U(x, y) = u_1x^2 + u_2xy + u_3y^2 + u_4x + u_5y + u_6, \tag{2.2}$$

then the two conics intersect in four points in $\mathbb{C}^2$, counting multiplicities and counting intersections at points at infinity, provided A and U are irreducible and not multiples of each other. This is the content of *Bézout's theorem*. To take into account the points of intersection at infinity, algebraic geometers like to replace the affine plane $\mathbb{C}^2$ with the complex projective plane $\mathbb{P}^2$. In the following, when we write 'count', we always mean counting solutions in projective space. Nevertheless, we work with $\mathbb{C}^2$ for our exposition.

Figure 2.1: The red ellipse is tangent to four blue ellipses and one blue hyperbola.

A solution (x, y) of the system $A(x, y) = U(x, y) = 0$ has multiplicity ≥ 2 if it is a zero of the *Jacobian determinant*

$$\frac{\partial A}{\partial x} \cdot \frac{\partial U}{\partial y} - \frac{\partial A}{\partial y} \cdot \frac{\partial U}{\partial x} \;=\; 2(a_1u_2 - a_2u_1)x^2 + \cdots + (a_4u_5 - a_5u_4). \tag{2.3}$$

Geometrically, the conic U is *tangent* to the conic A if (2.1), (2.2) and (2.3) are zero for some $(x, y) \in \mathbb{C}^2$. For instance, Figure 2.1 shows a red ellipse and five other blue conics that are tangent to the red ellipse. *Steiner's conic problem* asks the following question:

How many conics in the plane are tangent to five given conics in general position?

The number is five because each tangency condition removes one of the five degrees of freedom in a conic.

There are two fundamental questions related to Steiner's problem. "How many conics are tangent to five given conics?" and "How do we find all conics tangent to five given conics?". The first question is the original conic problem, first asked in 1848 by Steiner who suggested the answer 7776. That number turned out to be incorrect. In the year 1864, Chasles gave the correct answer of 3264. This was further developed by Schubert, whose 1879 book "Kalkül der abzählenden Geometrie" [Sch79] led to Hilbert's 15th problem, and thus to the 20th-century development of enumerative algebraic geometry. The number 3264 appears prominently in the title of the textbook by Eisenbud and Harris [EH16]. A delightful introduction to Steiner's problem was presented by Bashelor, Ksir and Traves in [BKT08].

To answer the second question, we need to find equations whose solutions describe the conics tangent to five given conics. An instance of our problem is given by a list of $30 = 5 \times 6$ coefficients in $\mathbb{R}$ or $\mathbb{C}$:

$$\begin{aligned}
A(x, y) &= a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6, \\
B(x, y) &= b_1x^2 + b_2xy + b_3y^2 + b_4x + b_5y + b_6, \\
C(x, y) &= c_1x^2 + c_2xy + c_3y^2 + c_4x + c_5y + c_6, \\
D(x, y) &= d_1x^2 + d_2xy + d_3y^2 + d_4x + d_5y + d_6, \\
E(x, y) &= e_1x^2 + e_2xy + e_3y^2 + e_4x + e_5y + e_6.
\end{aligned} \tag{2.4}$$

By eliminating the two unknowns x and y from the three equations (2.1), (2.2) and (2.3), we can write the tangency condition directly in terms of the $12 = 6 + 6$ coefficients $a_1, \ldots, a_6$, $u_1, \ldots, u_6$ of A and U:

$$\mathcal{T}(A, U) \;=\; 256a_1^4a_3^2u_3^2u_6^4 - 128a_1^4a_3^2u_3u_5^2u_6^3 + 16a_1^4a_3^2u_5^4u_6^2 + \cdots + a_5^4a_6^2u_1^2u_2^4. \tag{2.5}$$

The polynomial $\mathcal{T}$ is a sum of 3210 terms. It is of degree six in the variables $a_1, \ldots, a_6$ and of degree six in $u_1, \ldots, u_6$. Known classically as the *tact invariant*, it vanishes precisely when the two conics are tangent.

If the coefficients are general, we can assume that each conic U that is tangent to the conics A, B, C, D and E has a nonzero constant term u_6. We can then set $u_6 = 1$. Steiner's problem for the conics A, B, C, D, E now translates into a system of five polynomial equations in five unknowns u_1, u_2, u_3, u_4, u_5. Each of the five tangency constraints is an

equation of degree six:

$$\mathcal{T}(A,U) \,=\, \mathcal{T}(B,U) \,=\, \cdots \,=\, \mathcal{T}(E,U) \,=\, 0. \tag{2.6}$$

Steiner used Bézout's theorem to argue that these equations have $6^5 = 7776$ solutions. However, this number overcounts because there is a *Veronese surface* of extraneous solutions U, namely the squares of linear forms. These degenerate conics have the form

$$U(x,y) \;=\; (x,y,1)\cdot \ell^T \ell \cdot (x,y,1)^T,$$

where $\ell = (\ell_1,\ell_2,\ell_3)$ is a row vector in $\mathbb{C}^3$. Since

$$U(x,y) = (x,y,1)\begin{pmatrix} 2u_1 & u_2 & u_4 \\ u_2 & 2u_3 & u_5 \\ u_4 & u_5 & 2u_6 \end{pmatrix}(x,y,1)^T,$$

the condition for U to be a square is equivalent to

$$\operatorname{rank}\begin{pmatrix} 2u_1 & u_2 & u_4 \\ u_2 & 2u_3 & u_5 \\ u_4 & u_5 & 2u_6 \end{pmatrix} \;\leq\; 1. \tag{2.7}$$

This discussion leads us to the following algebraic reformulation of Steiner's conic problem:

Find all solutions of the equations (2.6) so that the matrix in (2.7) has rank ≥ 2. (2.8)

We offer a convenient way for you to compute the 3264 complex conics that are tangent to your chosen conics via the web interface at `www.juliahomotopycontinuation.org/DIY`. This web interface uses the software `HomotopyContinuation.jl` to solve problem (2.8) in less than a second.

A related question to Steiner's conic problem is "Do there exist five real conics whose 3264 tangent conics are all real?". This was answered affirmative by Ronga, Tognoli and Vust [RTV97]. In their argument, they do not give an explicit instance but rather show that in the neighborhood of some particular conic arrangement, there must be an instance that has all of the 3264 conics real. Hence, this raises the following problem:

Find an explicit instance of five conics A,B,C,D,E *such that the* 3264 *solutions to (2.8) are all real.* (2.9)

Using numerical nonlinear algebra, we discovered the solution in Figure 2.2. We claim that all the 3264 conics that are tangent to those five conics are real.

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\ e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{bmatrix} = \begin{bmatrix} \frac{10124547}{662488724} & \frac{8554609}{755781377} & \frac{5860508}{2798943247} & \frac{-251402893}{1016797750} & \frac{-25443962}{277938473} & 1 \\ \frac{520811}{1788018449} & \frac{2183697}{542440933} & \frac{9030222}{652429049} & \frac{-12680955}{370629407} & \frac{-24872323}{105706890} & 1 \\ \frac{6537193}{241535591} & \frac{-7424602}{363844915} & \frac{6264373}{1630169777} & \frac{13097677}{39806827} & \frac{-29825861}{240478169} & 1 \\ \frac{13173269}{2284890206} & \frac{4510030}{483147459} & \frac{2224435}{588965799} & \frac{33318719}{219393000} & \frac{92891037}{755709662} & 1 \\ \frac{8275097}{452566634} & \frac{-19174153}{408565940} & \frac{5184916}{172253855} & \frac{-23713234}{87670601} & \frac{28246737}{81404569} & 1 \end{bmatrix}$$

Figure 2.2: The five conics from Proposition 2.1.

Proposition 2.1. *There are* 3264 *real conics tangent to those given by the* 5×6 *matrix in Figure 2.2.*

We provide an animation showing all the 3264 real conics of this instance at this URL:

`www.juliahomotopycontinuation.org/3264/`

The construction of our example originates from an arrangement of double lines, which we call the *pentagon construction*. One can see the pentagon in the middle of Figure 2.3. There are points where the red conic seems to intersect a blue line but they are actually points where the red conic touches one branch of a blue hyperbola. See [Sot] for further details.

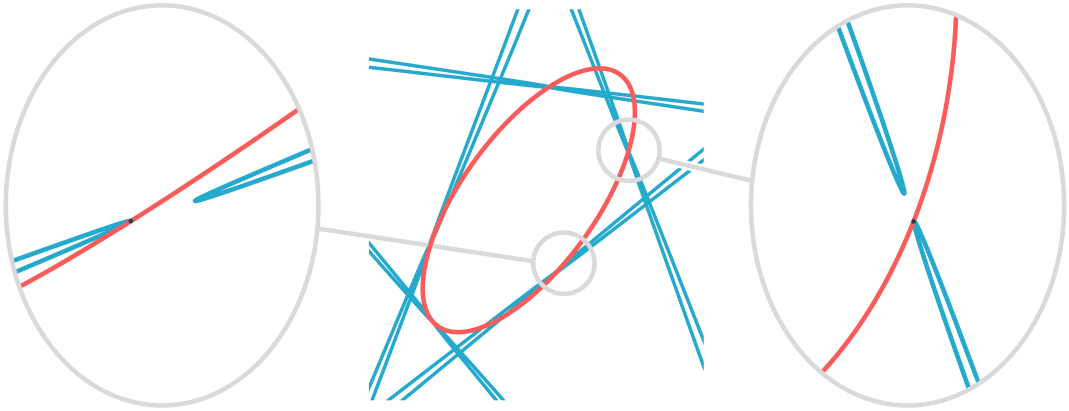

Figure 2.3: The five blue conics in the central picture are those in Proposition 2.1. Shown in red is one of the 3264 real conics that are tangent to the blue conics. Each blue conic looks like a pair of lines, but it is a thin hyperbola whose branches are close to each other. The two pictures on the sides show close-ups around two of the five points of tangency. The red conic is tangent to one of the two branches of the blue hyperbola.

In the following two sections, we discuss the algebro-geometric meaning of the pentagon construction and present a rigorous computer-assisted proof that indeed all of the 3264 conics tangent to our five conics are real. Additionally, we give some insight into how we accomplished finding our five conics.

2.2 Chow Rings and Pentagons

To construct the fully real instance in Proposition 2.1, it was necessary to understand the approach to deriving the number 3264 along the lines of the article [BKT08].

Steiner phrased his problem as that of solving five equations of degree six on the five-dimensional space $\mathbb{P}^5$. The incorrect count occurred because of the locus of double conics in $\mathbb{P}^5$. This is a surface of extraneous solutions. One fixes the problem by replacing $\mathbb{P}^5$ with another five-dimensional manifold, namely the *space of complete conics*. This space is the blow-up of $\mathbb{P}^5$ at the locus of double lines. It is a compactification of the space of nonsingular

conics that has desirable geometric properties. A detailed description of this construction can be found in [BKT08, Sec. 5.1].

In order to answer enumerative geometry questions about the space of complete conics, one considers its Chow ring, as explained in [BKT08, Sec. 5.2]. Elements in the Chow ring of the space of complete conics correspond to subvarieties of this space. More precisely, to *classes* of subvarieties. Two subvarieties belong to the same class if and only if they are *rationally equivalent*. We refer to the textbook by Eisenbud and Harris [EH16] for a formal definition of this concept. The Chow ring for the space of complete conics is worked out in [EH16, Sec. 8.2.4]. Nevertheless, the idea behind studying Chow rings is crystal clear: taking intersections of varieties is translated to multiplication in the Chow ring. In the remainder of this section, we will see this in action.

The Chow ring of the space of complete conics contains two special classes P and L. The class P encodes the conics passing through a fixed point, while the class L encodes the conics tangent to a fixed line. The following relations hold in the Chow ring:

$$P^5 = L^5 = 1,\ P^4L = PL^4 = 2,\ P^3L^2 = P^2L^3 = 4.$$

These relations are derived in [BKT08, Sec. 4.4–5.3]. For instance, the first equation means that, if we take five general conics passing through a fixed point, then the intersection contains one point (namely the point we fixed in the first place). See [BKT08, Table 3] for the geometric meaning of the other equations.

We write C for the class of conics that are tangent to a given conic. In the Chow ring, we have

$$C \ = \ 2P + 2L.$$

This identity is derived in [BKT08, equation (8)]. Our preferred proof is to inspect the first three terms in the expression (2.5) for the tact invariant $\mathcal{T}(A, U)$:

$$\mathcal{T} = 16 \cdot u_6^2(4u_3u_6 - u_5^2)^2 \cdot a_1^4a_2^3 \mod\langle a_2, a_3^3, a_4, a_5, a_6\rangle.$$

This has the following intuitive interpretation. We assume that the given fixed conic A satisfies

$$|a_1| \gg |a_3| \gg \max\{|a_2|, |a_4|, |a_5|, |a_6|\}. \tag{2.10}$$

Thus the conic A is close to $x^2 - \epsilon y^2$, where ϵ is a small quantity. The process of letting ϵ tend to zero is understood as a degeneration in the sense of algebraic geometry. With this, the condition for U to be tangent to A degenerates to $u_6^2 \cdot (4u_3u_6 - u_5^2)^2 = 0$.

The first factor u_6 represents all conics that pass through the point $(0, 0)$. The second factor $4u_3u_6 - u_5^2$ represents all conics tangent to the line $\{x = 0\}$. The Chow ring classes of these factors are P and L. Each of these arises with multiplicity 2, as seen from the

exponents. The desired intersection number is now obtained from the Binomial Theorem:

$$\begin{aligned} C^5 &= 32(L+P)^5 \\ &= 32(L^5 + 5L^4P + 10L^3P^2 + 10L^2P^3 + 5LP^4 + P^5) \\ &= 32(1 + 5 \cdot 2 + 10 \cdot 4 + 10 \cdot 4 + 5 \cdot 2 + 1) \\ &= 32 \cdot 102 = 3264\,. \end{aligned}$$

The final step in turning this into a rigorous proof of Chasles' result is carried out in Section 7 of [BKT08].

The degeneration idea in (2.10) can be used to construct real instances of Steiner's problem whose 3264 solutions are all real. Fulton first observed this and communicated it to Sottile, who then wrote down Fulton's proof in detail [Sot97, Sot]. Ronga, Tognoli, and Vust [RTV97] independently gave a proof. Apparently, they did not know about Fulton's ideas.

Fix a convex pentagon in $\mathbb{R}^2$ and one special point somewhere in the relative interior of each edge. Consider all conics C such that, for each edge of the pentagon, C either passes through the special point or is tangent to the line spanned by the edge. By the count above, there are $(L+P)^5 = 102$ such conics C. If the pentagon is chosen sufficiently asymmetric, then the 102 conics are all real. We now replace each pointed edge by a nearby hyperbola, satisfying (2.10). For instance, if the edge has equation $x = 0$ and $(0, 0)$ is its special point, then we take the hyperbola $x^2 - \epsilon y^2 + \delta$, where $\epsilon > \delta > 0$ are very small. After making appropriate choices of these parameters along all edges of the pentagon, each of the 102 conics splits into 32 conics, each tangent to the five hyperbolas. Here 'splits' means, if the process is reversed, then the 32 different conics collapse into one solution of multiplicity 32. By construction, all 3264 conics are real.

The argument shows that there *exists* an instance in the neighborhood of the pentagon whose 3264 conics are all real, but it does not say *how close* they should be.

2.3 Approximation and Certification

The pentagon construction gives us a guide on how to find a fully real instance. But for finding the instance in Proposition 2.1, serious hands-on experimentation was necessary. This experimentation required us to solve Steiner's conic problem repeatedly for many different instances. This amounts to repeatedly solving the system of polynomial equations (2.6). Here, solve means to compute numerical approximations of all 3264 isolated solutions of an instance using techniques from numerical nonlinear algebra. We describe these in Chapter 3.

To perform numerical computations, it is beneficial to consider the following alternative formulation to the equations (2.6). We use five copies of the equations (2.1)–(2.3), each with a different point of tangency (x_i, y_i), for $i = 1, 2, 3, 4, 5$. The ten equations from (2.1) and (2.2) are quadrics. The five equations from (2.3) are cubics. Altogether, we get the

following system of 15 equations that we display as a 5×3 matrix $F_{(A,B,C,D,E)}$:

$$\begin{bmatrix} A(x_1,y_1) & U(x_1,y_1) & (\frac{\partial A}{\partial x}\frac{\partial U}{\partial y} - \frac{\partial A}{\partial y}\frac{\partial U}{\partial x})(x_1,y_1) \\ B(x_2,y_2) & U(x_2,y_2) & (\frac{\partial B}{\partial x}\frac{\partial U}{\partial y} - \frac{\partial B}{\partial y}\frac{\partial U}{\partial x})(x_2,y_2) \\ C(x_3,y_3) & U(x_3,y_3) & (\frac{\partial C}{\partial x}\frac{\partial U}{\partial y} - \frac{\partial C}{\partial y}\frac{\partial U}{\partial x})(x_3,y_3) \\ D(x_4,y_4) & U(x_4,y_4) & (\frac{\partial D}{\partial x}\frac{\partial U}{\partial y} - \frac{\partial D}{\partial y}\frac{\partial U}{\partial x})(x_4,y_4) \\ E(x_5,y_5) & U(x_5,y_5) & (\frac{\partial E}{\partial x}\frac{\partial U}{\partial y} - \frac{\partial E}{\partial y}\frac{\partial U}{\partial x})(x_5,y_5) \end{bmatrix}. \tag{2.11}$$

Each matrix entry is a polynomial in the 15 variables $u_1, \ldots, u_5, x_1, y_1 \ldots, x_5, y_5$. The 30 parameters of this system are the coefficients of the conics A, B, C, D, E. The system of five equations seen in (2.6) is obtained by eliminating the 10 variables $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_5, y_5$ from the new system $F_{(A,B,C,D,E)}(x)$ introduced in (2.11).

At first glance, it looks like the new formulation (2.11) is worse than the one in (2.6). Indeed, the number of variables has increased from 6 to 15, and the Bézout number has increased from $6^5 = 7776$ to $2^{10}3^5 = 248832$. However, the new formulation is better suited for numerical computations because the equations in the former formulation have a lower degree and can be evaluated faster and more accurately.

One of the key ideas in numerical nonlinear algebra is that once we have the 3264 solutions to $F_{(A',B',C',D',E')}(x)$ for some general set of conics (A', B', C', D', E') we can compute the solutions to *all other* sets of conics (A, B, C, D, E) by using the *parameter homotopy*

$$H(x,t) = F_{t\cdot(A,B,C,D,E)+(1-t)\cdot(A',B',C',D',E')}(x). \tag{2.12}$$

The conic $tA + (1-t)A'$ is defined by the coefficients $ta_i + (1-t)a_i'$, where a_i and a_i' are the coefficients of A and A'. By construction we know the 3264 solutions of $H(x, 0)$. To obtain the solutions at $t = 1$, we move from $t = 0$ to $t = 1$ while keeping track of all 3264 solutions. At the end we obtain the 3264 solutions for the sets of conics (A, B, C, D, E). This technique is called *homotopy continuation*. It is the basic building block for most computations in numerical nonlinear algebra and it is explained in detail in Chapter 3.

Given a general instance of five conics $(A', B', C', D', E') \in \mathbb{C}^{30}$ and its 3264 isolated solutions, we can solve a general real instance $(A, B, C, D, E) \in \mathbb{R}^{30}$ of Steiner's conic problem using the parameter homotopy (2.12). Using `HomotopyContinuation.jl`, this takes a second on a modern laptop. The easiest way to obtain a general instance (A', B', C', D', E') with its 3264 isolated solutions is to use the *monodromy method* described in Section 3.5.

```
julia> certify(F, solution_candidates, target_parameters = totally_real)
Certifying 3264 solutions... 100%|████████████████████████████████████| Time: 0:00:02
  # solutions candidates considered:      3264
  # certified solution intervals (real):  3264 (3264)
CertificationResult
===================
• 3264 solution candidates given
• 3264 certified solution intervals (3264 real, 0 complex)
• 3264 distinct certified solution intervals (3264 real, 0 complex)
```

Figure 2.4: A proof for Proposition 2.1 obtained using `HomotopyContinuation.jl`.

Now we turn back to the fully real instance in Proposition 2.1 and its proof. Given the 3264 computed approximations to this instance, we can *certify* the result *a-posteriori*. Here, *certify* means a computational procedure that takes as input the 3264 computed approximations and our instance and returns a *certificate* guaranteeing that there are at least 3264 distinct real solutions to our instance. Given that we know that Steiner's problem has 3264 solutions, we obtained a computer-assisted mathematical *proof* that our instance is fully real. Figure 2.4 shows the output the certification procedure for the certification routine developed in Chapter 5 and implemented in `HomotopyContinuation.jl`. We discuss the certification of solutions in detail in Section 3.6 and Chapter 5.

2.4 Conclusion

In this chapter, we introduced nonlinear algebra and numerical nonlinear algebra by considering the problem of computing the 3264 conics that are tangent to five given conics in the plane. We demonstrated that the 3264 tangent conics for a given instance of five conics correspond to the 3264 isolated solutions of a system of polynomial equations and discussed briefly how we can use numerical nonlinear algebra to quickly compute these solutions. Additionally, we presented in Proposition 2.1 an instance of five real conics for which there are 3264 real conics tangent to all five given conics. For finding this instance, it was necessary to understand the enumerative geometry approach to deriving the number 3264. The proof of Proposition 2.1 was computer-assisted and relied on the certification of the isolated zeros of a polynomial system. After this first exposure to numerical nonlinear algebra, we will discuss in the next chapter the foundations for the numerical solution of polynomial systems.

3 Background: The Numerical Solution of Polynomial Systems

In this chapter, we outline the foundations for the numerical solution of polynomials systems using homotopy continuation methods. We focus on the necessary concepts and techniques to solve a wide range of systems appearing in applications. For a more comprehensive introduction to the subject, we recommend the book by Sommese and Wampler [SW05], as well as the article by Hauenstein [HS17] for an overview of recent developments.

Before we start, the term "numerical solution" needs more explanation. In general, we want to compute all solutions of a polynomial system and we think of a numerical solution as a point sufficiently close to a solution of the system. For a regular isolated solution, we can make this notion more precise. We require that Newton's method starting with our numerical solution converges to this solution. If a system has infinitely many solutions, then these can be represented by a finite collection of *witness sets* (we introduce these in Section 3.7). Each witness set consists of a finite set of isolated solutions and some additional information. Therefore, the numerical solution of polynomial systems can always be reduced to the computation of isolated solutions of polynomial systems.

We focus on the computation of isolated solutions based on the *homotopy continuation method*. The basic idea for a given polynomial system F that we want to solve is the following.

1. Put the polynomial system F into a *family* of polynomial systems $\mathcal{F}_Q$ depending on a parameter set Q. Then there exists a $p \in Q$ such that $F = F_p \in \mathcal{F}_Q$.
2. Solve a general system $F_q \in \mathcal{F}_Q$.
3. Deform the start system F_q to the target system F_p by moving inside the family $\mathcal{F}_Q$ along a path $\gamma : [0,1] \to Q$ with $\gamma(1) = q$ and $\gamma(0) = p$ and track the solutions of F_q as it is deformed to F_p.

In the last step, we constructed the *homotopy* $H(x,t) = F_{\gamma(t)}(x)$ and the problem of tracking a solution is a *continuation* problem giving the method its name.

From this description, it is not clear why this procedure should allow us to compute *all* isolated solutions of F. Similarly, there is the question of how to embed F into a family of polynomial systems $\mathcal{F}_Q$ and how to solve the start system F_q. The system F_q has to be easier to solve otherwise we would end nowhere. We use algebraic geometry to answer these questions in the following sections.

3.1 Preliminaries

We assume basic familiarity with concepts from algebraic geometry and commutative algebra on the level of the undergraduate textbook "Ideals, Varieties, and Algorithms" by Cox, Little, and O'Shea [CLO15]. In the following, we only perform a brief review of the necessary concepts to introduce our notation and refer for details to [CLO15].

In this thesis, a polynomial f is always considered to be an element of the polynomial ring $\mathbb{C}[x_1, \ldots, x_n]$ in n variables $x_1, \ldots, x_n$ with coefficients in $\mathbb{C}$. A real polynomial f is a polynomial such that f and its conjugate $\bar{f}$ are identical. A polynomial system $F : \mathbb{C}^n \to \mathbb{C}^N$ is a collection of N polynomials $f_1, \ldots, f_N \in \mathbb{C}[x_1, \ldots, x_n]$ with $F = (f_1, \ldots, f_N)$. F is a *square* system if $n = N$ and *overdetermined* if $N > n$. A point $x \in \mathbb{C}^n$ is a solution (or zero) of F if $F(x) = 0$ or equivalently $f_1(x) = \ldots = f_N(x) = 0$. Algebraically, a polynomial system F generates an ideal $\mathcal{I}$. For an ideal $\mathcal{I}$, the *affine variety* $\mathcal{V}(\mathcal{I})$ is defined by

$$\mathcal{V}(\mathcal{I}) = \{x \in \mathbb{C}^n \mid f(x) = 0 \text{ for all } f \in \mathcal{I}\} \subseteq \mathbb{C}^n .$$

If the ideal $\mathcal{I}$ is generated by F, then we also write $\mathcal{V}(F)$ for the variety defined by $\mathcal{I}$. We refer to $\mathcal{V}(F)$ also as the *zero set* or *solution set* of F.

To prove statements related to algebraic varieties, it is immensely helpful to replace $\mathbb{C}^n$ with the n-dimensional projective space $\mathbb{P}^n$. $\mathbb{P}^n$ is a compact space given by the quotient $(\mathbb{C}^{n+1} \setminus \{0\})/\sim$ where $x \sim y$ if only if $x = \lambda y$ for some $\lambda \in \mathbb{C} \setminus \{0\}$. We write the equivalence class of $(x_0, x_1, \ldots, x_n) \in \mathbb{C}^{n+1} \setminus \{0\}$ in $\mathbb{P}^n$ as $[x_0 : x_1 : \ldots : x_n]$. A polynomial $f \in C[x_0, x_1, \ldots, x_n]$ is *homogeneous* of degree d if for all $x \in \mathbb{C}^{n+1}$ and $\lambda \in \mathbb{C}$ we have $f(\lambda x) = \lambda^d f(x)$. Only a homogeneous polynomial is well defined as a map $\mathbb{P}^n \to \mathbb{P}$.

Let $\bar{F} = (\bar{f}_1, \ldots, \bar{f}_N)$ be a system of N homogeneous polynomials in $n+1$ variables. We call such a system *homogeneous*. A homogeneous system $\bar{F}$ generates a homogeneous ideal $\mathcal{J}$. For a homogeneous ideal $\mathcal{J}$, the *projective variety* $\mathcal{V}(\mathcal{J})$ is defined by

$$\mathcal{V}(\mathcal{J}) = \{x \in \mathbb{P}^n \mid f(x) = 0 \text{ for all } f \in \mathcal{J}\} \subseteq \mathbb{P}^n .$$

As in the affine case, we write $\mathcal{V}(\bar{F})$ for the projective variety defined by the homogeneous ideal generated by $\bar{F}$. A homogeneous ideal $\mathcal{J}$ also defines an affine variety

$$\mathcal{V}(\mathcal{J}) = \{x \in \mathbb{C}^{n+1} \mid f(x) = 0 \text{ for all } f \in \mathcal{J}\} \subseteq \mathbb{C}^{n+1}$$

called the *affine cone* over $\mathcal{V}(\mathcal{J})$.

Consider the open set $U_i = \mathbb{P}^n \setminus \mathcal{V}(x_i)$. Since any point in projective space has some nonzero coordinate, the U_i cover $\mathbb{P}^n$ and every point in U_i has a unique representative of the form

$$\left(\frac{x_0}{x_i}, \ldots, \frac{x_{i-1}}{x_i}, 1, \frac{x_{i+1}}{x_i}, \ldots, \frac{x_n}{x_i}\right) .$$

The maps

$$\varphi_i : U_i \to \mathbb{C}^n,\ x \mapsto \left(\frac{x_0}{x_i}, \ldots, \frac{x_{i-1}}{x_i}, \frac{x_{i+1}}{x_i}, \ldots, \frac{x_n}{x_i}\right)$$

are the *standard affine charts* of $\mathbb{P}^n$ and give $\mathbb{P}^n$ the structure of a manifold. It follows that *locally* every projective variety looks like an affine variety. The construction also generalizes to any nonzero homogeneous linear polynomial $\ell(x) = a_0x_0 + \ldots + a_nx_n$ since every point in $U_\ell = \mathbb{P}^n \setminus \mathcal{V}(\ell)$ has a unique affine representative of the form $(x_0/\ell(x), \ldots, x_n/\ell(x))$. Thus, we can identify U_ℓ with the affine space $\mathcal{V}(\ell - 1) \cong \mathbb{C}^n$.

Given a projective variety $X = \mathcal{V}(\bar{F}) \subseteq \mathbb{P}^n$, the intersection of the affine cone of X with the hyperplane $\mathcal{V}(x_i - 1)$, considered as a subset of $\mathbb{C}^n$, is the *dehomogenization* of X with respect to x_i. The dehomogenization of X is cut out by $\bar{F}(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$. Conversely, assume we have an affine variety $X = \mathcal{V}(F) \subseteq \mathbb{C}^n$. We define the *projective closure* $\bar{X}$ of X by introducing a new variable x_0 and

$$\overline{X} = \overline{\{[1 : x] \in \mathbb{P}^n \,|\, x \in X\}}$$

where the closure is taken in the Zariski topology. To determine defining equations for $\bar{X}$ is in general a difficult task. The naive strategy is to use the *homogenization* $\bar{F} = (\bar{f}_1, \ldots, \bar{f}_N)$ of F given by

$$\bar{f}_i = x_0^{d_i} f_i \left(\frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0} \right) \in \mathbb{C}[x_0, \ldots, x_n], \quad i = 1, \ldots, N$$

since $\bar{F}(1, x_1, \ldots, x_n) = F(x_1, \ldots, x_n)$. However, in general $\overline{X} \subsetneq \mathcal{V}(\bar{F}) \subseteq \mathbb{P}^n$. Instead, it is necessary to compute a homogenization of the ideal generated by F. This is in general *not* the ideal generated by the homogenization of F.

Given an projective variety $X \subset \mathbb{P}^n$ of dimension k, the *degree* of X, denoted by $\deg(X)$, is the number of intersection points with a general linear space $L \subset \mathbb{P}^n$ of codimension k, i.e., $\deg(X) = |X \cap L|$. The analogous definition holds for affine varieties $X \subset \mathbb{C}^n$.

A *parameterized* polynomial system is a system $F = (f_1, \ldots, f_N)$ where each f_i is an element of $\mathbb{C}[p_1, \ldots, p_m][x_1, \ldots, x_n]$. To indicate a parameterized polynomial system, we often just write $F(x; p)$. We also consider $F(x; p)$ as a polynomial map $\mathbb{C}^n \times \mathbb{C}^m \to \mathbb{C}^N$. If we want to refer to a polynomial system for a fixed $q \in \mathbb{C}^m$ we write F_q. A *family* of polynomial systems $\mathcal{F}_Q$ with $Q \subseteq \mathbb{C}^m$ is the set $\mathcal{F}_Q := \{F_q \,|\, q \in Q\}$ for a fixed parameterized polynomial system $F(x; p)$.

We say that a point $x \in \mathcal{V}(F) \subseteq \mathbb{C}^n$ is an *isolated solution* of the polynomial system F if there exists an open ball $B \subseteq \mathbb{C}^n$ such that $\mathcal{V}(F) \cap B = \{x\}$. A solution x of F is a *regular* solution if the Jacobian $J_F(x)$ of F at x has full column rank.

3.2 Path Tracking

Before discussing the details on how to choose a suitable family of polynomial systems and how to obtain the solutions to a start system, we take a look at the very last step; tracking a solution as the start system is deformed into the target system. The tracking step is of critical importance for the performance and robustness of the homotopy continuation method. But at this point, we do not go into the fine details of the path tracking procedure since in Chapter 4 a new path tracking algorithm is presented in detail.

Let $H(x,t) : \mathbb{C}^n \times \mathbb{C} \to \mathbb{C}^N$ be a polynomial homotopy. The adjective polynomial refers to the property of $H(x,t)$ that it is a polynomial system for all $t \in \mathbb{C}$. Additionally, we require that H is holomorphic in t. We assume to have a solution x_1 to $H(x,1) = 0$ and we want to compute a solution of $H(x,0) = 0$. For this, we consider the solution path $x(t)$ implicitly defined by the conditions

$$H(x(t),t) = 0 \text{ for all } t \in (0,1] \text{ and } x(1) = x_1. \tag{3.1}$$

From this condition follows that the solution path $x(t)$ is also the solution to the Davidenko differential equation

$$H_x(x(t),t)\dot{x}(t) + H_t(x(t),t) = 0 \tag{3.2}$$

with initial value $x(0) = x_0$ where H_x and H_t denote the partial derivatives with respect to x and t. We say that a solution path $x(t)$ is *smooth* (or *nonsingular*) if there exists a continuous path $x : (0,1] \to \mathbb{C}^N$ such that $x(1) = x_1$, $H(x(1),1) = 0$ and for all $t \in (0,1]$ the point $x(t)$ is a regular isolated solution of $H(x,t)$.

There is a large existing literature on how to track a smooth path $x(t)$ numerically; see, e.g., [AG90]. The basic idea is to treat the problem (3.1) as a sequence of problems

$$H(x(t_k),t_k) = 0\ , \quad k = 0,1,2,\ldots \tag{3.3}$$

with an (a-priori unknown) subdivision $1 = t_0 > t_1 > \ldots > t_M = 0$ of the interval $[0,1]$. Each of the problems (3.3) is then solved by a *correction method*, usually Newton's method, under the assumption that a *prediction method*, e.g., Euler's method, provides a good starting point. The prediction method makes use of the Davidenko differential equation (3.2). The choice of step size $\Delta t_k = t_k - t_{k+1}$ is often given by an *adaptive step size control*. The step size must be chosen appropriately: if the step size is too large, the prediction can be outside the zone of convergence of the corrector, while a too small step size means progress is slow.

A challenge is that a solution path $x(t)$ does not necessarily converge in the limit $t \to 0$ or even if so, not necessarily to a regular isolated solution of $H(x,0)$. This part of the continuation method is referred to as the *endgame* and handled as a distinct problem from the general path tracking procedure outlined previously. More precisely, three different cases are possible.

1. The limit $\lim_{t\to 0} x(t)$ exists and is a regular isolated solution of $H(x,0)$. Then, the Jacobian $H_x(x(0),0)$ has full column rank and the normal path tracking algorithm is sufficient.
2. The limit $\lim_{t\to 0} x(t)$ exists but the Jacobian $H_x(x(0),0)$ does not have full column rank. In this case, $x(0)$ is either an isolated solution with multiplicity greater than one or it is not an isolated solution. We refer in both cases to $x(0)$ as a singular solution.
3. The limit $\lim_{t\to 0} x(t)$ does not exist. In this case, the path is diverging and we say that the solution is 'at infinity'.

To detect and deal with these cases, it is necessary to exploit the additional structure of the problem. Given a solution path $x(t)$, there exists a positive integer m, integers

$w_1, \ldots, w_n \in \mathbb{Z}$ and $\epsilon > 0$ such that

$$x(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix} = \begin{bmatrix} a^{(1)}\, t^{\frac{w_1}{m}} + \sum_{j>w_1}^{\infty} a_j^{(1)}\, t^{\frac{j}{m}} \\ \vdots \\ a^{(n)}\, t^{\frac{w_n}{m}} + \sum_{j>w_n}^{\infty} a_j^{(n)}\, t^{\frac{j}{m}} \end{bmatrix} \text{ for } 0 < t < \epsilon \tag{3.4}$$

with $a^{(1)}, \ldots, a^{(n)} \neq 0$. That is, locally around 0 the path $x(t)$ is analytic in $t^{\frac{1}{m}}$. For a proof of this, see, e.g., [Wal50] or [MSW92a]. We can view the elements of the expression (3.4) as elements in the field of Puiseux series $\mathbb{C}\{\{t\}\}$ and we refer to it as the Puiseux series expansion of the path $x(t)$. The field of Puiseux series is an algebraically closed, valued field with valuation map $\operatorname{val}(x_i(t)) = \frac{w_i}{m}$ for $x_i(t) = \sum_{j=w_i}^{\infty} a^{(j)}\, t^{\frac{j}{m}}$. We refer to $\operatorname{val}(x_i(t)) = \frac{w_i}{m}$ as the valuation of $x_i(t)$. The smallest m such that the Puiseux series expansion in (3.4) exists is called the *winding number* of the path $x(t)$.

Using this additional structure, several algorithms [BHS11b, MSW90, MSW92b, MSW92a] got developed to compute the value of $x(0)$, if it exists, without tracking the path to zero. But these algorithms still require tracking the solution path sufficiently close to the limit. This often poses severe numerical difficulties and makes the computation of singular solutions challenging.

To detect diverging paths and paths with singular endpoints, it is very useful to know the valuation of the solution path $x(t)$. Let $\operatorname{val}(x_i(t)) = \frac{w_i}{m}$ be the valuation of the i-th entry $x_i(t)$ of $x(t)$. We observe that

- if $w_i = 0$, then $x_i(0) = a^{(i)}$;
- if $w_i > 0$, then $x_i(0) = 0$;
- if $w_i < 0$, then $x(t)$ is a diverging path.

We state this observation in following lemma.

Lemma 3.1. *Let $x(t)$ be a solution path, and let $(\operatorname{val}(x_1(t)), \ldots, \operatorname{val}(x_n(t)))$ be the valuations of the Puiseux expansion at $t = 0$. Then, the following holds for $t \to 0$.*

1. *$x(t)$ diverges, if and only if $\operatorname{val}(x_i(t)) < 0$ for at least one $1 \leq i \leq n$.*
2. *In all other cases, $x(t)$ converges but $x(0)$ can be a regular or singular solution.*

Huber and Verschelde used in [HV98] the first item of Lemma 3.1 to develop an algorithm to detect diverging paths. This approach is implemented in `PHCpack` and a similar approach is used in `HOM4PS-2/3` [LLT08]. An improved version of the Huber and Verschelde algorithm is implemented in `HomotopyContinuation.jl` and described in Section 9.2.3.

3.3 Parameter Homotopy

In the previous section, we discussed how to track a solution of a homotopy $H(x, t)$ along a smooth solution path $x(t)$. We also discussed the possible scenarios for the path $x(t)$ in the

limit $t \to 0$. Now we want to start to formalize the general strategy outlined at the beginning of this chapter. For this, we introduce the general framework of a *parameter homotopy*. This framework gives us a powerful tool to find *all* isolated zeros of a polynomial system F. The only assumption is that we can embed F into a family of polynomial systems $\mathcal{F}_Q$ depending on an irreducible variety $Q \subseteq \mathbb{C}^m$. The following theorem is the theoretical foundation for this.

Theorem 3.2 (Parameter Continuation [MS89])**.** *Let $Q \subseteq \mathbb{C}^m$ be an irreducible algebraic variety and let $F(x;p) : \mathbb{C}^n \times Q \to \mathbb{C}^N$ be a system of N polynomials in n variables, $N \geq n$, with m parameters. Given $p \in Q$, we write $F_p(x) = F(x;p)$. Consider the incidence variety*

$$Z = \{(x,p) \in \mathbb{C}^n \times Q \,|\, F(x;p) = 0\} \subseteq \mathbb{C}^n \times Q$$

and let $Y \subset Z$ be a top-dimensional irreducible component of Z. We denote by $\pi_1 : Y \to \mathbb{C}^n,\ (x,p) \mapsto x$ and $\pi_2 : Y \to Q,\ (x,p) \mapsto p$ the projections onto the first and second factor. Furthermore, let $\mathcal{N}(p,Q)$ denote the number of regular isolated solutions of F_p contained in $\pi_1(\pi_2^{-1}(p))$ as a function of $p \in Q$. Then

1. *$\mathcal{N}(p,Q)$ is finite, and there exists a Zariski open set $U \subset Q$ such that $\mathcal{N}(p,Q)$ is the same, say $\mathcal{N}(Q)$, for all $p \in U$. We denote the exceptional set by $\Sigma = Q \setminus U$.*
2. *The homotopy $H(x,t) = F_{\gamma(t)}(x)$ with $\gamma(t) : [0,1] \to U$ has $\mathcal{N}(Q)$ continuous, isolated smooth solution paths $x(t) \in \pi_1(\pi_2^{-1}(\gamma(t)))$;*
3. *As $t \to 0$, the limits of the solution paths, if they exist, of the homotopy $F_{\gamma(t)}(x)$ with $\gamma(t) : [0,1] \to Q$ and $\gamma(t) \in U$ for $t \in (0,1]$ include all the regular isolated solutions of $F_{\gamma(0)}$ contained in $\pi_1(\pi_2^{-1}(\gamma(0)))$.*

   Additionally, if for all $p \in U$ the set $\pi_1(\pi_2^{-1}(p))$ has cardinality $\mathcal{N}(Q)$, then the limits of the solution paths include all isolated solutions of $F_{\gamma(0)}$ contained in $\pi_1(\pi_2^{-1}(\gamma(0)))$. This includes the isolated solutions with multiplicity greater than one.

If we have all regular isolated solutions for some general parameter value $q \in Q$, then item 3 allows us to find all isolated solutions of any system in the family by continuation. This includes all those solutions with multiplicity greater than one. For this, we track all the solutions along a path $\gamma(t) : [0,1] \to Q$ that avoids the exceptional set Σ for $t \in (0,1]$. In the case $Q \cong \mathbb{C}^m$, we can construct the path $\gamma(t)$ easily due to the following lemma.

Lemma 3.3. *Fix a point $p \in \mathbb{C}^m$ and a proper algebraic variety $\Sigma \subseteq \mathbb{C}^m$. For almost all $q \in \mathbb{C}^m$, the one-real-dimensional line segment $\gamma(t) := tq + (1-t)p$ with $t \in (0,1]$ is contained in $\mathbb{C}^m \setminus \Sigma$.*

Proof. The set Σ has complex dimension at most $m-1$ and real dimension at most $2m-2$. Consider the union of all real one-dimensional lines through p and any point of Σ. It has only real dimension of at most $2m-1$ and thus for almost all $q \in \mathbb{C}^m$ it the line segment $\{tq + (1-t)p \,|\, t \in (0,1]\}$ does not intersect Σ. □

Let $F(x;p)$ be a parameterized polynomial system and $p \in \mathbb{C}^m$ some parameters. Theorem 3.2 and Lemma 3.3 state combined that for almost all choices $q \in \mathbb{C}^m$ the *parameter*

homotopy

$$H(x,t) = F(x; tq + (1-t)p) \tag{3.5}$$

has solution paths whose endpoints, for t from 1 to 0, include all isolated zeros of F_p since the line segment $\{tq+(1-t)p \,|\, t \in (0,1]\}$ avoids with probability one the exceptional set Σ. Therefore, if we can, by some means, compute all isolated solutions of F_q, then we can use the homotopy (3.5) to compute all isolated solutions of F_p.

3.4 General Homotopies

The previous section showed that we can compute all isolated solutions of a polynomial system F by embedding it into a family of polynomial systems where we can compute *all* isolated solutions for one general member. Here, the last part is the difficult part for most families since often we do not even know a priori how many isolated solutions a general member of the family has.

To our advantage, there are two families of square polynomial systems that every square polynomial system can be embedded in. These are the family of square dense polynomial systems and the family of square sparse polynomial systems. Both of them are extensively studied in algebraic geometry. We know the number of isolated solutions of a general member and an algorithm to compute all isolated solutions of such a general member.

3.4.1 Square Dense Polynomial Systems

We start with the family of dense polynomial systems. Consider a fixed n-tuple $(d_1, \ldots, d_n)$ of positive natural numbers. Then the family of dense polynomial systems with a degree of at most $(d_1, \ldots, d_n)$ is given by

$$\mathcal{F}^{(d_1,\ldots,d_n)} := \{(g_1, \ldots, g_n) \,|\, g_i \in \mathbb{C}[x_1, \ldots, x_n]_{\leq d_i} \text{ for } i = 1, \ldots, n\},$$

where $\mathbb{C}[x_1, \ldots, x_n]_{\leq d_i}$ is the space of polynomials of degree at most d_i in n variables. A classic result about the number of isolated solutions of an element of $\mathcal{F}^{(d_1,\ldots,d_n)}$ is Bézout's theorem.

Theorem 3.4 (Bézout's theorem)**.** *Given positive integers $d_1, \ldots, d_n$, the number of isolated solutions of a system $F \in \mathcal{F}^{(d_1,\ldots,d_n)}$ is at most the total degree $\prod_{i=1}^n d_i$. The bound is sharp for almost all $F \in \mathcal{F}^{(d_1,\ldots,d_n)}$.*

Every square polynomial system $F = (f_1, \ldots, f_n)$ with $\deg(f_i) = d_i$, $i = 1, \ldots, n$, can be considered as an element of $\mathcal{F}^{(d_1,\ldots,d_n)}$. A general member of $\mathcal{F}^{(d_1,\ldots,d_n)}$ is the system $G = (x_1^{d_1} - 1, \ldots, x_n^{d_n} - 1)$ since the $\prod_{i=1}^n d_i$ isolated zeros of G are the points $(z_1, \ldots, z_n)$ where z_i is any of the d_i-th roots of unity. The basic idea is to use the straight line homotopy

$$H(x,t) = tG(x) + (1-t)F(x) \tag{3.6}$$

to track the known solutions of G to solutions of F. To apply Theorem 3.2, we need to ensure that for all $t \in (0,1]$ the polynomial system $H(x,t)$ still has $\prod_{i=1}^n d_i$ isolated zeros.

Given the deterministic construction of G, this is often not the case. In particular, if F is a real polynomial system. Fortunately, there is a very simple trick to resolve this problem called the γ-trick [SW05, Lemma 7.1.3]. The γ-trick is to multiply G with a random complex number $\gamma \in \mathbb{C}$ since then we do not meet with probability one the exceptional set. This follows by similar reasoning as in Lemma 3.3. As a result, instead of the homotopy (3.6) the following homotopy

$$H(x,t) = t\gamma G(x) + (1-t)F(x) \tag{3.7}$$

is with probability one sufficient to compute all isolated solutions of F. It is called a *total degree homotopy* or sometimes also Bézout homotopy.

3.4.2 Square Sparse Polynomial Systems

The total degree homotopy is very simple to construct but the tradeoff is that the bound given by Bézout's theorem is often substantially larger than the actual number of isolated solutions. In particular, the Bézout bound becomes quickly infeasible for larger systems. We can improve the situation if we do not just look at the degrees d_i of each polynomial contained in the system but rather also their *support*.

Definition 3.5. The *support* of a polynomial $f = \sum_\alpha c_\alpha x^\alpha \in \mathbb{C}[x_1, \ldots, x_n]$ is the set $\operatorname{supp}(f) := \{\alpha \in \mathbb{N}^n \mid c_\alpha \neq 0\}$. The *Newton polytope* of f, $\operatorname{Newt}(f)$, is the convex hull of the support of f, $\operatorname{conv}(\operatorname{supp}(f))$.

Given a tuple of supports $(A_1, \ldots, A_n)$ with $A_1, \ldots, A_n \subseteq \mathbb{N}^n$, we define the family of *sparse* polynomial systems

$$F^{(A_1,\ldots,A_n)} := \{(f_1, \ldots, f_n) \mid f_i \in \mathbb{C}[x_1, \ldots, x_n], \operatorname{supp}(f_i) = A_i\}.$$

The number of isolated solutions of a general member of $F^{(A_1,\ldots,A_n)}$ in the complex algebraic torus $(\mathbb{C}^*)^n := (\mathbb{C} \setminus \{0\})^n$ was first proven by Bernstein [Ber75] after Kushnirenko [Kus75] proved the special case $A_1 = \cdots = A_n$. Before we state the theorem we need to introduce the concept of a *mixed volume*.

Definition 3.6. Let $C_1, C_2, \ldots, C_n \subseteq \mathbb{R}^n$ be bounded convex sets. The function

$$\Lambda : \mathbb{R}^n_{\geq 0} \to \mathbb{R},\ (\lambda_1, \ldots, \lambda_n) \mapsto \text{Volume}(\lambda_1 C_1 + \cdots + \lambda_n C_n)$$

is a homogeneous polynomial in $\lambda_1, \lambda_2, \ldots, \lambda_n$. The coefficient of $\lambda_1\lambda_2\cdots\lambda_n$ is the *mixed volume* of $C_1, \ldots, C_n$ denoted by $\operatorname{MixVol}(C_1, \ldots, C_n)$.

Theorem 3.7 (BKK theorem [Ber75])**.** *Given support sets $A_1, \ldots, A_n \subseteq \mathbb{N}^n$, the number of isolated solutions of a system $F \in \mathcal{F}^{(A_1,\ldots,A_n)}$ in $(\mathbb{C}^*)^n$ is upper bounded by the mixed volume $\operatorname{MixVol}(\operatorname{conv}(A_1), \ldots, \operatorname{conv}(A_n))$. The bound is sharp for almost all $F \in \mathcal{F}^{(A_1,\ldots,A_n)}$.*

For a system $F = (f1, \ldots, f_n)$, we refer to $\operatorname{MixVol}(\operatorname{Newt}(f_1), \ldots, \operatorname{Newt}(f_n))$ as the *BKK bound* or *mixed volume* of the system. The BKK theorem only counts solutions over $(\mathbb{C}^*)^n$ but often all affine solutions are of interest. Huber and Sturmfels introduce in [HS97]

the concept of the *stable mixed volume* of a system that gives the general root count in $\mathbb{C}^n$. If all polynomials f_i of the system have a nonzero constant term then the mixed volume and the stable mixed volume coincide and the BKK bound gives the root count in $\mathbb{C}^n$.

The *polyhedral homotopy* developed in [HS95] by Huber and Sturmfels is a realization of Theorem 3.7. Huber and Sturmfels use methods from polyhedral geometry, in particular, mixed subdivisions, to construct a start system with mixed volume many solutions. Their result is one of the precursors of *tropical geometry* [MS15]. We will only very briefly outline the idea of the procedure and refer for a detailed description to the survey [Li03] by Li. For the necessary concepts in polyhedral geometry, we also refer to the book "Triangulations" by De Loera, Rambau, and Santos [DLRS10].

Given a polynomial system $F = (f_1, \dots, f_n)$ with $f_i = \sum_{\alpha \in \operatorname{supp}(f_i)} c_{\alpha,i} x^\alpha$, consider the homotopy $H(x,t) = (h_1(x,t), \dots, h_n(x,t))$ with

$$h_i(x,t) = \sum_{\alpha \in \operatorname{supp}(f_i)} c_{\alpha,i} t^{\omega_{\alpha,i}} x^\alpha \tag{3.8}$$

where $\omega_{\alpha,i} \in \mathbb{R}$ is picked randomly. With probability one, $H(x,t)$ degenerates in the limit $t \to 0$ into finitely many binomial systems whose total number of solutions equals the mixed volume of F. The binomial systems occurring in the limit can be computed directly by computing the mixed subdivision of the point sets $A_i = \operatorname{supp}(f_i)$, $i = 1, \dots, n$ induced by the liftings $\omega_i = (\omega_{\alpha,i})_{\alpha \in A_i}$, $i = 1, \dots, n$. Each binomial system corresponds to a mixed cell of the induced mixed subdivision. The reduction to binomial systems is immensely helpful since they can be solved directly; see ,e.g., [HS95, Li03]. The solutions to the binomial systems allow finding mixed volume many solutions to $H(x, \varepsilon)$ for some sufficiently small $\varepsilon > 0$. Starting from $\varepsilon > 0$ we then obtain mixed volume many solution paths. By Theorem 3.2 all solutions of F in $(\mathbb{C}^*)^n$ are obtained as the endpoint of one of the solution paths if all paths avoid the exceptional set where the BKK bound is not sharp. We achieve this with probability one if the outlined procedure is applied to a general member $G \in \mathcal{F}^{(A_1,\dots,A_n)}$. The solutions of F are then obtained by tracking the solutions of G along the straight line homotopy $H(x,t) = tG(x) + (1-t)F(x)$.

For many polynomial systems, the BKK bound is substantially lower than the Bézout bound. This lower bound makes the polyhedral homotopy more attractive than a simple total degree homotopy. A downside of the polyhedral homotopy is that it requires the computation of a mixed subdivision. This computation can potentially be very expensive, offsetting the computational savings obtained by the lower solution bound. Since the conception of the polyhedral homotopy method, a series of algorithmic improvements substantially reduced the cost of computing mixed subdivisions [CLL17, Jen16b, Jen16a, GLW05, LL11, MT08, VGC96]. In particular, the algorithm developed in [Jen16b, Jen16a] by Anders Jensen is in practice[1] so efficient that the cost of computing a mixed subdivision is negligible compared to the cost of path tracking. This makes the polyhedral homotopy an excellent choice for solving general square polynomial systems.

[1] An open source implementation of the algorithm is available in the Julia package `MixedSubdivisions.jl`. The code is available at `github.com/saschatimme/MixedSubdivisions.jl`.

3.4.3 Overdetermined Systems

The total degree and the polyhedral homotopy can only be applied to square polynomial systems. But in many applications, the solution set is defined by overdetermined systems. A strategy to still apply a total degree or polyhedral homotopy is to reduce an overdetermined system $F : \mathbb{C}^n \to \mathbb{C}^N$, $N > n$, to a square system $G : \mathbb{C}^n \to \mathbb{C}^n$ such that $\mathcal{V}(F) \subseteq \mathcal{V}(G)$. We refer to this process as *squaring the system up*. For this, a random matrix $A \in \mathbb{C}^{n \times N}$ is used to construct the *squared up system* $A \cdot F$. The following theorem of Bertini type shows that the isolated solutions of $A \cdot F$ are a superset of the isolated solutions of F.

Theorem 3.8. *[SW05, Theorem 13.5.1] Let $F : \mathbb{C}^n \to \mathbb{C}^N$, $N > n$, be a system of polynomials. Assume that $X \subseteq \mathbb{C}^n$ is an irreducible affine variety. Then, there is a nonempty Zariski open set U of $k \times N$ matrices $A \in \mathbb{C}^{k \times N}$ such that for $A \in U$:*

1. *if $\dim X > n - k$, then X is an irreducible component of $\mathcal{V}(F)$ if and only if it is an irreducible component of $\mathcal{V}(A \cdot F)$;*
2. *if $\dim X = n - k$, then X is an irreducible component of $\mathcal{V}(F)$ implies that X is also an irreducible component of $\mathcal{V}(A \cdot F)$;*
3. *if $\dim X$ is an irreducible component of $\mathcal{V}(F)$, its multiplicity as a zero component of $A \cdot F$ is greater than or equal to its multiplicity as a zero component of F, with equality if either multiplicity is 1.*

In particular, in our setting each regular isolated solution of F is also a regular isolated solution of $A \cdot F$. Therefore, to compute all isolated solutions of an overdetermined system $F : \mathbb{C}^n \to \mathbb{C}^N$, we proceed as follows. We first square the system up using a randomly chosen $A \in \mathbb{C}^{n \times N}$. Then, we compute all isolated solutions of $A \cdot F$ using either a total degree or a polyhedral homotopy. From the computed isolated solutions, we then select all those solutions that are also a solutions to F. The last step is usually based on heuristics but in [DHS20] the authors develop techniques to make it rigorous under the assumption that additional global information on $\mathcal{V}(F)$ is available.

3.5 Monodromy Method

In the previous section, we computed all isolated solutions of a square polynomial system by embedding it in a family of dense or sparse polynomial systems. But this approach ignored almost everything about the structure of our specific system. In most applications, we deal naturally with highly structured parameterized polynomial systems $F(x; p) : \mathbb{C}^n \times \mathbb{C}^m \to \mathbb{C}^N$. Since these systems are so structured, the number of isolated solutions is often even substantially lower than the BKK bound. If we have an overdetermined system, then the process of squaring the system up often reinforces this gap even more.

An alternative to the general strategies from the previous section is given by the *monodromy method* [MdCR17, DHJ+18]. This is a *probabilistic* method that computes under certain, often satisfied, conditions all isolated solutions of parameterized polynomial system $F(x; p)$ for a general parameter value $q \in Q$ where $Q \subseteq \mathbb{C}^m$ is an irreducible variety.

This is achieved by only constructing homotopies that stay in the family $\mathcal{F}_Q$ determined by $F(x;p)$. The only requirement for this method is that a *start pair* $(x_0, p_0) \in \mathbb{C}^n \times Q$ with $F(x_0;p_0) = 0$ is given. Before we state the method in detail, we need to introduce some of the necessary concepts.

3.5.1 Covering Spaces and the Monodromy Group

In this subsection, we use methods from algebraic topology to introduce the concept of a *monodromy group*. A good reference is the textbook by Hatcher [Hat02, Section 1.3].

For an irreducible algebraic variety $Q \subseteq \mathbb{C}^m$ consider the incidence variety

$$Z = \{(x,p) \in \mathbb{C}^n \times Q \,|\, F(x;p) = 0\} \subseteq \mathbb{C}^n \times Q\,. \tag{3.9}$$

Denote by $\pi : Z \to Q,\ (x,p) \mapsto p$ the projection onto the second factor. There exists an open set $U \subseteq Q$ such that for every $q \in U$ the fiber $\pi^{-1}(q)$ is finite and has the same cardinality d. Denote by $Y \subseteq Z$ the Zariski closure of $\pi^{-1}(U)$. Y is not necessarily an irreducible variety but we assume this for now.

The map $\pi : Y \to Q$ is a *branched cover* with of degree d and U is the *set of regular values*. Furthermore, there exists an open cover $(\Omega_\beta)_\beta$ of U such that for each β, the fiber $\pi^{-1}(\Omega_\beta)$ is a disjoint union of d open sets in $\pi^{-1}(U)$ such that each set is mapped homeomorphically onto Ω_β. The map $\pi|_U : \pi^{-1}(U) \to U$ is a *d-sheeted covering space*.

We say that a path $\hat{\gamma} : [0,1] \to Y$ is a *lift* of the path $\gamma : [0,1] \to U$ if $\pi|_U \circ \hat{\gamma} = \gamma$. When the start and endpoint of γ coincide, $\gamma(0) = \gamma(1)$, then γ is a *loop* based at $\gamma(0)$. The *path lifting property* for a covering space states that for any path $\gamma : [0,1] \to U$ and any $\hat{q} \in \pi^{-1}(U)$ with $\pi(\hat{q}) = q = \gamma(0)$ there is a unique path $\hat{\gamma} : [0,1] \to \pi^{-1}(U) \subseteq Y$ such that $\hat{\gamma}$ is a lift of γ and $\hat{\gamma}(0) = \hat{q}$. Since $\pi|_U$ is a d-sheeted covering space, there are d distinct paths $\hat{\gamma}_1, \ldots, \hat{\gamma}_d$ lifting γ. As a result, for a loop γ we get for all $1 \leq i \leq d$, $\hat{\gamma}_i(0) = \hat{\gamma}_j(1)$ for some $1 \leq j \leq d$. Thus, the loop γ induces a permutation of the fiber over $\gamma(0)$.

Now consider the fundamental group $\pi_1(U,q)$ of U based at $q \in U$. It follows that we have a group homomorphism $\varphi : \pi_1(U,q) \to S_d$ where S_d is the symmetric group on d elements. The image of φ is the *monodromy group* associated to $\pi^{-1}(q)$. It acts on the fiber $\pi^{-1}(q)$ by permuting the solutions of F_q.

The construction of the monodromy group holds for an arbitrary covering space with finitely many sheets and relied purely on topological arguments. From the construction also follows that the monodromy group is a transitive subgroup of S_d whenever Y is connected. In our setting, this is equivalent to the condition that Y is irreducible.

Lemma 3.9. *The monodromy group of an irreducible branched cover $\pi : Y \to Q$ is transitive.*

Proof. Let $x, y \in \pi^{-1}(q)$ for some $q \in U$. The set $\pi^{-1}(U)$ is path-connected since Y is irreducible. Thus, there exists a path $\hat{\gamma}$ with $\hat{\gamma}(0) = x$ and $\hat{\gamma}(1) = y$. The statement follows since the projection $\gamma = \pi \circ \hat{\gamma}$ of $\hat{\gamma}$ is a loop in U. □

3.5.2 Monodromy Solve

In the previous subsection, we assumed that the variety Y is irreducible. However, this is not necessarily the case. In general, Y has finitely many top-dimensional irreducible components $Y_1, \dots, Y_\ell$. For each irreducible component Y_k there exists an open set $U_k \subseteq Q$ such that for every $q \in U_k$ the fiber $\pi|_{Y_k}^{-1}(q)$ has the same cardinality d_k. Also note that for each Y_k the map $\pi|_{Y_k} : Y_k \to Q$, is dominant (its image is dense). The results from the previous subsection hold for each component Y_k separately.

Given an initial point $(x_0, p_0) \in Y_k$ with $p_0 \in U_k$, we want to fill the fiber $\pi|_{Y_k}^{-1}(p_0)$. For this, we use the monodromy group associated with the fiber by repeatedly tracking solutions of F_{p_0} around loops γ_i based at p_0.

If $Q = \mathbb{C}^m$, then we can construct a loop γ by choosing randomly $p_1, p_2 \in \mathbb{C}^m$ and define

$$\gamma(t) = \begin{cases} (1-3t)p_0 + 3tp_1 & 0 \leq t \leq \frac{1}{3} \\ (2-3t)p_1 + (3t-1)p_2 & \frac{1}{3} < t \leq \frac{2}{3} \\ (3-3t)p_2 + (3t-2)p_0 & \frac{1}{3} < t \leq \frac{2}{3} \end{cases}$$

The image $\gamma([0,1])$ of the constructed loop is with probability one contained in U_k following Lemma 3.3. The algorithm doesn't require that $Q = \mathbb{C}^m$, but then it is necessary to provide an oracle that generates loops $\gamma(t) : [0,1] \to U_k$ based at p_0. We track a zero of F_{p_0} along $\gamma(t)$ by performing the parameter homotopy $H(x,t) = F(x; \gamma(t))$.

We start with the set $S = \{x_0\}$ and a first loop γ_1. We track each element of S along the loop γ_1 resulting in the set S'. If $S \neq S'$, then the procedure is repeated with $S := S \cup S'$. This continues until $S = S'$. At this point, we can generate a new loop γ_2 and repeat the above procedure with the difference that the solutions are now tracked along the loop γ_2. This procedure can be continued infinitely but at some point the set $S \times \{p_0\}$ and $\pi|_{Y_k}^{-1}(p_0)$ are identical and we successfully filled the fiber. If we know the correct number of solutions, then we can stop the computation as soon as the cardinality of S reached this number. Otherwise, we have to rely on a heuristic stopping criterion. The most common heuristic is to stop the procedure if for K loops no new solutions were found. A typical value for K is 5. In certain situations, it is possible to avoid the heuristic stopping criterion and to use instead a *trace test* as stopping criterion. See Section 3.6 for a description of the trace test. Various improvements to the outlined algorithm are presented in [DHJ+18].

The monodromy method is only a probabilistic method. Thus, what is the expected number of loops we need to generate until the fiber is saturated? In [DHJ+18], this question is partially addressed. For this, the authors consider the equivalent problem of what the expected number ℓ of loops $\gamma_1, \dots, \gamma_\ell$ is such that the loops generate the monodromy group associated to $\pi|_{Y_k}^{-1}(p_0)$. For their result, they have the very simplified assumptions that the monodromy group is the full symmetric group on d_k elements and that the monodromy loops γ_i are picked uniformly. With these assumptions, they show that the expected value is finite and asymptotically approaches 2 as $d_k \to \infty$. In practice, these simplified assumptions are often not given and the number of loops necessary varies depending on the particular problem.

3.5.3 Practical Considerations

The monodromy method provides a probabilistic method to compute isolated zeros of a parameterized polynomial system $F(x;p)$. To compute all isolated solutions of F for a general parameter value, this method requires that the incidence variety Z defined in (3.9) has a single top-dimensional irreducible component Y. For many applied problems, this is the case and so it is reasonable to start from this assumption. Additionally, if F is linear in the parameters, then Z always has only a single top-dimensional irreducible component [DHJ$^+$18, Remark 2.2]. Even if there is not only a single top-dimensional irreducible component the monodromy method is still applicable but the result depends on the component the initial point lies on. This is not necessarily a downside of the method. Sometimes there are multiple irreducible top-dimensional components but only one contains isolated solutions relevant to the particular problem. In this case, the monodromy method allows computing only the relevant isolated solutions.

The monodromy method has a major advantage compared to a direct method as, e.g, the polyhedral homotopy. It is much more resilient against numerical problems. If in a polyhedral homotopy a path cannot be tracked due to numerical problems, then this results possibly in a missing solution that can only be recovered by rerunning the whole computation. In contrast, if a solution cannot be tracked along a loop during the monodromy method, then the missed solution can still be recovered as the result of another, numerically better behaved, loop at some later point. Another advantage of the monodromy method is that the path tracking is simpler since by design no solution path diverges or has a singular endpoint.

Algorithm 3.10 FindStartPair

Input: A parameterized polynomial system $F : \mathbb{C}^n \times \mathbb{C}^m \to \mathbb{C}^N$ and a maximal number of tries M.
Output: If successful, an approximation of pair $(x, p) \in \mathbb{C}^n \times \mathbb{C}^m$ such that $F(x;p) = 0$. Otherwise, `false`.

procedure FindStartPair(F, M)
  Consider F as a system $\hat{F}$ in $n+m$ variables and with no parameters.
  $k \leftarrow 0$
  while $k < M$ **do**
    Construct $y \in \mathbb{C}^{n+m}$ by drawing each entry from a complex Normal distribution
    Perform generalized Newton method for $\hat{F}$ starting at y
    if Newton method successfull **then**
      Return the last Newton iterate as a start pair (x, p)
    end if
    $k \leftarrow k + 1$
  end while
  Return `false`
end procedure

To start the monodromy method, an initial *start pair* $(x_0, q_0) \in Z$ is necessary. We assume in the following $Q = \mathbb{C}^m$. If for a general $x_0 \in \mathbb{C}^n$ the map $p \to F(x_0;p)$ is affine linear, then a start pair can be computed using linear algebra. Otherwise, a start pair can also often be constructed by insight into the concrete problem. Assuming that Y has

only a single top-dimensional irreducible component, it is also possible to use the algorithm outlined in Algorithm 3.10. The algorithm performs for a parameterized polynomial system a random search for a start pair by repeatedly sampling points and performing Newton's method. Although the algorithm is very simple, for many problems a start pair is reliably found with less than 200 tries.

3.6 Certification and Trace Test

So far, we discussed numerical methods to compute (all) isolated zeros of a polynomial system F. Since we use numerical methods, we do not obtain exact zeros of F but rather numerical approximations. For many applications, this is sufficient. But for some applications, in particular in pure mathematics, we want to *certify* that the obtained approximations correspond to actual zeros of F. Additionally, we want to certify that we found a certain number of distinct zeros, establishing a *lower bound* on the number of solutions of F.

It is of great interest to also certify that we found *all* isolated solutions of F. Unfortunately, this is so far only possible if an upper bound is already established and the lower bound obtained from the certification routine matches the upper bound. Establishing an upper bound for a polynomial system based on a computational method that does not involve Gröbner basis computations is an important open problem. A numerical method to test whether all solutions are found is given by the *trace test*. However, the trace test does not produce a rigorous certificate. The result of the trace test can be interpreted similarly to the numerical computation of the smallest eigenvalue of a matrix. If the computed smallest eigenvalue is on the order of the machine precision, then you probably conclude that the matrix is singular. But this is *not* a proof (or certificate) that the matrix is singular.

3.6.1 Certification

We focus on two strategies to certify solutions to square polynomial systems. Smale's α-theory [Sma86] and Krawczyk's method [Moo77]. The restriction to square polynomial systems is necessary since to certify solutions to overdetermined systems additional global information is necessary. In [DHS20], the authors develop various techniques to certify overdetermined systems requiring different global information. A certification technique based on Krawczyk's method is presented in Chapter 5. There, we also demonstrate an implementation of Krawczyk's method in `HomotopyContinuation.jl` that provides a significant computational improvement over the existing certification methods based on Smale's α-theory. However, since Smale's α-theory only requires data from one point, it has valuable features for the theory of computation. In particular, it is the building block for the complexity analysis of polynomial homotopy continuation methods [BP09, BC11, BL13, Lai17] and certified path tracking algorithms [BL13].

In [Sma86], Smale introduced the notion of an *approximate zero*, the α-number and the

α-theorem. For a square polynomial system F in n variable, consider the Newton iteration

$$\begin{aligned} J_F(x^{(j)})\Delta x^{(j)} &= F(x^{(j)}) \\ x^{(j+1)} &= x^{(j)} - \Delta x^{(j)} \end{aligned} \quad , \quad j = 0, 1, 2, \ldots$$

starting at the initial guess $x^{(0)} \in \mathbb{C}^n$ where J_F is the Jacobian of F. An approximate zero of F is any point $x^{(0)} \in \mathbb{C}^n$ such that Newton's method converges quadratically towards a zero of F. This means that the number of correct significant digits roughly doubles with each iteration of Newton's method.

Definition 3.11 (Approximate zero)**.** The point $x^{(0)} \in \mathbb{C}^n$ is an approximate zero of F if the Newton iterates $x^{(j)}$ are defined for $j = 1, 2, \ldots$ and satisfy

$$\|\Delta x^{(j)}\| \leq \left(\frac{1}{2}\right)^{2^j - 1} \|\Delta x^{(0)}\| .$$

If $x^{(0)}$ is an approximate zero, then the true zero $x^* \in \mathbb{C}^n$ of F to that the iterates are converging is the *associated zero* of $x^{(0)}$.

Smale's α-theorem gives a sufficient condition for $x^{(0)}$ to be an approximate zero. The theorem uses

$$\gamma(F, x) \;=\; \sup_{k \geq 2} \left\| \frac{1}{k!} J_F(x)^{-1} D^k F(x) \right\|^{\frac{1}{k-1}} \quad \text{and } \beta(F, x) \;=\; \|J_F(x)^{-1} F(x)\| \qquad (3.10)$$

where $D^k F$ is the tensor of order-k derivatives of F and the tensor $J_F^{-1} D^k F$ is understood as a multilinear map $A : (\mathbb{C}^n)^k \to \mathbb{C}^n$.

Theorem 3.12 (Smale's α-theorem [Sma86])**.** *There is a naturally defined number α_0 approximately equal to 0.1307 such that if $\alpha(F, x^{(0)}) := \beta(F, x^{(0)})\, \gamma(F, x^{(0)}) < \alpha_0$, then $x^{(0)}$ is an approximate zero of F.*

To avoid the computation of the γ-number, Shub and Smale [SS93] derived an upper bound for $\gamma(F, x)$ that can be computed exactly and efficiently. Hence, one can decide algorithmically whether x is an approximate zero using only the data of the point x itself and F. Hauenstein and Sottile implemented these ideas in the software `alphaCertified` [HS12].

3.6.2 Trace Test

The trace test was first introduced in [SVW02] by Sommese, Verschelde, and Wampler and recently generalized in [LRS18] by Leykin, Rodriguez, and Sottile to subvarieties of products of projective spaces. Suppose we have an irreducible one-dimensional variety $X \subseteq \mathbb{C}^n$. The intersection of X with a general hyperplane L consists of $\deg(X)$-many points. The *trace* of X with respect to the hyperplane L is the coordinate-wise sum of the points in $L \cap X$. Let $l(x)$ be an affine linear polynomial defining L. Denote by L_t the zero set of $l(x) + t$. L_t is a hyperplane that depends linearly on t and the trace of X with respect to L_t depends on t. We have the following result.

Proposition 3.13. *[SVW02, LRS18] Using the notation and definitions above, the trace of X with respect to L_t is affine linear in t. Moreover, the coordinate-wise sum of any non-empty proper subset of $X \cap L_t$ is not affine linear in t.*

This leads to the idea of the trace test. Given a subset $W \subseteq X \cap L$, denote by W_t the points obtained by tracking W from $X \cap L_0$ to $X \cap L_t$. By construction we have $W = W_0$. Also, denote by $w(t)$ the sum of the points of W_t. Following Proposition 3.13, $w(t)$ is an affine linear function if and only if $W_t = X \cap L_t$. Since W_t is the result of tracking W along $X \cap L_t$, it follows that $w(t)$ is an affine linear function if and only if $W = X \cap L$. The trace test computes for a general $\tau \in \mathbb{C} \setminus \{0\}$ the points $w(0)$, $w(\tau)$ and $w(-\tau)$ and tests whether these three points are colinear. By our previous argument, this is with probability one only the case if $W = X \cap L$. A numerical stable method to test this is to compute the singular values $\sigma_1 \geq \sigma_2 \geq \sigma_3$ of the matrix

$$\begin{bmatrix} w(0) & w(\tau) & w(-\tau) \\ 1 & 1 & 1 \end{bmatrix}.$$

Here, a row of ones is appended to account for the case that X is a plane curve. The smallest singular value σ_3 is zero if $w(t)$ is affine linear. Thus, a good numerical colinearity test is to check that for the computed singular values the value of σ_3/σ_1 is on the order of the machine precision.

Above we assumed that $X \subseteq \mathbb{C}^n$ is a one-dimensional affine variety. But this is not necessary and the trace test can be extended to positive-dimensional irreducible affine varieties by replacing L_t with a *pencil of affine linear spaces* M_t such that $\operatorname{codim}(M_0) = \dim(X)$. A pencil of affine linear spaces is a family M_t, $t \in \mathbb{C}$, of affine linear spaces that depends affinely on the parameter t. M_t is the span of an affine linear space H and a point t on a line h that is disjoint from H.

So far, the presented trace test does not apply to isolated solutions of a general parameterized polynomial system $F(x;p)$. For this, we consider again for some irreducible variety $Q \subset \mathbb{C}^m$ the incidence variety

$$Z = \{(x,p) \in \mathbb{C}^n \times Q \,|\, F(x;p) = 0\} \subseteq \mathbb{C}^n \times Q\,.$$

and the projection $\pi : Z \to Q, (x,p) \mapsto p$ onto the second factor. We consider an irreducible subvariety $Y \subseteq Z$ with the property that there exists an open set $U \subseteq Q$ such that for all $q \in U$ the fiber $\pi|_Y^{-1}(q)$ is finite and has the same cardinality d. In Section 3.5, we demonstrated how the monodromy method allows us to compute all elements of $\pi|_Y^{-1}(q)$ for $q \in U$. But we had the difficulty that we needed a heuristic to stop the computation.

If U is non-empty, then the variety $Y \subseteq \mathbb{C}^n \times Q$ is an m-dimensional irreducible variety. Consider the closure $\overline{Y}$ of Y in the product $\mathbb{P}^n \times \mathbb{P}^m$ of projective spaces $\overline{Y}$. In [LRS18], the authors give a trace test for subvarieties of products of projective spaces. Dehomogenizing the procedure allows us to give a trace test for Y. We follow the approach outlined in [MdCR17].

In a first step, we intersect Y with a linear space Λ of codimension $m-1$ defined by $m-1$ general affine linear polynomials G in p resulting in the one-dimensional subvariety $Y \cap \Lambda \subseteq Y \subseteq \mathbb{C}^n \times \mathbb{C}^m$. This dimension reduction is an application of a version of Bertini's

theorem [LRS18, Theorem 12] and preserves the irreducibility of Y. Whereas we previously intersected with an affine linear space, we now intersect with an affine bilinear space $\mathcal{B}$ defined by the product of two affine linear polynomials $\ell_1(x)$ and $\ell_2(x)$ in the unknowns x and p respectively. Since we reduced to a curve in $\mathbb{C}^{n+m} \cong \mathbb{C}^n \times \mathbb{C}^m$, we can apply again Proposition 3.13. That is, if we define the family $\mathcal{B}_t$ of bilinear spaces by $\ell_1(x)\ell_2(p) + t$, then the trace of $Y \cap \Lambda$ with respect to $\mathcal{B}_t$ is affine linear in t for the x and p coordinates. Moreover, the coordinate-wise sum for any proper non-empty subset of $Y \cap \Lambda \cap \mathcal{B}_t$ is not affine linear in t.

Now assume we attempted to compute with the monodromy method all elements of $\pi|_Y^{-1}(q)$ for some $q \in U$ and obtained as a result the subset $W \subseteq \pi|_Y^{-1}(q)$. To check for the completeness of W, we want to apply the trace test. For this, we need to choose $G(p)$ and $\ell_2(p)$ such that q is the unique solution of $G(p) = \ell_2(p) = 0$. After choosing a general affine linear polynomial $\ell_1(x)$, we need to start a *second* monodromy computation for the polynomial system

$$T(x, p; t) = \begin{bmatrix} F(x, p) \\ G(p) \\ \ell_1(x)\ell_2(p) + t \end{bmatrix}$$

at the parameter value $t = 0$. The points $W \times \{q\}$ are solutions to T_0 and serve as start solutions for the monodromy computation. At some point the monodromy computation recovers $Y \cap \Lambda \cap \mathcal{B}_0 \subseteq \mathbb{C}^n \times \mathbb{C}^m$. This can be verified by testing that the trace of $Y \cap \Lambda$ with respect to $\mathcal{B}_t$ is colinear. Finally, if the trace is colinear and $W \times \{q\} = Y \cap (\mathbb{C}^n \times \{q\})$, then $W = \pi|_Y^{-1}(q)$ and the first monodromy computation found all solutions.

3.7 Witness Sets

So far, we focused on computing isolated solutions of a polynomial system. Now we want to briefly discuss the numerical solution of polynomial systems F whose zero sets are positive-dimensional. Before we can compute a positive-dimensional zero set $\mathcal{V}(F) \subseteq \mathbb{C}^n$, we need a suitable data structure to represent $\mathcal{V}(F)$ on a computer. This data structure is a *witness set* [SW05, Section 13.3].

Definition 3.14 (Witness set)**.** A witness set for an irreducible variety $X \subseteq \mathbb{C}^n$ is a triple (F, L, S) where F is polynomial system such that X is an irreducible component of $\mathcal{V}(F)$, $L \subseteq \mathbb{C}^n$ is a general affine linear space with $\operatorname{codim}(L) = \dim(X)$ and $S = X \cap L$.

If X is a reducible variety with top dimensional irreducible components $X_1, \ldots, X_\ell$, then a witness set for X is the triple (F, L, S) where $S = S_1 \cup \cdots \cup S_\ell$ such that (F, L, S_i) is a witness set for X_i.

In a witness set $W = (F, L, S)$ for a variety $X \subseteq \mathcal{V}(F)$, we refer to L as a *witness slice* and to S as *witness points*. If L is a general affine linear space, then the cardinality of S is the degree $\deg(X)$ of X.

We can obtain a witness set for the k-dimensional components of $\mathcal{V}(F)$ by computing the isolated solutions S of $F(x) = Ax + b = 0$ where $A \in \mathbb{C}^{(n-k)\times n}$, $b \in \mathbb{C}^{n-k}$ and

$L = \mathcal{V}(Ax + b)$. To verify that the tuple (F, L, S) constitutes a witness set W, we can use the trace test described in Section 3.6.2. Given a witness set $W = (F, L, S)$, we can obtain a witness set for a different linear space L' by moving L onto L' and tracking the solutions S as the linear space is moved. The result is the solution set S' and another witness set $W' = (F, L', S')$.

It is also possible to compute a collection of witness sets $W_1, \ldots, W_\ell$ where each W_i corresponds to an irreducible component X_i of the irreducible decomposition $X_1 \cup \cdots \cup X_\ell = \mathcal{V}(F)$. The methods to compute such a collection are of great importance for numerical algebraic geometry but they are not a focus of this thesis. We refer the interested reader to [SW05, Chapter 13–15].

In applications, many varieties of interest are defined as projections of other varieties. Because witness sets are geometric, they behave well with respect to projections. Consider an irreducible variety $Z \subseteq \mathcal{V}(F) \subseteq \mathbb{C}^n$ and a coordinate projection $\pi : \mathbb{C}^n \to \mathbb{C}^k$. We are interested in $X = \overline{\pi(Z)}$. This is an irreducible variety since Z is irreducible. Getting defining equations for $\overline{\pi(Z)}$ is a very challenging symbolic computation. However, a *pseudo-witness set* allows us to perform computations with $\overline{\pi(Z)}$ without the need to compute defining equations for it. The notion of pseudo-witness set was introduced in [HS10].

Definition 3.15. A *pseudo-witness set* for a variety $X \subseteq \mathbb{C}^k$ defined as the Zariski closure $\overline{\pi(Z)}$ of an irreducible variety $Z \subseteq \mathcal{V}(F)$ with $\dim(Z) = \dim(X)$ and coordinate projection $\pi : \mathbb{C}^n \to \mathbb{C}^k$ is the quadruple (F, π, L, W) where $W = X \cap \pi^{-1}(L)$.

Remark 3.16. A pseudo-witness set actually performs computations 'upstairs' on Z and not on X.

Often, it is the case that a variety $X \subseteq \mathbb{C}^k$ is described as as the image $X = \overline{\varphi(Z)}$ of a projection $\varphi : \mathbb{C}^n \to \mathbb{C}^k$ where the dimension of Z is larger than its image. To construct a pseudo-witness set for X, it is sufficient to reduce Z to a lower dimensional variety by intersecting it with a linear space $M \subseteq \mathbb{C}^n$ of correct dimension such that $X = \overline{\varphi(Z \cap M)}$ and $\dim(Z \cap M) = \dim(X)$.

3.8 Conclusion

In this chapter, we discussed the foundations for the numerical solution of polynomial systems. First, we focused on the computation of isolated solutions of a polynomial system. For this, we introduced the general framework of a parameter homotopy. A parameter homotopy is a powerful tool that allows us to find all isolated zeros of a polynomial system by embedding the system into a suitable family of polynomial systems. We discussed two cases of the parameter homotopy that apply to any square polynomial system. These two cases are the total degree and the polyhedral homotopy. We also described the monodromy method. The monodromy method is a probabilistic method for computing the isolated zeros of a polynomial system that allows us to solve problem which are infeasible with a total degree or polyhedral homotopy. Complementary to the computation of isolated solutions, we discussed the certification of the obtained numerical solutions. Similarly, we introduced the concept of a trace test. The trace test gives us a possibility to verify, but not to certify, that we found all isolated solutions. Finally, we also discussed the concept of a witness set that allows describing positive-dimensional varieties. Together, these concepts give us a solid foundation to solve a wide range of problems appearing in applications.

4 Mixed Precision Path Tracking

This chapter is based on the article "Mixed Precision Path Tracking for Polynomial Homotopy Continuation" [Tim20] by Sascha Timme. The article is currently under review. A preprint is available at `https://arxiv.org/abs/1902.02968`.

Recall from Chapter 3 that a critical step in the homotopy continuation method is to track a solution $x_0 \in \mathbb{C}^n$ of a homotopy $H(x,t) : \mathbb{C}^n \times \mathbb{C} \to \mathbb{C}^n$ at $t = 0$ to a solution at $t = 1$. A solution x_0 at $t = 0$ gives rise to a solution path $x(t)$ implicitly defined by the conditions

$$H(x(t),t) = 0 \text{ for all } t \in [0,1) \text{ and } x(0) = x_0 \,. \tag{4.1}$$

In order to track a path $x(t)$, the problem (4.1) is treated as a sequence of problems

$$H(x(t_k),t_k) = 0 \,, \quad k = 0,1,2,\ldots \tag{4.2}$$

with an (a-priori unknown) subdivision $0 = t_0 < t_1 < \ldots < t_M = 1$ of the interval $[0,1]$. Each of the problems in (4.2) is then solved by a *correction method*, usually Newton's method, under the assumption that a *prediction method*, e.g., Euler's method, provides a good starting point. Often, the choice of step size $\Delta t_k = t_{k+1} - t_k$ is given by an *adaptive step size control*. The step size must be chosen appropriately: if the step size is too large, the prediction can be outside the zone of convergence of the corrector, while a too small step size means progress is slow. There have been many efforts to design such adaptive step size controls [SC87, KX94, GS04].

In the context of polynomial homotopy continuation methods, two phenomena need particular attention. Polynomial systems often have singular solutions, and thus, the paths leading to these solutions are necessarily ill-conditioned at the end. While endgame methods [BHS11b, MSW90, MSW92b, MSW92a] exist to compute singular solutions, these still require to track the solution path sufficiently close to the singularity. Usually, homotopies guarantee, with probability one, that no path passes through a singularity before reaching its endpoint. However, there is a non-negligible chance that a near-singular condition is encountered during the tracking.

Also, if two different solution paths are near to each other, then this can cause *path jumping*. That is, the solution that is tracked 'jumps' from one path to another. The typical reason is that starting from a point on the tracked path, the prediction method returns a point that is, according to the correction method, a numerical approximation of a point on a different path. A possible result of path jumping is that not all isolated solutions of a polynomial system are computed. Recently, Telen, Van Barel and Verschelde [TVBV20] introduced an algorithm that is very robust against path jumping. An implementation of this

algorithm is available in the software package `PHCpack` [Ver99].

Therefore, path tracking algorithms are required to reduce the risk of path jumping and they need to be able to handle ill-conditioned situations during the tracking. Existing software packages, e.g., `Bertini` [BHSW] and `PHCpack`, use a version of the following path tracking algorithm. The algorithm has the following parameters: An initial step size $\Delta t > 0$, a number of corrector iterations $N \geq 1$ allowed per step, a step adjustment factor $\lambda \in (0, 1)$, a step expansion integer $E \geq 1$, and a minimum step size $\Delta t_{\min}$. Additionally, there is a tracking tolerance $\varepsilon > 0$. This means that for a given t an approximate solution $x \approx x(t)$ has to satisfy a normwise absolute error $\|x - x(t)\|_\infty \leq \varepsilon$.

Given an approximate solution $x \approx x(t)$, the prediction method provides an initial guess $\hat{x}(\Delta t) \approx x(t + \Delta t)$. Then, Newton's method iteratively improves the approximation $\hat{x}(\Delta t)$. If the required tracking tolerance ε is achieved with a most N iterations, then the solution is updated and $t = t + \Delta t$. If there are E successes in a row, then the step size is expanded to $\Delta t = \lambda^{-1}\Delta t$. If on the other hand the tolerance is not achieved with at most N iterations, then the step size is reduced to $\Delta t = \lambda \Delta t$. If $\Delta t < \Delta t_{\min}$, the algorithm terminates with a failure. Otherwise, the procedure is repeated until $t = 1$ is reached.

The key to avoiding path jumping is to allow only a small number of Newton iterations, typically only $N = 2$ or $N = 3$. In practice, this is often sufficient for the initial guess $\hat{x}(\Delta t)$ to stay within a small enough region surrounding the path such that no path jumping occurs. However, if two paths are closer than the required tracking tolerance ε for some $t^* \in (0, 1)$, then this algorithm tends to fail for these paths. This is shown in the computational experiments in Section 4.4 for two different examples. Therefore, it is necessary to choose the path tracking tolerance ε smaller than the minimal pairwise distance of any two paths. However, knowing the optimal choice of ε a priori is impossible. Thus, one has to use either a pessimistic value for ε or resort to trial and error. But choosing ε small does not only slow down the tracking of *all* paths, it also can result in new tracking failures. The reason for this is that Newton's method in floating-point arithmetic *cannot* always produce solutions whose relative normwise error is smaller than ε. This was shown by Tisseur in [Tis01] and is explained in detail in Section 4.1.2.

To avoid tracking failures due to insufficient precision, Bates, Hauenstein, Sommese, and Wampler [BHSW08] developed an adaptive precision version of the above-described path tracking algorithm. During the tracking, the algorithm dynamically changes the working precision such that Newton's method can theoretically always produce solutions accurate enough for the desired tracking tolerance. This eliminates the problem of insufficient precision in exchange for a possibly high computational cost. But it also still leaves open the problem of picking a suitable tolerance ε.

In this chapter, we introduce a new path tracking algorithm, Algorithm 4.14, that does not require the choice of a path tracking tolerance ε or a maximal number N of corrector iterations allowed per step. This allows the algorithm to handle numerically challenging situations. The key idea is to use a more intrinsic measure for accepting an initial guess in the Newton corrector: An initial guess $\hat{x}(\Delta t)$ should only be accepted if the Newton iterates

$\hat{x}(\Delta t) = x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ satisfy

$$\|x^{(j+1)} - x^{(j)}\| \leq a^{2^j - 1}\|x^{(1)} - x^{(0)}\| \tag{4.3}$$

for $j = 1, 2, \ldots$ and some fixed constant $a \in (0, \frac{1}{2}]$. If the initial guess satisfies (4.3), then it is an *approximate zero*. This notion was introduced by Smale [Sma86] for $a = \frac{1}{2}$ and plays an important role in the complexity analysis of polynomial homotopy continuation methods [BC11, Lai17]. Based on this idea, we develop in this Chapter a new Newton corrector algorithm, Algorithm 4.5. The algorithm rejects an initial guess if (4.3) is not satisfied for some $j = 1, \ldots, m$ where m is dynamically chosen as the maximal number of iterations for which (4.3) can be satisfied in fixed precision floating-point arithmetic.

The proposed path tracking algorithm combines the new Newton corrector algorithm with an adaptive step size control that chooses Δt based on local geometric information. The step size control extends an adaptive step size control developed by Deuflhard [Deu79] and combines it with the insight of [TVBV20] to use Padé approximants as prediction methods. In particular, the algorithm builds a local understanding of the region of convergence of Newton's method and following Telen, Van Barel and Verschelde [TVBV20] obtains an estimate of the distance to the closest singularity. This keeps the risk of path jumping low, but the algorithm cannot guarantee that path jumping does not happen. To handle numerically challenging situations, the algorithm uses *mixed-precision* arithmetic. That is, while the bulk of the computations is performed in double precision some computations are performed, if necessary, in extended precision. A version of this algorithm is implemented `HomotopyContinuation.jl` .

This chapter is organized as follows. Section 4.1.1 reviews a Kantorovich style convergence theory of Newton's method and Section 4.1.2 develops a new Newton corrector algorithm, Algorithm 4.5, based on requirement (4.3). In Section 4.2, the use of Padé approximants as prediction method is developed. In Section 4.3, the results from the previous sections are used to develop an adaptive step size control. Finally, the new path tracking algorithm, Algorithm 4.14, is stated. The algorithm's effectiveness and ability to handle challenging paths is shown through several numerical experiments in Section 4.4.

4.1 Newton's Method: Theory and Computational Aspects

The path tracking algorithm for a path $x(t)$ consists of three main components: An adaptive step size routine that provides a step size Δt, a predictor that produces an initial guess $\hat{x}$ of $x(t + \Delta t)$, and a corrector that takes $\hat{x}$ and returns either an approximation of $x(t + \Delta t)$ or rejects $\hat{x}$. This section focuses on Newton's method as a corrector. The goal is to understand the size of the region of convergence of Newton's method as well as the behavior of Newton's method in floating-point arithmetic and to translate this into a Newton corrector algorithm.

4.1.1 Convergence results

Let $D \subseteq \mathbb{C}^n$ an open set. Let $F : D \subseteq \mathbb{C}^n \to \mathbb{C}^n$ be an analytic function and $J_F : \mathbb{C}^n \to \mathbb{C}^n$ its Jacobian. Consider the Newton iteration

$$\begin{aligned} J_F(x^{(j)})\Delta x^{(j)} &= F(x^{(j)}) \\ x^{(j+1)} &= x^{(j)} - \Delta x^{(j)} \end{aligned}, \quad j = 0, 1, 2, \ldots \tag{4.4}$$

starting at the initial guess $x^{(0)} \in D$. In 1948 Kantorovich [Kan48] already showed sufficient conditions for the convergence of Newton's method and the existence of solutions. He also showed the uniqueness region of solutions and provided error estimates. A particular property of Newton's method is that the iterates (4.4) are invariant under general linear transformations of F. That is, given a start value $x^{(0)} \in D$ and $A \in \mathrm{GL}_n(\mathbb{C})$ the Newton iterates of $AF(x)$ and $F(x)$ coincide. This property is referred to as *affine covariance* [Deu11]. In the following, an affine covariant version of a Kantorovich style convergence theorem for Newton's method is given. The statement is due to Deuflhard and Heindl [DH79] with error bounds from Yamamoto [Yam85].

Theorem 4.1 (Newton-Kantorovich [DH79, Yam85])**.** *Let $F : D \subseteq \mathbb{C}^n \to \mathbb{C}^n$ be analytic. For some $x^{(0)} \in D$, assume that $J_F(x^{(0)})$ is invertible and that for all $x, y \in D$*

$$\begin{aligned} &\|J_F(x^{(0)})^{-1}(J_F(x) - J_F(y))\| \leq \omega\|x - y\|, \\ &\|\Delta x^{(0)}\| = \|J_F(x^{(0)})^{-1}F(x^{(0)})\| \leq \beta \end{aligned}$$

and $h_0 := \omega\beta \leq \frac{1}{2}$.
Let $r^* = (1 - \sqrt{1 - 2h_0})/\omega$ and $\bar{S}(x^{(0)}, r^*) = \{x \,|\, \|x - x^{(0)}\| \leq r^*\} \subseteq D$. Then:

1. *The iterates (4.4) are well-defined, remain in $\bar{S}(x^{(0)}, r^*)$ and converge to a solution x^* of $F(x) = 0$.*
2. *The solution is unique in $S(x^{(0)}, r^{**}) \cap D$ where $r^{**} = (1 + \sqrt{1 - 2h_0})/\omega$.*

Furthermore, assume $h < \frac{1}{2}$ and define the recursive sequence $h_j = \frac{h_{j-1}^2}{2(1-h_{j-1})^2}$. Then also the following error estimates hold.

$$\|\Delta x^{(j)}\| \leq \frac{1}{2}\omega\frac{\sqrt{1 - 2h_j}}{\sqrt{1 - 2h_0}}\|\Delta x^{(j-1)}\|^2, \quad j = 1, 2, 3, \ldots \tag{4.5}$$

$$\|x^{(j)} - x^*\| \leq \frac{2\|\Delta x^{(j)}\|}{1 + \sqrt{1 - 2\omega\frac{\sqrt{1-2h_j}}{\sqrt{1-2h_0}}\|\Delta x^{(j)}\|}}, \quad j = 0, 1, 2, \ldots \tag{4.6}$$

A drawback of the Newton-Kantorovich theorem is that it is not possible to obtain sufficient conditions for the convergence of Newton's method by only using data from the initial guess $x^{(0)}$. Instead, local information about the Lipschitz constant ω is required. The necessity of local information motivated Smale to develop his α-theory [Sma86], which only requires data from the initial guess $x^{(0)}$ to compute sufficient conditions for the convergence of Newton's method.

Recall from Definition 3.11 the definition of an approximate zero. A point $x^{(0)} \in \mathbb{C}^n$ is an approximate zero of F if all Newton iterates $x^{(j)}, j = 1, 2, \ldots,$ are defined and satisfy

$$\|\Delta x^{(j)}\| \leq \left(\frac{1}{2}\right)^{2^j-1} \|\Delta x^{(0)}\| .$$

Smale's α-Theorem 3.12 gives a sufficient condition for $x^{(0)}$ to be an approximate zero.

It is also possible to give sufficient conditions for $x^{(0)}$ to be an approximate zero under the assumptions of the Newton-Kantorovich Theorem 4.1.

Lemma 4.2. *Using notation from Theorem 4.1, assume*

$$h_0 = \omega\|\Delta x^{(0)}\| \leq 2(\sqrt{4a^4 + a^2} - 2a^2) =: h(a)$$

for a parameter $0 < a < 1$. *Then, the contraction factors*

$$\Theta_j := \frac{\|\Delta x^{(j+1)}\|}{\|\Delta x^{(j)}\|} \leq a^{2^j} \quad , j = 0, 1, 2, \ldots \tag{4.7}$$

and the error bounds

$$\|\Delta x^{(j)}\| \leq a^{2^j-1}\|\Delta x^{(0)}\| \quad , j = 1, 2, \ldots \tag{4.8}$$

are satisfied. In particular, $x^{(0)}$ *is an approximate zero if* $h_0 \leq \sqrt{2} - 1$.

Proof. From the error estimate (4.5) follows

$$\begin{aligned} \Theta_j &\leq \frac{1}{2}\omega\frac{\sqrt{1-2h_j}}{\sqrt{1-2h_0}}\|\Delta x^{(j)}\| \leq \frac{1}{2}\omega\sqrt{\frac{1}{1-2h_0}}\|\Delta x^{(j)}\| \\ &= \frac{1}{2}\omega\|\Delta x^{(0)}\|\sqrt{\frac{1}{1-2h_0}}\prod_{\ell=0}^{j-1}\Theta_j = \frac{h_0}{2\sqrt{1-2h_0}}\prod_{\ell=0}^{j-1}\Theta_\ell . \end{aligned} \tag{4.9}$$

From $h_0 \leq 2(\sqrt{4a^4 + a^2} - 2a^2) < \frac{1}{2}$ follows $\frac{h_0}{2\sqrt{1-2h_0}} \leq a$ and therefore

$$\Theta_j \leq a\prod_{\ell=0}^{j-1}\Theta_\ell \leq a a^{\sum_{\ell=0}^{j-1} 2^\ell} = a^{2^j} .$$

Statement (4.8) follows from (4.7) by observing

$$\frac{\|\Delta x^{(j)}\|}{\|\Delta x^{(0)}\|} = \prod_{\ell=0}^{j-1}\Theta_\ell \leq a^{2^j-1} .$$

□

The Newton-Kantorovich theorem and Smale's α-theorem both give sufficient conditions

for an initial guess to be an approximate zero. For the Newton-Kantorovich theorem, a (local) estimate of the Lipschitz constant ω needs to be obtained. For Smale's α-theorem, the constant γ, defined in (3.10), needs to be computed. The path tracking algorithm developed in this chapter is based on the Newton-Kantorovich theorem since a (rough) estimate of ω can be computed with almost no additional cost during the Newton iteration.

A computational estimate $[\omega]$ of ω is

$$[\omega] = 2\frac{\|\Delta x^{(1)}\|}{\|\Delta x^{(0)}\|^2}\,. \tag{4.10}$$

This can be seen as follows. Using the error estimate (4.5)

$$\|\Delta x^{(1)}\| \leq \frac{1}{2}\omega\sqrt{\frac{1-2h_1}{1-2h_0}}\|\Delta x^{(0)}\|^2$$

together with the observation $\frac{1-2h_1}{1-2h_0} = (1-h_0)^{-2}$ follows

$$\|\Delta x^{(1)}\| \leq \frac{1}{2}\omega\frac{1}{1-\omega\|\Delta x^{(0)}\|}\|\Delta x^{(0)}\|^2$$

and this is equivalent to

$$\frac{2\|\Delta x^{(1)}\|}{\|\Delta x^{(0)}\|^2 + 2\|\Delta x^{(0)}\|\|\Delta x^{(1)}\|} \leq \omega\,. \tag{4.11}$$

The computational estimate (4.10) is now obtained by upper bounding (4.11) with

$$\frac{2\|\Delta x^{(1)}\|}{\|\Delta x^{(0)}\|^2 + 2\|\Delta x^{(0)}\|\|\Delta x^{(1)}\|} \leq 2\frac{\|\Delta x^{(1)}\|}{\|\Delta x^{(0)}\|^2} = [\omega]\,.$$

4.1.2 Computational Aspects and Floating-Point Arithmetic

After establishing the theoretical foundations of Newton's method as well as a method to obtain a computational estimate of the Lipschitz constant ω, these results are now used to guide the development of a Newton corrector algorithm. For this, the behavior of Newton's method in floating-point arithmetic has to be taken into account.

Limit Accuracy. The following assumes the standard model of floating-point arithmetic [Hig02, section 2.3]

$$fl(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta), \quad |\delta| \leq u, \quad \operatorname{op} = +, -, *, /$$

where u is the unit roundoff. In standard double precision arithmetic, $u = 2^{-53} \approx 2.2 \cdot 10^{-16}$. In [Tis01], Tisseur analyzed the limit accuracy of Newton's method in floating-point arithmetic. Let $x^* \in \mathbb{C}^n$ be a zero of F with $J_F(x^*)$ non-singular, and let $x^{(0)} \in \mathbb{C}^n$ be an

approximate zero of F with associated zero x^*. In floating-point arithmetic, we have

$$x^{(j+1)} = x^{(j)} - (J_F(x^{(j)}) + E_j)^{-1}(F(x^{(j)}) + e_j) + \varepsilon_j$$

where

- e_j is the error made when computing the residual $F(x^{(j)})$,
- E_j is the error incurred in forming $J_F(x^{(j)})$ and solving the linear system for $\Delta x^{(j)}$,
- ε_j is the error made when adding the correction to $x^{(j)}$.

Assume that $F(x^{(j)})$ is computed in the possibly extended precision $\bar{u} \leq u$ before rounding back to working precision u and assume that there exists a function ψ depending on F, $x^{(j)}$, u and $\bar{u}$ such that

$$\|e_j\| \leq u\|F(x^{(j)})\| + \psi(F, x^{(j)}, u, \bar{u})\,.$$

Similarly, assume that the error E_j satisfies

$$\|E_j\| \leq u\phi(F, x^{(j)}, n, u)$$

for some function ϕ that reflects both the instability of the linear solver and the error made when forming $J_F(x^{(j)})$. Then the following statement holds [Tis01, Corollary 2.3].

Theorem 4.3 ([Tis01])**.** *Let $x^{(0)}$ be an approximate zero with associated zero x^*, $x^* \neq 0$, assume that $J_F(x^*)$ is non-singular, satisfies $u\,\kappa(J_F(x^*)) \leq \frac{1}{8}$ and assume that for all j*

$$u\|J_F(x^{(j)})^{-1}\|\phi(F, x^{(j)}, n, u) \leq \frac{1}{8}\,.$$

Then, Newton's method in floating-point arithmetic generates a sequence of iterates $x^{(j+1)}$ whose normwise relative error decreases until the first j for which

$$\frac{\|x^{(j+1)} - x^*\|}{\|x^*\|} \approx \frac{\|J_F(x^*)^{-1}\|}{\|x^*\|}\psi(F, x^*, u, \bar{u}) + u =: \mu(x^*, u, \bar{u})\,. \tag{4.12}$$

In the following, the value $\mu(x^*, u, \bar{u})$ is referred to as the *limit accuracy*. Theorem 4.3 shows that the limit accuracy is influenced by three factors: the working precision u, the accuracy of the evaluation of the residual (in possibly extended precision $\bar{u}$), and the conditioning of the Jacobian. The essential consequence of this is that Newton's method *cannot* always produce solutions whose normwise relative error is on the order of the working precision u. From the error estimate (4.5) follows that if for a given j

$$\|\Delta x^{(j)}\| \leq \frac{\omega\|\Delta x^{(j-1)}\|^2}{2\sqrt{1-2h_0}} \leq \mu(x^*, u, \bar{u})\|x^*\| \tag{4.13}$$

then the normwise relative accuracy of $x^{(j)}$ in floating-point arithmetic is only of order $\mu(x^*, u, \bar{u})$. Assume that for a given j (4.13) is satisfied. Then a computational estimate $[\mu]$ of $\mu(x^*, u, \bar{u})$ can be obtained by computing $\|\Delta x^{(j)}\|/\|x^{(j+1)}\|$.

The following lemma shows that using extended precision improves the limit accuracy.

Lemma 4.4. *For extended precision $\bar{u} \leq u$, it holds $\mu(x^*, u, \bar{u}) \approx \mu(x^*, u, u)\frac{\bar{u}}{u} + u$.*

Proof. From Section 4.3.2 in [BHSW08] follows that for a system of polynomials given as a straight line program $\psi(F, x^{(j)}, u, \bar{u})$ in Theorem 4.3 is a linear function in $\bar{u}$, i.e., $\psi(F, x^{(j)}, u, \bar{u}) = \frac{\bar{u}}{u}\psi(F, x^{(j)}, u, u)$. The statement then follows from (4.12). □

If the working precision u is standard double-precision arithmetic, then computing with extended precision can be accomplished by using double-double arithmetic. A double-double number is represented as an unevaluated sum of a leading double and a trailing double, resulting in a unit roundoff of $2^{-106} = u^2$. Bailey [Bai95] pioneered double-double arithmetic, and implementations are nowadays available for a wide variety of programming languages and architectures.

Assume that for a fixed parameter $a \in (0, \frac{1}{2}]$ the Newton iterates starting at the initial guess $x^{(0)}$ are required to satisfy the contraction factors

$$\Theta_j = \frac{\|\Delta x^{(j+1)}\|}{\|\Delta x^{(j)}\|} \leq a^{2^j} \;\; j = 0, 1, 2, \ldots .$$

If the Newton iterates are computed with precision $\bar{u} = u$ then (4.13) implies together with Lemma 4.2 that if

$$\omega\mu(x^*, u, u) > a^{2^k - 1}h(a)\|x^*\| \tag{4.14}$$

then there does not need to exist an initial guess $x^{(0)}$ such that the first k contraction factors are satisfied. Given a fixed k, for instance, $k = 2$, this gives a criterion when to use extended precision. Similarly, if the Newton iteration is performed with extended precision, then it is possible to use only working precision again if $\omega\mu(x^*, u, u) < a^{2^k - 1}h(a)$.

The working precision u is insufficient if the combination of the error in the evaluation of the Jacobian and the instability in the linear system solver become too large. In this case, a multi-precision path tracking algorithm as [BHSW08] is necessary. However, as demonstrated in Section 4.4, using only double-precision arithmetic for the linear system solver is sufficient for most applications. Nevertheless, even if the precision u is sufficient to achieve the limit accuracy, the analysis of Tisseur also shows that the convergence speed of Newton's method can decrease due to a too unstable linear system solver. In this case, the theoretical convergence speed may not be achieved, which in turn can lead to not satisfying the required contraction factors. To circumvent this, the Newton updates are improved using mixed-precision iterative refinement [Hig97] if $\bar{u} < u$. This stabilizes the linear system solver sufficiently to achieve the theoretical convergence speed.

Stopping Criteria. Criteria for stopping the Newton iteration are now derived. Assume that ω and the limit accuracy $\mu = \mu(x^*, u, \bar{u})$ are known. If for any j the contraction factor

$$\Theta_j = \frac{\|\Delta x^{(j+1)}\|}{\|\Delta x^{(j)}\|} \leq a^{2^j} \tag{4.15}$$

is not satisfied, then the iteration is stopped and the initial guess is rejected. The iteration

is stopped successfully at step j if the next update would be smaller than the limit accuracy, i.e.,

$$\frac{\omega\|\Delta x^{(j-1)}\|^2}{2\sqrt{1-2h(a)}} \leq \|x^{(j)}\|\mu\,. \tag{4.16}$$

An additional Newton update is computed to obtain a computational estimate of μ.

Scaling. Before the full Newton corrector algorithm is stated, a final point is addressed. So far, a simple rescaling of variables can change the behavior of the algorithm since ω, μ and $\|\Delta x^{(j)}\|$ are not invariant under rescaling of variables. Additionally, if $x^* = 0$ the accuracy needs to be measured with an absolute, and not a relative, normwise error. A rescaling of variables is formally the change of variables

$$y = D^{-1}x, \quad D = \operatorname{diag}(d_1, \ldots, d_n),\ d_i \in \mathbb{R}_{>0}\,.$$

With $x = (x_1, \ldots, x_n)$ and $|x_i| \neq 0$, the choice $d_i \approx |x_i|$ results in new coordinates y_i of unit order. To deal with the case $|x_i| = 0$ as well as with possible overflows in floating-point arithmetic, an absolute threshold value $d_{\min} > 0$ of the form

$$d_i = \max\{|x_i|, d_{\min}\} \tag{4.17}$$

has to be imposed. For instance, $d_{\min} = \max(\sqrt{u}\max_i |x_i|, u)$. To not introduce rounding errors, the scaling factors D should be powers of the floating-point radix β ($\beta = 2$ in the case of IEEE-754 floating-point standard arithmetic). Instead of performing the change of variables explicitly in Newton's method, the size of the Newton updates can also be measured with the *weighted error* $\|D^{-1}\Delta x^{(j)}\|$. Using the scaling factors D allows the algorithm to perform independent of the initial provided variable scaling (assuming that the initial scaling is not too extreme).

The Algorithm. Finally, a new Newton corrector algorithm, Algorithm 4.5, is stated. The algorithm builds on the results developed in this section. The idea of the algorithm is to reject an initial guess $x^{(0)}$ if the Newton iterates $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ do not satisfy

$$\|x^{(j+1)} - x^{(j)}\| \leq a^{2^j-1}\|x^{(1)} - x^{(0)}\|$$

for $j = 1, 2, \ldots, m$ and some fixed constant $a \in (0, \frac{1}{2}]$. Here, m is decided dynamically based on equation (4.16). The rejection of an initial guess is performed using the slightly stricter criterion (4.15). The algorithm needs as input estimates of the limit accuracy μ and the Lipschitz constant ω and returns updated estimates of these quantities. During the path tracking, estimates are available by using the returned estimates from the previous steps. What to do at the beginning of the tracking, if these estimates are not available, is addressed after the algorithm.

The algorithm requires estimates of the limit accuracy μ and the Lipschitz constant ω. During the path tracking, the computational estimates for both of them are available by using the computed estimates of the Newton corrector from the previous step. However, this leaves open what to do for the first step. There are two possibilities. If the start solution is the solution of a previous tracking, then computational estimates of μ and ω are already

Algorithm 4.5 Newton Corrector

Input: $F : \mathbb{C}^n \to \mathbb{C}^n$, $x^{(0)} \in \mathbb{C}^n$, $a \in (0, \frac{1}{2}]$, estimate $[\mu] > 0$ of the limit accuracy μ, estimate $[\omega] > 0$ of ω, evaluation precision $\bar{u} \leq u$, and positive scaling factors D such that the coordinates of $D^{-1}x^{(0)}$ are of unit order.

Output: Boolean indicating whether $x^{(0)}$ was accepted, approximation $\bar{x} \in \mathbb{C}^n$ of a zero x^* of F, updated estimate of the (limit) accuracy μ at x^*, updated estimate of ω, number of updates j and last contraction factor Θ_{j-2}.

procedure NEWTON($F, x^{(0)}, a, [\mu], [\omega], \bar{u}, D$)
  $j \leftarrow 0$
  while true do
    $r \leftarrow$ Evaluate F at $x^{(j)}$ with precision $\bar{u}$ and round result to precision u
    Solve $J_F(x^{(j)})\Delta x^{(j)} = r$
    $x^{(j+1)} \leftarrow x^{(j)} - \Delta x^{(j)}$
    if $j = 1$ **then**
      $[\omega] \leftarrow 2\frac{\|D^{-1}\Delta x^{(1)}\|}{\|D^{-1}\Delta x^{(0)}\|^2}$ ▷ Compute ω estimate
    end if
    if $j \geq 1$ **and** $\frac{\|D^{-1}\Delta x^{(j)}\|}{\|D^{-1}\Delta x^{(j-1)}\|} > a^{2^{j-1}}$ **then** ▷ Check sufficient contraction
      return (**false**, $x^{(j+1)}, [\mu], [\omega], j+1, \frac{\|D^{-1}\Delta x^{(j)}\|}{\|D^{-1}\Delta x^{(j-1)}\|}$)
    else if $\frac{\omega\|D^{-1}\Delta x^{(j)}\|^2}{2\sqrt{1-2h(a)}} \leq [\mu]$ **then** ▷ Approaching limit accuracy
      $r \leftarrow$ Evaluate F at $x^{(j+1)}$ with precision $\bar{u}$ and round result to precision u
      Solve $J_F(x^{(j+1)})\Delta x^{(j+1)} = r$
      $x^{(j+2)} \leftarrow x^{(j+1)} - \Delta x^{(j+1)}$
      $[\mu] \leftarrow \|D^{-1}\Delta x^{(j+2)}\|$ ▷ Update of the limit accuracy
      return (**true**, $x^{(j+2)}$, $[\mu]$, $[\omega]$, j+2, $\frac{\|D^{-1}\Delta x^{(j+1)}\|}{\|D^{-1}\Delta x^{(j)}\|}$)
    end if
    $j \leftarrow j + 1$
  end while
end procedure

available. If this is not the case, the following heuristic, Algorithm 4.6, to determine values for $[\mu]$ and $[\omega]$ proved to be helpful. The idea is to add a small perturbation to the provided start solution and to perform two Newton steps. If the perturbation is sufficiently small, then the perturbed solution still converges to the provided start solution, and an estimate of $[\omega]$ and $[\mu]$ can be obtained. As an added benefit, this provides a test to point out invalid start solutions, e.g., due to user error. For simplicity, it is assumed that it is sufficient to compute the residual with precision u.

Algorithm 4.6 Model Initialization Heuristic

Input: Candidate $x^{(0)} \in \mathbb{C}^n$, $a \in (0, 1)$, and scaling factors D such that the coordinates of $D^{-1}x^{(0)}$ are of unit order.

Output: Boolean indicating whether the initialization was successful, estimate $[\mu]$ of the limit accuracy μ of the associated zero of $x^{(0)}$, and an estimate $[\omega]$ of ω.

```
1: procedure ModelInitialization(x^(0), a, D)
2:     v ← ‖D^{-1} J_F(x^(0))^{-1} F(x^(0))‖ + u
3:     ε ← √v
4:     for k ← 1 : 3 do                          ▷ Try up to 3 different sizes of perturbations
5:         x̄ ← x^(0) + εD                        ▷ Add relative perturbation
6:         Δ_0 ← J_F(x̄)^{-1} F(x̄)
7:         x^(1) ← x̄ − Δ_0
8:         Δ_1 ← J_F(x^(1))^{-1} F(x^(1))
9:         x^(2) ← x^(1) − Δ_1
10:        if ‖D^{-1}Δ_1‖/‖D^{-1}Δ_0‖ < a then
11:            [ω] ← 2 ‖D^{-1}Δx^(1)‖ / ‖D^{-1}Δx^(0)‖^2
12:            [μ] ← ‖D^{-1}Δ_1‖
13:            return (true, [μ], [ω])
14:        else                                   ▷ ω larger than ε, reduce ε
15:            ε ← ε u^{2^{-k}}
16:        end if
17:    end for
18:    return (false, [μ], [ω])
19: end procedure
```

4.2 Predictors and Padé Approximants

After carefully studying the Newton corrector in the previous section, we now shift the attention to the predictor. Recall that the role of the predictor is to produce for a given step size Δt an initial guess $x^{(0)}$ such that the corrector converges sufficiently fast. The choice of the step size Δt will be addressed in the next section, but before it is essential to understand the influence of Δt on the distance of the initial guess $x^{(0)}$ to the solution path.

Consider the homotopy $H(x,t) : \mathbb{C}^n \times \mathbb{C} \to \mathbb{C}^n$ and a constant $\bar{t} > 0$. Given a solution $s \in \mathbb{C}^n$ of the system $H(x,0) = 0$, assume that there is a solution path $x(t) : [0,\bar{t}\,] \to \mathbb{C}^n$ implicitly defined by the conditions

$$H(x(t),t) = 0 \;\text{ for all } t \in [0,\bar{t}\,] \text{ and } x(0) = s\,. \tag{4.18}$$

Also assume that $H_x(x(t),t)$ is nonsingular for all $t \in [0,\bar{t}\,]$. Then, $x(t)$ can be extended to a holomorphic function with $H(x(t^*),t^*) = 0$ for all t^* in some nonempty open neighborhood of 0. Without loss of generality, in the following only the situation at $t = 0$ is considered, thus $\Delta t = t$.

A predictor generates a *prediction path* $\hat{x}(t) : [0,\bar{t}\,] \to \mathbb{C}^n$ with $\hat{x}(0) = x(0)$ and they can be classified by the local order of the prediction error $\|\hat{x}(t) - x(t)\|$.

Definition 4.7 (Local order of a predictor)**.** A predictor is of local order p if there exists a $\tau > 0$ and a constant $\eta_p \geq 0$ such that for all $t \in [0,\tau]$

$$\|\hat{x}(t) - x(t)\| \leq \eta_p t^p\,.$$

The constant τ is the *trust region* of the predictor.

Example. The Euler predictor $\hat{x}(t) = x(0) + t\dot{x}(0)$ is of local order $p = 2$ with $\tau = \bar{t}$ since

$$\|x(t) - \hat{x}(t)\| = \|x(t) - x(0) - t\dot{x}(0)\| \leq \frac{1}{2} \max_{t\in[0,\bar{t}]} \|\ddot{x}(t)\| t^2 \,.$$

There are many different families of predictors known in the literature, with the most famous ones probably being (embedded) Runge-Kutta methods. In [BHS11a], it is shown that for polynomial homotopy continuation higher-order Runge-Kutta methods are substantially more efficient than the Euler predictor. However, in the following another particular class of predictors is considered: Padé approximants. See [BGM96] for an exhaustive treatment of Padé approximants. The use of Padé approximants is strongly motivated by the recent results from Telen, Van Barel, and Verschelde [TVBV20] where a path tracking algorithm is developed that is very robust against path jumping. But it is to note that the use of Padé approximants as predictors was already considered in [SC87].

Definition 4.8 (Padé approximant)**.** Let $x(t) = \sum_{\ell=0}^{\infty} c_\ell t^\ell$ be a convergent power series. The type (L, M) Padé approximant is the rational function

$$[L/M]_x = \frac{a_0 + a_1 t + a_2 t^2 + \cdots + a_L t^L}{1 + b_1 t + b_2 t^2 + \cdots + b_M t^M}$$

such that $x(t)$ and $[L/M]_x$ (considered as formal power series) satisfy

$$[L/M]_x - x(t) \in O(t^{L+M+1}) \,.$$

Remark 4.9. A type (L, M) Padé approximant is a predictor of local order $L + M + 1$.

The following use of Fabry's ratio theorem is the key result from [TVBV20] that is used to obtain an estimate for the trust-region of a Padé predictor. Since $x(t)$ is holomorphic in a nonempty open neighborhood of 0, there is a coordinatewise expansion of $x(t)$ as a convergent power series around 0. Write $x_j(t) = \sum_{\ell=0}^{\infty} c_\ell t^\ell$ for the Taylor expansion of the coordinate function $x_j(t)$ at 0. For sufficiently large L, the pole of the Padé approximant $[L/1]_{x_j}$ indicates the *distance* to the nearest singularity (also if it is a branch point). This is seen as follows. A computation shows that if $c_L \neq 0$,

$$[L/1]_{x_j} = c_0 + c_1 t + \ldots + c_{L-1} t^{L-1} + \frac{c_L t^L}{1 - t c_{L+1}/c_L}. \tag{4.19}$$

Hence the pole of $[L/1]_{x_j}$ is c_L/c_{L+1} (or it is ∞ if $c_{L+1} = 0$). Fabry's ratio theorem [Fab96] now states that if the limit $\lim_{L\to\infty} c_L/c_{L+1}$ exists it is a singularity of $x(t)$.

Theorem 4.10. *Suppose that the coefficients of the power series $\sum_{\ell=0}^{\infty} c_\ell t^\ell$ are such that the limit $\lim_{L\to\infty} c_L/c_{L+1} = \lambda \neq 0$ exists. Then, the series converges uniformly inside the disk $\{|t| < |\lambda|\}$ and λ is a singular point of $x_j(t) = \sum_{\ell=0}^{\infty} c_\ell t^\ell$.*

For a fixed L, the modulus $|c_L/c_{L+1}|$ therefore can be assumed to be an approximation of the distance to the nearest singularity of $x(t)$ and a computational estimate of the trust-

region τ of the Padé approximant. It is recommended to see Section 3 of [TVBV20] to learn more about Padé approximants in the context of homotopy continuation.

For the computation of a Padé approximant of type (L, M), it is necessary to compute the local derivatives $x^{(\ell)}(0)$ for $\ell = 1, \ldots, L+M$. For this, Mackens [Mac89] observed the following useful identity.

Lemma 4.11 ([Mac89]). *The local derivatives $x^{(\ell)}(t)$ can be computed using the formula*

$$x^{(\ell)}(t) = -H_x(x(t), t)^{-1} R_\ell(t)\,,$$

where

$$R_\ell(t) \;= \left(\tfrac{d}{d\lambda}\right)^\ell H\Big(x(t) + \textstyle\sum_{i=1}^{\ell-1} \frac{1}{i!} x^{(i)}(t)\lambda^i, t+\lambda\Big)\Big|_{\lambda=0}\,. \tag{4.20}$$

In [Mac89], this identity is used for the computation of $x^{(\ell)}(0)$ by numerical differentiation. A downside of numerical differentiation is that it can suffer from catastrophic cancellation resulting in useless results. Instead of using numerical differentiation the expression (4.20) can be computed efficiently and accurately by using automatic differentiation [GW08, Ch. 13]. In particular, the cost of computing R_ℓ using automatic differentiation is at most $2\ell^2 + O(\ell)$ times the cost of evaluating H by a straight-line-program. The dominating factor for the accuracy of $x^{(\ell)}(t)$ is the forward error of the linear system solving. To ensure that computed derivatives are sufficiently accurate, the forward error of the linear system solving should be monitored and if necessary be reduced by using mixed-precision iterative refinement [Hig97].

A robust Padé approximant implementation also needs to handle the edge cases that $x_j^{(\ell)}(0) = 0$ for some $1 \le j \le n$ and $1 \le \ell \le L+M$. In [GGT13], a robust algorithm is proposed for computing Padé approximants. The provided implementation uses this algorithm for the computation of the Padé approximants.

It is also possible to obtain an estimate of the local approximation error of a Padé approximant as was shown in [TVBV20]. By comparing for each coordinate function $x_j(t)$ the coefficient of t^{L+M+1} in

$$(a_0 + a_1 t + \ldots + a_L t^L) - (1 + b_1 t + \ldots + b_M t^M)(c_0 + c_1 t + c_2 t^2 + \ldots)$$

it follows

$$e_{0,j} = -(c_{L+M+1} + b_1 c_{L+M} + \ldots + b_M c_{L+1})\,.$$

Considering the Taylor expansion of $[L/M]_{x_j}$ at 0 it follows

$$x_j(t) - [L/M]_{x_j}(t) = e_{0,j} t^{L+M+1} + O(t^{L+M+2})\,.$$

Therefore, for a Padé approximant a computational estimate $[\eta_{L+M+1}]$ of η_{L+M+1} is

$$[\eta_{L+M+1}] = \|D^{-1}(e_{0,1}, \ldots, e_{0,n})\| \tag{4.21}$$

where $D > 0$ are the same scaling factors as used for the Newton corrector in Section 4.1.

4.3 Step Size Control and Path Tracking Algorithm

After studying Newton's method as a correction method in Section 4.1 and Padé approximants as a prediction method in Section 4.2, the results are now combined to derive an adaptive step size control. Afterward, the path tracking algorithm is stated.

Consider the homotopy $H(x,t) : \mathbb{C}^n \times \mathbb{C} \to \mathbb{C}^n$ with a given solution $s \in \mathbb{C}^n$ of the system $H(x,0) = 0$ and constant $\bar{t} > 0$. Assume there is a solution path $x(t) : [0,\bar{t}] \to \mathbb{C}^n$ implicitly defined by the conditions $x(0) = s$ and (4.18), and assume that $H_x(x(t),t)$ is non-singular for all $t \in [0,\bar{t}]$. Denote by $\hat{x}(t) : [0,\bar{\tau}] \to \mathbb{C}^n$ the prediction path produced by the Padé approximant. As in Section 4.2 only the situation at $t = 0$ is considered such that $\Delta t = t$.

The goal of the step size routine is to provide a step size t such that the Newton iterates $x^{(j)}$ starting at the initial guess $\hat{x}(t) = x^{(0)}$ satisfy for $j = 0, 1, 2, \ldots$ the contraction factors

$$\Theta_j = \frac{\|\Delta x^{(j+1)}\|}{\|\Delta x^{(j)}\|} \leq a^{2^j} \tag{4.22}$$

for a fixed parameter $a \in (0,1)$, for instance $a = 0.2$. Recall from Lemma 4.2 that if $a \leq \frac{1}{2}$, then $\hat{x}(t)$ is an approximate zero. Assuming knowledge about the Lipschitz constant ω in a neighborhood of the path $x(t)$ and the theoretical quantities introduced in Section 4.2, it is possible to give a maximal theoretical feasible step size $t_{\max}$ such that this is the case. The approach to use the theoretical quantities ω, η_p, and τ to determine a maximal feasible step size such that Newton's method converges was pioneered by Deuflhard in [Deu79].

Theorem 4.12 ([DH79]). *Let $D \subseteq \mathbb{C}^n$ such that for all $x \in D$ and $t \in [0,\bar{t}]$, $\bar{t} > 0$, the Jacobian $H_x(x,t)$ is non-singular. Assume that for each $t \in [0,\bar{t}]$ there exists a convex subset $D(t) \subseteq D$ with $x(t) \in D(t)$ where $x(t)$ denotes the unique solution path in $D \times [0,\bar{t}]$. Let $\hat{x}(t) : [0,\bar{t}] \to D$ denote a prediction path of order p with trust-region τ, i.e., with*

$$\|\hat{x}(t) - x(t)\| \leq \eta_p t^p \quad \text{for all } t \in [0,\tau]\,.$$

Moreover, assume for all $t \in [0,\bar{t}]$ the affine covariant Lipschitz condition

$$\|H_x(\hat{x}(t),t)^{-1}(H_x(u,t) - H_x(v,t))\| \leq \omega\|u - v\| \text{ for all } u, v \in D(t)\,.$$

For fixed $h \leq \frac{1}{2}$, let $t_{\max} = \min(t^*, \tau, \bar{t})$ where

$$t^* := \left(\frac{\sqrt{1+2h}-1}{\omega\eta_p}\right)^{1/p} \tag{4.23}$$

and for all $t \leq t_{\max}$ let $B(t)$ denote a ball around $\hat{x}(t)$ with radius $(1-\sqrt{1-2h})/\omega$ and assume $B(t) \subseteq D(t)$. Then, for all step sizes $t \leq t_{\max}$ the Newton iterates starting at $\hat{x}(t) = x^{(0)}$ are well-defined, remain in $B(t)$, converge towards $x(t)$ and satisfy $\|H_x(\hat{x}(t),t)^{-1}H(\hat{x}(t),t)\| \leq \frac{h}{\omega}$.

Proof. For $h = \frac{1}{2}$, the statement is Theorem 1.3 in [Deu79]. The more general maximal step

size (4.23) and the inequality $\|H_x(\hat{x}(t),t)^{-1}H(\hat{x}(t),t)\| \leq \frac{h}{\omega}$ follows from equations (1.14a) and (1.14b) in the proof of Theorem 1.3 in [Deu79]. □

In Theorem 4.12, there is a choice of the parameter $h \leq \frac{1}{2}$. If this is sufficiently small then the contraction factors (4.22) are satisfied. The following corollary makes this precise.

Corollary 4.13. *In Theorem 4.12, choose $h \leq h(a) = 2(\sqrt{4a^4+a^2} - 2a^2)$ for $a \in (0,1)$. Then, the Newton iterates starting at $\hat{x}(t)$ satisfy the contraction factors*

$$\frac{\|\Delta x^{(j+1)}\|}{\|\Delta x^{(j)}\|} \leq a^{2^j}.$$

Proof. Since $\|H_x(\hat{x}(t),t)^{-1}H(\hat{x}(t),t)\| \leq \frac{h}{\omega}$, it follows $h_0 \leq h$ and using Lemma 4.2 it follows the statement. □

If the step size is chosen according to Theorem 4.12, then path jumping cannot happen. However, to obtain the theoretical quantities ω, η_p and τ is very hard. Instead the theoretical quantities are replaced by easy to obtain computational estimates $[\omega]$, $[\eta_p]$ and $[\tau]$. Using the computational estimates and Corollary 4.13, an estimate $[t_{\max}]$ of the maximal feasible step size $t_{\max}$ is given by

$$[t_{\max}] = \min\left([t^*],\, \beta_\tau[\tau],\, \bar{t}\right) \tag{4.24}$$

where

$$[t^*] := \left(\frac{\sqrt{1+2h(a)}-1}{\beta_\omega[\omega][\eta_p]}\right)^{1/p}$$

where $0 < \beta_\tau < 1$ and $\beta_\omega \geq 1$ are additional safety factors, for instance $\beta_\tau = 0.75$ and $\beta_\omega = 10$. Instead of choosing β_ω fixed it seems worthwhile to develop a more adaptive criterion for choosing β_ω in the future.

Since $[\eta_p]$ and $[\omega]$ are only lower bounds and $[\tau]$ is only an upper bound for the theoretical quantities, it is possible that the step size $t = [t_{\max}]$ is larger than $t_{\max}$. Then it can happen that the Newton corrector algorithm 4.5 rejects the initial guess since $\Theta_k > a^{2^k}$ for some k. In this case, a suitable step size correction formula is

$$t' = \left(\frac{\sqrt{1+2h(\frac{1}{2}a)}-1}{\sqrt{1+2h\left(\Theta_k^{2^{-k}}\right)}-1}\right)^{1/p} t \tag{4.25}$$

which is a clear reduction since $\Theta_k^{2^{-k}} > a$.

Before the path tracking algorithm is stated, the similarities and differences to the step size control developed in [TVBV20] are stated. In [TVBV20], the authors develop an adaptive step size control similar to (4.24) with the difference that their algorithm uses instead of $[t^*]$ the step size candidate Δt_1 computed as follows. For Δt_1, an estimate δ of the distance to the nearest path is computed based on a second-order Taylor expansion around $x(0)$. This involves the computation of the Hessian of H and multiple singular value decompositions.

Then $\Delta t_1 = (\beta_1 \delta / [\eta_p])^{1/p}$ where β_1 is a safety factor to unknown region of convergence of Newton's method, for instance $\beta_1 = 0.005$.

Finally, the path tracking algorithm, Algorithm 4.14, is stated. It is assumed that the computational estimates $[\omega]$ and $[\mu]$ for the start solution s are available. These are either available as a result of a previous path tracking or by using Algorithm 4.6.

Algorithm 4.14 Path Tracking Algorithm

Input: Homotopy $H(x,t) : \mathbb{C}^n \times \mathbb{C} \to \mathbb{C}^n$, $s \in \mathbb{C}^n$ such that s is an approximate zero of $G(x) := H(x,0)$, estimates $[\omega]$ and $[\mu]$ of ω and μ, $a \in (0, \frac{1}{2}]$, safety factors β_ω and β_τ, minimal step size $\Delta t_{\min}$, type of Padé approximant L.

Output: Approximate zero of $F(x) := H(x,1)$ or **false** if the tracking failed.

1: **procedure** TRACK($H, s, [\omega], [\mu], a, \beta_\omega, \beta_\tau, \Delta t_{\min}, L$)
2: $(t, \Delta t) \leftarrow (0, \infty)$
3: $x \leftarrow s$
4: $\bar{u} \leftarrow u$
5: Initialize scaling factors D using x and (4.17)
6: **while** $t < 1$ **do**
7: Compute $x^{(1)}(t), \ldots, x^{(L+2)}(t)$ from (x,t) by using the identity (4.20)
8: Compute $[\eta_{L+2}]$ and $[\tau]$ for $(L,1)$ Padé approximant using (4.19) and (4.21)
9: $\Delta t \leftarrow \min\left(\left[\frac{\sqrt{1+2h(a)}-1}{\beta_\omega [\omega][\eta_{L+2}]}\right]^{1/(L+2)}, 1-t, \beta_\tau[\tau]\right)$
10: Use $(L,1)$ Padé approximant to obtain initial guess $\hat{x}$ at $t + \Delta t$
11: Update scaling factors D using $\hat{x}$ and (4.17)
12: $(\texttt{success}, \bar{x}, [\mu], [\omega], \Theta_k) \leftarrow$ NEWTON($\hat{x}, a, [\mu], [\omega], \bar{u}, D$)
13: **if** success **then**
14: $t \leftarrow t + \Delta t$
15: $x \leftarrow \bar{x}$
16: **else**
17: $\Delta t \leftarrow \Delta t \left(\frac{\sqrt{1+2h(0.5a)}-1}{\sqrt{1+2h\left(\Theta_k^{2^{-k}}\right)}-1}\right)^{1/(L+2)}$
18: **if** $\Delta t < \Delta t_{\min}$ **then**
19: **return false**
20: **end if**
21: **go to** Line 10
22: **end if**
23: **if** $\bar{u} = u$ **and** $[\omega][\mu] > a^5 h(a)$ **then** ▷ Use extended precision, see (4.14)
24: $\bar{u} \leftarrow u^2$
25: **else if** $\bar{u} = u^2$ **then**
26: Compute estimate μ_u of $\mu(x,u,u)$ as described in Subsection 4.1.2
27: **if** $[\omega]\mu_u < a^7 h(a)$ **then**
28: $\bar{u} \leftarrow u$
29: **end if**
30: **end if**
31: **end while**
32: **return** x
33: **end procedure**

4.4 Computational Experiments

In this section, numerical experiments are shown to illustrate the effectiveness of the proposed path tracking algorithm. Additionally, the algorithm is compared against the adaptive precision path tracking algorithm [BHSW08] as it is implemented in the state of the art

package `Bertini` [BHSW]. For the different solvers, the following notations are used in the experiments:

`HC.jl`	HomotopyContinuation.jl v2.1
`Bertini DP`	Bertini v1.6 using double precision arithmetic (MPTYPE $= 0$)
`Bertini AP`	Bertini v1.6 using adaptive precision (MPTYPE $= 2$)

All solvers are used intentionally with the default settings, unless otherwise mentioned, since for a non-expert user it is very hard to understand which parameters need to be changed. `Bertini` uses by default a path tracking tolerance of 10^{-5} before and 10^{-6} during the endgame. The experiments are performed on a 24 GB RAM machine with an Intel Core i5-7500 CPU working at 3.40 GHz. All solvers use only one core for the experiments. The experiments are designed such that the tracked solution paths are all smooth. Therefore, endgame algorithms are not necessary. In all experiments, the implementation of the proposed algorithm uses a $(2, 1)$ Padé approximant with parameters $a = 0.2$, $\beta_\omega = 10$ and $\beta_\tau = 0.75$.

4.4.1 Mixtures of Gaussians and the Method of Moments

This example illustrates the behavior of the path tracking algorithm with respect to the two computed quantities $[t^*]$ and $\beta_\tau[\tau]$ that restrict the step size $[t_{\max}]$ in (4.24). Consider two univariate Gaussian random variables X_1 and X_2 with means μ_1, μ_2 and variances σ_1^2, σ_2^2. A mixture of the two random variables X_1 and X_2 is the random variable where for a given value $\lambda \in [0, 1]$ a value is drawn from X_1 with probability λ and with probability $1 - \lambda$ from X_2. Only given observations of the mixture of the two random variables, the parameters $\mu_1, \mu_2, \sigma_1^2, \sigma_2^2$ and λ can be recovered by using the method of moments. See [AFS16] for a detailed algebraic treatment of this problem. As shown in [AFS16], for a mixture of two univariate Gaussians, the method of moments results in a polynomial system consisting of five polynomials of degree two to six in five variables and with five sample moments as parameters. For generic sample moments, this system has 18 isolated solutions. Starting from generic sample moments, the 18 solutions were tracked to 50 different real sample moments drawn elementwise independently from a normal distribution. For this, a path needed on average 14.55 steps (including rejected steps) and on average only 0.18 steps got rejected. These results shows that the step size control effectively avoids taking too large steps. In 62.25% of the steps, the step size was more restricted by the curvature condition $[t^*]$ than the distance to the closest singularity $\beta_\tau[\tau]$. This shows that both conditions, $[t^*]$ and $\beta_\tau[\tau]$, have also in practice an effect on the chosen step size.

4.4.2 Alt's problem

Alt's problem, formulated in 1923, is to count the number of four-bar linkages whose coupler curve interpolates nine general points in the plane. In 1992, Morgan, Sommese, and Wampler [WMS92] provided a numerical proof to Alt's problem that there are generically 1442 non-degenerate four-bar linkages. Due to Roberts cognates and a two-fold symmetry, the resulting polynomial system generically has 8652 regular solutions. Here, the formulation of the

polynomial system as an affine polynomial system in 24 variables and the 16 parameters $(\delta, \hat{\delta}) \in \mathbb{C}^8 \times \mathbb{C}^8$ is considered. Since the problem is formulated in isotropic coordinates, the physically meaningful configurations correspond to choices of parameters such that δ and $\hat{\delta}$ are complex conjugates. Consider the 'general' situation where solutions are tracked from generic parameter values $\delta_1 \in \mathbb{C}^8 \times \mathbb{C}^8$ to generic physically meaningful parameter values $(\delta_0, \bar{\delta_0})$ with $\delta_0 \in \mathbb{C}^8$. The results in Table 4.1 show that even this general situation results in numerically challenging paths that `Bertini AP` cannot handle with its default settings. After decreasing the path tracking tolerance to 10^{-8}, all solutions are found in half of the cases. The proposed algorithm, on the other hand, reliably computes all 1442 solutions without any path jumping in a fraction of the time.

		runtime (seconds)				# solutions		
	`tol`	mean	median	min	max	median	min	max
`HC.jl`	-	6.38	6.41	5.08	7.12	1442	1442	1442
`Bertini AP`	1e-5	1931.63	1832.59	1120.43	2695.13	1436	1433	1441
`Bertini AP`	1e-8	10950.09	11218.97	8371.76	12115.66	1441	1437	1442

Table 4.1: Results for 10 runs of Alt's problem using the same generic start solutions to a generic physically meaningful configuration. The `tol` column refers to the assigned path tracking tolerance. Missing solutions are a result of path tracking failures.

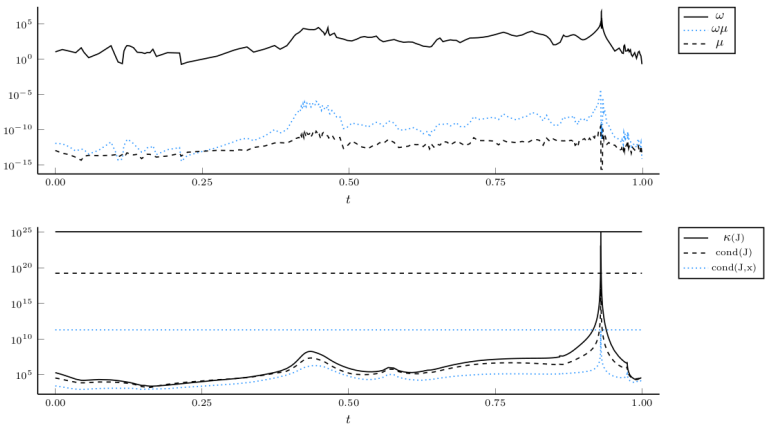

Figure 4.2: Behavior of the path tracking algorithm along a single challenging path.

Figure 4.2 illustrates the behavior of the proposed algorithm for one particular numerically challenging path. This path closely passes at around $t = 0.93$ a singularity. As expected, this increases the Lipschitz constant ω and the limit accuracy μ. The limit accuracy decreases sharply as soon as the algorithm switches to extended precision. After passing the problematic region, the algorithm quickly switches back to only using double-precision arithmetic, as indicated by a sharp increase of μ. The lower part of Figure 4.2 depicts different values associated with the Jacobian $J = H_x(x(t), t)$ along the path: the condition number $\kappa(J)$, the componentwise relative condition number $\mathrm{cond}(J)$ and $\mathrm{cond}(J, x)$. See [Hig02, Sec. 7] for the relevant definitions. The values of $\kappa(J)$ and $\mathrm{cond}(J)$ obtain a maximum of 2.1×10^{25}

resp. 1.6×10^{19}. From these values, it seems hopeless to track the path in double-precision arithmetic due to the well-known rule of thumb that one expects to lose around $\log_{10}(\kappa(J))$ digits of accuracy in the linear system solving. However, the componentwise relative forward error of the computed Newton updates is only governed by the much tamer $\mathrm{cond}(J, x)$, which is at most 5.8×10^{10}. This explains why it is still possible to track the path by only using mixed-precision iterative refinement. Using the proposed path tracking algorithm, the path needs 253 steps in total with only two rejected steps and a total runtime of 13 milliseconds. Trying to compute the path with `Bertini AP` results in a path failure after around 90 seconds and over 5000 steps due to insufficient precision of at most 1024 bits. After changing the required tracking tolerance to 10^{-12}, `Bertini AP` successfully tracks the path in 120 seconds.

4.4.3 Steiner's Conic Problem

In Chapter 2, we considered Steiner's conic problem. Here, formulation (2.11) of Steiner's conic problem is used where the polynomial system consists of 10 quadratic and 5 cubic polynomials and has 15 variables and 30 parameters. To test the path tracking algorithm, we consider the case of a parameter homotopy from generic complex parameters $c \in \mathbb{C}^{30}$ to generic real parameters $r \in \mathbb{R}^{30}$.

		runtime (seconds)				# solutions		
	`tol`	mean	median	min	max	median	min	max
`HC.jl`		2.47	2.49	1.64	3.11	3264	3264	3264
`Bertini DP`	1e-5	34.42	34.57	23.85	45.34	3189	3094	3216
`Bertini AP`	1e-5	130.53	126.61	78.70	251.01	3261	3256	3264
`Bertini AP`	1e-8	688.50	691.59	368.36	1125.72	3264	3261	3264

Table 4.3: Results for 50 runs of Steiner's problem from generic complex parameter values to generic real parameter values. The `tol` column refers to the assigned path tracking tolerance. Missing solutions are a result of path tracking failures.

The results for 50 parameter homotopies are shown in Table 4.3. As for Alt's problem, the proposed path tracking algorithm handles all instances without any failure or path jumping. However, even these generic instances pose problems for other path tracking algorithms with `Bertini AP` losing almost always solutions using the default settings. After the path tracking tolerance is decreased to 10^{-8} in most instances all solutions are found.

4.5 Conclusion

In this chapter, we developed a new predictor-corrector algorithm for numerical path tracking in the context of polynomial homotopy continuation. For the corrector step, we designed a new Newton corrector algorithm based on the Newton-Kantorovich theorem that rejects an initial guess if it is not an approximate zero. As the result of the Newton corrector algorithm, we also obtain a local estimate of a Lipschitz-constant that governs the convergence radius

and speed of Newton's method. By using Padé approximants as a predictor, we get a method to estimate locally the distance to the closest singularity. We combined these estimates to design an adaptive step size control. To handle numerically challenging situations, we developed criteria to use extended precision for the evaluation of the homotopy. We demonstrated the efficiency and robustness of the developed mixed-precision algorithm in several numerical examples.

5 Certifying Zeros of Polynomial Systems using Interval Arithmetic

This chapter is based on the article "Certifying Zeros of Polynomial Systems using Interval Arithmetic" [BRT20] by Paul Breiding, Kemal Rose, and Sascha Timme. The article is currently under review. A preprint is available at `https://arxiv.org/abs/2011.05000`.

In Chapter 3, we discussed methods for the numerical solution of polynomial systems. Hauenstein and Sottile remark in [HS12] that while software packages implementing these methods "routinely and reliably solve systems of polynomial equations with dozens of variables having thousands of solutions" they have the shortcoming that "the output is not certified" and that "this restricts their use in some applications, including those in pure mathematics". To remedy this, Hauenstein and Sottile developed the software `alphaCertified` [HS12]. It can rigorously certify whether a given numerical approximation converges quadratically to a true zero by using Smale's α-theory [Sma86] which we introduced in Section 3.6. Hauenstein and Sottile's contribution to numerical algebraic geometry was a milestone. Yet, `alphaCertified` produces rigorous certificates using expensive rational arithmetic. This turns the big advantage of numerical computations, namely that they are fast, upside-down, and makes certification of large problems prohibitively expensive.

Up to this point, the majority of researchers in nonlinear algebra were kept from using numerical methods, because certification was too expensive and because without certification numerical methods can't be used for proofs. With this chapter we want to initiate a paradigm shift in numerical nonlinear algebra: with a fast implementation, certification becomes the default and is not just an option. This enables the extensive use of numerical methods for rigorous proofs.

Contribution

Our contribution to the field of computational algebraic geometry and numerical nonlinear algebra is an extremely fast and easy-to-use implementation of a certification method. This implementation outperforms `alphaCertified` by several orders of magnitude. It makes the certification of solutions often a matter of seconds and not hours or days. This leap in performance is the basis for the proposed paradigm shift in numerical nonlinear algebra where certification is the default and not an option.

Starting from version 2.1 `HomotopyContinuation.jl` has a function `certify`[1]. The function `certify` takes as input a *square polynomial system* F and a numerical approxima-

[1] The technical documentation is available at `www.juliahomotopycontinuation.org/HomotopyContinuation.jl/stable/certification`

tion of a complex zero $x \in \mathbb{C}^n$ (or a list of zeros). If the output says "certified", then this is a rigorous proof that a solution of $F = 0$ is near x. If the output says "not certified", then this does not necessarily mean that there is no zero near x, just that the method couldn't find one. An example of `certify` is shown in Figure 2.4 in the Chapter on Steiner's conic problem. See also the example [BRT] on `https://www.juliahomotopycontinuation.org`.

We combine interval arithmetic and Krawczyk's method with numerical algebraic geometry to rigorously certify solutions to square systems of polynomial equations. In technical terms, our implementation returns *strong interval approximate zeros*. We introduce this notion in Definition 5.8 below. The strong interval approximate zero consists of a box in $\mathbb{C}^n$ that contains a unique true zero of the polynomial system. If the input is a list of zeros, then the routine returns a list of distinct strong interval approximate zeros.

Therefore, our method can be used to *prove* hard lower bounds on the number of zeros of a polynomial system. Combined with theoretical upper bounds this can constitute rigorous mathematical proofs on the number of zeros of such systems.

Also, if the given polynomial system is real, we give a certificate of whether the certified zero is a real zero. The returned boxes may also be used to verify if a real zero is positive real. Therefore, our method can also be used to prove lower bounds on the number of real and positive real zeros of a polynomial system.

It is also possible to give a square system of rational functions as input to our implementation. Although this chapter is formulated in terms of polynomial systems, Krawczyk's method also applies to square systems of rational functions. Consequently, all statements about using our implementation for proofs are equally valid of square systems of rational functions. Nevertheless, we think that the focus on polynomial systems simplifies the exposition.

Comparison to previous works

There are other implementations of certification methods using Krawczyk's method and interval arithmetic, e.g., the commercial `MATLAB` package `INTLAB` [Rum99], the `Macaulay2` package `NumericalCertification` [Lee19] and the `Julia` package `IntervalRootFinding.jl` [BS].

Compared to `INTLAB` the source code of our implementation is freely available and can be verified by anyone. Additionally, `INTLAB` doesn't support the use of arbitrary precision interval arithmetic. This limits its capability to certify poorly conditioned solutions. `NumericalCertification`, as of version 1.0, takes as input not the numerical approximation of a complex zero $x \in \mathbb{C}^n$ but instead a box I in $\mathbb{C}^n$. Then, `NumericalCertification` attempts to certify that interval I is a strong interval approximate. The process of going from a numerical approximation x to a good candidate interval I needs particular care as illustrated in Section 5.4. `Intlab` and `NumericalCertification` also both require manual work to obtain a list of all distinct distinct strong interval approximate zeros. The package `IntervalRootFinding.jl` can find all zeros of a multivariate function inside a given box in $\mathbb{R}^n$, whereas our implementation works in $\mathbb{C}^n$ and additionally certifies reality of zeros; see Section 5.3.

Our contribution is a significant advancement over these previous works since it not only provides an implementation of Krawczyk's method but also combines it with the necessary tools and techniques to deliver an easy to use and robust certification routine.

We want to note that there are also many other approaches to certifying the existence of solutions in a region, e.g., based on quantifier elimination by partial cylindrical algebraic decomposition as implemented in QEPCAD B [Bro03]. However, to keep this presentation concise we focus in the following only on interval arithmetic.

Outline

The rest of this chapter is organized as follows: In the next section, we demonstrate our implementation on three applications. This shows both the speed of the implementation and how it can be used for proofs. We discuss the details of our implementation in Section 5.4. For completeness, we include a short introduction to interval arithmetic in Section 5.2 and a proof of Krawczyk's method in Section 5.3.

5.1 Applications

Certification methods are useful when one wants to prove statements on the number of zeros, the number of real zeros, or the number of positive real zeros of a polynomial system. When determining these numbers it is often most challenging to obtain lower bounds. Methods from algebraic geometry provide upper bounds and applying our certification method can give a proof that the upper bound for the number of zeros is attained. A computation with our certification method always reveals lower bounds.

In the following, we discuss the application of our implementation in three different fields. All reported timings were obtained on an desktop computer with a 3.4 GHz processor running `Julia` 1.5.2 [BEKS17] and `HomotopyContinuation.jl` version 2.2.2.

3264 real conics

In Chapter 2, we discussed Steiner's conic problem. There we used our certification routine to prove that the instance proposed in Proposition 2.1 is fully real. The results in Chapter 2 were first reported in [BST20] where we used the software `alphaCertified` to carry out the certification. The certification with `alphaCertified` took us *more than 36 hours*. In contrast, our implementation certifies the reality and distinctness of the 3264 conics in *less than three seconds*. The output of the certification procedure is shown in Figure 2.4.

Numerical Synthesis of Six-Bar Linkages

Now we demonstrate that the certification routine can cope with large problems. With our computation, we improve a result from the literature.

We consider the kinematic synthesis of six-bar linkages that use eight prescribed accuracy points as described in [PM14]. In this article, the authors derive the synthesis equations for six-bar linkages of the Watt II, Stephenson II, and Stephenson III type. Additionally, in [PM14, Eq. (35)] they construct a system of 22 polynomials in 22 unknowns and 224 parameters that can be used as a start system in a parameter homotopy to solve the synthesis equations of all three considered six-bar linkage types.

The number of non-singular zeros of this generalized start system is reported as 92,736. It was computed using `Bertini` and a multi-homogeneous start system. To certify the reported count, we solved the generalized start system using the monodromy method [DHJ$^+$18] implementation in `HomotopyContinuation.jl`. In our computation, we obtained 92,752 non-singular zeros for a generic choice of the 224 parameters. These are sixteen *more* than reported in [PM14]. We certified this count using our certification routine and obtained 92,752 distinct strong interval approximate zeros. Therefore, we have a certificate that the generalized system has in general (at least) 92,752 non-singular solution. This establishes that the result in [PM14] undercounts the true number of solutions. The certification needed only 38.34 seconds which underlines the scalability of the certification routine.

In Section 5.4, below we discuss that part of the certification process is checking if the intervals are pairwise disjoint. In this example, there are over 4 billion such pairs. This underlines the need for having an efficient algorithm for comparing pairs.

Stress response of Bacillus Subtilis

In this section, we demonstrate that our implementation certifies the positivity of zeros. In many applications, variables represent magnitudes so that only positive real solutions are physically meaningful. If such zeros exist, our method provides a rigorous proof for their existence. It also gives a certified interval that contains the true zero. This is of interest to researchers working at the intersection of algebraic geometry and (bio-)chemical reaction networks.

Our example is from biochemistry. The environmental and energy stress response of the bacterium *Bacillus subtilis* are modeled in [NTI16]. The protein σ_B is the focus of this paper. It is responsible for activating a stress-response of the bacterium. σ_B belongs to the family of σ factors. These are a type of transcription factors; proteins that govern the expression of genes.

In [NTI16], regulatory networks are studied. They consist of other proteins involved in feedback loops that influence the σ-factors. Since there can be many possible reactants involved in many reactions, the resulting system of differential equations might be very complicated. The model for *Bacillus subtilis* in [NTI16] is claimed to be backed up by experimental data. The activity of σ_B is regulated by a network consisting of an anti-σ factor RsbW and an anti-σ factor RsbV.

In [NTI16], this biochemical reactions dynamical system is modelled by a system of differential equations in the 10 variables w, w_2, $w_{2\mathrm{v}}$, v, $w_{2\mathrm{v}2}$, v_P, σ_B, $w_{2\sigma\mathrm{B}}$, v_{Pp} and phos. These represent the total amounts of σ_B, RsbW, σ factor RsbV, and of various protein complexes formed by these components. The variable phos measures the concentration of the phosphatase which serves as a measure for the amount of stress the bacterium experiences.

With our implementation, we can determine the steady states of the described dynamical system. The vanishing of the differentials of each of the concentrations with respect to time is equivalent to the vanishing of the ten polynomials below.

$$
\begin{aligned}
(-k_{\mathrm{Deg}}w - 2k_{\mathrm{bw}}\tfrac{w^2}{2} + 2k_{\mathrm{dw}}w_2)(K+\sigma_B) + \lambda_W v_0(1+F\sigma_B) &= 0\\
-k_{\mathrm{Deg}}w_2 + k_{\mathrm{bw}}\tfrac{w^2}{2} - k_{\mathrm{dw}}w_2 - k_{\mathrm{B1}}w_2 v + k_{\mathrm{D1}}w_{2\mathrm{v}} + k_{\mathrm{K1}}w_{2\mathrm{v}} - k_{\mathrm{B3}}w_2\sigma_B + k_{\mathrm{D3}}w_{2\sigma\mathrm{B}} &= 0\\
-k_{\mathrm{Deg}}w_{2\mathrm{v}} + k_{\mathrm{B1}}w_2 v - k_{\mathrm{D1}}w_{2\mathrm{v}} - k_{\mathrm{B2}}w_{2\mathrm{v}}v + k_{\mathrm{D2}}w_{2\mathrm{v2}} - k_{\mathrm{K1}}w_{2\mathrm{v}} + k_{\mathrm{K2}}w_{2\mathrm{v2}} + k_{\mathrm{B4}}w_{2\sigma\mathrm{B}}v - k_{\mathrm{D4}}w_{2\mathrm{v}}\sigma_B &= 0\\
(-k_{\mathrm{Deg}}v - k_{\mathrm{B1}}w_2 v + k_{\mathrm{D1}}w_{2\mathrm{v}} - k_{\mathrm{B2}}w_{2\mathrm{v}}v + k_{\mathrm{D2}}w_{2\mathrm{v2}} - k_{\mathrm{B4}}w_{2\sigma\mathrm{B}}v + k_{\mathrm{D4}}w_{2\mathrm{v}}\sigma_B + k_{\mathrm{P}}v_{\mathrm{Pp}})(K+\sigma_B)&\\
+\lambda_V v_0(1+F\sigma_B) &= 0\\
-k_{\mathrm{Deg}}w_{2\mathrm{v2}} + k_{\mathrm{B2}}w_{2\mathrm{v}}v - k_{\mathrm{D2}}w_{2\mathrm{v2}} - k_{\mathrm{K2}}w_{2\mathrm{v2}} &= 0\\
-k_{\mathrm{Deg}}v_P + k_{\mathrm{K1}}w_{2\mathrm{v}} + k_{\mathrm{K2}}w_{2\mathrm{v2}} - k_{\mathrm{B5}}v_P\mathrm{phos} + k_{\mathrm{D5}}v_{\mathrm{Pp}} &= 0\\
(-k_{\mathrm{Deg}}\sigma_B - k_{\mathrm{B3}}w_2\sigma_B + k_{\mathrm{D3}}w_{2\sigma\mathrm{B}} + k_{\mathrm{B4}}w_{2\sigma\mathrm{B}}v - k_{\mathrm{D4}}w_{2\mathrm{v}}\sigma_B)(K+\sigma_B) + v_0(1+F\sigma_B) &= 0\\
-k_{\mathrm{Deg}}w_{2\sigma\mathrm{B}} + k_{\mathrm{B3}}w_2\sigma_B - k_{\mathrm{D3}}w_{2\sigma\mathrm{B}} - k_{\mathrm{B4}}w_{2\sigma\mathrm{B}}v + k_{\mathrm{D4}}w_{2\mathrm{v}}\sigma_B &= 0\\
-k_{\mathrm{Deg}}v_{\mathrm{Pp}} + k_{\mathrm{B5}}v_P\mathrm{phos} - k_{\mathrm{D5}}v_{\mathrm{Pp}} - k_{\mathrm{P}}v_{\mathrm{Pp}} &= 0\\
\mathrm{phos} + v_{\mathrm{Pp}} - p_{\mathrm{tot}} &= 0
\end{aligned}
$$

The 23 parameters k_{bw}, k_{dw}, kD, k_{B1}, k_{B2}, k_{B3}, k_{B4}, k_{B5}, k_{D1}, k_{D2}, k_{D3}, k_{D4}, k_{D5}, k_{K1}, k_{K2}, k_{P}, k_{Deg}, v_0, F, K, λ_W, λ_V, p_{tot} describe the speed of different reactions. The following parameter values are derived from experimental data.

$$
\begin{aligned}
&k_{\mathrm{Bw}} = 3600; k_{\mathrm{Dw}} = 18; k_D = 18 k_{\mathrm{B1}} = 3600; k_{\mathrm{B2}} = 3600; k_{\mathrm{B3}} = 3600; k_{\mathrm{B4}} = 1800; k_{\mathrm{B5}} = 3600;\\
&k_{\mathrm{D1}} = 18; k_{\mathrm{D2}} = 18; k_{\mathrm{D3}} = 18; k_{\mathrm{D4}} = 1800; k_{\mathrm{D5}} = 18; k_{\mathrm{K1}} = 36; k_{\mathrm{K2}} = 36; k_{\mathrm{P}} = 180; k_{\mathrm{Deg}} = 0.7;\\
&v_0 = 0.4; F = 30; K = 0.2; \lambda_W = 4; \lambda_V = 4.5; p_{\mathrm{tot}} = 2;
\end{aligned}
$$

As discussed above, only real positive zeros are physically meaningful. Using our implementation, we can certify that there are 12 real zeros for this system and that among them there is a unique positive one. It has the following values:

$$
\begin{aligned}
\mathrm{phos} &= 0.00406661084 \pm 5.25 \cdot 10^{-12}, & v &= 0.0557971948 \pm 4.87 \cdot 10^{-12}\\
v_P &= 27.0899869 \pm 3.85 \cdot 10^{-8}, & v_{\mathrm{Pp}} &= 1.99593338916 \pm 5.20 \cdot 10^{-12}\\
w &= 0.10633375735 \pm 8.47 \cdot 10^{-12}, & w_2 &= 0.303554095 \pm 5.47 \cdot 10^{-10}\\
w_{2\mathrm{v}} &= 2.25701026 \pm 2.08 \cdot 10^{-9}, & w_{2\mathrm{v2}} &= 8.288216246 \pm 9.27 \cdot 10^{-10}\\
w_{2\sigma\mathrm{B}} &= 10.42034597 \pm 7.94 \cdot 10^{-9}, & \sigma_B &= 0.240800757 \pm 5.17 \cdot 10^{-10}
\end{aligned}
$$

This is a proof that the dynamical system has a physically meaningful steady state. The intervals above provably contain this steady state. The certification took 0.012 seconds. Hence, our implementation makes it possible to certify solutions for large numbers of parameters in a short time.

The code for this example is available at [BRT]. We thank Torkel Loman from the Sainsbury Laboratory at the University of Cambridge for pointing out this example to us.

5.2 Interval Arithmetic

Since the 1950s researchers [Moo66, Sun58] have worked on interval arithmetic. Interval arithmetic allows certified computations while still using floating-point arithmetic. We briefly introduce the concepts from interval arithmetic that are relevant for our certification procedure.

Real Interval Arithmetic

Real interval arithmetic concerns computing with compact real intervals. Following [May17] we denote the set of all compact real intervals by

$$\mathbb{IR} := \{[a, b] \mid a, b \in \mathbb{R}, a \leq b\}.$$

For $X, Y \in \mathbb{IR}$ and the binary operation $\circ \in \{+, -, \cdot, /\}$, we define

$$X \circ Y = \{x \circ y \,|\, x \in X, y \in Y\} \tag{5.1}$$

where we assume $0 \notin Y$ in the case of division. The interval arithmetic version of these binary operations, as well as other standard arithmetic operations, have explicit formulas. See, e.g., [May17, Sec. 2.6] for more details.

Complex Interval Arithmetic

We define the set of *rectangular complex intervals* as

$$\mathbb{IC} := \{X + iY \mid X, Y \in \mathbb{IR}\}$$

where $X + iY = \{x + iy \mid x \in X, y \in Y\}$ and $i = \sqrt{-1}$. Following [May17, Ch. 9] we define the algebraic operations for $I = X + iY, J = W + iZ \in \mathbb{IC}$ in terms of operations on the real intervals from (5.1):

$$\begin{aligned} I + J &:= (X + W) + i(Y + Z), & I \cdot J &:= (X \cdot W - Y \cdot Z) + i(X \cdot Z + Y \cdot W) \\ I - J &:= (X - W) + i(Y - Z), & \frac{I}{J} &:= \frac{X \cdot W + Y \cdot Z}{W \cdot W + Z \cdot Z} + i\frac{Y \cdot W - X \cdot Z}{W \cdot W + Z \cdot Z} \end{aligned} \tag{5.2}$$

It is necessary to use (5.1) instead of complex arithmetic for the definition of algebraic operations in $\mathbb{IC}$. The following example from [May17] demonstrates this. Consider the intervals $I = [1, 2] + i[0, 0]$ and $J = [1, 1] + i[1, 1]$. Then, $\{x \cdot y | x \in I, y \in J\} = \{t(1 + i) \mid 1 \leq t \leq 2\}$ is not a rectangular complex interval, while $I \cdot J = [1, 2] + i[1, 2]$ is.

The algebraic structure of $\mathbb{IC}$ is given by following theorem; see, e.g., [May17, Theorem 9.1.4].

Theorem 5.1. *The following holds.*

1. $(\mathbb{IC}, +)$ *is a commutative semigroup with neutral element.*
2. $(\mathbb{IC}, +, \cdot)$ *has no zero divisors.*

Furthermore, if $I, J, K, L \in \mathbb{IC}$*, then*

3. $I \cdot (J + K) \subseteq I \cdot J + I \cdot K$*, but equality does not hold in general.*
4. $I \subseteq J, K \subseteq L$*, then* $I \circ K \subseteq J \circ L$ *for* $\circ \in \{+, -, \cdot, /\}$.

Working with interval arithmetic is challenging because of the third item from the previous theorem: distributivity does not hold in $\mathbb{IC}$ As a consequence, in $\mathbb{IC}$ the evaluation of

polynomials depends on the exact order of the evaluation steps. Therefore, the evaluation of polynomial maps $F : \mathbb{IC}^n \to \mathbb{IC}$ is only well-defined if F is defined by a straight-line program, and not just by a list of coefficients. Figure 5.1 demonstrates this issue in an example. See, e.g., [BCS13, Sec. 4.1] for an introduction to straight-line programs.

Figure 5.1: The picture shows two straight-line programs for evaluating the polynomial $f(x, y, z) = (x + y)z$. Let $I = ([-1, 0], [1, 1], [0, 1])^T$. Then, the program on the left evaluated at I yields $f(I) = ([-1, 0] + [1, 1])[0, 1] = [0, 1]$, while the program on the right yields $f(I) = [-1, 0][0, 1] + [1, 1][0, 1] = [-1, 1]$.

Arithmetic in $\mathbb{IC}^n$ is defined in the expected way. If $I = (I_1, \ldots, I_n), J = (J_1, \ldots, J_n) \in \mathbb{IC}^n$,

$$I + J = (I_1 + J_1, \ldots, I_n + J_n).$$

Scalar multiplication for $I \in \mathbb{IC}$ and $J \in \mathbb{IC}^n$ is defined as $I \cdot J = (I \cdot J_1, \ldots, I \cdot J_n)$. The product of an interval matrix $A = (A_{i,j}) \in \mathbb{IC}^{n\times n}$ and an interval vector $I \in \mathbb{IC}^n$ is

$$A \cdot I := I_1 \cdot \begin{bmatrix} A_{1,1} \\ \vdots \\ A_{n,1} \end{bmatrix} + \cdots + I_n \cdot \begin{bmatrix} A_{1,n} \\ \vdots \\ A_{n,n} \end{bmatrix}. \tag{5.3}$$

Similar to the one-dimensional case $(\mathbb{IC}^n, +)$ is a commutative semigroup with neutral element.

5.3 Certifying Zeros with Interval Arithmetic

In 1969, Krawczyk [Kra69] developed an interval arithmetic version of Newton's method. Later in 1977 Moore [Moo77] recognized that Krawczyk's method can be used to certify the existence and uniqueness of a solution to a system of nonlinear equations. Interval arithmetic and interval Newton's method are a prominent tool in many areas of applied mathematics; e.g., in chemical engineering [GS05], thermodynamics [GD05] and robotics [KSS15].

The results in this section are stated for square polynomial systems but they hold equally for square systems of rational functions. Krawczyk's method is even valid for general square systems of analytic functions. Nevertheless, all statements here are only formulated for polynomial systems. We think that this simplifies the exposition.

Krawczyk's method

In this section, we recall Krawczyk's method for zeros of polynomial systems. First, we need three definitions.

Definition 5.2 (Interval enclosure)**.** Let $F : \mathbb{C}^n \to \mathbb{C}^n$ be a system of polynomials. A map $\square F : \mathbb{IC}^n \to \mathbb{IC}^n$ is an interval enclosure of the system F if for every $I \in \mathbb{IC}^n$ we have $\{F(x) \mid x \in I\} \subseteq \square F(I)$.

In the rest of this chapter, we use the notation $\square F$ to denote the interval enclosure of F. Also, we do not distinguish between a point $x \in \mathbb{C}^n$ and the complex interval $[\mathrm{Re}(x), \mathrm{Re}(x)] + i[\mathrm{Im}(x), \mathrm{Im}(x)]$ defined by x. We simply use the symbol "x" for both terms so that $\square F(x)$ is well-defined.

Definition 5.3 (Interval matrix norm)**.** Let $A \in \mathbb{IC}^{n\times n}$. We define the operator norm of A as $\|A\|_\infty := \max\limits_{B\in A} \max\limits_{v\in\mathbb{C}^n} \frac{\|Bv\|_\infty}{\|v\|_\infty}$, where $\|(v_1, \ldots, v_n)\|_\infty = \max_{1\le i\le n} |v_i|$ is the infinity norm in $\mathbb{C}^n$.

Next, we introduce an interval version of the Newton operator, the *Krawczyk operator* [Kra69].

Definition 5.4. Let $F : \mathbb{C}^n \to \mathbb{C}^n$ be a system of polynomials, and $\mathrm{J}F$ be its Jacobian matrix seen as a function $\mathbb{C}^n \to \mathbb{C}^{n\times n}$. Let $\square F$ be an interval enclosure of F and $\square \mathrm{J}F$ be an interval enclosure of $\mathrm{J}F$. Furthermore, let $I \in \mathbb{IC}^n$ and $x \in \mathbb{C}^n$ and let $Y \in \mathbb{C}^{n\times n}$ be an invertible matrix. We define the Krawczyk operator

$$K_{x,Y}(I) := x - Y \cdot \square F(x) + (\mathbf{1}_n - Y \cdot \square \mathrm{J}F(I))(I - x).$$

Here, $\mathbf{1}_n$ is the $n \times n$-identity matrix.

Remark 5.5. In the literature, $K_{x,Y}(I)$ is often defined using $F(x)$ and not $\square F(x)$. Here, we use this definition, because in practice it is usually not feasible to evaluate $F(x)$ exactly. Instead, $F(x)$ is replaced by an interval enclosure.

Remark 5.6. The second part of Theorem 5.7 motivates to find a matrix $Y \in \mathbb{C}^{n\times n}$ such that $||1_n - Y \cdot \square \mathrm{J}F(I)||_\infty$ is minimized. A good choice is an approximation of the inverse of $\mathrm{J}F(x)$.

We are now ready to state the theorem behind Krawczyk's method. The first proof for real interval arithmetic is due to Moore [Moo77]. One of the few sources that states the theorem in the complex setting is [BLL19]. For completeness, we recall their proof in this section. Note that all the data in the theorem can be computed using interval arithmetic.

Theorem 5.7. *Let $F : \mathbb{C}^n \to \mathbb{C}^n$ be a system of polynomials and $I \in \mathbb{IC}^n$. Let $x \in I$ and $Y \in \mathbb{C}^{n\times n}$ be an invertible complex $n \times n$ matrix. The following holds.*

1. *If $K_{x,Y}(I) \subset I$, there is a zero of F in I.*
2. *If additionally $\sqrt{2}\,\|\mathbf{1}_n - Y\square \mathrm{J}F(I)\|_\infty < 1$, then F has exactly one zero in I.*

To simplify our language when talking about intervals $I \in \mathbb{IC}^n$ satisfying Theorem 5.7, we introduce the following definitions.

Definition 5.8. Let $F : \mathbb{C}^n \to \mathbb{C}^n$ be a square system of polynomials and $I \in \mathbb{IC}^n$. Let $K_{x,Y}(I)$ be the associated Krawczyk operator (see Definition 5.4). If there exists an invertible matrix $Y \in \mathbb{C}^{n\times n}$ such that $K_{x,Y}(I) \subset I$, we say that I is an *interval approximate zero* F. We call I a *strong interval approximate zero* of F if in addition $\sqrt{2}\|\mathbf{1}_n - Y\square\mathrm{J}F(I)\|_\infty < 1$.

Definition 5.9. If I is an interval approximate zero, then, by Theorem 5.7, I contains a zero of F. We call such a zero an *associated zero* of I. If I is a strong interval approximate zero, then there is a unique associated zero and we refer to is as *the* associated zero of I.

The notion of strong interval approximate zero is stronger than the definition suggests at first sight. We do not only certify that a unique zero of F exists inside I but we even certify that we can approximate this zero with arbitrary precision. This is shown in the next proposition. We prove the proposition at the end of this section.

Proposition 5.10. *Let I be a strong interval approximate zero of F and let $x^* \in I$ be the unique zero of F inside I. Let $x \in I$ be any point in I. We define $x_0 := x$ and for all $i \geq 1$ we define the iterates $x_i := x_{i-1} - Y\,F(x_{i-1})$, where $Y \in \mathbb{C}^{n\times n}$ is the matrix from Definition 5.8. Then, the sequence $(x_i)_{i\geq 0}$ converges to x^*.*

The idea for the proof of both Theorem 5.7 and Proposition 5.10 is to verify that for strong interval approximate zeros I the map $G_Y(x) = x - Y \cdot F(x)$ defines a contraction on I. If this is true, by Banach's Fixed Point Theorem there is exactly one fixed-point of this map in I. Since Y is invertible, this implies that there is exactly one zero to $F(x)$ in I.

Before we give the proof of Theorem 5.7, we need a lemma. It is a direct sequence of a complex version of the mean-value theorem which is shown implicitly in the proof of [BLL19, Lemma 2].

Lemma 5.11. *Fix a matrix $Y \in \mathbb{C}^{n\times n}$ and define $G_Y(x) = x - YF(x)$. Let $I \in \mathbb{IC}^n$ be an interval vector and $x, z \in I$. Then, we have*

1. $G_Y(z) - G_Y(x) \in (\mathbf{1}_n - Y \cdot \square\mathrm{J}F(I))\,\mathrm{Re}(z-x) + (\mathbf{1}_n - Y \cdot \square\mathrm{J}F(I))\,i\mathrm{Im}(z-x)$.
2. $G_Y(I) \subset K_{x,Y}(I)$.

The following proof is adapted from [BLL19, Lemma 2].

Proof of Lemma 5.11. In the proof, we abbreviate $G := G_Y$. We first show the second part assuming the first part of the lemma. Then, we prove the first part. We fix an interval $I \in \mathbb{IC}^n$ and $x, z \in \mathbb{C}^n$.

For the second part, we have to show that for all $I \in \mathbb{IC}^n$ we have $G(I) \subset K_{x,Y}(I)$. To show this, we define the interval matrix $M := (\mathbf{1}_n - Y\square\mathrm{J}F(I)) \in \mathbb{IC}^{n\times n}$. By definition of $K_{x,Y}$ we have $G(x) + M(I-x) \subset K_{x,Y}(I)$. Thus, we have to show that $G(z) - G(x) \in M(I-x)$, since $z \in I$ is arbitrary. The first part of the lemma implies that we can find

matrices $M_1, M_2 \in M$ such that $G(z)-G(x) = M_1\mathrm{Re}(z-x)+iM_2\mathrm{Im}(z-x)$. Decomposing the matrices into real and imaginary part we find

$$\begin{aligned} G(z) - G(x) =& \mathrm{Re}(M_1)\mathrm{Re}(z-x) - \mathrm{Im}(M_2)\mathrm{Im}(z-x) + \\ & i(\mathrm{Im}(M_1)\mathrm{Re}(z-x) + \mathrm{Re}(M_2)\mathrm{Im}(z-x)). \end{aligned}$$

Since $z - x \in I$ and by definition of the complex interval multiplication from (5.2) and the interval matrix-vector-multiplication (5.3), we see that $G(z) - G(x) \in M(I - x)$. This finishes the proof for the second part.

The first part of the lemma may be shown entry-wise. We will show this by combining a complex version of the mean value theorem with the following observation: $\mathrm{J}G(x) = \mathbf{1}_n - Y \cdot \mathrm{J}F(x)$, so we have the inclusion

$$\mathrm{J}G(I) = \mathbf{1}_n - Y \cdot \mathrm{J}F(I) \subseteq \mathbf{1}_n - Y \cdot \square\mathrm{J}F(I). \tag{5.4}$$

We relate $G(z) - G(x)$ to (5.4) using the mean value theorem. First, we define $w := \mathrm{Re}(z)+i\mathrm{Im}(x)$. Let $1 \leq j \leq n$ and let G_j denote the j-th entry of G. We define the function $h(t) := G_j(tz+(1-t)w)$. The real and imaginary part of $h(t)$ are real differentiable functions of the real variable t. The mean value theorem can be applied, and we find $0 < t_1, t_2 < 1$ such that $\mathrm{Re}(h(1)) - \mathrm{Re}(h(0)) = \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{Re}(h(t_1))$ and $\mathrm{Im}(h(1)) - \mathrm{Im}(h(0)) = \frac{\mathrm{d}}{\mathrm{d}t}\mathrm{Im}(h(t_2))$. Setting $c_1 = t_1 z + (1 - t_1)w$ and $c_2 = t_2 z + (1 - t_2)w$ this implies

$$G_j(w) - G_j(z) = (\nabla_{\mathrm{Re}}\mathrm{Re}(G_j(c_1)))^T(z - w) + i\nabla_{\mathrm{Re}}\mathrm{Im}(G(c_2)))^T(z - w),$$

where $\nabla_{\mathrm{Re}}G$ denotes the vector of partial derivatives with respect to the real variable. Let us denote by G_j' the complex derivative of G_j; that is, $G_j' : \mathbb{C}^n \to \mathbb{C}^n$ as a function. From the Cauchy Riemann equations it follows that $\nabla_{\mathrm{Re}}\mathrm{Re}(G_j'(c_1)) = \mathrm{Re}(G_j'(c_1))$ and likewise $\nabla_{\mathrm{Re}}\mathrm{Im}(G(c_2)) = \mathrm{Im}(G'(c_2))$. This yields $G_j(z)-G_j(w) = (\mathrm{Re}(G_j'(c_1)+i\mathrm{Im}(G_j'(c_2)))^T(z-w)$. Putting these equations ranging over j together we find $G(z)-G(w) = (\mathrm{Re}(\mathrm{J}G(c_1)) + i\mathrm{Im}(\mathrm{J}G(c_2)))(z - w)$. By construction, c_1 and c_2 are contained in I, because w and z are contained in I, and I is a product of rectangles and thus convex. Combined with (5.4) this yields

$$G(z) - G(w) \in (\mathbf{1}_n - Y \cdot \square\mathrm{J}F(I))(z - w).$$

Using essentially the same arguments for the path from x to w, we also find

$$G(w) - G(x) \in (\mathbf{1}_n - Y \cdot \square\mathrm{J}F(I))(w - x).$$

By construction, $z - w = i\mathrm{Im}(z - x)$ and $w - x = \mathrm{Re}(z - x)$. This implies

$$G(z) - G(x) \in (\mathbf{1}_n - Y \cdot \square\mathrm{J}F(I))\,\mathrm{Re}(z - x) + (\mathbf{1}_n - Y \cdot \square\mathrm{J}F(I))\,i\mathrm{Im}(z - x).$$

This finishes the proof. □

Proof of Theorem 5.7 and Proposition 5.10. We fix $Y \in \mathbb{C}^{n\times n}$. The second part of Lemma 5.11 implies that, if we have $K_{x,Y}(I) \subseteq I$, then $G_Y(I) \subseteq I$. Brouwer's fixed point Theorem

shows that G_Y has a fixed point in I. Since Y is assumed to be invertible, the fixed point is a zero of F. This finishes the proof for the first part of Theorem 5.7. For the second part, let $z_1, z_2 \in I$. The first part of Lemma 5.11 implies

$$G_Y(z_1) - G_Y(z_2) \in (\mathbf{1}_n - Y \cdot \square \mathrm{J}F(I))\mathsf{Re}(z_1 - z_2) + (\mathbf{1}_n - Y \cdot \square \mathrm{J}F(I))\ i\mathsf{Im}(z_1 - z_2).$$

(Note that we can't apply the distributivity law because of Theorem 5.1 3.). Applying norms and using submultiplicativity yields

$$\|G_Y(z_1) - G_Y(z_2)\|_\infty \leq \|(\mathbf{1}_n - Y \cdot \square \mathrm{J}F(I))\|_\infty \left(\|\mathsf{Re}(z_1 - z_2)\|_\infty + \|\mathsf{Im}(z_1 - z_2)\|_\infty\right).$$

Since $\|\mathsf{Re}(z_1 - z_2)\|_\infty + \|\mathsf{Im}(z_1 - z_2)\|_\infty \leq \sqrt{2}\|z_1 - z_2\|_\infty$, it holds

$$\|G_Y(z_1) - G_Y(z_2)\|_\infty \leq \sqrt{2}\|\mathbf{1}_n - Y \cdot \square \mathrm{J}F(I)\|_\infty \|z_1 - z_2\|_\infty.$$

By assumption $\sqrt{2}\|\mathbf{1}_n - Y \cdot \square \mathrm{J}F(I)\|_\infty$ is smaller than 1 so G_Y is a contraction. Banach's Fixed Point Theorem implies that G_Y has a unique zero in I. This shows the second part of Theorem 5.7. The fact that G_Y is a contraction on I also proves Proposition 5.10. □

Certifying Reality

For many applications, only the real zeros of a polynomial system are of interest. Since numerical homotopy continuation computes in $\mathbb{C}^n$, it is important to have a rigorous method to determine whether a zero is real.

Recall from Definition 5.8 the notion of *strong interval approximate zero*.

Lemma 5.12. *Let $F : \mathbb{C}^n \to \mathbb{C}^n$ be a real square system of polynomials and $I \in \mathbb{IC}^n$ a strong interval approximate zero of F. Then there exists $x \in I$ and $Y \in \mathbb{C}^{n\times n}$ satisfying $K_{x,Y}(I) \subset I$ and $\sqrt{2}\,\|\mathbf{1}_n - Y\square \mathrm{J}F(I)\|_\infty < 1$. If additionally $\{\bar{z} \,|\, z \in K_{x,Y}(I)\} \subset I$, the associated zero of I is real.*

Proof. Theorem 5.7 implies that F has a unique zero $s \in K_{x,Y}(I) \subset I$. Since F is a real polynomial system, it follows that also the element-wise complex conjugate $\bar{s}$ is a zero of F. If we have that $\bar{s} \in \{\bar{z} \,|\, z \in K_{x,Y}(I)\} \subset I$, then $\bar{s} = s$, since otherwise $\bar{s}$ and s would be two distinct zeros of F in I, contradicting the uniqueness result from Theorem 5.7. □

For a wide range of applications, positive real zeros are of particular interest.

Corollary 5.13. *Let $F : \mathbb{C}^n \to \mathbb{C}^n$ be a real square system of polynomials and $I \in \mathbb{IC}^n$ a strong interval approximate zero of F satisfying the conditions of Lemma 5.12. If $\mathsf{Re}(I) > 0$, then the associated zero of I is real and positive.*

If the reality test in Lemma 5.12 fails for a strong interval approximate zero $I \in \mathbb{C}^n$, then this does not necessarily mean that the associated zero of I is not real. A sufficient condition that I is not real is that there is a coordinate such that the imaginary part of it does not contain zero.

Lemma 5.14. *Let $F(x)$ be a square system of polynomials or rational functions and let $I \in \mathbb{IC}^n$ be a strong interval approximate zero of F. If there exists $k \in \{1, \ldots, n\}$ such that $0 \notin \operatorname{Im}(I_k)$, then the associated zero of I is not real.*

Proof. The associated zero x of I is contained in I. Since $0 \notin \operatorname{Im}(I_k)$, it follows $x_k \notin \mathbb{R}$ and $x \notin \mathbb{R}^n$. □

Now assume that the certification routine produced a list $\mathcal{I}$ of m distinct strong interval approximate zeros for a given system F and that m also agrees with the theoretical upper bound on the number of isolated zeros of F. If we apply Lemma 5.12 to $I_k \in \mathcal{I}$, then we obtain only a *lower bound*, say r, on the number of real zeros of F. However, combined with Lemma 5.14, we can also obtain an *upper bound* of the number of real zeros. If these two bounds agree, we obtain a certificate that F has *exactly* r real zeros. An application of this is, e.g., the study of the distribution of the number of real solutions of the power flow equations [LZBL20].

5.4 Implementation Details

In this section, we describe the necessary considerations to implement Krawczyk's method described in Section 5.3 as well as the technical realization in `HomotopyContinuation.jl`. The certification routine takes as input a square polynomial system $F : \mathbb{C}^n \to \mathbb{C}^n$ and a finite list $X \subset \mathbb{C}^n$ of (suspected) approximations of isolated zeros of F. It is also possible to provide a square system of rational functions as input. Similar to Section 5.3, we restrict to the polynomial case to simplify our exposition. It returns a list of strong interval approximate zeros $\mathcal{I} = \{I_1, \ldots, I_m\} \in \mathbb{IC}^n$ such that no two intervals I_k and I_ℓ, $k \neq \ell$, overlap. If two strong interval approximate zeros don't overlap, then this implies that their associated zeros are distinct. Additionally, if F is a real polynomial system then for each $I_k \in \mathcal{I}$, it is determined whether its associated zero is real. The prototypical application of the certification routine is to take as input approximations of all isolated solutions $X \subset \mathbb{C}^n$ of F as computed by numerical homotopy continuation methods.

Interval Enclosures for Polynomial Systems

As already discussed in Section 5.2, the fact that distributivity doesn't hold in $\mathbb{IC}$ requires that the polynomial system $F : \mathbb{C}^n \to \mathbb{C}^n$ and its interval enclosure $\square F$ have to be defined by a straight-line program, and not just by a list of coefficients. The overestimation of the interval enclosure $\square F$ increases with the size of the straight-line program. Therefore, it is good to express F and its enclosure $\square F$ by the smallest straight-line program possible. To achieve this, `HomotopyContinuation.jl` automatically applies a multivariate version of Horner's rule to reduce the number of operations necessary to evaluate F and $\square F$. The representation of straight-line programs in `HomotopyContinuation.jl` is also discussed in Section 9.2.1.

Remark 5.15. Our implementation of interval enclosures can also be used to prove that a polynomial map $F : \mathbb{C}^n \to \mathbb{C}^m$ with real coefficients evaluated at a real point $p \in \mathbb{R}^n$ is

positive. To verify this, one takes an interval $I \in \mathbb{IC}^n$ of the form $I = J + i[0,0]^{\times n}$ such that $p \in J$. If $\square F$ is an interval enclosure of F, and if $\square F(I) \subset \mathbb{R}^m_{>0} + i[0,0]^{\times m}$, then this is a proof that $F(p) \in \mathbb{R}^m_{>0}$.

Machine Interval Arithmetic

In the next subsection, we give a method to construct a candidate $I \in \mathbb{IC}^n$ for a strong interval approximate zero. Before, we need to study *machine interval arithmetic*; the realization of interval arithmetic with finite-precision floating-point arithmetic. We assume the standard model of floating-point arithmetic [Hig02, Section 2.3] where the result of a floating-point operation is accurate up to relative unit roundoff u: $\mathrm{fl}(x \circ y) = (x \circ y)(1+\delta)$, where $|\delta| \leq u$ and $\circ \in \{+, -, *, /\}$. For instance, following the IEE-754 standard, the unit roundoff in double precision arithmetic is $u = 2^{-53} \approx 2.2 \cdot 10^{-16}$. The key property in the context of interval arithmetic is that each result of a floating-point operation can be rounded outwards such that the resulting *interval* contains the true (exact) result; see, e.g., [May17, Section 3.2]. Therefore, given $X, Y \in \mathbb{IC}$, the result of $X \circ Y$, $\circ \in \{+, -, *, /\}$, is $\mathrm{fl}(X \circ Y) := \{(x \circ y)(1+\delta) \,|\, |\delta| \leq u, x \in X, y \in Y\}$ in machine arithmetic. This interval contains $X \circ Y$. It is *larger*. Additionally, for a given $x \in \mathbb{IC}$, all intervals of the form $\{x + (|\mathrm{Re}(x_j)| + i|\mathrm{Im}(x_j)|)\delta \,|\, |\delta| \leq \mu\}$ with $0 < \mu \leq u$ are indistinguishable when working with precision u.

As a consequence, it is possible that the Krawczyk operator $K_{\tilde{x},Y}$, see Definition 5.4, is a contraction for the interval I but that machine arithmetic can't verify this because $\mathrm{fl}(X \circ Y)$ is larger than $X \circ Y$. In such a case, the unit roundoff u needs to be sufficiently decreased. For this reason, our implementation uses machine interval arithmetic based on double-precision arithmetic as well as, if necessary, the arbitrary precision interval arithmetic implemented in `Arb` [Joh17].

Determining Strong Interval Approximate Zeros

In a first step, the certification routine attempts to produce for a given $x \in X$ a strong interval approximate zero $I \in \mathbb{IC}^n$. Recall that for $I \in \mathbb{IC}^n$ to be a strong interval approximate zero, we need by Theorem 5.7 to have a point $\tilde{x} \in I$ and a matrix $Y \in \mathbb{C}^{n \times n}$ such that $K_{\tilde{x},Y}(I) \subset I$ and $\sqrt{2}\,\|\mathbf{1}_n - Y \square \mathrm{J}F(I)\|_\infty < 1$.

Given a point $x \in X$ and a unit roundoff u, the point x is refined using Newton's method to maximal accuracy. We denote this refined point $\tilde{x}$. Here, we assume that x is already in the region of quadratic convergence of Newton's method. Next, the point $\tilde{x}$ needs to be inflated to an interval I with $\tilde{x} \in I$. This process is called ε-inflation in the literature [May17, Sec. 4.3]. However, choosing the correct I is a hard problem: if I is too small or too large, then the Krawczyk operator is not a contraction.

Despite these difficulties, we found the following heuristic to determine I work very well. If we assume $\tilde{x}$ to be in the region of quadratic convergence of Newton's method, it follows from the Newton-Kantorovich theorem that $||\mathrm{J}F(\tilde{x})^{-1}F(\tilde{x})||_\infty$ is a good estimate of the distance between $\tilde{x}$ and the convergence limit x^*. Therefore, we set $Y \approx \mathrm{J}F(\tilde{x})^{-1}$ (computed in floating-point arithmetic) and use $I = (\tilde{x}_j \pm |(Y \cdot \square F(x))_j| u^{-\frac{1}{4}})_{j=1,\ldots,n}$ where the factor

$u^{-\frac{1}{4}}$ accounts for the overestimation by machine interval arithmetic. If I doesn't satisfy the conditions in Theorem 5.7, the procedure is repeated with a smaller unit roundoff u. This repeats until either a minimal unit roundoff is reached or the certification is successful.

Producing Distinct Intervals

Assume now that the steps in Section 5.4 have been performed for all $x \in X$. We obtain a list of strong interval approximate zeros $I_1, \ldots, I_r \in \mathbb{IC}^n$. In a final step, we want to select a subset $M \subset \{1, \ldots, r\}$ such that for all $k, j \in M$, $k \neq j$, the intervals I_k and I_j do not overlap. If two strong interval approximate zeros do not overlap, then it is guaranteed that they have distinct associated zeros. A simple approach to determine M is to compare all intervals pairwise. However, this approach requires us to perform $\binom{r}{2}$ interval vector comparisons. For larger problems, this becomes prohibitively expensive.

Instead, we employ the following improved scheme to determine all non-overlapping intervals. First, we pick a random point $q \in \mathbb{C}^n$ and compute in interval arithmetic for each I_k, $k \in M$, the squared Euclidean distance $d_k \in \mathbb{IR}$ between I_k and q. Due to the guarantees of interval arithmetic, we have that d_k and d_ℓ overlap if I_k and I_ℓ overlap (but the converse is not necessarily true). Next, we check for all overlapping intervals d_k and d_ℓ, $1 \leq k < \ell \leq r$, whether I_k and I_ℓ overlap, and if so, we group them accordingly. This allows us to construct the set M by selecting those intervals that don't overlap with any other and by picking one representative of each cluster of overlapping intervals. The worst-case complexity of this procedure still requires $O(r^2)$ operations, but in the common case where no or only a small number of intervals overlap, $O(r \log r)$ operations are sufficient.

5.5 Conclusion

In this chapter, we established interval arithmetic as a practical tool for certification in numerical nonlinear algebra. We described how Krawczyk's method allows us to obtain from a numerical solution a strong interval approximate zero. A strong interval approximate zero describes an interval box that contains a unique isolated regular solution to a square system of polynomial equations. The certification of isolated solutions is an important tool in numerical nonlinear algebra since it allows to use numerical results for mathematical proofs. We saw already in Chapter 2 an example of this with our fully real instance of Steiner's conic problem in Proposition 2.1. We also presented an implementation of the developed certification routine in form of the function `certify` in `HomotopyContinuation.jl`. We demonstrated that our implementation dramatically outperforms earlier approaches to certification. In the next chapter, we will see a problem where only this increase in certification performance made certification practically feasible.

6 Computing the Degree of the Linear Orbit of a Cubic Surface

This chapter is based on the article "96120: The degree of the linear orbit of a cubic surface" [BIMTW20] by Laura Brustenga I Moncusì, Sascha Timme and Madeleine Weinstein. The article is published in the Le Mathematiche *special issue on "Twenty-Seven Questions about the Cubic Surface".*

This chapter demonstrates an application of numerical nonlinear algebra to a problem in classic algebraic geometry. We study the action of the projective linear group $\mathrm{PGL}(\mathbb{C}, 4)$ on cubic surfaces parameterized by points in $\mathbb{P}^{19}$. Automorphism groups of varieties and group actions on varieties are of much interest to researchers of algebraic geometry, arithmetic, and representation theory [AF93, BI17, MM64, Vai03]. In particular, we compute the degree of the 15-dimensional projective variety in $\mathbb{P}^{19}$ defined by the Zariski closure of the orbit of a general cubic surface under this action. This degree is also meaningful in enumerative geometry: It is the number of translates of a cubic surface that pass through 15 points in general position. This formulation provides an alternate method for obtaining the degree.

Aluffi and Faber considered the analogous problem for plane curves of arbitrary degree. First, the smooth case in [AF93] and then the general case in [AF00]. They obtained a closed formula for the degree of the orbit closure of a plane curve under the action of $\mathrm{PGL}(\mathbb{C}, 3)$. This was a significant undertaking, involving long and detailed calculations in intersection rings using advanced techniques from intersection theory.

Instead of adopting the techniques developed by Aluffi and Faber, we use tools from numerical nonlinear algebra. The general idea is as follows. We fix a cubic surface f and 15 points in general position in $\mathbb{P}^3$. The condition that a translate of f passes through these 15 points results in a polynomial system for which we compute all isolated numerical solutions by homotopy continuation and the monodromy method using `HomotopyContinuation.jl`. Using the certification technique developed in Chapter 5, we establish a hard lower bound on the number of solutions to this system. Finally, we use the *trace test* described in Section 3.6.2 to check that no solution is missing. With these techniques, we conclude that the number of numerical solutions we obtain, 96120, is in fact the degree of the orbit closure. This result is a "numerical theorem" rather than a theorem in the classical sense since we only have strong numerical evidence.

In the following, we introduce the linear orbit problem in detail and derive the polynomial systems used in our computations. Afterward, we describe the computations performed to arrive at the result.

6.1 Linear Orbits and Polynomial Systems

A cubic surface in $\mathbb{P}^3$ is defined by a cubic homogeneous polynomial in 4 variables with complex coefficients. The parameter space for cubic surfaces is $\mathbb{P}^{19}$ and we fix coordinates $[c_0 : \cdots : c_{19}] \in \mathbb{P}^{19}$.

The projective space $\mathbb{P}^{15}$ of homogeneous 4×4 matrices $A = (a_{ij})_{1 \le i,j \le 4}$ is a compactification of the projective general linear group

$$\mathrm{PGL}(\mathbb{C}, 4) = \{A \in \mathbb{P}^{15} \mid \det(A) \neq 0\} \subseteq \mathbb{P}^{15} .$$

The group $\mathrm{PGL}(\mathbb{C}, 4)$ acts on a cubic surface $f \in \mathbb{P}^{19}$, with $\varphi \in \mathrm{PGL}(\mathbb{C}, 4)$ sending f to the cubic surface $\varphi \cdot f$ defined by the equation

$$f(\varphi(x, y, z, w)) = 0 .$$

This corresponds to a linear change of the coordinates x, y, z, w. We say that $\varphi \cdot f$ is the *translate* of f by φ. Then $\mathrm{PGL}(\mathbb{C}, 4) \cdot f$ is the orbit of f in $\mathbb{P}^{19}$ and its Zariski closure $\Omega_f := \overline{\mathrm{PGL}(\mathbb{C}, 4) \cdot f}$ is a 15-dimensional projective variety.

Example. To illustrate this idea, we consider the action of $\mathrm{PGL}(\mathbb{C}, 2)$ on pairs of points defined by homogeneous polynomials

$$f(x, y) = b_0 x^2 + b_1 xy + b_2 y^2 .$$

The parameter space for pairs of points is $\mathbb{P}^2$, that is $f = (b_0 : b_1 : b_2) \in \mathbb{P}^2$. Let

$$\varphi = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} .$$

Then

$$\begin{aligned} f(\varphi(x, y)) =& b_1(a_{11}x + a_{12}y)^2 + b_2(a_{11}x + a_{12}y)(a_{21}x + a_{22}y) + b_3(a_{21}x + a_{22}y)^2 \\ =& (b_1 a_{11}^2 + b_2 a_{11} a_{21} + b_3 a_{21}^2)x^2 + \\ & (2b_1 a_{11} a_{12} + b_2(a_{11}a_{22} + a_{12}a_{21}) + 2b_3 a_{21} a_{22})xy + \\ & (b_1 a_{12}^2 + b_2 a_{12} a_{22} + b_3 a_{22}^2)y^2 . \end{aligned}$$

and thus

$$\begin{aligned} \varphi \cdot f = (& b_1 a_{11}^2 + b_2 a_{11} a_{21} + b_3 a_{21}^2 : \\ & 2b_1 a_{11} a_{12} + b_2(a_{11}a_{22} + a_{12}a_{21}) + 2b_3 a_{21} a_{22} : \\ & b_1 a_{12}^2 + b_2 a_{12} a_{22} + b_3 a_{22}^2) \in \mathbb{P}^2 . \end{aligned}$$

To compute the degree of the orbit closure of a general cubic surface under the action of $\mathrm{PGL}(\mathbb{C}, 4)$, we construct as follows polynomial systems whose number of isolated regular solutions correspond to the desired degree.

Fix a general cubic surface $f \in \mathbb{P}^{19}$ and a general linear subspace $L \subseteq \mathbb{P}^{19}$ of dimension 4, the codimension of Ω_f. Consider the rational map

$$\Theta_f : \mathbb{P}^{15} \to \mathbb{P}^{19}$$

sending a 4×4 matrix φ to $\varphi \cdot f$. By definition, the image of Θ_f is Ω_f. By [MM64, Theorem 5], a generic hypersurface of degree at least three in at least four variables has a trivial stabilizer (we note that in [BI17, Propostion 7.5] it is stated that argument in [MM64] has an error but that it does not affect the correctness of the statement). Hence, the map Θ_f is one-to-one, so the degrees of the zero-dimensional varieties $\Omega_f \cap L$ and $\Theta_f^{-1}(\Omega_f \cap L) = \Theta_f^{-1}(L)$ are equal.

Note that $\Theta_f^{-1}(L)$ includes non-invertible matrices whose kernel does not contain f. But since we assume $L \subseteq \mathbb{P}^{19}$ to be general, $\Theta_f^{-1}(L)$ will not intersect the codimension 1 subvariety of $\mathbb{P}^{15}$ of matrices with determinant equal to 0. It follows that the degree of the orbit closure is the number of regular isolated solutions of the polynomial system

$$\tilde{L}\, \varphi \cdot f = 0 \tag{6.1}$$

in the entries of $\varphi \in \mathbb{P}^{15}$, where $\tilde{L} \in \mathbb{C}^{15 \times 20}$ is a matrix representing the general linear subspace $L \subseteq \mathbb{P}^{19}$ of dimension 4.

The degree of Ω_f can be thought of in enumerative terms as the number of translates of f that pass through 15 points $p_1, \ldots, p_{15} \in \mathbb{P}^3$ in general position. Consider the translated cubic surface $\varphi \cdot f$. Note that $\varphi \cdot f$ passes through a point $p \in \mathbb{P}^3$ if and only if $f(\varphi(p)) = 0$. Therefore we obtain the polynomial system

$$f(\varphi(p_i)) = 0 \ , \ i = 1, \ldots, 15 \tag{6.2}$$

in the entries of $\mathbb{P}^{15}$. By Bertini's theorem, we assume that the hypersurfaces $f(\varphi(p_i)) = 0$ intersect transversally. Hence, the degree of Ω_f is equal to the number of matrices satisfying equation (6.2).

Formulations (6.1) and (6.2) both result in a system of 15 homogeneous cubic polynomials in the 16 unknowns $(a_{ij})_{1 \leq i,j \leq 4}$, but they have different computational advantages. To perform numerical homotopy continuation, it is beneficial to pass to an affine chart of projective space. This can be done in formulation (6.1) by fixing a coordinate, say adding the polynomial $a_{11} - 1 = 0$. But this introduces artificial solutions. For example, for every solution $\phi \in \mathbb{C}^{16}$, we have that $e^{i\frac{2}{3}\pi}\phi$ and $e^{i\frac{4}{3}\pi}\phi$ are also solutions. The formulation (6.2) does not produce these undesired artificial solutions. Additionally, this formulation is also faster to evaluate and produces less numerical error. To check whether we found all solutions, we want to perform a trace test. But for this, the formulation (6.1) is better suited. Since this formulation corresponds to the intersection of a positive-dimensional variety with a linear space, we can apply the trace test directly. To perform a trace test with the formulation (6.2) would require us to first compute additional solutions as explained in Section 3.6.2.

6.2 A Numerical Approach

In this section, we explain our use of numerical nonlinear algebra to obtain Theorem* 6.1 below. We refrain from stating this result as a theorem since we currently cannot certify the last step of our computation. We add the asterisk to acknowledge this gap.

Theorem* 6.1. *The degree of the orbit closure of a general cubic surface under the action of* $\mathrm{PGL}(\mathbb{C}, 4)$ *is 96120.*

All computations performed to arrive at this result are available from the authors upon request.

To compute the degree of the orbit closure, we sample a general cubic surface $f \in \mathbb{P}^{19}$ by drawing the real and imaginary parts of each of its coordinates independently from a univariate normal distribution. We then solve the polynomial system (6.2) encoding the enumerative geometry problem. A naive strategy is to sample 15 points $p_1, \ldots, p_{15} \in \mathbb{P}^3$ in general position and use a total degree homotopy, but in this case the Bézout bound is $3^{15} = 14,348,907$. Here, the monodromy method described in Chapter 3.5 is substantially more efficient.

To apply the monodromy method, we consider (6.2) as a polynomial system on the entries of φ parameterized by 15 points $p_1, \ldots, p_{15}$ in $\mathbb{P}^3$. We consider the incidence variety

$$V = \{(\varphi, (p_1, \ldots, p_{15})) \in \mathbb{P}^{15} \times (\mathbb{P}^3)^{15} \mid F(\varphi(p_i)) = 0,\, i = 1, \ldots, 15\}$$

and we denote by π the projection $\mathbb{P}^{15} \times (\mathbb{P}^3)^{15} \to (\mathbb{P}^3)^{15}$ restricted to V.

We find a start pair $(\varphi_0; p_1, \ldots, p_{15}) \in V$ and then we use the monodromy action on the fiber $\pi^{-1}(p_1, \ldots, p_{15})$ to find all solutions in this fiber. Such a start pair can be found by exchanging the role of variables and parameters. First, we sample a $\varphi_0 \in \mathbb{P}^{15}$ and the first three coordinates of 15 points $p_i \in \mathbb{P}^3$ in general position. This yields a system of 15 polynomials each depending only on a unique variable: The ith polynomial depends only on the fourth coordinate of p_i. Such a system is easy to solve. Solving it yields a start pair $(\varphi_0; p_1, \ldots, p_{15}) \in V$, on which we run the monodromy method implemented in `HomotopyContinuation.jl`. In less than an hour, this method found 96120 approximate solutions corresponding to the start points $p_1, \ldots, p_{15} \in \mathbb{P}^3$. Applying the certification routine implemented in `HomotopyContinuation.jl` and described in Chapter 5, we certify in less than 5 minutes that our approximate solutions correspond to 96120 distinct isolated solutions of the system.

Remark 6.2. The article [BIMTW20] on which this chapter is based performed the certification using the software `alphaCertified` [HS12]. However, we were only able to obtain a certificate using floating-point arithmetic due to computational limits. This certificate is not rigorous due to possible floating-point errors. For a rigorous certificate, it would have been necessary to use rational arithmetic. The computation for the floating-point certificate needed 5 hours and 80GB of memory. The new certification routine described in Chapter 5 allowed us to close this gap. In contrast to the `alphaCertified` computation, it only needed around one minute and substantially less memory.

The certification process establishes a *lower* bound on the degree of the orbit closure. As the last step, we run a trace test to verify that we have indeed found *all* solutions. For this, we construct a linear subspace L from the 15 points $p_1, \ldots, p_{15}$ such that our solutions from the monodromy computation are also solutions to (6.1). The result of the trace test is theoretically zero if we found all solutions. In our numerical computation, we obtained a value on the order of the machine precision giving us very high confidence that we found indeed all solutions. We conclude that the degree of the orbit closure of a general cubic surface under the action $\mathrm{PGL}(\mathbb{C}, 4)$ is 96120.

We note that as a test of our methods, we confirmed known degrees of other varieties. In agreement with a theoretical result of Aluffi and Faber [AF93], we computed that the degree of the orbit closure of a general quartic curve in the plane is 14280. Additionally, we computed that the degree of the orbit closure of the Cayley cubic, defined by the equation $yzw+xzw+xyw+xyz=0$, is 305. Due to the symmetry of the variables in the Cayley cubic, there are $4!$ matrices corresponding to every polynomial in the orbit. As expected, we computed $7320 = 4! \cdot 305$ solutions. This coincides with a theoretical result of Vainsencher [Vai03].

6.3 Conclusion

In this chapter, we computed the degree of the orbit closure of the action of the projective linear group $\mathrm{PGL}(\mathbb{C}, 4)$ on cubic surfaces parameterized by points in $\mathbb{P}^{19}$ using methods from numerical nonlinear algebra. Our result is that the degree is 96120. To compute and verify this degree, we used the monodromy method, a trace test and the certification of isolated solutions. The monodromy method allowed us to quickly compute the 96120 isolated solutions corresponding to the degree of the orbit closure. The certification of the computed 96120 isolated solutions established a rigorous lower bound on the degree of the orbit closure. To strengthen our result, we also employed a trace test to check that we did not miss any solutions with the monodromy method. Unfortunately, the trace test is not rigorous as discussed in Section 3.6. This chapter illustrates the importance of finding a rigorous trace test. Given a rigorous trace test, we could drop the asterisks from Theorem* 6.1.

7 Estimating Linear Covariance Models

This chapter is based on the article "Estimating linear covariance models with numerical nonlinear algebra" [STZ20] by Bernd Sturmfels, Sascha Timme and Piotr Zwiernik. The article appeared in Algebraic Statistics *volume 11 number 1.*

In this chapter, we demonstrate the application of numerical nonlinear algebra in statistics. In many statistical applications, the covariance matrix Σ has a special structure. A natural setting is that one imposes linear constraints on Σ or its inverse Σ^{-1}. We here study models for Gaussians whose covariance matrix Σ lies in a given linear space. Such linear Gaussian covariance models were introduced by Anderson [And70]. He was motivated by the Toeplitz structure of Σ in time series analysis. Recent applications of such models include repeated time series, longitudinal data, and a range of engineering problems [Pou99]. Other occurrences are Brownian motion tree models [SUZ20], as well as pairwise independence models, where some entries of Σ are set to zero.

The literature on estimating a covariance matrix is extremely rich. Its development has been particularly dynamic in high-dimensional statistics under sparsity assumption on Σ or its inverse; see [FLL16] for an overview. Although the sample covariance matrix is known to have poor statistical properties, for many Gaussian models the maximum likelihood estimator (MLE) remains an important reference point.

Maximum likelihood estimation for linear covariance models is a nonlinear algebraic optimization problem over a spectrahedral cone, namely the convex cone of positive definite matrices Σ that satisfy the linear constraints of interest. The objective function is not convex and can have multiple local maxima. Yet, if the sample size is large relative to the dimension, then the problem is essentially convex. This was shown in [ZUR17]. In general, however, the MLE problem is poorly understood, and there is a need for accurate methods that reliably identify all local maxima.

Nonlinear algebra furnishes such a method, namely solving the score equations [Sul18, Section 7.1] using numerical homotopy continuation [SW05]. This is guaranteed to find all critical points of the likelihood function and hence all local maxima. A key step is the knowledge of the maximum likelihood degree (ML degree). This is the number of complex critical points. The ML degree of a linear covariance model is an invariant of a linear space of symmetric matrices which is of interest in its own right.

Our presentation is organized as follows. In Section 7.1, we introduce various models to be studied, ranging from generic linear equations to colored graph models. In Section 7.2, we discuss the maximum likelihood estimator as well as the dual maximum likelihood estimator. Starting from [Sul18, Proposition 7.1.10], we derive a convenient form of the score equations. The natural point of entry for an algebraic geometer is the study of generic linear constraints. This is our topic in Section 7.3. We compute a range of ML degrees and we compare them

to the dual degrees in [SU10, Section 2.2].

In Section 7.4, we present our software `LinearCovarianceModels.jl`. This is written in `Julia` and it is easy to use. It computes the ML degree and the dual ML degree for a given subspace $\mathcal{L}$, and it determines all complex critical points for a given sample covariance matrix S. Among these, it identifies the real and positive definite solutions, and it then selects those that are local maxima. The package is available at

`https://github.com/saschatimme/LinearCovarianceModels.jl` (7.1)

and rests on `HomotopyContinuation.jl`.

Section 7.5 discusses instances where the likelihood function has multiple local maxima. This is meant to underscore the strength of our approach. We then turn to models where the maximum is unique and the MLE is a rational function.

In Section 7.6, we examine Brownian motion tree models. Here the linear constraints are determined by a rooted phylogenetic tree. We study the ML degree and dual ML degree. We show that the latter equals one for binary trees, and we derive the explicit rational formula for their MLE. A census of these degrees is found in Table 7.6.

7.1 Linear Covariance Models

Let $\mathbb{S}^n$ be the $\binom{n+1}{2}$-dimensional real vector space of $n \times n$ symmetric matrices $\Sigma = (\sigma_{ij})$. The subset $\mathbb{S}^n_+$ of positive definite matrices is a full-dimensional open convex cone. Consider any linear subspace $\mathcal{L}$ of $\mathbb{S}^n$ whose intersection with $\mathbb{S}^n_+$ is nonempty. Then, $\mathbb{S}^n_+ \cap \mathcal{L}$ is a relatively open convex cone. In optimization, where one uses the closure, this is known as a *spectrahedral cone*. In statistics, the intersection $\mathbb{S}^n_+ \cap \mathcal{L}$ is a *linear covariance model*. These are the models we study in this chapter. In what follows, we discuss various families of linear spaces $\mathcal{L}$ that are of interest to us.

Generic linear constraints: Fix a positive integer $m \leq \binom{n+1}{2}$ and suppose that $\mathcal{L}$ is a generic linear subspace of $\mathbb{S}^n$. Here "generic" is meant in the sense of algebraic geometry, i.e. $\mathcal{L}$ is a point in the Grassmannian that lies outside a certain algebraic hypersurface. This hypersurface has measure zero, so a random subspace will be generic with probability one. For a geometer, it is natural to begin with the generic case, since its complexity controls the complexity of any special family of linear spaces. In particular, the ML degree for a generic $\mathcal{L}$ depends only on m and n, and this furnishes an upper bound for the ML degree of the special families below.

Diagonal covariance matrices: Here we take $m \leq n$ and we assume that $\mathcal{L}$ is a linear space that consists of diagonal matrices. Restricting to covariance matrices that are diagonal is natural when modeling independent Gaussians. We use the term *generic diagonal model* when $\mathcal{L}$ is a generic point in the $(n-m)m$-dimensional Grassmannian of m-dimensional subspaces inside the diagonal $n \times n$ matrices.

Brownian motion tree models: A tree is a connected graph with no cycles. A rooted tree is obtained by fixing a vertex, called the root, and directing all edges away from the root. Fix a rooted tree T with n leaves. Every vertex v of T defines a *clade*, namely the set

of leaves that are descendants of v. For the Brownian motion tree model on T, the space $\mathcal{L}$ is spanned by the rank-one matrices $e_A e_A^T$, where $e_A \in \{0,1\}^n$ is the indicator vector of A. Hence, if $\mathcal{C}$ is the set of all clades of T then

$$\Sigma \;=\; \sum_{A \in \mathcal{C}} \theta_A e_A e_A^T, \qquad \text{where } \theta_A \text{ are model parameters.} \tag{7.2}$$

The linear equations for the subspace $\mathcal{L}$ are $\sigma_{ij} = \sigma_{kl}$ whenever the least common ancestors $\mathrm{lca}(i,j)$ and $\mathrm{lca}(k,l)$ agree in the tree T. Assuming $\theta_A \geq 0$, the union of the models for all trees T is characterized by the ultrametric condition $\sigma_{ij} \geq \min\{\sigma_{ik}, \sigma_{jk}\} \geq 0$. Matrices of this form play an important role also in hierarchical clustering [HTF09, Section 14.3.12], phylogenetics [Fel73], and random walks on graphs [DMSM14].

Maximum likelihood estimation for this class of models is generally complicated but recently there has been progress (cf. [THA14, SUZ20]) on exploiting the nice structure of the matrices Σ above. In Section 7.6, we study computational aspects of the MLE, and, more importantly, we provide a significant advance by considering the dual MLE.

Covariance graph models: We consider models $\mathcal{L}$ that arise from imposing zero restrictions on entries of Σ. This was studied in [CDR07, DR02]. This is similar to Gaussian graphical models where zero restrictions are placed on the inverse Σ^{-1}. We encode the sparsity structure with a graph whose edges correspond to nonzero off-diagonal entries of Σ. Zero entries in Σ correspond to pairwise marginal independences. These arise in statistical modeling in the context of causal inference [CW93]. Models with zero restrictions on the covariance matrix are known as covariance graph models. Maximum likelihood in these Gaussian models can be carried out using Iterative Conditional Fitting [CDR07, DR02], which is implemented in the `ggm` package in R [Mar06].

Toeplitz matrices: Suppose $X = (X_1, \ldots, X_n)$ follows the autoregressive model of order 1, that is, $X_t = \rho X_{t-1} + \epsilon_t$, where $\rho \in \mathbb{R}$ and $\epsilon_t \sim N(0, \sigma)$ for some σ. Assume that the ϵ_t are mutually uncorrelated. Then $\mathrm{cov}(X_t, X_{t-k}) = \rho^k$, and hence Σ is a Toeplitz matrix. More generally, covariance matrices from stationary time series are Toeplitz. Multichannel and multidimensional processes have covariance matrices of block Toeplitz form [BLW82, MS87a]. Similarly, if X follows the moving average process of order q then $\mathrm{cov}(X_t, X_{t-k}) = \gamma_k$ if $k \leq q$ and it is zero otherwise; see, for example, [Ham94, Section 3.3]. Thus, in time series analysis, we encounter matrices like

$$\begin{bmatrix} \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 \\ \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 \\ \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 \\ \gamma_4 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 \end{bmatrix} \qquad \text{or} \qquad \begin{bmatrix} \gamma_0 & \gamma_1 & 0 & 0 & 0 \\ \gamma_1 & \gamma_0 & \gamma_1 & 0 & 0 \\ 0 & \gamma_1 & \gamma_0 & \gamma_1 & 0 \\ 0 & 0 & \gamma_1 & \gamma_0 & \gamma_1 \\ 0 & 0 & 0 & \gamma_1 & \gamma_0 \end{bmatrix}. \tag{7.3}$$

We found that the ML degree for such models is surprisingly low. This means that nonlinear algebra can reliably estimate Toeplitz matrices that are fairly large.

Colored covariance graph models: A generalization of covariance graph models is obtained by following [HL08], which introduces graphical models with vertex and edge sym-

metries. Models of this type also generalize the Toeplitz matrices and the Brownian motion tree models. Following the standard convention we use the same colors for edges or vertices when the corresponding entries of Σ are equal. The black color is considered neutral and encodes no restrictions.

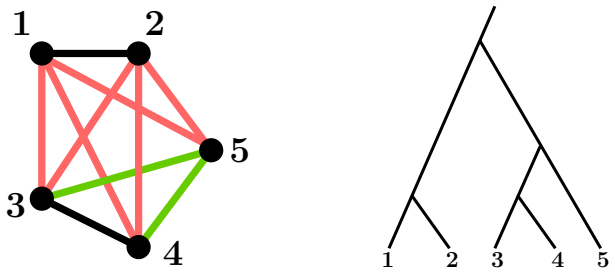

Figure 7.1: A covariance graph model with edge symmetries and the rooted tree for the corresponding Brownian motion tree model.

The Brownian motion tree model corresponds to a colored model over the complete graph, where edge symmetries are encoded by the tree; cf. Figure 7.1. Also, both matrices in (7.3) represent covariance graph models with edge and vertex symmetries.

7.2 Maximum Likelihood Estimator and Its Dual

Now that we have seen motivating examples, we formally define the MLE problem for a linear covariance model $\mathcal{L}$. Suppose we observe a random sample $X^{(1)}, \ldots, X^{(N)}$ in $\mathbb{R}^n$ from $N_n(0, \Sigma)$. The sample covariance matrix is $S = \frac{1}{N} \sum_{i=1}^{N} X^{(i)} X^{(i)^T}$. The matrix S is positive semidefinite. Our aim is to maximize the function

$$\ell(\Sigma) \;=\; \log\det \Sigma^{-1} - \mathrm{tr}(S\Sigma^{-1}) \qquad \text{subject to } \Sigma \in \mathcal{L}.$$

Following [Sul18, Proposition 7.1.10], this equals the log-likelihood function times $N/2$.

We fix the standard inner product $\langle A, B\rangle = \mathrm{tr}(AB)$ on the space $\mathbb{S}^n$ of symmetric matrices. The orthogonal complement $\mathcal{L}^\perp$ to a subspace $\mathcal{L} \subset \mathbb{S}^n$ is defined as usual.

Proposition 7.1. *Finding all the critical points of the log-likelihood function $\ell(\Sigma)$ amounts to solving the following system of linear and quadratic equations in $2 \cdot \binom{n+1}{2}$ unknowns:*

$$\Sigma \in \mathcal{L}, \qquad K\Sigma = I_n, \qquad KSK - K \in \mathcal{L}^\perp. \tag{7.4}$$

Proof. The matrix Σ is a critical point ℓ if and only if, for every $U \in \mathcal{L}$, the derivative of ℓ at Σ in the direction U vanishes. This directional derivative equals

$$-\mathrm{tr}(\Sigma^{-1}U) + \mathrm{tr}(S\Sigma^{-1}U\Sigma^{-1}).$$

This formula follows by multivariate calculus from two facts: (i) the derivative of the matrix mapping $\Sigma \mapsto \Sigma^{-1}$ is the linear transformation $U \mapsto \Sigma^{-1}U\Sigma^{-1}$; (ii) the derivative of the

function $\Sigma \mapsto \log\det\Sigma$ is the linear functional $U \mapsto \mathrm{tr}(\Sigma^{-1}U)$.

Using the identity $K = \Sigma^{-1}$, vanishing of the directional derivative is equivalent to

$$-\langle K, U\rangle + \langle KSK, U\rangle \;=\; 0.$$

The condition $\langle KSK - K, U\rangle = 0$ for all $U \in \mathcal{L}$ is equivalent to $KSK - K \in \mathcal{L}^\perp$. □

Example (3×3 Toeplitz matrices)**.** Let $\mathcal{L}$ be the space of Toeplitz matrices

$$\Sigma \;=\; \begin{bmatrix} \gamma_0 & \gamma_1 & \gamma_2 \\ \gamma_1 & \gamma_0 & \gamma_1 \\ \gamma_2 & \gamma_1 & \gamma_0 \end{bmatrix}.$$

This space has dimension 3 in $\mathbb{S}^3 \simeq \mathbb{R}^6$. Fix a sample covariance matrix $S = (s_{ij})$ with real entries. We need to solve the system (7.4). This consists of $3 + 9 + 3 = 15$ equations in $6 + 6 = 12$ unknowns, namely the entries of the covariance matrix $\Sigma = (\sigma_{ij})$ and its inverse $K = (k_{ij})$. The condition $\Sigma \in \mathcal{L}$ gives three linear polynomials:

$$\sigma_{11} - \sigma_{33}\,,\;\; \sigma_{12} - \sigma_{23}\,,\;\; \sigma_{22} - \sigma_{33}.$$

The condition $K\Sigma = I_3$ translates into nine bilinear polynomials:

$$\begin{array}{c} \sigma_{11}k_{11} + \sigma_{12}k_{12} + \sigma_{13}k_{13} - 1,\, \sigma_{12}k_{11} + \sigma_{22}k_{12} + \sigma_{23}k_{13},\, \sigma_{13}k_{11} + \sigma_{23}k_{12} + \sigma_{33}k_{13}, \\ \sigma_{11}k_{12} + \sigma_{12}k_{22} + \sigma_{13}k_{23},\, \sigma_{12}k_{12} + \sigma_{22}k_{22} + \sigma_{23}k_{23} - 1,\, \sigma_{13}k_{12} + \sigma_{23}k_{22} + \sigma_{33}k_{23}, \\ \sigma_{11}k_{13} + \sigma_{12}k_{23} + \sigma_{13}k_{33},\, \sigma_{12}k_{13} + \sigma_{22}k_{23} + \sigma_{23}k_{33},\, \sigma_{13}k_{13} + \sigma_{23}k_{23} + \sigma_{33}k_{33} - 1. \end{array}$$

Finally, the condition $KSK - K \in \mathcal{L}^\perp$ translates into three quadratic polynomials:

$$\begin{array}{c} k_{11}^2 s_{11} + k_{12}^2 s_{11} + k_{13}^2 s_{11} + 2k_{11}k_{12}s_{12} + 2k_{12}k_{22}s_{12} + 2k_{13}k_{23}s_{12} + 2k_{11}k_{13}s_{13} \\ +2k_{12}k_{23}s_{13} + 2k_{13}k_{33}s_{13} + k_{12}^2 s_{22} + k_{22}^2 s_{22} + k_{23}^2 s_{22} + 2k_{12}k_{13}s_{23} + 2k_{22}k_{23}s_{23} \\ +2k_{23}k_{33}s_{23} + k_{13}^2 s_{33} + k_{23}^2 s_{33} + k_{33}^2 s_{33} - k_{11} - k_{22} - k_{33}, \end{array}$$

$$\begin{array}{c} k_{23}s_{13} + k_{12}k_{33}s_{13} + k_{12}k_{22}s_{22} + k_{22}k_{23}s_{22} + k_{13}k_{22}s_{23} + k_{12}k_{23}s_{23} \\ +k_{23}^2 s_{23} + k_{22}k_{33}s_{23} + k_{13}k_{23}s_{33} + k_{23}k_{33}s_{33} - k_{12} - k_{23}, \end{array}$$

$$\begin{array}{c} k_{11}k_{13}s_{11} + k_{12}k_{13}s_{12} + k_{11}k_{23}s_{12} + k_{13}^2 s_{13} + k_{11}k_{33}s_{13} \\ +k_{12}k_{23}s_{22} + k_{13}k_{23}s_{23} + k_{12}k_{33}s_{23} + k_{13}k_{33}s_{33} - k_{13}. \end{array}$$

The zero set of these 15 polynomials in 12 unknowns consists of three points $(\hat{\Sigma}, \hat{K})$. We present a concrete instance with multiple local solutions:

$$S \;=\; \begin{bmatrix} 4/5 & -9/5 & -1/25 \\ -9/5 & 79/16 & 25/24 \\ -1/25 & 25/24 & 17/16 \end{bmatrix} \;\approx\; \begin{bmatrix} 0.8000 & -1.8000 & -0.0400 \\ -1.8000 & 4.9375 & 1.0417 \\ -0.0400 & 1.0417 & 1.0625 \end{bmatrix}. \qquad (7.5)$$

For this sample covariance matrix all three critical points are real and positive definite. The

three Toeplitz matrices that solve the score equations for this S are

$[\hat{\gamma}_0\,,\ \hat{\gamma}_1\,,\ \hat{\gamma}_2\,]$	log-likelihood value	
$[2.52783, -0.215929, -1.45229]$	-5.35	global maximum
$[2.39038, -0.286009, 0.949965]$	-5.41	local maximum
$[2.28596, -0.256394, 0.422321]$	-5.42	saddle point

So, even in this tiny example, our optimization problem has multiple local maxima in the cone $\mathbb{S}^3_+$. A numerical study of this phenomenon will be presented in Section 7.5.

In this chapter, we also consider the *dual* maximum likelihood estimator as a more computationally efficient alternative. Dual estimation is based on the maximization of a dual likelihood function. In the Gaussian case, this is motivated by interchanging the role of the parameter matrix Σ and the empirical covariance matrix S. The *Kullback-Leibler divergence* of two Gaussian distributions $N(0, \Sigma_0)$ and $N(0, \Sigma_1)$ on $\mathbb{R}^n$ is equal to

$$\mathrm{KL}(\Sigma_0, \Sigma_1) \ = \ \frac{1}{2}\left(\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) - n + \log\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right)\right).$$

Computing the MLE is equivalent to minimizing $\mathrm{KL}(\Sigma_0, \Sigma_1)$ with respect to Σ_1 with $\Sigma_0 = S$. On the other hand, the dual MLE is obtained by minimizing $\mathrm{KL}(\Sigma_0, \Sigma_1)$ with respect to Σ_0 with $\Sigma_1 = S$. Equivalently, we set $W = S^{-1}$ and we maximize

$$\ell^\vee(\Sigma) \ = \ \log\det\Sigma - \mathrm{tr}(W\Sigma).$$

The idea of utilizing the "wrong" Kullback-Leibler distance is ubiquitous in variational inference and is central for mean field approximation and related methods. The idea of using this estimation method for Gaussian linear covariance models is very natural. It results in a unique maximum, since $\Sigma \mapsto \ell^\vee(\Sigma)$ is a convex function on the positive definite cone $\mathbb{S}^n_+$. See [Chr89] and also [CDR07, Section 3.2] and [Kau96, Section 4].

The following algebraic formulation is the analogue to Proposition 7.1.

Proposition 7.2. *Finding all the critical points of the dual log-likelihood function $\ell^\vee$ amounts to solving the following system of equations in $2 \cdot \binom{n+1}{2}$ unknowns:*

$$\Sigma \in \mathcal{L}, \qquad K\Sigma = I_n, \qquad K - W \in \mathcal{L}^\perp. \tag{7.6}$$

Proof. After switching the role of K and Σ, and of W and S, our problem becomes MLE for linear concentration models. Formula (7.6) is found in [SU10, equation (10)]. □

The next result lists properties of the dual MLE that are important for statistics.

Proposition 7.3. *The dual maximum likelihood estimator of a Gaussian linear covariance model is consistent, asymptotically normal, and first-order efficient.*

Proof. See Theorem 3.1 and Theorem 3.2 in [Chr89]. □

First-order efficiency means that the asymptotic variance of the properly normalized dual MLE is optimal, that is, it equals the asymptotic variance of the MLE.

In this chapter, we focus on algebraic structures, and we note the following important distinction between our two estimators. The MLE requires the quadratic equations $KSK - K \in \mathcal{L}^\perp$ in (7.4), whereas the dual MLE requires the linear equations $K - W \in \mathcal{L}^\perp$ in (7.6). The latter are easier to solve than the former, and they give far fewer solutions. This is quantified by the tables for the ML degrees in the next sections.

We are particularly interested in models whose dual ML estimator $(\check{\Sigma}, \check{K})$ can be written as an explicit expression in the sample covariance matrix S. We identify such scenarios in Sections 7.5 and 7.6. Here is a first example to illustrate this point.

Example. We revisit the Toeplitz model in Example 7.2. For the dual MLE, the three quadratic polynomials in K are now replaced by three linear polynomials:

$$k_{11} + k_{22} + k_{33} - w_{11} - w_{22} - w_{33}\,,\ k_{12} + k_{23} - w_{12} - w_{23}\,,\ k_{13} - w_{13}.$$

The w_{ij} are the entries of the inverse sample covariance matrix $W = S^{-1}$. The new system has two solutions, and we can write the $\check{\sigma}_{ij}$ and $\check{k}_{ij}$ in terms of the w_{ij} (or the s_{ij}) using the familiar formula for solving quadratic equations in one variable. Specifically, for the covariance matrix S in (7.5) we find that the dual MLE is given by

$$\begin{aligned}[\check{\gamma}_0, \check{\gamma}_1, \check{\gamma}_2\,] \ &= \ \bigl[0.203557267562\,,\ -0.189349961613\,,\ 0.1963649733282\bigr] \\ &= \ \bigl[\,\tfrac{1284368265268038839512}{12363704694314904961417} + \tfrac{52\,\sqrt{56164777765459298768970215002736466708 1}}{12363704694314904961417}\,, \\ &\quad -\tfrac{5817390611804320873051}{6181852347157452480 7085} - \tfrac{655679934637\,\sqrt{56164777765459298768970215002736466708 1}}{16314690552471559970524472988 6305}\,, \\ &\quad \tfrac{1990451408446510673691859}{22254668449766828930550600} + \tfrac{264990063915733\,\sqrt{56164777765459298768970215002736466708 1}}{58732885988897615893888102759069800}\,\bigr].\end{aligned}$$

Needless to say, nonlinear algebra goes much beyond the quadratic formula. In what follows, we shall employ state-of-the-art methods for solving polynomial equations.

7.3 General Linear Constraints

The *maximum likelihood degree* of a linear covariance model $\mathcal{L}$ is, by definition, the number of complex solutions to the likelihood equations (7.4) for generic data S. This is abbreviated *ML degree*; see [Sul18, Section 7.1]. To compute the ML degree, take S to be a random symmetric $n \times n$ matrix and count all complex critical points of the likelihood function $\ell(\Sigma)$ for $\Sigma \in \mathcal{L}$. Equivalently, the ML degree of the model $\mathcal{L}$ is the number of complex solutions (Σ, K) to the polynomial equations in (7.4).

We also consider the complex critical points of the dual likelihood function $\ell^\vee(\Sigma)$. Their number, for a generic matrix $S \in \mathbb{S}^n$, is the *dual ML degree* of $\mathcal{L}$. It coincides with the number of complex solutions (Σ, K) to the polynomial equations in (7.6).

Our ML degrees can be computed symbolically in a computer algebra system that rests on Gröbner bases. However, this approach is limited to small instances. To get further, we use the methods from numerical nonlinear algebra described in Section 7.4.

We here focus on a generic m-dimensional linear subspace $\mathcal{L}$ of $\mathbb{S}^n$. In practice, this means that a basis for $\mathcal{L}$ is chosen by sampling m matrices at random from $\mathbb{S}^n$.

Proposition 7.4. *The ML degree and the dual ML degree of a generic subspace $\mathcal{L}$ of dimension m in $\mathbb{S}^n$ depends only on m and n. It is independent of the particular choice of $\mathcal{L}$. For small parameter values, these ML degrees are listed in Table 7.2.*

Proof. The independence rests on general results in algebraic geometry [SW05, Cor. A.14.2], to the effect that the system (7.4) (resp. (7.6)) can be considered as a system parametrized by the coordinates of $\mathcal{L}$ and S (resp. W). The ML degree will be the same for all specializations to $\mathbb{R}$ that remain outside a certain discriminant hypersurface. Table 7.2 and further values are computed rapidly using the software described in Section 7.4. □

m	n				
	2	3	4	5	6
2	1	3	5	7	9
3	1	7	19	37	61
4		7	45	135	299
5		3	71	361	1121
6		1	81	753	3395
7			63	1245	8513
8			29	1625	17867
9			7	1661	31601
10			1	1323	47343
11				801	60177
12				347	64731
13				97	58561
14				15	44131
15				1	27329
16					13627
17					5341
18					1511
19					289
20					31
21					1

m	n				
	2	3	4	5	6
2	1	2	3	4	5
3	1	4	9	16	25
4		4	17	44	90
5		2	21	86	240
6		1	21	137	528
7			17	188	1016
8			9	212	1696
9			3	188	2396
10			1	137	2886
11				86	3054
12				44	2886
13				16	2396
14				4	1696
15				1	1016
16					528
17					240
18					90
19					25
20					5
21					1

Table 7.2: ML degrees and dual ML degrees for generic models

The dual ML degree was already studied by Sturmfels and Uhler in [SU10, Section 2]. Our table on the right is in fact found in their paper. The symmetry along its columns is proved in [SU10, Theorem 2.3]. It states that the dual ML degree for dimension m coincides with the dual ML degree for codimension $m-1$. This is derived from the equations (7.6) by an appropriate homogenization. Namely, the middle equation is clearly symmetric under

switching the role of K and Σ, and the linear equations on the left and on the right in (7.6) can also be interchanged under this switch.

It was conjectured in [SU10, Section 2] that, for fixed m, the dual ML degree is a polynomial of degree $m-1$ in the matrix size n. This is easy to see for $m \leq 3$. The polynomials for $m = 4$ and $m = 5$ were also derived in [SU10, Section 2].

The situation is similar but more complicated for the ML degree. First of all, the symmetry along columns no longer holds as seen on the left in Table 7.2. This is explained by the fact that the linear equation $K - W \in \mathcal{L}^\perp$ is now replaced by the quadratic equation $KSK - K \in \mathcal{L}^\perp$. However, the polynomiality along the rows of Table 7.2 seems to persist. For $m = 2$, the ML degree equals $2n - 3$, as shown recently by Coons, Marigliano and Ruddy [CMR20]. For $m \geq 3$, we propose the following conjecture.

Conjecture 7.5. The ML degree of a linear covariance model of dimension m is a polynomial of degree $m-1$ in the ambient dimension n. For $m = 3$, this ML degree equals $3n^2-9n+7$, and for $m = 4$ it equals $11/3n^3 - 18n^2 + 85/3n - 15$.

We now come to *diagonal linear covariance models*. For these models, $\mathcal{L}$ is a linear subspace of dimension m inside the space $\mathbb{R}^n$ of diagonal $n \times n$-matrices. We wish to determine the ML degree and dual ML degree when $\mathcal{L}$ is generic in $\mathbb{R}^n$.

In the diagonal case, the score equations simplify as follows. Both the covariance matrix and the concentration matrix are diagonal. We eliminate the entries of Σ by setting $K = \mathrm{diag}(k_1, \ldots, k_n)$ and $\Sigma = \mathrm{diag}(k_1^{-1}, \ldots, k_n^{-1})$. We also write $s_1, \ldots, s_n$ for the diagonal entries of the sample covariance matrix S, and $w_i = s_i^{-1}$ for their reciprocals. Finally, let $\mathcal{L}^{-1}$ denote the *reciprocal linear space* of $\mathcal{L}$, i.e. the variety obtained as the closure of the set of coordinatewise reciprocals of vectors in $\mathcal{L} \cap (\mathbb{R}^*)^n$.

m \ n	3	4	5	6	7
2	3	5	7	9	11
3	1	7	17	31	49
4		1	15	49	111
5			1	31	129
6				1	63
7					1

m \ n	3	4	5	6	7
2	2	3	4	5	6
3	1	3	6	10	15
4		1	4	10	21
5			1	5	21
6				1	15
7					1

Table 7.3: ML degrees and dual ML degrees for generic diagonal models

Proposition 7.6. *Let $\mathcal{L} \subset \mathbb{R}^n$ be a linear space, viewed as a Gaussian covariance model of diagonal matrices. The score equations for the likelihood in (7.4) and the dual likelihood in (7.6) can be written as systems of n equations in n unknowns as follows:*

(7.4') $(k_1, \ldots, k_n) \in \mathcal{L}^{-1}$ *and* $(s_1k_1^2 - k_1, s_2k_2^2 - k_2, \ldots, s_nk_n^2 - k_n) \in \mathcal{L}^\perp$,

(7.6') $(k_1, \ldots, k_n) \in \mathcal{L}^{-1}$ *and* $(k_1 - w_1, k_2 - w_2, \ldots, k_n - w_n) \in \mathcal{L}^\perp$.

The number of complex solutions to (7.6') for generic $\mathcal{L}$ of dimension m equals $\binom{n-1}{m-1}$.

Proof. The translation of (7.4) and (7.6) to (7.4') and (7.6') is straightforward. The equations (7.6') represent a general linear section of the reciprocal linear space $\mathcal{L}^{-1}$. Proudfoot and Speyer showed that the degree of $\mathcal{L}^{-1}$ equals the Möbius invariant of the underlying matroid. We refer to [KV19] for a recent study and many references. This Möbius invariant equals $\binom{n-1}{m-1}$ in the generic case, when the matroid is uniform. □

The left side of Table 7.3 indicates that as before the mth row gives the values of a polynomial of degree $m-1$. For instance, for $m=3$ we find $2n^2-8n+7$, and for $m=4$ we find $4/3n^3-10n^2+68/3n-15$. After the publication of the article [STZ20] on which this chapter is based Eur, Fife, Samper and Seynnaeve achieved in [EFSS20] to express the number of complex solutions to (7.4') as as a matroid invariant.and thereby giving a closed formula for the number of complex solutions to (7.4').

Theorem 7.7 ([EFSS20, Cor. 1.2]). *Let $\mathcal{L} \subset \mathbb{R}^n$ be a linear space of dimension m, viewed as a Gaussian covariance model of diagonal matrices. The general number of complex solutions to (7.4') is*

$$\sum_{i=1}^{m} \binom{n-i-1}{m-i} 2^{m-i}$$

This underlines how experimental numerical computations can spur new theoretical results.

7.4 Numerical Nonlinear Algebra in Action

One of our main contributions is the `Julia` package `LinearCovarianceModels.jl` for estimating linear covariance models; see (7.1). Given $\mathcal{L}$, our package computes the ML degree, and the dual ML degree. For any S, it finds all critical points and it selects those that are local maxima. The following example explains how this is done.

Example. We use the package to verify Example 7.2:

```
julia> using LinearCovarianceModels
julia> Σ = toeplitz(3)
3-dimensional LCModel:
 θ1 θ2 θ3
 θ2 θ1 θ2
 θ3 θ2 θ1
```

We compute the ML degree of the family Σ by computing all solutions for a generic instance using the monodromy method described in Section 3.5. For this it is necessary to construct a start pair for the score equations (7.4) of our MLE problem. To accomplish this, we first pick a random matrix Σ_0 in the subspace $\mathcal{L}$. We next compute K_0 by inverting Σ_0. Finally we need to find a symmetric matrix S_0 such that $K_0 S_0 K_0 - K_0 \in \mathcal{L}^{\perp}$. Note that this is a linear system of equations and hence directly solvable. In this manner, we easily find a start pair (x_0, p_0) by setting $p_0 = S_0$ and $x_0 = (\Sigma_0, K_0)$. The pair of solution and generic instance is called an *ML degree witness*.

```
julia> W = ml_degree_witness(Σ)
MLDegreeWitness:
 ○ ML degree → 3
 ○ model dimension → 3
 ○ dual → false
```

By default, the computation of the ML degree witness relies on a heuristic stopping criterion. We can numerically verify the correctness by using the trace test described in Section 3.6.2.

```
julia> verify(W)
Compute additional witnesses for completeness check...
Computed 10 additional witnesses
Compute trace using two parameter homotopies...
Norm of trace:  2.5977211651689205e-16
true
```

We now input the specific sample covariance matrix in (7.5), and we compute all critical points of this MLE problem using the ML degree witness from the previous step.

```
julia> S = [4/5 -9/5 -1/25; -9/5 79/16 25/24; -1/25 25/24 17/16];
julia> critical_points(W, S)
3-element Array{Tuple{Array{Float64,1},Float64,Symbol},1}:
 ([2.39038, -0.286009, 0.949965], -5.421751313919751, :local_maximum)
 ([2.52783, -0.215929, -1.45229], -5.346601549034418, :global_maximum)
 ([2.28596, -0.256394, 0.422321], -5.424161999175718, :saddle_point)
```

If only the global maximum is of interest then this can also be computed directly.

```
julia> mle(W, S)
3-element Array{Float64,1}:
 2.527832268219689
 -0.21592947057775033
 -1.4522862659134732
```

By default only positive definite solutions are reported. To list all critical points we run the command with an additional option.

```
julia> critical_points(W, S, only_positive_definite=false)
```

In this case, since the ML degree is 3, we are not getting more solutions.

7.5 Local Maxima versus Rational MLE

The theme of this chapter is maximum likelihood inference for linear covariance models. We developed some numerical nonlinear algebra for this problem, and we offer a software package (7.1). From the applications perspective, this is motivated by the fact that the

likelihood function is non-convex. It can have multiple local maxima. A concrete instance for 3×3 Toeplitz matrices was shown in Example 7.2.

In this section, we undertake a more systematic experimental study of local maxima. Our aim is to answer the following question: there is the theoretical possibility that $\ell(\Sigma)$ has many local maxima, but can we also observe this in practice?

To address this question, we explored a range of linear covariance models $\mathcal{L}$. For each model, we conducted the following experiment. We repeatedly generated sample covariance matrices $S \in \mathbb{S}^n_+$. This was done as follows. We first sample a matrix $X \in \mathbb{R}^{n \times n}$ by picking each entry independently from a normal distribution with mean zero and variance one. And, then we set $S := XX^T/n$. This is equivalent to sampling $nS \in \mathbb{S}^n_+$ from the standard Wishart distribution with n degrees of freedom.

	m												
	2	3	4	5	6	7	8	9	10	11	12	13	14
ML degree	7	37	135	361	753	1245	1625	1661	1323	801	347	97	15
max	2	3	3	5	5	5	5	6	7	5	4	2	1
max pd	1	2	3	3	4	4	4	4	5	5	4	2	1
multiple	0.4%	5.8	13.8	31.2	37.2	39.0	40.6	37.4	32.0	20.4	13.8	3.0	0.0
multiple pd	0.0%	4.6	11.2	22.4	25.2	31.6	33.0	34.8	29.6	19.4	13.0	3.0	0.0

Table 7.4: Experiments for generic m-dimensional linear subspaces of $\mathbb{S}^5$.

For each of the generated sample covariance matrices S, we computed the real solutions of the likelihood equations (7.4). From these, we identified the set of all local maxima in $\mathbb{S}^n$, and we extracted its subset of local maxima in the positive definite cone $\mathbb{S}^n_+$. We recorded the numbers of these local maxima. Moreover, we kept track of the fraction of instances S for which there were multiple (positive define) local maxima. In Table 7.4, we present our results for $n = 5$ and generic linear subspaces $\mathcal{L}$.

For each m between 2 and 14, we selected five generic linear subspaces $\mathcal{L}$ in the 15-dimensional space $\mathbb{S}^5$. Each linear subspace $\mathcal{L}$ was constructed by choosing a basis of positive definite matrices. The basis elements were constructed with the same sampling method as the sample covariance matrices. The ML degree of this linear covariance model is the corresponding entry in the $n = 5$ column on the left in Table 7.2. These degrees are repeated in the row named *ML degree* in Table 7.4.

For each model $\mathcal{L}$, we generated 100 sample covariance matrices S, and we solved the likelihood equations (7.4) using our software `LinearCovarianceModels.jl`. The row *max* denotes the largest number of local maxima that was observed in these 100 experiments. The row *multiple* gives the fraction of instances that resulted in two or more local maxima. These two numbers pertain to local maxima in $\mathbb{S}^5$. The rows *max pd* and *multiple pd* are the analogues restricted to the positive definite cone $\mathbb{S}^5_+$.

For an illustration, let us discuss the models of dimension $m = 7$. These equations (7.4) have 1245 complex solutions, but the number of real solutions is much smaller. Nevertheless, in two-fifths of the instances (39.0%) there were two or more local maxima in $\mathbb{S}^5$. In one-third of the instances (31.6%), the same happened in $\mathbb{S}^5_+$. The latter is the case of interest

in statistics. One instance had four local maxima in $\mathbb{S}^5_+$.

The second experiment we report concerns a combinatorially defined class of linear covariance models, namely the Brownian motion tree models in (7.2). We consider eleven combinatorial types of trees with 5 leaves. For each model, we perform the experiment described above, but we now used 500 sample covariance matrices per model. Our results are presented in Table 7.5, in the same format as in Table 7.4.

	tree number										
	1	2	3	4	5	6	7	8	9	10	11
ML degree	37	37	81	31	27	31	31	27	13	17	17
max	3	3	4	3	3	3	4	3	3	3	3
max pd	3	2	3	3	3	3	2	2	3	3	3
multiple	21.2%	22.8	24.2	15.6	23.0	21.2	21.2	15.4	13.8	16.2	12.4
multiple pd	8.2%	9.4	14.0	10.0	15.8	13.0	12.2	8.8	13.8	16.2	12.4

Table 7.5: Experiments for eleven Brownian motion tree models with 5 leaves.

The eleven trees are numbered by the order in which they appear in Table 7.6. For instance, tree 1 gives the 7-dimensional model in $\mathbb{S}^5_+$ whose covariance matrices are

$$\Sigma \;=\; \begin{bmatrix} \gamma_1 & \gamma_6 & \gamma_6 & \gamma_6 & \gamma_7 \\ \gamma_6 & \gamma_2 & \gamma_6 & \gamma_6 & \gamma_7 \\ \gamma_6 & \gamma_6 & \gamma_3 & \gamma_6 & \gamma_7 \\ \gamma_6 & \gamma_6 & \gamma_6 & \gamma_4 & \gamma_7 \\ \gamma_7 & \gamma_7 & \gamma_7 & \gamma_7 & \gamma_5 \end{bmatrix}.$$

This model has ML degree 37. Around eight percent of the instances led to multiple maxima among positive definite matrices. Up to three such maxima were observed.

The results reported in Tables 7.4 and 7.5 show that the maximal number of local maxima increases with the ML degree. But, they do not increase as fast as one would expect from the growth of the ML degree. On the other hand, the frequency of observing multiple local maxima seems to be roughly related to the ML degree

Here is an interesting observation to be made in Table 7.5. The last three trees, labeled 9, 10 and 11, are the binary trees. These have the maximum dimension $2n-2$. For these models, every local maximum in $\mathbb{S}^n$ is also in the positive definite cone $\mathbb{S}^n_+$. We also verified this for all binary trees with $n=6$ leaves. This is interesting since the positive definiteness constraint is the hardest to respect in an optimization routine. It is tempting to conjecture that this persists for all binary trees with $n \geq 7$.

There is another striking observation in Table 7.6. The dual ML degree for binary trees is always equal to one. We shall prove in Theorem 7.10 that this holds for any n. This means that the dual MLE can be expressed as a rational function in the data S. Hence there is only one local maximum, which is therefore the global maximum.

We close this section with a few remarks on the important special case when the ML

degree or the dual ML degree is equal to one. This holds if and only if each entry of the estimated matrix $\hat{\Sigma}$ or $\check{\Sigma}$ is a rational function in the $\binom{n+1}{2}$ quantities s_{ij}.

Rationality of the MLE has received a lot of attention in the case of *discrete random variables*. See [Sul18, Section 7.1] for a textbook reference. If the MLE of a discrete model is rational then its coordinates are alternating products of linear forms in the data [Sul18, Theorem 7.3.4]. This result due to Huh was refined in [DMS21, Theorem 1]. At present we have no idea what the analogue in the Gaussian case might look like.

Problem 7.8. Characterize all Gaussian models whose MLE is a rational function.

In addition to the binary trees in Theorem 7.10, statisticians are familiar with a number of situations when the dual MLE is rational. The dual MLE is the MLE of a linear concentration model with the sample covariance matrix S replaced by its inverse W. This is studied in [SU10] and in many other sources on Gaussian graphical models and exponential families. The following result paraphrases [SU10, Theorem 4.3].

Proposition 7.9. *If a linear covariance model $\mathcal{L}$ is given by zero restrictions on Σ, then the dual ML degree is equal to one if and only if the associated graph is chordal.*

It would be interesting to extend this result to other combinatorial families, such as colored covariance graph models (cf. [HL08]), including structured Toeplitz matrices.

The following example illustrates Problem 7.8 and it raises some further questions.

Example. We present a linear covariance model such that both the MLE and the dual MLE are rational functions. Fix $n \geq 2$ and let $\mathcal{L}$ be the hyperplane with equation $\sigma_{12} = 0$. By Proposition 7.9, the dual ML degree of $\mathcal{L}$ is one. The model is dual to the decomposable undirected graphical model with missing edge $\{1, 2\}$.

Following [Lau96, Sul18], we obtain the rational formula for its dual MLE:

$$\check{k}_{12} = W_{1,R}W_{R,R}^{-1}W_{R,2}, \qquad \text{and} \qquad \check{k}_{ij} = w_{ij} \quad \text{for } (i,j) \neq (1,2). \tag{7.7}$$

Here $R = \{3, \ldots, n\}$ and $W_{\bullet,\bullet}$ is our notation for submatrices of $W = (w_{ij}) = S^{-1}$.

The ML degree of the model $\mathcal{L}$ is also one. To see this, we note that $\mathcal{L}$ is the DAG model with edges $i \to j$ whenever $i < j$ unless $(i,j) = (1,2)$. By [Lau96, Section 5.4.1], the MLE of any Gaussian DAG model is rational. In our case, we find $\hat{K} = W + A$, where A is the $n \times n$ matrix that is zero apart from the upper left 2×2 block

$$A_{12,12} \;=\; \begin{bmatrix} s_{11}^{-1} & 0 \\ 0 & s_{22}^{-1} \end{bmatrix} \;-\; \frac{1}{s_{11}s_{22} - s_{12}^2} \begin{bmatrix} s_{22} & -s_{12} \\ -s_{12} & s_{11} \end{bmatrix}.$$

The entries in $\check{\Sigma} = (\check{K})^{-1}$ and $\hat{\Sigma} = (\hat{K})^{-1}$ are rational functions in the data s_{ij}. But, unlike in the discrete case of [DMS21], here the rational functions are not products of linear forms. Problem 7.8 asks for an understanding of its irreducible factors.

Example 7.5 raises many questions. First of all, can we characterize all linear spaces $\mathcal{L}$ with rational formulas for their MLE, or their dual MLE, or both of them? Second, it would be interesting to study arbitrary models $\mathcal{L}$ that are hyperplanes. Consider the entries for

$m = \binom{n+1}{2} - 1$ in Tables 7.2 and 7.4. We know from [SU10, Section 2.2] that the dual ML degree equals $n-1$. The ML degree seems to be $2^{n-1}-1$. In all cases, there seems to be only one local (and hence global) maximum. How to prove these observations? Finally, it is worthwhile to study the MLE when $\mathcal{L}^\perp$ is a generic symmetric matrix of rank r. What is the ML degree in terms of r and n?

7.6 Brownian Motion Tree Models

We now study the linear space $\mathcal{L}_T$ associated with a rooted tree T with n leaves. The equations of $\mathcal{L}_T$ are $\sigma_{ij} = \sigma_{kl}$ whenever $\mathrm{lca}(i,j) = \mathrm{lca}(k,l)$. In the literature (cf. [Fel73, SUZ20]), one assumes that the parameters θ_A in (7.2) are nonnegative. Here, we relax this hypothesis: we allow all covariance matrices in the spectrahedron $\mathcal{L}_T \cap \mathbb{S}^n_+$.

The ML degree and its dual do not depend on how the leaves of a tree are labeled but only on the tree topology. For fixed n, each tree topology is uniquely identified by the set of clades. Since the root clade $\{1,\ldots,n\}$ and the leaf-clades $\{1\}$, ...,$\{n\}$ are part of every tree, they are omitted in our notation. For example, if $n=5$ then the tree $\{\{1,2\},\{3,4\},\{3,4,5\}\}$ is the binary tree with four inner vertices corresponding to the three non-trivial clades mentioned explicitly. This tree is depicted in Figure 7.1.

We computed the ML degree and the dual ML degree of $\mathcal{L}_T$ for many trees T. In Table 7.6, we report results for five and six leaves. We notice that the dual ML degree is exactly one for all binary trees. This suggests that the dual MLE is a rational function. Our main result in this section (Theorem 7.10) says that this is indeed true.

The equations (7.6) for the dual ML degree can be written as $e_A^T(K-W)e_A = 0$ for all clades A. Here $W=(w_{ij})$ is given and $K^{-1} \in \mathcal{L}_T$ is unknown. We abbreviate

$$w_{A,B} \;=\; \sum_{i\in A}\sum_{j\in B} w_{ij} \;=\; e_A^T W e_B. \tag{7.8}$$

The same notation is used for general matrices. We present two examples with $n=4$.

Example. Consider the tree with clades $\{1,2\}$, $\{3,4\}$, shown in [SUZ20, Figure 1]. The dual MLE $\check{K}$ satisfies $\check{k}_{ii} = w_{ii}$ for $i=1,2,3,4$, and $\check{k}_{12} = w_{12}$, $\check{k}_{34} = w_{34}$, and

$$\check{k}_{ij} \;=\; w_{12,34}\frac{w_{i,12}w_{j,34}}{w_{12,12}w_{34,34}} \qquad \text{for } i\in\{1,2\},\, j\in\{3,4\}.$$

Example. The tree with clades $\{1,2\}$, $\{1,2,3\}$ has $\check{k}_{ii} = w_{ii}$, $\check{k}_{12} = w_{12}$, and

$$\check{k}_{13} \;=\; w_{12,3}\frac{w_{1,12}}{w_{12,12}},\;\; \check{k}_{14} \;=\; w_{123,4}\frac{w_{1,12}w_{12,123}}{w_{12,12}w_{123,123}},\;\; \check{k}_{23} \;=\; w_{12,3}\frac{w_{2,12}}{w_{12,12}},$$

$$\check{k}_{24} \;=\; w_{123,4}\frac{w_{2,12}w_{12,123}}{w_{12,12}w_{123,123}},\;\; \check{k}_{34} \;=\; w_{123,4}\frac{w_{123,3}}{w_{123,123}}.$$

Both examples were computed in `Mathematica` using the description of the Brownian motion tree model in terms of the inverse covariance matrix given in [SUZ20].

n	Clades	ML degree	dual ML degree
5	{1, 2, 3, 4}	37	11
5	{1, 2}	37	11
5	{1, 2, 3}	81	16
5	{1, 2}, {3, 4, 5}	31	4
5	{1, 2}, {3, 4}	27	4
5	{1, 2, 3}, {1, 2, 3, 4}	31	4
5	{1, 2}, {1, 2, 3}	31	4
5	{1, 2}, {1, 2, 3, 4}	27	4
5	**{1, 2}, {3, 4}, {1, 2, 3, 4}**	13	1
5	**{1, 2}, {3, 4}, {1, 2, 5}**	17	1
5	**{1, 2}, {1, 2, 3}, {1, 2, 3, 4}**	17	1
6	{1, 2, 3, 4, 5}	95	26
6	{1, 2}	95	26
6	{1, 2, 3, 4}	259	44
6	{1, 2, 3}	259	44
6	{1, 2, 3}, {4, 5, 6}	221	16
6	{1, 2}, {3, 4, 5, 6}	101	11
6	{1, 2, 3, 4}, {1, 2, 3, 4, 5}	101	11
6	{1, 2}, {3, 4}	81	11
6	{1, 2}, {1, 2, 3}	101	11
6	{1, 2}, {3, 4, 5}	181	16
6	{1, 2}, {1, 2, 3, 4, 5}	81	11
6	{1, 2, 3}, {1, 2, 3, 4}	221	16
6	{1, 2, 3}, {1, 2, 3, 4, 5}	181	16
6	{1, 2}, {1, 2, 3, 4}	181	16
6	{1, 2}, {3, 4}, {5, 6}	63	4
6	{1, 2}, {3, 4}, {1, 2, 3, 4}	99	4
6	{1, 2}, {1, 2, 3}, {4, 5, 6}	115	4
6	{1, 2}, {3, 4, 5}, {3, 4, 5, 6}	115	4
6	{1, 2}, {3, 4, 5}, {1, 2, 3, 4, 5}	99	4
6	{1, 2}, {3, 4}, {1, 2, 5, 6}	83	4
6	{1, 2}, {3, 4}, {1, 2, 3, 4, 5}	63	4
6	{1, 2, 3}, {1, 2, 3, 4}, {1, 2, 3, 4, 5}	115	4
6	{1, 2}, {3, 4}, {1, 2, 5}	83	4
6	{1, 2}, {1, 2, 3}, {1, 2, 3, 4}	115	4
6	{1, 2}, {1, 2, 3}, {1, 2, 3, 4, 5}	83	4
6	{1, 2}, {1, 2, 3, 4}, {1, 2, 3, 4, 5}	83	4
6	**{1, 2}, {3, 4}, {5, 6}, {1, 2, 3, 4}**	53	1
6	**{1, 2}, {3, 4}, {1, 2, 5}, {3, 4, 6}**	61	1
6	**{1, 2}, {3, 4}, {1, 2, 3, 4}, {1, 2, 3, 4, 5}**	53	1
6	**{1, 2}, {3, 4}, {1, 2, 5}, {1, 2, 5, 6}**	61	1
6	**{1, 2}, {3, 4}, {1, 2, 5}, {1, 2, 3, 4, 5}**	53	1
6	**{1, 2}, {1, 2, 3}, {1, 2, 3, 4}, {1, 2, 3, 4, 5}**	61	1

Table 7.6: ML degrees and dual ML degrees for Brownian motion tree models with five and six leaves. Binary trees are highlighted.

Recall that for $v \in V$ we write $\mathrm{de}(v)$ for the set of leaves of T that are descendants of v. The following theorem generalizes formulas in the above two examples.

Theorem 7.10. *Consider the model $\mathcal{L}_T$ given by a rooted binary tree T with n leaves. The dual MLE $\check{K} = (\check{k}_{ij})$ satisfies $\check{k}_{ii} = w_{i,i}$ for all i, and its off-diagonal entries are*

$$\check{k}_{ij} \;=\; w_{A,B} \prod_{u \to v} \frac{w_{\mathrm{de}(v),\mathrm{de}(u)}}{w_{\mathrm{de}(u),\mathrm{de}(u)}} \qquad \text{for } 1 \leq i < j \leq n. \tag{7.9}$$

Here A, B are the clades of the two children of $\mathrm{lca}(i,j)$. The product is over all edges $u \to v$ of T, except for the two edges with $u = \mathrm{lca}(i,j)$, on the path from i to j in T.

Proof. We refer for the proof to [STZ20, Theorem 7.3]. □

In our concluding example, we compare the MLE and its dual in a special case.

Example. Fix the five-leaf tree in Figure 7.1, with clades $\{1,2\}, \{3,4\}, \{3,4,5\}$. For simplicity assume that the data generating distribution has all parameters θ_A in (7.2) equal to one. For each sample size $n = 50, 200, 500, 5000$, we run 1000 iterations to obtain a simple Monte Carlo estimator of the mean squared errors as measured by $\mathbb{E}\|\hat{\Sigma} - \Sigma^*\|^2$ and $\mathbb{E}\|\check{\Sigma} - \Sigma^*\|^2$, where Σ^* is the true covariance matrix and $\|\cdot\|$ is a given matrix norm. To have a direct comparison between both estimators, we also approximate $\mathbb{E}\|\hat{\Sigma} - \check{\Sigma}\|^2$. We obtain the following numbers for the operator norm.

	50	200	500	5000
approx. $\mathbb{E}\|\hat{\Sigma} - \Sigma^*\|^2$	5.44	1.30	0.55	0.05
approx. $\mathbb{E}\|\check{\Sigma} - \Sigma^*\|^2$	5.28	1.31	0.55	0.05
approx. $\mathbb{E}\|\hat{\Sigma} - \check{\Sigma}\|^2$	0.38	0.02	0.00	0.00

We see that the two estimators have essentially the same statistical performance. On average, they lie very close to each other. The dual MLE, which is available in closed form, thus offers a very attractive alternative to the MLE. Similar results were obtained for the Frobenius norm and the ℓ_∞-norm but they are not reported here.

The estimates in the previous example were computed by evaluating the function in Theorem 7.10. The expression (7.9) is an alternating product of linear forms, reminiscent of [DMS21, Theorem 1]. However, this structure does not generalize, by Example 7.5, thus underscoring Problem 7.8.

7.7 Conclusion

In this chapter, we applied numerical nonlinear algebra to maximum likelihood estimation for Gaussian models defined by linear constraints on the covariance matrix. We examined the generic case as well as special models (e.g. Toeplitz, sparse, trees) that are of interest in statistics. We studied the maximum likelihood degree and its dual analogue, and we introduced a new software package `LinearCovarianceModels.jl` for reliably solving the

score equations. `LinearCovarianceModels.jl` is based on `HomotopyContinuation.jl`. In addition, we identified several scenarios for which the estimator is a rational function. We stress that numerical nonlinear algebra and our software `LinearCovarianceModels.jl` played an essential role in getting to this point. Namely, with computations as described in Section 7.4, we created Table 7.6. After seeing that table, we conjectured that the dual MLE for binary trees is one. This led us to find the rational formula (7.9). This is most likely just one of many instance where numerical experimentation can lead to new theoretical insights.

8 Catastrophe in Elastic Tensegrity Frameworks

This chapter is based on the article "Catastrophe in elastic tensegrity frameworks" [HT20] by Alexander Heaton and Sascha Timme. The article is currently under review. A preprint is available at `https://arxiv.org/abs/2009.13408`.

Tensegrity structures appear in nature and engineering, scaling in size from nanometers [LHT+10] to meters [TP03], used on the earth [Mot03, SdO09] and in outer space [Tib02, ZGS+12]. Since the tension in the lightweight cables provides stability [Cal78, ZO15], they can hold their shape without any locking mechanisms. This and other advantages make tensegrity highly appealing for *deployable structures* [Pel01]. They can significantly change size and shape, using several different functional configurations during their application. In this chapter, we discuss *elastic tensegrity frameworks* (Definition 8.1) made from rigid bars and elastic cables, similar to those appearing in [LWPQ17, SJM18], but also similar to the *tensegrity frameworks* defined in [CW96] which are popular in the mathematics and combinatorics literature. Instead of edge length inequalities as in [CW96], we use Hooke's law to introduce an energy function that distinguishes between bars and elastic cables. The configuration is then determined by solving a constrained optimization problem. This provides a large family of simple models that are effectively treated using the theory of elasticity and energy minimization (see Definition 8.3). We use numerical nonlinear algebra to calculate *all* equilibrium positions, in contrast to the more widely-used iterative methods (e.g. Newton-Raphson) that can only find one solution at a time, with no guarantees on finding them all. Elastic tensegrity frameworks depend on many parameters, e.g., the length of its rigid bars or the fixed position of some nodes. For a given framework, we can choose a space of control parameters Ω whose values we view as the parameters we can manipulate. A *path* $y : [0,1] \subseteq \mathbb{R} \to \Omega$ is a map from the unit interval to the space of control parameters Ω that describes how the controls $y(t)$ vary in time. We use numerical nonlinear algebra to track the changes in stable equilibrium positions of the framework as the control parameters vary. Most importantly, we are interested in a positive-dimensional semialgebraic subset $\mathcal{C}_\Omega \subseteq \Omega$ called the *catastrophe set* (Definition 8.9). This set records those values of control parameters whose crossing could result in a discontinuous jump in the location of the nearest local equilibrium since the current equilibrium can disappear after crossing $\mathcal{C}_\Omega$. This loss of equilibrium and the resulting behavior is called a *catastrophe*. The importance of this set is well-known (see [Arn86] for an overview), but we find that studying it from the algebraic perspective provides useful benefits. Therefore, the purpose of this chapter is to show how techniques from numerical nonlinear algebra can be used to compute the catastrophe set $\mathcal{C}_\Omega$. For this, we introduce an algebraic reformulation in Section 8.2 that we use to compute a

superset $\mathcal{D}_\Omega \supseteq \mathcal{C}_\Omega$ which contains the relevant information for the original problem (Section 8.1). This algebraic set $\mathcal{D}_\Omega$, the *catastrophe discriminant*, detects the merging of equilibrium solutions from a parametrized family of constrained optimization problems.

Hooke's law provides a simple model that has proven extremely effective in an enormous amount of real-world situations. Also, in the article *The Catastrophe Controversy* [Guc79] Guckenheimer writes "The application of Catastrophe - Singularity Theory to problems of elastic stability has been the greatest success of the theory thus far." Thus, catastrophe discriminants are of known importance, but they are very difficult to explicitly compute and this has limited their usefulness. With the development of efficient techniques in numerical nonlinear algebra, explicit computation of catastrophe discriminants is now within reach. Therefore, another purpose of this chapter is to explicitly describe these computations for a family of simple models (elastic tensegrity frameworks) that will be useful in many different applications.

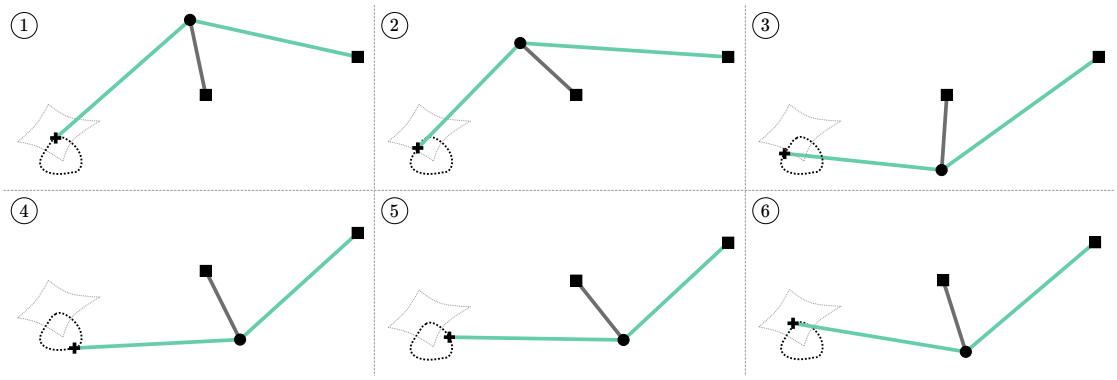

Figure 8.1: Loop crossing the catastrophe set. The black edge is a rigid bar and the green edges are elastic cables. Square nodes have fixed positions, the cross node is controlled around a loop, and the circular node's position is determined by minimizing the potential energy in the green elastic cables.

A running example, simple enough to understand yet complicated enough to illustrate the advantage of knowing $\mathcal{C}_\Omega$, is *Zeeman's catastrophe machine*. Zeeman's catastrophe machine consists of a rigid bar that can rotate freely around one of its endpoints. Attached to the non-fixed endpoint are two elastic cables. The end of one of the cables is fixed, the other can be moved freely. The machine and its behavior are depicted in Figure 8.1 at six discrete-time snapshots. For more on this example, see [PW73], where they give a parametrization of $\mathcal{C}_\Omega$ for a simplified machine and implicit equations defining $\mathcal{C}_\Omega$ for the actual machine. See also [Arn86, Section 4]. In contrast, we use sample points to encode $\mathcal{C}_\Omega$ not only for Zeeman's machine but for any elastic tensegrity framework. The basic idea of Zeeman's machine is to control the free endpoint $y(t) \in \Omega \simeq \mathbb{R}^2$ of one cable while the rotating rigid bar settles into a position of minimum energy. Using numerical nonlinear algebra, we can reliably compute all complex solutions to this constrained optimization problem and find among them the real-valued and stable local minima. In addition, we compute a pseudo-witness set for $\mathcal{D}_\Omega$ allowing effective sampling of the catastrophe set $\mathcal{C}_\Omega$, and therefore easily computable information on when catastrophes may occur, and how to avoid them entirely.

For those readers new to Zeeman's machine, consider the behavior depicted in Figure 8.1. The black bar can rotate around its base, as the green elastic cables pull on its free endpoint. As one of the cable endpoints moves smoothly, the stable equilibrium position of the machine also moves smoothly... usually. Upon crossing $\mathcal{C}_\Omega$ it can happen that this stable equilibrium disappears. This forces the machine to rapidly change shape, moving towards some new equilibrium. Without knowledge of $\mathcal{C}_\Omega$, these behaviors can be very surprising. For example, moving the control point in a small loop does not ensure a return to the original position for the machine (see Figure 8.1). Playing with this example, one quickly discovers the advantages of knowing $\mathcal{C}_\Omega$. Seemingly random catastrophes become easily predictable.

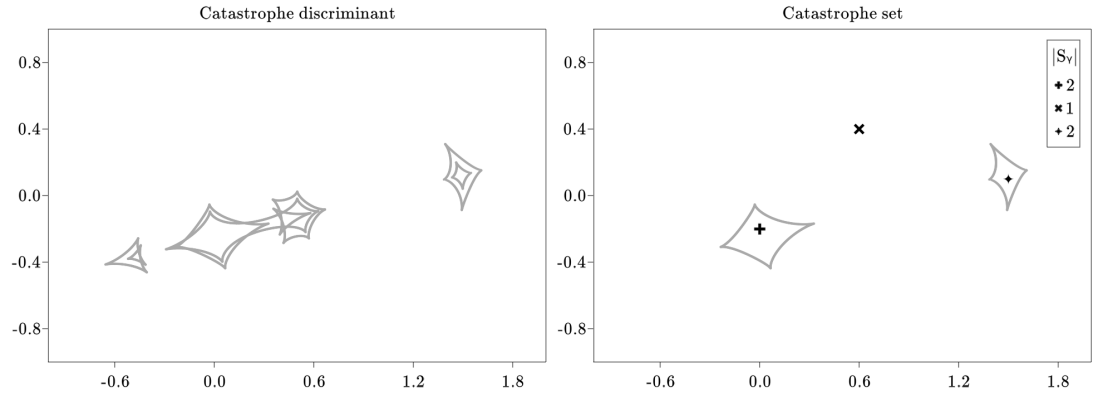

Figure 8.2: Catastrophe discriminant $\mathcal{D}_\Omega$ (left, degree 72) and catastrophe set $\mathcal{C}_\Omega$ (right) for Zeeman's machine, sampled numerically using homotopy continuation methods.

Section 8.1 gives the basic definitions for elastic tensegrity frameworks. In Section 8.2, we describe an algebraic reformulation of the relevant energy minimization problem. In doing so, we naturally arrive at the *equilibrium degree* of an elastic tensegrity framework (Definition 8.6), and the *catastrophe degree* of its catastrophe discriminant (Definition 8.8). These numbers are intrinsic to the algebraic approach and characterize the algebraic complexity of each elastic tensegrity framework for a dense set of control parameters. Though the algebraic approach naturally deals with the algebraic set $\mathcal{D}_\Omega$, the original problem deals with the semialgebraic set $\mathcal{C}_\Omega$ (Definition 8.9). For Zeeman's machine, both sets are shown in Figure 8.2. We note that $\mathcal{C}_\Omega$ in Figure 8.2 is the *envelope* of a family of curves, each of which is a *conchoid of Nicomedes* [Kle95, PW73]. Propositions 8.10 and 8.14 and Theorem 8.15 precisely relate the algebraic reformulation with the original setup. In Section 8.3, we give more details on the required computations using numerical nonlinear algebra. Finally, in Section 8.4 we demonstrate our newly developed tools on a four-bar linkage that becomes an elastic tensegrity framework upon the attachment of two elastic cables (Figure 8.5). We compute both $\mathcal{D}_\Omega$ and $\mathcal{C}_\Omega$ (Figure 8.6) and explicitly demonstrate one possible catastrophe (Figure 8.7). Code that reproduces all examples in this chapter can be found at

`https://doi.org/10.5281/zenodo.4056121`.

8.1 Elastic Tensegrity Frameworks

In this section, we formally introduce elastic tensegrity frameworks and the necessary definitions and concepts to talk about their equilibrium positions. Let $G = ([n], E)$ be a graph on $[n] := \{1, 2, \ldots, n\}$ nodes and $E = B \cup C$ edges. Edges are two-element subsets of $[n]$. Every $ij \in B$ is a rigid bar between nodes i and j and we have ℓ_{ij} as its length. Similarly, every $ij \in C$ is an elastic cable between nodes i and j that has natural resting length r_{ij} and a constant of elasticity c_{ij}. The graph G is embedded by a map $p : [n] \to \mathbb{R}^d$ and we denote the coordinates of the n nodes of G by $p_1 = (p_{11}, \ldots, p_{1d}), \ldots, p_n \in \mathbb{R}^d$ and identify the space of coordinates with $\mathbb{R}^{nd}$.

Example (Zeeman's catastrophe machine)**.** We illustrate the definitions and concepts of this and the next section on Zeeman's catastrophe machine. Zeeman's machine is an elastic tensegrity framework on $n = 4$ nodes with edges $E = \{14, 24, 34\}$ partitioned as $B = \{14\}$ and $C = \{24, 34\}$. See Figure 8.3 for an illustration.

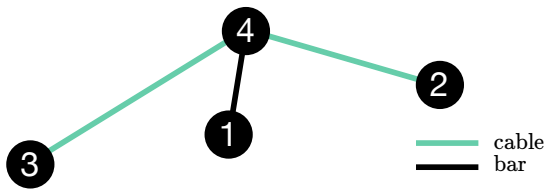

Figure 8.3: Our setup of Zeeman's catastrophe machine

For every rigid bar $ij \in B$, we define the bar constraint polynomial

$$b_{ij} := \sum_{k \in [d]} (p_{ik} - p_{jk})^2 - \ell_{ij}^2 \tag{8.1}$$

and denote by b the polynomial system whose component functions are the b_{ij} for $ij \in B$. For each elastic cable $ij \in C$, we define its potential energy q_{ij} using Hooke's law

$$q_{ij} := \frac{1}{2} c_{ij} \left(\max \left\{ 0, \sqrt{\sum_{k \in [d]} (p_{ik} - p_{jk})^2} - r_{ij} \right\} \right)^2 \qquad \text{with} \qquad Q = \sum_{ij \in C} q_{ij} \,. \tag{8.2}$$

This says that the energy q_{ij} is proportional to the square of the distance the elastic cable has been stretched past its natural resting length. Though we have only introduced rigid bars and elastic cables, one could add compressed elastic edges that want to expand according to Hooke's law. For ease of exposition, we proceed with elastic cables and rigid bars, rather than also including compressive struts in our notation.

We have introduced several variables. As shorthand we use the symbols p, ℓ, r, c to refer

to the variables

$$\begin{array}{ll} p_{ik} & \text{for } i \in [n], k \in [d] \\ \ell_{ij} & \text{for } ij \in B \\ r_{ij} & \text{for } ij \in C \\ c_{ij} & \text{for } ij \in C. \end{array} \tag{8.3}$$

In various examples, some of these variables are viewed as *control parameters* $y \in Y$ whose values we can fix or manipulate at will, while the other variables are viewed as *internal parameters* $x \in X$ whose values are determined by the controls y and the principle of energy minimization. Often we may fix several control parameters and let others vary in some subset $\Omega \subseteq Y$.

Example (Zeeman's catastrophe machine (continued))**.** We continue with Example 8.1. We choose X, Y as

$$X = \{ (p_{41}, p_{42}) \} \simeq \mathbb{R}^2$$
$$Y = \{ (p_{11}, p_{12}, p_{21}, p_{22}, p_{31}, p_{32}, \ell_{14}, r_{24}, r_{34}, c_{24}, c_{34}) \} \simeq \mathbb{R}^{11}$$

but only consider the subset $\Omega \subseteq Y$ as in

$$\Omega = \Big\{ \ (0, 0, 2, -1, p_{31}, p_{32}, 1, 1, 1, 0.5, 0.5) \ : \ (p_{31}, p_{32}) \in \mathbb{R}^2 \Big\} \subseteq Y.$$

In this setup, we have fixed everything except the coordinates of nodes 3 and 4. We *control* the $y = (p_{31}, p_{32}) \in \Omega$ and *solve for* the $x = (p_{41}, p_{42}) \in X$. This means that for a given $y = (p_{31}, p_{32}) \in \Omega$ we find the coordinates $x = (p_{41}, p_{42}) \in X$ that minimize

$$Q(p_{41}, p_{42}) = \frac{1}{4}\mathsf{max}\left\{0, \sqrt{(2 - p_{41})^2 + (-1 - p_{42})^2} - 1\right\} + \frac{1}{4}\mathsf{max}\left\{0, \sqrt{(p_{31} - p_{41})^2 + (p_{32} - p_{42})^2} - 1\right\}$$

restricted to the set $\{(x, y) : b(x, y) = 0\} \cap X \times \Omega$. In this case, since $B = \{14\}$ the constraints $b(x, y) = 0$ have only one equation $b_{14}(x, y) = 0$ that reads

$$b_{14}(x, y) = (0 - p_{41})^2 + (0 - p_{42})^2 - 1^2 = 0.$$

Definition 8.1. An *elastic tensegrity framework* is a graph on nodes $[n]$ with edges $E \mathcal{V}(\binom{[n]}{2})$ along with the energy function Q of (8.2), a partition $E = B \cup C$ of the edge set into rigid bars and elastic cables, and a partition of variables p, ℓ, r, c of (8.3) into internal and control parameters X and $\Omega \subseteq Y$. A *configuration* of an elastic tensegrity framework is a tuple $(x, y) \in X \times Y$ satisfying the bar constraints (8.1).

Remark 8.2. We note that [CW96] used the concept of an energy function as motivation for their definition of prestress stability. Their definition of a *tensegrity framework* uses inequalities on edge lengths to distinguish bars from cables and struts. Our definition puts the energy function at center stage and also allows for a space of control parameters Ω that

we need in order to define the catastrophe discriminant $\mathcal{D}_\Omega \subseteq \Omega$ below.

Definition 8.3. We describe the interaction between an elastic tensegrity framework and the energy function given in (8.2) with the following definitions.

1. Fix a tuple of control parameters $y \in Y$. An elastic tensegrity framework in configuration (x, y) is *stable* if the internal parameters $x \in X$ are a *strict local minimum* of the energy function Q restricted to the algebraic set $\{x \in X : b(x, y) = 0\}$ of internal parameters satisfying the bar constraints $b(x, y) = 0$ of (8.1).
2. For fixed controls $y \in Y$, we collect all strict local minima in the *stability set* $\mathcal{S}_y$, defined as all internal parameters $x \in X$ such that the corresponding elastic tensegrity framework (x, y) is stable.
3. The *stability correspondence* $\mathcal{SC}$ is the set of pairs $(x, y) \in X \times Y$ such that $x \in \mathcal{S}_y$. For a given subset $\Omega \subseteq Y$ of controls, we let $\mathcal{SC}_\Omega$ be all $(x, y) \in X \times \Omega \subseteq X \times Y$ such that $x \in \mathcal{S}_y$.

If we are only interested in a subset of control parameters $\Omega \subseteq Y$, these definitions apply verbatim with Ω replacing Y.

Example (Zeeman's catastrophe machine (continued))**.** We continue with Example 8.1. Figure 8.4 shows Zeeman's catastrophe machine in a stable configuration. However, for that specific value of y the stability set $\mathcal{S}_y$ contains two points, with the second configuration shown in grey. Since the constraints $b(x, y) = 0$ essentially describe a circle, we can also plot the periodic energy function in Figure 8.4. For the particular value of the controls $y \in \Omega$ we chose, there are two local minima, and hence $|\mathcal{S}_y| = 2$.

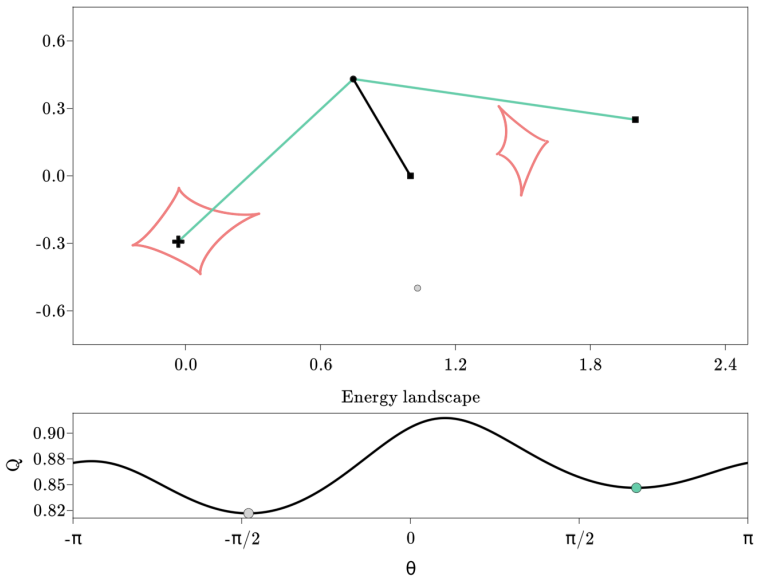

Figure 8.4: Zeeman machine in a stable configuration with $|\mathcal{S}_y| = 2$. The second stable position of node 4 is depicted in gray.

In the following, we focus on stable elastic tensegrity frameworks and the behavior when control parameters $y \in \Omega \subseteq Y$ change. For this, consider a *smooth path of control parameters*

$$\begin{aligned} y : [0,1] \subseteq \mathbb{R} &\to \Omega \subseteq Y \\ t &\mapsto y(t) \end{aligned} \tag{8.4}$$

and an *initial condition* $(x(0), y(0))$ that is stable according to Definition 8.3. We are interested in the time evolution of the internal parameters $x(t)$ determined by minimizing Q constrained by b for the given path $y(t)$ of control parameters. In particular, can we identify certain regions where small changes in $y(t)$ can cause large changes in the tensegrity framework? In Section 8.2, we solve an *algebraic reformulation* of this problem. We define the *catastrophe discriminant* $\mathcal{D}_\Omega \subseteq \Omega \subseteq Y$ and prove in Theorem 8.11 that as long as $y(t) \notin \mathcal{D}_\Omega$ we can always lift a smooth path of controls $y(t)$ to a smooth path of equilibria. We also show in Theorem 8.13 that stable local minima are preserved along this lift. Additionally, we relate the algebraic reformulation back to the original problem by showing in Theorem 8.15 the stronger statement that stable local minima are preserved when the smaller, semialgebraic set $\mathcal{C}_\Omega \subseteq \mathcal{D}_\Omega$ is avoided.

8.2 Algebraic Reformulation

In this section, we transfer questions about the stability of elastic tensegrity frameworks into an algebraic problem. The motivation is as follows. The computation of $\mathcal{S}_y$ is in general a very hard problem since the points in $\mathcal{S}_y$ are *all* local minima of a constrained optimization problem. Thus, standard optimization methods are not sufficient since they yield in each run at most one local minimum and cannot provide guarantees to find all local minima. In contrast, if we work with systems of polynomial equations we can apply tools from numerical nonlinear algebra to obtain all solutions. This is discussed in more detail in Section 8.3.

In the following, let $([n], E)$ be an elastic tensegrity framework with variables p, ℓ, r, c from (8.3) partitioned into the internal parameters $x \in X \simeq \mathbb{C}^{m_1}$ and the control parameters $y \in Y \simeq \mathbb{C}^{m_2}$. Compared to the previous section we now work over the complex numbers. Let Ω be an affine subvariety of the control parameters Y we wish to manipulate with controls $y(t) \in \Omega$. Why an affine subvariety? This allows us, among other things, to consider movement of a node constrained to motion in a sphere, perhaps determined by a rigid bar. We denote by $\Omega_\mathbb{R}$ the real part of Ω and assume that the dimension of $\Omega_\mathbb{R}$ and Ω coincide. We introduce variables δ_{ij} for $ij \in C$ to eliminate the square roots in the potential energies q_{ij}. For $ij \in E$, let

$$g_{ij} = \begin{cases} \ell_{ij}^2 - \sum_{k\in[d]}(p_{ik} - p_{jk})^2 & \text{if } ij \in B \\ \delta_{ij}^2 - \sum_{k\in[d]}(p_{ik} - p_{jk})^2 & \text{if } ij \in C \end{cases}$$

and denote by $g(x, \delta, y) : X \times \mathbb{C}^{|C|} \times Y \to \mathbb{C}^{|E|}$ the polynomial systems whose component-

wise entries are the g_{ij}. Furthermore, denote by $\mathcal{G}_y$ the zero set of g for a fixed $y \in Y$

$$\mathcal{G}_y := \{(x, \delta) \in X \times \mathbb{C}^{|C|} \mid g(x, \delta, y) = 0\}.$$

For $ij \in C$, let

$$\widetilde{q_{ij}} = \frac{1}{2} c_{ij}(\delta_{ij} - r_{ij})^2 \qquad \text{with} \qquad \widetilde{Q}_y = \sum_{ij \in C} \widetilde{q_{ij}}$$

an algebraic energy function $\widetilde{Q}_y$. The subscript emphasizes possible dependency on $y \in Y$.

To study the stability set $\mathcal{S}_y$, we look at the critical points of $\widetilde{Q}_y(x, \delta)$ subject to $(x, \delta) \in \mathcal{G}_y$. A point $(x, \delta) \in \mathcal{G}_y$ is a critical point of the energy function $\widetilde{Q}$ if the gradient $\nabla \widetilde{Q}$ is orthogonal to the tangent space of $\mathcal{G}_y$ at (x, δ). If the variety $\mathcal{G}_y$ is a complete intersection, i.e., the codimension of $\mathcal{G}_y$ equals $|E|$, then we can directly apply the technique of Lagrange multipliers to compute the critical points. In the following, we assume for ease of exposition that this is the case. However, this is not a critical assumption and the results can be extended to non-complete intersections by using standard numerical nonlinear algebra techniques for randomizing overdetermined systems (see [SW05, Chapter 13]). We introduce the variables λ_{ij} for $ij \in E$ to act as *Lagrange multipliers* and let

$$L_y(x, \delta, \lambda) = \widetilde{Q}_y + \sum_{ij \in E} \lambda_{ij} g_{ij}(x, \delta, y). \tag{8.5}$$

Definition 8.4. Define the polynomial map dL_y by letting its component functions be the various partial derivatives of L_y with respect to x, δ and λ.

$$dL_y := \frac{\partial L_y}{\partial(x, \delta, \lambda)} : X \times \mathbb{C}^{|C|} \times \mathbb{C}^{|E|} \to X \times \mathbb{C}^{|C|} \times \mathbb{C}^{|E|},\ (x, \delta, \lambda, y) \mapsto dL_y(x, \delta, \lambda).$$

Let its zero set be the affine algebraic variety denoted $\mathcal{L}_y := dL_y^{-1}(0) \subseteq X \times \mathbb{C}^{|C|} \times \mathbb{C}^{|E|}$. Similarly, we define

$$\mathcal{LC} := \{(x, \delta, \lambda, y) \,|\, (x, \delta, \lambda) \in \mathcal{L}_y\} \subseteq X \times \mathbb{C}^{|C|} \times \mathbb{C}^{|E|} \times \Omega$$

and let $\mathcal{LC}_{reg}$ denote its open, dense subset of smooth points, $\mathcal{LC}_{sing}$ its singular locus, and $\mathcal{LC}_{\mathbb{R}}$ its real part.

Proposition 8.5. *If the dimension of Ω and $\mathcal{LC}$ coincide, then for almost all $y \in \Omega$ the variety $\mathcal{L}_y$ is finite and has the same cardinality $\mathcal{N}$. For all $y \in \Omega$ the variety $\mathcal{L}_y$ contains at most $\mathcal{N}$ isolated points.*

Proof. Since the projection $\pi : \mathcal{LC} \to \Omega$ is dominant and the dimension of Ω and $\mathcal{LC}$ coincide, it follows that the fiber $\mathcal{L}_y$, $y \in \Omega$ is generically finite. The result then follows from Theorem 3.2. □

Definition 8.6. Given $\Omega \subseteq Y$, we define the *equilibrium degree of a framework* to be the cardinality of $\mathcal{L}_y$ for general $y \in Y$. Proposition 8.5 implies that the equilibrium degree is well-defined.

Example (Zeeman's catastrophe machine (continued))**.** We continue our running example with edges $E = \{14, 24, 34\}$ partitioned as $B = \{14\}$ and $C = \{24, 34\}$. Recall from Example 8.1 we had $\Omega = \{\ (0, 0, 2, -1, p_{31}, p_{32}, 1, 1, 1, 0.5, 0.5)\ :\ (p_{31}, p_{32}) \in \mathbb{R}^2\} \subseteq Y$, and $X = \{\ (p_{41}, p_{42})\ \} \simeq \mathbb{R}^2$. We write $x = (p_{41}, p_{42})$. The polynomials defining our constraints are

$$g(x, \delta, y) = \begin{bmatrix} 1^2 - (0 - p_{41})^2 - (0 - p_{42})^2 \\ \delta_{24}^2 - (2 - p_{41})^2 - (-1 - p_{42})^2 \\ \delta_{34}^2 - (p_{31} - p_{41})^2 - (p_{32} - p_{42})^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

We obtain the Lagrangian of (8.5) as

$$L_{(p_{31},p_{32})} = \frac{1}{4}(\delta_{24} - 1)^2 + \frac{1}{4}(\delta_{34} - 1)^2 + \Big(1 - {p_{41}}^2 - {p_{42}}^2\Big)\lambda_{14} + \\ \Big(\delta_{24}^2 - (2 - p_{41})^2 - (-1 - p_{42})^2\Big)\lambda_{24} + \Big(\delta_{34}^2 - (p_{31} - p_{41})^2 - (p_{32} - p_{42})^2\Big)\lambda_{34}.$$

The polynomial system dL_y is given by

$$dL_{(p_{31},p_{32})}(x, \delta, \lambda) = \begin{bmatrix} -2\,\lambda_{14}p_{41} - 2\,\lambda_{24}(2 - p_{41}) - 2\,\lambda_{34}(p_{31} - p_{41}) \\ -2\,\lambda_{14}p_{42} - 2\,\lambda_{24}(-1 - p_{42}) - 2\,\lambda_{34}(p_{32} - p_{42}) \\ \frac{1}{2}(\delta_{24} - 1) + 2\,\delta_{24}\lambda_{24} \\ \frac{1}{2}(\delta_{34} - 1) + 2\,\delta_{34}\lambda_{34} \\ 1 - {p_{41}}^2 - {p_{42}}^2 \\ \delta_{24}^2 - (2 - p_{41})^2 - (-1 - p_{42})^2 \\ \delta_{34}^2 - (p_{31} - p_{41})^2 - (p_{32} - p_{42})^2 \end{bmatrix}.$$

The equilibrium degree for this framework is 16. This means that for generic $(p_{31}, p_{32}) \in \Omega$ the equations $dL_{(p_{31},p_{32})}(x, \delta, \lambda) = 0$ have 16 isolated solutions.

We have particular interest in those parameter values $y \in \Omega$ where the number of regular isolated solutions $|\mathcal{L}_y|$ of $dL_y(x, \delta, \lambda) = 0$ is less than the equilibrium degree of the framework, since for those parameters local minima can disappear.

Definition 8.7. Define the *catastrophe discriminant* $\mathcal{D}_\Omega \subseteq \Omega$ as the Zariski closure of the set of critical values of the projection map

$$\pi : \mathcal{LC} \to \Omega, \qquad z = (x, \delta, \lambda, y) \mapsto y = \pi(z)$$

where the *critical values* are defined as those $\pi(z) \in \Omega$ such that either $z \in \mathcal{LC}_{sing}$ or there exists a tangent vector $v \in T_z\mathcal{LC}$ in the kernel of $d\pi_z$. The catastrophe discriminant is an algebraic subvariety of Ω of codimension 1.

Definition 8.8. The *catastrophe degree* of an elastic tensegrity framework is the degree of the algebraic variety $\mathcal{D}_\Omega$.

Example (Zeeman's catastrophe machine (continued))**.** We continue with Example 8.2. Refer back to Figure 8.2 which shows the catastrophe discriminant $\mathcal{D}_\Omega \subseteq \Omega$ for Zeeman's

machine with controls Ω defined in Example 8.1. Note that $\mathcal{D}_\Omega$ does not depend on $y \in \Omega$ but just on the choice of X and $\Omega \subseteq Y$ itself. Here, $\mathcal{D}_\Omega$ is an algebraic plane curve of degree 72. That is, the catastrophe degree is 72. Over the finite field $\mathbb{Z}_{65521}$ the catastrophe discriminant $\mathcal{D}_\Omega$ is the zero set of the 2701-term polynomial

$$p_{31}^{72} + 13109\, p_{31}^{71} p_{32} - 13055\, p_{31}^{70} p_{32}^2 + 10676\, p_{31}^{69} p_{32}^3 + 7407\, p_{31}^{68} p_{32}^4 + 4476\, p_{31}^{67} p_{32}^5 + \\ 31981\, p_{31}^{66} p_{32}^6 + 12338\, p_{31}^{65} p_{32}^7 - 8796\, p_{31}^{64} p_{32}^8 + \ldots - 709\, p_{31} - 32406\, p_{32} + 540.$$

Figure 8.2 also shows the catastrophe set $\mathcal{C}_\Omega$ that we define below. As we move controls $y(t) \in \Omega$, the set $\mathcal{C}_\Omega$ detects changes in the number of local minima, and hence possible catastrophe.

Definition 8.9. We define

$$\mathcal{C}_\Omega := \{ y \in \mathcal{D}_\Omega \cap \Omega_\mathbb{R} \mid \text{there exists } (x, \delta, \lambda, y) \in \pi^{-1}(y) \text{ with } \delta \geq 0 \} \subseteq \mathcal{D}_\Omega \cap \Omega_\mathbb{R}$$

to be the *catastrophe set*. This is the part of the catastrophe discriminant $\mathcal{D}_\Omega$ that relates to the original problem. We note that the *catastrophe set* $\mathcal{C}_\Omega$ partitions $\Omega_\mathbb{R}$ into cells within which the number of strict local minima is constant. Figure 8.2 depicts the number $|\mathcal{S}_y|$ of stable local minima for a typical point y in each connected component of the complement $\Omega_\mathbb{R} \setminus \mathcal{C}_\Omega$. Look ahead to Figure 8.6 for another illustration of this phenomenon for the elastic four-bar linkage discussed in Section 8.4.

Proposition 8.10. *The catastrophe set $\mathcal{C}_\Omega$ is a semialgebraic set.*

Proof. From the definition of $\mathcal{C}_\Omega$ follows that it is the projection of a semialgebraic set that by the Tarski-Seidenberg principle is again a semialgebraic set. □

We now begin to prove theorems justifying our interest in $\mathcal{D}_\Omega$ and $\mathcal{C}_\Omega$. Theorem 8.11 shows that if a smooth path of controls $y(t)$ avoids $\mathcal{D}_\Omega$, then there is always a corresponding smooth path $z(t) = (x, \delta, \lambda, y) \in \mathcal{LC}$. We will combine this with Theorem 8.13 and Proposition 8.14 to prove Theorem 8.15, which says that controls $y(t)$ avoiding the semialgebraic catastrophe set $\mathcal{C}_\Omega$ always correspond to stable local minima, and thus avoid catastrophes where local minima disappear discontinuously. This is called *catastrophe* since a real-world system would be forced to move rapidly towards the nearest remaining local minima, and since without knowledge of $\mathcal{C}_\Omega$ this sudden change in behavior would be very surprising (loss of equilibrium). For the remainder of this section, we aim to prove Theorems 8.11, 8.13, and 8.15.

Theorem 8.11. *Let $y : [0,1] \to \Omega_\mathbb{R}$ with $[0,1] \subseteq \mathbb{R}$ be a smooth path of control parameters with initial conditions $y(0) \in \Omega_\mathbb{R}$ and $z(0) \in \mathcal{LC}_{reg}$ such that $\pi(z(0)) = y(0)$ and the expected dimension $\dim(T_{z(0)}\mathcal{LC}) = \dim(T_{y(0)}\Omega)$ holds. If $y(t) \notin \mathcal{D}_\Omega$ for all t, then there exists a smooth lifting $z : [0,1] \to \mathcal{LC}$ with $\pi(z(t)) = y(t)$ for all t.*

Proof. Since $\dim(T_{z(0)}\mathcal{LC}) = \dim(T_{y(0)}\Omega)$ and since $y(0) \notin \mathcal{D}_\Omega$, we know that the differential $d\pi_{z(0)}$ is an isomorphism. By the inverse function theorem, π is a local diffeomorphism at $z(0)$. Hence there is some open neighborhood U of $y(0)$ in Ω mapped diffeomorphically to

some open neighborhood of $z(0)$ in $\mathcal{LC}$. Therefore we can define $z(t) = \pi|_U^{-1}(y(t))$ for all t such that $y(t) \in U$. Since $y(t)$ avoids $\mathcal{D}_\Omega$, we know that $\pi|_U^{-1}(y(t))$ avoids the singular locus of $\mathcal{LC}$ so that the dimension conditions $\dim(T_{z(t)}\mathcal{LC}) = \dim(T_{y(t)}\Omega)$ continue to hold for all t. Since $y(t)$ avoids $\mathcal{D}_\Omega$, we also know that $d\pi_{z(t)}$ will continue to have full rank, since none of the $\dim(T_{z(t)}\mathcal{LC})$ many tangent vectors are in the kernel. This shows we can continue defining $z(t)$ by the local diffeomorphisms that exist by the inverse function theorem. Hence there exists a lifting $z : [0, 1] \to \mathcal{LC}$ with $\pi(z(t)) = y(t)$ for all t. □

We now want to show that avoiding $\mathcal{D}_\Omega$ preserves the stability of the corresponding elastic tensegrity framework. For this, we first need a precise definition of what it means to be a local minimum of $\widetilde{Q}_y(x, \delta)$ subject to the constraint $\mathcal{G}_y$.

Definition 8.12. Let $d^2\widetilde{Q}$ and $d^2 g_{ij}$ be the Hessian matrices of $\widetilde{Q}_y$ and g_{ij} respectively, when viewed as functions of the variables x and δ. Let dg denote the Jacobian of the constraints g viewed again as functions of the variables x and δ. We say that $z = (x, \delta, \lambda, y) \in \mathcal{LC}_\mathbb{R}$ is a *strict local minimum* for the energy $\widetilde{Q}_y$ subject to constraints g if (x, δ) satisfy the sufficient condition that the projected Hessian

$$V^T \left[d^2\widetilde{Q} + \sum_{ij \in E} \lambda_{ij} d^2 g_{ij} \right] V \tag{8.6}$$

is positive definite. Here V is a real basis of the null space of dg. The term projected refers to the fact that the Hessian is projected onto the tangent space of the constraints $g = 0$ at (x, δ). See, e.g., [GMW82, page 81].

The next theorem shows that controls $y(t)$ avoiding $\mathcal{D}_\Omega$ preserve the stability of the corresponding elastic tensegrity framework.

Theorem 8.13. *Let $y(t)$ be a smooth path of control parameters with initial conditions as in Theorem 8.11. Furthermore, if the initial condition $z(0)$ is a strict local minimum according to Definition 8.12, then all the lifts $z(t)$ are strict local minima as well.*

Proof. Let V be a matrix whose columns form a basis for $\mathrm{Null}(dg)$, the tangent space of the constraint variety. Then $\mathrm{Null}(V^T) = \mathrm{Col}(dg^T)$ is the normal space. Set

$$H := \left[d^2\widetilde{Q} + \sum_{ij \in E} \lambda_{ij} d^2 g_{ij} \right].$$

As the controls $y(t)$ vary smoothly, by Theorem 8.11 so does the point $z(t) \in \mathcal{LC}_{reg}$. Therefore the eigenvalues of the symmetric matrix $V^T H V$ also vary smoothly. Since $z(0)$ began as a strict local minimum, $V^T H V$ began with all positive eigenvalues. Suppose that at some $z(t)$ there appears a zero eigenvalue of $V^T H V$. Then there is a null vector $V^T H V u = 0$. Placing parentheses $V^T(HVu) = 0$ we see that HVu is in the normal space and by construction $Vu \in$ the tangent space. But then there must exist a linear combination w

writing HVu in terms of the columns of dg^T, and hence $(Vu, -w) \in \mathrm{Null}(d^2L)$ where

$$d^2L = \begin{bmatrix} H & dg^T \\ dg & 0 \end{bmatrix}$$

is the Hessian of the Lagrangian L_y of (8.5). Note that the null vector $(Vu, -w)$ of d^2L extends to a tangent vector of $T_{z(t)}\mathcal{LC}$ by appending zeros in the Ω components. This tangent vector clearly projects to zero by $d\pi_{z(t)}$. But this means that $y(t) \in \mathcal{D}_\Omega$, completing the proof. □

We now discuss how our algebraic reformulation relates back to the original problem. In our algebraic reformulation, we removed the square roots by introducing the additional variables δ_{ij} for $ij \in C$. In the following proposition, we assume that all elastic cables are in tension since such systems are only structurally stable when self-stress is induced.

Proposition 8.14. *Consider a framework in stable configuration $(x, y) \in \mathcal{SC}$ of Definition 8.3 with $y \notin \mathcal{C}_\Omega$ that also satisfies*

$$\sqrt{\sum_{k\in[d]} (p_{ik} - p_{jk})^2} - r_{ij} > 0 \tag{8.7}$$

for every $ij \in C$ so that all elastic cables are in tension. Then there exists $\delta \in \mathbb{R}^{|C|}_{\geq 0}$ and $\lambda \in \mathbb{R}^{|E|}$ such that $(x, \delta, \lambda, y) \in \mathcal{LC}_{reg}$.

Proof. Let $V_b := \{(x, y) : b(x, y) = 0\}$ and $V_g := \{(x, \delta, y) : g(x, \delta, y) = 0\}$. Now consider the map $s : X \times \Omega \to \mathbb{R}^{|C|}$ defined by coordinate functions $s_{ij}(x, y) := \sqrt{\sum_{k\in[d]}(p_{ik} - p_{jk})^2}$. Restricting this map to V_b we have its *graph*

$$\{\, (x,\, s(x, y),\, y) \,:\, (x, y) \in V_b\} \subseteq X \times \mathbb{R}^{|C|}_{\geq 0} \times \Omega$$

which provides a local diffeomorphism between V_b and V_g near any point $(x, y) \in V_b$ satisfying (8.7). Observe that, by construction, $\widetilde{Q}_y$ takes values on the image points equal to the values taken by Q on the domain V_b, provided condition (8.7) holds. Therefore, if $(x, y) \in V_b$ is a strict local minimum of Q on V_b, then $(x, s(x, y), y) \in V_g$ is a strict local minimum of $\widetilde{Q}_y$ on V_g. Also, since we assume $y \notin \mathcal{C}_\Omega$ we conclude that $(x, s(x, y), y) \in V_g$ is a non-singular point of V_g. Hence, by the Lagrange multipliers condition for local extrema we know that there exists λ such that $(x, s(x, y), \lambda, y) \in \mathcal{LC}_{reg}$, concluding the proof. □

Finally, we are able to prove that the stability of the corresponding elastic tensegrity framework is preserved by avoiding only $\mathcal{C}_\Omega \subseteq \Omega$.

Theorem 8.15. *Let $y : [0, 1] \to \Omega$ with $[0, 1] \subseteq \mathbb{R}$ be a smooth path of control parameters with initial conditions $y(0) \in \Omega$ and $(x(0), y(0)) \in \mathcal{SC}$ satisfying the conditions of Proposition 8.14. If $y(t) \notin \mathcal{C}_\Omega$ and condition (8.7) remains satisfied for all $t \in [0, 1]$, then there exists a smooth lifting $z : [0, 1] \to \mathcal{LC}$ with $z(t) = (x(t), \delta(t), \lambda(t), y(t))$ for all t such that $(x(t), y(t)) \in \mathcal{SC}$.*

Proof. By Proposition 8.14, if we have $(x(0), y(0)) \in \mathcal{SC}$ satisfying (8.7), then there exist δ, λ which we call $\delta(0), \lambda(0)$ such that $(x(0), \delta(0), \lambda(0), y(0)) \in \mathcal{LC}_{reg}$. But then, by Theorems 8.11 and 8.13 we have lifts $z(t) = (x(t), \delta(t), \lambda(t), y(t))$ satisfying Definition 8.12 as strict local minima for $\widetilde{Q}_y$ on V_g. Using the graphs of the maps $s(x(t), y(t))$ as in the proof of Proposition 8.14, we have local diffeomorphisms for every t such that strict local minima of Q on V_b are mapped to strict local minima of $\widetilde{Q}_y$ on V_g, since (8.7) is satisfied for all t. But then each $(x(t), y(t)) \in \mathcal{SC}$ as required. □

8.3 Computations Using Numerical Nonlinear Algebra

In this section, we use the algebraic reformulation developed in the previous section to describe numerical nonlinear algebra routines that can be used to answer the following three computational problems:

1. Given $\gamma \in \Omega_{\mathbb{R}}$, compute $\mathcal{S}_\gamma$.
2. Given an algebraic path $y(t) : [0, 1] \to \Omega_{\mathbb{R}} \subseteq Y$ and an initial configuration $x_0 \in \mathcal{S}_{y(0)}$, compute the path points $\gamma \subseteq y([0, 1])$ where a catastrophe might occur (a local minimum disappears).
3. Given a control set Ω, compute the catastrophe set $\mathcal{C}_\Omega$.

We start with the first question. Recall that dL_y is a polynomial system. We can compute all isolated solutions of a polynomial system using the methods described in Chapter 3. For our computations, we use `HomotopyContinuation.jl`. To compute $\mathcal{S}_\gamma$ for a given $\gamma \in \Omega_{\mathbb{R}}$, we first solve $dL_\gamma(x, \delta, \lambda) = 0$. This results in finitely many complex solutions $\mathcal{L}_\gamma$. Of these complex solutions, we then select those solutions whose components are real-valued and then further select those real-valued solutions where the projected Hessian defined in (8.6) is positive definite. Note that computing solutions to $dL_\gamma(x, \delta, \lambda) = 0$ usually requires that we track many more paths than the equilibrium degree of $\mathcal{L}_\gamma$. If the goal is to compute $\mathcal{S}_y$ for many different $y \in \Omega_{\mathbb{R}}$, it is advantageous to use a parameter homotopy as outlined in Section 3.3. There, the idea is to first compute $\mathcal{L}_{y_0}$ for a general (complex) $y_0 \in \Omega$ and then to use the parameter homotopy $H(z, t) = dL_{ty_0+(1-t)y}(z)$ to efficiently compute $\mathcal{L}_y$. The parameter homotopy approach allows us to track only the minimal number of paths, equilibrium degree of $\mathcal{L}_\gamma$ many, to still guarantee that $\mathcal{S}_y$ is computed correctly.

Consider the second question where we have an algebraic path $y(t) : [0, 1] \to \Omega_{\mathbb{R}} \subseteq Y$ and an initial configuration $x_0 \in \mathcal{S}_{y(0)}$. We want to compute the path points $\gamma \subseteq y([0, 1])$ where a catastrophe might occur. From the results in Section 8.2, it follows that we want to compute the intersection of $\mathcal{C}_\Omega$ and $y([0, 1])$. For this, we first compute the intersection $\mathcal{D}_\Omega \cap \alpha$ where $\alpha \subseteq \Omega$ is an algebraic curve containing $y([0, 1])$. For simplicity, we assume that we have the general situation that $\alpha \not\subseteq \mathcal{D}_\Omega$. The catastrophe discriminant $\mathcal{D}_\Omega$ is given by $\pi(H_\Omega^{-1}(0)))$ with π from Definition 8.7 and

$$H_\Omega(x, \delta, \lambda, y) = \begin{bmatrix} dL_y(x, \delta, \lambda) \\ \det\, d^2L_y(x, \delta, \lambda) \end{bmatrix}.$$

Since the evaluation of a determinant is numerically unstable, it is better to instead use the formulation that there exists a $v \in \mathbb{P}^n$ such that $d^2L_y(x, \delta, \lambda) \cdot v = 0$. Consider the collection $\{H_\Omega, \pi, \alpha, \mathcal{W}\}$ where H_Ω and π are the polynomial maps defined above and the set $\mathcal{W} = \pi^{-1}(\alpha) \cap H_\Omega^{-1}(0)$ contains finitely many solution points. In the case that α is a line, this is a pseudo-witness set as described in Section 3.7. This allows us to perform computations on $\mathcal{D}_\Omega$ without knowing its defining polynomials explicitly. Since $\mathcal{W}$ is the zero set of a polynomial system, it can again be computed by using homotopy continuation techniques. If α is a line, then the cardinality of $\mathcal{W}$ is the catastrophe degree of the tensegrity framework. To compute $\mathcal{C}_\Omega \cap y([0, 1])$ given $\mathcal{W}$, we have to select from $(x, \delta, \lambda, \gamma) \in \mathcal{W}$ all those solutions that have real-valued coordinates, $\delta > 0$, and $\gamma \in y([0, 1]) \subseteq \alpha$.

We move to the third question and discuss the computation of the catastrophe set $\mathcal{C}_\Omega$. This is more involved since $\mathcal{C}_\Omega$ is a positive-dimensional set and we have to decide what "compute" means in our context. Since $\mathcal{C}_\Omega$ is a semialgebraic set, it can theoretically be described by a finite list of polynomial equations and inequalities. However, computing the describing polynomials is a very challenging computational problem since it requires extensive Gröbner bases computations. We were able to compute the polynomial defining $\mathcal{D}_\Omega$ in Example 8.2, but only over a finite field, and larger examples will likely fail to terminate. Instead, we opt to obtain a sufficiently dense point sample of $\mathcal{C}_\Omega$. The idea is to apply the previously described technique to compute repeatedly the intersection of $\mathcal{C}_\Omega$ and a real line $\ell \subseteq \Omega_\mathbb{R}$. To proceed, we first compute a pseudo-witness set $\{H_\Omega, \pi, \pi^{-1}(\ell_0), \mathcal{W}_0\}$ for a general (complex) line $\ell_0 \subseteq \Omega$ and then we can compute the pseudo witness set $\{H_\Omega, \pi, \pi^{-1}(\ell), \mathcal{W}\}$ by utilizing a parameter homotopy. As discussed above, this is much more efficient for the repeated solution of our equations. Note that even if the real lines $\ell \subseteq \Omega_\mathbb{R}$ are sampled uniformly, this does not guarantee that the obtained sample points converge to a uniform sample of $\mathcal{C}_\Omega$. If uniform sampling is of interest, the procedure can be augmented with a rejection step as described in [BM20]. Figure 8.2 depicts the point samples obtained for Zeeman's catastrophe machine using this method, while Figure 8.6 depicts those obtained for the elastic four-bar framework of Section 8.4.

8.4 Example: Elastic Four-Bar Framework

We demonstrate the developed techniques in another example. For this, we consider a planar four-bar linkage that is constructed from four bars connected in a loop by four rotating joints where one link of the chain is fixed. The resulting mechanism has one degree of freedom. Four-bar linkages are extensively studied in mechanics as well as numerical algebraic geometry [WS11]. Here, we extend a four-bar linkage to an elastic tensegrity framework by introducing two nodes that are attached to the two non-fixed joints by elastic cables. Formally, we introduces six nodes with coordinates $p_1, \ldots, p_6 \in \mathbb{R}^2$, bars $B = \{12, 23, 34, 41\}$ and elastic cables $C = \{35, 46\}$. See Figure 8.5 for an illustration of this basic setup.

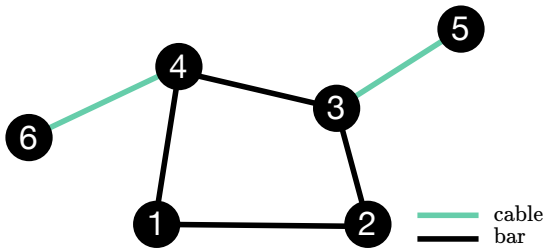

Figure 8.5: Setup of a four-bar elastic tensegrity framework.

The zero set of the bar constraints b_{ij}, $ij \in B$, is a curve of degree 6 that can be parameterized by the plane curve traced out by the motion of the midpoint $(p_3 + p_4)/2$. In kinematics terminology, the midpoint is a *coupler point* and the plane curve is called the *coupler curve* of the *mechanism*.

The idea is to fix nodes 1, 2 and 5, and to control node 6. For our model, this means choosing $X = \{(p_{31}, p_{32}, p_{41}, p_{42})\} \simeq \mathbb{R}^4$ as internal parameters and $\Omega = \{(p_{61}, p_{62})\} \simeq \mathbb{R}^2$ as control parameters. Furthermore, we fix nodes $p_1 = (-1, 0)$, $p_2 = (1, 0)$, $p_5 = (4, 3)$, bar lengths $l_{23} = 3$, $l_{34} = 1$, $l_{14} = 1.5$, resting lengths $r_{35} = r_{46} = 0.1$ and elasticities $c_{35} = 1$, $c_{46} = 2$.

In this setup, the framework has an equilibrium degree of 64. The resulting catastrophe discriminant $\mathcal{D}_\Omega$ is a curve of degree 288. $\mathcal{D}_\Omega$ and the catastrophe set $\mathcal{C}_\Omega$ are depicted in Figure 8.6. The typical sizes of the stability set $\mathcal{S}_\gamma$, $\gamma \in \Omega$, are 2, 3, and 4.

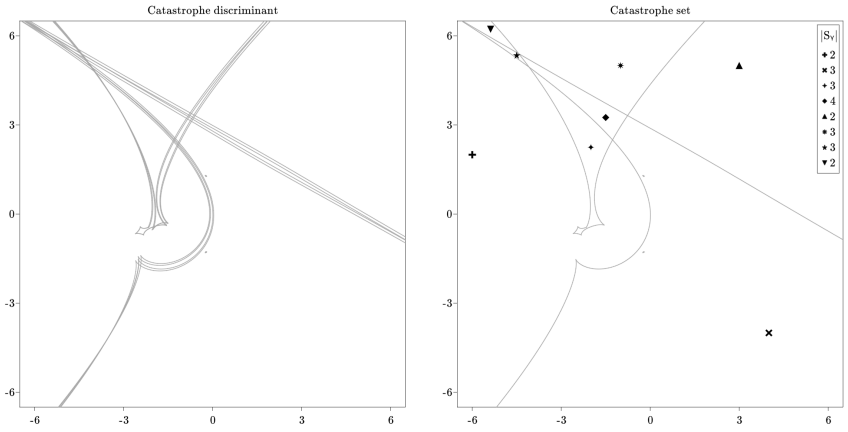

Figure 8.6: The catastrophe discriminant (left) and catastrophe set (right) of the elastic four bar framework. The cardinality of the stability set for points in each chamber of the complement of the catastrophe set is shown in the upper right corner.

Finally, we also want to give in Figure 8.7 another concrete example of a catastrophe. There, the control node 5 is depicted by a cross and it is dragged in a straight line between its position in the left figure and its position in the right figure. When the control node crosses the catastrophe set $\mathcal{C}_\Omega$, its previously stable position disappears from $\mathcal{S}_y$, and the

framework "jumps" to a new position. Again, without knowledge of $\mathcal{C}_\Omega$ these catastrophes are extremely surprising. With knowledge of $\mathcal{C}_\Omega$ and Theorem 8.15 they become predictable.

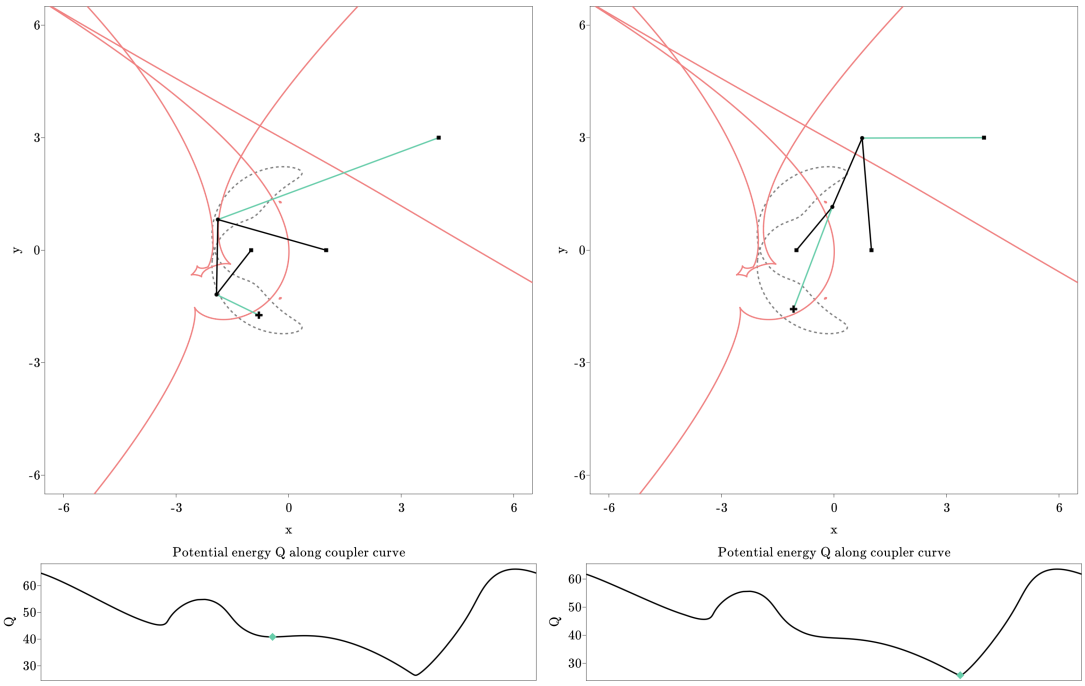

Figure 8.7: Left: The elastic four bar framework in a stable configuration. Right: Configuration of the framework after crossing the catastrophe set. The gray dashed line is the coupler curve of the four-bar linkage traced out by the coupler point defined as the midpoint of the bar connecting nodes 2 and 3. The coupler curve allows parameterizing all possible four bar positions. The catastrophe set $\mathcal{C}_\Omega$ is depicted in red. At the bottom are the energy landscapes along the coupler curve with the current position depicted in green.

8.5 Conclusion and Future Work

This chapter described *elastic tensegrity frameworks* as a large family of simple models based on Hooke's law and energy minimization. For this family, we showed how to explicitly calculate and track all stable equilibrium positions of a given framework. More importantly, we showed how to calculate the catastrophe set $\mathcal{C}_\Omega$ by using pseudo-witness sets to encode a superset $\mathcal{D}_\Omega \supseteq \mathcal{C}_\Omega$. To do this, we reformulated the problem algebraically to take advantage of tools in numerical nonlinear algebra. Knowing the catastrophe set provides extremely useful information since Theorem 8.15 shows that paths of control parameters avoiding $\mathcal{C}_\Omega$ will also avoid discontinuous loss of equilibrium, and hence avoid surprising large-scale shape changes.

In our two illustrative examples, we chose the controls Ω as a two-dimensional space overlaid with the configuration itself. These choices were made to demonstrate the ideas. However, the calculation and tracking of all stable local minima by parameter homotopy and the encoding of $\mathcal{D}_\Omega \supseteq \mathcal{C}_\Omega$ by pseudo-witness sets applies much more generally. The

control set Ω can be chosen in any way, and all the same methods apply, even if there are no easy visualizations for the controls desired. Therefore, for more complicated sets of control parameters Ω it is of interest to develop more efficient local sampling techniques based on Monte Carlo methods [ZHCG18], perhaps only sampling $\mathcal{C}_\Omega$ locally near the initial configuration $(x(0), y(0))$ or locally near the intended path $y([0,1])$. For example, it may be enough to know only the points of $\mathcal{C}_\Omega$ nearest to a given initial or current position $y(t)$.

We would also like to mention recent work [BHM$^+$20] that details a sampling scheme whose goal is to learn the real discriminant of a parametrized polynomial system, as well as the number of real solutions on each connected component. They combine homotopy continuation methods with k-nearest neighbors and deep learning techniques. For elastic tensegrity frameworks, these techniques might be used to learn $\mathcal{D}_\Omega \cap \Omega_\mathbb{R}$.

Finally, we discuss the potential of our results for use in *mechanobiology* [IWS14], where scientists have frequently and successfully used tensegrity to model cell mechanics. Even small and simple elastic tensegrity frameworks (e.g. with 6 or 12 rigid bars, plus more cables) have been used to explain and predict experimental results observed in actual cells and living tissue [DSLB$^+$11, VVB00, CS03, WTNC$^+$02]. However, the tensegrity paradigm is not universally accepted in mechanobiology, in part because it is viewed as a static theory, unable to explain dynamic, time-dependent phenomena [IWS14, see pages 13-16]. It is here where catastrophe sets could play a role. We believe *qualitative* phenomena observed in actual experiments could be predicted or explained by elastic tensegrity frameworks. Knowing the catastrophe set for a simple tensegrity model with a biology-informed choice of Ω would give catastrophe predictions that could then be tested experimentally.

9 HomotopyContinuation.jl

In this chapter, we present the software package `HomotopyContinuation.jl`[1] for the numerical solution of polynomial systems using methods from numerical nonlinear algebra. The package is written in the programming language Julia [BEKS17]. It has been developed by the author and Paul Breiding since late 2017. The author is responsible for the majority of the development and design of the software. In addition to the software, we maintain the website `JuliaHomotopyContinuation.org` containing technical documentation, tutorials, and a wide range of examples. An early version of `HomotopyContinuation.jl` (abbreviated as `HC` in the following) was described in the article [BT18].

Figure 9.1: The logo of `HomotopyContinuation.jl`.

There are several actively maintained software packages implementing polynomial homotopy continuation methods besides `HC`: `Bertini` [BHSW], `HOM4-PS-2/3` [LLT08, CLL14], `NAG4M2` [Ley11] and `PHCpack` [Ver99]. All packages have different capabilities, user interfaces and made (implicitly or explicitly) different design decisions. This results in packages with different strengths and weaknesses. Here, we describe the choices made by `HC`.

The primary goal in the development of `HC` has been to create a software package that allows engineers, researchers, and scientists to solve hard problems in nonlinear algebra without the need to first become an expert in numerical nonlinear algebra. We are positive that we achieved this goal due to the wide range of research results already obtained using `HC`. The impact of `HC` is discussed in more detail in Section 9.3. The quick adoption of `HC` is the result of its ease of use combined with the support for the various solving strategies described in Chapter 3. The functionality of `HC` is demonstrated in Section 9.1 and we describe various noteworthy design and implementation details of `HC` in Section 9.2.

9.1 Functionality

In this section, we demonstrate the functionality of `HC` as of version 2.3.3 released in January 2021. We structure this section by the typical steps necessary to solve a problem. For this,

[1] The package is free and open source. The source code is maintained at `https://github.com/JuliaHomotopyContinuation/HomotopyContinuation.jl` and permanently archived at `https://doi.org/10.5281/zenodo.4371499`.

we first have to take the mathematical formulation of our problem and represent it in a way understandable to the software. Then, we can compute all isolated solutions of our problem with one of the techniques described in Chapter 3. Finally, we certify all regular isolated solutions to our problem using the method described in Chapter 5.

To illustrate this workflow, we consider the *Euclidean distance degree* problem [DHO+16] for a plane curve $X = \mathcal{V}(f) \subseteq \mathbb{C}^2$: Given a point $u \in \mathbb{R}^2$, what is the point on X with minimal Euclidean distance to u? To answer this, we compute all critical points of the squared distance function $d_u : X \to \mathbb{R}, x = (x_1, x_2) \mapsto (x_1 - u_1)^2 + (x_2 - u_2)^2$. The number of critical points for a general point $u \in \mathbb{R}^2$ is the Euclidean distance degree of X. A point x^* is a critical point of d_u if and only if $x^* - u$ is in the normal space of X at x^*. Thus, the critical points of d_u satisfy the equations

$$\begin{aligned} (x - u) - \lambda \nabla f(x) &= 0 \\ f(x) &= 0 \end{aligned} \tag{9.1}$$

where λ is an additional variable.

In the following, we solve the Euclidean distance degree problem for the plane curve given by the zero set of $f = (x_1^4 + x_2^4 - 1)(x_1^2 + x_2^2 - 2) + x_1^5 x_2$ and the point $u_0 = (-0.32, -0.1)$. The zero set of f and the point u_0 are depicted in Figure 9.2.

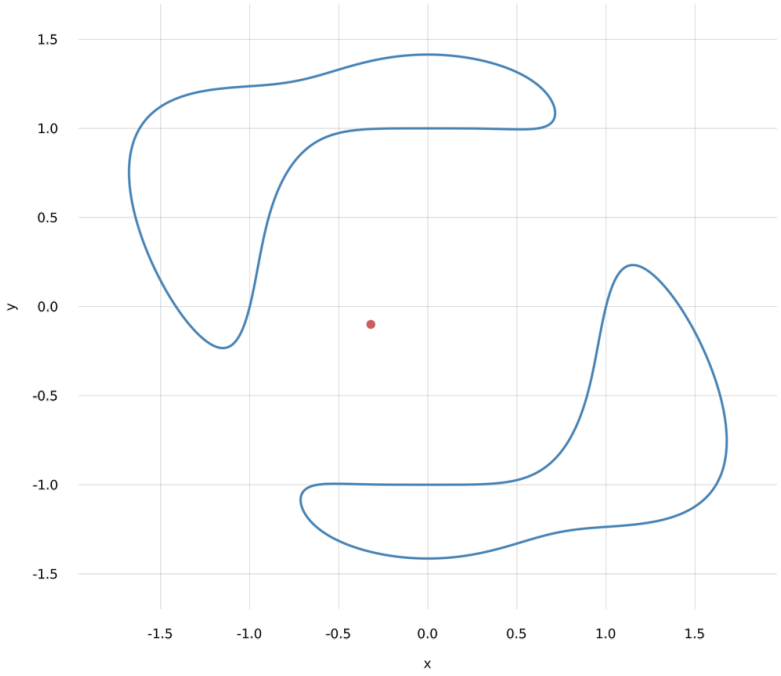

Figure 9.2: The zero set of f in blue and the point u_0 in red.

9.1.1 Problem formulation

`HC` provides an embedded domain-specific language called `ModelKit`. We discuss `ModelKit` in more detail in Section 9.2.1. `ModelKit` allows us to formulate problem (9.1) in a very

convenient way. Here is the necessary code.

```
1 using HomotopyContinuation
2 @var x[1:2] λ u[1:2]
3 f = (x[1]^4 + x[2]^4 - 1) * (x[1]^2 + x[2]^2 - 2) + x[1]^5 * x[2]
4 eqs = [(x - u) - λ * differentiate(f, x); f]
5 system = System(eqs, variables = [x;λ], parameters = u)
```

The first line loads the package. In line 2, the variables $x = (x_1, x_2), \lambda$, and $u = (u_1, u_2)$ are declared. The compact syntax `x[1:2]` allows us to easily construct a vector of variables. In line 3, we define our polynomial f. In line 4, we define our equations in the same way as (9.1). Finally, in line 5, we construct from the equations a proper system that we call `system`. In the system construction, we declare which variables we consider as parameters, and we also fix a variable order. The description of the constructed system is shown below.

```
System of length 3
 3 variables: x₁, x₂, λ
 2 parameters: u₁, u₂

 -u₁ + x₁ - λ*(5*x₂*x₁^4 + 2*x₁*(-1 + x₂^4 + x₁^4) + 4*x₁^3*(-2 + x₂^2 + x₁^2))
 -u₂ + x₂ - λ*(2*x₂*(-1 + x₂^4 + x₁^4) + 4*x₂^3*(-2 + x₂^2 + x₁^2) + x₁^5)
 x₂*x₁^5 + (-1 + x₂^4 + x₁^4)*(-2 + x₂^2 + x₁^2)
```

The description reveals one of the distinct properties of `ModelKit`. The polynomials are *not* represented in a monomial basis. Instead, the original symbolic definition of f is preserved and the differentiation of f was performed by directly applying the chain rule without expanding any expressions. For many applied problems, this approach gives us a much more efficient and numerically stable evaluation of the system. Before any numerical computations are performed, the symbolic expression is transformed into a straight-line program. We discuss this transformation in Section 9.2. If desired, a representation in the monomial basis is also possible by using the `expand` function on `system`.

Similarly, it is also possible to declare a custom homotopy. This expects, in addition to polynomials and variables, a dedicated continuation parameter. Moreover, the expression passed to `System` can also represent a system of rational functions. Not all features of `HC` work with rational functions but, e.g, performing a parameter homotopy and solving via the monodromy method works.

9.1.2 Solving

The central function in `HC` is `solve`. To illustrate it, we solve problem (9.1) for the parameter value $u_0 = (-0.32, -0.1)$.

```
u₀ = [-0.32, -0.1]
@time result = solve(sys; target_parameters = u₀)
```

```
 0.009087 seconds (43.40 k allocations: 1.727 MiB)
Result with 36 solutions
========================
* 36 paths tracked
* 36 non-singular solutions (8 real)
* random_seed: 0x21c2b5ca
* start_system: :polyhedral
```

Here, `HC` used a polyhedral homotopy, described in Section 3.4.2, with 36 paths to track. We used the `@time` macro to report the computation time. We find 36 non-singular solutions with 8 determined as real (based on a heuristic). See Figure 9.3 for a visualization of the 8 real solutions.

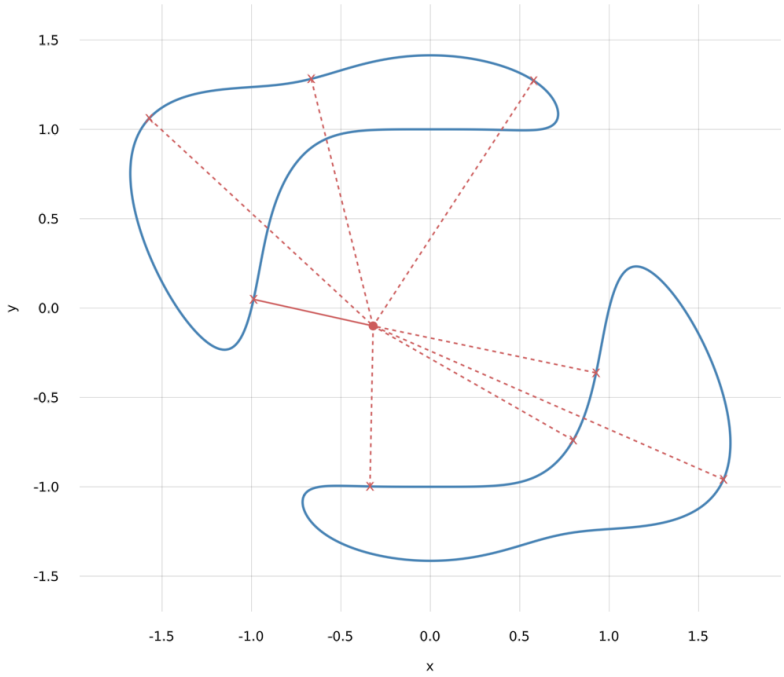

Figure 9.3: The zero set of f in blue and the point u_0 in red as well as all 8 real critical points of the Euclidean distance function.

It is also possible to solve the problem with a total degree homotopy as described in Section 3.4.1.

```
@time solve(sys; target_parameters = u₀, start_system = :total_degree)
```

```
 0.018335 seconds (281.34 k allocations: 7.178 MiB)
Result with 36 solutions
========================
* 216 paths tracked
* 36 non-singular solutions (8 real)
* random_seed: 0x858a8573
* start_system: :total_degree
```

We obtain the same 36 solutions but this time it was necessary to track 216 paths. In general, `solve` either performs a parameter homotopy or uses one of the general start systems described in Section 3.4. The behavior depends on the arguments with which it is called. All solving methods are automatically parallelized by using multiple threads.

In the previous two cases, we solved the problem directly for our parameter value u_0. If the problem should be solved repeatedly for different parameter values $u \in \mathbb{C}^2$, it is more efficient

to perform for each parameter value a parameter homotopy starting from a general $v \in \mathbb{C}^2$. For this, we have to compute the 36 solutions corresponding to a general parameter value. We use the monodromy method implementation described in Section 3.5 for this.

```
monodromy_result = monodromy_solve(sys)
```

```
MonodromyResult
===============
* return_code: :heuristic_stop
* 36 solutions
* 432 tracked loops
* random_seed: 0x4e3c389d
```

We also find 36 solutions, but this time for an automatically chosen general parameter value $v \in \mathbb{C}^2$. `HC` automatically computed a start pair $(y, v) \in \mathbb{C}^3 \times \mathbb{C}^2$ by performing the random search procedure described in Algorithm 3.10 in Section 3.5. Since we did not provide the correct number of solutions, the computation stopped based on a heuristic. In this case, the computation stopped after no new solutions were found for k loops where the default value is $k = 5$. The result shows that we needed to track 432 times around the generated loops until the computation stopped. This makes the computation, in this case, substantially more expensive than the previous methods. But for larger problems, the monodromy method is often the better choice.

To verify that the monodromy method didn't stop too early, we can use the trace test described in Section 3.6.2 by calling the function `verify_solution_completeness`.

```
verify_solution_completeness(system, monodromy_result)
```

```
[ Info: Compute additional witnesses for completeness check...
| Info: MonodromyResult
| * return_code: :heuristic_stop
| * 36 solutions
| * 288 tracked loops
| * random_seed: 0x3a0a7ce1
[ Info: Computed 36 additional witnesses
[ Info: Compute trace using two parameter homotopies...
[ Info: Norm of trace: 1.2555979342223942e-17
true
```

To obtain the solutions of (9.1) for the parameter value $u = u_0$, we perform a parameter homotopy using the solutions from the previous monodromy computation.

```
@time result = solve(system, solutions(monodromy_result);
 start_parameters = parameters(monodromy_result);
 target_parameters = u₀)
```

```
 0.002728 seconds (10.52 k allocations: 482.289 KiB)
Result with 36 solutions
========================
* 36 paths tracked
* 36 non-singular solutions (8 real)
* random_seed: 0x857740a9
```

Although for the polyhedral and the parameter homotopy approach the number of paths tracked is the same, the parameter homotopy is around three times faster since we do not need to solve a start system.

9.1.3 Certification

Our previous computations consistently found 36 solutions for the system (9.1) with the parameter value $u = u_0$ and classified eight of them as real solutions. By using the certification technique developed in Chapter 5 we can make this rigorous.

```
@time certificate = certify(system, result, target_parameters = u₀)
```

```
 0.003479 seconds (11.09 k allocations: 609.820 KiB)
CertificationResult
===================
* 36 solution candidates given
* 36 certified solution intervals (8 real, 28 complex)
* 36 distinct certified solution intervals (8 real, 28 complex)
```

We obtain a rigorous certificate that the system (9.1) has for u_0 at least 36 distinct solutions and that 8 of them are real. Recall from Section 3.4.2 that the mixed volume of a polynomial system is an upper bound for the number of its solutions. Since the mixed volume of the system (9.1) is 36, we have a hard mathematical proof that the system has 36 solutions over the complex numbers and 8 over the real numbers.

Additionally, we can use the obtained strong interval approximate zeros (Definition 5.8) corresponding to the real solutions to show that there is a unique closest point to u_0 on our plane curve. For this, we compute for each real solution interval the Euclidean distance to the point u_0 using interval arithmetic.

```
real_certificates = filter(is_real, certificates(certificate))
S = map(c -> real.(certified_solution_interval(c)[1:2]), real_certificates)
distances = map(s -> sqrt(sum((u₀ .- s).^2)), S)
```

```
8-element Array{Arblib.Arb,1}:
 [0.684877553854 +/- 4.05e-13]
 [0.898148085636 +/- 1.88e-13]
 [1.28842142248 +/- 4.43e-12]
 [2.13910089335 +/- 6.79e-12]
 [1.274017312810 +/- 7.05e-13]
 [1.63992496605 +/- 4.06e-12]
 [1.42513399923 +/- 3.12e-12]
 [1.70785729922 +/- 4.23e-12]
```

From the computed distances, we see that the unique closest point is guaranteed to be contained in the first certified real solution interval.

```
S[1]
```

```
2-element Array{Arblib.Arb,1}:
 [-0.988531743089 +/- 5.93e-13]
 [0.048736586807 +/- 6.33e-13]
```

This result also matches our intuition from looking at Figure 9.3.

9.1.4 Witness Sets

So far, we only computed the isolated zeros of a polynomial system. In the case that the solution set is positive-dimensional, we use a *witness set* to encode it. Recall the definition of a witness from Section 3.7.

To illustrate witness sets in HC, we consider the twisted cubic, a curve $C \subset \mathbb{P}^3$, described by the following system.

```
@var x y z w
twisted_cubic = System([x * z - y^2, y*w - z^2, x*w - y * z])
```

```
System of length 3
 4 variables: w, x, y, z

 x*z - y^2
 w*y - z^2
 w*x - y*z
```

To compute a witness set for C, we need to intersect it with a general linear space of codimension one and compute all isolated solutions of this intersection. This can be accomplished using the `witness_set` function. It is necessary to provide the function with information about the dimension of the component for which we want to compute a witness set.

```
W = witness_set(twisted_cubic; dim = 1)
```

```
Witness set for dimension 1 of degree 3
```

The result is a witness for C containing 3 isolated points since the degree of C is 3. To test whether the witness set is complete, we can run a trace test as described in Section 3.6.2.

```
trace_test(W)
```

```
1.1079535712329777e-16
```

Since the result is almost zero, we are certain that we have a complete witness set.

Recall that a witness set $W = (F, L, S)$ is a triple where F is a polynomial system, L a linear space and S a (sub-)set of $\mathcal{V}(F) \cap L$. For many algorithms, it is necessary to construct from a given witness set $W = (F, L, S)$ a new witness set $W' = (F, L', S')$ where S' is the result of tracking the solutions S from $\mathcal{V}(F) \cap L$ to $\mathcal{V}(F) \cap L'$. To demonstrate this in HC, we first sample a new random linear space and then compute a new witness set.

```
L' = rand_subspace([x,y,z,w];
 # we need a linear space of codimension 1
 codim = 1,
 # By default we would produce an affine subspace but here we need a linear one
 affine = false)
W' = witness_set(W, L')
```

```
Witness set for dimension 1 of degree 3
```

HC currently only supports these basic computations with witness sets but we have plans to support more computations, e.g., numerical irreducible decomposition, in the future.

9.2 Miscellaneous Details

In this section, we discuss some of the implementation and design details of HC.

9.2.1 System Representation and Evaluation

As already discussed, HC uses for the problem formulation an embedded domain-specific language called ModelKit[2]. It allows to work with symbolic expressions and supports besides the basic arithmetic operations $+$, $-$, $*$ and $/$, also $\sin, \cos$ and $\log$. Symbolic expressions are well suited for the problem formulation since they preserve the original structure of the problem. Internally, the symbolic expressions are converted to a straight-line program for fast numerical evaluation. This is illustrated in Figure 9.4 and see, e.g., [BCS13, Sec. 4.1] for an introduction to straight-line programs.

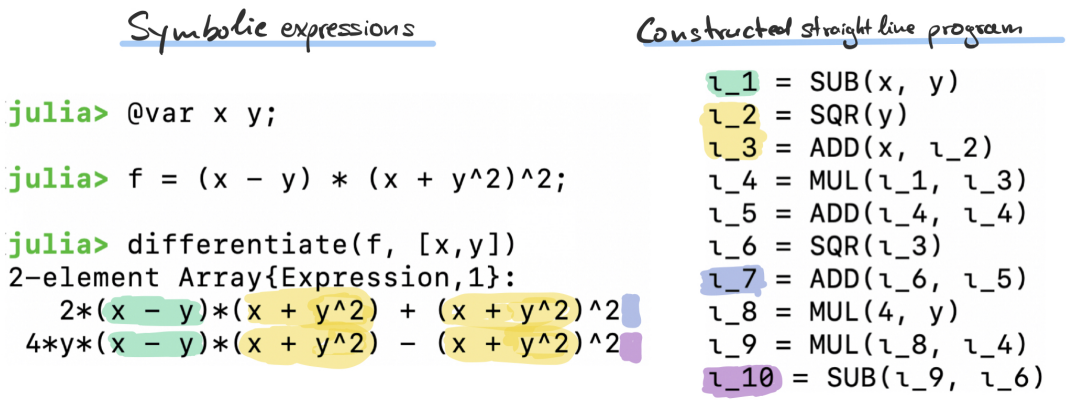

Figure 9.4: Transformation from symbolic expressions to a straight-line program while also performing common subexpression elimination. On the right-hand side is the straight-line program for evaluating the last two expressions on the left-hand side. The highlighting shows common subexpressions and the output.

Working with symbolic expressions has the advantage that the resulting straight-line programs are substantially smaller compared to the programs constructed from polynomials in a monomial basis. For example, the expression $(x + y)^5$ can be evaluated with just 4 basic arithmetic operations. Compare this to the expanded expression $x^5 + 5x^4y + 10x^3y^2 + 10x^2y^3 + 5xy^4 + y^5$, which requires more than 20 operations. Before converting a symbolic expression to a straight-line program, we perform transformations to reduce the number of operations necessary to evaluate the expression. These include the elimination of common subexpressions and, if possible, a multivariate version of Horner's method.

Once a symbolic expression is transformed into a straight-line program, HC supports two evaluation methods. The first method uses Julia's meta-programming capabilities to automatically generate and compile a function representing the straight-line program. The compilation results in the fastest possible evaluation of the given straight-line program but

[2] In the background, most symbolic operations are performed by the open source C++ library symengine. Source code available at github.com/symengine/symengine.

it also introduces a small constant overhead the first time a program is evaluated. Another downside of this approach is that the compilation cost scales, in our observation, super-linear with the size of the straight-line program. For large programs, the compilation can become prohibitively expensive.

The second method `HC` supports doesn't have any of the downsides of the compiled approach in exchange for a slower program evaluation. It evaluates a straight-line program by iterating over the list of operations the program consists of. For each operation in the list, the operation is performed and the result is stored in memory. Internally, `HC` performs optimizations to improve the efficiency of this approach. These include preallocation of the necessary memory, overwriting no more needed temporary results, and reducing cache misses by improving the memory access pattern. We refer to this approach as *interpreting* the straight-line program and to the previous approach as *compiling* the program.

To illustrate the performance difference between the compiled and interpreted approach, we consider the evaluation cost of the straight-line program in Figure 9.4. In `HC`, a straight-line program needs to be evaluated with different arithmetic types. During the path tracking, it is typically evaluated with complex double-precision arithmetic but as described in Chapter 4 sometimes we resort to the use of extended precision arithmetic in the form of complex double-double arithmetic. Additionally, for the certification routine described in Chapter 5 we possibly need to use high precision complex interval arithmetic provided by the `Arb` [Joh17] library via the `acb` arithmetic type. The different evaluation costs are shown in Table 9.5. The results demonstrate that with complex double-precision arithmetic the interpreted approach is almost ten times slower than the compiled approach. But with extended precision, the overhead reduces to only a factor of two. For the 256-bit complex interval arithmetic provided by `Arb`, the interpreted approach is even faster since we were able to implement efficient reuse of memory allocated by `Arb`. In our experience, these ratios between the different approaches are typical for a wide range of programs.

	compiled	interpreted
complex double	5 ns	45 ns
complex double double	91 ns	185 ns
acb (256 bit)	2049 ns	1762 ns

Table 9.5: Cost of evaluating the straight-line program in Figure 9.4 with complex double, complex double-double or complex interval arithmetic using the compiled or interpreted approach.

Whether it is faster to use the compiled or interpreted approach for a general `solve` computation is not always known a priori. For path tracking, it is not only necessary to evaluate a given homotopy but also to evaluate its Jacobian and to compute the local derivatives of a solution path. All these computations involve the evaluation of different straight-line programs. By default, `HC` uses for evaluating the homotopy and its Jacobian the compiled approach and for computing the local path derivatives the interpreted approach. In our experience, this results in a good tradeoff between compilation and run time.

Internally, the path tracking doesn't require a homotopy that is the direct result of the

compiled or interpreted approach. Instead, only a certain set of functions has to be implemented. This allows the user to implement custom homotopies, either to optimize performance or to express homotopies, that are not possible using the symbolic methods provided. In `HC`, this is used to, e.g., optimize the performance of a parameter homotopy and to efficiently express the composition of polynomial maps.

9.2.2 Computing Derivatives via Automatic Differentiation

`HC` uses the mixed-precision path tracking algorithm described in Chapter 4. For this, it is necessary to compute for a solution path $x(t)$ the local higher-order derivatives $x^{(\ell)}(t)$ for $\ell = 1, \dots, 4$. As observed in Lemma 4.11, this requires for $\ell = 1, \dots, 4$ the evaluation of

$$\frac{\partial^{\ell+1}}{\partial \lambda^{\ell+1}} H\Big(\sum_{k=0}^{\ell} x^{(k)} \lambda^k, t + \lambda\Big)|_{\lambda=0} .$$

These expressions can be evaluated efficiently and numerically robust by performing higher-order automatic differentiation. An excellent reference for automatic differentiation is the textbook [GW08] by Griewank and Walther where higher-order automatic differentiation is described in Chapter 13. In `HC`, we follow their approach.

We illustrate the computational advantage of automatic differentiation on the example of a parameter homotopy. For this consider the homotopy $H(x,t) := F(x, \phi(t))$ with $\phi(t) = (1-t)p + tq$. The first local derivative $x^{(1)}$ of a path $x(t)$ at (x_0, t_0) is given by

$$x^{(1)}(t_0) = H_x(x_0,t_0)^{-1} H_t(x_0,t_0) = H_x(x_0,t_0)^{-1} \frac{\partial}{\partial t} F(x_0; \phi(t_0)).$$

Using the chain rule, we obtain

$$\frac{\partial}{\partial t} F(x_0; \phi(t_0)) = F_p(x_0; \phi(t_0)) \dot{\phi}(t_0) \tag{9.2}$$

where F_p denotes the Jacobian with respect to the second argument. The naive computation of $\frac{\partial}{\partial t} F(x_0; \phi(t_0))$ is done by computing the different parts in (9.2) separately. This is often very inefficient since the Jacobian F_p is an $n \times m$ matrix where m is the number of parameters. Given that it is not uncommon to have more than 100 parameters, this Jacobian quickly blows up in size and with it the computational cost to compute (9.2). In contrast, using automatic differentiation the Jacobian vector product (9.2) can be computed in at most three times the number of operations of the original straight-line program [GW08, Section 4.6].

9.2.3 Tropical Endgame

Recall from Section 3.2 that a solution path $x(t) = (x_1(t), \dots, x_n(t))$ does not necessarily converge in the limit $t \to 0$ or even if so, not necessarily to a regular isolated solution of our target system. We also discussed that $x(t)$ has a coordinate wise expansion as a convergent Puiseux series and in Lemma 3.1 we showed that the valuation of this Puiseux expansion allows us to better understand the limit behavior of the path. We can consider these results

in the context of tropical geometry [MS15]. One of the main topics in tropical geometry is the study of the valuations of solutions to polynomial systems over a valued field. This is relevant for us since we can consider a polynomial homotopy $H(x,t)$ also as a polynomial system $F(x)$ with coefficients in the field of Puiseux series $\mathbb{C}\{\{t\}\}$. An alternative viewpoint also studied in tropical geometry, see, e.g., [MS15, Sec. 1.4] and [Mik04], is to consider families of complex varieties $V_t \subseteq (\mathbb{C}^*)^n$ with a real parameter t and to study the limit of the coordinatewise log-absolute value of the points in V_t as $t \to 0$. Considering the zero set of $H(x,t)$ as a family of complex varieties depending on t, the connection to our setting is apparent.

The standard approach to compute the valuation of a solution path is due to Huber and Verschelde [HV98]. Let us briefly recall their ideas. For this, we extend the notation from equation (3.4) and write the i-th entry $x_i(t)$ of the path $x(t)$ as

$$x_i(t) = a^{(i)}\, t^{\frac{w_i}{m}} + b^{(i)}\, t^{\frac{w_i+\delta_i}{m}} + \sum_{j>w_i+\delta_i} c_j^{(i)}\, t^{\frac{j}{m}} \tag{9.3}$$

where $a^{(i)}, b^{(i)} \neq 0$ and $\delta_i > 0$. For computing the valuations Huber and Verschelde use finite differences of the form $\frac{\log|x_i(t)|-\log|x_i(t')|}{\log|t|-\log|t'|} = \frac{w_i}{m} + O(t'^{\frac{\delta_i}{m}})$ for $t' > t$ with t and t' sufficiently small. Based on this formula, they develop an extrapolation method to compute the winding number m. This finite differences scheme is used in PHCpack [Ver99] and HOM4PS-2/3 [LLT08] to detect diverging paths.

In `HC`, we use an alternative approach for computing the valuations $\frac{w_i}{m}$. The basic idea of our method is to use a different formula for the valuation and combine it with higher-order information. Compared to Huber and Verschelde, our method does not suffer from roundoff or approximation errors introduced by applying a finite differences formula.

Given an analytic function $h : D \to \mathbb{C}$ on an open set $D \subseteq \mathbb{C}$, we define the following differential operator

$$\nu(h(t)) := t\frac{\mathrm{d}}{\mathrm{d}t}\log|h(t)|. \tag{9.4}$$

Decomposing $h(t) = u(t) + \sqrt{-1}v(t)$ into real and imaginary part we have the alternative formula

$$\nu(h(t)) = t\,\frac{u(t)\dot{u}(t) + v(t)\dot{v}(t)}{u(t)^2 + v(t)^2}. \tag{9.5}$$

If $h = (h_1, \ldots, h_n)$ is a tuple of analytic functions, we write $\nu(h) := (\nu(h_1), \ldots, \nu(h_n))$. The operator has the following important properties.

Proposition 9.1. *For $t > 0$ sufficiently small we have*

1. $\nu(x_i(t)) = \frac{w_i}{m} + O(t^{\frac{\delta_i}{m}})$.
2. $t\frac{\mathrm{d}}{\mathrm{d}t}\nu(x_i(t)) = O(t^{\frac{\delta_i}{m}})$.
3. *If $w_i \neq 0$, then $\nu(t\dot{x}_i(t)) = \frac{w_i}{m} + O(t^{\delta_i'/m})$, where $\delta_i' \geq \delta_i$.*
4. *If $w_i = 0$, then $\nu(t\dot{x}_i(t)) = \frac{\delta_i}{m} + O(t^{1/m})$.*

Proof. We first prove item (1). For this, we rewrite the formula in (9.3) as

$$x_i(t) = a^{(i)}\, t^{\frac{w_i}{m}} \cdot \Big(1 + \beta^{(i)}\, t^{\frac{\delta_i}{m}} \cdot (1 + \sum_{j>0} \gamma^{(i)}\, t^{\frac{j}{m}})\Big),$$

where $\beta^{(i)} = \frac{b^{(i)}}{a^{(i)}}$ and $\gamma^{(i)} = \frac{c^{(i)}}{b^{(i)}}$. Recall that $0 < t \leq 1$. Taking the logarithm of the absolute value on both sides of this yields

$$\log|x_i(t)| = \log|a^{(i)}| + \frac{w_i}{m}\log t + \log\Big|1 + \beta^{(i)}\, t^{\frac{\delta_i}{m}} \cdot (1 + \sum_{j>0} \gamma^{(i)}\, t^{\frac{j}{m}})\Big|.$$

Taking derivatives on both sides and multiplying by t yields

$$\nu(x_i(t)) = t\,\frac{\mathrm{d}}{\mathrm{d}t}\log|x_i(t)| = \frac{w_i}{m} + \nu(|1 + \beta^{(i)}\, t^{\frac{\delta_i}{m}} \cdot (1 + \sum_{j>0} \gamma^{(i)}\, t^{\frac{j}{m}})|).$$

Applying (9.5) to the last term in this equation shows that it has order $O(t^{\frac{\delta_i}{m}})$. This finishes the proof for item (1), and it also implies item (2).

For the remaining items, we derive from equation (9.3) the following formula for $t\dot{x}(t)$:

$$t\dot{x}(t) = \frac{w_i}{m}\, a^{(i)}\, t^{\frac{w_i}{m}} + \frac{w_i + \delta_i}{m}\, b^{(i)}\, t^{\frac{w_i+\delta_i}{m}} + \sum_{j>w_i+\delta_i} \frac{j}{m}\, c_j^{(i)}\, t^{\frac{j}{m}}.$$

We can now argue as above. If $w_i \neq 0$, then the leading term is $\frac{w_i}{m}\, a^{(i)}\, t^{\frac{w_i}{m}}$ and the second term is of order $O(t^{\frac{w_i+\delta_i}{m}})$. Item (1) applied to this case shows (3). Case (4) is shown analogously. This finishes the proof. □

Remark 9.2. The lemma is closely related to the limit process $\lim_{t\to 0} \frac{\log|x_i(t)|}{\log|t|} = \frac{w_i}{m}$ known as Maslov dequantization that relates the non-Archimedean valuation $\log|\cdot|$ with the Archimedean valuation $\mathrm{val}(\cdot)$; see, e.g., [dW17]. In [HS14], this is used to compute the vertices of the Newton polytope of a polynomial system using numerical methods.

All the differential operators in Proposition 9.1 can be evaluated using formula (9.5) and the local derivatives $x^{(1)}(t), x^{(2)}(t)$ and $x^{(3)}(t)$. These derivatives can be computed efficiently and accurately by the automatic differentiation techniques described in the previous subsection.

A challenge in using the valuation of a solution path $x(t)$ is that the results in Proposition 9.1 are only valid for sufficiently small t. Unfortunately, we did not find a method to quantify "sufficient small" and instead have to rely on heuristics.

To detect diverging paths, we make use of the higher-order derivatives. Let us denote the relative error between $\nu(x_i(t))$ and $\nu(t\dot{x}_i(t))$ by

$$\Delta := \frac{|\nu(x_i(t)) - \nu(t\dot{x}_i(t))|}{|\nu(x_i(t))|}.$$

Following the second item of Proposition 9.1, we also denote relative and absolute approximation error of $\nu(x_i(t))$ and $\nu(t\dot{x}_i(t))$ by

$$\varepsilon_{\text{abs}} = \max\{t\frac{\mathrm{d}}{\mathrm{d}t}\nu(x_i(t)),\, t\frac{\mathrm{d}}{\mathrm{d}t}\nu(t\dot{x}_i(t))\}, \quad \varepsilon_{\text{rel}} = \frac{\varepsilon_{\text{abs}}}{|\nu(x_i(t))|}. \tag{9.6}$$

If $\max(\Delta, \varepsilon_{\text{rel}})$ is smaller than a given threshold, then this indicates that the valuation doesn't change much anymore and that t is sufficiently small. However, this is not always the case and it is possible to construct adverse scenarios where this heuristic fails.

In addition to the valuation, we also consider during the endgame the *matrix condition number*

$$\kappa(t) := \|H_x(x(t),t)^{-1}\|_\infty \|H_x(x(t),t)\|_\infty.$$

$\kappa(t)$ has the property that $\lim_{t\to 0}\kappa(t) < \infty$, if and only if $x(t)$ converges to a regular solution; see, e.g., [MSW92b]. It is therefore suitable to detect whether the solution path ends in a singular solution. A problem with $\kappa(t)$ is that it is sensitive to the scaling of the rows of $H_x(x(t),t)$. In particular, a bad scaling of the homotopy $H(x,t)$ can therefore result in artificially large condition numbers. An alternative is the componentwise relative condition number $\text{cond}(t) = \min_D \|(DH_x(x(t),t))^{-1}\|_\infty \|DH_x(x(t),t)\|_\infty$ where the minimum is over all diagonal matrices; see [Hig02, Chapter 7]. However, this is has the undesirable property that it is possible that $\text{cond}(\tau)$ is small for all $\tau > 0$ but $\text{cond}(0) = \infty$, e.g., if a single row of the Jacobian converges to zero. A good balance we found between these two is to use

$$\kappa_D(t) := \|DH_x(x(t),t)^{-1}\|_\infty \|DH_x(x(t),t)\|_\infty$$

where $D = \arg\min_D \|(DH_x(x(\tau),\tau))^{-1}\|_\infty \|DH_x(x(\tau),\tau)\|_\infty$ for some fixed $\tau > 0$.

The endgame in `HC` is based on these heuristics but with small modifications to account for various ill-behaved situations. For the exact heuristics we refer to the source code of `HC`.

9.3 Impact

We have developed `HC` to allow engineers, researchers, and scientists to solve hard problems in nonlinear algebra without the need to first become an expert in numerical nonlinear algebra. Therefore, the metric on which the significance of the software should be judged is whether it is used to solve such problems. In particular, we do not attempt to provide a comprehensive set of benchmarks to compare `HC` against other packages. Benchmarks tend to only compare the performance of the general homotopies, e.g., total degree or polyhedral homotopy. But many applications are solved using the monodromy method and parameter homotopies and benchmarks typically do not reflect this.

While we don't provide benchmarks, we believe that a comprehensive set of example problems is very useful for the development of numerical nonlinear algebra software. A wide range of examples helps to expose weaknesses in the software that then can be improved. In the following, we want to give an example of a polynomial system that helped in the development of `HC`.

Example. The following system of three polynomials in three variables was communicated to us by Mohab Safey El-Din.

$$\begin{array}{l}(-9091098778555951517x^3y^4z^2 + 5958442613080401626y^2z^7 + 1759673386554817099 6x^2z^6 - \\ 1797917098637848647 4xyz^6 - 2382961149475678300x^4y^3 - 1541275815477198621 4xy^3z^3 + 133, \\ -1079819888181254963 2x^6y^3z - 1131827222545411145 0xy^9 - 1429141686930676684 1y^9z - \\ 5851790090514210599y^2z^8 + 1506706869524279972 7x^2y^3z^4 + 7716112995720175148x^3yz^3 + 171, \\ 1300541623984648518 3x^7y^3 + 4144861898662531651x^5z^4 - 8026818640767362673x^6 - \\ 6882178109031199747x^2y^4 + 7240929562177127812x^2y^3z + 5384944853425480296xyz^4 + 88)\end{array} \tag{9.7}$$

The system has 693 regular isolated solutions which agrees with its mixed volume. The Bézout bound of the system is 900. Due to the very small relative size of the constant coefficients, the system has many badly scaled solutions. This requires extra care in the path tracking and the endgame. This system was communicated to use since HC version 1.x was not able to solve the system correctly. Improvements to the endgame in HC 2.0 and the mixed-precision path tracking algorithm allow us to now solve the system correctly. In Table 9.6 we compare HC against other homotopy continuation solvers. The table shows that HC is the only solver that is able to solve the system correctly. NAG4M2 and Bertini report too many solutions and PHCpack misses solutions.

		NAG4M2	Bertini	PHCpack	HC
original	solutions	0	x	681	693
	runtime	0.5s	x	2.0s	0.4s
scaled	solutions	757	704	682	693
	runtime	9.6s	37.6s	2.0s	0.4s

Table 9.6: Results for solving the system (9.7) with different homotopy continuation solvers. Bertini and NAG4M2 were only able to produce solutions for the system after diving the system by the largest coefficient occuring in the system. The result for this scaled system are depicted in the scaled row. NAG4M2 and Bertini used a total degree homotopy to solve the system. PHCpack and HC used a polyhedral homotopy.

Finally, we demonstrate the impact of HC in the research community. For this, we show a variety of recent articles that use HC.

In [ST20], Sturmfels and Telen relate scattering amplitudes in particle physics to maximum likelihood estimation for discrete models in algebraic statistics. Using HC, they compute and certify all 188,112 critical points of a certain statistical model in 47 variables in a few minutes using a parameter homotopy. The symbolic modeling in HC allowed them to avoid clearing denominators and to directly work with rational functions. This greatly improved the performance and robustness of the computations.

In [LZBL20], Lindberg, Zachariah, Boston, and Lesieutre study the distribution of the number of real solutions to the power flow equations over varying electrical parameters. This study is performed using a combination of the monodromy method and the parameter homotopy implementation in HC. In [LBL20], Lindberg, Boston, and Lesieutre demonstrate how the monodromy method allows exploiting the symmetry in the power flow equations. This substantially decreases the computational cost. Note that the monodromy method in

HC makes it very easy to exploit symmetry[3]. Using HC, they solve the power flow equations for the cyclic graph on 20 vertices. This system has 1,847,560 complex solutions but ignoring the trivial solutions and up to symmetry it has 330,818. This example is the largest network to the authors' knowledge for which all solutions to the power flow equations have been found for a power system model.

In [BKK20], Brysiewicz, Kozhasov and Kummer classify transversal quintic spectrahedra by the location of 20 nodes on the respective real determinantal surface of degree 5. They identify 65 classes of such surfaces and find an explicit representative in each of them. Using HC, they compute explicit witnesses of spectrahedra for each combinatorial type. The computation for each combinatorial result is certified using the certification routine implemented in HC.

In [DTWY20], Duff, Telen, Walker, and Yahl introduce the Cox homotopy algorithm for solving a sparse system of polynomial equations on an associated compact toric variety. Their numerical experiments combine `Polymake.jl` [KLT20] with HC. This is just one example of the possibilities for combining state of the art methods in polyhedral geometry with numerical nonlinear algebra through Julia.

In [LSZ20], Lelièvre, Stoltz, and Zhang propose new Markov Chain Monte Carlo (MCMC) algorithms to sample probability distributions on submanifolds. These are some of the first MCMC methods that allow working with submanifolds consisting of multiple connected components. In the case of algebraic submanifolds, their implementation relies on HC to guarantee that the correct probability distribution is sampled.

Further notable work using HC includes "Sampling and homology via bottlenecks" [DREG20], "Tangent Quadrics in Real 3-Space" [BFS20], "Asymptotics of degrees and ED degrees of Segre products" [OSV20], "Dynamics of ERK regulation in the processive limit" [COST20], "Moment Ideals of Local Dirac Mixtures" [GKW20], "Autocovariance varieties of moving average random fields" [AP20] and "Logarithmic Voronoi cells" [AH21].

It is a great pleasure to see this diverse and growing set of researchers using HC and we hope that in the future it becomes a valuable tool for even more researchers.

9.4 Conclusion

In this chapter, we presented the software package `HomotopyContinuation.jl` for the numerical solution of polynomial systems. We demonstrated its functionality and discussed some of its implementation and design details. The software was of significant importance for this thesis since the computational results from Chapter 2, 6, 7, and 8 were all obtained with `HomotopyContinuation.jl`. We also showed the impact of `HomotopyContinuation.jl` on the broader research community by highlighting several recent articles where it played a critical role.

[3] This can be done via the `group_action` option for `monodromy_solve`.

10 Conclusion

In this thesis, we worked on different aspects of numerical nonlinear algebra. For the numerical solution of polynomial systems, we focused on the homotopy continuation method where the core numerical computation is tracking a solution path. We presented in Chapter 4 a new mixed-precision path tracking algorithm specifically designed for the demands of polynomial homotopy continuation methods. We demonstrated that this new algorithm is efficient while still being robust enough to handle numerically challenging situations.

Important for the use of numerical nonlinear algebra in mathematical proofs is the possibility to certify that computed approximate solutions of a polynomial system correspond to true (distinct) solutions of the system. We presented in Chapter 5 an implementation of a certification routine based on interval arithmetic and Krawczyk's method. We demonstrated that our method outperforms the existing state of the art `alphaCertified` by multiple orders of magnitude. An example where we used certification was in Chapter 2 Steiner's conic problem. There, we could reduce the certification time for our fully real instance to 3 seconds from the previous 36 hours with `alphaCertified`.

The new path tracking algorithm and the certification routine are both implemented in the software package `HomotopyContinuation.jl` that the author has developed together with Paul Breiding since late 2017. We presented in Chapter 9 the functionality of `HomotopyContinuation.jl` and shared some of its implementation and design details. We also demonstrated in Chapter 9 the significant impact of `HomotopyContinuation.jl` on the research community.

Besides Steiner's conic problem, we presented in this thesis three applications of numerical nonlinear algebra. All applications relied on `HomotopyContinuation.jl` to perform necessary computations. In the first application, we computed in Chapter 6 the degree of the orbit closure of the action of the projective linear group $\mathrm{PGL}(\mathbb{C}, 4)$ on cubic surfaces parameterized by points in $\mathbb{P}^{19}$. The result was 96120. The second application was in Chapter 7 the problem of maximum likelihood estimation for models of Gaussians whose covariance matrices lie in a given linear space. Using numerical nonlinear algebra, we computed the ML degree and dual ML degree for various models of linear covariance matrices. The last application was in Chapter 8 the study of tensegrity frameworks made from rigid bars and elastic cables. We used numerical nonlinear algebra to sample the semi-algebraic "catastrophe set" that characterizes a region of the parameter space that can trigger sudden large-scale shape changes.

Bibliography

[AF93] Paolo Aluffi and Carel Faber. Linear orbits of smooth plane curves. *J. Algebraic Geom.*, 2(1):155–184, 1993.

[AF00] Paolo Aluffi and Carel Faber. Linear orbits of arbitrary plane curves. *Michigan Math. J.*, 48(1):1–37, 2000.

[AFS16] Carlos Améndola, Jean-Charles Faugère, and Bernd Sturmfels. Moment Varieties of Gaussian Mixtures. *Journal of Algebraic Statistics*, 7(1):14–28, 2016.

[AG90] Eugene L. Allgower and Kurt Georg. *Numerical Continuation Methods: An Introduction*, volume 13 of *Series in Computational Mathematics*. Springer, 1990.

[AH21] Yulia Alexandr and Alexander Heaton. Logarithmic Voronoi cells. *Algebraic Statistics*, 11(3), 2021.

[And70] Theodore W. Anderson. Estimation of covariance matrices which are linear combinations or whose inverses are linear combinations of given matrices. *Essays in probability and statistics*, pages 1–24, 1970.

[AP20] Carlos Améndola and Viet Son Pham. Autocovariance varieties of moving average random fields. *Journal of Symbolic Computation*, 2020.

[Arn86] Vladimir I. Arnold. *Catastrophe Theory*. Springer, Berlin, 1986.

[Bai95] David H. Bailey. A Fortran-90 Based Multiprecision System. *ACM Transactions on Mathematical Software (TOMS)*, 21(4):379–387, 1995.

[BC11] Peter Bürgisser and Felipe Cucker. On a problem posed by Steve Smale. *Annals of Mathematics*, pages 1785–1836, 2011.

[BCS13] Peter Bürgisser, Michael Clausen, and Mohammad A. Shokrollahi. *Algebraic Complexity Theory*, volume 315. Springer, 2013.

[BEKS17] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59(1):65–98, 2017.

[Ber75] David N. Bernstein. The number of roots of a system of equations. *Functional Analysis and its applications*, 9(3):183–185, 1975.

[BFS20] Taylor Brysiewicz, Claudia Fevola, and Bernd Sturmfels. Tangent Quadrics in Real 3-Space. *arXiv preprint arXiv:2010.10879*, 2020.

[BGM96] George A. Baker and Peter Graves-Morris. *Padé Approximants*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2nd edition, 1996.

[BHM+20] Edgar A. Bernal, Jonathan D. Hauenstein, Dhagash Mehta, Margaret H. Regan, and Tingting Tang. Machine learning the real discriminant locus. *arXiv preprint arXiv:2006.14078*, 2020.

[BHS11a] Daniel J. Bates, Jonathan D. Hauenstein, and Andrew J. Sommese. Efficient path tracking methods. *Numerical Algorithms*, 58(4):451–459, Dec 2011.

[BHS11b] Daniel J. Bates, Jonathan D. Hauenstein, and Andrew J. Sommese. A parallel endgame. *Contemp. Math*, 556:25–35, 2011.

[BHSW] Daniel J. Bates, Jonathan D. Hauenstein, Andrew J. Sommese, and Charles W. Wampler. Bertini: Software for numerical algebraic geometry. Available at bertini.nd.edu with permanent doi: dx.doi.org/10.7274/R0H41PB5.

[BHSW08] Daniel J. Bates, Jonathan D. Hauenstein, Andrew J. Sommese, and Charles W. Wampler. Adaptive multiprecision path tracking. *SIAM Journal on Numerical Analysis*, 46(2):722–746, 2008.

[BI17] Peter Bürgisser and Christian Ikenmeyer. Fundamental invariants of orbit closures. *J. Algebra*, 477:390–434, 2017.

[BIMTW20] Laura Brustenga I Moncusì, Sascha Timme, and Madeleine Weinstein. 96120: The degree of the linear orbit of a cubic surface. *Le Matematiche*, 75(2):425–437, 2020.

[BKK20] Taylor Brysiewicz, Khazhgali Kozhasov, and Mario Kummer. Nodes on quintic spectrahedra. *arXiv preprint arXiv:2011.13860*, 2020.

[BKT08] Andrew Bashelor, Amy Ksir, and Will Traves. Enumerative algebraic geometry of conics. *The American Mathematical Monthly*, 115(8):701–728, 2008.

[BL13] Carlos Beltrán and Anton Leykin. Robust certified numerical homotopy tracking. *Foundations of Computational Mathematics*, 13(2):253–295, 2013.

[BLL19] Michael Burr, Kisun Lee, and Anton Leykin. Effective Certification of Approximate Solutions to Systems of Equations Involving Analytic Functions. In *Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation*, ISSAC '19, pages 267–274, 2019.

[BLW82] John P. Burg, David G. Luenberger, and Daniel L. Wenger. Estimation of structured covariance matrices. *Proceedings of the IEEE*, 70(9):963–974, 1982.

[BM20] Paul Breiding and Orlando Marigliano. Random points on an algebraic manifold. *SIAM Journal on Mathematics of Data Science*, 2(3):683–704, 2020.

[BP09] Carlos Beltrán and Luis Pardo. Smale's 17th problem: average polynomial time to compute affine and projective solutions. *Journal of the American Mathematical Society*, 22(2):363–385, 2009.

[Bro03] Christopher W Brown. QEPCAD B: a program for computing with semi-algebraic sets using CADs. *ACM Sigsam Bulletin*, 37(4):97–108, 2003.

[BRT] Paul Breiding, Kemal Rose, and Sascha Timme. Bacillus subtilis. `https://www.JuliaHomotopyContinuation.org/examples/bacillus-subtilis/`.

[BRT20] Paul Breiding, Kemal Rose, and Sascha Timme. Certifying zeros of polynomial systems using interval arithmetic. *arXiv preprint arXiv:2011.05000*, 2020.

[BS] Luis Benet and David P. Sanders. IntervalRootFinding.jl. `https://juliaintervals.github.io/IntervalRootFinding.jl`.

[BST20] Paul Breiding, Bernd Sturmfels, and Sascha Timme. 3264 Conics in a Second. *Notices of the American Mathematical Society*, 67:30–37, 2020.

[BT18] Paul Breiding and Sascha Timme. HomotopyContinuation.jl: A Package for Homotopy Continuation in Julia. In *International Congress on Mathematical Software*, pages 458–465. Springer, 2018.

[Cal78] Christopher R. Calladine. Buckminster Fuller's "Tensegrity" structures and Maxwell's rules for the construction of stiff frames. *International Journal of Solids and Structures*, 14(2):161–172, 1978.

[CDR07] Sanjay Chaudhuri, Mathias Drton, and Thomas S. Richardson. Estimation of a covariance matrix with zeros. *Biometrika*, 94(1):199–216, 2007.

[Chr89] E. Susanne Christensen. Statistical properties of I-projections within exponential families. *Scandinavian Journal of Statistics*, 16(1):307–318, 1989.

[CLL14] Tianran Chen, Tsung-Lin Lee, and Tien-Yien Li. Hom4PS-3: A Parallel Numerical Solver for Systems of Polynomial Equations Based on Polyhedral Homotopy Continuation Methods. In *Mathematical Software – ICMS 2014*, pages 183–190. Springer, 2014.

[CLL17] Tianran Chen, Tsung-Lin Lee, and Tien-Yien Li. Mixed cell computation in Hom4PS-3. *Journal of Symbolic Computation*, 79:516 – 534, 2017.

[CLO15] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undegraduate Texts in Mathematics. Springer, 4th edition, 2015.

[CMR20] Jane I. Coons, Orlando Marigliano, and Michael Ruddy. Maximum likelihood degree of the two-dimensional linear Gaussian covariance model. *Algebraic Statistics*, 11(2), 2020.

[COST20] Carsten Conradi, Nida Obatake, Anne Shiu, and Xiaoxian Tang. Dynamics of ERK regulation in the processive limit. *arXiv preprint arXiv:1910.14452*, 2020.

[CS03] Mark F. Coughlin and Dimitrije Stamenović. A Prestressed Cable Network Model of the Adherent Cell Cytoskeleton. *Biophysical Journal*, 84(2):1328–1336, 2003.

[CW93] David R. Cox and Nanny Wermuth. Linear dependencies represented by chain graphs. *Statistical Science*, pages 204–218, 1993.

[CW96] Robert Connelly and Walter Whiteley. Second-order rigidity and prestress stability for tensegrity frameworks. *SIAM J. Discrete Math.*, 9(3):453–491, 1996.

[Deu79] Peter Deuflhard. A stepsize control for continuation methods and its special application to multiple shooting techniques. *Numerische Mathematik*, 33(2):115–146, 1979.

[Deu11] Peter Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*, volume 35. Springer, 2011.

[DH79] Peter Deuflhard and Gerhard Heindl. Affine invariant convergence theorems for Newton's method and extensions to related methods. *SIAM Journal on Numerical Analysis*, 16(1):1–10, 1979.

[DHJ+18] Timothy Duff, Cvetelina Hill, Anders Jensen, Kisun Lee, Anton Leykin, and Jeff Sommars. Solving polynomial systems via homotopy continuation and monodromy. *IMA Journal of Numerical Analysis*, 39(3):1421–1446, 04 2018.

[DHO+16] Jan Draisma, Emil Horobeţ, Giorgio Ottaviani, Bernd Sturmfels, and Rekha R Thomas. The euclidean distance degree of an algebraic variety. *Foundations of computational mathematics*, 16(1):99–149, 2016.

[DHS20] Timothy Duff, Nickolas Hein, and Frank Sottile. Certification for polynomial systems via square subsystems. *Journal of Symbolic Computation*, 2020.

[DLRS10] Jesús De Loera, Jörg Rambau, and Francisco Santos. *Triangulations: Structures for Algorithms and Applications*, volume 25 of *Algorithms and Computation in Mathematics*. Springer, 2010.

[DMS21] Eliana Duarte, Orlando Marigliano, and Bernd Sturmfels. Discrete statistical models with rational maximum likelihood estimator. *Bernoulli*, 27(1):135–154, 02 2021.

[DMSM14] Claude Dellacherie, Servet Martinez, and Jaime San Martin. *Inverse M-matrices and Ultrametric Matrices*, volume 2118 of *Lecture Notes in Mathematics*. Springer, 2014.

[DR02] Mathias Drton and Thomas S. Richardson. A new algorithm for maximum likelihood estimation in Gaussian graphical models for marginal independence. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 184–191, 2002.

[Dre77] Franz-Josef Drexler. Eine Methode zur berechnung sämtlicher Lösungen von Polynomgleichungssystemen. *Numerische Mathematik*, 29(1):45–58, 1977.

[DREG20] Sandra Di Rocco, David Eklund, and Oliver Gäfvert. Sampling and homology via bottlenecks. *arXiv preprint arXiv:2011.14182*, 2020.

[DSLB+11] Gianluca De Santis, Alex B. Lennon, Federica Boschetti, Benedict Verhegghe, Pascal Verdonck, and Patrick J. Prendergast. How can cells sense the elasticity of a substrate? An analysis using a cell tensegrity model. *Eur Cell Mater*, 22:202–213, Oct 2011.

[DTWY20] Timothy Duff, Simon Telen, Elise Walker, and Thomas Yahl. Polyhedral Homotopies in Cox Coordinates. *arXiv preprint arXiv:2012.04255*, 2020.

[dW17] Timo de Wolff. Amoebas and their Tropicalizations – a Survey. In *Analysis Meets Geometry*, pages 157–190. Springer, 2017.

[EFSS20] Christopher Eur, Tara Fife, José Alejandro Samper, and Tim Seynnaeve. Reciprocal maximum likelihood degrees of diagonal linear concentration models. *arXiv preprint arXiv:2011.14182*, 2020.

[EH16] David Eisenbud and Joe Harris. *3264 and All That: A Second Course in Algebraic Geometry*. Cambridge University Press, 2016.

[Fab96] Eugène Fabry. Sur les points singuliers d'une fonction donnée par son développement en série et l'impossibilité du prolongement analytique dans des cas très généraux. In *Annales scientifiques de l'École Normale Supérieure*, volume 13, pages 367–399. Elsevier, 1896.

[Fel73] Joseph Felsenstein. Maximum-likelihood estimation of evolutionary trees from continuous characters. *American journal of human genetics*, 25(5):471, 1973.

[FLL16] Jianqing Fan, Yuan Liao, and Han Liu. An overview of the estimation of large covariance and precision matrices. *The Econometrics Journal*, 19(1):C1–C32, 2016.

[GD05] Hatice Gecegormez and Yasar Demirel. Phase stability analysis using interval Newton method with NRTL model. *Fluid Phase Equilibria*, 237(1-2):48—58, 2005.

[GGT13] Pedro Gonnet, Stefan Guttel, and Lloyd N. Trefethen. Robust padé approximation via svd. *SIAM review*, 55(1):101–117, 2013.

[GKW20] Alexandros Grosdos K. and Markus Wageringel. Moment Ideals of Local Dirac Mixtures. *SIAM Journal on Applied Algebra and Geometry*, 4(1):1–27, 2020.

[GLW05] Tangan Gao, Tien-Yien Li, and Mengnien Wu. Algorithm 846: MixedVol: a software package for mixed-volume computation. *ACM Transactions on Mathematical Software (TOMS)*, 31(4):555–560, 2005.

[GMW82] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical optimization*. Emerald Group Publishing Limited, 1982.

[GS04] J.-J. Gervais and Hassan Sadiky. A continuation method based on a high order predictor and an adaptive steplength control. *Zeitschrift für Angewandte Mathematik und Mechanik*, 84(8):551–563, 2004.

[GS05] Balajit Gopalan and Jay-Dean Seader. Application of interval Newton's method to chemical engineering problems. *Reliable Computing*, 1(3):215—223, 2005.

[Guc79] John Guckenheimer. The catastrophe controversy. *Math. Intelligencer*, 1(1):15–20, 1978/79.

[GW08] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, volume 105. Siam, 2008.

[GZ79] Camilo B. Garcia and Willard I. Zangwill. Finding all solutions to polynomial systems and other systems of equations. *Mathematical Programming*, 16(1):159–176, 1979.

[Ham94] James D. Hamilton. *Time series analysis*. Princeton University Press, 1994.

[Hat02] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[Hig97] Nicholas J. Higham. Iterative refinement for linear systems and LAPACK. *IMA Journal of Numerical Analysis*, 17(4):495–509, 1997.

[Hig02] Nicholas J. Higham. *Accuracy and stability of numerical algorithms*. Siam, 2002.

[HL08] Søren Højsgaard and Steffen L. Lauritzen. Graphical gaussian models with edge and vertex symmetries. *Journal of the Royal Statistical Society, Series B*, 70(5):1005–1027, 2008.

[HS95] Birkett Huber and Bernd Sturmfels. A polyhedral method for solving sparse polynomial systems. *Mathematics of computation*, 64(212):1541–1555, 1995.

[HS97] Birkett Huber and Bernd Sturmfels. Bernstein's theorem in affine space. *Discrete & Computational Geometry*, 17(2):137–141, 1997.

[HS10] Jonathan D. Hauenstein and Andrew J. Sommese. Witness sets of projections. *Applied Mathematics and Computation*, 217(7):3349–3354, 2010.

[HS12] Jonathan D. Hauenstein and Frank Sottile. Algorithm 921: alphaCertified: Certifying Solutions to Polynomial Systems. *ACM Trans. Math. Softw.*, 38(4), Aug 2012.

[HS14] Jonathan D. Hauenstein and Frank Sottile. Newton polytopes and witness sets. *Mathematics in Computer Science*, 8(2):235–251, 2014.

[HS17] Jonathan D. Hauenstein and Andrew J. Sommese. What is numerical algebraic geometry? *Journal of Symbolic Computation*, 79:499 – 507, 2017.

[HSW11] Jonathan Hauenstein, Andrew Sommese, and Charles Wampler. Regeneration homotopies for solving systems of polynomials. *Mathematics of Computation*, 80(273):345–377, 2011.

[HT20] Alexander Heaton and Sascha Timme. Catastrophe in elastic tensegrity frameworks. *arXiv preprint arXiv:2009.13408*, 2020.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.

[HV98] Birkett Huber and Jan Verschelde. Polyhedral end games for polynomial continuation. *Numerical Algorithms*, 18(1):91–108, 1998.

[IWS14] Donald E. Ingber, Ning Wang, and Dimitrije Stamenović. Tensegrity, cellular biophysics, and the mechanics of living systems. *Rep Prog Phys*, 77(4), Apr 2014.

[Jen16a] Anders Nedergaard Jensen. An implementation of exact mixed volume computation. In *Mathematical Software – ICMS 2016*, volume 9725 of *Lecture Notes in Computer Science*, pages 198–205. Springer, 2016.

[Jen16b] Anders Nedergaard Jensen. Tropical Homotopy Continuation. *arXiv preprint arXiv:1601.02818*, 2016.

[Joh17] Frederik Johansson. Arb: efficient arbitrary-precision midpoint-radius interval arithmetic. *IEEE Transactions on Computers*, 66:1281–1292, 2017.

[Kan48] Leonid V. Kantorovich. On Newton's method for functional equations. 59:1237–1240, 1948.

[Kau96] Göran Kauermann. On a Dualization of Graphical Gaussian Models. *Scandinavian Journal of Statistics*, 23(1):105–116, 1996.

[Kle95] Felix Klein. *Vorträge über ausgewählte Fragen der Elementargeometrie*. Teubner, Leipzig, 1895.

[KLT20] Marek Kaluba, Benjamin Lorenz, and Sascha Timme. Polymake.jl: A new interface to polymake. In *Mathematical Software – ICMS 2020*, volume 12097 of *Lecture Notes in Computer Science*, pages 377–385, 2020.

[Kra69] Rudolf Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4(3):187–201, 1969.

[KSS15] Virendra Kumar, Soumen Sen, and Sankar Shome. Inverse Kinematics of Redundant Manipulator using Interval Newton Method. *International Journal of Engineering and Manufacturing*, 2:19—-20, 2015.

[Kus75] Anatoli G. Kushnirenko. A Newton polyhedron and the number of solutions of a system of k equations in k unknowns. *Uspekhi Mat. Nauk*, 30:266–267, 1975.

[KV19] Mario Kummer and Cynthia Vinzant. The chow form of a reciprocal linear space. *The Michigan Mathematical Journal*, 68(4):831–858, 2019.

[KX94] R. Baker Kearfott and Zhaoyun Xing. An interval step control for continuation methods. *SIAM Journal on Numerical Analysis*, 31(3):892–914, 1994.

[Lai17] Pierre Lairez. A deterministic algorithm to compute approximate roots of polynomial systems in polynomial average time. *Foundations of Computational Mathematics*, 17(5):1265–1292, Oct 2017.

[Lau96] Steffen L. Lauritzen. *Graphical models*. Oxford University Press, 1996.

[LBL20] Julia Lindberg, Nigel Boston, and Bernard C. Lesieutre. Exploiting Symmetry in the Power Flow Equations Using Monodromy. *arXiv preprint arXiv:2011.14977*, 2020.

[Lee19] Kisun Lee. Certifying approximate solutions to polynomial systems on Macaulay2. *ACM Communications in Computer Algebra*, 53(2):45–48, 2019.

[Ley11] Anton Leykin. Numerical Algebraic Geometry for Macaulay2. *The Journal of Software for Algebra and Geometry: Macaulay2*, 3:5–10, 2011.

[LHT+10] Tim Liedl, Björn Högberg, Jessica Tytell, Donald E. Ingber, and William M. Shih. Self-assembly of three-dimensional prestressed tensegrity structures from DNA. *Nature Nanotechnology*, 5(9):520–524, 2010.

[Li03] Tien-Yien Li. Numerical Solution of Polynomial Systems by Homotopy Continuation Methods. In *Handbook of Numerical Analysis*, volume 11 of *Handbook of Numerical Analysis*, pages 209–304. Elsevier, 2003.

[LL11] Tsung-Lin Lee and Tien-Yien Li. Mixed volume computation in solving polynomial systems. *Contemp. Math*, 556:97–112, 2011.

[LLT08] Tsung-Lin Lee, Tien-Yien Li, and Chin-Ho Tsai. HOM4PS-2.0: a software package for solving polynomial systems by the polyhedral homotopy continuation method. *Computing*, 83(2):109, Oct 2008.

[LRS18] Anton Leykin, Jose Israel Rodriguez, and Frank Sottile. Trace test. *Arnold Mathematical Journal*, 4(1):113–125, 2018.

[LSZ20] Tony Lelièvre, Gabriel Stoltz, and Wei Zhang. Multiple projection MCMC algorithms on submanifolds. *arXiv preprint arXiv:2003.09402*, 2020.

[LWPQ17] Ke Liu, Jiangtao Wu, Glaucio H. Paulino, and H. Jerry Qi. Programmable deployment of tensegrity structures by stimulus-responsive polymers. *Scientific Reports*, 7(1):3511, 2017.

[LZBL20] Julia Lindberg, Alisha Zachariah, Nigel Boston, and Bernard C. Lesieutre. The Distribution of the Number of Real Solutions to the Power Flow Equations. *arXiv preprint arXiv:2010.03069*, 2020.

[Mac89] Wolfgang Mackens. Numerical differentiation of implicitly defined space curves. *Computing*, 41(3):237–260, 1989.

[Mar06] Giovanni M. Marchetti. Independencies induced from a graphical Markov model after marginalization and conditioning: the R package ggm. *Journal of Statistical Software*, 15(6):1–15, 2006.

[May17] Günter Mayer. *Interval Analysis*. De Gruyter, Berlin, Boston, 2017.

[MdCR17] Abraham Martín del Campo and Jose Israel Rodriguez. Critical points via monodromy and local methods. *J. Symbolic Comput.*, 79(3):559–574, 2017.

[Mik04] Grigory Mikhalkin. Decomposition into pairs-of-pants for complex algebraic hypersurfaces. *Topology*, 43(5):1035–1065, 2004.

[MM64] Hideyuki Matsumura and Paul Monsky. On the automorphisms of hypersurfaces. *J. Math. Kyoto Univ.*, 3:347–361, 1963/1964.

[Moo66] Ramon E. Moore. *Interval Analysis*, volume 4. Prentice-Hall, 1966.

[Moo77] Ramon E. Moore. A Test for Existence of Solutions to Nonlinear Systems. *SIAM Journal on Numerical Analysis*, 14(4):611–615, 1977.

[Mot03] René Motro. *Tensegrity: Structural systems for the future*. Butterworth-Heinemann, Oxford, 2003.

[MS87a] Michael I. Miller and Donald L. Snyder. The role of likelihood and entropy in incomplete-data problems: Applications to estimating point-process intensities and Toeplitz constrained covariances. *Proceedings of the IEEE*, 75(7):892–907, 1987.

[MS87b] Alexander P. Morgan and Andrew J. Sommese. A homotopy for solving general polynomial systems that respects m-homogeneous structures. *Applied Mathematics and Computation*, 24(2):101–113, 1987.

[MS89] Alexander P. Morgan and Andrew J. Sommese. Coefficient-parameter polynomial continuation. *Appl. Math. Comput.*, 29(2):123–160, 1989.

[MS15] Diane Maclagan and Bernd Sturmfels. *Introduction to Tropical Geometry*, volume 161 of *Graduate Studies in Mathematics*. American Mathematical Society, 2015.

[MS21] Mateusz Michałek and Bernd Sturmfels. *Invitation to Nonlinear Algebra*, volume 211 of *Graduate Studies in Mathematics*. American Mathematical Society, 2021.

[MSW90] Alexander P. Morgan, Andrew J. Sommese, and Charles W. Wampler. Computing singular solutions to nonlinear analytic systems. *Numerische Mathematik*, 58:669–684, 1990.

[MSW92a] Alexander P. Morgan, Andrew J. Sommese, and Charles W. Wampler. Computing singular solutions to polynomial systems. *Advances in Applied Mathematics*, 13(3):305 – 327, 1992.

[MSW92b] Alexander P. Morgan, Andrew J. Sommese, and Charles W. Wampler. A power series method for computing singular solutions to analytic systems. *Numerische Mathematik*, 63(1):391–409, 1992.

[MT08] Tomohiko Mizutani and Akiko Takeda. DEMiCs: A software package for computing the mixed volume via dynamic enumeration of all mixed cells. In *Software for Algebraic Geometry*, pages 59–79. Springer, 2008.

[NTI16] Jatin Narula, Abhinav Tiwari, and Oleg A. Igoshin. Role of Autoregulation and Relative Synthesis of Operon Partners in Alternative Sigma Factor Networks. *PLoS Comput. Biology*, 12(12), 2016.

[OSV20] Giorgio Ottaviani, Luca Sodomaco, and Emuanuele Ventura. Asymptotics of degrees and ED degrees of Segre products. *arXiv preprint arXiv:2008.11670*, 2020.

[Pel01] Sergio Pellegrino. *Deployable Structures*. Springer, Vienna, 2001.

[PM14] Mark M. Plecnik and John M. McCarthy. Numerical Synthesis of Six-Bar Linkages for Mechanical Computation. *Journal of Mechanisms and Robotics*, 6(3), 06 2014.

[Pou99] Mohsen Pourahmadi. Joint mean-covariance models with applications to longitudinal data: Unconstrained parameterisation. *Biometrika*, 86(3):677–690, 1999.

[PW73] Tim Poston and Alexander E. R. Woodcock. Zeeman's Catastrophe Machine. *Mathematical Proceedings of the Cambridge Philosophical Society*, 74(2):217—-226, 1973.

[RTV97] Felice Ronga, Alberto Tognoli, and Thierry Vust. The number of conics tangent to five given conics: the real case. *Rev. Mat. Univ. Compl. Madrid*, 10(2):391–421, 1997.

[Rum99] Siegfried M. Rump. INTLAB - INTerval LABoratory. In *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, 1999.

[SC87] Hubert Schwetlick and Jürgen Cleve. Higher order predictors and adaptive steplength control in path following algorithms. *SIAM Journal on Numerical Analysis*, 24(6):1382–1393, 1987.

[Sch79] Hermann Schubert. *Kalkül der abzählenden Geometrie*. Springer, 1879.

[SdO09] R.E. Skelton and M.C. de Oliveira. *Tensegrity Systems*. Springer-Verlag US, Boston, 2009.

[SJM18] Menachem Stern, Viraaj Jayaram, and Arvind Murugan. Shaping the topology of folding pathways in mechanical systems. *Nature Communications*, 9(1):4303, 2018.

[Sma86] Steve Smale. Newton's Method Estimates from Data at One Point. In *The Merging of Disciplines: New Directions in Pure, Applied, and Computational Mathematics*, pages 185–196. Springer, 1986.

[Sot] Frank Sottile. 3264 real conics. Available at `www.math.tamu.edu/~sottile/research/stories/3264/`.

[Sot97] Frank Sottile. Enumerative geometry for real varieties. In *Proceedings of Symposia in Pure Mathematics*, volume 62, pages 435–447. American Mathematical Society, 1997.

[SS93] Michael Shub and Steve Smale. Complexity of Bézout's theorem. I. Geometric aspects. *J. Amer. Math. Soc.*, 6(2):459–501, 1993.

[ST20] Bernd Sturmfels and Simon Telen. Likelihood Equations and Scattering Amplitudes. *arXiv preprint arXiv:2012.05041*, 2020.

[STZ20] Bernd Sturmfels, Sascha Timme, and Piotr Zwiernik. Estimating linear covariance models with numerical nonlinear algebra. *Algebraic Statistics*, 11(1):31–52, 2020.

[SU10] Bernd Sturmfels and Caroline Uhler. Multivariate gaussians, semidefinite matrix completion, and convex algebraic geometry. *Annals of the Institute of Statistical Mathematics*, 62(4):603–638, 2010.

[Sul18] Seth Sullivant. *Algebraic Statistics*, volume 194 of *Graduate Studies in Mathematics*. American Mathematical Society, 2018.

[Sun58] Teruo Sunaga. Theory of an interval algebra and its application to numerical analysis. *Research Association of Applied Geometry*, 2:29–46, 1958.

[SUZ20] Bernd Sturmfels, Caroline Uhler, and Piotr Zwiernik. Brownian motion tree models are toric. *Kybernetika*, 56(6):1154–1175, 2020.

[SV00] Andrew J. Sommese and Jan Verschelde. Numerical Homotopies to Compute Generic Points on Positive Dimensional Algebraic Sets. *J. Complex.*, 16(3):572—-602, September 2000.

[SVW01] Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler. Numerical Decomposition of the Solution Sets of Polynomial Systems into Irreducible Components. *SIAM Journal on Numerical Analysis*, 38(6):2022–2046, 2001.

[SVW02] Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler. Symmetric functions applied to decomposing solution sets of polynomial systems. *SIAM Journal on Numerical Analysis*, 40(6):2026–2046, 2002.

[SVW04] Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler. Homotopies for Intersecting Solution Components of Polynomial Systems. *SIAM Journal on Numerical Analysis*, 42(4):1552–1571, 2004.

[SW96] Andrew J. Sommese and Charles W. Wampler. Numerical algebraic geometry. In *The Mathematics of Numerical Analysis, volume 32 of Lectures in Applied Mathematics*, 1996.

[SW05] Andrew J. Sommese and Charles W. Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. World Scientific, 2005.

[THA14] Lam si Tung Ho and Cécile Ané. A linear-time algorithm for Gaussian and non-Gaussian trait evolution models. *Systematic biology*, 63(3):397–408, 2014.

[Tib02] Gunnar Tibert. *Deployable Tensegrity Structures for Space Applications*. PhD thesis, KTH Royal Institute of Technology, 2002.

[Tim20] Sascha Timme. Mixed Precision Path Tracking for Polynomial Homotopy Continuation. *arXiv preprint arXiv:1902.02968*, 2020.

[Tis01] Françoise Tisseur. Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 22(4):1038–1057, 2001.

[TP03] Gunnar Tibert and Sergio Pellegrino. Deployable Tensegrity Masts. In *44th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, 2003.

[TVBV20] Simon Telen, Marc Van Barel, and Jan Verschelde. A Robust Numerical Path Tracking Algorithm for Polynomial Homotopy Continuation. *SIAM Journal on Scientific Computing*, 42(6):3610–3637, 2020.

[Vai03] Israel Vainsencher. Hypersurfaces with up to six double points. *Communications in Algebra*, 31(8):4107–4129, 2003.

[Ver99] Jan Verschelde. Algorithm 795: PHCpack: A General-Purpose Solver for Polynomial Systems by Homotopy Continuation. *ACM Trans. Math. Softw.*, 25(2):251—276, 1999.

[VGC96] Jan Verschelde, Karin Gatermann, and Ronald Cools. Mixed-volume computation by dynamic lifting applied to polynomial system solving. *Discrete & Computational Geometry*, 16(1):69–112, 1996.

[VVB00] Konstantin Y. Volokh, Oren Vilnay, and M. Belsky. Tensegrity architecture explains linear stiffening and predicts softening of living cells. *J Biomech*, 33(12):1543–1549, Dec 2000.

[Wal50] Robert J. Walker. *Algebraic curves*, volume 642. Princeton University Press Princeton, 1950.

[WMS92] Charles W. Wampler, Alexander P. Morgan, and Andrew J. Sommese. Complete Solution of the Nine-Point Path Synthesis Problem for Four-Bar Linkages. *Journal of Mechanical Design*, 114(1):153–159, 1992.

[WS11] Charles W. Wampler and Andrew J. Sommese. Numerical algebraic geometry and algebraic kinematics. *Acta Numerica*, 20:469–567, 2011.

[WTNC+02] Ning Wang, Iva Marija Tolić-Nørrelykke, Jianxin Chen, Srboljub M. Mijailovich, James P. Butler, Jeffrey J. Fredberg, and Dimitrije Stamenović. Cell prestress. i. stiffness and prestress are closely associated in adherent contractile cells. *American Journal of Physiology-Cell Physiology*, 282(3):C606–C616, 2002.

[Yam85] T. Yamamoto. A unified derivation of several error bounds for Newton's process. *Journal of Computational and Applied Mathematics*, 12-13:179–191, 1985.

[ZGS+12] V.S. Zolesi, P.L. Ganga, L. Scolamiero, A. Micheletti, P. Podio-Guidugli, G. Tibert, A. Donati, and M. Ghiozzi. On an innovative deployment concept for large space structures. In *42nd International Conference on Environmental Systems*. American Institute of Aeronautics and Astronautics, 2012.

[ZHCG18] Emilio Zappa, Miranda Holmes-Cerfon, and Jonathan Goodman. Monte Carlo on manifolds: sampling densities and integrating functions. *Communications on Pure and Applied Mathematics*, 71(12):2609–2647, 2018.

[ZO15] Jing Yao Zhang and Makoto Ohsaki. *Tensegrity Structures: Form, Stability and Symmetry*. Springer Japan, Boston, 2015.

[ZUR17] Piotr Zwiernik, Caroline Uhler, and Donald Richards. Maximum likelihood estimation for linear Gaussian covariance models. *Journal of the Royal Statistical Society, Series B*, 79(4):1269–1292, 2017.