Preface

Systems of polynomial equations are a common occurrence in problem formulations in engineering, science, and mathematics. Solution sets of such systems, i.e., algebraic sets, are well-behaved, and effective algorithms have been developed in recent years to numerically compute and manipulate them. This field is called *numerical algebraic geometry*. Moreover, there is free software implementing these algorithms both for single computers and for computer clusters. As the developers of Bertini [22], we focus on this open-source software package in this book.

The goal of this book is threefold:

1. Explain enough background on solution sets of polynomial systems so that the reader has a basic understanding of the geometry of such sets and can understand both the data types and the breadth of the various computations of numerical algebraic geometry.

2. Show the reader how to use Bertini to solve polynomial systems efficiently and interpret the output correctly.

3. Provide a detailed manual of commands and settings for Bertini.

In addition to simple introductory problems, we present examples of polynomial systems arising in applications and treat each as a case study to show how the software can be used to deduce different sorts of information needed by a user. Files for all of the examples in this book can be found at the website for this book, `www.siam.org/books/se25`, where one may also find a link to the Bertini software website, `bertini.nd.edu`.

Applications

Polynomial systems have been solved numerically in many fields, including mechanism and robot kinematics, chemistry, computer-aided design (CAD), systems of nonlinear differential equations (ordinary and partial), differential algebra, economics, automatic control systems, magnetization, string theory, algebraic geometry, and others. Let us begin by discussing the polynomial systems that occur in three of these areas. These and other applications are considered later in the book.

In the theory of mechanisms, a large set of problems leads to polynomial systems. For example, for a serial-link robotic arm, the equations relating the robot's hand position to the rotations of its joints and its physical parameters, such as the length of the links making up the arm, are naturally polynomials. Typically the robot can reach a given hand location in multiple poses, each given by a set of joint rotations. Finding all such poses is a straightforward application of polynomial solving software [171, 181].

Of course, mechanisms have uses far beyond robotics, e.g., medical instruments, windshield wipers, airplane landing gear, metal-stamping presses, and all manner of devices

with jointed moving parts fall under the same heading. Any time rigid parts are connected by the most common types of joints—from simple hinges to ball-and-socket joints—their motion is well-modeled by a system of polynomial equations [37, 146, 126, 105, 184]. These equations might be used to analyze the motion of a proposed design (mechanism analysis) or to find a design that produces a desired motion (mechanism synthesis). Synthesis problems can be especially challenging, as they often involve several copies of the basic polynomial system, each copy expressing the requirement to reach a different position along the desired motion and all to be solved simultaneously.

Another source of polynomial systems arises in chemistry (see, e.g., [131]) and in combustion engineering [3]. Each equation of the form

$$2H_2 + O_2 \rightleftharpoons 2H_2O$$

leads to a polynomial relation between the molar quantities arising in the equation, e.g., $[H_2]$, $[O_2]$, and $[H_2O]$ in the above equation. The condition for equilibrium between these species takes the form

$$k[H_2O]^2 = [H_2]^2[O_2],$$

where k is an appropriate rate constant. After writing one such equation for each possible reaction between species in a chemical system and adding mass conservation equations according to the initial concentrations of the species, one has a system of polynomials whose solutions are all possible equilibrium states for the reaction.

In many cases, discretization of nonlinear differential equations leads to systems of polynomials. Methods based on a polynomial solver such as Bertini give a general approach to boundary value problems for single ordinary differential equations (ODEs) [4, 5, 6]. For systems of ODEs, such methods yield solutions not easily found by other methods [74]. Systems of nonlinear partial differential equations (PDEs) arising in such areas as tumor growth can lead to large polynomial systems with thousands of variables. The methods in this book yield new solutions and much useful numerical information for such problems; see, e.g., [75, 76, 77, 78, 79, 82, 81]. These methods, and in particular Bertini's endgame routines to compute singular solutions, have led to a homotopy method based on weighted essentially nonoscillatory (WENO) schemes [122] for solving steady-state problems of hyperbolic conservation laws. This advance is an order-of-magnitude improvement over the usual approaches [80] for moderate-sized problems and is an even greater improvement as the size of the problems increases.

Software

Advances in methodology and in computer speed now make problems that once seemed intractable easy and straightforward to solve. Consider, for example, the nine-point path synthesis problem for four-bar linkages, which we discuss in detail in §5.5.2. This was first posed in 1923 [9], but it took until 1962 for a few solutions to be found by a precursor to modern polynomial continuation [148, 149]. The first computation of complete solution sets in 1992 [182, 183], using continuation, took weeks of expert formulation of the equations and 300 hours of mainframe CPU time. As of 2010, a computer running Bertini on one processor* could solve an initial example of the problem in 8 hours, and a parallel computer with 65 cores* could do it in under 8 minutes [93]. After an initial solve, just eight cores of the same parallel computer can find all 1442 triplets of solutions of any

*2.4-GHz Opteron processor, 64-bit Linux.

*Eight nodes of 2.33-GHz quad-core Xeon 5410 plus 1 manager, 64-bit Linux.

other instance of the nine-point problem using a parameter homotopy (as described in §§6.6 and 6.11) in just 8 seconds! This kind of computing power is now available in many consumer-grade personal computers. Due to improved algorithms such as regeneration (§5.4), the user no longer needs to be an expert in continuation to get results like these.

Our software package, Bertini, offers easy access to the latest advances in homotopy methods. A key feature is the use of adaptive multiprecision, which makes the computations robust and accurate. Also, Bertini supports both serial processing on a single CPU and parallel processing,* allowing the user to take advantage of the availability of multicore technology in personal computers and larger installations with hundreds of CPUs.

Starting from the first release of Bertini version 1.0 in April 2008 up to the current release (version 1.4) used in this book, there has been a steady growth in the Bertini user community. It has become apparent that an up-to-date and thorough manual would sustain this growth by helping new users get started more quickly and by helping experienced users get the most out of all that Bertini has to offer. It is in that spirit that we write this book.

Related work

Two of the authors of this book, Sommese and Wampler, already published a full-length monograph on the subject of polynomial continuation [161]. However, that book concentrates on presenting the theory behind the algorithms, while this new book concentrates on practical usage of the algorithms. Although we follow roughly the same order of development, this book is very different from its predecessor. Here, we develop material in terms of applications and case studies, which we solve completely using Bertini. We try to convey what can be done, how to do it with Bertini, and how this can answer questions in engineering and science. Compared to [161], we give relatively little information on why the algorithms work and only enough information on how they work to enable the user to understand the optional settings that can be tweaked to optimize performance. We also take the opportunity to give the reader a glimpse of newer developments in the field, some of which are fully implemented in Bertini and some that can be explored by interfacing to Bertini through its facility for "user-defined homotopy."

For academics interested in the field, the combination of [161] with this book provides a full complement of material, in both theory and practice, suitable for a course in numerical algebraic geometry. Mechanical engineers might prefer to use the survey article [184] to overview the relationship between algebraic kinematics and numerical algebraic geometry and use this book to equip students with the latest computational tools in the field, a combination that could be called "numerical algebraic kinematics."

Overview of this book

Part I begins by introducing in Chapter 1 the main objects of interest, namely, polynomial systems and their solution sets. We focus on isolated solutions of polynomial systems (positive-dimensional solution sets become the main focus of attention starting in Chapter 8), using a number of simple examples to illustrate the key concepts, including a first discussion of singularities and multiplicity. We note that our methods depend on some random selection of constants and discuss how this makes the algorithms work correctly with "probability one." We give a brief introduction to straight-line programs for evaluation of polynomials and Bertini's facilities to make it easy to input polynomials in this form.

*Bertini's parallel facilities are primarily based on the message passing interface (MPI).

We also discuss in the Chapter 1 situations where the number of equations is not equal to the number of variables. This is known as a *nonsquare system*. We treat the case of both too few and too many equations. The treatment of systems with excess equations again requires the use of random numbers, so we discuss the related topic of genericity and what it means in practice.

We also give a brief overview of some algebraic geometry background involving algebraic sets, randomization, and Bertini theorems. These and similar sections throughout the book are marked with a star (*) to indicate that they may be skipped by those who only need access to the core computational routines for polynomial solving. These sections are included to help those who desire a deeper understanding of the theoretical underpinnings of numerical algebraic geometry.

In Chapter 2, we discuss the fundamental continuation method and issues that arise in using it. We start this initial discussion with isolated solutions defined by N polynomials in N variables. Many polynomial systems arising in practice are in this format or can easily be put into this format. We show how to run examples with Bertini using the simplest algorithm in the toolset, the total-degree homotopy. We discuss some of the basic components of continuation, e.g., ODE predictors and a Newton corrector, and how settings in Bertini control these components.

In Chapter 3, we consider two of the most important advances implemented in Bertini: adaptive precision and endgames. Both of these algorithms are useful in treating "bad" (singular) and nearly bad points that we encounter during continuation. Adaptive precision allows us to move beyond near singularities without losing accuracy. Endgames allow us to accurately compute singular solutions without relying on computations at or very near the singularity.

In Chapter 4, we discuss projective space, which provides a completion of Euclidean space. This makes the notion of solutions at infinity meaningful and straightforward for computation.

Chapter 5 covers the main alternatives to the total-degree homotopy. We discuss the Bézout number of solutions and why acknowledging the different structures of systems significantly decreases the computation needed to solve a polynomial system. Up to this point, if the reader has been running the example problems in Bertini, he or she will have already experienced multihomogeneous continuation by simply placing variables into logical groups. With one change in the settings, the same input file will invoke the most recent homotopy type, the regeneration homotopy. For a casual user, it may be enough to know what switch to flip, but a more serious user will find it worthwhile to understand all the homotopy types that are available and how they may sometimes offer advantages or suffer from disadvantages. By understanding the homotopies in more detail, the reader will be better able to monitor the runtime progress of each homotopy and can understand the statistical information provided with each computational run.

In Chapter 6, we introduce parameter spaces. Many problems in engineering and science need to be solved many times for different values of parameters. Recalling the serial-link robot example mentioned earlier, it is easy to envision scenarios where one may wish to find the arm poses that achieve many successive hand positions. One of the important properties of continuation methods is that if a problem is solved once generically (at a random point in the complex parameter space), the resulting solutions can be used to initiate successive solves for different parameter values. Many times the successive solves take a small fraction of the time that was needed for the initial solve. Indeed, speedups by a factor of 100 are common; see, e.g., [133, 134, 182]. We also discuss complex and real solutions. Even though real solutions are often the object of interest, continuing real solutions alone from the solution at a generic value of the parameters will not generally

lead to all the real solutions for other values of the parameters, i.e., complex solutions may continue to real solutions and vice versa.

Chapter 7 covers a number of advanced topics related to the computation of isolated solutions. One benefit of numerical algebraic geometry is that approximations to solutions may be computed to extremely high accuracy with little work. This is called endpoint sharpening and is covered in detail in this chapter. There are also brief discussions on how Bertini determines matrix ranks and how polynomial systems may be scaled for more efficient computation.

Part II treats positive-dimensional solution sets of polynomial systems, beginning with Chapter 8, in which we discuss different ways of representing such sets, including the classical irreducible decomposition of an algebraic set and its numerical analogue, the numerical irreducible decomposition. We give a careful discussion of witness sets and supersets. Witness sets are the data structure we use to represent positive-dimensional sets, and witness supersets are the first step in their computation. We discuss how Bertini can be used to sample a component once a witness set has been found and to determine whether a given solution point lies on a particular component.

With the concepts of witness sets and witness supersets in place, Chapter 9 discusses how witness supersets are computed. We discuss how one can obtain a witness superset for components at each dimension independently. When one wants to find solution components at every dimension, cascade algorithms are often a good choice, as described in this chapter. All of this can be accomplished with no knowledge of the inner workings of Bertini and via a single Bertini call, so this chapter may not be essential for some readers.

In Chapter 10, we describe algorithms used to convert a witness superset for the union of all solution components in a given dimension into a union of witness sets, one for each irreducible component. Even though many users will be content using the default settings for this conversion, it is useful for some users to have a more thorough understanding of the techniques happening behind the scenes, especially if they push the limits of what Bertini can handle.

Chapter 11 wraps up the treatment of positive-dimensional solutions with brief discussions of some advanced topics. We explain how to ask Bertini to print out detailed information concerning a witness set and how to compute solution sets at specific dimensions instead of performing a full decomposition of all dimensions. We also discuss a special software module that, given a polynomial system and a solution point, analyzes the derivatives of the system to determine the local dimension of the solution set. For example, this is of interest in kinematics, where one may have a mechanism in one assembly configuration and would like to know the number of degrees of freedom with which it can move from that pose. The method used is highly related to the concept of multiplicity and to the deflation method for reducing multiplicity, so these are also discussed.

In Part III, the book moves beyond the irreducible decomposition and looks at some of the things that can be done once witness sets are in hand. Many of these algorithms have not yet been fully implemented in Bertini, but users can explore them via user-defined homotopies. We mainly include these topics to give a view of the current state of research in the area.

First, in Chapter 12, we look at intersecting components. A more general operation is to intersect the projection of components, which gives rise to the idea of a fiber product. Fiber products are of particular interest as a means of finding exceptional sets: places where the solution set of a parameterized system increases in dimension. One application is in finding exceptional mechanisms, known as overconstrained mechanisms in the kinematics literature.

Chapter 13 looks at the issue of singularities on positive-dimensional sets. We see how to specify rank-dropping conditions and how this can be used to find isosingular sets: sets of points that share a common singularity structure. This prepares us for Chapter 14.

In Chapter 14, real solutions are considered. The standard methods of numerical algebraic geometry work over the complex numbers, and the problem of finding real solutions is surprisingly difficult. This is an area that is under active development. We discuss briefly the numerical approach to solving these systems and what "solving" means.

Next, Chapter 15 looks at some applications to algebraic geometry and, in particular, ways to compute invariants of algebraic sets, such as the geometric genus of a curve and characteristic classes. This chapter will be of particular interest to algebraic geometers.

In Chapter 16, we discuss how numerical algebraic geometry deals with images of algebraic sets under algebraic maps. The numerical approach to this situation, which may be called "numerical elimination theory," is both easier to use and applicable to larger problems than classical elimination theory.

One of the relatively new application areas for polynomial systems is in the solution of systems of nonlinear PDEs. These typically lead to very large systems with hundreds to thousands of variables. To follow even a single path of such a system can be challenging, requiring techniques such as sparse linear algebra. Chapter 17 touches on what has been done in this area.

In the appendices of Part IV, we give a thorough users manual for Bertini, beginning with a "Quick Start Guide" in Appendix A. We highly recommend that the reader install and experiment with Bertini while reading the book.

Supplementary materials

A link to a webpage containing supplementary materials for this book can be found at `www.siam.org/books/se25`. In particular, the various short Bertini input files appearing in the text are available there along with input files for longer systems that are not printed verbatim here, such as ones for the six-revolute (6R) robot inverse problem, the Stewart–Gough forward kinematics problem (and its several Griffis–Duffy variants), and several more.

Acknowledgments

This book could not have come to fruition without the various forms of support we have received from a number of generous individuals and organizations.

First, we wish to express our gratitude to the funding agencies that have financially supported so much of our work: the National Science Foundation, the Air Force Office of Sponsored Research, and the Department of Defense. We have also benefitted from the support of various research institutes, including the Fields Institute, the Institute for Mathematics and Its Applications (IMA), and Institut Mittag-Leffler. We also are grateful to our employers during the course of writing this book—Colorado State University, General Motors, North Carolina State University, Texas A&M University, and the University of Notre Dame—for providing various opportunities to work on this book and the developments described within it.

Several people read and commented on various drafts of this manuscript. Their contributions improved the final quality of this book, and their efforts are greatly appreciated. Those individuals are Daniel Baker, Noah Daleo, Brent Davis, Eric Hanson, Wenrui Hao, Matthew Niemerg, and Francesco Pancaldi.