The IMA Volumes in Mathematics and its Applications

Volume 148

Series Editors
Douglas N. Arnold Arnd Scheel

Institute for Mathematics and its Applications (IMA)

The Institute for Mathematics and its Applications was established by a grant from the National Science Foundation to the University of Minnesota in 1982. The primary mission of the IMA is to foster research of a truly interdisciplinary nature, establishing links between mathematics of the highest caliber and important scientific and technological problems from other disciplines and industries. To this end, the IMA organizes a wide variety of programs, ranging from short intense workshops in areas of exceptional interest and opportunity to extensive thematic programs lasting a year. IMA Volumes are used to communicate results of these programs that we believe are of particular value to the broader scientific community.

The full list of IMA books can be found at the Web site of the Institute for Mathematics and its Applications:

http://www.ima.umn.edu/springer/volumes.html

Presentation materials from the IMA talks are available at

http://www.ima.umn.edu/talks/

Douglas N. Arnold, Director of the IMA

* * * * * * * * * *

IMA ANNUAL PROGRAMS

1982–1983	Statistical and Continuum Approaches to Phase Transition
1983–1984	Mathematical Models for the Economics of Decentralized Resource Allocation
1984–1985	Continuum Physics and Partial Differential Equations
1985–1986	Stochastic Differential Equations and Their Applications
1986–1987	Scientific Computation
1987–1988	Applied Combinatorics
1988–1989	Nonlinear Waves
1989–1990	Dynamical Systems and Their Applications
1990–1991	Phase Transitions and Free Boundaries
1991–1992	Applied Linear Algebra
1992–1993	Control Theory and its Applications
1993–1994	Emerging Applications of Probability
1994–1995	Waves and Scattering
1995–1996	Mathematical Methods in Material Science
1996–1997	Mathematics of High Performance Computing

(Continued at the back)

Michael Stillman Nobuki Takayama
Jan Verschelde
Editors

Software for Algebraic Geometry

Springer

Editors
Michael Stillman
Department of Mathematics
Cornell University
Ithaca, NY 14853-4201
USA

Nobuki Takayama
Department of Mathematics
Kobe University
Rokko, Kobe 657-8501
Japan

Jan Verschelde
Department of Mathematics
University of Illinois
Chicago, IL 60607-7045
USA

Series Editors
Douglas N. Arnold
Arnd Scheel
Institute for Mathematics and its
Applications
University of Minnesota
Minneapolis, MN 55455
USA

ISBN: 978-0-387-78132-7 e-ISBN: 978-0-387-78133-4
DOI: 10.1007/978-0-387-78133-4

Library of Congress Control Number: 2008923357

Mathematics Subject Classification (2000): 11R09, 11Y99, 12D05, 13P10, 14P05, 14Q05, 14Q10
52A39, 52B20, 52B55, 65D18, 65F15, 65F20, 65F22, 65H10, 65H20, 65-04, 91-08

Camera-ready copy provided by the IMA.

9 8 7 6 5 4 3 2 1

springer.com

FOREWORD

This IMA Volume in Mathematics and its Applications

SOFTWARE FOR ALGEBRAIC GEOMETRY

contains papers presented at a highly successful one-week workshop on the same title. The event was an integral part of the 2006-2007 IMA Thematic Year on "Applications of Algebraic Geometry." We are grateful to all the participants for making this workshop a very productive and stimulating event. Special thanks to Michael E. Stillman (Department of Mathematics, Cornell University), Nobuki Takayama (Department of Mathematics, Kobe University), and Jan Verschelde (Department of Mathematics, Statistics and Computer Science, University of Illinois at Chicago) for their superb role as workshop organizers and editors of these proceedings.

We take this opportunity to thank the National Science Foundation for its support of the IMA.

Series Editors

Douglas N. Arnold, Director of the IMA

Arnd Scheel, Deputy Director of the IMA

PREFACE

The workshop on "Software for Algebraic Geometry" was held in the week from 23 to 27 October 2006, as the second workshop in the thematic year on Applications of Algebraic Geometry at the IMA.

Algorithms in algebraic geometry go hand in hand with software packages that implement them. Together they have established the modern field of computational algebraic geometry which has come to play a major role in both theoretical advances and applications. Over the past fifteen years, several excellent general purpose packages for computations in algebraic geometry have been developed, such as CoCoA, Singular and Macaulay 2. While these packages evolve continuously, incorporating new mathematical advances, they both motivate and demand the creation of new mathematics and smarter algorithms.

Surrounding the general packages, a host of specialized packages for dedicated and focused computations have created a platform for the interaction of algebraic geometry with numerous other areas of mathematics including optimization, combinatorics, polyhedral geometry, numerical analysis and computer science. The workshop brought together a wide array of theoreticians and practitioners interested in the development of algorithms and software in algebraic geometry at this workshop. Such interactions are essential for dramatic increases in the power and applicability of algorithms in the field.

There were 89 registered participants at the workshop. At four talks a day, 20 regular 50 minutes talks were scheduled. On Monday evening, 10 posters were presented. On Wednesday and Thursday evening we had respectively 5 and 6 software demonstrations. The list of featured software packages includes Macaulay 2, SAGE, HomLab, Bertini, APAtools, PHClab, PHCmaple, PHCpack, KNOPPIX/Math, D-modules for Macaulay 2, Singular, Risa/Asir, CRACK, diffalg, RIFsimp, Gambit, FGb/RS, CoCoALib, 4ti2, PHoMpara, SYNAPS, DEMiCs, Magma, Kronecker, SOSTOOLS, Gfan, Maple 11.

The IMA systems group had installed many of these programs on the computers at the IMA. At the poster session, the participants were given the opportunity to install the featured software systems on their laptop. A demonstration cluster computer of Rocketcalc was running during the poster session and accessible to all participants during the workshop.

The evening before the workshop dinner on Tuesday started with a problem session. Prior to this session we made a list of problem descriptions available on the web site. The workshop ended on Friday evening with some additional problems, discussion on the posted problems, and a presentation of Jiawang Nie about the application to semidefinite programming to solve systems of algebraic equations which arise from differential equations. We

are also happy that several new research projects were stimulated by this problem session. Some results are going to appear elsewhere.

Instead of the "second chances" (usual for IMA workshops), the participants were given the opportunity to test the software systems on Wednesday and Thursday evening. The evening session started with a one hour plenary session, where each software system on demo in the evening was briefly explained. Following this plenary session, the participants moved to the 4th floor of Lind Hall, to experience the software systems on the computers in the open poster area, or in parallel, in the classroom 409.

The IMA systems group worked hard in the weeks leading up to the workshop to install the software systems. Their effort benefited not only the workshop participants, but all all subsequent participants to the thematic year, as they found their workstations equipped with the latest software tools in algebraic geometry.

The papers in this volume describe the software packages Bertini, PHClab, Gfan, DEMiCs, SYNAPS, TrIm, Gambit, ApaTools, and the application of Risa/Asir to a conjecture on multiple zeta values. We thank the participants to the workshop, the authors and the anonymous referees. We are grateful to the editorial staff of the IMA, Patricia V. Brick and Dzung N. Nguyen, for their dedication and care.

Michael E. Stillman
Department of Mathematics
Cornell University
http://www.math.cornell.edu/People/Faculty/stillman.html

Nobuki Takayama
Department of Mathematics
Kobe University
http://www.math.sci.kobe-u.ac.jp/ taka/

Jan Verschelde
Department of Mathematics, Statistics and Computer Science
University of Illinois at Chicago
http://www2.math.uic.edu/ jan/

CONTENTS

Foreword v

Preface vii

Software for numerical algebraic geometry: A paradigm and progress towards its implementation 1
Daniel J. Bates, Jonathan D. Hauenstein, Andrew J. Sommese, and Charles W. Wampler II

PHClab: A MATLAB/Octave interface to PHCpack 15
Yun Guan and Jan Verschelde

Computing Gröbner fans and tropical varieties in Gfan 33
Anders Nedergaard Jensen

On a conjecture for the dimension of the space of the multiple zeta values 47
Masanobu Kaneko, Masayuki Noro, and Ken'ichi Tsurumaki

DEMiCs: A software package for computing the mixed volume via dynamic enumeration of all mixed cells 59
Tomohiko Mizutani and Akiko Takeda

SYNAPS, a library for dedicated applications in symbolic numeric computing 81
Bernard Mourrain, Jean-Pascal Pavone, Philippe Trebuchet, Elias P. Tsigaridas, and Julien Wintz

Tropical implicitization and mixed fiber polytopes 111
Bernd Sturmfels and Josephine Yu

Towards a black-box solver for finite games: Computing all equilibria with Gambit and PHCpack 133
Theodore L. Turocy

ApaTools: A software toolbox for approximate polynomial algebra 149
Zhonggang Zeng

List of workshop participants 169

SOFTWARE FOR NUMERICAL ALGEBRAIC GEOMETRY: A PARADIGM AND PROGRESS TOWARDS ITS IMPLEMENTATION

DANIEL J. BATES*, JONATHAN D. HAUENSTEIN†, ANDREW J. SOMMESE‡, AND CHARLES W. WAMPLER II§

Abstract. Though numerical methods to find all the isolated solutions of nonlinear systems of multivariate polynomials go back 30 years, it is only over the last decade that numerical methods have been devised for the computation and manipulation of algebraic sets coming from polynomial systems over the complex numbers. Collectively, these algorithms and the underlying theory have come to be known as numerical algebraic geometry. Several software packages are capable of carrying out some of the operations of numerical algebraic geometry, although no one package provides all such capabilities. This paper contains an enumeration of the operations that an ideal software package in this field would allow. The current and upcoming capabilities of Bertini, the most recently released package in this field, are also described.

Key words. Homotopy continuation, numerical algebraic geometry, polynomial systems, software, Bertini.

AMS(MOS) subject classifications. 65H10, 65H20, 65-04, 14Q99.

1. Introduction.

Numerical algebraic geometry refers to the application of numerical methods to compute the solution sets of polynomial systems, generally over $\mathbb{C}$. In particular, basic numerical algebraic geometry embodies probability one algorithms for computing all isolated solutions of a polynomial system as well as the numerical irreducible decomposition of an algebraic set, i.e., one or more points on each irreducible component in each dimension. More recently, numerical algebraic geometry has grown to include more advanced techniques which make use of the basic methods in order to compute data of interest in both real-world applications and pure algebraic geometry.

*Institute for Mathematics and its Applications, University of Minnesota, Minneapolis, MN 55122 (dbates1@nd.edu, www.nd.edu/~dbates1). This author was supported by the Duncan Chair of the University of Notre Dame, the University of Notre Dame, NSF grant DMS-0410047, the Arthur J. Schmitt Foundation, and the Institute for Mathematics and its Applications in Minneapolis (IMA).

†Department of Mathematics, University of Notre Dame, Notre Dame, IN 46556 (jhauenst@nd.edu, www.nd.edu/~jhauenst). This author was supported by the Duncan Chair of the University of Notre Dame, NSF grant DMS-0410047, and the Institute for Mathematics and its Applications in Minneapolis (IMA).

‡Department of Mathematics, University of Notre Dame, Notre Dame, IN 46556 (sommese@nd.edu, www.nd.edu/~sommese). This author was supported by the Duncan Chair of the University of Notre Dame, the University of Notre Dame, NSF grant DMS-0410047, and the Institute for Mathematics and its Applications in Minneapolis (IMA).

§General Motors Research and Development, Mail Code 480-106-359, 30500 Mound Road, Warren, MI 48090 (Charles.W.Wampler@gm.com, www.nd.edu/~cwample1). This author was supported by NSF grant DMS-0410047 and the Institute for Mathematics and its Applications in Minneapolis (IMA).

One of the key tools used in the algorithms of numerical algebraic geometry is homotopy continuation [1, 16], a method for finding all zero-dimensional solutions of a polynomial system. Given a polynomial system $f : \mathbb{C}^N \to \mathbb{C}^n$ to be solved by homotopy continuation, one first forms a polynomial system g that is related to f in a prescribed way but has known, or easily computable, solutions. The systems g and f are combined to form a homotopy, such as the linear homotopy $H(x,t) = f \cdot (1-t) + \gamma \cdot t \cdot g$ where $\gamma \in \mathbb{C}$ is randomly chosen. For a properly formed homotopy, there are continuous solution paths leading from the solutions of g to those of f which may be followed using predictor-corrector methods. Singular solutions cause numerical difficulties, so singular endgames [17, 18, 19] are typically employed. Zero-dimensional solving is discussed further in Section 2.2.

Numerical algebraic geometry treats both zero-dimensional (isolated) solutions and positive dimensional solution sets. The building blocks of the solution set of a set of equations are the irreducible components, i.e., the algebraic subsets of the solution set that consist of connected sets of points with neighborhoods biholomorphic to a neighborhood of a Euclidean space. The solution set breaks up into a union of a finite number of irreducible components, none of which is contained in the union of the remaining components. In numerical algebraic geometry, we associate to each component a witness set, which is the basic data structure used to numerically describe and manipulate positive dimensional solution sets. Given a multiplicity one irreducible component Z of a system of polynomials $f(x) = 0$, a witness set consists of a triple (f, L, W), where L is a random linear space of dimension complementary to that of Z and where W is a set of points such that $W := Z \cap L$. There is a slightly more involved definition for the case of components of multiplicity greater than one as described in [24].

Many of the algorithms of numerical algebraic geometry make abundant use of homotopy continuation. For example, the cascade algorithm [20], one of the basic methods involved in computing the numerical irreducible decomposition of an algebraic set, uses repeated applications of homotopy continuation at different dimensions in order to produce points on each component in each dimension. Monodromy and trace tests then lead to the complete numerical irreducible decomposition of the solution set of f. In the end, the user obtains from the methods of numerical algebraic geometry a wealth of information regarding the characteristics of the solution set of a given polynomial system, some of which may be difficult to procure by purely symbolic means. For good references on zero-dimensional solving see [13, 16] and for numerical algebraic geometry see [24].

There are several software packages available to the public which carry out some of the operations of numerical algebraic geometry. However, no one package contains all such capabilities. These packages include HOM4PS [6], PHoM [9], POLSYS [27], PHCpack [28], and HomLab [30]. The most recently released package, Bertini [2], is under ongoing develop-

ment by the authors. Although all software packages were developed at different times for different reasons, they share the goal of solving polynomial systems by numerical means.

The purpose of the present paper is two-fold. One purpose is to present a paradigm for software in the field of numerical algebraic geometry. The following section contains an elaboration on the aforementioned algorithms and an enumeration of the various operations required for carrying out those algorithms, broken into four levels. Implementation-specific details, such as data structures, are omitted. The other purpose is to provide a brief introduction to Bertini and indicate its partial fulfillment of the paradigm of Section 2. That is the content of Section 3, which is also broken up into four levels to mirror Section 2. The final section includes planned extensions of the Bertini software package.

2. A paradigm for numerical algebraic geometry software. All good software packages share several characteristics. In particular, good software should be reliable (i.e., it provides correct output with clear signals upon failure), as fast as possible with estimates of time remaining for large jobs, modular for easy modification, and user-friendly. In addition, good *numerical* software must be accurate and provide error estimates for all solutions. Reliability and accuracy generally have an adverse impact on speed, so while efficiency is important, it should not be emphasized over finding correct answers.

Numerical accuracy may be approached in two ways. One approach is to use a fixed level of precision and find as accurate a solution as possible, possibly using higher levels of precision for subsequent runs to attain more accuracy, if necessary. The other approach is to select an accuracy before the run and adjust the precision during the run to attain that accuracy. Either way, it has recently become generally accepted that it is important to have available multiple levels of precision when implementing numerical routines.

The purpose of this section is to provide an enumerated paradigm for software specifically in the field of numerical algebraic geometry. This detailed list is broken into four levels, beginning with very basic operations not specific to polynomial system solving at level 0 in Section 2.1 and moving through extensions of basic numerical algebraic geometry at level 3 in Section 2.4. The operations of each level build upon the capabilities of the previous level. Each of the following four sections begins with a discussion of the necessary operations of the given level and the resulting capabilities of the software. Each section then concludes with a brief list of the main operations to be implemented at that level. All operations should be implemented for various levels of precision, ideally for arbitrarily high precision.

2.1. Level 0: Basic operations. At the very core of a numerical polynomial solver, one must of course have access to basic arithmetic both

of complex numbers and complex matrices. It is important to optimize the efficiency of this arithmetic as much as possible, particularly in high precision, as most operations in numerical algebraic geometry rely heavily upon arithmetic. In addition to basic matrix arithmetic, standard techniques from numerical linear algebra are needed. Among the most important are Gaussian elimination for linear solving, QR factorization for least squares and orthogonal complements, and the SVD, or a related technique such as the efficient method of [14], for finding numerical ranks. See [5, 26] for general references on numerical linear algebra, while [14] provides a more efficient method for determining the numerical rank of a matrix.

Random numbers play a key role in numerical algebraic geometry as many statements hold generically, i.e., for almost all random choices, thereby making the resulting algorithms hold with probability one. Any standard random number generator will suffice, although it is best to have the chosen random complex numbers of approximately unit modulus, for stability. It is also important to have a consistent mechanism for extending the precision of a randomly chosen complex number. In particular, upon extending the precision of a random number α to make $\hat{\alpha}$, truncating back to lower precision, and then again extending the precision, one should once again obtain $\hat{\alpha}$.

It is of course necessary to somehow obtain the polynomials of interest from the user, although the specific procedure for doing so is implementation-specific. To build a general solver, it is important to allow the functions to be defined as expressions built from subexpressions. This is beneficial not only for ease of use, but also for efficiency and numerical stability. It is also necessary for generality to allow for homotopies that depend on parameters, including analytic expressions as discussed in more detail in the following section. If the user is specifying the entire homotopy, it is also necessary to have a way for the user to provide solutions to the start system g. Otherwise, the automatically generated start system should be solved by the software.

Regardless of how the input data is provided, parsed, and stored, it is at times necessary for the software to automatically homogenize the polynomials provided by the user. Homogenization is simply the mechanism for moving from a product of one or more complex spaces to a product of complex projective spaces. This is a purely symbolic operation which should be implemented in such a way as to be easily reversed in case the need arises. Suppose the homogenized system involves the cross product of v projective spaces. Then v random nonhomogeneous linear equations, one for each projective space in the product, should be appended to the system in order to choose a patch on each complex projective space. These v linear equations are known as the patch polynomials.

Basic path tracking, a level 1 operation, makes heavy use of both function evaluation and Jacobian evaluation. Function evaluation is straightforward, although it could be optimized via specialized techniques such as

Horner's method. The Jacobian (i.e., the matrix of partial derivatives of the polynomials) should be computed automatically by some form of automatic differentiation so that the user does not need to provide it as input. It is generally believed that the Jacobian should be computed explicitly rather than approximated numerically, for stability.

Some of the operations at this level are common to numerical software in general, regardless of the specific application. It should be noted that existing libraries, such as LAPACK, provide robust implementations of some of these operations, albeit in one level of precision.

Summary of level 0 operations:

- Complex scalar and matrix arithmetic
- Matrix operations (e.g., linear solving, numerical ranks, and orthogonal complements)
- Random number generation
- Parsing of input data
- Function homogenization
- Function evaluation
- Jacobian evaluation
- All numerical operations should be available in multiprecision.

2.2. Level 1: Basic tracking and zero-dimensional solving. It is now possible to glue together the capabilities of level 0 to build algorithms leading up to a full zero-dimensional solver, i.e., a method for computing all isolated solutions of f. The concept of "solving" a system could be interpreted two ways. Given a desired accuracy $\epsilon \in \mathbb{R}^+$, let $S := \{s \in \mathbb{C}^N : |f(s)| \leq \epsilon\}$. This set may break up into k disconnected pieces, say $S = S_1 \cup \ldots \cup S_k$. Then one interpretation of solving is to find a $z \in S_i$ for each i. The second way to interpret "solving" f is to find, for each $s \in \mathbb{C}^N$ such that $f(s) = 0$ and s has multiplicity m as a root of f, a set $\{z_1, ..., z_m\} \subset \mathbb{C}^N$ such that $|z_i - s| \leq \epsilon$ for all i. The solutions in the former sense of solving depend upon the scaling of the polynomials while those of the latter depend upon the scaling of the variable. The latter is more widely accepted as correct and is thus the definition used throughout this paper.

As discussed in Section 1, homotopy continuation casts the system f in a family of polynomial systems, another of which is g. Such parameterized families sometimes arise naturally and are of great interest in some applications. Other families are artificially constructed for the specific purpose of solving f by continuation, in a so-called *ab initio* homotopy. Often a naturally parameterized family can be used to define a homotopy that has fewer paths to follow than an *ab initio* homotopy. As a result, software in numerical algebraic geometry should address both *ab initio* and natural parameter homotopies. The natural parameter spaces may be complex analytic and thus the need for evaluating complex analytic expressions of parameters and complex numbers as mentioned in the previous section.

At its core, homotopy continuation is simply a sequence of steps advancing the path variable and updating the solution vector x to match. Each step forward consists of a predictor step followed by a sequence of corrector steps. The prediction is commonly carried out by Euler's method, which steps along the tangent to the path. This is generally regarded as an acceptable approach, although secant prediction can also be employed, as can higher order methods such as Runge-Kutta. Once a prediction is made to a new value of t, it is necessary to refine the point back towards the solution curve. This may be accomplished by fixing t and running a Newton-like corrector until the norm of the Newton residual has dropped below a prespecified tolerance.

Naturally, there are times when steps will fail, where failure is declared when the corrector does not converge sufficiently fast. Such step failures need not trigger path failure. Rather, adaptive steplength should be utilized. Upon step failure when using adaptive steplength, the steplength is decreased by a prespecified factor in the hope that convergence will occur for a smaller step. Only if progress along the path becomes excessively slow is the path declared a failure. Conversely, if the steps are progressing well, it is worthwhile to try to increase the steplength. More details regarding prediction, correction, and adaptive steplength may be found in [1, 16, 24].

Path failure may occur for a number of reasons, but the presence of a singularity, particularly at $t = 0$, is a common cause. There are several sophisticated algorithms known as endgames that help to speed convergence at $t = 0$ for both nonsingular and singular endpoints. These endgames are typically employed for every path, so it is important to have at least one implemented in any software package for numerical algebraic geometry. Details regarding endgames may be found in [17, 18, 19].

For zero-dimensional solving, polynomial systems given by the user could be nonsquare with $n > N$. Fortunately, Bertini's theorem [24] guarantees that a new system consisting of N generic linear combinations of the original n polynomials will have among its solutions all the isolated solutions of the original system, though possibly with increased multiplicity. It may also have nonsingular isolated extraneous solutions. The extraneous solutions are easily detected as they do not satisfy the original system.

Unless the user chooses to specify a parameter homotopy, the software must be able to automatically produce an appropriate start system g, solve it, and attach it to the homogeneous, square system f in order to create the homotopy H. There are several methods for producing start systems, although the general rule is that the computational cost increases in order to produce start systems with fewer paths to track. Among the common choices, total degree start systems, consisting of polynomials of the form $x_i^{d_i} - 1$, where d_i is the degree of the i^{th} polynomial in f, are trivial to build and solve but have the largest number of paths to be tracked (that being the product of the degrees of the functions). At the other end of the spectrum, the construction and solution of sophisticated polyhedral homo-

topies involve far more computation time but may result in far fewer paths (the number of which is the mixed volume). It is not clear *a priori* which type of start system is best-suited for an arbitrary polynomial system, so it is important to have multiple types of start systems available.

Once all (or most) of the aforementioned operations have been implemented, it is possible to compute the zero-dimensional solutions of a given polynomial system. Two other useful tools belong at this level. First, deflation [11] is a means of constructing a new polynomial system $\hat{f}$ from f such that $\hat{f}$ has a nonsingular solution in place of a particular singular solution of f. This makes it possible to compute singular solutions more accurately without relying on higher precision. The major drawback of implementing deflation is that decisions must be made about the rank of the Jacobian matrix at the solution point before the solution point is known accurately. The use of endgames can improve the accuracy of the solution estimate before deflation, helping to ensure that the correct deflation sequence is performed but adding the cost of endgame computations. Exploration of the numerical stability and efficiency of deflation and endgames is a topic of ongoing research. The big advantage of deflation comes when dealing with positive dimensional components of multiplicity greater than one.

The other useful tool at this level is a post-processor to manipulate and display the solutions computed by the solver as well as any statistics gathered during tracking. As the functionality of such a tool is application-specific, no more details will be discussed here.

Summary of level 1 operations:

- Differential equation solving, e.g., Euler's method
- Newton's method
- Basic path tracking with adaptive steplength control
- Adaptive precision path tracking
- Squaring of systems
- Start system and homotopy generation
- Start system solving
- Full zero-dimensional solving
- Endgames
- Deflation
- Post-processing of zero-dimensional data.

2.3. Level 2: Positive-dimensional solving. The solution set Z of a polynomial system f may have several components and these may not all have the same dimension. Letting $D := \dim Z$, the irreducible decomposition may be written as $Z = \cup_{i=0}^{D} Z_i = \cup_{i=0}^{D} \cup_{j \in \mathbb{I}_i} Z_{i,j}$, where each $Z_{i,j}$ is an irreducible component of dimension i, and accordingly each Z_i is the pure i-dimensional component of Z. (Symbol $\mathbb{I}_i$ in the above is just an index set for the components of dimension.)

One of the key objectives in numerical algebraic geometry is to find a numerical irreducible decomposition of Z, which consists of sets $W_{i,j} =$

$Z_{i,j} \cap L_{N-i}$, where L_{N-i} is a generic linear subspace of dimension $N-i$. $W_{i,j}$, together with L_{N-i}, is known as a witness set for $Z_{i,j}$, and by abuse of notation, $W_i = \cup_{j \in \mathbb{I}_i} W_{i,j}$ is called a witness set for Z_i. We briefly describe the algorithms for computing the irreducible decomposition below. Full details may be found in the references cited below or in [24].

The main steps in computing a numerical irreducible decomposition are:

- find witness supersets $\hat{W}_i \supset W_i$ for each dimension i,
- prune out "junk points" from $\hat{W}_i$ to extract the witness sets W_i, and
- break W_i into distinct sets $W_{i,j}$, the witness sets for the irreducible components $Z_{i,j}$.

The witness supersets $\hat{W}_i$ are generated by the application of zero-dimensional solving to find the isolated points in the slice $Z \cap L_{N-i}$. All of the $\hat{W}_i$, for $0 \leq i \leq D$, can be obtained using the cascade algorithm [20], starting at $i = D$ and cascading sequentially down to $i = 0$. The junk points in $\hat{W}_i$ must lie on some Z_j with $j > i$. Thus, the junk may be removed by testing each point $p \in \hat{W}_i$ for membership in a higher-dimensional component. This can be done using continuation on slices to see if any of the witness sets W_j, $j > i$ connect to p as the slicing linear space L_{N-i} is moved continuously until it contains p. The final break up of W_i into irreducibles is accomplished by first using monodromy, which comes down to discovering connections between witness points as the linear slicing space is moved around a closed loop in the associated Grassmannian [21]. This is followed by checking if the connected groups so discovered are complete, by means of the trace test [22]. The trace method may also be used to complete a partial decomposition. Both monodromy and the trace method involve specialized continuation of slices and careful bookkeeping.

Squaring is a concern for positive-dimensional solving just as it is for zero-dimensional solving. Although much carries over, one difference is that the size to which the system should be squared depends on the dimension of the component, e.g., for components of dimension k, the defining system should be randomized to a system of $N-k$ equations.

Given witness data for an algebraic set Z, there are three operations of particular interest for users. First, a user might want to find many points on a specific component. This is known as sampling and is very closely related to monodromy, as both use the continuation of slices to move witness points around on the component. Second, a user might want to know if a given point lies on an irreducible component of Z. This is the same component membership test used in the junk removal stage of computing an irreducible decomposition. Finally, a user might want the accuracy of some endpoint sharpened. This is just a matter of running Newton's method appropriately, perhaps after deflating f. Of course, as in

the previous section, a post-processor would be appropriate, although the exact functionality is again application-specific.

Deflation is particularly valuable as a method for multiple components, e.g., to track intersections of a multiplicity greater than one component with a one-parameter family of linear spaces of complementary dimension, as is done to sample a multiple component. Roughly speaking, if Z is a k-dimensional component of the solution set of a system $f = 0$, then the computation of the deflation of $Z \cap L$ for f restricted to a generic k-codimensional linear space gives rise to a "deflation" of the whole component. At the expense of increasing the number of variables, this allows us to numerically treat all components as multiplicity one components.

Summary of level 2 operations:

- Continuation of slices
- Monodromy
- Traces
- Squaring of systems for positive-dimensional solving
- Cascade algorithm
- Full numerical irreducible decomposition
- Sampling
- Component membership
- Endpoint sharpening
- Post-processing of witness data
- Deflation for components.

2.4. Level 3: Extensions and applications of the basics. This highest level consists of operations that make use of the basic numerical algebraic geometry maneuvers described in the previous three levels. For example, for two algebraic sets X and Y that are the solution sets of polynomial systems f and g, respectively, suppose one has witness sets W_X and W_Y for X and Y but would like a witness set W for $X \cap Y$. There is a now a method [23] for computing the numerical irreducible decomposition of such an intersection, the inclusion of which, while not essential for basic software in numerical algebraic geometry, will be important as the field continues to develop.

There are several other advanced algorithms that should be included in a complete state-of-the-art implementation. Another such technique is that of [25], which provides a way of finding exceptional sets, i.e., the sets of points in the parameter space above which the fiber has dimension higher than the generic fiber dimension. Fiber products play a key role in this algorithm and therefore need to be available in the software before the method for finding exceptional sets may be implemented. Also, in real-world applications, real solutions are often of more interest than complex solutions, so the extraction of real solutions from the numerical irreducible decomposition, for example, by an extension of the method of [15], would be very useful.

This is not intended to be a complete list, and it is anticipated that many more operations could be added in the near future. One capability, though, that is important now and will only become more essential over time, is parallelization. Although not every algorithm of numerical algebraic geometry is fully parallelizable, basic path tracking is easily parallelized so great savings can be made throughout levels 1, 2, and 3 by doing so [10, 12, 27, 29].

Summary of level 3 operations:

- Intersection of components
- Fiber products
- Finding exceptional sets
- Extracting real solution sets from complex components
- Parallelization.

3. Bertini. Bertini [2] is a new software package under ongoing development by the authors for computation in the field of numerical algebraic geometry. Bertini itself has evolved from a program called Polysolve created by Bates, C. Monico (Texas Tech University), Sommese and Wampler, although nothing substantial remains in Bertini from Polysolve.

Bertini is written in the C programming language and makes use of several specialized libraries, particularly lex and yacc for parsing and GMP and MPFR for multiple precision support. The beta version of Bertini was released in October 2006 to coincide with the Software for Algebraic Geometry workshop of the Institute for Mathematics and its Applications. It is currently anticipated that Bertini 1.0 will be made available to the public sometime in 2007. Bertini is currently only available as an executable file for 32- or 64-bit Linux and for Windows, via Cygwin. Specific instructions for using Bertini are included with the distribution and on the Bertini website.

Among other things, Bertini is capable of producing all complex isolated solutions of a given polynomial and witness sets for each positive-dimensional irreducible component. The goal of the Bertini development team is to eventually include in Bertini all operations described above in Sections 2.1 through 2.4. The purpose of this section is to indicate briefly which of those operations are already available in the beta version of Bertini. The specific algorithms implemented are also described, when appropriate. Details regarding the development plans for Bertini 1.0 and beyond may be found in Sections 4.1 and 4.2, respectively.

3.1. Level 0. By default, Bertini uses IEEE double precision, although it also allows for any fixed level of precision available in MPFR. In particular, precision is available starting from 64 bits, increasing in 32 bit increments. Furthermore, the beta version of Bertini allows the user to select adaptive precision for zero-dimensional solving. In adaptive precision mode, Bertini begins in double precision and increases precision as necessary, determined by the algorithm described in [3].