Recovering exact results from inexact numerical data in algebraic geometry

Daniel J. Bates * Jonathan D. Hauenstein†
Timothy M. McCoy‡ Chris Peterson§ Andrew J. Sommese¶

Abstract

Let $\{f_1, f_2, \ldots, f_t\} \subset \mathbb{Q}[z_1, \ldots, z_N]$ be a set of homogeneous polynomials. Let Z denote the complex, projective, algebraic set determined by the homogeneous ideal $I = (f_1, f_2, \ldots, f_t) \subset \mathbb{C}[z_1, \ldots, z_N]$. Numerical continuation-based methods can be used to produce arbitrary precision numerical approximations of generic points on each irreducible component of Z. Consider the prime decomposition $\sqrt{I} = \bigcap_i P_i$ over $\mathbb{Q}[z_1, \ldots, z_N]$. In this article, it is shown that these approximated generic points may be used in an effective manner to extract exact elements $G_{i,j} \in \mathbb{Z}[z_1, \ldots, z_N]$ from each P_i. A collection of examples and applications serve to illustrate the approach.

Keywords. witness set, witness superset, generic points, homotopy continuation, irreducible components, multiplicity, numerical algebraic geometry, polynomial system, numerical irreducible decomposition, primary decomposition, algebraic set, algebraic variety, LLL

AMS Subject Classification. 65H10, 68W30, 14Q99

*Colorado State University, Fort Collins, CO 80523 (bates@math.colostate.edu, http://www.math.colostate.edu/∼bates). This author was supported by NSF grant DMS-0914674

†Department of Mathematics, Texas A&M University (jhauenst@math.tamu.edu, `www.math.tamu.edu/∼jhauenst`). This author was supported by NSF grant DMS-0915211.

‡Department of Mathematics, University of Notre Dame, Notre Dame, IN 46556 (tmccoy@nd.edu, `www.nd.edu/∼tmccoy`).

§Colorado State University, Fort Collins, CO 80523 (peterson@math.colostate.edu, http://www.math.colostate.edu/∼peterson). This author was supported by NSF grant DMS-0901770.

¶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, NSF grant DMS-0410047, and NSF grant DMS-0712910

Introduction

Recent progress on homotopy based methods in the context of polynomial systems has expanded the arsenal of tools that can be brought to bear in computational algebraic geometry. A potential shortcoming of the output provided by numerical tools, in relation to symbolic methods, is a loss of exactness. Despite this feature, numerical approaches are still favored in several contexts due, in part, to their ability to provide meaningful output when working with dozens or even hundreds of variables, well beyond the reach of most non-numerical methods. It has been the common goal of a number of researchers to develop algorithms which combine the best properties of numerical methods (parallelizability, ability to work with inexact coefficients, slower complexity growth, ability to work with a large number of variables) with the best properties of symbolic methods (exact results, straightforward certification, wide flexibility in algebraic structures). This has led to a number of hybrid numeric/symbolic algorithms which have extended the arena of applicability of computational algebraic geometry. In this paper, we focus our attention on a hybrid numeric/symbolic algorithm which uses the inexact data provided by numerical homotopy methods to derive exact results through lattice basis reduction techniques. This exact data can then be post-processed and certified using symbolic methods.

Let I be a homogeneous ideal in $\mathbb{C}[z_1, \ldots, z_N]$, let Z be the associated complex projective algebraic variety, and let S be the associated complex projective scheme. Through numerical homotopy continuation methods, combined with monodromy breakup, it is practical to produce sets of numerical data points (of prescribed precision) which are in one to one correspondence with the irreducible components of Z [31, 34]. More precisely, for each irreducible component V of Z, one can determine a set of numerical data points that approximate a set of general points on V with a prescribed precision. The set of numerical data points which corresponds to a given irreducible component of Z is called a *witness set* for the component. It is interesting to note that with further computational effort, it is sometimes feasible to produce witness sets in correspondence with the irreducible components of the scheme S [24]. The examples section of this paper will touch lightly on properties of S. However, the primary focus of this paper will concern the algebraic variety Z and its decomposition into irreducible components. In particular, let $Z = \bigcup_d Z_d$ denote the decomposition of Z into pure d-dimensional components and let $Z_d = \bigcup_j Z_{d,j}$ denote the decomposition of Z_d (the d-dimensional component of Z) into irreducible d-dimensional varieties. A *full witness set* for Z is a collection of witness point sets $W_{d,j}$,

one set for each $Z_{d,j}$, with $W_{d,j}$ denoting a non-empty set of numerically approximated generic points on $Z_{d,j}$. It is a key observation that each of these sets may be expanded to include an arbitrary number of points approximating generic points on the corresponding component and that each of these approximations may be computed to arbitrarily high precision [7]. Please refer to §1.1 or [34] for further details.

Let $F = \{f_1, \ldots, f_t\} \subset \mathbb{C}[z_1, \ldots, z_N]$ be a finite collection of homogeneous polynomials and let $I(F) \subset \mathbb{C}[z_1, \ldots, z_N]$ denote the homogeneous ideal they generate. Let $Z = V(I(F))$ denote the associated complex projective algebraic variety. Suppose further that $\{f_1, \ldots, f_t\} \subset \mathbb{Q}[z_1, \ldots, z_N]$ (i.e. that the set of generators of $I(F)$ all have rational coefficients). A goal of this paper is to use the full witness set for Z to compute exact elements in each component of the prime decomposition of $\sqrt{I}$ over $\mathbb{Q}[z_1, \ldots, z_N]$. More precisely, we describe a hybrid numeric/symbolic algorithm with the following input and output:

INPUT: A finite subset $F = \{f_1, f_2, \ldots, f_t\} \subset \mathbb{Q}[z_1, \ldots, z_N]$.

OUTPUT: A collection of finite sets $G_1, G_2, \ldots, G_k$ such that

- For each i, $G_i \subset \mathbb{Q}[z_1, \ldots, z_N]$.
- For each $i \neq j$, no associated prime of $\sqrt{I(G_i)}$ is contained in an associated prime of $\sqrt{I(G_j)}$.
- For each i, the scheme defined by $I(G_i)$ is generically smooth and the variety defined by $I(G_i)$ is equidimensional.
- $\bigcap_i \sqrt{I(G_i)} = \sqrt{I(F)}$.

As described in §2, the roots of this algorithm are found in lattice basis reduction techniques. In particular, given a set of vectors $V = \{V_1, \ldots, V_t\} \in \mathbb{C}^r$, we will use a lattice basis reduction algorithm to find linearly independent vectors $W_1, \ldots, W_s \in \mathbb{Z}^r$ which are *nearly orthogonal* to every vector in V. It should be noted that the use of lattice basis reduction algorithms, to find integer relations on a set of vectors, is not new. For instance, such algorithms have been applied to diverse problems in cryptography, number theory, and integer linear programming. However, the ability to approximate generic points to arbitrary precision, for every irreducible component of an algebraic set Z, is a recent development. As a consequence, the input and the corresponding value of the output is different than in previous considerations of related problems. In particular, the applications of the

fundamental algorithm of this article, described in §3, have not previously been considered in the literature in the setting of adaptive multiprecision for generic points on algebraic varieties.

The following section, §1, provides a brief introduction both to numerical algebraic geometry and to what we call exactness recovery algorithms, such as LLL. In §2 there is a description of the algorithm, a discussion of several choices available for subalgorithms, remarks on computational complexity, and remarks on an initial, Maple14 based implementation with numerical data supplied by *Bertini* [6]. Within Maple 14 there is an implementation of the LLL algorithm and basic Gröbner basis commands. For more extensive Gröbner basis calculations, we have utilized programs such as *Macaulay2*, *Singular*, and *CoCoA* [18, 19, 10]. In §3, we describe modifications of the fundamental algorithm to solve problems in other settings, including elimination theory, the desingularization of curves, and the construction of vanishing ideals for join and secant varieties. Finally, in §4, we conclude the paper and summarize the results.

We would like to thank a number of individuals and organizations for various forms of support during this project, aside from those listed on the first page. In particular, we appreciate the useful comments and sound advice of H. Hong, E. Kaltofen, M. Singer, S. Sullivant, and A. Szänto during a visit by the first author to North Carolina State University. The first, second, fourth, and fifth authors would also like to express their gratitude to Doug Arnold and the organizers of the IMA (Institute for Mathematics and Its Applications) Thematic Year on *Applications of Algebraic Geometry*. The final draft of the paper was completed at the Spring 2011 thematic semester of the Mittag-Leffler Institute on *Algebraic Geometry with a view towards applications*.

1 Background

The algorithms described in Sections 2 and 3 make use of two types of computation: *numerical computation* in the context of tracking paths while considering homotopies between polynomial systems and *symbolic computation* in the context of finding a special almost orthogonal basis for a lattice. The symbolic phase of the computation is used to determine linear forms with integer coefficients which approximately vanish at prescribed numerically approximated points arising as output of the numerical phase of the computation. The point of this section is to give brief introductions to numerical homotopies between polynomials systems and to lattice basis reduction al-

gorithms along with appropriate references for the interested reader.

1.1 Numerical algebraic geometry

The term *numerical algebraic geometry* is often used to describe the set of numerical methods used to extract algebraic and geometric information from polynomial systems. The field is growing rapidly and now includes a wide range of algorithms (both numeric and numeric-symbolic). The class of numerical algorithms that are extensively used in this paper are rooted in homotopy continuation. In the context of numerical algebraic geometry, the idea of homotopy continuation is to link a pair of polynomial systems through a deformation and to relate features of the two systems through this deformation. For instance, one can track known isolated, complex solutions of one polynomial system to unknown, complex solutions of a second polynomial system through a deformation of system parameters. Please refer to [1, 25, 34] for further details.

The core algorithm of homotopy continuation can be combined with monodromy breakup to produce the witness sets described in the Introduction. This involves reducing positive-dimensional irreducible components to the zero dimensional case by slicing with a linear space of complementary dimension. In essence, a flag of linear spaces is used in a cascade algorithm to produce witness points for each irreducible component. As a byproduct, the algorithm also determines the dimension and degree of each irreducible component [31]. A witness point further serves as a numerical approximation to a generic point on the component that it labels. A set of non-empty witness sets for a variety (one witness set for each irreducible component), together with an identification of the degree and dimension of each component, is called a *numerical irreducible decomposition* of the variety. Parts of this decomposition are unique (the degree and dimension of each irreducible component) while others are not (the specific witness point set for a given positive dimensional component).

There are several important features of the methods of numerical algebraic geometry that should be remembered later in this article. The first feature is the ability to refine witness points to arbitrarily high precision via Newton's method. A second feature is the ability to produce an arbitrary number of witness points on any given component. A third feature is the parallelizability of these numerical methods. For instance, 1000 processors could be used in parallel to track 1000 paths and could be used in parallel to refine the accuracy of each witness point to arbitrarily high precision.

1.2 Algorithms for exactness recovery

Given a subspace W of a vector space V, a fundamental problem in linear algebra is to find an orthogonal basis for W. A related (but harder) problem is the following:

Problem 1.1 *Let $w_1, w_2, \ldots, w_t \in \mathbb{C}^k$ and let ϵ, B be prescribed positive real numbers. Find a maximal linearly independent set of vectors $v_1, v_2, \ldots, v_s \in \mathbb{Z}^k$ such that $||v_i|| < B$ for all i and $|v_i \cdot w_j| < \epsilon$ for all i, j.*

This problem gained particular prominence in the 1970s and 1980s and led to a substantial literature on *lattice basis methods* including the well-known LLL algorithm [23]. Much was written on lattice based methods during the ensuing years, and several other algorithms have been discovered. Since the goal of this paper is the application of such methods rather than the development of a new algorithm of this type, we will restrict our attention to the LLL algorithm as described in [23, 17]. It is worth noting that there are improvements to the LLL algorithm [26, 28, 27], as well as alternatives (such as BKZ and PSLQ [16, 30]), but a full analysis and description of the various options would not significantly enhance the value of this article.

As a component of our main algorithm, we will be interested in an LLL-reduced lattice basis for a lattice $L \in \mathbb{R}^k$. With respect to the data that we will need, the following are two of the most relevant properties of the LLL algorithm:

1. Given an input basis $B = b_1, b_2, \ldots, b_t$ for a lattice $L \in \mathbb{R}^k$, the LLL algorithm will output an LLL-reduced lattice basis $B^* = b_1^*, \ldots, b_t^*$ for L in polynomial time.

2. The length of b_1^* is no larger than $c_1 \cdot M$ where M is the length of the shortest vector in the lattice and c_1 is a known factor. Similarly, b_2^* is within a known factor, c_2, of the second shortest vector, etc.

For more details on the LLL algorithm and some applications, please see [11, 23].

2 Fundamental algorithm

The core algorithm of the paper is provided in this section. An initial, conceptual discussion of the algorithm, along with definitions needed for the algorithm, is given in §2.1. The technical form of the algorithm is given in

§2.2, while an illustrative example and implementation details are provided in §2.3 and §2.4, respectively. Finally, brief discussions of symbolic verification of the results of this numeric-symbolic method and a few words about complexity are provided in §2.5.

2.1 Conceptual discussion

Let $\{f_1, f_2, \ldots, f_t\} \subset \mathbb{Q}[z_1, \ldots, z_N]$ be a set of homogeneous polynomials with rational coefficients. Consider the ideal $J = (f_1, f_2, \ldots, f_t) \subset \mathbb{C}[z_1, \ldots, z_N]$ generated in the ring of polynomials with complex coefficients. There exist homogenous polynomials $\{g_1, g_2, \ldots, g_s\} \subset \mathbb{Q}[z_1, \ldots, z_N]$ such that $\sqrt{J} = (g_1, g_2, \ldots, g_s) \subset \mathbb{C}[z_1, \ldots, z_N]$. Let $\sqrt{J}_{\mathbb{Q}} = (g_1, g_2, \ldots, g_s) \subset \mathbb{Q}[z_1, \ldots, z_N]$. Consider the prime decomposition $\sqrt{J}_{\mathbb{Q}} = \bigcap_{\ell=1}^{r} P_{\mathbb{Q},\ell}$ of $\sqrt{J}_{\mathbb{Q}}$ in $\mathbb{Q}[z_1, \ldots, z_N]$. Note that $P_{\mathbb{Q},\ell}$ is prime when considered as an ideal in $\mathbb{Q}[z_1, \ldots, z_N]$ and has a set of generators, $G_\ell \subset \mathbb{Q}[z_1, \ldots, z_N]$. However, the ideal $P_{\mathbb{C},\ell}$, generated by G_ℓ in $\mathbb{C}[z_1, \ldots, z_N]$, may fail to be prime. A simple example of such phenomenon is the ideal $L_{\mathbb{Q}} = (z^2 + 1) \subset \mathbb{Q}[z]$ compared to the ideal $L_{\mathbb{C}} = (z^2 + 1) \subset \mathbb{C}[z]$. $L_{\mathbb{Q}}$ is prime but $L_{\mathbb{C}}$ is not prime.

Let $V(J)$ denote the complex algebraic variety determined by J. Recall that a witness set for $V(J)$ consists of a list of sets $W_1, \ldots, W_t$ corresponding to the irreducible components $V_1, \ldots, V_t$ of $V(J)$ and with the elements in W_i consisting of numerically approximated generic points on the corresponding irreducible component V_i. The goal of Algorithm 1 (see Section 2.2) is to use the LLL algorithm, and a high accuracy witness set for $V(J)$, to obtain sets of polynomials, lying in $\mathbb{Q}[z_1, \ldots, z_N]$, that generate $P_{\mathbb{Q},\ell}$ for each ℓ. In other words, our goal is to compute a prime decomposition of $\sqrt{J}_{\mathbb{Q}}$ from generic points on each irreducible component of $V(J)$. In general, we do not accomplish this goal using solely Algorithm 1. However, the data that we do produce (relying heavily on numerical homotopy algorithms and the LLL algorithm) makes significant progress towards computing a prime decomposition of $\sqrt{J}_{\mathbb{Q}}$. Furthermore, this data can sometimes be utilized by a Gröbner basis algorithm to complete the prime decomposition in a setting where neither Gröbner basis techniques alone nor Algorithm 1 alone could complete the decomposition. This combination of numerical homotopy methods for determination and refinement of generic points, lattice basis reduction for recovery of exactness, and Gröbner basis algorithms for final processing appears to be novel and effective.

As a starting point for the data fed to the LLL portion of the algorithm, suppose we know an upper bound D on the degree of a set of minimal generators of P_ℓ (for some particular value of ℓ), and that we have computed the

generic points in a corresponding witness set to an adequate level of precision given D and the expected size of the coefficients of the generators. While the complex variety determined by P_ℓ may have several irreducible components, this set of components is a subset of the set of irreducible components of $V(J)$. We do not know, *a priori*, how many irreducible components of $V(J)$ comprise $V(P_\ell)$. However, we do know that the homogeneous ideals of any two irreducible components of $V(P_\ell)$ will have identical Hilbert functions.

The algorithm proceeds as follows: let $\hat{x}$ be a numerical approximation, of prescribed accuracy, to a general point x on an irreducible component, V_i of $V(J)$. This corresponds to a generic point on an irreducible component of the complex variety $V(P_\ell)$ for some ℓ. For each degree $d \leq D$, find a set K_d of linearly independent, degree d polynomial relations with integral coefficients using an exactness recovery algorithm (in our case, we use LLL). Let Q_1 be a set of minimal generators, with integral coefficients, for the ideal generated by $\bigcup_{i \leq D} K_d$. This finite set can be determined using exact linear algebra in a straightforward manner. If *a priori* bounds are known on the degree of a set of minimal generators of P_ℓ and on the size of the coefficients in the generators, then the ideal generated by Q_1 is expected to be equal to P_ℓ. One can use numerical algebraic geometry to determine if $V(Q_1)$ is generically reduced, if $V(Q_1)$ contains V_i as an irreducible component, if the homogeneous ideals of the irreducible components of $V(Q_1)$ all have the same Hilbert function, and if the irreducible components of $V(Q_1)$ are a subset of the irreducible components of $V(J)$. Thus it can be determined which of the irreducible components of $V(J)$ correspond to irreducible components of $V(Q_1)$ and they can be removed from further consideration. Now repeat the procedure with a generic point on an irreducible component of $V(J)$ that has yet to be considered. An iteration of this procedure will exhaust the list of irreducible components of $V(J)$ leading to a collection of sets of polynomials, $Q_1, \ldots, Q_t$ with integer coefficients. Furthermore, by the manner in which they were constructed, $\sqrt{J}_{\mathbb{Q}} = \bigcap_{i=1}^{t} \sqrt{(Q_i)}$. Without the aid of Gröbner basis techniques, we have difficulty determining if each ideal (Q_i) is saturated and free of *non-distinguished embedded components* [15]. Additionally, if $V(Q_i)$ is reducible as a complex variety, we have difficulty verifying that (Q_i) is a prime ideal in the ring $\mathbb{Q}[z_1, \ldots, z_N]$. However, if $V(Q_i)$ is smooth and irreducible as a complex variety, then the techniques of numerical linear algebra can be used to verify that the *saturation* of Q_i (with respect to the irrelevant ideal) is prime.

Let $\mathcal{M}$ denote the set of all monomial exponents of a particular degree d. The core of the method is to detect homogeneous polynomials $F(z_0, \ldots, z_N) = \sum_{\alpha \in \mathcal{M}} c_\alpha z^\alpha \in \mathbb{Z}[z_0, \ldots, z_N]$ which vanish at x by using

the LLL algorithm to find an LLL reduced lattice basis for a lattice built from $\hat{x}$. To do so, we first define the standard *degree d Veronese embedding* of a point $p \in \mathbb{P}^N$. This is the map $v_d^N : \mathbb{P}^N \to \mathbb{P}^{\binom{N+d}{N}-1}$ sending a point p to the tuple of all degree d monomials, each evaluated at p. By utilizing this map and invoking continuity, the problem of finding homogeneous polynomials of degree d with integer coefficients that vanish at x is embedded in the problem of finding integer vectors whose dot product with $v_d^N(\hat{x})$ is very small. LLL and its variants are well-suited to finding such integer vectors. To apply LLL, we first build a lattice whose basis is the columns of a particular $(\binom{N+d}{N}+1) \times \binom{N+d}{N}$ matrix A. The matrix A is built by stacking an $\binom{N+d}{N} \times \binom{N+d}{N}$ identity matrix on the vector $C \cdot v_d^N(\hat{x})$ where C is a large number chosen by the user. One then finds an LLL reduced basis, $b_1^*, \ldots, b_{\binom{N+d}{N}}^*$, from the lattice basis generated by the columns of A. This is an ordered basis of vectors of length $\binom{N+d}{N}+1$ whose first $\binom{N+d}{N}$ entries will be integers and whose final entry will be a real number typically of small to moderate size. Let L be a vector in the lattice. The dot product of the first $\binom{N+d}{N}$ entries of L with the vector $C \cdot v_d^N(\hat{x})$ is the real number appearing as the final entry in L. If the final entry in L is small then the first $\binom{N+d}{N}$ entries of L correspond to an integer vector which is almost orthogonal to $C \cdot v_d^N(\hat{x})$. This integer vector is the coefficient vector of a degree d polynomial. If there exist small integer vectors that are nearly orthogonal to the vector $C \cdot v_d^N(\hat{x})$, then these tend to appear in the span of the earlier vectors in the LLL-reduced basis. In general, the earlier entries of the LLL-reduced basis tend to have relatively small entries followed by a dramatic jump in entry size. The vectors that are of interest are the ones whose entries are relatively small. A key point that drives the computation is that if a polynomial F evaluates to zero at x and if $\hat{x}$ is very close to x then F evaluated at $\hat{x}$ will be small [23]. There is a growing collection of polynomial time algorithms which may also be used and which may demonstrate increased efficiency in some situations. However, an exploration and comparison of these alternative algorithms goes beyond the scope of this paper, see [20].

By fixing the required input precision and the maximum degree of relations, it is clear that the above process terminates. It should be noted that upper bounds do exist on the largest degree generator of P_ℓ given the degree of the variety $V(P_\ell)$, see [9], but these tend to be impractically large. To make matters worse, bounds on the coefficient size may not be known, and hence the precision requirements may be ambiguous. While these issues can be mitigated, as discussed in §2.4, they cannot at present be eliminated.

2.2 The fundamental algorithm

Let J, $\hat{x}$, D, and $v_d^N(\cdot)$ be as in the previous section. We assume for the moment that $\hat{x}$ has an adequate level of accuracy, as mentioned in §2.1. For each degree d from 1 to D, we store in K_d all integer relations of degree d discovered by feeding the degree d Veronese embedding of $\hat{x}$ into the LLL algorithm. We then store in $G(\hat{x})$ a subset of the elements in $\bigcup_{i \leq D} K_d$ that minimally generate the ideal $I(\bigcup_{i \leq D} K_d)$. Thus, at the end of the algorithm, we have collected in $G(\hat{x})$ a set of independent integer polynomials which "almost" vanish at $\hat{x}$.

The following pseudocode describes the LLL component of the algorithm: the computation of $G(\hat{x})$ from a single witness point $\hat{x}$.

Algorithm 1 Basic algorithm

Input: A positive integer D and a floating point approximation $\hat{x}$ of a general point x on an irreducible component, V_i, of $V(J)$.
Output: $G(\hat{x})$.

```
K := {}
G(x̂) := {}
for d = 1 to D do
   Compute v_d^N(x̂)
   Compute integer relations on v_d^N(x̂)
   K ← {relations on v_d^N(x̂^ℓ)}
end for
Compute a minimal generating set for (K)
G(x̂) ← {minimal generating set for (K)}
```

As mentioned earlier, one can use numerical algebraic geometry to determine if $V(G(\hat{x}))$ is generically reduced, if $V(G(\hat{x}))$ contains V_i as an irreducible component, if the homogeneous ideals of the irreducible components of $V(G(\hat{x}))$ all have the same Hilbert function, and if the irreducible components of $V(G(\hat{x}))$ are a subset of the irreducible components of $V(J)$. Thus it can be determined which of the irreducible components of $V(J)$ correspond to irreducible components of $V(G(\hat{x}))$ and they can be removed from further consideration. The algorithm can then iterate on a generic point of an irreducible component of $V(J)$ that has not yet been considered.

2.3 An illustrative example

As an illustrative example, consider the twisted cubic curve realized as the image of the Veronese map $v_3^1 : \mathbb{P}^1 \hookrightarrow \mathbb{P}^3$ where $v_3^1([s : t]) = [s^3 : s^2t : st^2 : t^3]$. It is simple to produce high accuracy approximations of generic points on this curve by choosing random floating point pairs and considering their image via the Veronese map. Given these, we can run the algorithm to recover the defining equations of the twisted cubic.

Running the algorithm with a witness point (numerical approximation of a general point) computed to 200 digits and $D = 4$, Algorithm 1 provides the following polynomials in the ideal of the twisted cubic curve:

Component 1:	$-zw + yx$
	$-yw + x^2$
	$-zx + y^2$

In fact, the algorithm found no polynomials of degree 1, three of degree 2, and many of degrees 3 and 4. The polynomials of degrees 3 and 4 were determined to be dependent on the three degree 2 polynomials.

The polynomials above are known to be the defining equations of the twisted cubic. What we can check without a Gröbner basis computation is that each of the generators of the ideal vanish identically on any point in the image of v_3^1 and that the scheme defined by the ideal is smooth and irreducible. From this we can conclude that the ideal is correct up to saturation with the irrelevant ideal. The final post-processing uses Gröbner bases to check that the ideal is saturated. The ideal we have computed can be found symbolically by forming the ideal $J = (w - s^3, x - s^2t, y - st^2, z - t^3) \subset \mathbb{Q}[s, t, w, x, y, z]$ then computing the elimination ideal $J \cap \mathbb{Q}[w, x, y, z]$. By explicitly choosing a generic point in our method, we are in fact choosing generic values for s and t, and then we only implicitly look for generators of the elimination ideal without having to actually consider the variables s and t. We will see a similar trend in other problems that can be solved with elimination ideals.

2.4 Implementation details

There are many choices to be made when implementing this algorithm. The goal of this section is to describe the choices made by the authors, though it is an active research direction to evaluate these choices.

Computation of $\hat{x}$
The creation of the input data is well understood. In particular, the computation of $\hat{x}$ to arbitrarily high accuracy, from an ideal, is now relatively simple, for example, using the software package *Bertini* [6]. Please refer to §1.1 for further details. If the computation involves the image of a map, as in the previous example, one may obtain approximations to generic points in other ways.

Choice of exactness recovery algorithm and software
As described previously, there are many choices for exactness recovery algorithms, and many more choices of implementations. As the authors of this article developed the associated software in Maple, the Maple implementation of the LLL algorithm was used. There is a significant pool of literature comparing different implementations of the LLL algorithm and comparing various alternatives. There is little value in repeating the various benefits and shortfalls of each algorithm and each implementation.

Checking independence of a set of generators for an ideal
To check whether a polynomial f is independent of a set of polynomials K (i.e., that f is not in the ideal generated by the polynomials in K), one obvious option is a Gröbner basis computation, i.e., the standard symbolic ideal membership test. It may seem strange to perform a set of Gröbner basis computations as part of an algorithm which may be used in place of a single Gröbner basis computation. However, these Gröbner basis computations are typically simpler than the direct symbolic computation of the prime decomposition of J.

Another option for checking whether f is not in the ideal generated by the polynomials in K is to expand all polynomials in K to the degree of f (by multiplying each polynomial by all monomials of the appropriate degree), write each polynomial as a vector of coefficients, and perform a rank computation. We have found that this method is much less computationally expensive than using Gröbner bases. Further refinements and replacements to this method form a current stream of research.

Precision requirements
Another potential issue is the necessity of high precision. Even if we know (or assume) bounds for both coefficient size and degrees of relations, the necessary precision requirements can be very large. This is due primarily to the length of the vectors representing the images of points under the Veronese embedding; while a point in $\mathbb{P}^n$ is represented by a vector of length

$n+1$, the image of a point in $\mathbb{P}^n$ under the degree d embedding is represented by a vector of length $\binom{n+d}{d}$. To find a reliable relation with coefficients of up to k digits, we require a minimum of $\binom{n+d}{d}k$ digits of precision. For example, in the case $n = 4$, a degree 6 relation with up to 3-digit coefficients will require at least 627 digits of precision. It is worth noting that our precision requirement is polynomial with respect to d given a fixed n, and likewise is polynomial with respect to n given a fixed d.

The higher the precision we use, the longer the computations will take. If generic points are being computed with homotopy continuation methods, then computing high precision approximations is not a major issue. Paths can typically be tracked in lower precision to arrive at a low precision approximation; sharpening is then carried out with Newton's method which tends to be computationally inexpensive. The main slowdown occurs in the computation of integer relations. Several optimizations are available for LLL or PSLQ. For example both can be adjusted to use machine precision (64-bit) in much of the intermediate work. This and several other technical improvements for LLL may be found in [29], additional heuristic improvements are given in [2]. A low working precision variation of PSLQ is covered in [3], along with a parallelizable version of the algorithm.

Working without degree bounds or precision requirements
The algorithm presented in §2.2 requires that $\hat{x}$ be provided with *adequate* precision and that a bound D on the degrees of a set of minimal generators is known. However, the level of precision needed is practically impossible to know *a priori*, and the only general degree bounds are impractically high. However, there is a clear meta-algorithm that has a stopping criterion not depending on *a priori* knowledge of precision needs or degree bounds. Let V_i be an irreducible component of $V(J)$. Suppose that V_i is smooth and that it is the unique component of $V(J)$ with a given Hilbert function. Then the homogeneous ideal of V_i will have a minimal set of generators with integer coefficients. Given a witness point $\hat{x}$ for V_i, choose a bound D and some level of precision, then run Algorithm 1. Let $G(\hat{x})$ denote the output of the algorithm. It can be checked (with numerical homotopy methods) whether the resulting ideal, $(G(\hat{x}))$, defines a smooth complex scheme whose dimension and degree is the same as the complex variety V_i. If not, then the computation can be repeated with a higher precision and/or a higher bound D. In addition, one can determine other witness points on V_i and refine them to very high accuracy. If each element of $G(\hat{x})$, when evaluated at these points, is very small then with very high probability, $(G(\hat{x}))$ is

equal to the saturation of $I(V_i)$. If one of these tests fail (for instance if $G(\hat{x})$ does not define a smooth scheme with the same invariants as V_i or if the elements of $G(\hat{x})$ do not numerically evaluate to zero at high precision witness points of V_i) then one can refine $\hat{x}$, increase the value of D and rerun Algorithm 1. Note that problems can arise if the irreducible component V_i of $V(J)$ is not smooth. For instance, there could be embedded components, at the singular points of V_i, in the scheme defined by $G(\hat{x})$ which happen to be non-distinguished. With present numerical homotopy techniques, these are difficult to detect. One, potentially expensive, work around is to use Gröbner basis methods to determine if the ideal $(G(\hat{x}))$ is radical and/or saturated. It is important to note that in many problems, $G(\hat{x})$ will be much more manageable than the original system from the vantage point of Gröbner basis computations. An additional problem can arise if one of the ideals P_ℓ is not prime. This is an interesting problem that is beyond the scope of the present paper.

A thorough complexity analysis of the algorithms involved with this procedure would indicate whether it is favorable to increase D or increase the accuracy of $\hat{x}$. Since the generators of P_ℓ have a maximum degree and since all generators will be recovered if adequately high accuracy is used, there is at least a theoretical guarantee that the algorithm will terminate.

Irreducibility

On a final note, in finding integer relations we are intrinsically restricting ourselves to finding relations over $\mathbb{Q}$, whereas numerical homotopy methods naturally work over $\mathbb{C}$. One consequence is that the output witness point data is complex, which introduces some redundancy when we look for integer relations. In particular, any integer relation on a complex vector is simultaneously a relation on both its real and imaginary parts. We can take advantage of this fact to reduce the amount of data passed to the integer relation algorithm. For each degree, instead of using the Veronese embedding of the complex-valued point directly, we use a random linear combination of its real and imaginary parts. Having a random linear combination ensures that, with probability one, any relations found are relations on both the real and imaginary parts independently, and hence gives a polynomial relation on the original complex point.

An interesting related point is the ability to compute the intersection of ideals by considering general linear combinations of compatible Veronese embeddings of high accuracy witness points on different components. For instance, suppose $V(J)$ is a variety in $\mathbb{P}^N$ with r irreducible components. If $\hat{x_1}, \ldots, \hat{x_r}$ are witness points for each of the irreducible components of

$V(J)$ then the algorithm can be run on general linear combinations of $v_d^N(\hat{x_1}), \ldots, v_d^N(\hat{x_r})$ for each d. The output will consist of integral polynomials vanishing on every component of $V(J)$ thus recovering elements in $\sqrt{J}$.

Proof-of-concept software for this method, in the form of several Maple scripts, is currently available to the public from the website of the first author. Several additional examples, not found in this paper, are also available at the website as Maple scripts.

2.5 Complexity

Complexity is a difficult issue to address. First, the complexity of the computation of the numerical irreducible decomposition of a variety is unknown. There are some back-of-the-envelope estimates in the literature, but there is no solid foundation that allows for a careful complexity analysis at this point. Second, the use of Gröbner basis methods as an intermediate step in the algorithm will add significantly to the total computational cost of the method. The use of characteristic sets or triangular sets may reduce this cost, as might the use of numerical rank-finding methods. Due to the unknowns surrounding complexity estiates, we treat such issues as beyond the scope of the current paper.

3 Applications

3.1 Prime Ideal Decomposition

One of the more straightforward uses of the algorithm is to compute the prime decomposition of (the radical of) some ideal with known generators. Consider the radical ideal

$$\begin{aligned} J = <& w^3xz^2 - w^3y^2z + 3wx^2yz^2 - 3wxy^3z + 7wxz^4 - 7wy^2z^3 + 2xy^4z - 2y^6, \\ & w^4xz - w^3yz^2 + 3w^2x^2yz + 7w^2xz^3 + 2wxy^4 - 3wxy^2z^2 - 7wyz^4 - 2y^5z, \\ & w^5yz - w^4yz^2 - w^4z^3 + 3w^3xy^2z + 7w^3yz^3 + w^3z^4 - 3w^2xy^2z^2 - 3w^2xyz^3 \\ & + 2w^2y^5 - 7w^2yz^4 - 7w^2z^5 + 3wxyz^4 - 2wy^5z - 2wy^4z^2 + 7wz^6 + 2y^4z^3 > . \end{aligned}$$

The astute reader might notice that these generators factor rather nicely:

$$\begin{aligned} J = <& (w^3z + 3wxyz + 7wz^3 + 2y^4)(xz - y^2), \\ & (w^3z + 3wxyz + 7wz^3 + 2y^4)(wx - yz), \\ & (w^3z + 3wxyz + 7wz^3 + 2y^4)(wy - z^2)(w - z) > . \end{aligned}$$

Since the ideal generators do not exceed degree seven, we will set $D = 7$. Allowing for two-digit coefficients, we need at least $2(\binom{3+7}{3} - 1) = 238$ digits of precision, so to be conservative we use 300 digits of precision. It should be noted that it would suffice to set precision to 100 digits and D to four, though we have no way of knowing that *a priori*.

We first run Bertini on the non-factored generator list, which in turn outputs witness points on each of three components. Each witness point is an approximation of a generic point accurate to 300 digits.

Next, for one witness point on the first component, we compute the Veronese embeddings starting at degree one and continuing to degree seven, then use LLL to search for integer relations on each embedding. As we find relations, we reinterpret them as polynomial relations, and filter out any that are algebraically dependent on lower-degree polynomial relations. Finally, we repeat the process for one point on each of the other two components. Our final output is the following:

Component 1:	$zw^3 + 3zyxw + 7z^3w + 2y^4$
Component 2:	$-zx + y^2$ $-xw + zy$ $-yw + z^2$
Component 3:	x y $-w + z$

Given that the precision we used significantly overshoots what we would require to deduce the same relations, this gives us strong evidence (but does not prove!) that

$$\begin{aligned} J = &\left\langle zw^3 + 3zyxw + 7z^3w + 2y^4 \right\rangle \\ &\cap \left\langle -zx + y^2, -xw + zy, -yw + z^2 \right\rangle \\ &\cap \left\langle x, y, -w + z \right\rangle . \end{aligned}$$

For reference, symbolic computation of the same result with Gröbner bases is described in [14], along with computation of radical ideals, which is the topic of the next example.

3.2 Radical Ideals

Since varieties are in natural correspondence to radical ideals, we expect to be able to recover generators for these as well. A subtle point, however, is

that if the variety of a radical ideal, $\sqrt{J}$, is reducible, a random point chosen on the variety will lie on some irreducible component. This means we will always find elements in a prime component of $\sqrt{J}$, which will in general properly contain $\sqrt{J}$. That is, using only one point in the algorithm, we will find some relations not in $\sqrt{J}$. To get around this, we will use the fact that $\sqrt{J}$ is the intersection of its prime components, and hence consists of all polynomials which simultaneously vanish on generic points on each irreducible component.

We can tweak the algorithm to handle this new condition as follows: instead of treating each irreducible component separately, we consider a set of generic points with one generic point for each irreducible component. For each degree, we take a random linear combination of the embedded images of the points. We then use Algorithm 1 to find integer relations on the general linear combination (much like we do to compress complex points, as described in §2.4). Again, the random linear combination ensures that, with probability one, the only integer relations that exist will be those that vanish on each of the original embedded points independently, which is precisely the condition we require for generators of $\sqrt{J}$.

For example, consider the ideal

$$\begin{aligned} J = < & -512w^3y + 1728w^2xy - 1944wyx^2 + 729yx^3, \\ & - 45x^2z^2 - 230zwx^2 + 200x^3z + 5x^2zy - 20zxwy - 50wyx^2 \\ & + 200yx^3 - 200w^2x^2 - 42z^3x - 98z^2wx - 22z^2xy - 5z^4 \\ & + 50x^3w - 80w^2xz - 10z^3w - 3yz^3 - 2z^2wy - 8w^2z^2 > . \end{aligned}$$

The degrees and coefficient size of the generators give us reasonable guesses for appropriate bounds on the radical generators, so we set $D = 4$ and precision to 180 digits to allow coefficients with four digits. Running the tweaked algorithm gives the following output:

Component 1:	$8yw - 9yx,$ $-40xw^2 + 8yw^2 - 8zw^2 + 10x^2w + 13yxw - 48zxw$ $-10zyw - 10z^2w + 4yx^2 + 40zx^2 + 2zyx - 17z^2x$ $-3z^2y - 5z^3$

Hence, we deduce that

$$\begin{aligned} \sqrt{J} = < & 8yw - 9yx, -40xw^2 + 8yw^2 - 8zw^2 + 10x^2w + 13yxw - 48zxw \\ & - 10zyw - 10z^2w + 4yx^2 + 40zx^2 + 2zyx - 17z^2x - 3z^2y - 5z^3 > . \end{aligned}$$

The reader may verify that the original ideal is identical to the following, and thereby check the results:

$$J = < y^3(8w - 9x)^3,$$
$$(5x + z)^2(-5z^2 - 10zw + 8zx - 3zy + 2wx - 2wy + 8xy - 8w^2) > .$$

3.3 Elimination Ideals

An elimination ideal is the intersection of an ideal with a ring in fewer variables. For example, if $J \subset \mathbb{C}[x_0, x_1, x_2, x_3]$, then $J_2 \doteq J \cap \mathbb{C}[x_2, x_3]$ is an elimination ideal (often called the *second* elimination ideal, since it eliminates the first two variables). The ideal J_2 consists of all polynomials in J that only depend on the variables x_2 and x_3. If J has a minimal generating set lying in $\mathbb{Q}[x_0, x_1, \ldots, x_N]$ then, for each k, $J_k = J \cap \mathbb{C}[x_k, \ldots, x_N]$ will have a minimal generating set lying in $\mathbb{Q}[x_k, \ldots, x_N]$. See [12], for example, for more details about elimination ideals and elimination theory. If $V(J)$ is an irreducible variety, then the radical of J_k can be found by determining the set of integer relations on the last $N-k+1$ coordinates of a generic point of $V(J)$. For general J, we can recover generators for the radical of J_k by using projections of generic points of $V(J)$ combined with the approach described in §3.2 for radical ideals.

As a concrete example, consider the ideal

$$J = < -5z^2w - 2z^2y - 4zwx - 8wyz - zxy + 8wxy + 8w^3,$$
$$-4z^2 + 10wx, 6z + 3w > .$$

Since we are unsure of the degree and coefficient bounds, we use higher settings of 780-digit precision and $D = 7$. Projecting generic points to the last two coordinates before running the algorithm, we obtain:

Component 1:	$-87zy + 278z^2$

This suggests the radical of $J \cap \mathbb{Q}[y, z]$ is $\langle -87zy + 278z^2 \rangle$. Projecting instead onto the last three coordinates gives us:

Component 1:	$5zx + z^2$
	$-70zx - 87zy + 264z^2$

From which we deduce the radical of $J \cap \mathbb{Q}[x, y, z]$ is $\langle 5zx + z^2, -70zx - 87zy + 264z^2 \rangle$.

Actually using our method to find elimination ideals with the intent of solving a polynomial system would be rather redundant given that we must solve the system first to get witness points. However, elimination does come up in other symbolic computations. In the next few examples we will see instances that are typically carried out, in a symbolic setting, with elimination theory.

3.4 Join and Secant Varieties

If U and W are irreducible varieties, then their *join* is the Zariski closure of the union of all lines which intersect both U and W. The *secant variety* of U is the join of U with itself. It is the Zariski closure of the union of all secant lines of U.

Given a point on U and one on W, any linear combination of these points is in the join. To get a generic point in the join, we can choose generic points on U and W and take a random linear combination of the two generic points. Using numerical algebraic geometry methods and random number generators, this is a simple procedure. There is no fundamental difference when we want to compute a secant variety, as long as we are careful to choose two distinct generic points. Depending on the degree of the variety, however, we may only require one homotopy run to do so, since the witness set of an irreducible component in fact contains points equal in number to the degree of the component.

We will consider a classical example of a secant variety. Let U be the variety given by the image of the Veronese embedding $v_2^2 : \mathbb{P}^2 \hookrightarrow \mathbb{P}^5$, a cousin of the twisted cubic. It is equivalently defined as $V(J)$, where $J = \langle UX - V^2, UY - VW, UZ - W^2, VY - WX, VZ - WY, XZ - Y^2 \rangle \subseteq \mathbb{C}[U, V, W, X, Y, Z]$. Like in the example of the twisted cubic, we could have determined J from a generic point on $V(J)$. A generic point on $V(J)$ can be created by evaluating $[x^2 : xy : xz : y^2 : yz : z^2]$ at a generic value of x, y and z. A general point on the secant variety to the Veronese surface can be found by generating two general points on the surface then taking a general linear combination of these two points. Without knowing what to expect for coefficient size and degree, we might choose to pick relatively large bounds. We chose 700 digits and $D = 4$ then passed a general point on the secant variety to Algorithm 1. The result is:

Component 1:	$-XZU + Y^2U + V^2Z - 2WYV + W^2X$

Once again, we can draw a close comparison to the symbolic method for computing the same kind of results. Briefly, suppose I and J are ideals

in $\mathbb{P}^5$. Points in the join variety of $V(I)$ and $V(J)$ will have the form $[sx_0 + ty_0 : sx_1 + ty_1 : \cdots : sx_5 + ty_5]$ with s and t free parameters, with $[x_0 : x_1 : \cdots : x_5]$ satisfying the relations of I and $[y_0 : y_1 : \cdots : y_5]$ satisfying those of J. We pick new variables $z_0, z_1, \ldots, z_5$ to represent the coordinates points in the join, which implicitly requires that $z_i = sx_i + ty_i$ for each i. Then we define a new ideal H generated by all of the polynomials $z_i - sx_i - ty_i$ (our constraints on each z_i), and the relations on the x_i's and y_i's (in essence the generators of I and J). Finally, we saturate H with respect to the ideal (s, t) (since s and t can't both be zero at the same time) and compute the elimination ideal $(H : (s, t)^\infty) \cap \mathbb{C}[z_0 : z_1 : \cdots : z_5]$.

When using our numerical approach, we effectively bypass the explicit introduction of variables s, t, x_i, and y_i by picking generic values for each; homotopy methods give us generic x_i and y_i, and our random linear combination of the points defines s and t. Also, the elimination step is once again only implicitly done. Consequently, we are able avoid more than tripling the number of variables as occurs in the symbolic case. In our approach, the cost is incurred in the precision requirements and in the use of the LLL algorithm.

3.5 Desingularization through blowing up

Let V be a singular variety (i.e. a variety containing at least one singular point). In 1964, Hironaka proved in a celebrated paper [22] that every singular projective variety defined over a field of characteristic zero has a resolution. In other words, a non-singular variety X with a proper birational map $X \to V$. In this section, we show how to remove a simple singularity on a curve by applying Algorithm 1 to a generic point on the blow-up. We will give a concrete example of the process here without going into great detail about the underlying theory.

Consider the variety $V = V(J)$, where J is the ideal

$$< x^4 + 15x^3y + 31x^2y^2 + 5xy^3 + 4y^4 + 3x^3z + 26x^2yz + 16xyz^2 + 15y^3z + 11x^2z^2 + 2xyz^2 + y^2z^2 > .$$

Notice that V has a singularity at $[0 : 0 : 1]$, since all the partial derivatives of its defining equation vanish whenever $x = y = 0$. We define new variables A, B, C, D, and E to correspond to the degree two monomials in $\mathbb{Q}[x, y, z]$ which vanish at the singular point. More specifically, we set $A = x^2$, $B = xy$, $C = xz$, $D = y^2$, and $E = yz$ (the remaining monomial, z^2, does not vanish at the singular point). The idea is to find the set of relations which the new

coordinates must satisfy, given the implicit relations on the corresponding monomials induced by the generator of J.

To do this symbolically, we would construct a new ideal $J' = (J, A - x^2, B - xy, C - xz, D - y^2, E - yz) \subset \mathbb{Q}[x, y, z, A, B, C, D, E]$, which derive from the above equations. We would then compute the elimination ideal $J' \cap \mathbb{Q}[A, B, C, D, E]$.

We will use the symbolic setup to guide our attack on the problem using numerical techniques. The main issue is producing a generic point on $V(J' \cap \mathbb{Q}[A, B, C, D, E])$. Since A, B, C, D, and E are defined in terms of x, y, and z, we need generic values for these variables, which are only constrained to come from the coordinates of a point $[x : y : z] \in V$. In other words, our strategy is to first find a witness point of V, then explicitly compute a generic point $[A : B : C : D : E]$ using the equations $A = x^2$, $B = xy$, $C = xz$, $D = y^2$, and $E = yz$.

Since A, B, C, D, and E are computed from degree two monomials and the original variety is generated by a degree four polynomial, it turns out we can set $D = 2$. Coefficient size, as usual, is more difficult to predict, so we will use a precision of 600 digits to start.

Component 1:	$-DA + B^2$
	$-EA + CB$
	$-EB + DC$
	$A^2 + 15BA + 3CA + 4D^2 + 14EA + 17B^2 + 12CB$
	$\ldots + 5DB + 11C^2 + 18EC + 14DA + 15ED + E^2$

It can be checked that these equations determine a smooth curve in $\mathbb{P}^4$.

3.6 Extensions of the approach

A feature of numerical homotopy methods is the ability to produce arbitrarily many witness points of prescribed precision on any given irreducible component of a variety V. The entire collection of witness points on the irreducible component can be used in the LLL algorithm in a single run. The computational advantages of this approach are unclear. However, both experimentation and intuition suggest that utilizing many witness points of lower precision should carry similar information to a single witness point of higher precision. This leads to the possibility of obtaining exact equations by considering very large sets of very low precision witness points and utilizing the main algorithm of this paper.

A second feature of numerical homotopy methods is the ability to produce witness points on every irreducible component of a variety. This allows for a potentially nice interaction between numeric and symbolic methods. For instance, let V be a variety in a high dimensional projective space whose irreducible components (over $\mathbb{C}$) have a wide range of dimensions. Assume further that the homogeneous ideal of V has a generating set with rational coefficients. A typical such problem may initially be too complicated for a Gröbner basis algorithm to handle. Numerical methods can produce witness points on every irreducible component of V and can realistically hope to identify exact low degree equations (if they exist) in the ideal of each irreducible component of V (over $\mathbb{Q}$). This additional information can sometimes be used to quotient away (using symbolic methods) some of the irreducible components leading to a simpler pair of problems that are within reach of a Gröbner basis algorithm. In addition, symbolic methods can be used to provide a certificate for the data produced by the LLL algorithm. Thus, one can envision problems where the identification of some of the components in a prime decomposition of $J_{\mathbb{Q}}$ through a numerical homotopy over the complex numbers combined with the LLL algorithm provides enough simplification to allow a Gröbner basis algorithm to complete the prime decomposition where it was initially stymied.

A third feature of numerical homotopy methods is the ability to access generic points of a variety in a concrete manner. As we already saw in Section 3.5, simple manipulations of generic points can correspond to relatively intricate operations. For instance, consider a collection of 3 curves meeting transversely at the point $[0:0:0:1]$ in $\mathbb{P}^3$. Blow up the three curves at this point of intersection through a map to $\mathbb{P}^8$. The join of these 3 blown up curves will be a 5-fold in $\mathbb{P}^8$. It is a simple matter to produce a witness point on this 5-fold with arbitrary precision. One starts with three generic points, one on each curve in $\mathbb{P}^3$, follows them to $\mathbb{P}^8$ to get three generic points on the blown up curves, then takes a general linear combination of the three generic points on the blown up curves to get a generic point on the join variety.

4 Conclusions

This article presents a set of novel numeric-symbolic methods for carrying out an array of computational problems in algebraic geometry that were previously outside the realm of numerical computation. These methods marry numerical techniques in the field of numerical algebraic geometry

to exactness recovery techniques such as LLL or PSLQ. Post processing and further certification can be provided with Gröbner basis techniques. Several applications were presented, including the prime decomposition of the radical of an ideal, elimination, the computation of defining equations for the join of two varieties, and desingularization of a singular curve.

Much work remains in this direction; indeed, this article is intended to be the first in a series of articles aimed at the recovery of exactness after an application of numerical methods. At a fundamental level, one direction of research to consider is that LLL and PSLQ both require very high precision. While numerical algebraic geometry methods can produce any necessary level of accuracy, their use in the LLL algorithm can lead to a time-consuming computation. Similarly, the post processing via Gröbner basis computations is certainly valid, but can also be time-consuming. It is hoped that the vector space (numerical rank computation) method for checking for algebraic independence will ultimately save considerable computational resources. It has also been suggested to the authors that characteristic sets or triangular sets would be a more efficient symbolic means for checking for independence of a set of generators for an ideal.

This project is part of a larger series of projects aimed at providing numerical alternatives and augmentations to common symbolic computations in algebraic geometry and at providing tools that extend the range of applicability of symbolic methods. For example, it is known how to numerically compute the dimension of a linear series on a curve, the multiplicity structure of a zero-scheme, intersection numbers of Chern classes, and progress has been made on a numerical primary decomposition [21, 8, 13, 4, 24]. With 1000-10000 core machines on the horizon, it is hoped that numerical methods will play a growing role in extending the applicability of computational algebraic geometry to settings that were unimaginable a decade ago.

References

[1] E. Allgower and K. Georg, *Introduction to numerical continuation methods*, Classics in Applied Mathematics 45, SIAM Press, Philadelphia, 2003.

[2] W. Backes and S. Wetzel, Heuristics on lattice basis reduction in practice, *J. Exp. Algorithmics*, 7 (2002), pp. 1-21.

[3] D. Bailey and D. Broadhurst, Parallel integer relation detection: techniques and applications, *Mathematics of Computation*, 70 (2001), pp. 1719–1736.

[4] D. Bates, D. Eklund, C. Peterson, Computing Intersection Numbers of Chern Classes, *Submitted*

[5] D.J. Bates, J.D. Hauenstein, C. Peterson, and A.J. Sommese, A numerical local dimensions test for points on the solution set of a system of polynomial equations, *SIAM J. Numer. Anal.*, 47 (2009), no. 5, pp. 3608–3623.

[6] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler, Bertini: software for numerical algebraic geometry, Available at `www.nd.edu/∼sommese/bertini`.

[7] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler, Adaptive multiprecision path tracking, *SIAM J. Numer. Anal.*, 46 (2008), no. 2, pp. 722–746.

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

[9] D. Bayer and D. Mumford, What can be computed in algebraic geometry? in *Computational Algebraic Geometry and Commutative Algebra*, University Press, 1992, pp. 148.

[10] CoCoA Team, CoCoA: a system for doing Computations in Commutative Algebra. Available at http://cocoa.dima.unige.it

[11] H. Cohen, *A course in computational algebraic number theory*, Graduate Texts in Mathematics, vol. 138, Springer, New York, 1993.

[12] D. Cox, J. Little, and D. O'Shea, *Ideals, varieties, and algorithms*, third ed., Undergraduate Texts in Mathematics, Springer, New York, 2007.

[13] B.H. Dayton and Z. Zeng, Computing the multiplicity structure in solving polynomial systems, In Proceedings of ISSAC'05, (2005), pp. 116-123.

[14] D. Eisenbud, C. Huneke, and W. Vasconcelos, Direct methods for primary decomposition, *Inventiones Mathematicae*, 110 (1992), 207235.

[15] W. Fulton, *Intersection Theory*, Second edition. Ergebnisse der Mathematik und ihrer Grenzgebiete, 3. Folge, no. 2. Springer-Verlag, Berlin (1998).

[16] H. Ferguson and D. Bailey, A polynomial time, numerically stable integer relation algorithm, *Technical report*, 1991.

[17] J. von zur Gathen, J. Gerhard, *Modern computer algebra*, Second edition. Cambridge University Press, Cambridge, 2003

[18] D. Grayson, M. Stillman, *Macaulay2: a software system for research in algebraic geometry*, available at http://www.math.uiuc.edu/Macaulay2.

[19] G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3.0. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern (2005). `http://www.singular.uni-kl.de`.

[20] J. Hastad, B. Just, J.C. Lagarias, C.P. Schnorr Polynomial Time Algorithms for Finding Integer Relations among Real Numbers *SIAM Journal of Computing* 18, pp 859-881 (1989).

[21] J. Hauenstein, J. Migliore, C. Peterson, A. Sommese, Numerical Computation of the Dimension of the Cohomology of Twists of Ideal Sheaves, *Contemporary Mathematics* 496 "Interactions of Classical and Numerical Algebraic Geometry (Notre Dame, IN 2008)", pp 235-242, (2009)

[22] H. Hironaka, Resolution of singularities of an algebraic variety over a field of characteristic zero, *Annals of Mathematics* 79(1) (1964), pp. 109–203.

[23] A. Lenstra, H. Lenstra, and L. Lovász, Factoring polynomials with rational coefficients, *Mathematische Annalen*, 261 (1982), pp. 515–534.

[24] A. Leykin, Numerical primary decomposition, In Proceedings of ISSAC'08, (2008), pp. 165-172.

[25] T.Y. Li, Numerical solution of polynomial systems by homotopy continuation methods, in *Handbook of Numerical Analysis, Volume XI, Special Volume: Foundations of Computational Mathematics*, F. Cucker, ed., North-Holland, 2003, pp. 209–304.

[26] F. Luk and D. Tracy, An improved LLL algorithm, Linear Algebra Appl. 428 (2008), no. 2-3, pp 441–452.

[27] D. Micciancio and P. Voulgaris, Faster exponential time algorithms for the shortest vector problem, Electronic Colloquium on Computational Complexity, Report No. 65, (2009), pp.1–19

[28] P.Q. Nguyen and D. Stehlé, An LLL algorithm with quadratic complexity, *SIAM J. Computing* 39 (2009), pp. 874–903.

[29] C. Schnorr, Progress on LLL and lattice reduction, Proceedings of *LLL+25*, (2009), pp. 1–24.

[30] C. Schnorr and M. Euchner, Lattice basis reduction: Improved practical algorithms and solving subset sum problems, Mathemtical programming, 66(1-3), (1994), pp. 181–199.

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

[32] A.J. Sommese, J. Verschelde, and C.W. Wampler, Using monodromy to decompose solution set of polynomial systems into irreducible componets, in Proceedings of the 2001 NATO Advance Research Conference, Eilat, Israel, on Applications of Algebraic Geometry to Coding Theory, Physics, and Computation, edited by C. Ciliberto, F. Hirzebruch, R. Miranda, and M. Teicher, (2001), pp. 297-315.

[33] Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler, Symmetric functions applied to decomposing polynomial systems, *SIAM J. Numer. Anal.*, 40 (2002), pp. 2026-2046

[34] A.J. Sommese and C.W. Wampler, *The numerical solution to systems of polynomials arising in engineering and science*, World Scientific, Singapore, 2005.