Regenerative cascade homotopies for solving polynomial systems

Jonathan D. Hauenstein* Andrew J. Sommese† Charles W. Wampler ‡

May 27, 2011

Abstract

A key step in the numerical computation of the irreducible decomposition of a polynomial system is the computation of a *witness superset* of the solution set. In many problems involving a solution set of a polynomial system, the witness superset contains all the needed information. Sommese and Wampler gave the first numerical method to compute witness supersets, based on dimension-by-dimension slicing of the solution set by generic linear spaces, followed later by the cascade homotopy of Sommese and Verschelde. Recently, the authors of this article introduced a new method, *regeneration*, to compute solution sets of polynomial systems. Tests showed that combining regeneration with the dimension-by-dimension algorithm was significantly faster than naively combining it with the cascade homotopy. However, in this article, we combine an appropriate randomization of the polynomial system with the regeneration technique to construct a new cascade of homotopies for computing witness supersets. Computational tests give strong evidence that *regenerative cascade* is superior in practice to previous methods.

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

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

Introduction

Numerical algebraic geometry is the computation and manipulation of the solution sets of systems of polynomials using numerical algorithms (see [19] for a comprehensive development of the area). Let

$$f(z) = f(z_1, \ldots, z_N) = \begin{bmatrix} f_1(z_1, \ldots, z_N) \\ \vdots \\ f_n(z_1, \ldots, z_N) \end{bmatrix} = 0 \tag{1}$$

*Department of Mathematics, Mailcode 3368, Texas A&M University, College Station, TX 77843 (jhauenst@math.tamu.edu, `www.math.tamu.edu/~jhauenst`). This author was supported by the Fields Institute, the Duncan Chair of the University of Notre Dame, NSF grant DMS-0712910, and the Mittag-Leffler Institute.

†Department of Applied and Computational Mathematics and Statistics, 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-0712910, and the Mittag-Leffler Institute.

‡General Motors Research and Development, Mail Code 480-106-224, 30500 Mound Road, Warren, MI 48090 (Charles.W.Wampler@gm.com, `www.nd.edu/~cwample1`). This author was supported by NSF DMS-0712910 and the Mittag-Leffler Institute.

denote a system of n polynomials with $z \in \mathbb{C}^N$. The basic data structure to describe an irreducible component X of the solution set

$$V(f) := \{ z \in \mathbb{C}^N \mid f(z) = 0 \}$$

of this system is a *witness set* for X. When X is a multiplicity one irreducible component of $V(f)$, a witness set of X is a triple $(f, \mathcal{L}, X \cap \mathcal{L})$ consisting of the polynomial system f, a random affine linear subspace $\mathcal{L}$ of $\mathbb{C}^N$ of dimension complementary to the dimension of X, plus the points $X \cap \mathcal{L}$. For a component of multiplicity greater than one, a witness set contains extra information that is computable from the triple $(f, \mathcal{L}, X \cap \mathcal{L})$ [19, §13.3]. In particular, the triple $(f, \mathcal{L}, X \cap \mathcal{L})$ is the key object to compute, and for practical purposes, may be regarded as the witness set of the irreducible component X.

Witness sets are a natural data structure to numerically describe the decomposition of a solution set into irreducible components and to enable the computation of other detailed information about algebraic sets, e.g., their intersections [17]. Using homotopy continuation, it is computationally inexpensive to compute as many random and widely-distributed points on X as desired, from which, for instance, polynomials cutting out the component X may be computed.

A witness set of X is computed from a witness superset of X. A *witness superset* of X is a triple $(f, \mathcal{L}, S)$ where f and $\mathcal{L}$ are as above and S is a finite set of points with $(X \cap \mathcal{L}) \subset S \subset (V(f) \cap \mathcal{L})$. The points in $S \setminus X$ are called *junk points* and are filtered out to create a witness set.

The numerical irreducible decomposition of $V(f)$ is a collection, say $\mathcal{W}$, of witness sets for each irreducible component of $V(f)$. The computation of $\mathcal{W}$ proceeds in three steps:

1. computation of a witness superset $\widehat{\mathcal{W}}$ of $V(f)$;
2. computation of a witness set $\mathcal{W}$ of $V(f)$ from $\widehat{\mathcal{W}}$;
3. decomposition of the witness set $\mathcal{W}$ into witness sets for the irreducible components of $V(f)$.

There are two existing algorithms to perform Step 1, namely a dimension-by-dimension algorithm presented in [18] and a cascade algorithm presented in [16]. While the computational cost of the two approaches is usually roughly comparable, these methods may differ in the number of junk points they produce. In this way, the choice of method directly affects the computational cost of Step 2. In this regard, the cascade algorithm is favored as it generally computes many fewer junk points than the dimension-by-dimension algorithm.

It should be mentioned that a version of the regeneration algorithm presented in [7] performs Steps 1 and 2 together in an integrated way. A significant drawback of that version of regeneration is that it may require deflation of intermediate systems that arise. Deflation is a procedure for replacing a system of polynomials $f(z) = 0$ on $\mathbb{C}^N$ and an isolated singular solution x^* with a new polynomial system $F(z, \xi) = 0$ on $\mathbb{C}^N \times \mathbb{C}^M$ and an isolated nonsingular solution (x^*, ξ^*). It applies to deflate components, i.e., to replace a system of polynomials $f(z) = 0$ on $\mathbb{C}^N$ and a nonreduced irreducible component X of the solution set with a new polynomial system $F(z, \xi) = 0$ on $\mathbb{C}^N \times \mathbb{C}^M$ and a generically reduced irreducible component Y of the solution set such that Y maps generically one-to-one to X under the projection $(z, \xi) \to z$. See [6, 11, 12, 14, 15] for more details of deflation applied to isolated points and [19, §13.3.2, §15.2.2] for deflating components with [7, §4.1] presenting an improvement of deflation directly related to regeneration. Neither the dimension-by-dimension method nor the cascade method have this drawback. Like these, the new algorithm presented in this paper also does not require deflation

to compute a witness superset. It is based on the version of regeneration from [7] that seeks only to find isolated solution points. In this article, we limit our computational experiments to the three methods that never require deflation.

Step 3 is performed using a monodromy algorithm that is certified using a trace test. This step is independent of Steps 1 and 2, so we shall say nothing further about it here.

For many problems, Step 1 is the most computationally intensive part of finding a numerical irreducible decomposition. The subject of this article, the regenerative cascade, addresses this step. We provide evidence that the new approach matches the efficiency of the older cascade algorithm in regards to the number of junk points it produces while reducing the cost of Step 1 as compared to both of the prior algorithms.

This article is organized as follows. Section 1 provides an overview of regeneration [7]. Section 2 presents the regenerative cascade algorithm; Section 2.3 describes the advantages of the regenerative cascade algorithm; and Section 3 provides computational results.

1 Regeneration

Before describing the regenerative cascade algorithm in §2, we summarize the regeneration algorithm presented in [7] for computing the nonsingular isolated solutions of a square polynomial system using an equation-by-equation approach with a general linear product decomposition.

Let $f : \mathbb{C}^N \to \mathbb{C}^n$ be a polynomial system as in Eq. 1 with $d_i = \deg f_i$. For $1 \le i \le n$ and $1 \le j \le d_i$, let $L_{i,j}(z)$ be a general linear function on $\mathbb{C}[z_1, \ldots, z_n]$ and, for $0 \le m \le n$, define

$$\begin{array}{rcl} F_m(z) & = & [\ f_{1:m}(z), \quad L_{m+1,1}(z), \quad L_{m+2:n,1}(z)\ ]^T \quad \text{and} \\ G_m(z) & = & [\ f_{1:m}(z), \quad \prod_{j=1}^{d_{m+1}} L_{m+1,j}(z), \quad L_{m+2:n,1}(z)\ ]^T \end{array}$$

where $f_{1:m}(z) = [f_1(z), \ldots, f_m(z)]$ and $L_{m+2:n,1}(z) = [L_{m+2,1}(z), \ldots, L_{n,1}(z)]$.

Regeneration starts with the solution of the linear system $F_0(z) = 0$ and computes the solutions of $F_n(z) = f(z) = 0$ using a sequence of two types of homotopies. For $0 \le m \le n-1$ and $1 \le a \le d_i$, the first type of homotopy is a parameter homotopy that moves from the linear space $L_{m,1} = 0$ to $L_{m,a} = 0$, namely

$$H^{\text{parm}}_{m,a}(z,t) = \left[\ f_{1:m}(z), (1-t)L_{m+1,a}(z) + tL_{m+1,1}(z), L_{m+2:n,1}(z)\ \right]^T.$$

For $0 \le m \le n-1$, the second type of homotopy is a general linear product homotopy that moves from the product of linear spaces $\prod_{i=1}^{d_{m+1}} L_{m+1,i} = 0$ to $f_{m+1} = 0$, namely

$$\begin{array}{rcl} H^{\text{prod}}_m(z,t) & = & (1-t)F_{m+1}(z) + tG_m(z) \\ & = & \left[\ f_{1:m}(z), (1-t)f_{m+1}(z) + t\prod_{i=1}^{d_{m+1}} L_{m+1,i}(z), L_{m+2:n,1}(z)\ \right]^T. \end{array}$$

The following algorithm computes the nonsingular isolated points of $V(f)$ using regeneration.

Procedure $S = \textbf{Regenerate}(f)$

Input A set $f = \{f_1, \ldots, f_n\}$ of n polynomials on $\mathbb{C}^n$.

Output A set $S \subset \mathbb{C}^n$ consisting of the nonsingular isolated points of $V(f)$.

Begin 1. Let $d_i = \deg f_i$. For $i = 1, \ldots, n$ and $j = 1, \ldots, d_i$, let $L_{i,j}(z)$ be a random linear function on $\mathbb{C}[z_1, \ldots, z_n]$.

2. Use numerical linear algebra to compute the set S_0 consisting of the solution of the linear system $F_0 = 0$.
3. For $m = 0, \ldots, n-1$, do the following:
   (a) Solve for T_m, the set of nonsingular isolated points of $V(G_m)$, using the homotopies $H_{m,j}^{\text{parm}}$, $j = 1, \ldots, d_{m+1}$, with start points S_m.
   (b) Solve for S_{m+1}, a superset of the nonsingular isolated points of $V(F_{m+1})$, using the homotopy H_m^{prod} with start points T_m.
   (c) Expunge any singular points in S_{m+1}.

Return $S = S_n$**.**

Note 1.1 $H_{m,1}^{\text{parm}}(z,t) \equiv F_m(z)$.

2 Regenerative cascade

The regenerative cascade algorithm is applicable to the following basic problem in numerical algebraic geometry.

Problem 1 (Witness Superset) *Given a polynomial system $f : \mathbb{C}^N \to \mathbb{C}^n$ find a witness superset for $V(f)$.*

2.1 Regenerative cascade algorithm

Let $f : \mathbb{C}^N \to \mathbb{C}^n$ be a polynomial system as in Eq. 1 with $d_1 \geq \cdots \geq d_n$ where $d_i = \deg f_i$. Let $r = \text{rank}(f)$ (see [19, §13.4]) and $\alpha_{i,j} \in \mathbb{C}$ be general for $1 \leq i \leq r$ and $i < j \leq n$. Define

$$A = \begin{bmatrix} 1 & \alpha_{1,2} & \alpha_{1,3} & \cdots & \cdots & \cdots & \alpha_{1,n} \\ & 1 & \alpha_{2,3} & \cdots & \cdots & \cdots & \alpha_{2,n} \\ & & \ddots & \ddots & & & \vdots \\ & & & 1 & \alpha_{r,r+1} & \cdots & \alpha_{r,n} \end{bmatrix} \text{ and } \widehat{f} = A \cdot f. \tag{2}$$

The polynomial system $\widehat{f}$ consists of r polynomials in N variables with $r \leq N$ and $\deg \widehat{f}_i = d_i$. Since $\dim V(\widehat{f}) \geq N - r$, we will append $N - r$ general linear functions onto $\widehat{f}$. That is, let $\mathcal{L}_i(z)$ be a generic linear function on $\mathbb{C}[z_1, \ldots, z_N]$ and $\mathcal{L} = [\mathcal{L}_1, \ldots, \mathcal{L}_{N-r}]^T$. Define

$$\mathcal{F} = \begin{bmatrix} \mathcal{L} \\ \widehat{f} \end{bmatrix}. \tag{3}$$

The following lemma shows how to compute a witness superset for $V(f)$ using a witness superset for $V(\mathcal{F})$.

In the above, "generic" means that the generality needed to obtain the results we seek holds for all but a proper algebraic subset of the set of all possible choices of $\alpha_{i,j}$ and the coefficients defining the linear functions $\mathcal{L}$ and L_i, $i = 1, \ldots, N-r$. Let us call the complex Euclidean space of all these parameters the "algorithm parameter space."

Lemma 2.1 *For a Zariski open dense subset of the algorithm parameter space, the following holds. Let $\mathcal{W}_i = \{\mathcal{F}, L_i, \mathcal{X}_i\}$ be a witness superset for the ith dimensional irreducible components of $V(\mathcal{F})$. If $X_i = \{x \in \mathcal{X}_i \mid f(x) = 0\}$, then $W_{N-r+i} = \{f, \{\mathcal{L}, L_i\}, X_i\}$ is a witness superset for the $(N-r+i)$-th dimensional irreducible components of $V(f)$.*

Proof. Altogether $\{\mathcal{L}, L_i\}$ are $N-r+i$ generic linear equations, hence $\mathcal{X}_i$ is a witness superset for the $(N-r+i)$-dimensional irreducible components of $\widehat{f}$. By the randomization theorem, Theorem 13.5.1 of [19], all the $(N-r+i)$-dimensional irreducible components of f will be among these, but only points in $V(f)$ can be witness points for components of $V(f)$. □

For $1 \leq i \leq N-r$ and $1 \leq j \leq d_i$, let $L_{i,j}(z)$ be a general linear function on $\mathbb{C}[z_1, \ldots, z_n]$. The following are analogous to $F_m, G_m, H_m^{\text{prod}}$ and $H_{m,a}^{\text{parm}}$, defined in §1:

$$\begin{aligned}
\mathcal{F}_m(z) &= [\ \mathcal{L}(z),\ \widehat{f}_{1:m}(z),\ L_{m+1,1}(z),\ L_{m+2:r,1}(z)\ ]^T,\\
\mathcal{G}_m(z) &= [\ \mathcal{L}(z),\ \widehat{f}_{1:m}(z),\ \textstyle\prod_{j=1}^{d_{m+1}} L_{m+1,j}(z),\ L_{m+2:r,1}(z)\ ]^T,\\
\mathcal{H}_{m,a}^{\text{parm}}(z,t) &= [\ \mathcal{L}(z),\ \widehat{f}_{1:m}(z),\ (1-t)L_{m+1,a}(z) + tL_{m+1,1}(z),\ L_{m+2:r,1}(z)\ ]^T,
\end{aligned}$$

$$\text{and} \quad \mathcal{H}_m^{\text{prod}}(z,t) = (1-t)\mathcal{F}_{m+1}(z) + t\mathcal{G}_m(z).$$

A point $x \in V(\mathcal{F}_m)$ (or $V(\mathcal{G}_m)$) is a *nonsolution* with respect to f if $f(x) \neq 0$. Bertini's theorem and genericity provide that nonsolutions of $\mathcal{F}_m$ and $\mathcal{G}_m$ are nonsingular isolated solutions of $\mathcal{F}_m$ and $\mathcal{G}_m$, respectively. The following lemma shows that regenerating the nonsolutions of $\mathcal{F}_m$ will yield a superset of the isolated solutions of $\mathcal{F}_{m+1}$.

Lemma 2.2 *For a Zariski open dense subset of the algorithm parameter space, the set of solutions of $\mathcal{F}_{m+1}$ obtained by regenerating the nonsolutions of $\mathcal{F}_m$ contains the isolated solutions of $\mathcal{F}_{m+1}$.*

Proof. According to the theory underlying the regeneration algorithm (§1), the isolated solutions of $\mathcal{F}_{m+1}$ are contained in the set of endpoints of $\mathcal{H}_m^{\text{prod}}$ using the isolated solutions of $\mathcal{G}_m$ as start points, which themselves are obtained by the homotopies $\mathcal{H}_{m,j}^{\text{parm}}$, $j = 1, \ldots, d_{m+1}$ using the isolated solutions of $\mathcal{F}_m$ as start points. If, for $m < r$, x is an isolated solution of $\mathcal{F}_m$ with $f(x) = 0$, then it lies on a component of $V(f)$ of dimension at least m. The paths from such a point must remain on this component during the homotopies $\mathcal{H}_m^{\text{prod}}$ and $\mathcal{H}_{m,j}^{\text{parm}}$, $j = 1, \ldots, d_{m+1}$. and hence cannot lead to nonsingular solutions to $\mathcal{F}_{m+1}$. Thus, the paths originating from the nonsolutions of $\mathcal{F}_m$ suffice to find all isolated solutions of $\mathcal{F}_{m+1}$. □

Procedure $S = \textbf{RegenerativeCascade}(f)$

Inputs A set $f = \{f_1, \ldots, f_n\}$ of n polynomials on $\mathbb{C}^N$.

Output A witness superset S for $V(f)$.

Begin

1. Rename f so that $d_i \geq \cdots \geq d_n$ where $d_i = \deg f_i$ and compute $r = \text{rank}(f)$.
2. For $1 \leq i \leq r$ and $i < j \leq n$, let $\alpha_{i,j} \in \mathbb{C}$ be random. Construct A and $\widehat{f}$ as in Eq. 2.
3. Let $\mathcal{L}(z)$ be a set of $N-r$ random linear functions on $\mathbb{C}[z_1, \ldots, z_N]$ and construct $\mathcal{F}$ as in Eq. 3.
4. For $i = 1, \ldots, r$ and $j = 1, \ldots, d_i$, let $L_{i,j}(z)$ be a random linear function on $\mathbb{C}[z_1, \ldots, z_N]$.
5. Use numerical linear algebra to compute the set X_0 consisting of the solution of the linear system $\mathcal{F}_0 = 0$.
6. For $m = 0, \ldots, r-1$, do the following:

(a) Solve for T_m, the set of nonsolutions with respect of f in $V(\mathcal{G}_m)$, using the homotopies $\mathcal{H}^{\text{parm}}_{m,j}$, $j = 1, \ldots, d_{m+1}$, with start points X_m.

(b) Solve for U_{m+1}, a set containing the nonsolutions with respect to f and the isolated points in $V(\mathcal{F}_{m+1})$, using the homotopy $\mathcal{H}^{\text{prod}}_m$ with start points T_m.

(c) Let $X_{m+1} \subset U_{m+1}$ be the nonsolutions of f and $S_{m+1} = \{f, \{\mathcal{L}, L_{m+2,1}, \ldots, L_{r,1}\}, U_{m+1} \setminus X_{m+1}\}$.

Return $S = \{S_1, \ldots, S_r\}$.

The following theorem justifies the regenerative cascade algorithm **RegenerativeCascade**.

Theorem 2.3 *For a Zariski open dense subset of the algorithm parameter space, the* **RegenerativeCascade** *solves Problem 1.*

Proof. This is a straightforward consequence of Lemmas 2.1 and 2.2. □

2.2 Extrinsic vs. intrinsic slicing

Similar to [7, §6.3], the homotopies $\mathcal{H}^{\text{parm}}$ and $\mathcal{H}^{\text{prod}}$ are called extrinsic regenerative cascade homotopies due to the presence of extrinsic linear slicing functions. Since the linear functions $\mathcal{L}, L_{m+2,1}, \ldots, L_{r,1}$ do not change during the path tracking for these homotopies, we may use an intrinsic formulation that is more efficient when $m \ll N$. Numerical linear algebra can be used to compute $B \in \mathbb{C}^{N\times(m+1)}, b \in \mathbb{C}^N$ such that rank $B = m + 1$ and for all $u \in \mathbb{C}^{m+1}$, $Bu + b \in V(\mathcal{L}, L_{m+2,1}, \ldots, L_{r,1})$. The homotopies $\mathcal{H}^{\text{parm}}$ and $\mathcal{H}^{\text{prod}}$ can then be replaced with ones of the form $\widehat{H}(u, t) = H(Bu + b, t)$. Since the linear functions are always zero and can be dropped, these new homotopies consist of $m + 1$ functions and variables, instead of N.

For efficiency, the polynomials should be evaluated in a straight-line manner rather than expanded in the new variables. When the intrinsic formulation is heuristically advantageous, Bertini [3] automatically uses it.

2.3 Advantages of the regenerative cascade algorithm

The regenerative cascade algorithm presented in §2 has several advantages over the dimension-by-dimension algorithm of [18] and the cascade algorithm of [16]. For a polynomial system $f : \mathbb{C}^n \to \mathbb{C}^n$ of rank n with $d_1 \geq \cdots \geq d_n$, where $d_i = \deg f_i$, this section describes the theoretical advantages with computational evidence presented in §3.

The first advantage is that the regenerative cascade is amenable to intrinsic slicing. This reduces the number of variables for tracking the paths, which can reduce the computational costs associated with linear algebra. While the dimension-by-dimension algorithm can also use intrinsic slicing, the original cascade cannot. It uses a homotopy in n variables at every stage.

A second advantage is that the regenerative cascade often tracks fewer paths than either of the other two algorithms. In the case of the original cascade algorithm, it is the first stage of the cascade that is expensive to solve. If one uses a total degree homotopy for this stage, the total degree number of paths, namely $d_1 \cdots d_n$, must be tracked. One could instead use a polyhedral homotopy [8, 13] to possibly reduce the number of paths to track, but this comes at the cost of constructing the polyhedral homotopy, which can be a significant cost to bear, e.g. [7, §9]. In the case of the dimension-by-dimension algorithm, the last stage is the one that must track the total degree number of paths or use a polyhedral homotopy. Due to the structure of the regenerative cascade algorithm, in which endpoints on higher-dimensional components

	paths tracked (slices moved)			number of junk points		
n	Cascade	Dimension-by-dimension	Regenerative cascade	Cascade	Dimension-by-dimension	Regenerative cascade
3	56	30	26 (12)	6	10	6
4	295	126	96 (47)	30	60	30
5	1,380	510	340 (169)	125	295	125
6	6,050	2,046	1,190 (594)	486	1,342	486
7	25,465	8,190	4,150 (2,074)	1,813	5,853	1,813
8	104,247	32,766	14,456 (7,227)	6,600	24,910	6,600
9	418,289	131,070	50,336 (25,167)	23,665	104,399	23,665
10	1,653,320	524,286	175,246 (87,622)	84,028	433,068	84,028

Table 1: Comparison for computing a witness superset for $P_{2,3,n}$ using various algorithms

(including ones at infinity) do not initiate paths at the next stage, the total number of paths that need to be tracked can be less than the total degree.

Finally, the termination of further tracking on all but the nonsolutions generally leads to a witness superset that contains fewer junk points. The regenerative cascade and original cascade algorithms share this advantage. In contrast, the dimension-by-dimension algorithm handles each dimension independently, and so has no mechanism for results obtained at a higher dimension to affect the computation at a lower dimension. This generally leads to a witness superset containing more junk points.

3 Computational results

The regenerative cascade algorithm **RegenerativeCascade** is implemented in Bertini [3], but the algorithm, in principle, could also use other path trackers such as PHC [21] or POLSYS_GLP [20]. The examples presented below using **RegenerativeCascade** were run using Bertini v1.2 with adaptive precision path tracking [1, 2, 4]. The examples using polyhedral methods were run using HOM4PS-2.0 [10] and PHC v2.3.62 [21]. The serial processing timings result from the computations being performed on a 2.4 GHz Opteron 250 processor with 64-bit Linux. The parallel processing timings result from the computations being performed on a cluster consisting of a manager that uses one core of a Xeon 5410 processor and 8 computing nodes each containing two 2.33 GHz quad-core Xeon 5410 processors running 64-bit Linux, i.e., one manager and 64 workers.

3.1 A collection of high-dimensional examples

Consider computing a witness superset for the polynomial system, denoted $P_{2,3,n}$ constructed by taking the 2×2 adjacent permanents of a $3 \times n$ matrix with indeterminant entries [9]. The polynomial system $P_{2,3,n}$ consists of $2(n-1)$ polynomials in $3n$ variables. Since the irreducible components of $V(P_{2,3,n})$ are known and exist in multiple dimensions for $n \geq 3$, we used this collection of examples to compare the algorithms.

Table 1 compares the number of paths tracked and the number of junk points in the witness superset for the cascade algorithm, dimension-by-dimension algorithm, and the regenerative cascade algorithm. Table 2 lists the time needed for computing a witness superset for $P_{2,3,n}$, $3 \leq n \leq 9$, using a single processor. Table 3 lists the time needed for computing a witness superset for $P_{2,3,n}$, $8 \leq n \leq 10$, using parallel processing. In Tables 2 and 3, the columns labeled *ratio* present the ratio of the running time of the algorithm to the running time of the

n	Cascade		Dimension-by-dimension		Regenerative cascade	
	time	ratio	time	ratio	time	ratio
3	0.26s	1.4	0.22s	1.2	0.19s	1
4	2.46s	1.6	2.20s	1.4	1.52s	1
5	18.8s	1.9	17.7s	1.8	9.92s	1
6	2m13s	1.9	2m12s	1.9	1m9s	1
7	13m58s	2.1	217m25s	2.7	6m33s	1
8	1h22m16s	2.3	1h51m8s	3.0	36m30s	1
9	7h30m1s	2.5	10h57m17s	3.7	2h56m46s	1

Table 2: Computing a witness superset for $P_{2,3,n}$ using various algorithms

n	Cascade		Dimension-by-dimension		Regenerative cascade	
	time	ratio	time	ratio	time	ratio
8	2m16s	2.6	1m52s	2.2	51.4s	1
9	10m12s	2.8	10m50s	3.0	3m38s	1
10	54m3s	3.3	60m37s	3.7	16m27s	1

Table 3: Computing a witness superset for $P_{2,3,n}$ using various algorithms in parallel

regenerative cascade. One may see that the regenerative cascade consistently tracks fewer paths and uses less computation time than the alternatives. For example, when $n = 8$, the cascade and dimension-by-dimension algorithms take 2.5 and 3.7 times longer, respectively, than the regenerative cascade algorithm using serial processing. Moreover, the two cascade algorithms are equal in the number of junk points generated, beating the dimension-by-dimension method in this regard. Fewer junk points means less work in the next stage of computing a witness set, which is the removal of junk points from the witness superset.

As mentioned in Section 2.3, we note that we could have used a polyhedral homotopy to solve the first stage of the cascade algorithm or each stage of the dimension-by-dimension algorithm. Since each irreducible component of $V(P_{2,3,n})$ has dimension at least $3n - 2(n-1) = n+2$, the above computations were actually performed intrinsically on a general $n+2$-codimensional linear space. The resulting polynomial system then consists of $2(n-1)$ dense quadratic polynomials in $2(n-1)$ variables. In this case, all of the polyhedral root counts are equal to the total degree.

If we work extrinsically on a general $n+2$-codimensional linear space, many of the polyhedral root counts are less than the total degree, but the computation is at the expense of using $3n$ variables. To explicitly demonstrate this expense, consider $P_{2,3,8}$, i.e., $n = 8$. Working intrinsically, solving the first stage of the cascade algorithm using intrinsic slicing required tracking $2^{14} = 16{,}384$ paths using 14 variables. As presented in Table 2, running the complete cascade algorithm using Bertini on a single processor took approximately 1.4 hours. Working extrinsically, the polynomial system consists of 24 variables with the polyhedral root count for the first stage of the cascade algorithm being 7,229. This computation took approximately 6.5 hours using HOM4PS-2.0 [10] and approximately 29 hours using PHC v2.3.62 [21].

3.2 An example related to a secant variety

In [5], Bates and Oeding computed a numerical irreducible decomposition for a polynomial system, denoted $\mathcal{M}_6$, consisting on ten homogeneous degree 6 polynomials in 36 variables which are contained in the ideal of the secant variety $\sigma_4(\mathbb{P}^2 \times \mathbb{P}^2 \times \mathbb{P}^3)$. The polynomials are listed in the ancillary file `deg_6_salmon.txt` at [5]. They provide computational evidence in Computation 4.1

codim	paths tracked	witness points	junk points	slices moved
1	6	0	0	30
2	36	0	0	180
3	216	0	0	1080
4	1296	345	0	4755
5	5706	0	2844	14,310
6	17,172	84	11,790	26,490
7	31,788	0	26,460	26,640
8	31,968	0	29,196	13,860
9	16,632	0	16,120	2560
10	3072	0	3072	
total	107,892	429	89,482	89,905

Table 4: Solving $\mathcal{M}_6$

that $V(\mathcal{M}_6)$ consists of two irreducible components, one of codimension 4 of degree 345 and the other of codimension 6 of degree 84.

Since $\mathcal{M}_6$ consists of only 10 polynomials, each irreducible component of $V(\mathcal{M}_6) \subset \mathbb{C}^{36}$ has dimension at least 26. We used intrinsic slicing to construct a polynomial system consisting of 10 polynomials in 10 variables. We note that the resulting polynomial system is completely dense and that standard Gaussian elimination would require on the order of $3.6^3 \approx 46$ times more operations if we used extrinsic slicing, which would be needed to possibly yield a system having a polyhedral root counts less than its total degree. Based on the results of the computation using polyhedral methods with extrinsic slicing in Section 3.1 and the fact that both the cascade and dimension-by-dimension algorithms implemented in Bertini would need to track at least the total degree ($6^{10} \approx 6 \cdot 10^7$) many paths, we only performed this computation using the regenerative cascade with intrinsic slicing.

As discussed in Section 2.2, we used a straight-line format for the intrinsic slices. Additionally, we reduced the number of operations needed to evaluate the polynomials via a multivariate Horner scheme. Using parallel processing, this computation took approximately 20 hours and verified Computation 4.1 of [5]. Table 4 summarizes the number of paths tracked by the regenerative cascade algorithm and the number of witness points and junk points obtained at each codimension.

4 Summary

The regenerative cascade algorithm **RegenerativeCascade** combines the advantages of the cascade, dimension-by-dimension, and regeneration algorithms for computing a witness superset. It uses randomization to avoid the disadvantage of the version of regeneration that sometimes needs to deflate systems during the algorithm. Computational evidence suggests that it computes a witness superset with a similar number of junk points as the cascade algorithm, which is generally less than the number of junk points generated by the dimension-by-dimension algorithm. Moreover, in our tests, the regenerative cascade tracks the fewest number of paths, which coupled with the additional advantage of being amenable to intrinsic path tracking, makes it currently the most efficient general method for generating a witness superset.

References

[1] D.J. Bates, J.D. Hauenstein, and A.J. Sommese. Efficient path tracking methods. To appear in *Numer. Algorithms*.

[2] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. Adaptive multiprecision path tracking. *SIAM J. Numer. Anal.*, 46(2), 722–746, 2008.

[3] 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`.

[4] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. Stepsize control for adpative multiprecision path tracking. *Contemp. Math.*, 496, 21–31, 2009.

[5] D.J. Bates and L. Oeding. Toward a salmon conjecture. Available at `arxiv.org/abs/1009.6181`.

[6] B.H. Dayton and Z. Zeng. Computing the multiplicity structure in solving polynomial systems. In *ISSAC'05*, 116–123 (electronic), ACM, New York, 2005.

[7] J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. Regeneration homotopies for solving systems of polynomials. *Math. Comp.* 80, 345–377, 2011.

[8] B. Huber and B. Sturmfels. A polyhedral method for solving sparse polynomial systems. *Math. Comp.* 64, 1541–1555, 1995.

[9] R.C. Laubenbacher and I. Swanson. Permanental ideals. *J. Symbolic Comput.* 30, 195–205, 2000.

[10] T.-L. Lee, T.Y. Li, and C.-H. Tsai. HOM4PS-2.0, Solving Polynomial Systems by the Polyhedral Homotopy Method. Available at `www.math.msu.edu/~li`.

[11] A. Leykin, J. Verschelde, and A. Zhao. Higher-order deflation for polynomial systems with isolated singular solutions. In *IMA Volume 146: Algorithms in Algebraic Geometry*, edited by A. Dickenstein, F.-O. Schreyer, and A.J. Sommese, Springer, 79–97, 2008.

[12] A. Leykin, J. Verschelde, and A. Zhao. Newton's method with deflation for isolated singularities of polynomial systems. *Theor. Comp. Sci.* 359, 111–122, 2006.

[13] T.Y. Li. Numerical solution of polynomial system by homotopy continuation methods. In *Handbook of Numerical Analysis, Vol. XI*, 209–304. North-Holland, Amsterdam, 2003.

[14] T. Ojika. Modified deflation algorithm for the solution of singular problems. I. A system of nonlinear algebraic equations, *J. Math. Anal. Appl.* 123, 199–221, 1987.

[15] T. Ojika, S. Watanabe, and T. Mitsui. Deflation algorithm for the multiple roots of a system of nonlinear equations. *J. Math. Anal. Appl.* 96, 463–479, 1983.

[16] A.J. Sommese and J. Verschelde. Numerical homotopies to compute generic points on positive dimensional algebraic sets. *J. Complexity* 16, 572–602, 2000.

[17] A.J. Sommese, J. Verschelde, and C.W. Wampler. Homotopies for intersecting solution components of polynomial systems. *SIAM J. Numer. Anal.* 42, 1552–1571, 2004.

[18] A.J. Sommese and C.W. Wampler. Numerical algebraic geometry. in *The Mathematics of Numerical Analysis*, J. Renegar, M. Shub, and S. Smale, eds., volume 32 of *Lectures in Applied Mathematics*, 1996, 749–763. Proceedings of the AMS-SIAM Summer Seminar in Applied Mathematics, Park City, Utah, July 17-August 11, 1995, Park City, Utah.

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

[20] H.-J. Su, J.M. McCarthy, M. Sosonkina, and L.T. Watson. Algorithm 857: POLSYS_GLP: A parallel general linear product homotopy code for solving polynomial systems of equations, *ACM Trans. Math. Softw.* 32(4), 561–579, 2006.

[21] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Trans. Math. Softw.* 25(2), 251–276, 1999. Available at `www.math.uic.edu/~jan`.