Perturbed Regeneration for Finding All Isolated Solutions of Polynomial Systems

Daniel J. Bates[a,1,*], Brent Davis[a,1], David Eklund[b], Eric Hanson[a,1], Chris Peterson[a,2]

[a] *Department of Mathematics, Colorado State University, Fort Collins, CO 80523–1874*

[b] *Stockholm, Sweden*

Abstract

Given a polynomial system $f : \mathbb{C}^N \to \mathbb{C}^n$, the methods of numerical algebraic geometry produce numerical approximations of the isolated solutions of $f(z) = 0$, as well as points on any positive-dimensional components of the solution set, $\mathbf{V}(f)$. One of the most recent advances in this field is regeneration, an equation-by-equation solver that is often more efficient than other methods. However, the basic form of regeneration will not necessarily find all isolated singular solutions of a polynomial system.

In this article, we describe a technique for using regeneration to find all isolated solutions of a polynomial system, including all isolated singular solutions. This technique also yields the multiplicity of each isolated solution. This method – *perturbed regeneration* – slightly decreases the efficiency of regeneration while increasing its applicability. This two-part method consists of computing the solution of a perturbed problem via regeneration, followed by a generally inexpensive parameter homotopy to the target system $f(z)$. This article describes the use of this method to find all isolated solutions and briefly investigates the effect of this sort of perturbation on problems having positive-dimensional solution sets.

Keywords: Polynomial systems; homotopy continuation; regeneration; perturbation.
2010 MSC: 65H10,65H20

1. Introduction

Let $f := (f_1, \ldots, f_N) : \mathbb{C}^N \to \mathbb{C}^N$ be a polynomial system[3] with solution set

$$\mathbf{V}(f) := \left\{ z \in \mathbb{C}^N : f_i(z) = 0, \text{ for } i = 1, \ldots, N \right\}.$$

The field of *numerical algebraic geometry* [3, 25] includes a wide array of algorithms for finding and manipulating the solution sets $\mathbf{V}(f)$ of polynomial systems, including both isolated solutions (points) and positive-dimensional solution sets (curves, surfaces, etc.). We provide a very brief introduction to standard homotopy continuation in §2.1. These methods form the core computational engine for the field. One of the recent advances in this area is *regeneration*, a technique for finding the isolated solutions of $\mathbf{V}(f)$ efficiently.

*Principal corresponding author.

[1] Partially support by NSF DMS–1115668.

[2] NSF DMS-1228308, NSF DMS-1322508, DARPA N66001-11-1-4184, and the AFOSR.

[3] We make the assumption here that the system is square, i.e., the number of polynomials equals the number of variables. This is not actually necessary – there are simple methods based on Bertini's Theorem [3] to square up nonsquare systems – but we follow this convention in this article for cleaner exposition.

Regeneration is an "equation-by-equation" method. The solutions of $f_1(z)$ and $N-1$ random linear functions are used to solve $f_1(z)$ and $f_2(z)$, along with $N-2$ linear functions, and so on. This method was first introduced in [10] and extended to the case of positive-dimensional solution sets in [11]. In practice, regeneration is typically very efficient, often producing the nonsingular isolated solutions of a polynomial system much faster than standard homotopy methods, by automatically taking advantage of any symmetries or structure in $f(z)$. We present the basic method in §2.2.

Recall that a point $p \in \mathbb{C}^N$ is an *isolated solution* of $f(z)$ if $f(p) = 0$ and if there is some small ϵ such that $f(w) \neq 0$ for all points w in the punctured neighborhood

$$B_p(\epsilon) = \left\{ z \in \mathbb{C}^N : ||z - p|| < \epsilon \right\}.$$

Alternatively, p is an isolated solution if the local dimension of $V(f)$ at p is zero. An isolated solution p of $f(z)$ is said to be *singular* if the Jacobian matrix of $f(z)$ is not full rank when evaluated at p. All isolated solutions have associated to them a positive integer, the *multiplicity* of the solution, which is greater than 1 for singular solutions [3].

Basic regeneration, as presented in [10], is not guaranteed to find all singular, isolated solutions of $f(z)$. For example, as described in §2.3, basic regeneration fails to find the singular (multiplicity two) solution $(x, y) = (2, 3)$ of the system

$$\begin{bmatrix} y(x-2)^2 \\ x(y-3) \end{bmatrix},$$

though it does find the other solution, $(0, 0)$, which is nonsingular. This is an unfortunate drawback to regeneration since it is such an efficient technique. Thus, we are led to pose the following

Fundamental Problem: Modify regeneration to find a numerical approximation of each isolated point of $V(f)$, including isolated singular solutions.

The authors of [10] provide one solution to this problem. Their solution is rooted in deflation [22, 23, 16, 15]. Unfortunately, deflation causes an undesirable increase in the size of the polynomial system and is therefore rather costly, particularly for solutions of high multiplicity.

In this article, we provide a more efficient solution to the Fundamental Problem. There are two simple steps:

1. Find the isolated solution of a perturbation $\hat{f}(z)$ of $f(z)$.
2. Solve $f(z)$ by tracking the solutions as we deform from $\hat{f}(z)$ back to $f(z)$.

This method is the focus of §2 and is the main contribution of this article.

In §3, we describe the connection of this perturbation approach to the deflation approach of [10], the method of regenerative cascade [11], and a very early technique in the field known as the cheater's homotopy [18] in which the authors made use of a perturbation of $\hat{f}(z)$ for somewhat different reasons.

Finally, though we focus primarily on the use of perturbation to solve the Fundamental Problem, it is interesting to consider the effect on positive-dimensional irreducible components. It was observed in [18] and [25] that such a perturbation can cause positive-dimensional irreducible components to "break" into a (possibly very large) number of isolated solutions. In §4, we investigate this phenomenon and describe our attempts to extract from it useful information.

2. Perturbed regeneration for isolated solutions

Regeneration relies on a sequence of standard homotopies. We begin with a very brief overview of these central methods in §2.1. This overview is followed by a description of basic regeneration (§2.2) and an

illustration of the failure of this method to find some singular solutions (§2.3). Section 2.4 indicates the main algorithmic advance in this paper, followed by some justification (§2.5) and examples (§2.6).

2.1. Standard homotopy methods

Given a polynomial system $f : \mathbb{C}^N \to \mathbb{C}^N$, homotopy continuation is a three-step method for approximating all isolated solutions of $f(z) = 0$:

1. Choose a polynomial system $g : \mathbb{C}^N \to \mathbb{C}^N$ that is in some way "similar" to $f(z)$ but that is somehow easier to solve.
2. Solve $g(z) = 0$ and form the *homotopy function*

$$H : \mathbb{C}^N \times \mathbb{C} \to \mathbb{C}^N,$$

   given by

$$H(z;t) = (1-t)f(z) + tg(z),$$

   so that $H(z;1) = g(z)$ and $H(z;0) = f(z)$. Such a homotopy is sometimes denoted $g \to f$.
3. As t varies from 1 to 0, track the solutions of $H(z;t) = 0$, starting with the known solutions of $g(z) = 0$ and leading to the solutions of $f(z) = 0$. This can be accomplished via numerical predictor-corrector methods.

The key point in this section is that there are well-studied methods for computing numerical approximations of the isolated solutions of a polynomial system. The major software packages for carrying out such computations include Bertini [2], PHCpack [26], and HOM4PS-2.0 [14].[4]

2.2. Basic regeneration

This section outlines in broad strokes the regeneration homotopy method for computing the isolated nonsingular solutions of a polynomial system as first developed in [10] and stated succinctly for the nonsingular case in [11].

Let d_i denote the degree of polynomial f_i for $i = 1, \ldots, N$, and let $L_i^{(j)}(z)$ be a linear polynomial with randomly-chosen coefficients for each $2 \le i \le N$ and $1 \le j \le d_i$. We hereafter suppress the argument z to simplify notation.

Assume that we have already solved $f_1 = L_2^{(1)} = \ldots = L_N^{(1)} = 0$.[5] After this initial solve, consider the following sequence of homotopies

$$\begin{bmatrix} f_1 \\ L_2^{(1)} \\ L_3^{(1)} \\ \vdots \\ L_N^{(1)} \end{bmatrix} \to \begin{bmatrix} f_1 \\ L_2^{(2)} \\ L_3^{(1)} \\ \vdots \\ L_N^{(1)} \end{bmatrix} \to \cdots \to \begin{bmatrix} f_1 \\ L_2^{(d_2)} \\ L_3^{(1)} \\ \vdots \\ L_N^{(1)} \end{bmatrix}$$

Only the second function depends on parameter t as the other functions do not change.

[4] It is important to note that robust methods require fundamental tools such as adaptive precision, endgames, and the "gamma trick." See [25, 3] to learn more about the many subroutines, nuances, and variations. Also, see [13, 27, 17] for details on the related polyhedral homotopy approach to solving polynomial systems.

[5] This initial system can easily be reduced to a degree d_1 univariate polynomial and solved with univariate methods.

Let $S_1, \ldots, S_{d_2}$ be the sets of *nonsingular* solutions to each system in the sequence of homotopies above. We will describe shortly why we take only the nonsingular solutions.

The union of $S_1, \ldots, S_{d_2}$ is clearly the set of all isolated solutions of the system

$$\begin{bmatrix} f_1 \\ \prod_{j=1}^{d_2} L_2^{(j)} \\ L_3^{(1)} \\ \vdots \\ L_N^{(1)} \end{bmatrix},$$

for which we note that the second equation now has the same degree as $f_2(z)$.

The *regeneration* of f_2 is completed with the following homotopy:

$$\begin{bmatrix} f_1 \\ \prod_{j=1}^{d_2} L_2^{(j)} \\ L_3^{(1)} \\ \vdots \\ L_N^{(1)} \end{bmatrix} \rightarrow \begin{bmatrix} f_1 \\ f_2 \\ L_3^{(1)} \\ \vdots \\ L_N^{(1)} \end{bmatrix}.$$

It is important to note that homotopy paths starting at singular points cannot be tracked with simple predictor-corrector methods. It is precisely for this reason that S_i, defined above, may contain only *nonsingular* points. Deflation could be used to desingularize any singular start points (see §3.1), but this is a costly approach.

Once f_2 has been regenerated, we regenerate f_3 similarly, and so on. For more details see [10]. The final result is the set of all nonsingular isolated solutions in $\mathbf{V}(f)$, along with any singular solutions that come from paths that stay nonsingular until the final homotopy to f_N.

2.3. The need for a perturbation

A very simple example illustrates how regeneration can fail to find singular solutions. Consider the following polynomial system of equations:

$$\begin{bmatrix} y(x-2)^2 \\ x(y-3) \end{bmatrix}$$

It is easy to see that this system has isolated solutions $(0,0)$ and $(2,3)$. The solution $(2,3)$ is singular, with multiplicity two.

Consider the system after regenerating the first equation:

$$\begin{bmatrix} y(x-2)^2 \\ r_1 x + r_2 y + r_3 \end{bmatrix}$$

where $r_1, r_2, r_3 \in \mathbb{C}$ are random. This system has nonsingular solution $(0,0)$ and singular solution $(2, -2(r_1/r_2))$. This singular solution would therefore be discarded before moving on, leaving us to follow only the path originating from $(0,0)$.

Proceeding in the regeneration algorithm, we follow the homotopy

$$\begin{bmatrix} y(x-2)^2 \\ r_1x + r_2y + r_3 \end{bmatrix} \to \begin{bmatrix} y(x-2)^2 \\ s_1x + s_2y + s_3 \end{bmatrix}$$

where $s_1, s_2, s_3 \in \mathbb{C}$ are random. Finally, we complete regeneration via

$$\begin{bmatrix} y(x-2)^2 \\ (r_1x + r_2y + r_3)(s_1x + s_2y + s_3) \end{bmatrix} \to \begin{bmatrix} y(x-2)^2 \\ x(y-3) \end{bmatrix}$$

to arrive at only the nonsingular solution $(0, 0)$.

2.4. Algorithm

We remedy the problem of not finding the singular solutions of $f(z)$ by replacing $f(z)$ with a perturbed polynomial system $f_p(z) = f(z) - p$ for a randomly chosen point $p \in \mathbb{C}^N$. It may seem surprising that this trivial change to $f(z)$ could significantly alter the behavior of the solutions, but the result of this perturbation is that the singular solutions of $f(z)$ each become several isolated nonsingular solutions of $f_p(z)$. This is discussed in more detail in §2.5.

First, though, we present the pseudocode for the main algorithm of this article.

Algorithm 1 Main Algorithm: Perturbed Regeneration

Input: Polynomial system $f : \mathbb{C}^N \to \mathbb{C}^N$.
Output: Superset $\widehat{V}$ of all isolated solutions V of $f(z)$ or, optionally, V.

1. Choose random $p \in \mathbb{C}^N$.
2. Use regeneration to find all isolated nonsingular solutions T of $f_p(z) = f(z) - p$.
3. Follow all paths beginning at points of T through homotopy $f(z) - tp$, letting t go from 1 to 0, storing all finite endpoints in $\widehat{V}$.
4. (Optional) Remove from $\widehat{V}$ all non-isolated $z \in \widehat{V}$ via a local dimension test to produce V. There are several options for this local dimension test. One standard choice was first described in [1].

Naturally, any technique for optimizing basic regeneration will also reduce run time for perturbed regeneration. For example, ordering of the polynomials by degree has an impact on run time for regeneration, so the same can be said in the perturbed case. This and other optimizations of regeneration are described in [10].

2.5. Justification

Much of the theory underlying the ideas of this article was known by Morgan and Sommese in the 1980s [19] and has since been repeated in various forms, for example in [25, 9]. The main contribution of this article is in the application of this theory in the setting of regeneration, not in the theory itself. In this section, we provide justification for the correctness of the algorithm, pointing to appropriate sources for proofs and further background.

Let $\mathrm{rk}(f)$ denote the rank of the polynomial system $f(z)$, i.e., the dimension of the closure of the image of $f(z)$, $\overline{f(\mathbb{C}^N)} \subseteq \mathbb{C}^N$. The rank of $f(z)$ is an upper bound on the codimension of the irreducible components of $\mathbf{V}(f)$ [25]. Thus, $f(z)$ may only have isolated solutions if $\mathrm{rk}(f) = N$.

Theorem 2.1. *For a polynomial system $f : \mathbb{C}^N \to \mathbb{C}^N$, the procedure of Algorithm 1 produces, with probability one, a superset of numerical approximations to all isolated solutions of $f(z)$.*

There are three key facts supporting Theorem 2.1:

Lemma 2.2. *Given a polynomial system as in Theorem 2.1, there is a Zariski open subset W of $f(\mathbb{C}^N)$ such that for $p \in W$, the solution set of $f(z)-p$ consists of only smooth irreducible components of dimension $N - rk(f)$. In the case that $rk(f) = N$, $f(z) - p$ will have only nonsingular isolated solutions.*

Lemma 2.3. *If $rk(f) = N$, then f is a dominant map, i.e., $\overline{f(\mathbb{C}^N)} = \mathbb{C}^N$.*

Lemma 2.4. *In the case that $rk(f) = N$, for each isolated solution w of $f(z)$, there is, with probability one, at least one path starting at a solution of $f(z) - p$ and ending at w as $p \to 0$.*

Before discussing the justification of these lemmas, we provide a simple proof of the main result, Theorem 2.1.

Proof of Theorem 2.1. The statement is vacuously true if there are no isolated solutions, so we assume $\mathrm{rk}(f) = N$. According to Lemma 2.2 and in light of Lemma 2.3, almost all perturbations $p \in \mathbb{C}^N$ of $f(z)$ will result in a polynomial system having only nonsingular isolated solutions. For some specific choice $\hat{p} \in \mathbb{C}$, we refer to this set of solutions as $\mathbf{V}(f(z) - \hat{p})$. Regeneration can compute all of these nonsingular solutions [10].

Lemma 2.4 then guarantees that, for each isolated solution q of $f(z) = 0$, there is a homotopy path beginning from some point in $\mathbf{V}(f(z) - \hat{p})$ that ends at q, as t moves from 1 to 0. Thus, all isolated solutions of $f(z)$, including singular isolated solutions, will be produced as output by Algorithm 1. □

Remark 2.5. Note that we find *a priori* a superset of the isolated solutions of $f(z) = 0$, not the set itself. This is because points on positive-dimensional components may also be found by Algorithm 1 (see §4 for more on this). As mentioned in §2.4, there are known methods for removing such points, if desired.

Generalizations of all three lemmas appear in Appendix A of [25] as consequences of an algebraic version of Sard's Theorem. Indeed, Lemma 2.2 is proved as Theorem A.6.1 in [25]. Similarly, Lemma 2.4 is proven in more generality as Corollary A.4.19 of [25].

A more general statement than Lemma 2.3 is given as an exercise in [8] and a related result for a pure d-dimensional algebraic subset is presented in [9], for $d > 0$. For the specific setting of the lemma, the proof is trivial:

Proof of Lemma 2.3. Since $\mathbf{V}(f)$ contains a pure 0-dimensional algebraic subset, we must have that the rank of $\mathrm{rk}(f) = N$. So f is full rank and, equivalently, f is dominant. □

Now that we have completed the justification of Theorem 2.1, we may discuss a few extensions.

First, we may trivially compute the multiplicity, $\mu(z_i)$, of each isolated solution z_i of $f(z) = 0$, as defined in [25]:

Corollary 2.6. *Algorithm 1 produces not only the isolated solutions of $f(z) = 0$ but also the multiplicity $\mu(z_i)$ of each solution z_i.*

This is based on the fact, proved as Theorem A.14.1(3) in [25], that each isolated solution z_i will be the endpoint of $\mu(z_i)$ paths beginning at points in $\mathbf{V}(f - \hat{p})$.

Second, we may consider the case of non-square system $f : \mathbb{C}^N \to \mathbb{C}^n$, with $N \neq n$. Of course, if $n < N$, there are no isolated solutions, so there is nothing to say for that case.

If $n > N$, the standard technique in numerical algebraic geometry is to randomize $f(z)$ down to $A \cdot f(z)$, where $A \in \mathbb{C}^{N \times n}$ is a randomly chosen matrix of complex numbers. This yields a system of N equations

and variables, with the property that $\mathbf{V}(f(z)) \subset \mathbf{V}(A \cdot f(z))$. We also have that isolated solutions in $\mathbf{V}(f(z))$ will be isolated solutions in $\mathbf{V}(A \cdot f(z))$, though the multiplicity of solutions having multiplicity greater than one (with respect to $f(z)$) might increase. It is easy to filter any "new" isolated solutions of $\mathbf{V}(A \cdot f(z))$ that are not actually solutions of $f(z) = 0$, simply by evaluating each isolated solution in the polynomials $f(z)$.

Thus, Algorithm 1 can easily be extended to find all isolated solutions of non-square polynomial systems. Note, of course, that the computed multiplicity of a solution of a non-square system that has been randomized down to a square system might be larger than the multiplicity of that solution with respect to the original system.

2.6. Examples and timings

In this section we consider several examples where perturbed regeneration provides some benefit. All reported timings come from runs on a 3.2 GHz core of a Dell Precision Workstation with 12 GB of memory.

2.6.1. A very simple illustrative example

Let's first consider the simple example from §2.3 to illustrate Algorithm 1. Recall the system,

$$f(x, y) = \begin{bmatrix} y(x-2)^2 \\ x(y-3) \end{bmatrix}$$

which had isolated solutions $(0, 0)$ and $(2, 3)$. Basic regeneration for isolated solutions will not find the solution $(2, 3)$ because it is singular with multiplicity 2.

For perturbed regeneration, we first solve the perturbed system,

$$f_p(x, y) = \begin{bmatrix} y(x-2)^2 - p_1 \\ x(y-3) - p_2 \end{bmatrix},$$

using basic regeneration, where $p = (p_1, p_2) \in \mathbb{C}^2$ is chosen randomly. Suppose $p = (-0.521957 + 0.810510i, -0.0312394 - 0.602051i)$, then the perturbed system $f_p(x, y)$ has three solutions, approximated as:

$$(x, y) = (2.289555226266273 - 0.4818472697343892i, 3.039927414618357 - 0.2545525699394496i),$$

$$(x, y) = (1.696540899823075 + 0.4894908386839950i, 2.888481735852828 - 0.3226942032443160i),$$

$$(x, y) = (0.024317020583462 + 0.1930401252456662i, -0.09013793095821 - 0.2274320305094367i).$$

Then we use the homotopy

$$h(x, y; t) = \begin{bmatrix} y(x-2)^2 - tp_1 \\ x(y-3) - tp_2 \end{bmatrix}$$

to deform the solutions to $f_p(x, y)$ to solutions of $f(x, y)$. Two solutions above converge to $(2, 3)$; the other converges to $(0, 0)$.

2.6.2. An example with several isolated singular solutions

Next, we consider the system cpdm5, from the repository of systems [26] but originally considered in [7]. This system has five equations and five variables, with solutions having multiplicities 1, 5, 10, 20, and 30. Here, again, basic regeneration missed all of the singular solutions. Timings are provided in Table 1.

It is easy to replace basic regeneration in Step 2 of Algorithm 1 with some other homotopy method, e.g., a total degree homotopy. It is perhaps interesting to note that the timings for the perturbed runs (regeneration or total degree) vary much less than those of the unperturbed runs.

Homotopy Type	Time (seconds)
Perturbed regeneration	3.5
Total degree	1.83
Perturbed total degree	1.86

Table 1: Run times for the cpdm5 problem. Each timing is an average over 10 runs.

A priori, users may wish to use regeneration. While most examples of this section show that perturbed regeneration should be used instead, this example further shows that total degree (or perturbed total degree) can sometimes be faster.

2.6.3. Another example with many singular isolated solutions

In the article [21], Morrison and Swinarski study a polynomial system with 13 equations, having 51 isolated solutions. All of these solutions are singular, 30 with multiplicity 2, 20 with multiplicity 8, and one with multiplicity 32.

Basic regeneration failed to find any of the solutions, but perturbed regeneration found all of them. Some timings are provided in Table 2.

Homotopy Type	Time (seconds)
Perturbed regeneration:	24
Total degree	17
Perturbed total degree	11
2-Homogeneous	8
Perturbed 2-Homogeneous	7

Table 2: Run times for the Morrison-Swinarski problem. Each timing is an average over 10 runs.

Here again, while perturbed regeneration finds all the solutions, it is not the most efficient method. As with the previous example, total degree (and perturbed total degree) are more efficient. A more specialized sort of homotopy, the 2-homogeneous homotopy [25], performs even better in this case.

2.6.4. The Butcher problem: Positive-dimensional components

Finally, we consider the following system, originally due to C. Butcher,

$$f = \begin{bmatrix} zu + yv + tw - w^2 - 1/2w - 1/2 \\ zu^2 + yv^2 - tw^2 + w^3 + w^2 - 1/3t + 4/3w \\ xzv - tw^2 + w^3 - 1/2tw + w^2 - 1/6t + 2/3w \\ zu^3 + yv^3 + tw^3 - w^4 - 3/2w^3 + tw - 5/2w^2 - 1/4w - 1/4 \\ xzuv + tw^3 - w^4 + 1/2tw^2 - 3/2w^3 + 1/2tw - 7/4w^2 - 3/8w - 1/8 \\ xzv^2 + tw^3 - w^4 + tw^2 - 3/2w^3 + 2/3tw - 7/6w^2 - 1/12w - 1/12 \\ -tw^3 + w^4 - tw^2 + 3/2w^3 - 1/3tw + 13/12w^2 + 7/24w + 1/24 \end{bmatrix},$$

first appeared in [26]. Computing the numerical irreducible deomposition [3, 25], the solution set consists of 10 irreducible components of various dimensions, provided in Table 3. All isolated solutions are nonsingular.

Dimension	Components	Degree
3	3	1
2	2	1
0	5	1

Table 3: Summary of the solution set of the Butcher problem.

When basic regeneration is applied to this system, only the five nonsingular points are approximated. Thus, in this case, basic regeneration finds all isolated solutions. If perturbed regeneration is applied, there are eleven nonsingular points in the solution set of the perturbed system. Five points converge to nonsingular solutions, two points converge to the positive-dimensional components, and the remaining diverge. Note that a total degree homotopy will also find points on the positive-dimensional components, but computation time increases, since hundreds of points converge to the positive-dimensional components.

Homotopy Type	Time (seconds)
Perturbed regeneration	31
Basic regeneration	32
Regenerative cascade	100
Total degree	1097

Table 4: Run times for the Butcher problem. Each timing is an average over 10 runs.

Table 4 shows the timings for this problem using a total degree homotopy, regenerative cascade, basic regeneration, and perturbed regeneration. As opposed to the previous examples, perturbed regeneration was faster for this problem, even faster than the shorter basic regeneration method. This is due to the fact that basic regeneration encounters singular solutions on positive-dimensional components throughout the algorithm, which slows down the path tracker. The perturbed system has only nonsingular isolated solutions, which can be handled much more efficiently. It is interesting to note that more paths were tracked with perturbed regeneration than with any of the other methods.

3. Connections to related techniques

As mentioned above, the theory behind perturbed regeneration is not new and other alternatives to perturbed regeneration exist for using regeneration to find isolated singular solutions. Perhaps the earliest reference to this sort of perturbation for homotopy methods was the cheater's homotopy, described briefly

in §3.3. In this section, we very briefly describe these related techniques and indicate the differences between them and perturbed regeneration.

3.1. Regeneration with deflation

As outlined in [10], regeneration techniques can be combined with deflation to find singularities. Deflation is a technique that replaces a polynomial system f on $\mathbb{C}^N$ and an isolated singular solution x^* with a new polynomial system $f(x,\xi)$ on $\mathbb{C}^N \times \mathbb{C}^M$ with an isolated nonsingular soluition (x^*, ξ^*). For more on deflation in the polynomial setting see [16, 15] and in a more general context [22, 23]. A more recent and more general version of deflation is called isosingular deflation [12]. To find singularities using regeneration, deflation must be applied to any intermediate system that has a singular solution. Each application of deflation increases the number of variables and equations in the system, thus making computation more difficult. Algorithm 1 will find isolated singular solutions while avoiding intermediate deflation steps.

3.2. Regenerative cascade

The regenerative cascade of [11] provides an equation-by-equation approach to computing the numerical irreducible decomposition of the solution set of a polynomial system. A consequence of this method is that isolated singular solutions of the system will also be identified. However, if only isolated solutions are of interest, this information comes at a significant cost increase, namely the cost of cascading through a number of dimensions. Perturbed regeneration avoids this cost, but if the complete information provided by a numerical irreducible decomposition is desired, the regenerative cascade is clearly the better choice.

3.3. Parameter homotopy and the cheater's homotopy

Parameterized polynomial systems $f(v,p)$ arise with some frequency in applications, so it is sometimes useful to solve the same polynomial system at numerous points $p = p_1, \ldots, p_k$ in some parameter space. Here, v is the set of variables and p is the set of parameters. An efficient approach to such a problem is to approximate the solutions of $f(v,\hat{p})$ where $\hat{p}$ is a generic complex number, then use these solutions as start points of a *parameter homotopy* between $f(v,\hat{p})$ and $f(v,p_i)$ for $i = 1,\ldots,k$. This reduces the number of paths which must be followed to compute the solutions at each parameter point p_i to the number of solutions at a generic parameter point. This idea has been implemented in *Bertini* [2] and *Paramotopy* [5]. Some background may be found in [20] and [18].

The trick to such methods is choosing an intermediate system $f(v,\hat{p})$ which satisfies some necessary properties, including that the solutions are smooth. The cheaters homotopy in [18] addresses this issue by including the same perturbation parameter as in Lemma 2.2. In that case, the primary motivation for using such a perturbation is to have smooth solutions as start points of another homotopy. We require the smooth solutions so that regeneration can be used to compute the approximations. Thus, the methods are quite similar but have different goals.

4. The effect of perturbation on positive-dimensional irreducible components

The focus of this article is the extension of basic regeneration to find all isolated solutions. However, it is natural to consider the effect of this method on positive-dimensional solution components. This issue has arisen previously [25], but there has never been a careful, thorough analysis. Unfortunately, there is actually rather little to conclude.

4.1. Failure to find the numerical irreducible decomposition

The *numerical irreducible decomposition* is the data type used in numerical algebraic geometry to store positive-dimensional solution sets. The technical definition is not necessary here but may be found in [3, 25]. The main idea is that, for each irreducible component Z of degree d_Z, we find numerical approximations to d_Z generic points on Z. These arise as the intersection of Z with a linear space of complementary dimension.

There are a number of methods for computing the numerical irreducible decomposition of the solution set of a polynomial system. It is tempting to try to use the perturbation method of this article to compute such a decomposition in just one step. Given system $f(z)$, the idea would be to solve a perturbed system $\hat{f}(z)$ for which all irreducible components have been broken into points, then move from $\hat{f}(z)$ back to $f(z)$ with some number of points landing on each irreducible component. If desired, monodromy and the trace test [3, 24] could be used to find d_Z points on each component Z.

At first, there seems to be some hope for this. As described in [19], such a perturbation will lead to at least one point per connected (complex) component. Of course, with all of this, we must assume that f has full rank, or else the perturbation does not break positive-dimensional components into points.

However, a simple example shows that this method will not work in general. Let

$$f(x,y) = \begin{bmatrix} x^2 \\ xy \end{bmatrix}.$$

The solution set of this polynomial system is just the line $x = 0$, with a singular embedded point at the origin. It is easy to see that the four points from any perturbed system will necessarily all go to the origin, not a generic point on $x = 0$.

This can similarly be seen from the Butcher problem of §2.6.4. The solution set for that problem consists of five positive-dimensional components, but the perturbation yields only two points on those five components.

Based on these examples, it seems that there is little hope of directly computing a numerical irreducible decomposition via this sort of perturbation. Perhaps there is some modification of the ideas of this article that will yield the numerical irreducible decomposition in a similar manner, but such a modification is not treated in this paper.

4.2. Extracting useful information

It would be interesting to know exactly how many points come from each component when breaking an algebraic set into points via perturbation. Of course, if $f(z)$ is not full rank, each component will break into some number of positive-dimensional components, possibly of lower dimension.

Unfortunately, given that there is not even a guarantee that there will be even one point per irreducible component, this is a moot point. The solution of Morgan and Sommese [19] seems to be the best we can hope for – there is at least one point on each connected component – unless perhaps more conditions are added to the polynomial system. However, this is beyond the scope of this paper.

5. Conclusions

In this article, we presented a variation of regeneration that will yield all isolated solutions of a polynomial system, not just those that are nonsingular. We refer to this method as *perturbed regeneration*. While this sort of perturbation is not new, it is new in the context of regeneration and provides a significant improvement to the output of basic regeneration for only a little more computational cost. While it

would be interesting to better understand how this perturbation affects positive-dimensional irreducible components, the results of the previous section seem to indicate that there is little that can be said in general.

References

[1] D.J. Bates, J.D. Hauenstein, C. Peterson, and A.J. Sommese, A local dimension test for numerically approximated points on algebraic sets. *SIAM J. Num. An.* 47:3608–3623, 2009.

[2] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler, *Bertini: Software for Numerical Algebraic Geometry*, Software available at `www.bertini.nd.edu`, 2006.

[3] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler, *Numerically Solving Polynomial Systems with Bertini*, *SIAM*, 2013.

[4] W. Boege, R. Gebauer, and H. Kredel. Some examples for solving systems of algebraic equations by calculating Groebner bases. *J. Symbolic Computation*, 2:83-98, 1986.

[5] D. Brake, M. Niemerg, and D. Bates. Paramotopy: parallel parameter homotopy through bertini, Software available at `www.paramotopy.com`, 2013.

[6] C. Butcher. An application of the Runge-Kutta space. BIT, 24:425-440, 1984.

[7] K. Gatermann. Symbolic solution of polynomial equation systems with symmetry. *Proceedings of ISSAC-90*, pp. 112–119, ACM, New York, 1990.

[8] R. Hartshorne. *Algebraic Geometry*. Springer-Verlag, New York, 1977.

[9] Jonathan D. Hauenstein. Numerically computing real points on algebraic sets. *Acta Applicandae Mathematicae* 125(1):1–15, 2013.

[10] Jonathan D. Hauenstein, Andrew J. Sommese, and Charles W. Wampler. Regeneration homotopies for solving systems of polynomials, *Mathematics of Computation* 80:345–377, 2011.

[11] Jonathan D. Hauenstein, Andrew J. Sommese, Charles W. Wampler. Regenerative cascade homotopies for solving polynomial systems, *Applied Mathematics and Computation* 218(4):1240–1246, 2011.

[12] Jonathan D. Hauenstein and Charles W. Wampler. Isosingular sets and deflation. *Foundations Comp. Math.* 13(3):371–403, 2013.

[13] B. Huber and B. Sturmfels, A polyhedral method for solving sparse polynomial systems, *Math. Comp.* 64(212):1541–1555, 1995.

[14] T.L. Lee, T.Y. Li, and C.H. Tsai, HOM4PS–2.0, A software package for solving polynomial systems by the polyhedral homotopy continuation method, *Computing*, 83(2–3):109–133, 2008.

[15] Anton Leykin, Jan Verschelde, Ailing Zhao. Newton's method with deflation for isolated singularities of polynomial systems. *Theoretical Computer Science* 359(1–3):111–122, 2006.

[16] A. Leykin, J. Verschelde, A. Zhao. Higher-Order Deflation for Polynomial Systems with Isolated Singular Solutions. *Algorithms in Algebraic Geometry* 147:79–97, 2008.

[17] T.Y. Li, Numerical solution of polynomial systems by homotopy continuation methods, in *Handbook of Numerical Analysis. Volume XI. Special Volume: Found. Comp. Math.*, edited by F. Cucker, North-Holland, 2003, pp. 209–304.

[18] T.Y. Li, Tim Sauer, J.A. Yorke. The Cheaters Homotopy: An Efficient Procedure For Solving System Of Polynomial Equations, *SIAM J. Numer. Anal.* 26(5):1241–1251, 1989.

[19] A.P. Morgan and A.J. Sommese. Computing all solutions to polynomial systems using homotopy continuation, *Applied Math. Comp.* 24:115–138, 1987.

[20] A.P. Morgan and A.J. Sommese. Coefficient-parameter polynomial continuation, *Applied Math. Comp.* 29(2):123–160, 1989.

[21] I. Morrison and D. Swinarski. Can you play a fair game of craps with a loaded pair of dice? Submitted, 2013.

[22] Takeo Ojika, Satoshi Watanabe, Taketomo Mitsui. Deflation algorithm for the multiple roots of a system of nonlinear equations, *Journal of Mathematical Analysis and Applications* 96(2): 463–479, 1983.

[23] Takeo Ojika. Modified deflation algorithm for the solution of singular problems. I. A system of nonlinear algebraic equations, *Journal of Mathematical Analysis and Applications* 123(1):199–221, 1987.

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

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

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

[27] J. Verschelde, P. Verlinden, and R. Cools, Homotopies exploiting Newton polytopes for solving sparse polynomial systems, *SIAM J. Numer. Anal.*, 31(3):915–930, 1994.