

Newton's Method with Deflation for Isolated Singularities of Polynomial
Systems

BY

AILING ZHAO

B.E., Taiyuan University of Science and Technology, 1993
M.S., University of Illinois at Chicago, 2001

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mathematics
in the Graduate College of the
University of Illinois at Chicago, 2007

Chicago, Illinois

Copyright by

Ailing Zhao

2007

This thesis is dedicated to my parents Xiangyang Zhao and Guimei Xue,
my husband Yinwan Li, my son Kevin and my daughter Maggie.

ACKNOWLEDGMENTS

I would like to thank everyone who was involved in this work and give me every kind of help and encouragement that makes it possible to be finished. I would like to share my joy and excitement with them.

I want to thank Anton Leykin for his selflessly helping for my research and study.

I especially want to thank Dr. Zhonggang Zeng for his valuable suggestions and discussion at Notre Dame and Xi'an.

I would like to thank other committee members: Shmuel Friedland, David Nicholls and Jeremy Teitelbaum.

I would like to express my deep gratitude to my thesis advisor, Dr. Jan Verschelde, for giving me tremendous guide on research. Without his inspiration and support, I wouldn't have my achievement today. During the academic year 2002-2005, I was supported as research assistant by NSF Award 0105739, NSF CAREER Award 0134611 and NSF Award 0410036. These grants provided me with equipment and gave me travel support.

I also want to give my thanks to the Department of Mathematics, Statistics, and Computer Science at UIC for its support and enjoyable environment for my study and research. Also thank the department and WISE for contributions to my trip to International Workshop on Symbolic-Numeric Computation on July 19-21, 2005, in Xi'an, China and 113th Annual Meeting of AMS on January 5-8, 2007, in New Orleans, LA.

ACKNOWLEDGMENTS (Continued)

This material is based upon work supported by the National Science Foundation under Grant No. 0105739, Grant No. 0134611 and Grant No. 0410036.

Finally, I want to thank all my family members. I thank my parents and my sisters for their love and constant support. I especially thank my husband, Yinwan Li, for his support, understanding and patience.

TABLE OF CONTENTS

<u>CHAPTER</u>	<u>PAGE</u>
1 INTRODUCTION	1
1.1 Motivation	1
1.2 Newton's Method	5
1.3 Homotopy Continuation Method	8
1.4 Related Work	10
1.5 Contributions of this Thesis	13
2 NEWTON'S METHOD WITH DEFLATION FOR ISOLATED SINGULARITIES OF POLYNOMIAL SYSTEMS	14
2.1 A Modified Deflation Method	14
2.2 A Bound on the Number of Deflations	16
2.2.1 Standard Bases for Local Orderings	18
2.2.2 Understanding the Deflation Method	21
2.2.2.1 A Symbolic Deflation Method	22
2.2.2.2 A Numeric Deflation Method	24
2.3 A Symbolic-Numeric Implementation	28
2.3.1 Special Case Implementations and Extensions	28
2.3.2 A Posteriori Validation of the Numerical Rank	29
2.3.3 Avoiding the Expression Swell	30
3 EVALUATION OF JACOBIAN MATRICES FOR NEWTON'S METHOD WITH DEFLATION	31
3.1 Problem Statement	31
3.2 Unwinding the Multipliers	33
3.3 A Directed Acyclic Graph of Jacobian Matrices	34
3.4 The Deflation Algorithm in <i>PHCmaple</i>	39
4 HIGHER ORDER DEFLATION FOR POLYNOMIAL SYSTEMS WITH ISOLATED SINGULAR SOLUTIONS	43
4.1 Statement of the Main Theorem & Algorithms	43
4.2 Multiplicity Structure	48
4.2.1 Standard Bases	49
4.2.2 Dual Space of Differential Functionals	50
4.2.3 Dual Bases versus Standard Bases	51
4.3 Computing the Multiplicity Structure	53
4.3.1 The Dayton-Zeng Algorithm	54
4.3.2 The Stetter-Thallinger Algorithm	55

TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>		<u>PAGE</u>
4.4	Proofs and Algorithmic Details	59
4.4.1	First Order Deflation	59
4.4.2	Higher Order Deflation with Fixed Multipliers	61
4.4.3	Indeterminate Multipliers	65
4.5	Computational Experiments	66
4.5.1	A First Example	67
4.5.2	A Larger Example	68
4.6	Conclusion	69
5	APPLICATIONS AND NUMERICAL RESULTS	71
5.1	Applications	71
5.2	Numerical Results	73
5.3	Conclusions	75
	APPENDICES	76
	Appendix A	77
	CITED LITERATURE	83
	VITA	90

LIST OF TABLES

<u>TABLE</u>		<u>PAGE</u>
I	TANGENT LINES TWO SOLUTIONS (REAL PART)	6
II	TANGENT LINES TWO SOLUTIONS (IMAGINARY PART) . . .	6
III	GROWTH OF THE NUMBER OF DISTINCT DERIVATIVE OPERATORS, AS NODES IN A DIRECTED ACYCLIC GRAPH, FOR INCREASING DEFLATIONS K	39
IV	NUMERICAL RESULTS ON A COLLECTION OF TEST SYSTEMS	74

LIST OF FIGURES

<u>FIGURE</u>		<u>PAGE</u>
1	Tangent lines for singular configuration (left) and regular configuration (right)	3
2	Flowchart for Newton's method.	7
3	Maple procedure for solving the linear system $Ax = b$ by SVD	8
4	Maple procedure for one step Newton method	9
5	Maple procedure for Newton iteration	9
6	Flowchart for a modified deflation method.	17
7	Maple procedure for adding equations	17
8	Two different staircases of the standard basis of \mathcal{I} with respect to different local orderings $\leq_{(-1,-2)}$ (at the left) and $\leq_{(-2,-1)}$ (at the right). Monomials generating $\mathbf{in}(\mathcal{I})$ are represented by black disks, while the standard monomials are shown as empty circles.	20
9	Directed acyclic graph illustrating the dependency relations in the evaluation of A_3	38
10	Two monomial staircases for two different monomial orderings applied to the same system. The full circles represent the generators of the initial ideals. The multiplicity is the number of standard monomials, represented by the empty circles under the staircase.	50

LIST OF MATHEMATICS SYMBOLS

The symbols used in the thesis with their explanations are listed below. Different meanings for the same symbol occur when there is no confusion from context.

F	Polynomial System
f	Polynomial Equation
\mathbf{x}	Variables
x	Variable
N	Number of Equations
n	Number of Variables
m	Multiplicity
$A(\mathbf{x})$	Jacobian Matrix
SVD	Singular Value Decomposition
R	Numerical Rank
ϵ	Tolerance for Numerical Rank

SUMMARY

Newton's method slows down when approaching a singular root and convergence may even be lost if the working precision and accuracy of input coefficients are not at least as high as the multiplicity of the root multiplied with the distance of the current approximation to the root. To restore the quadratic convergence of Newton's method, in 1983, T. Ojika, S. Watanabe, and T. Mitsui developed a symbolic-numerical deflation algorithm. More recently, in 2002, G. Lecerf proposed a symbolic algorithm to restore the quadratic convergence of Newton's method. In this thesis, we implemented and experimented with two modifications to the deflation algorithm of Ojika et al:

1. Instead of replacing equations of the original system by conditions from derivatives of the system, we propose to add equations, introducing random constants for uniform treatment.
2. Instead of using Gaussian Elimination, we propose to apply Singular Value Decomposition to determine the numerical rank of the Jacobian matrix. The threshold on the numerical rank is our only critical numerical parameter to decide whether to deflate or not to deflate.

In particular, if the numerical rank of the Jacobian matrix at the current approximation equals R , we introduce $R + 1$ additional variables which serve as multipliers to selections of random combinations of columns of the Jacobian matrix. We showed that no more than $m - 1$ successive deflations are needed to restore the quadratic convergence of Newton's method converging to an isolated root of multiplicity m . Once the precise location of the isolated singularity is known,

SUMMARY (Continued)

numerical techniques allow the calculation of the multiplicity structure, using the methods of Dayton-Zeng or Stetter-Thallinger.

Our algorithm is a symbolic-numeric algorithm, as we produce by numerical means new equations which regularize or recondition the problem. In particular, our algorithm gives a new system of polynomial equations for which the isolated singular solution is a regular root. We developed a modified Newton's method in Maple, exploiting its facilities for polynomial manipulations and convenient multiprecision arithmetic, and then implemented in PHCpack. While the performance of the method is promising on selected applications, the method suffers from expression swell after a couple of deflations. Therefore we describe a way to “unfold” the extra multiplier variables, exploiting the special structure of the matrices which arise in the deflation process.

Recently, a new idea coming from computing the multiplicity structure at a singular isolated solution gives rise to our new higher order deflation method. Using higher order partial derivatives of the original polynomials, the new algorithm reduces the multiplicity faster than the original one for the systems, where several first order deflation steps are necessary.

CHAPTER 1

INTRODUCTION

1.1 Motivation

To give a quick insight into the purpose of our modified Newton's method with deflation, let's look at three polynomial systems which have singular solutions. In this section, we describe the problems, in section 5.1, we describe the solutions by deflation.

Example 1.1.1 *A simple monomial system (from [68] and [70]):*

$$\begin{cases} x^2 = 0; \\ xy = 0; \\ y^2 = 0. \end{cases} \quad (1.1)$$

This system is a “monomial ideal” and trivial for computer algebra. Because we can see that origin is an isolated singular root with multiplicity 3 by staircase of the standard basis of the ideal $I = \{xy, x^2, y^2\}$. It can be solved using Newton's method by Singular Value Decomposition (SVD), but it is impossible to see the multiplicity of origin by perturbation.

For overdetermined systems, to use continuation methods, there is a procedure to obtain a square system, see [68]. For this special overdetermined system, the randomized square system is:

$$\begin{cases} x(y + \mu_1 x) = 0; \\ y(x + \mu_2 y) = 0. \end{cases} \quad (1.2)$$

It has the origin as an isolated solution of multiplicity 4. So, we cannot apply homotopy continuation method to it directly by randomization. From the proposition 2. 2. 1 of [68], we know that if the multiplicity of the solution is more than one, then the randomization doesn't help for overdetermined system.

Example 1.1.2 *The 4-fold cyclic-9 roots are part of the solution set of the system:*

$$\begin{cases} f_i = \sum_{j=0}^8 \prod_{k=1}^i x_{(k+j) \bmod 9} = \mathbf{0}; \\ f_9 = x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 - 1 = 0. \end{cases} \quad (1.3)$$

This system is a widely used benchmark in the field of polynomial system solving, e.g., [3; 4], [16], [18], [29], [47], with theoretical results in [30]. There are 5,594 (333 orbits of size 18) isolated regular cyclic 9-roots, in addition to the 162 isolated solutions of multiplicity four. If Newton's method with 64 decimal places is used for this system and if the tolerance is 10^{-60} , then for regular solutions, 4 iterations (quadratic convergence) is enough, but for isolated singular solutions, 79 iterations (> 1 step for one correct decimal place) is needed. This is about 20 times slower to reach same magnitude of residual.

Example 1.1.3 *Lines tangent to a special configuration of four spheres [71]:*

The problem is that finding all lines tangent to all 4 given spheres with same radius.

The original problem is classical for applications in “tomography”, it is common tangents to four spheres which were analyzed in [15] and [32]. For spheres of equal radius, it was studied by Macdonald, Pach, Theobald [52], and Sottile [71].



Figure 1. Tangent lines for singular configuration (left) and regular configuration (right)

First let’s look at a special case, as shown in the left of Figure 1: in which the positions of the centers of the spheres are located at each of the four vertices of a tetrahedron, say, $c_1 = (0, 0, 0)$, $c_2 = (1, 0, 0)$, $c_3 = (1/2, \sqrt{3}/2, 0)$, and $c_4 = (1/2, \sqrt{3}/6, \sqrt{6}/3)$, and each sphere

has radius $1/2$. In this case, the 12 solutions are grouped into 3, each has a multiplicity of 4. Reflected in the left of Figure 1, only 3 lines (but each of multiplicity 4) are observed. If we slightly perturb the positions of the centers of this special case, then, as shown in the right of Figure 1, there are 12 real solutions generally with multiplicity one.

To write out the equations explicitly, we can see [15], Durand gave a systematic algebraic treatment for regular cases. For this special case, following [52], let the tangent vector be $t = (x_0, x_1, x_2)$ and moment vector $m = (x_3, x_4, x_5)$. Then the first equation is $\|t\| = 1$, the second is $m \cdot t = 0$, and the other equations are $\|m - c_i x \times t\|^2 - r^2 = 0$, where the radius $r = 1/2$. The coefficients algebraic numbers $\sqrt{3}$, $\sqrt{6}$, etc. can be approximated by double floats. Multiprecision doesn't help for algebraic numbers $\sqrt{3}$, $\sqrt{6}$, it helps only if we expand the numbers $\sqrt{3}$, $\sqrt{6}$ with sufficient precision. The expanded polynomial system is

$$\left\{ \begin{array}{l} x_0^2 + x_1^2 + x_2^2 - 1 = 0; x_0x_3 + x_1x_4 + x_2x_5 = 0; x_3^2 + x_4^2 + x_5^2 - 0.25 = 0; \\ x_3^2 + x_4^2 - 2x_2x_4 + x_2^2 + x_5^2 + 2x_1x_5 + x_1^2 - 0.25 = 0; \\ x_3^2 + 1.73205080756888x_2x_3 + 0.75x_2^2 + x_4^2 - x_2x_4 + 0.25x_2^2 + x_5^2 \\ -1.73205080756888x_0x_5 + x_1x_5 + 0.75x_0^2 - 0.86602540378444x_0x_1 + 0.25x_1^2 - 0.25 = 0; \\ x_3^2 - 1.63299316185545x_1x_3 + 0.57735026918963x_2x_3 + 0.666666666666667x_1^2 \\ -0.47140452079103x_1x_2 + 0.083333333333333x_2^2 + x_4^2 + 1.63299316185545x_0x_4 - x_2x_4 \\ +0.666666666666667x_0^2 - 0.81649658092773x_0x_2 + 0.25x_2^2 + x_5^2 - 0.57735026918963x_0x_5 \\ +x_1x_5 + 0.083333333333333x_0^2 - 0.28867513459481x_0x_1 + 0.25x_1^2 - 0.25 = 0. \end{array} \right. \quad (1.4)$$

This system is an approximate input system in which some of the coefficients of polynomials have limited accuracy as called in "empirical", see definition 7.1 in Stetter's book [72]. It involves 6 variables and has 6 quadratic equations. It has 6 isolated real solutions, each of multiplicity 4. Because (t, p) and $(-t, -p)$ represent the same line, every line is represented by two solutions. Using PHCpack, the real parts of two solutions at the end of continuation in a cluster are in Table I. The imaginary parts of these two solutions are in Table II. We can see that they are all close to zero and the radius of the imaginary parts are 10^{-7} . These two solutions are not very accurate because they should be the same.

In this thesis we show how to compute all multiple roots more accurately without multi-precision.

1.2 Newton's Method

Newton's method is a general procedure that can be used to solve polynomial systems, also called¹ the method of Gauss-Newton when $N > n$, see [64].

When the system has more equations than unknowns, the method becomes known as the Gauss-Newton method, using least squares to solve the overdetermined linear systems, see [64].

The famous Newton's iterator is

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [A(\mathbf{x}_n)]^{-1}F(\mathbf{x}_n) \quad (1.5)$$

¹In the light of the historical development outlined in [79], one should call Newton's method the Newton-Raphson-Simpson method.

TABLE I
TANGENT LINES TWO SOLUTIONS (REAL PART)

Solution 1	Solution 2	Difference
-7.07106803165780E-01	-7.07106794356709E-01	-8.80907E-09
-4.08248430737360E-01	-4.08248217029256E-01	-2.13708E-07
5.77350143082334E-01	5.77350304985648E-01	-1.61903E-07
-2.50000000000000E-01	-2.50000000000000E-01	0
4.33012701892221E-01	4.33012701892220E-01	9.99201E-16
9.56878363411174E-08	-6.07788020445124E-08	1.56467E-07

TABLE II
TANGENT LINES TWO SOLUTIONS (IMAGINARY PART)

Solution 1	Solution 2
3.77452918725401E-08	-1.29682370414209E-07
-1.83624917064964E-07	1.11010906008961E-07
-8.36140714113780E-08	-8.03312536501087E-08
-1.57896818458518E-16	-1.74789416181029E-16
-9.11600174682333E-17	-1.00914936462574E-16
1.54062878745083E-07	-1.39412292964849E-07

where $A(\mathbf{x}_n)$ is the Jacobian matrix of the polynomial system $F(\mathbf{x}_n)$. Newton's method is described in Figure 2. The Maple procedures for solving the linear system $Ax = b$ by singular value decomposition, one step Newton's method and Newton iteration are followed in Figure 3, Figure 4 and Figure 5.

For simple or regular solutions, Newton's method is quadratically convergent, see [19] or any other numerical analysis book.

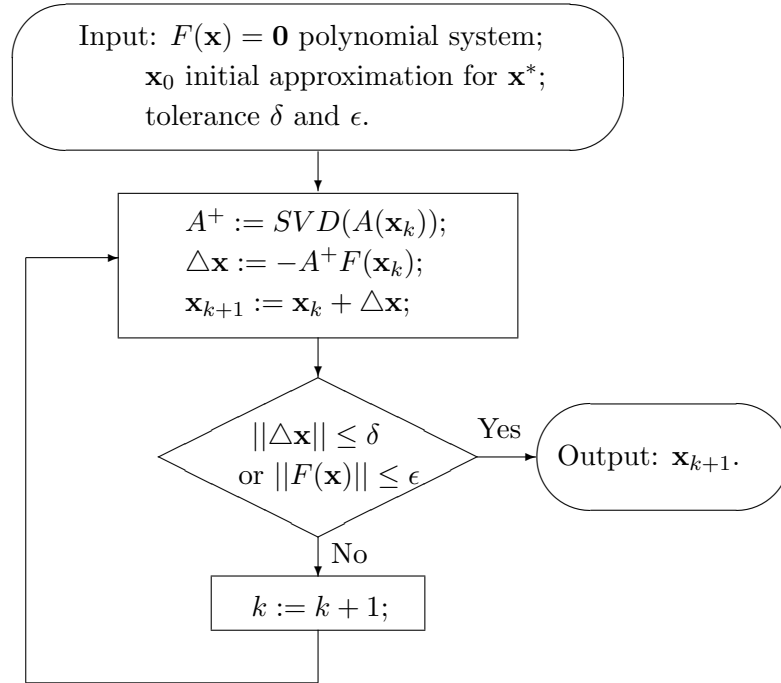


Figure 2. Flowchart for Newton's method.

Theorem 1.2.1 (quadratic convergence of Newton's method) *If the Jacobian matrix of F is regular at the root \mathbf{x}^* and the initial guess \mathbf{x}_0 is sufficiently close to \mathbf{x}^* , then the sequence of approximations \mathbf{x}_k produced by Newton's method converges quadratically to the root \mathbf{x}^* .*

With α -theory, we can certify what is an approximate root computing a radius for the region of convergence. This was developed by Shub and Smale, see [5], and was generalized to overdetermined systems in [11]. See [20; 21] for recent generalizations of this α -theory to multiple roots.

```

SVD:=A → LinearAlgebra[SingularValues](A,output=['U','S','Vt']):
SolvewithSVD:=proc(A::Matrix,b::Vector)
  description 'uses SVD to solve the system Ax=b':
  local U,S,Vt,UT,MS,IMS,IS,nr,nc,i,j,x:
  U,S,Vt:=SVD(A):
  UT:=Transpose(U);
  nr,nc:=Dimension(A);
  MS:=Matrix(nr);
  MS:=LinearAlgebra[DiagonalMatrix](S)[1..nr,1..nc];
  IMS:=MatrixInverse(MS);
  IS:=Matrix(nc,nr);
  IS[1..nc,1..nc]:=IMS[1..nc,1..nc];
  x:=Transpose(Vt)·(IS·(UT·b));
  return x;
end proc:

```

Figure 3. Maple procedure for solving the linear system $Ax = b$ by SVD

1.3 Homotopy Continuation Method

Homotopy continuation methods are globally convergent and exhaustive solvers for computing numerical approximations to all isolated solutions of polynomial systems, except perhaps at the very end of the paths if the system to be solved has singular solutions.

Definition 1.3.1 To solve a *target system* $F(\mathbf{x}) = \mathbf{0}$, we construct a *start system* $G(\mathbf{x}) = \mathbf{0}$ and define the *artificial-parameter homotopy*

$$H(\mathbf{x}, t) = \gamma(1 - t)G(\mathbf{x}) + tF(\mathbf{x}) = \mathbf{0}, \quad \gamma \in \mathbb{C}, \quad t \in [0, 1]. \quad (1.6)$$

```

Newton:=proc(sys, Vari, z0)
  local i,j, valuesys, temp, dx, y, jaco;
  valuesys:=Vector(nops(sys));
  jaco:=VectorCalculus[Jacobian](sys, Vari);
  for i from 1 to nops(sys) do
    valuesys[i]:=valueatx(sys[i], Vari, z0);
  od;
  temp:=valueofjacob(jaco, Vari, z0);
  dx:=SolvewithSVD(temp, valuesys);
  y:=Vector(nops(Vari));
  for i from 1 to nops(Vari) do
    y[i]:=z0[i]-dx[i];
  od;
  return y;
end proc:

```

Figure 4. Maple procedure for one step Newton method

```

NewIte:=proc(sys, Vari, GuessX, tol)
  local i,j,k, total, z0, NewV;
  z0:=GuessX;
  total:=sum('valueatx(sys[i], Vari, z0)
    *conjugate(valueatx(sys[i], Vari, z0))', 'i'=1..nops(sys));
  k:=0;
  while(sqrt(total) > tol) do
    z||(k+1):=Newton(sys, Vari, z||k);
    k:=k+1;
    NewV:=z||k;
    total:=sum('valueatx(sys[j], Vari, NewV)
      *conjugate(valueatx(sys[j], Vari, NewV))', 'j'=1..nops(sys));
    printf("Total Sum:=%e\n", Re(total));
  end do;
  printf("Total Iteration Steps:=%d", k);
  return z||k;
end proc:

```

Figure 5. Maple procedure for Newton iteration

The constant γ is a random constant number, it ensures no singularities occurs for $t \in [0, 1)$. The t is the *continuation parameter* and varies from 0 to 1, deforming the start system G into the target system F .

Consider the system (from [60])

$$F(x_1, x_2) = \begin{cases} x_1^2 + x_2 - 3 = 0; \\ x_1 + 0.125x_2^2 - 1.5 = 0. \end{cases} \quad (1.7)$$

The homotopy is

$$H(\mathbf{x}, t) = \gamma(1 - t) \begin{pmatrix} x_1^2 - 1 = 0 \\ x_2^2 - 1 = 0 \end{pmatrix} + t \begin{pmatrix} x_1^2 + x_2 - 3 = 0 \\ x_1 + 0.125x_2^2 - 1.5 = 0 \end{pmatrix} = \mathbf{0}. \quad (1.8)$$

For this system, there are two solutions, one is $(-3, -6)$, another one is $(1, 2)$ which is a triple root, so there are 3 paths converging to a cluster. For this triple root, we can not get quadratic convergence. The points in this cluster are input of our algorithm.

1.4 Related Work

Definition 1.4.1 \mathbf{x}^* is a *isolated solution* of $F(\mathbf{x}) = \mathbf{0}$ if it satisfies the following condition:

$$\text{for small enough } \epsilon > 0 : \{ \mathbf{y} \in \mathbb{C}^n : \|\mathbf{y} - \mathbf{x}^*\| < \epsilon \} \cap F^{-1}(\mathbf{0}) = \{\mathbf{x}^*\}. \quad (1.9)$$

We call \mathbf{x}^* a *singular solution* of $F(\mathbf{x}) = \mathbf{0}$ if and only if $\text{rank}(A(\mathbf{x}^*)) < n$, n is the number of variables. Let m be the *multiplicity* of the isolated solution \mathbf{x}^* of $F(\mathbf{x}) = \mathbf{0}$.

Newton’s method generates a sequence of approximations \mathbf{x}_k for \mathbf{x}^* . If \mathbf{x}^* is nonsingular, then the sequence converges quadratically (i.e.: $\|\mathbf{x}_k - \mathbf{x}_{k+1}\| = O(\|\mathbf{x}_{k-1} - \mathbf{x}_k\|^2)$) to \mathbf{x}^* , which justifies its widespread usage. But otherwise, if \mathbf{x}^* is singular, the convergence slows down and gets lost when $\mathbf{x}_k \approx \mathbf{x}^*$.

A straightforward approach is to use a working precision of $m \times D$ decimal places to achieve D correct decimal places in the final approximation, this is so-called “attainable accuracy” [80]. Even as multiprecision arithmetic is widely available and nowadays less expensive to use, this approach can only work if all coefficients in the system F have their first $m \times D$ decimal places correct. Our goal is to restore the quadratic convergence of a sequence converging to an isolated singular root without imposing extra requirements of precision on F . This means that we can compute isolated singularities with great accuracy in standard machine arithmetic, effectively *reconditioning* the problem.

Newton’s method for singular solutions has been extensively researched. The research up to the mid 1980’s is surveyed in [27]. We classify research on Newton’s method related to our work in two domains:

- 1. Detection and treatment of bifurcation points.** When following a solution path of a system defined by a parameter, the solution path may turn back or bifurcate for increasing values of the parameter. Techniques to detect and compute such bifurcation points are generally done via Lyapunov-Schmidt reduction. General references are [1], [14], [22] and [81]; see also [48; 49], [13], and [36]. One could interpret our method as a recursive application of the methods used to compute bifurcation points.

2. A deflation method for polynomial systems. A symbolic deflation method was presented in [63], and further developed in [60], [61], and [62]. We discovered this approach from the reference list of [40], which offers a symbolic deflation method whose complexity is quadratic in the multiplicity. As first announced in [78], we provide a numerically stable implementation of a modified symbolic deflation method.

A theoretical framework to study the complexity and numerical stability of Newton's method was developed by Shub and Smale, see [5], and was generalized to overdetermined systems in [11]. See [20; 21] for recent generalizations of this α -theory to multiple roots.

Singular solutions of polynomial systems are investigated in computational algebraic geometry, in particular we distinguish:

- 1. Standard bases in computer algebra.** SINGULAR [26] allows the computation of standard bases [25], implementing generalizations [24] of the algorithms in [55]. We use standard bases to show that the number of deflations to restore the quadratic convergence of Newton's method is bounded by the multiplicity.
- 2. Dual bases in numerical polynomial algebra.** The dual of an ideal, studied by Macaulay [51] with the goal of capturing multiplicity, is relevant from the point of numerical computations, see [72] and also [58]. Differential operators define Gröbner duality [56], providing suitable representations for multiple roots [53].

1.5 Contributions of this Thesis

The contribution of this thesis is twofold. First – as announced in [78] – we provide a numerically stable implementation of a modified symbolic deflation method. Second, using standard bases [24], we show that the number of deflations needed to restore the quadratic convergence of Newton’s method is bounded by the multiplicity. In the next chapter we describe our method in detail, followed by an introduction to standard bases and our proof in the second section. In the Chapter three, we describe a way to implement our symbolic-numeric method efficiently. Higher order deflation for polynomial systems is introduced in Chapter 4 as well as the multiplicity structure of isolated singularities. Numerical results are described in last chapter.

CHAPTER 2

NEWTON'S METHOD WITH DEFLATION FOR ISOLATED SINGULARITIES OF POLYNOMIAL SYSTEMS

In this chapter, we present a modification of Newton's method to restore quadratic convergence for isolated singular solutions of polynomial systems. Our method is symbolic-numeric: we produce a new polynomial system which has the original multiple solution as a regular root. In section 2, using standard bases, a tool for the symbolic computation of multiplicities, we show that the number of deflation stages is bounded by the multiplicity of the isolated root. This chapter corresponds to the paper [42] published in Theoretical Computer Science.

2.1 A Modified Deflation Method

A singular root \mathbf{x}^* of a square (i.e., $N = n$) system $F(\mathbf{x}) = \mathbf{0}$ with Jacobian matrix $A(\mathbf{x})$ satisfies

$$\begin{cases} F(\mathbf{x}) = \mathbf{0}; \\ \det(A(\mathbf{x})) = 0. \end{cases} \quad (2.1)$$

The augmented system (Equation 2.1) forms the basic idea for deflation. If \mathbf{x}^* is isolated and $\text{corank}(A(\mathbf{x}^*)) = 1$, then \mathbf{x}^* as root of (Equation 2.1) has a lower multiplicity.

We find deflation used repeatedly first in [63], and later modified in [60] and applied in [61; 62].

In theory, $\det(A(\mathbf{x})) = 0$ (or maximal minors) could be used to form new equations. But this is neither good symbolically because the determinant is usually of high degree and leads

to expression swell, nor numerically, as evaluating polynomials of high degree is numerically unstable.

Instead of using the determinant, on a system F of N equations in n variables, we proceed along the following three steps to form new equations:

1. Let $r = \text{rank}(A(\mathbf{x}_0), \epsilon)$ for $\mathbf{x}_0 \approx \mathbf{x}^*$ and tolerance ϵ , $0 < \epsilon \ll 1$. For numerical stability, we compute the rank via SVD of the matrix $A = A(\mathbf{x}_0)$. The numerical rank r equals the number of singular values larger than the tolerance ϵ .
2. Let $\mathbf{h} \in \mathbb{C}^{r+1}$ be a random vector. For numerical stability, we generate random numbers on the complex unit circle. We use \mathbf{h} in a scaling equation to obtain a unique vector in the kernel of the Jacobian matrix.
3. Let $B \in \mathbb{C}^{n \times (r+1)}$ be a random matrix, also with numbers on the complex unit circle. Using B , we form $C(\mathbf{x}) = A(\mathbf{x})B$. Notice that $C = [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{r+1}]$ is an $N \times (r+1)$ matrix with polynomial entries.

With probability one (exceptional pairs of vectors \mathbf{h} and matrices B belong to a proper algebraic subset of $\mathbb{C}^{r+1} \times \mathbb{C}^{n \times (r+1)}$) we have

$$\begin{aligned} \text{rank}(A(\mathbf{x}^*)) = r &\Leftrightarrow \text{corank}(C(\mathbf{x}^*)) = 1 \\ \Leftrightarrow \text{there is a unique } \boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{r+1} \end{pmatrix} : G(\mathbf{x}^*, \boldsymbol{\lambda}) = \begin{pmatrix} \sum_{i=1}^{r+1} \lambda_i \mathbf{c}_i(\mathbf{x}^*) \\ \sum_{i=1}^{r+1} h_i \lambda_i - 1 \end{pmatrix} = \mathbf{0}. \end{aligned} \quad (2.2)$$

The random \mathbf{h} and B guarantee the existence and uniqueness of the solution $\boldsymbol{\lambda}$ to $G(\mathbf{x}, \boldsymbol{\lambda})$ when $\mathbf{x} = \mathbf{x}^*$. Note that instead of multiplying the Jacobian matrix of $F(\mathbf{x}) = \mathbf{0}$ by B , we could use B for a generic coordinate change $\mathbf{x} = B\mathbf{y}$, which would after application of the chain rule on $F(B\mathbf{y})$ be equivalent to the formation of $C = A(\mathbf{x})B$.

In one deflation step, we add the equations of $G(\mathbf{x}, \boldsymbol{\lambda})$ instead of $\det(A(\mathbf{x})) = 0$ to the system $F(\mathbf{x}) = \mathbf{0}$, adding $r + 1$ extra variables $\lambda_1, \lambda_2, \dots, \lambda_{r+1}$. The flowchart for our modified deflation algorithm is displayed in Figure 6. The Maple procedure for adding equations is followed in Figure 7. See Section 2.3 for implementation discussion. For now we work with a basic Maple implementations, in Chapter 3 we give more efficient alternatives.

2.2 A Bound on the Number of Deflations

The termination of our algorithm in Figure 6 depends on the following theorem.

Theorem 2.2.1 *The number of deflations needed to restore the quadratic convergence of Newton's method converging to an isolated solution is strictly less than the multiplicity of the isolated solution.*

The answer to the question “How many deflations?” can be understood by looking at a standard basis for the ideal generated by the given polynomials in the system. We use standard bases to prove the termination of our algorithm, as explained in the next two subsections.

A duality analysis of our method was presented in [7]. Just like the analysis in [7] gives a better understanding on the number of needed deflations (establishing “depth” as a tighter bound), the shape of the standard basis (visualized by its staircase) leads to a more accurate bound.

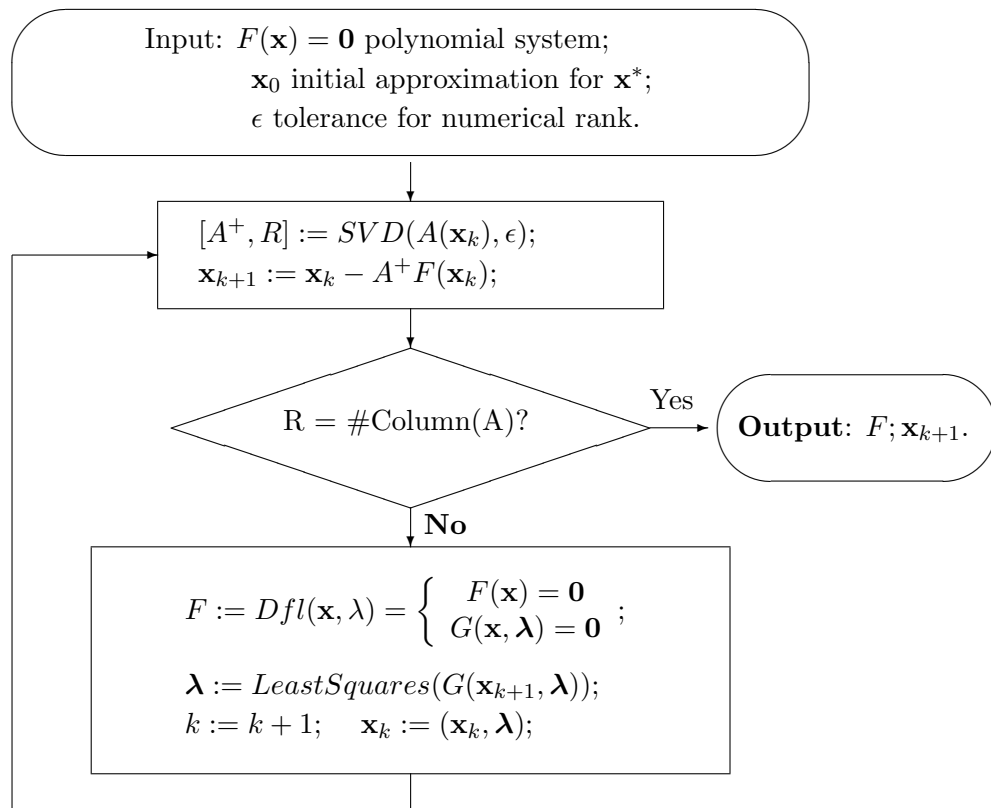


Figure 6. Flowchart for a modified deflation method.

```

Addsys:=proc(sys,Vari,RandomH,RandomB,R)
  local jacob,i,f,AB,MV,nsys,k;
  jacob:=VectorCalculus[Jacobian](sys,Vari);
  f:=(i)→ λ|i:
  AB:=jacob·RandomB;
  MV:=AB·Vector(R+1,f);
  nsys:=[seq(MV[i],i=1..nops(sys)),sum('λ|k'*RandomH[k], 'k'=1..(R+1))-1];
  return nsys;
end proc:
  
```

Figure 7. Maple procedure for adding equations

2.2.1 Standard Bases for Local Orderings

Let $R = k[x_1, \dots, x_n]$ be the ring of polynomials in n variables with coefficients in the field k .

We use the following multi-degree notation: $\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, where $\alpha = (\alpha_1, \dots, \alpha_n)$ is a vector of nonnegative integers.

A multiplicative ordering \leq on the monoid $\{\mathbf{x}^\alpha \mid \alpha \in \mathbb{Z}_{\geq 0}^n\}$ is a *local ordering* if $\mathbf{x}^\alpha < 1$ for all $\alpha \neq (0, 0, \dots, 0)$.

To any *weight vector* $\omega \in \mathbb{Z}_{< 0}^n$ we may associate the ordering \leq_ω by setting

$$\mathbf{x}^\alpha \leq_\omega \mathbf{x}^\beta \Leftrightarrow \langle \alpha, \omega \rangle \leq \langle \beta, \omega \rangle, \quad (2.3)$$

where $\langle \cdot, \cdot \rangle$ is the usual inner product. Note that it is possible to have $\mathbf{x}^\alpha =_\omega \mathbf{x}^\beta$ (unless there are no integer vectors orthogonal to ω). In this case, the order is refined by, say, a lexicographic order.

In presence of a monomial ordering \leq_ω , a polynomial

$$f(\mathbf{x}) = \sum_{\alpha \in \mathbb{Z}_{\geq 0}^n} c_\alpha \mathbf{x}^\alpha \in R, \text{ where } \text{supp}(f) = \{ \alpha \mid c_\alpha \neq 0 \} \text{ is finite,} \quad (2.4)$$

has the following attributes associated with it:

$$\mathbf{le}(f) = \text{the leading exponent} = \max_{\leq \omega} \text{supp}(f)$$

$$\mathbf{lm}(f) = \text{the leading monomial} = \mathbf{x}^{\mathbf{le}(f)}$$

$$\mathbf{lc}(f) = \text{the leading coefficient} = c_{\mathbf{le}(f)}$$

$$\mathbf{lt}(f) = \text{the leading term} = \mathbf{lc}(f)\mathbf{lm}(f)$$

Let $\mathcal{I} \subset R$ be an ideal. We call a set of polynomials $S \subset \mathcal{I}$ a *standard basis* of \mathcal{I} if for any $f \in \mathcal{I}$ there is $g \in S$ such that $\mathbf{lm}(g) | \mathbf{lm}(f)$. Alternatively, S is a standard basis iff the *initial ideal* $\mathbf{in}(\mathcal{I}) = \langle \{\mathbf{lm}(f) | f \in \mathcal{I}\} \rangle$ is generated by the leading monomials $\mathbf{lm}(S) = \{\mathbf{lm}(g) | g \in S\}$.

The monomials that do not belong to the initial ideal $\mathbf{in}(\mathcal{I})$ are called *standard monomials*. The minimal generators of $\mathbf{in}(\mathcal{I})$ shall be called the *corners* of the staircase. The corners of the form x_i^a for some i and a are called the *endpoints* of the staircase.

A standard basis S is *reduced* if the leading monomials of its elements form a minimal generating set for the initial ideal $\mathbf{in}(\mathcal{I})$ and the tail $g - \mathbf{lt}(g)$ contains only standard monomials.

Graphically, any monomial ideal can be represented by a staircase in the nonnegative integer lattice $\mathbb{Z}_{\geq 0}^n$. For example, let \mathcal{I} be the ideal of $R = k[x_1, x_2]$ generated by

$$\begin{aligned} f_1 &= x_1^3 + x_1x_2^2; \\ f_2 &= x_1x_2^2 + x_2^3; \\ f_3 &= x_1^2x_2 + x_1x_2^2. \end{aligned} \tag{2.5}$$

The initial ideal depends on the ordering chosen: the staircase at the left in Figure 8 represents $\mathbf{in}_\omega(\mathcal{I})$, where $\omega = (-1, -2)$. The staircase at the right in Figure 8 represents $\mathbf{in}_\omega(\mathcal{I})$, where $\omega = (-2, -1)$.

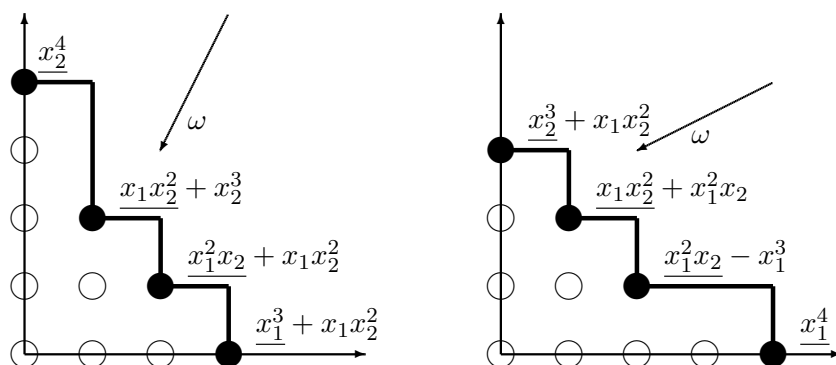


Figure 8. Two different staircases of the standard basis of \mathcal{I} with respect to different local orderings $\leq_{(-1,-2)}$ (at the left) and $\leq_{(-2,-1)}$ (at the right). Monomials generating $\mathbf{in}(\mathcal{I})$ are represented by black disks, while the standard monomials are shown as empty circles.

Observe in Figure 8 that the number of standard monomials is the same for both orderings. This is so for any local ordering, as the standard monomials form a basis of the k -linear space $R_{\langle x_1, \dots, x_n \rangle} / R_{\langle x_1, \dots, x_n \rangle} \mathcal{I}$, where $R_{\langle x_1, \dots, x_n \rangle}$ is the localization of the polynomial ring R at the origin and $R_{\langle x_1, \dots, x_n \rangle} \mathcal{I}$ is the extension of the ideal \mathcal{I} in this localized ring (see [6] for details). This linear space is of finite dimension if and only if the origin is an isolated solution; its dimension,

which is the multiplicity of the origin, then equals the number of the standard monomials for any local ordering.

Thanks to Mora [55], there is an algorithm for computing standard bases, generalized by Greuel and Pfister [24], and implemented in the computer algebra system Singular [25]. We will not use standard bases explicitly – except for theoretical purposes – but note an analytic interpretation of the local ordering \leq_w : as we approach the origin along a smooth curve

$$c : \mathbb{C} \rightarrow \mathbb{C}^n \quad \text{such that} \quad c(t) = \begin{cases} b_1 t^{-\omega_1} (1 + O(t)) \\ \vdots \\ b_n t^{-\omega_n} (1 + O(t)) \end{cases} \quad (2.6)$$

with $\omega \in \mathbb{Z}_{<0}^n$ and $(b_1, \dots, b_n) \in \mathbb{C}^n \setminus \{0\}$, for every $f \in \mathcal{I}$ the leading term $\mathbf{lt}_w(f)$ becomes dominant, i.e.:

$$f(c(t)) = \mathbf{lt}_w(f)(c(t)) + O(t^{-\langle \omega, \mathbf{le}(f) \rangle}). \quad (2.7)$$

2.2.2 Understanding the Deflation Method

First of all, let us formulate the goal of what we would call the *symbolic deflation* process: Given a system of polynomial equations $f_i = 0$, $i = 1, 2, \dots, N$, with the point $\mathbf{x}^* \in \mathbb{C}^n$ as an isolated solution of multiplicity $m > 1$, find a system $g_i = 0$, $i = 1, 2, \dots, N'$, such that \mathbf{x}^* is still an isolated solution of multiplicity less than m .

The best deflation one can devise is the one that corresponds to the maximal ideal annihilating x_i^* , i.e.: $g_i = x_i - x_i^*$, $i = 1, 2, \dots, n$. However, from a practical angle of numerical methods

what we actually need is an algorithm that would relate the deflated system to the original one in a numerically stable way and taking into account the fact that the isolated solution \mathbf{x}^* may be known only approximately.

2.2.2.1 A Symbolic Deflation Method

Here we assume that everything is exact and, therefore, without loss of generality we may assume that the isolated solution \mathbf{x}^* is the origin.

Consider the ideal \mathcal{I} generated by the polynomials f_i of the original system. We call an ideal \mathcal{I}' a *deflation* of \mathcal{I} if $\mathcal{I}' \supset \mathcal{I}$, $\mathcal{I}' \neq R$, and the multiplicity of the origin for \mathcal{I}' is lower than that for the original ideal \mathcal{I} .

If the multiplicity $m > 1$, it means that the initial ideal $\mathbf{in}(\mathcal{I})$ does not contain x_i for some i .

Proposition 2.2.2 *Suppose $m > 1$ and let g be an element of a reduced standard basis of \mathcal{I} with respect to a local monomial ordering \leq , such that $\mathbf{lm}(g) = x_i^d$, for some $i \in \{1, \dots, n\}$ and $d > 1$. Then the ideal $\mathcal{I}' = \mathcal{I} + \langle \partial g / \partial x_i \rangle$ is a deflation of \mathcal{I} .*

Proof. The derivative $\partial g / \partial x_i$ cannot contain monomials $> x_i^{d-1}$. Therefore, \mathcal{I}' contains \mathcal{I} properly, since $\mathbf{lm}(\partial g / \partial x_i) = x_i^{d-1}$ is a standard monomial for \mathcal{I} . The appended generator $\partial g / \partial x_i$ still vanishes at the origin, hence, $\mathcal{I}' \neq R$. \square

Mora's tangent cone algorithm [55] for computing standard bases is expensive symbolically and, more importantly, unstable numerically. Can we find x_i and g in the proposition in a less straightforward way? The next lemma gives a positive answer.

A linear coordinate change $T : \mathbb{C}^n \rightarrow \mathbb{C}^n$ induces an automorphism of the polynomial ring $R = \mathbb{C}[x_1, \dots, x_n]$, which we call T as well: $T(f)(x) = f(T(x))$. The ideal $T(\mathcal{I}) = \{T(f) \mid f \in \mathcal{I}\} = \langle T(f_1), \dots, T(f_N) \rangle$ represents the system after the change of coordinates.

Let $A(\mathbf{x})$ be the Jacobian matrix of the system $F(\mathbf{x}) = 0$, i.e.: an N -by- n matrix with polynomial entries $A_{ij}(\mathbf{x}) = \partial f_i / \partial x_j$. The origin is singular if and only if $c = \text{corank}(A(\mathbf{0})) > 0$. Since the Jacobian matrix is rank-deficient, the kernel of $A(\mathbf{0})$ is nonzero.

Lemma 2.2.3 *Take a nonzero vector $\boldsymbol{\lambda} \in \ker A(\mathbf{0}) \subset \mathbb{C}^n$ and let $T : \mathbb{C}^n \rightarrow \mathbb{C}^n$ be a linear coordinate transformation such that:*

$$T_i(\mathbf{x}) = \lambda_i x_1 + \sum_{j=2}^n \mu_{ij} x_j, \quad \text{for } i = 1, 2, \dots, n, \quad (2.8)$$

where $[\boldsymbol{\lambda}, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_n]$ is a nonsingular matrix. Then $\partial_1(T(\mathcal{I})) = \left\{ \frac{\partial}{\partial x_1} f \mid f \in T(\mathcal{I}) \right\}$ is a deflation of $T(\mathcal{I})$.

Proof. For all $i = 1, 2, \dots, N$,

$$\frac{\partial}{\partial x_1} (f_i(T(\mathbf{x}))) = \sum_{j=1}^n \frac{\partial f_i}{\partial x_j} (T(\mathbf{x})) \cdot \frac{\partial T_j}{\partial x_1}(\mathbf{x}) = \sum_{j=1}^n \left(\frac{\partial f_i}{\partial x_j} (T(\mathbf{x})) \right) \lambda_j. \quad (2.9)$$

The last expression is equal to 0 when $\mathbf{x} = \mathbf{0}$, since $\boldsymbol{\lambda} \in \ker A(\mathbf{0})$.

Take any $g = b_1T(f_1) + \cdots + b_NT(f_N) \in T(\mathcal{I})$, where $b_i \in R$ for all i . Then

$$\begin{aligned} \frac{\partial g}{\partial x_1} &= b_1 \frac{\partial(T(f_1))}{\partial x_1} + \cdots + b_N \frac{\partial(T(f_N))}{\partial x_1} \\ &+ T(f_1) \frac{\partial b_1}{\partial x_1} + \cdots + T(f_N) \frac{\partial b_N}{\partial x_1}. \end{aligned}$$

In view of (Equation 2.9), the last expression evaluates to 0 at $\mathbf{x} = \mathbf{0}$. Therefore, $\partial_1(T(\mathcal{I}))$ is a proper ideal annihilating the origin. On the other hand, there is an element g of a reduced standard basis of $T(\mathcal{I})$ with respect to a local ordering such that $\mathbf{in}(g) = x_1^d$ with $d > 1$. According to the Proposition 2.2.2 the ideal $\mathcal{I}' = T(\mathcal{I}) + \langle \partial g / \partial x_1 \rangle$ is a deflation of $T(\mathcal{I})$. So is $\partial_1(T(\mathcal{I}))$, for it contains \mathcal{I}' . \square

Lemma 2.2.3 leads to Algorithm 1.

Algorithm 1 $G = \text{Symbolic_Deflation}(F)$

Require: F , a finite set of polynomials in R , such that the ideal $\langle F \rangle$ has multiplicity $m > 1$ at the origin.

Ensure: G , a finite set of polynomials in R , such that the ideal $\langle G \rangle + \langle F \rangle$ is a deflation of $\langle F \rangle$.

Compute the Jacobian A of F at the origin;

Pick a nonzero vector $\boldsymbol{\lambda} \in \ker A(\mathbf{0})$;

$$G := \left\{ \sum_{i=1}^n \lambda_i \frac{\partial f}{\partial x_i} \mid f \in F \right\}.$$

2.2.2.2 A Numeric Deflation Method

Algorithm 2 $G = \text{NumericDeflation}(F, \mathbf{x}_0)$

Require: $F = \{f_1, \dots, f_N\}$, a finite set of polynomials in R , such that the ideal $\langle F \rangle$ has multiplicity $m > 1$ at the point $\mathbf{x}^* \approx \mathbf{x}_0$.

Ensure: G , a finite set of polynomials in $R' = R[\lambda_1, \dots, \lambda_{r+1}]$, where $r = \text{rank } A(\mathbf{x}_0)$, such that

- the ideal $\langle G \rangle \subset R'$ has an isolated solution at the point $P = (\mathbf{x}^*, \boldsymbol{\lambda}^*) \in \mathbb{C}^{n, r+1}$;
 - the vector $\boldsymbol{\lambda}^*$ is determined uniquely;
 - the multiplicity of P is less than m .
-

Compute the Jacobian matrix $A(\mathbf{x})$ of F ;

$r := \text{rank } A(\mathbf{x}_0)$; (*numerical rank at the approximate solution \mathbf{x}_0*)

(R1) Generate a random matrix $B \in \mathbb{C}^{n \times (r+1)}$;

$C(\mathbf{x}) := A(\mathbf{x})B$; ($N \times (r+1)$ matrix with polynomial entries)

Let $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_{r+1})^T$ be a vector of indeterminates;

Consider N new polynomials $g_i(\mathbf{x}, \boldsymbol{\lambda}) = (C(\mathbf{x})\boldsymbol{\lambda})_i \in R'$;

(R2) $h(\boldsymbol{\lambda}) := h_1\lambda_1 + \dots + h_{r+1}\lambda_{r+1} - 1$, where the h_i are random numbers in \mathbb{C} ;

$G := F \cup \{g_1, \dots, g_N\} \cup \{h\}$.

Our method formalized in Algorithm 2 is a numerical version of Algorithm 1. In this section we explain the transition from the symbolic to the numeric deflation method.

Consider a point $P = (\mathbf{x}, \boldsymbol{\lambda}) \in \mathbb{C}^{n, r+1}$ and let $\mathbf{x} = \mathbf{x}_0$. When this specialization is performed, the values for $\boldsymbol{\lambda}$ are determined by the following system of $N + 1$ linear equations:

$$\begin{cases} A(\mathbf{x}_0)\boldsymbol{\lambda} = 0 \\ \langle \mathbf{h}, \boldsymbol{\lambda} \rangle = 1 \end{cases} \quad (2.10)$$

where $\mathbf{h} = (h_1, \dots, h_{r+1})$ is a vector of random complex numbers.

Observe how C is created: the randomization step **R1** insures that the $r + 1$ columns of $C(\mathbf{x}^*)$ are random combinations of the columns of $A(\mathbf{x}^*)$ and, therefore, $\text{corank } C(\mathbf{x}^*) = 1$ with

probability one. Then $\boldsymbol{\lambda}$ is bound to live in the one-dimensional $\ker C(\mathbf{x}^*)$. The randomization step **R2** makes sure one nonzero vector is picked out from the kernel. This proves the uniqueness of $\boldsymbol{\lambda}^*$.

Assume that the original system is of corank 1 (we can always replace $F(\mathbf{x})$ with $F(\mathbf{x})B$, where B is as above). The correctness of the numeric deflation process relies on the following Proposition.

Proposition 2.2.4 *Let $\mathbf{x}^* \in \mathbb{C}^n$ be an isolated solution of $F(\mathbf{x}) = \mathbf{0}$ (in $\mathbb{C}[\mathbf{x}]$) and $\text{corank}(A(\mathbf{x}^*)) = 1$.*

Consider the augmented system

$$G(\mathbf{x}, \boldsymbol{\lambda}) = (f_1, \dots, f_N, g_1, \dots, g_N, h)(\mathbf{x}, \boldsymbol{\lambda}) = 0,$$

with the new $N + 1$ equations

$$g_i(\mathbf{x}, \boldsymbol{\lambda}) = \boldsymbol{\lambda} \cdot \nabla f_i(\mathbf{x}) = \sum_{j=1}^n \lambda_j \frac{\partial f_i(\mathbf{x})}{\partial x_j}, \quad (i = 1, \dots, N) \quad (2.11)$$

$$h(\boldsymbol{\lambda}) = \mathbf{h} \cdot \boldsymbol{\lambda} - 1 = \sum_{j=1}^n h_j \lambda_j - 1, \quad (2.12)$$

For a generic choice of coefficients $\mathbf{h} = (h_1, \dots, h_{r+1})$, there exists a unique $\boldsymbol{\lambda}^ \in \mathbb{C}^n$ such that system $G(\mathbf{x}, \boldsymbol{\lambda})$ of equations in $\mathbb{C}[\mathbf{x}, \boldsymbol{\lambda}]$ has an isolated solution at $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$.*

Moreover, the multiplicity of $(\mathbf{x}^, \boldsymbol{\lambda}^*)$ in $G(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$ is lower than that of \mathbf{x}^* in $F(\mathbf{x}) = \mathbf{0}$.*

Proof. Consider $g_1 = \cdots = g_N = h = 0$ as a system of equations in the local ring $R_* = \mathbb{C}[\mathbf{x}, \boldsymbol{\lambda}]_{(\mathbf{x}^*, \boldsymbol{\lambda}^*)}$ that is linear in $\boldsymbol{\lambda}$.

The specialization of this system at $\mathbf{x} = \mathbf{x}^*$ makes it a linear system (with constant coefficients) of full rank with the unique solution: $\boldsymbol{\lambda}^*$. Therefore, using row operations in the ring R_* it is possible to reduce the system to the system of the form

$$\left\{ \begin{array}{l} \lambda_1 = a_1(\mathbf{x}), \\ \vdots \\ \lambda_n = a_n(\mathbf{x}), \end{array} \right. \quad (2.13)$$

where $a_i(\mathbf{x})$ are rational expressions. Note that $a_i(\mathbf{x}^*) = \lambda_i^*$.

Now we conclude that considering multiplicity of the augmented system $G(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$ with indeterminate $\boldsymbol{\lambda}$ in the ring R^* is equivalent to looking at the system $G(\mathbf{x}) = \mathbf{0}$ with fixed $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$ in the local ring $\mathbb{C}[\mathbf{x}]_{\mathbf{x}^*}$.

Assuming \mathbf{x}^* is the origin, the multiplicity drops by Lemma 2.2.3. □

Proof of Theorem 2.2.1. To show that at most $m - 1$ deflation steps are needed to restore the quadratic convergence of an isolated root \mathbf{x}^* of multiplicity m , it suffices to show that after one deflation step, the augmented system has the same root with its multiplicity decreased by at least one. This statement is shown by Proposition 2.2.4 in case the Jacobian matrix $A(\mathbf{x}^*)$ has corank one. To reduce to the general case, our deflation algorithm replaces $A(\mathbf{x})$ by $A(\mathbf{x})B$ using a random matrix B of $r + 1$ columns, where $r = \text{rank}(A(\mathbf{x}^*))$. □

Remark 2.2.5 *By bounding the number of deflation steps with multiplicity, Theorem 2.2.1 guarantees the termination of the deflation procedure. However, a stronger bound can be given: one may show that the number of deflations is equal to or less than the maximal total degree of monomials under the staircase for any local monomial ordering.*

This statement can be rephrased in the language of dual bases of differential functionals; in this form it has been proved in [7]. The discussion of the correspondence between the two methods is provided in Chapter 4.

In practice, finer information – staircases, dual bases, multiplicity structure – is not available at the time when the deflation is applied. The algorithm of [7] for computing the multiplicity structure, in fact, depends on the precision of the solution approximation, hence, on deflation.

2.3 A Symbolic-Numeric Implementation

The method was tested and developed in Maple 9. Since release 2.3 of PHCpack [76], the deflation algorithm is part of the validation (`phc -v`) module. In this section we briefly address symbolic-numeric issues.

2.3.1 Special Case Implementations and Extensions

The case where the corank of the Jacobian matrix equals one occurs frequently and can be treated more efficiently than the general case. In [7] a modification of the deflation algorithm was proposed for the important case of corank one. In the corank one case, there is no need to multiply the Jacobian matrix by a random matrix. Moreover, as pointed out in [7], subsequent deflations should only concern the original set of equations. Deflating a system of N equations in n unknowns m times then leads to a system of mM equations in mn unknowns as shown

in [7]. Another extension, exemplified in [7], concerns the application of our deflation method to analytic systems.

2.3.2 A Posteriori Validation of the Numerical Rank

Our implementation defers the problem of the computation of the numerical rank to the established SVD and the recent techniques presented in [17] and [50]. Alternative to tolerance ϵ , $\max \sigma_{i+1}/\sigma_i, \sigma_1 > \sigma_2 > \dots > \sigma_n$ determines rank where the largest gap in σ occurs. The critical decision to deflate or not depends on the correct determination of the numerical rank of the Jacobian matrix. If we compute the rank correctly, then after the deflation – with an accurate root – we obtain an a posteriori validation of all decisions made to determine the numerical rank. If we deflate with an incorrect rank, then there are two cases: either our numerical rank is too low or too high. The first case of numerical rank too low will be detected early as the algorithm will then produce too many additional constraints. In that case, the calculation in Figure 6 of the initial values $\hat{\lambda}$ for the multipliers via the least squares problem is already likely to fail and an interactive application of the method may backtrack. In the second case, when the numerical rank is too high, one may not deflate at all in case of full rank and continue applying Newton’s method until one is close enough for the threshold to be crossed. A deflation with corank too small may be remedied by an extra deflation step, although we have no practical experience with this case.

2.3.3 Avoiding the Expression Swell

Although the symbolic implementation performs well for a couple of deflation steps, the multiplication of polynomial matrices by a random matrices leads to expression swell, amplified by the doubling of the size of the systems in each deflation stage.

Inspired by automatic differentiation [28; 65] and [37], we found that we should first evaluate the Jacobian matrices before multiplication with the random matrices. Furthermore, observing that the multiplier variables occur linearly in the block structure of the Jacobian matrices of the deflated systems, we presented in Chapter 3 a directed acyclic graph to evaluate the Jacobian matrices of the deflated systems efficiently. We refer to [43] or Chapter 3 for a detailed description of the efficient evaluation of the Jacobian matrices.

Another significant reduction of the expression swell lies in treating the corank one case separately, an issue we address in Section 2.3.1 above.

CHAPTER 3

EVALUATION OF JACOBIAN MATRICES FOR NEWTON'S METHOD WITH DEFLATION

In the previous chapter, Newton's method with deflation was introduced. While the performance of the method is promising on selected applications (such as the fourfold isolated roots of the cyclic 9-roots problem, see [42]), the method suffers from expression swell after a couple of deflations. In this chapter, we describe a way to “unfold” the extra multiplier variables, exploiting the special structure of the matrices which arise in the deflation process. In the unfolding process, we naturally arrive at trees (or more precisely, directed acyclic graphs), also employed in [36; 37].

Our Jacobian matrices have a particular sparse block structure which should be also be exploited when computing the numerical rank and solving the linear system. For this, we recommend the recent rank-revealing algorithms in [50].

This chapter corresponds to the paper [43] published in Symbolic-Numeric Computation.

3.1 Problem Statement

In this section we fix the notation. Consider $f(\mathbf{x}) = \mathbf{0}$, a system of N polynomial equations in n unknowns, with isolated multiple root \mathbf{x}^* . As we restrict ourselves to isolated roots, we

have $N \geq n$. At $\mathbf{x} = \mathbf{x}^*$, the Jacobian matrix $A(\mathbf{x})$ of f has rank $R < n$. At stage k in the deflation, we have the system

$$f_k(\mathbf{x}, \boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_{k-1}, \boldsymbol{\lambda}_k) = \begin{cases} f_{k-1}(\mathbf{x}, \boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_{k-1}) & = 0; \\ A_{k-1}(\mathbf{x}, \boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_{k-1})B_k\boldsymbol{\lambda}_k & = 0; \\ \mathbf{h}_k\boldsymbol{\lambda}_k & = 1. \end{cases} \quad (3.1)$$

with $f_0 = f$, $A_0 = A$, and where $\boldsymbol{\lambda}_k$ is a vector of $R_k + 1$ multiplier variables, $R_k = \text{rank}(A_{k-1}(\mathbf{x}^*))$, scaled using a random vector $\mathbf{h}_k \in \mathbb{C}^{R_k+1}$. The matrix B_k is a random matrix with as many rows as the number of variables in the system f_{k-1} and with $R_k + 1$ columns. The Jacobian matrix of f_k is A_k . We have the following relations

$$\#\text{rows in } A_k : N_k = 2N_{k-1} + 1, \quad N_0 = N, \quad (3.2)$$

$$\#\text{columns in } A_k : n_k = n_{k-1} + R_k + 1, \quad n_0 = n. \quad (3.3)$$

The second line in (Equation 3.1) requires the multiplication of N_{k-1} -by- n_{k-1} polynomial matrix A_{k-1} with the random n_{k-1} -by- $R_k + 1$ matrix B_k , with the vector of $R_k + 1$ multiplier variables $\boldsymbol{\lambda}_k$.

If the evaluation of $A_{k-1}B_k\boldsymbol{\lambda}_k$ is done symbolically, i.e., if we first compute the polynomial matrix $A_{k-1}(\mathbf{x}, \boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_{k-1})B_k$ before we give values to $(\mathbf{x}, \boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_{k-1})$, the expression swell will cause the evaluation to be very expensive. In this chapter we describe how to first evaluate the Jacobian matrices before the matrix multiplications are done. As this technical description

forms a blueprint for an efficient implementation, it also sheds light on the complexity of the deflation.

3.2 Unwinding the Multipliers

We observe in (Equation 3.1) that the multiplier variables in $\lambda_1, \lambda_2, \dots, \lambda_k$ all occur linearly. The Jacobian matrix of f_k in (Equation 3.1) has a nice block structure which already separates the linearly occurring λ_k from the other variables:

$$A_k(\mathbf{x}, \lambda_1, \dots, \lambda_{k-1}, \lambda_k) = \begin{bmatrix} A_{k-1} & \mathbf{0} \\ \left[\frac{\partial A_{k-1}}{\partial \mathbf{x}} \frac{\partial A_{k-1}}{\partial \lambda_1} \dots \frac{\partial A_{k-1}}{\partial \lambda_{k-1}} \right] B_k \lambda_k & A_{k-1} B_k \\ \mathbf{0} & \mathbf{h}_k \end{bmatrix}, \quad (3.4)$$

where the partial derivatives of an N -by- n matrix A with respect to a vector of variables $\mathbf{x} = (x_1, x_2, \dots, x_n)$ give rise to a vector of matrices:

$$\frac{\partial A}{\partial \mathbf{x}} = \left[\frac{\partial A}{\partial x_1} \quad \frac{\partial A}{\partial x_2} \quad \dots \quad \frac{\partial A}{\partial x_n} \right], \quad \frac{\partial A}{\partial x_k} = \left[\frac{\partial a_{ij}}{\partial x_k} \right] \text{ for } A = [a_{ij}], \quad \begin{matrix} i=1,2,\dots,N, \\ j=1,2,\dots,n, \\ k=1,2,\dots,n. \end{matrix} \quad (3.5)$$

The definition of $\frac{\partial A}{\partial \mathbf{x}}$ naturally satisfies the relation $\frac{\partial}{\partial \mathbf{x}}(A\mathbf{x}) = A$.

Notice that in (Equation 3.4) the evaluation of

$$\left[\frac{\partial A_{k-1}}{\partial \mathbf{x}} \quad \frac{\partial A_{k-1}}{\partial \lambda_1} \quad \dots \quad \frac{\partial A_{k-1}}{\partial \lambda_{k-1}} \right] B_k \lambda_k \quad (3.6)$$

yields the matrix

$$\left[\frac{\partial A_{k-1}}{\partial \mathbf{x}} B_k \boldsymbol{\lambda}_k \quad \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_1} B_k \boldsymbol{\lambda}_k \quad \cdots \quad \frac{\partial A_{k-1}}{\partial \boldsymbol{\lambda}_{k-1}} B_k \boldsymbol{\lambda}_k \right] \quad (3.7)$$

which has the same number of columns as A_{k-1} .

As we started this section observing the separation of $\boldsymbol{\lambda}_k$ in the block structure of A_k , we can carry this further through to all multiplier variables. By “unwinding” the multipliers, we mean the removal of the $\frac{\partial}{\partial \boldsymbol{\lambda}}$ -type derivatives. In the end, we will only be left with derivatives of the variables \mathbf{x} which are the only variables which may occur nonlinearly in the given system f .

In the k -th deflation stage, we will then need $\frac{\partial A}{\partial \mathbf{x}}, \frac{\partial^2 A}{\partial \mathbf{x}^2}, \dots, \frac{\partial^k A}{\partial \mathbf{x}^k}$. If we execute (Equation 3.5) blindly, with disregard of the symmetry, we end up with n^k matrices, e.g., for $n = 3$ and $k = 10$, we compute 59,049 3-by-3 polynomial matrices, which is quite discouraging. However, as $\frac{\partial^2 A}{\partial x_1 \partial x_2} = \frac{\partial^2 A}{\partial x_2 \partial x_1}$, we enumerate all different monomials in n variables of degree k which is considerably less than n^k , e.g., for $n = 3$ and $k = 10$ again, we now only have 66 distinct polynomial matrices to compute. To store $\frac{\partial A}{\partial \mathbf{x}}$, a tree with n children (some branches may be empty) is a natural data structure, see [36; 37].

3.3 A Directed Acyclic Graph of Jacobian Matrices

In this section, we explain by means of examples, how we build our data structures.

For example, consider $k = 2$:

$$A_2(\mathbf{x}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) = \begin{bmatrix} A_1 & \mathbf{0} \\ \left[\frac{\partial A_1}{\partial \mathbf{x}} \quad \frac{\partial A_1}{\partial \boldsymbol{\lambda}_1} \right] B_2 \boldsymbol{\lambda}_2 & A_1 B_2 \\ \mathbf{0} & \mathbf{h}_2 \end{bmatrix}. \quad (3.8)$$

The evaluation of A_2 requires

$$A_1(\mathbf{x}, \boldsymbol{\lambda}_1) = \begin{bmatrix} A & \mathbf{0} \\ \left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1 \boldsymbol{\lambda}_1 & AB_1 \\ \mathbf{0} & \mathbf{h}_1 \end{bmatrix} \quad (3.9)$$

and the derivatives

$$\frac{\partial A_1}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial A}{\partial \mathbf{x}} & \mathbf{0} \\ \left[\frac{\partial^2 A}{\partial \mathbf{x}^2}\right] B_1 \boldsymbol{\lambda}_1 & \left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1 \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad \frac{\partial A_1}{\partial \boldsymbol{\lambda}_1} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial A}{\partial \mathbf{x}}\right] * B_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (3.10)$$

For $\frac{\partial A}{\partial \mathbf{x}}$ as in (Equation 3.5), the product $\left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1$ is $\left[\frac{\partial A}{\partial x_1} B_1 \quad \frac{\partial A}{\partial x_2} B_1 \quad \dots \quad \frac{\partial A}{\partial x_n} B_1\right]$. On the other hand, $\left[\frac{\partial A}{\partial \mathbf{x}}\right] * B_1$ uses the operator $*$ which treats its second argument as a vector of columns, i.e. if $B_1 = (b_1, b_2, \dots, b_m)$ where b_i ($1 \leq i \leq m = R_1 + 1$) are the columns, then

$$\left[\frac{\partial A}{\partial \mathbf{x}}\right] * B_1 = \left[\frac{\partial A}{\partial \mathbf{x}} b_1 \quad \frac{\partial A}{\partial \mathbf{x}} b_2 \quad \dots \quad \frac{\partial A}{\partial \mathbf{x}} b_m\right]. \quad (3.11)$$

One may evaluate $\left[\frac{\partial A}{\partial \mathbf{x}}\right] * B_1$ by computing the product $\left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1$ followed by alternately permuting the columns of the result. By virtue of this operator $*$, we may write

$$\frac{\partial}{\partial \boldsymbol{\lambda}_1} \left(\left[\frac{\partial A}{\partial \mathbf{x}}\right] B_1 \boldsymbol{\lambda}_1 \right) = \left[\frac{\partial A}{\partial \mathbf{x}}\right] * \frac{\partial}{\partial \boldsymbol{\lambda}_1} (B_1 \boldsymbol{\lambda}_1) = \left[\frac{\partial A}{\partial \mathbf{x}}\right] * B_1. \quad (3.12)$$

All matrices share the same typical structure: the critical information is in the first two entries of the first column. To evaluate A_2 , the matrices we need to store are the Jacobian matrix A of the given system and its derivatives $\frac{\partial A}{\partial \mathbf{x}}$ and $\frac{\partial^2 A}{\partial \mathbf{x}^2}$. The role of the multipliers λ_1 and λ_2 in the evaluation is strictly linear, they occur only in matrix-vector products.

For $k = 3$, the evaluation of

$$A_3(\mathbf{x}, \lambda_1, \lambda_2, \lambda_3) = \begin{bmatrix} A_2 & \mathbf{0} \\ \left[\frac{\partial A_2}{\partial \mathbf{x}} \quad \frac{\partial A_2}{\partial \lambda_1} \quad \frac{\partial A_2}{\partial \lambda_2} \right] B_3 \lambda_3 & A_2 B_3 \\ \mathbf{0} & \mathbf{h}_3 \end{bmatrix} \quad (3.13)$$

requires the evaluation of $A_2(\mathbf{x}, \lambda_1, \lambda_2)$ – which we considered above – and its partial derivatives

$\frac{\partial A_2}{\partial \mathbf{x}}$, $\frac{\partial A_2}{\partial \lambda_1}$, $\frac{\partial A_2}{\partial \lambda_2}$ respectively listed below:

$$\frac{\partial A_2}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial A_1}{\partial \mathbf{x}} & \mathbf{0} \\ \left[\frac{\partial^2 A_1}{\partial \mathbf{x}^2} \quad \frac{\partial^2 A_1}{\partial \mathbf{x} \partial \lambda_1} \right] B_2 \lambda_2 & \left[\frac{\partial A_1}{\partial \mathbf{x}} \right] B_2 \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad (3.14)$$

$$\frac{\partial A_2}{\partial \lambda_1} = \begin{bmatrix} \frac{\partial A_1}{\partial \lambda_1} & \mathbf{0} \\ \left[\frac{\partial^2 A_1}{\partial \lambda_1 \partial \mathbf{x}} \quad \frac{\partial^2 A_1}{\partial \lambda_1^2} \right] B_2 \lambda_2 & \left[\frac{\partial A_1}{\partial \lambda_1} \right] B_2 \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad (3.15)$$

and

$$\frac{\partial A_2}{\partial \lambda_2} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial A_1}{\partial \mathbf{x}} & \frac{\partial A_1}{\partial \lambda_1} \right] * B_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (3.16)$$

As the multipliers occur linearly, we have that $\frac{\partial^2 A_1}{\partial \lambda_1^2} = \mathbf{0}$. Observing $\frac{\partial^2 A_1}{\partial \lambda_1 \partial \mathbf{x}} = \frac{\partial^2 A_1}{\partial \mathbf{x} \partial \lambda_1}$, we have two more matrices to compute:

$$\frac{\partial^2 A_1}{\partial \mathbf{x}^2} = \begin{bmatrix} \frac{\partial^2 A}{\partial \mathbf{x}^2} & \mathbf{0} \\ \left[\frac{\partial^3 A}{\partial \mathbf{x}^3} \right] B_1 \lambda_1 & \left[\frac{\partial^2 A}{\partial \mathbf{x}^2} \right] B_1 \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad \text{and} \quad \frac{\partial^2 A_1}{\partial \mathbf{x} \partial \lambda_1} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial^2 A}{\partial \mathbf{x}^2} \right] * B_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (3.17)$$

Thus the evaluation of $A_3(\mathbf{x}, \lambda_1, \lambda_2, \lambda_3)$ requires $\left[A \frac{\partial A}{\partial \mathbf{x}} \frac{\partial^2 A}{\partial \mathbf{x}^2} \frac{\partial^3 A}{\partial \mathbf{x}^3} \right]$. The partial derivatives with respect to the multiplier variables in λ_1 , λ_2 , and λ_3 do not need to be computed explicitly.

Just as a tree is a natural data structure to store the derivatives of A , a tree is used to represent the deflation matrices. For memory efficiency, the same node should only be stored once and a remember table is used in the recursive creation of the tree, which is actually a directed acyclic graph, see Figure 9.

The graph in Figure 9 has 14 nodes. If we perform the deflation three times on a 10-by-10 system (10 equations in 10 variables), each time with the corank 1 (worst case scenario: maximal number of multipliers), then A_3 would be a 87-by-80 matrix and the graph of evaluated matrices would occupy about 347Kb (8 bytes for one complex number of two doubles). In case the rank

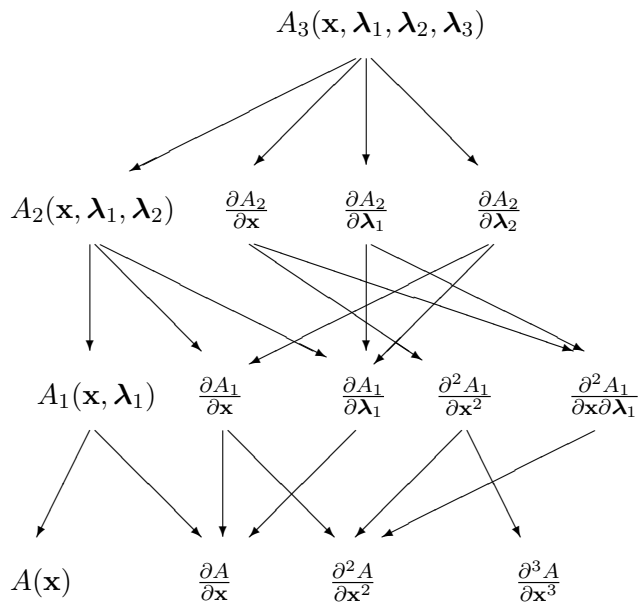


Figure 9. Directed acyclic graph illustrating the dependency relations in the evaluation of A_3 .

would always be zero (best case: only one multiplier in each case), then A_3 would be a 87-by-13 matrix and the graph of evaluated matrices would occupy about 257Kb.

The size of the graph grows modestly, see Table III. For example, for $k = 10$, we have 596 nodes. On the other hand, we must be careful to remove evaluated matrices when no longer needed. For $k = 10$ on a 10-by-10 system we would need at least 206Mb in the best case and 8Gb if the corank is always one.

We conclude this section with an anecdotal reference to actual timings, done on a 2.4 Ghz linux workstation, with the system cyclic 9-roots, illustrating that even for a modest number of deflations, it pays off to use a directed acyclic graph. Only to show the benefits of an efficient

k	1	2	3	4	5	6	7	8	9	10
#nodes	3	7	14	26	46	79	133	221	364	596

TABLE III

GROWTH OF THE NUMBER OF DISTINCT DERIVATIVE OPERATORS, AS NODES IN A DIRECTED ACYCLIC GRAPH, FOR INCREASING DEFLATIONS K .

evaluation, we deflate twice, assuming the corank is one each time, and end up with a 39-by-36 Jacobian matrix. It takes only 30 milliseconds to evaluate this matrix using the pre-calculated directed acyclic graph, which takes less than a second to set up. Computing the symbolic representation of this Jacobian matrix in a straightforward manner and evaluating it takes 25 minutes and 40 seconds!

3.4 The Deflation Algorithm in *PHCmaple*

Our Maple package *PHCmaple*[41] provides a convenient interface to the functions of *PHCpack*. The interface exploits the benefits of linking computer algebra with numerical software. *PHCmaple* is a first step in a larger project aimed at integration of a numerical solver in a computer algebra system.

Below we give an example of using the *PHCmaple* function `deflationStep` to deflate the system of equations `Ojika1` (copied from [60] and member of our benchmark collection in [42]):

$$\begin{cases} x^2 + y - 3 = 0; \\ x + 0.125y^2 - 1.5 = 0. \end{cases} \quad (3.18)$$

The system has two isolated solutions: $(x, y) = (-3, -6)$ which is regular and $(x, y) = (1, 2)$ which is multiple. The function `deflationStep` takes a system and a list of solutions approximations as parameters, constructs the deflated systems for all multiple roots in the list, and returns the list of lists of solutions. The latter are being grouped according to the rank of the Jacobian at the points, since for the points with the same rank a common deflated system may be constructed.

```
> T := makeSystem([x,y], [], [x^2+y-3, x+0.125*y^2-1.5]):
```

```
> sols := solve(T):
```

```
> printSolutions(T,sols);
```

```
(1) [x = 1.0000+.44913e-5*I, y = 2.0000-.89826e-5*I]
```

```
(2) [x = -3.0, y = -6.0]
```

```
(3) [x = 1.0000+.60400e-5*I, y = 2.0000-.12080e-4*I]
```

```
(4) [x = 1.0000-.38258e-5*I, y = 2.0000+.76515e-5*I]
```

To each group of points corresponds a table:

```
> l := map(i->table(i),deflationStep(sols,T)):
```

Solution (2) is simple (the rank of its Jacobian is full), the cluster of (1),(3),(4) approximates a multiple solution (rank = 1 < 2)

```
> map(g->[rank=g["rank"],num_points=nops(g["points"])],1);
```

$$[[rank = 2, num_points = 1], [rank = 1, num_points = 3]]$$

The multipliers used in the deflation step are

```
> multipliers := l[2] ["multipliers"];
```

$$multipliers := [\lambda_1, \lambda_2]$$

The deflated system is linear in the multipliers (the matrix below is a part of (1) for $k=1$)

```
> DT := l[2] ["deflated system"]:
```

```
> eqns :=
```

```
> map(p->p=0,DT:-polys[nops(T:-polys)+1..nops(DT:-polys)]):
```

```
> matrix(2,1,[A[0]*B[1],h[1]])=evalf[3](linalg[genmatrix](eqns,
```

```
> multipliers, b));
```

$$\begin{bmatrix} A_0 B_1 \\ h_1 \end{bmatrix} = \begin{bmatrix} -1.51x - 1.31Ix + 0.786 - 0.618I & 0.911x + 1.78Ix - 0.0562 + 0.998I \\ 0.197y - 0.154Iy - 0.755 - 0.656I & -0.0140y + 0.250Iy + 0.455 + 0.890I \\ -0.681 + 0.732I & -0.115 - 0.993I \end{bmatrix}$$

```
> DTmu := subsVariables
```

```
(DT,[seq(lambda[i]=mu[i],i=1..l[2] ["rank"]+1)]):
```

```
> DTmu:-vars;
```

$$[x, y, \mu_1, \mu_2]$$

```
> g := table(deflationStep(l[2] ["points"],DTmu)[1]):
```

```
> printSolutions(g["deflated system"],g["points"]);
```

```
(1) [x = 1.0-.87360e-15*I, y = 2.0+.18332e-14*I,
     mu[1] = -1.2402-.53612e-1*I, mu[2] = -.87951+.15347e-1*I,
     lambda[1] = -.85183-.21804*I, lambda[2] = -.91795+.57730*I,
     lambda[3] = .64096-.77868*I, lambda[4] = -.21656-.75758*I]
```

Get the multiplicity of the multiple solution:

```
> mult_solution := g["points"][1]: mult_solution:-mult;
```

3

Compare conditioning to that of the original approximation.

```
> mult_solution:-rco , sols[1]:-rco;
```

0.02542, 0.9301 10^{-11}

Above we have seen that the powerful symbolic-numeric capabilities of Maple are useful for presenting the original and deflated systems in a compact, non-expanded form. In the future we plan to represent the sequence of consecutive deflations simply by the sequences of matrices and vectors A_k , B_k , \mathbf{h}_k . Combined with the approach presented here, this would lead to a more efficient evaluation, as well as provide a better control over the deflation process to the user.

CHAPTER 4

HIGHER ORDER DEFLATION FOR POLYNOMIAL SYSTEMS WITH ISOLATED SINGULAR SOLUTIONS

Given an approximation to a multiple isolated solution of a system of polynomial equations, we provided a symbolic-numeric deflation algorithm to restore the quadratic convergence of Newton's method. Using first order derivatives of the polynomials in the system, our first order deflation method creates an augmented system that has the multiple isolated solution of the original system as a regular solution.

In this chapter we consider two approaches to computing the “multiplicity structure” at a singular isolated solution. An idea coming from one of them gives rise to our new higher-order deflation method. Using higher order partial derivatives of the original polynomials, the new algorithm reduces the multiplicity faster than our first method for systems which require several first order deflation steps. In particular: the number of higher order deflation steps is bounded by the number of variables.

This chapter corresponds to the paper [44], appears in the IMA Volume on Algorithms in Algebraic Geometry.

4.1 Statement of the Main Theorem & Algorithms

The matrices $A^{(d)}(\mathbf{x})$ we introduce below coincide for $d = 1$ with the Jacobian matrix of a polynomial system. They are generalizations of the Jacobian matrix, built along the same

construction as the matrices used in the computation of the multiplicity by Dayton and Zeng in [7].

Definition 4.1.1 The *deflation matrix* $A^{(d)}(\mathbf{x})$ of a polynomial system $F = (f_1, f_2, \dots, f_N)$ of N equations in n unknowns $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is a matrix with elements in $\mathbb{C}[\mathbf{x}]$. The rows of $A^{(d)}(\mathbf{x})$ are indexed by $\mathbf{x}^\alpha f_j$, where $|\alpha| < d$ and $j = 1, 2, \dots, N$. The columns are indexed by partial differential operators $\partial^\beta = \frac{\partial^{|\beta|}}{\partial x_1^{\beta_1} \dots \partial x_n^{\beta_n}}$, where $\beta \neq \mathbf{0}$ and $|\beta| \leq d$. The element at row $\mathbf{x}^\alpha f_j$ and column ∂^β of $A^{(d)}(\mathbf{x})$ is

$$\partial^\beta \cdot (\mathbf{x}^\alpha f_j) = \frac{\partial^{|\beta|}(\mathbf{x}^\alpha f_j)}{\partial \mathbf{x}^\beta}. \quad (4.1)$$

$A^{(d)}(\mathbf{x})$ has N_r rows and N_c columns, $N_r = N \cdot \binom{n+d-1}{n}$ and $N_c = \binom{n+d}{n} - 1$.

The number d will be referred to as the *order of deflation*.

corresponds to what we call a *deflation operator* – a linear differential operator with constant coefficients λ_β

$$Q = \sum_{\beta \neq \mathbf{0}, |\beta| \leq d} \lambda_\beta \partial^\beta \in \mathbb{C}[\partial]. \quad (4.2)$$

We use Q to define N_r new equations

$$g_{j,\alpha}(\mathbf{x}) = Q \cdot (\mathbf{x}^\alpha f_j) = 0, \quad j = 1, 2, \dots, N, \quad |\alpha| < d. \quad (4.3)$$

When we consider λ_β as indeterminate, we write $g_{j,\alpha}$ as $g_{j,\alpha}(\mathbf{x}, \boldsymbol{\lambda})$. In that case, we define m additional linear equations, for $m = \text{corank}(A^{(d)}(\mathbf{x}^*))$:

$$h_k(\boldsymbol{\lambda}) = \sum_{\beta} b_{k,\beta} \lambda_\beta - 1 = 0, \quad k = 1, 2, \dots, m, \quad (4.4)$$

where the coefficients $b_{k,\beta}$ are randomly chosen complex numbers. Then we define

$$G^{(d)}(\mathbf{x}, \boldsymbol{\lambda}) = \begin{cases} f_j(\mathbf{x}) = 0, & j = 1, 2, \dots, N; \\ g_{j,\alpha}(\mathbf{x}, \boldsymbol{\lambda}) = 0, & j = 1, 2, \dots, N, \quad |\alpha| < d; \\ h_k(\boldsymbol{\lambda}) = 0, & k = 1, 2, \dots, m. \end{cases} \quad (4.5)$$

With (Equation 4.5) we end Definition 4.1.3.

Now we are ready to state our main theorem.

Theorem 4.1.4 *Let $\mathbf{x}^* \in \mathbb{C}^n$ be an isolated solution of $F(\mathbf{x}) = \mathbf{0}$. Consider the system $G^{(d)}(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$ as in (Equation 4.5). For a generic choice of coefficients $b_{k,\beta}$, there exists*

a unique $\boldsymbol{\lambda}^* \in \mathbb{C}^{N_c}$ such that the system $G^{(d)}(\mathbf{x}, \boldsymbol{\lambda})$ has an isolated solution at $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$. Moreover, the multiplicity of $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ in $G(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$ is strictly less than the multiplicity of \mathbf{x}^* in $F(\mathbf{x}) = \mathbf{0}$.

Remark 4.1.5 Assuming coefficients $(b_{k,\beta})$ are chosen from a complex parameter space, the exceptional set of $(b_{k,\beta})$ that do not produce the conclusion of the Theorem 4.1.4 is contained in a (closed proper) algebraic subset of this parameter space. By genericity we mean staying away from this exceptional set, which is accomplished by choosing random numbers.

To determine the order d , we propose Algorithm 3. This d is then used in Algorithm 4.

Algorithm 3 $d = \text{MinOrderForCorankDrop}(F, \mathbf{x}_0, \varepsilon)$

Require: F is a finite set of polynomials;

$\mathbf{x}^0 \approx \mathbf{x}^*$, $\mathbf{x}^* \in F^{-1}(\mathbf{0})$, an isolated multiple solution;

$\varepsilon \geq 0$, a threshold parameter.

Ensure: d is the minimal number such that the system $G^{(d)}$ given via a generic deflation operator Q of order d has corank of the Jacobian at \mathbf{x}^* lower than $\text{corank } A(\mathbf{x}^*)$.

take a generic vector $\gamma = (\gamma_1, \dots, \gamma_n) \in \ker A(\mathbf{x}^0)$;

let $H(t) = F(\mathbf{x}^0 + \gamma t) = F(x_1^0 + \gamma_1 t, \dots, x_n^0 + \gamma_n t)$;

$d := \min\{a \mid \varepsilon < \text{coefficient of } t^a \text{ in } H(t)\} - 1$.

See the proof of correctness of this algorithm in the exact setting, i.e., $\mathbf{x}^* = \mathbf{x}^0$ and $\varepsilon = 0$, in the end of subsection 4.4.2.

Algorithm 4 $D^{(d)}F = \text{Deflate}(F, d, \mathbf{x}_0)$

Require: F is a finite set of polynomials in $\mathbb{C}[\mathbf{x}]$;

d is the order of deflation;

$\mathbf{x}^0 \approx \mathbf{x}^*$, $\mathbf{x}^* \in F^{-1}(\mathbf{0})$, an isolated multiple solution.

Ensure: $D^{(d)}F$ is a finite set of polynomials in $\mathbb{C}[\mathbf{x}, \boldsymbol{\lambda}]$ such that there is $\boldsymbol{\lambda}^*$ with $\mu_{D^{(d)}F}(\mathbf{x}^*, \boldsymbol{\lambda}^*) < \mu_F(\mathbf{x}^*)$.

determine the numerical corank m of $A^{(d)}(\mathbf{x}^0)$;

return $D^{(d)}F := G^{(d)}(\mathbf{x}, \boldsymbol{\lambda})$ as in (Equation 4.5).

Ideally, it would be nice to have a method to predict a number d such that the system can be regularized by a single deflation of order d . However, at this point, the iterative application of Algorithms 3 and 4 is the best practical strategy; This gives a procedure that deflates the system completely in at most the number of variables steps, as opposed to the ordinary deflation that may take more steps.

4.2 Multiplicity Structure

This section relates two different ways to obtain the multiplicity of an isolated solution, constructing its *multiplicity structure*. Note that by a “multiplicity structure” – a term without a precise mathematical definition – we mean any structure which provides more local information about the singular solution in addition to its multiplicity. In this section we mention two different approaches to describe this so-called multiplicity structure.

Example 4.2.1 (Running example 1) Consider the system

$$F(\mathbf{x}) = \begin{cases} x_2^3 = 0; \\ x_1^2 x_2^2 = 0; \\ x_1^4 + x_1^3 x_2 = 0. \end{cases} \quad (4.6)$$

The system $F(\mathbf{x}) = \mathbf{0}$ has only one isolated solution at $(0, 0)$ of high multiplicity. Below we will show how to compute the multiplicity of $(0, 0)$.

4.2.1 Standard Bases

Assume $\mathbf{0} \in \mathbb{C}^n$ is an isolated solution of the system $F(\mathbf{x}) = \mathbf{0}$. Let $I = \langle F \rangle \subset R = \mathbb{C}[\mathbf{x}]$ be the ideal generated by the polynomials in the system. Given a local monomial order \geq , the initial ideal $\mathbf{in}_{\geq}(I) = \{\mathbf{in}_{\geq}(f) \mid f \in I\} \subset R$ describes the multiplicity structure of $\mathbf{0}$ by means of *standard monomials*, i.e., monomials that are not contained in $\mathbf{in}_{\geq}(I)$. A graphical representation of a monomial ideal is a monomial *staircase*.

Example 4.2.2 (Initial ideals with respect to a local order) Consider the system (Equation 4.6) of Example 4.2.1.

Figure 10 shows the staircases for initial ideals of $I = \langle F \rangle$ w.r.t. two local weight orders \geq_{ω} . Computer algebra systems Macaulay 2 [23] and Singular [26] can be used for these kind of computations, see also [24; 25] for theory, in particular, on Mora's tangent cone algorithm [55].

In the example the leading monomials at the corners of the staircase come from the elements of the corresponding *standard basis*. For the weight vector $w = (-1, -2)$ the original generators give such a basis (initial terms underlined). For $w = (-2, -1)$ one more polynomial is needed.

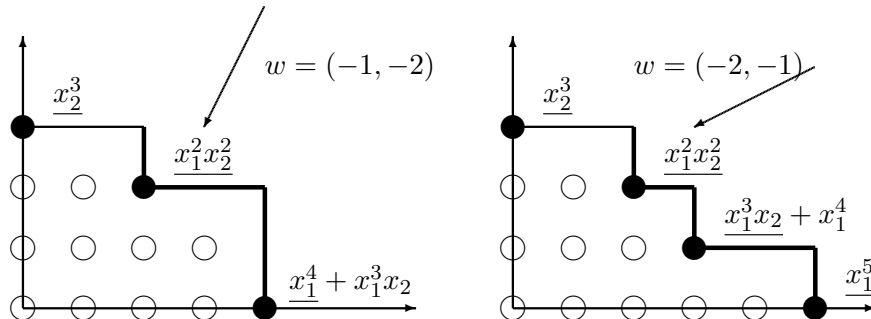


Figure 10. Two monomial staircases for two different monomial orderings applied to the same system. The full circles represent the generators of the initial ideals. The multiplicity is the number of standard monomials, represented by the empty circles under the staircase.

4.2.2 Dual Space of Differential Functionals

Another approach at the multiplicity structure is described in detail in [72; 75]; see also [58].

Using duality to define the multiplicity of a solution goes back to Macaulay [51]. In this approach, *differential functionals* are denoted by

$$\Delta_{\alpha}(f) = \frac{1}{\alpha_1! \cdots \alpha_n!} \cdot \frac{\partial^{|\alpha|} f}{\partial x^{\alpha_1} \cdots \partial x^{\alpha_n}} \Big|_{\mathbf{x}=\mathbf{0}}. \quad (4.7)$$

Observe that

$$\Delta_{\alpha}(\mathbf{x}^{\beta}) = \begin{cases} 1, & \alpha = \beta \\ 0, & \alpha \neq \beta. \end{cases} \quad (4.8)$$

We then define the local *dual space* of differential functionals $D_{\mathbf{0}}[I]$ as

$$D_{\mathbf{0}}[I] = \{L \in \text{Span}\{\Delta_{\alpha} \mid \alpha \in \mathbb{Z}_{\geq 0}^n\} \mid L(f) = 0 \text{ for all } f \in I\}, \quad (4.9)$$

Example 4.2.3 (Dual space of running example 1) For the ideal defined by the polynomials in the system (Equation 4.6) we have

$$D_{\mathbf{0}}[I] = \text{Span}\left\{ \begin{array}{l} \underline{\Delta}_{(4,0)} - \Delta_{(3,1)}, \underline{\Delta}_{(3,0)}, \underline{\Delta}_{(2,1)}, \underline{\Delta}_{(1,2)}, \\ \underline{\Delta}_{(2,0)}, \underline{\Delta}_{(1,1)}, \underline{\Delta}_{(0,2)}, \underline{\Delta}_{(1,0)}, \underline{\Delta}_{(0,1)}, \underline{\Delta}_{(0,0)} \end{array} \right\}. \quad (4.10)$$

Notice that here the basis of the dual space is chosen in such a way that the (underlined) leading terms with respect to the weight order $\geq_{(2,1)}$ correspond to the monomials under the staircase in Example 4.2.1 for the order $\geq_{(-2,-1)}$. We will show that it is not a coincidence later in this section.

4.2.3 Dual Bases versus Standard Bases

Since both local dual bases and initial ideals w.r.t. local orders describe the same, there exists a natural correspondence between the two.

Let \geq be an order on the nonnegative integer lattice $\mathbb{Z}_{\geq 0}^n$ that defines a local monomial order and let \succeq be the opposite of \geq , i.e., $\alpha \succeq \beta \Leftrightarrow \alpha \leq \beta$. (Note: \succeq defines a global monomial order.)

For a linear differential functional $L = \sum c_{\alpha} \Delta_{\alpha}$ define the *support*: $\text{supp}(L) = \{\alpha \in \mathbb{Z}_{\geq 0}^n \mid c_{\alpha} \neq 0\}$. For the dual space, $\text{supp}(D_{\mathbf{0}}[I]) = \bigcup_{L \in D_{\mathbf{0}}[I]} \text{supp}(L)$.

Using the order \succeq we can talk about the leading or *initial term* of L : let $\mathbf{in}_{\succeq}(L)$ be the maximal element of $\text{supp}(L)$ with respect to \succeq . Define the *initial support* of the dual space as $\mathbf{in}_{\succeq}(D_{\mathbf{0}}[I]) = \{\mathbf{in}_{\succeq}(L) \mid L \in D_{\mathbf{0}}[I]\}$. The initial support is obviously contained in the support, in our running example the containment is proper:

$$\begin{aligned} \mathbf{in}_{(2,1)}(D_{\mathbf{0}}[I]) &= \{(i, j) \mid i + j \leq 3\} \cup \{(4, 0)\} \\ &\subset \{(i, j) \mid i + j \leq 3\} \cup \{(4, 0) \cup (3, 1)\} = \text{supp}(D_{\mathbf{0}}[I]). \end{aligned}$$

Theorem 4.2.4 *The number of elements in the initial support equals the dimension of the dual space, therefore, is the multiplicity. Moreover, with the above assumptions on the orders \geq and \succeq , the standard monomials w.r.t. the local order \geq are $\{\mathbf{x}^{\alpha} \mid \alpha \in \mathbf{in}_{\succeq}(D_{\mathbf{0}}[I])\}$.*

Proof. Pick $L_{\beta} \in D_{\mathbf{0}}[I], \beta \in \mathbf{in}_{\succeq}(D_{\mathbf{0}}[I])$ such that $\mathbf{in}_{\succeq}(L_{\beta}) = \beta$. One can easily show that $\{L_{\beta}\}$ is a basis of $D_{\mathbf{0}}[I]$.

Take a monomial $\mathbf{x}^{\alpha} \in \mathbf{in}_{\geq}(I)$, then there is $f \in I$ such that $\mathbf{x}^{\alpha} = \mathbf{in}_{\geq}(f)$. Next, take any linear differential functional L with $\mathbf{in}_{\succeq}(L) = \alpha$. Since the orders \geq and \succeq are opposite, there are no similar terms in the tail of L and the tail of f , therefore, $L(f) = \mathbf{in}_{\succeq}(L)(\mathbf{in}_{\geq}(f)) \neq 0$.

It follows that, $L \notin D_{\mathbf{0}}[I]$, which proves that the set of standard monomials is contained in the initial support of $D_{\mathbf{0}}[I]$. They are equal since they both determine the dimension. \square

Consider the ring of linear differential operators $\mathcal{D} = \mathbb{C}[\partial]$ with the natural action (denoted by “.”) on polynomial ring $R = \mathbb{C}[\mathbf{x}]$.

Lemma 4.2.5 *Let $Q \in \mathbb{C}[\partial]$ and $f \in \mathbb{C}[\mathbf{x}]$ such that $\mathbf{in}_{\succeq}(Q) \succeq \mathbf{in}_{\succeq}(f)$ (in $\mathbb{Z}_{\geq 0}^n$).*

Then $\mathbf{in}_{\succeq}(Q \cdot f) = \mathbf{in}_{\succeq}(f) - \mathbf{in}_{\succeq}(Q) \in \mathbb{Z}_{\geq 0}^n$.

4.3 Computing the Multiplicity Structure

Let the ideal I be generated by f_1, f_2, \dots, f_N . Let $D_{\mathbf{0}}^{(d)}[I]$ the part of $D_{\mathbf{0}}[I]$ containing functionals of order at most d . We would like to have a criterion that for the differential functional L of degree at most d guarantees $L \in D_{\mathbf{0}}^{(d)}[I]$.

Below we describe two such criteria referred to as *closedness conditions*; their names are arranged to match the corresponding computational techniques of Dayton-Zeng [7] and Stetter-Thallinger [73] that we will describe later respectively as the DZ and ST algorithms.

A functional $L = \sum c_{\alpha} \Delta_{\alpha}$ with $c_{\alpha} \in \mathbb{C}$ of order d belongs to the dual space $D_{\mathbf{0}}[I]$ if and only if

- **(DZ-closedness)** $L(g \cdot f_i) = 0$ for all $i = 1, 2, \dots, N$ and polynomials $g(\mathbf{x})$ of degree at most $d - 1$.
- **(ST-closedness)** $L(f_i) = 0$ for all i and $\sigma_j(L) \in D_{\mathbf{0}}[I]$ for all $j = 1, 2, \dots, n$, where $\sigma_j : D_{\mathbf{0}}[I] \rightarrow D_{\mathbf{0}}[I]$ is a linear map such that

$$\sigma_j(\Delta_{\alpha}) = \begin{cases} 0, & \text{if } \alpha_j = 0, \\ \Delta_{\alpha - e_j}, & \text{otherwise.} \end{cases} \quad (4.11)$$

The basic idea of both DZ and ST algorithms is the same: build up a basis of $D_{\mathbf{0}}$ incrementally by computing $D_{\mathbf{0}}^{(d)}$ for $d = 1, 2, \dots$ using the corresponding closedness condition. The computation stops when $D_{\mathbf{0}}^{(d)} = D_{\mathbf{0}}^{(d-1)}$.

Example 4.3.1 (Running example 2) Consider the system in $\mathbb{C}[x_1, x_2]$ given by three polynomials $f_1 = x_1x_2$, $f_2 = x_1^2 - x_2^2$, and $f_3 = x_2^4$, which has only one isolated root at $(0, 0)$.

4.3.1 The Dayton-Zeng Algorithm

We shall outline only a summary of this approach, see [7] for details.

If $\mathbf{0}$ is a solution of the system, then $D_{\mathbf{0}}^{(0)} = \text{Span}\{\Delta_{\mathbf{0}}\}$.

At step $d > 0$, we compute $D_{\mathbf{0}}^{(d)}$. Let the functional

$$L = \sum_{|\alpha| \leq d, \alpha \neq \mathbf{0}} c_{\alpha} \Delta_{\alpha} \quad (4.12)$$

belong to the dual space $D_{\mathbf{0}}^{(d)}$. Then the vector of coefficients c_{α} is in the kernel of the following matrix $M_{DZ}^{(d)}$ with $NB(d-1)$ rows and $B(d)-1$ columns, where $B(d) = \binom{n+d}{n}$ is the number of monomials in n variables of degree at most d .

The rows of $M_{DZ}^{(d)}$ are labelled with $x^{\alpha}f_j$, where $|\alpha| < d$ and $j = 1, 2, \dots, N$. The columns correspond to Δ_{β} , where $\beta \neq \mathbf{0}$, $|\beta| \leq d$.

$$[\text{The entry of } M_{DZ}^{(d)} \text{ in row } x^{\alpha}f_j \text{ and column } \Delta_{\beta}] = \Delta_{\beta}(x^{\alpha}f_j). \quad (4.13)$$

At the step $d = 3$ we have the following $M_{DZ}^{(3)}$

	$\Delta_{(1,0)}$	$\Delta_{(0,1)}$	$\Delta_{(2,0)}$	$\Delta_{(1,1)}$	$\Delta_{(0,2)}$	$\Delta_{(3,0)}$	$\Delta_{(2,1)}$	$\Delta_{(1,2)}$	$\Delta_{(0,3)}$
f_1	0	0	0	1	0	0	0	0	0
f_2	0	0	1	0	-1	0	0	0	0
f_3	0	0	0	0	0	0	0	0	0
$x_1 f_1$	0	0	0	0	0	0	1	0	0
$x_1 f_2$	0	0	0	0	0	1	0	-1	0
$x_1 f_3$	0	0	0	0	0	0	0	0	0
$x_2 f_1$	0	0	0	0	0	0	0	1	0
$x_2 f_2$	0	0	0	0	0	0	1	0	-1
$x_2 f_3$	0	0	0	0	0	0	0	0	0
$x_1^2 f_1$	0	0	0	0	0	0	0	0	0
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

Note that the last block of 9 rows is entirely zero.

Analyzing the kernel of this matrix one sees that there are no functionals of degree 3 in the dual space, which is then equal to $D_0^{(2)}[I]$

$$D_0[I] = \text{Span}\{\Delta_{(0,0)}, \Delta_{(1,0)}, \Delta_{(0,1)}, \Delta_{(2,0)} + \Delta_{(0,2)}\}. \quad (4.14)$$

4.3.2 The Stetter-Thallinger Algorithm

The matrix $M_{ST}^{(d)}$ is a matrix consisting of $n + 1$ blocks stacked on top of each other:

- The top block contains the first N rows of $M_{DZ}^{(d)}$;
- For every $j = 1, 2, \dots, n$, let $S_j^{(d)}$ be the $(B(d-1) - 1) \times (B(d) - 1)$ -matrix for the linear map

$$\sigma_j : D_{\mathbf{0}}^{(d)} / \text{Span}\{\Delta_{\mathbf{0}}\} \rightarrow D_{\mathbf{0}}^{(d-1)} / \text{Span}\{\Delta_{\mathbf{0}}\} \quad (4.15)$$

w.r.t. standard bases of functionals.

The block $M_{ST}^{(d-1)} S_j$ represents the closedness condition for the “anti-derivation” σ_j .

Let us go through the steps of the algorithm for the Example 4.3.1.

Step 1. At the beginning we have $M_{ST}^{(1)}$ equal to

	$\Delta_{(1,0)}$	$\Delta_{(0,1)}$
f_1	0	0
f_2	0	0
f_3	0	0

Therefore, $D_{\mathbf{0}}^{(1)} = \text{Span}\{\Delta_{(0,0)}, \Delta_{(1,0)}, \Delta_{(0,1)}\}$.

The matrix $S_2^{(3)}$ can be defined similarly.

The top block of the matrix $M_{ST}^{(3)}$ is

	$\Delta_{(1,0)}$	$\Delta_{(0,1)}$	$\Delta_{(2,0)}$	$\Delta_{(1,1)}$	$\Delta_{(0,2)}$	$\Delta_{(3,0)}$	$\Delta_{(2,1)}$	$\Delta_{(1,2)}$	$\Delta_{(0,3)}$
f_1	0	0	0	1	0	0	0	0	0
f_2	0	0	1	0	-1	0	0	0	0
f_3	0	0	0	0	0	0	0	0	0

Despite the last 4 columns being 0, there are no new elements of order 3 in the dual space due to the other two blocks: $\tilde{M}_{ST}^{(2)}S_1^{(3)}$:

	$\Delta_{(1,0)}$	$\Delta_{(0,1)}$	$\Delta_{(2,0)}$	$\Delta_{(1,1)}$	$\Delta_{(0,2)}$	$\Delta_{(3,0)}$	$\Delta_{(2,1)}$	$\Delta_{(1,2)}$	$\Delta_{(0,3)}$
x_1f_1	0	0	0	0	0	0	1	0	0
x_1f_2	0	0	0	0	0	1	0	-1	0

and $\tilde{M}_{ST}^{(2)}S_2^{(3)}$:

	$\Delta_{(1,0)}$	$\Delta_{(0,1)}$	$\Delta_{(2,0)}$	$\Delta_{(1,1)}$	$\Delta_{(0,2)}$	$\Delta_{(3,0)}$	$\Delta_{(2,1)}$	$\Delta_{(1,2)}$	$\Delta_{(0,3)}$
x_2f_1	0	0	0	0	0	0	0	1	0
x_2f_2	0	0	0	0	0	0	1	0	-1

Comparing to DZ algorithm, in step 3, we managed to avoid the computation of 9 last zero rows of $M_{DZ}^{(3)}$ in this particular example. We now also see how its 4 last nonzero rows show up in the “closedness condition” blocks of $M_{DZ}^{(3)}$.

4.4 Proofs and Algorithmic Details

In this section we justify the main theorems stated before and give details about the algorithms presented above.

4.4.1 First Order Deflation

In this section we summarize our deflation method introduced in Chapter 2 or [44]. Not only it is done for the convenience of the reader, but also for our own convenience as we plan to build a higher order deflation algorithm in Section 4.4 using the algorithm following the pattern established in this section.

One deflation step with fixed λ . The basic idea of the method is relatively simple. Let $\lambda \in \mathbb{C}^n$ be a nonzero vector in $\ker(A(\mathbf{x}^*))$, then the equations

$$g_i(\mathbf{x}) = \lambda \cdot \nabla f_i(\mathbf{x}) = \sum_{j=1}^n \lambda_j \frac{\partial f_i(\mathbf{x})}{\partial x_j}, \quad i = 1, 2, \dots, N \quad (4.16)$$

have \mathbf{x}^* as a solution. Moreover,

Theorem 4.4.1 *The augmented system*

$$G(\mathbf{x}) = (f_1, \dots, f_N, g_1, \dots, g_N)(\mathbf{x}) = \mathbf{0} \quad (4.17)$$

of equations in $\mathbb{C}[\mathbf{x}]$ is a deflation of the original system $F(\mathbf{x}) = \mathbf{0}$ at x^ , i.e. $G(\mathbf{x}^*) = 0$ and the multiplicity of the solution x^* is lower in the new system.*

The original proof of this statement in [42] uses the notion of a standard basis of the ideal $I = (f_1, f_2, \dots, f_N)$ in the polynomial ring $R = \mathbb{C}[\mathbf{x}]$ w.r.t. a local order; this tool of computational commutative algebra can be used to obtain the multiplicity of \mathbf{x}^* , which is defined as the \mathbb{C} -dimension of the local quotient ring $R_{\mathbf{x}}/R_x I$.

On the other hand it is in correspondence with another way of looking at multiplicities – dual spaces of local functionals, so the proof can be written in that language as well (see Section 4.2).

One deflation step with indeterminate $\boldsymbol{\lambda}$. Without loss of generality, we may assume $\text{corank}(A(\mathbf{x}^*)) = 1$; consult Chapter 2 or [42] to see how the general case is reduced to this. Consider $N + 1$ additional polynomials in $\mathbb{C}[\mathbf{x}, \boldsymbol{\lambda}]$ in $2n$ variables:

$$g_i(\mathbf{x}, \boldsymbol{\lambda}) = \boldsymbol{\lambda} \cdot \nabla f_i(\mathbf{x}) = \sum_{j=1}^n \lambda_j \frac{\partial f_i(\mathbf{x})}{\partial x_j}, \quad (i = 1, 2, \dots, N) \quad (4.18)$$

$$h(\boldsymbol{\lambda}) = \sum_{j=1}^n b_j \lambda_j - 1, \quad (4.19)$$

where the coefficients b_j are random complex numbers.

Theorem 4.4.2 *Let $\mathbf{x}^* \in \mathbb{C}^n$ be an isolated solution of $F(\mathbf{x}) = \mathbf{0}$ (in $\mathbb{C}[\mathbf{x}]$).*

For a generic choice of coefficients b_j , $j = 1, 2, \dots, n$, there exists a unique $\boldsymbol{\lambda}^ \in \mathbb{C}^n$ such that the system*

$$G(\mathbf{x}, \boldsymbol{\lambda}) = (f_1, \dots, f_N, g_1, \dots, g_N, h)(\mathbf{x}, \boldsymbol{\lambda}) = 0 \quad (4.20)$$

of equations in $\mathbb{C}[\mathbf{x}, \boldsymbol{\lambda}]$ has an isolated solution at $(\mathbf{x}^, \boldsymbol{\lambda}^*)$.*

The multiplicity of $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ in $G(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$ is lower than that of \mathbf{x}^* in $F(\mathbf{x}) = \mathbf{0}$.

Proof. Follows from Proposition 3.4 in [44]. \square

Theorem 4.4.2 provides a recipe for the deflation algorithm: one simply needs to keep deflating until the solution of the augmented system corresponding to \mathbf{x}^* becomes regular.

As a corollary we have that the number of deflations needed to make a singular isolated solution \mathbf{x}^* regular is less than the multiplicity of \mathbf{x}^* .

4.4.2 Higher Order Deflation with Fixed Multipliers

We use the deflation operator to define an augmented system.

Theorem 4.4.3 *Let f_1, f_2, \dots, f_N form a standard basis of I w.r.t. the order opposite to \succeq .*

Consider the system $G^{(d)}(\mathbf{x}) = \mathbf{0}$ in $\mathbb{C}[\mathbf{x}]$, where

$$G^{(d)}(\mathbf{x}) = \begin{cases} f_j(\mathbf{x}) & (j = 1, 2, \dots, N) \\ g_{j,\alpha}(\mathbf{x}) & (j = 1, 2, \dots, N, |\alpha| < d) \end{cases}. \quad (4.21)$$

as in Definition 4.1.3.

(a) *The system $G^{(d)}(\mathbf{x}) = \mathbf{0}$ is a deflation of the original system $F(\mathbf{x}) = \mathbf{0}$ at \mathbf{x}^* .*

(b) *Let $I = (F)$ and $J = (G^{(d)})$ be the ideals generated by polynomials of the systems and \succeq be a global monomial order on $\mathbb{Z}_{\geq 0}^n$. Then the following relation holds for initial supports*

$$\mathbf{in}_{\succeq}(D_{\mathbf{0}}[J]) \subset \{\beta - \beta_Q \mid \beta \in \mathbf{in}_{\succeq}(D_{\mathbf{0}}[I])\} \cap \mathbb{Z}_{\geq 0}^n, \quad (4.22)$$

where β_Q is the maximal element of the set $\mathbf{in}_{\succeq}(D_{\mathbf{0}}[I]) \cap \{\beta : |\beta| \leq d\}$.

Proof. Let $\lambda \in \ker(A^{(d)}(\mathbf{x}^*))$ be the vector used above to construct the operator $Q \in \mathbb{C}[\boldsymbol{\theta}]$ and the equations $g_{j,\alpha}(\mathbf{x}) = 0$.

First of all, $g_{j,\alpha}(\mathbf{x}^*) = (Q \cdot (\mathbf{x}^\alpha f_j))|_{\mathbf{x}=\mathbf{x}^*} = 0$ provided $|\alpha| < d$ (by construction), hence, \mathbf{x}^* is a solution to $G^{(d)}(\mathbf{x}) = \mathbf{0}$.

To prove (a), it remains to show that the multiplicity drops, which follows from part (b) that is treated in the rest of this proof.

We shall assume for simplicity that $\mathbf{x}^* = \mathbf{0}$. This is done without the loss of generality using a linear change of coordinates: $\mathbf{x} \mapsto \mathbf{x} + \mathbf{x}^*$. It is important to note that in the new coordinates polynomials $Q \cdot (\mathbf{x}^\alpha f_j(\mathbf{x} + \mathbf{x}^*))$ generate the same ideal as the polynomials $Q \cdot ((\mathbf{x} - \mathbf{x}^*)^\alpha f_j(\mathbf{x} + \mathbf{x}^*))$.

Recall that $I = \langle F \rangle = \langle f_1, f_2, \dots, f_N \rangle$, let $J = \langle G^{(d)} \rangle \supset I$ be the ideal generated by the polynomials in the augmented system. The reversed containment holds for the dual spaces: $D_{\mathbf{0}}[I] \supset D_{\mathbf{0}}[J]$.

There is a 1-to-1 correspondence between linear differential operators and linear differential functionals:

$$\sum \lambda_\beta \boldsymbol{\theta}^\beta \longleftrightarrow \sum \lambda_\beta \beta! \Delta_\beta. \quad (4.23)$$

Let $\phi : \mathbb{C}[\boldsymbol{\theta}] \rightarrow D_{\mathbf{0}}$ and $\tau : D_{\mathbf{0}} \rightarrow \mathbb{C}[\boldsymbol{\theta}]$ be the corresponding bijections.

As in Section 4.2 we order terms Δ_β with \succeq , a global monomial order. Notice that since the choice of coefficients of the operator Q is generic, $\beta_Q = \mathbf{in}_{\succeq}(Q) = \mathbf{in}_{\succeq}(\phi(Q))$ is the maximal element of the set $\mathbf{in}_{\succeq}(D_{\mathbf{0}}[I]) \cap \{\beta : |\beta| \leq d\}$.

Next, we use the condition that f_i form a standard basis. Since the corners of the staircase correspond to the initial terms of f_i , by Lemma 4.2.5 the staircase created with the corners at $\mathbf{in}_{\geq}(Q \cdot (\mathbf{x}^\alpha f_i))$ bounds the set $\{\beta - \beta_Q \mid \beta \in \mathbf{in}_{\geq}(D_0[I])\} \cap \mathbb{Z}_{\geq 0}^n$, which, therefore, contains the initial support of $D_0[J]$. \square

Corollary 4.4.4 *If there exist a local monomial order \geq such that the minimal (standard) monomial in the set $\{\mathbf{x}^\alpha \notin \mathbf{in}_{\geq}(I) : |\alpha| \leq d\}$ is also minimal in the set of all standard monomials, then \mathbf{x}^* is a regular solution of $G^{(d)}(\mathbf{x}) = \mathbf{0}$.*

Theorem 4.4.3 and Corollary 4.4.4 are of purely theoretical nature, since the assumption of exactness in their statements can not be relaxed.

Ideally we would like to be able to drop the assumption of the original polynomials forming a standard basis, since computing such a basis is a complex symbolic task, whereas our interest lies in the further numericalization of the approach. The following weaker statement works around this restriction.

Assuming $\mathbf{x}^* = \mathbf{0}$, let $\text{supp}(F) = \bigcup_{j=1,2,\dots,N} \text{supp}(f_j)$.

Proposition 4.4.5 *Assume $A(\mathbf{0}) = \mathbf{0}$. Let $d_0 = \min\{|\alpha| : x^\alpha \in \text{supp}(F)\}$.*

Then, in the notation of Theorem 4.4.3, for a generic deflating operator Q the system $G^{(d)}(\mathbf{x}) = \mathbf{0}$, where, $d < d_0$ is a deflation of the original system $F(\mathbf{x}) = \mathbf{0}$ at the origin.

Moreover, if $d = d_0 - 1$ then the Jacobian of $G^{(d)}(\mathbf{0})$ is not equal to zero.

Proof. Fix a local monomial ordering that respects the degree. With the above assumptions, the initial ideal $\mathbf{in}(\langle F \rangle)$ will contain monomials of degree at least d_0 . On the other hand, for

a generic choice of the deflating operator Q the support $\text{supp}(G^{(d)})$ would contain a monomial of degree less than d_0 . Therefore, there exists a monomial in $\mathbf{in}(\langle \text{supp}(G^{(d)}) \rangle)$ that is not in $\mathbf{in}(\langle F \rangle)$, hence, $G^{(d)}$ is a deflation.

If $d = |d_0| - 1$, then there is such monomial of degree 1, which means that the Jacobian of the augmented system is nonzero. \square

Remark 4.4.6 Note that if the deflation order d is as in Proposition 4.4.5, then it suffices to take an arbitrary *homogeneous* deflation operator of order d .

Next we explain the practical value of Proposition 4.4.5. Let $K = \ker A(\mathbf{0})$ and $c = \text{corank } A(\mathbf{0}) = \dim K$. Without a loss of generality we may assume that K is the subspace of \mathbb{C}^n has $\{x_1, \dots, x_c\}$ as coordinates.

Now consider the system $F'(x_1, \dots, x_c) = F(x_1, \dots, x_c, 0, \dots, 0)$. This system has an isolated solution at the origin, and Proposition 4.4.5 is applicable, since the Jacobian is zero. Moreover, if we take the deflation of order $d = d_0 - 1$ of the original system F , with d_0 coming from the Proposition, the corank of the Jacobian the augmented system $G^{(d)}$ is guaranteed to be lower than that of $A(\mathbf{0})$.

Let us go back to the general setup: an arbitrary isolated solution \mathbf{x}^* , the Jacobian $A(\mathbf{x}^*)$ with a proper kernel K , etc. Algorithm 3 is a practical algorithm that can be executed numerically knowing only an approximation to \mathbf{x}^* .

Proof. [Proof of correctness of Algorithm 3 for $\mathbf{x}^0 = \mathbf{x}^*$ and $\varepsilon = 0$.] We can get to the special setting of Proposition 4.4.5 in two steps. First, apply an affine transformation that takes \mathbf{x}^*

to the origin and $\ker A(\mathbf{x}^*)$ to the subspace K of \mathbb{C}^n spanned by the first $c = \text{corank } A(\mathbf{x}^*)$ standard basis vectors. Second, make a new system $F'(x_1, \dots, x_c) = 0$ by substituting the $x_i = 0$ in F for $i > c$.

Let $\gamma' = (\gamma'_1, \dots, \gamma'_c) \in K$ be the image of the generic vector γ under the linear part of the affine transform. Then $H(t) = F'(\gamma'_1 t, \dots, \gamma'_c t)$.

Since γ' is generic, the lowest degree d_0 of the monomial in $\text{supp}(F')$ is equal to $\min\{a \mid t^a \in \text{supp } H(t)\}$. According to the Proposition 4.4.5 and the discussion that followed, $d = d_0 - 1$ is the minimal order of deflation that will reduce the rank of the system. \square

Remark 4.4.7 In view of Remark 4.4.6 it would be enough to use any *homogeneous* deflation operator of order d :

$$Q = \sum_{|\beta|=d} \lambda_\beta \partial^\beta \in \mathbb{C}[\partial], \quad (4.24)$$

such that the vector $\boldsymbol{\lambda}$ of its coefficients is in the kernel of the *truncated deflation matrix*, which contains only the rows corresponding to the original polynomials F and only the columns labelled with ∂^β with $|\beta| = d$.

4.4.3 Indeterminate Multipliers

As in Section 4.4.1, we now consider indeterminate λ_β . Now we should think of the differential operator $L(\boldsymbol{\lambda}) \in \mathbb{C}[\boldsymbol{\lambda}, \partial]$ and of additional equations $g_{j,\alpha}(\mathbf{x}, \boldsymbol{\lambda}) \in \mathbb{C}[\mathbf{x}, \boldsymbol{\lambda}]$ as depending on $\boldsymbol{\lambda}$.

Proof of Theorem 4.1.4. Picking $m = \text{corank}(A^{(d)}(\mathbf{x}^*))$ generic linear equations h_k guarantees that for $\mathbf{x} = \mathbf{x}^*$ the solution for λ exists and is unique; therefore, the first part of the statement is proved.

The argument for the drop in the multiplicity is similar to that of the proof of Theorem 4.4.2. □

4.5 Computational Experiments

We have implemented our new deflation methods in PHCpack [76] and Maple. Below we report on two examples.

One crucial decision in the deflation algorithm is the determination of the numerical rank, for which we may use SVD or QR in the rank-revealing algorithms. Both SVD and QR are numerically stable. As shown by the result from [12, page 118] for the problem of solving an overdetermined linear system $A\mathbf{x} = b$. The solution obtained by QR or SVD minimizes the residual $\|(A + \delta A)\tilde{\mathbf{x}} - (b + \delta b)\|_2$ where the relative errors have the same magnitude as the machine precision ϵ :

$$\max \left(\frac{\|\delta A\|_2}{\|A\|_2}, \frac{\|\delta b\|_2}{\|b\|_2} \right) = O(\epsilon). \quad (4.25)$$

To decide to tolerance for the numerical rank, we combine a fixed value – relative to the working precision – with the search for that value of i where the maximal jump σ_{i+1}/σ_i in the singular values σ_i 's occurs.

4.5.1 A First Example

To find initial approximations for the roots of the system

$$F(\mathbf{x}) = \begin{cases} x_1^3 + x_1x_2^2 = 0; \\ x_1x_2^2 + x_2^3 = 0; \\ x_1^2x_2 + x_1x_2^2 = 0. \end{cases} \quad (4.26)$$

we must first make the system “square”, i.e.: having as many equations as unknowns, so we may apply the homotopies available in PHCpack [76]. Using the embedding technique of [66] (see also [70]), we add one slack variable z to each equation of the system, multiplied by random complex constants γ_1 , γ_2 , and γ_3 :

$$E(\mathbf{x}, z) = \begin{cases} x_1^3 + x_1x_2^2 + \gamma_1z = 0; \\ x_1x_2^2 + x_2^3 + \gamma_2z = 0; \\ x_1^2x_2 + x_1x_2^2 + \gamma_3z = 0. \end{cases} \quad (4.27)$$

Observe that the solutions of the original system $F(\mathbf{x}) = \mathbf{0}$ occur as solutions of the embedded system $E(\mathbf{x}, z) = \mathbf{0}$ with slack variable $z = 0$. At the end points of the solution paths defined by a homotopy to solve $E(\mathbf{x}, z) = \mathbf{0}$, we find nine zeroes close to the origin. These nine approximate zeroes are the input to our deflation algorithm.

The application of our first deflation algorithm in [44] requires two stages. The Jacobian matrix of $F(\mathbf{x}) = \mathbf{0}$ has rank zero at $(0, 0)$. After the first deflation with one multiplier, the rank of the Jacobian matrix of the augmented system $G(\mathbf{x}, \boldsymbol{\lambda}_1) = \mathbf{0}$ equals one, so the

second deflation step uses two multipliers. After the second deflation step, the Jacobian matrix has full rank, and $(0, 0)$ has then become a regular solution. Newton's method on the final system then converges again quadratically and the solution can be approximated efficiently with great accuracy. Once the precise location of a multiple root is known, we are interested in its multiplicity. The algorithm of [7] reveals that the multiplicity of the isolated root equals seven.

Starting at a root of low accuracy, at a distance of 10^{-5} from the exact root, the numerical implementation of Algorithm 3 predicts two as the order, using 10^{-4} as the tolerance for the vanishing of the coefficients in the univariate interpolating polynomial. The Jacobian matrix of the augmented system $G^{(2)}$ has full rank so that a couple of iterations suffice to compute the root very accurately.

4.5.2 A Larger Example

The following system is copied from [40]:

$$F(\mathbf{x}) = \begin{cases} 2x_1 + 2x_1^2 + 2x_2 + 2x_2^2 + x_3^2 - 1 = 0; \\ (x_1 + x_2 - x_3 - 1)^3 - x_1^3 = 0; \\ (2x_1^3 + 2x_2^2 + 10x_3 + 5x_3^2 + 5)^3 - 1000x_1^5 = 0. \end{cases} \quad (4.28)$$

Counted with multiplicities, the system has 54 isolated solutions. We focus on the solution $(0, 0, -1)$ which occurs with multiplicity 18.

Although Algorithm 1 suggests that the first order deflation would already lower the corank of the system, we would like to search for a homogeneous deflation operator Q of order two.

To this end we construct the (truncated) deflation matrix $\bar{A}(x_1, x_2, x_3)$ which corresponds to $\{\partial_1^2, \partial_1\partial_2, \partial_1\partial_3, \partial_2^2, \partial_2\partial_3, \partial_3^2\}$, having 12 rows and only 6 columns.

The vectors $(1, 6, 8, -3, 0, 4)^T$ and $(0, 3, 3, -1, 1, 2)^T$ span the kernel of $\bar{A}(0, 0, -1)$. The operator corresponding to the former,

$$Q = \partial_1^2 + 6\partial_1\partial_2 + 8\partial_1\partial_3 - 3\partial_2^2 + 4\partial_3^2, \quad (4.29)$$

regularizes the system, since the equations

$$\begin{cases} Q \cdot (x_1 f_1) = 8x_1 + 24x_2 + 16x_3 + 16 = 0; \\ Q \cdot (x_2 f_1) = 24x_1 - 24x_2 = 0; \\ Q \cdot (x_3 f_1) = 32x_1 + 16x_3 + 16 = 0. \end{cases} \quad (4.30)$$

augmented to the original equations, give a system with the full-rank Jacobian matrix at $(0, 0, -1)$.

4.6 Conclusion

In this chapter we have described two methods of computing the multiplicity structure at isolated solutions of polynomial systems. We have developed a higher order deflation algorithm that reduces the multiplicity faster than the first order deflation in Chapter 2.

In our opinion, one of the main benefits of the higher order deflation for the numerical algebraic geometry algorithms is the possibility to regularize the system in a single step. For that one has to determine the minimal order of such a deflation or, even better, construct a

sparse ansatz for its deflation operator. Predicting these numerically could be a very challenging task, which should be explored in the future.

CHAPTER 5

APPLICATIONS AND NUMERICAL RESULTS

In this chapter, we show our implementation performs well on a large class of applications.

5.1 Applications

We encounter singular solutions when we solve polynomial systems using homotopy continuation methods, see e.g. [70]. This encounter can only happen – with probability one – at the end of the solution paths defined by the homotopy. So homotopy continuation methods deliver approximate solutions to the singularities which are then reconditioned by the deflation algorithm. One future project is the use of deflation to decide *locally* whether a solution at the end of a path is isolated or lies on a positive dimensional solution set. For this problem, deflation is needed because the solution set might be multiple.

The implementation has been tested on several examples, available at <http://www.math.uic.edu/~jan/demo.html>, almost all were obtained from surveying the literature. The initial approximations for Newton's method were taken from the end points of solution paths defined by a polynomial homotopy to find all isolated solutions (see [45; 46] for recent surveys). The numerical results reported in Table IV below are obtained with standard

machine arithmetic¹. We go back to our three examples in the first chapter for motivation and highlight these four examples from our benchmark collection.

A simple monomial ideal. Consider the simple polynomial system Equation 1.1. Viewing it as a monomial ideal, we immediately read off the multiplicity as 3. However, as explained in [69] and [70], this system presents a challenge to numerical solvers: making this overdetermined system square either by adding a random multiple of the last equation to the first two (and then removing the third equation), or by adding one slack variable which increases the multiplicity from three to four. As our deflation departs from Gauss-Newton, only one deflation step is needed to restore the quadratic convergence. Table IV opens with a summary of these calculations.

The 4-fold cyclic 9-roots. The so-called cyclic 9-roots problem (labelled as `cyclic9` in Table IV) is one of our largest examples. This system is a widely used benchmark in the field of polynomial system solving, e.g.: [3; 4], [16], [18], [29], [47], with theoretical results in [30]. In addition to six two dimensional cubics, there are 5,594 (333 orbits of size 18) isolated regular cyclic 9-roots, and most interestingly for this paper: 162 isolated solutions of multiplicity four. One deflation suffices to restore quadratic convergence on all 162 quadruple roots of this large application.

¹In case the multiple root is the origin, the number of correct digits in the last column of Table IV equals the negative exponent of ten in the magnitude of the root, i.e., “24” refers to 10^{-24} as the magnitude of the root. This explains why the accuracy is higher than what can be achieved from double precision floating-point arithmetic.

Lines tangent to a special configuration of 4 spheres. Given four spheres, how many real lines are tangent to all four spheres? A clue to the answer (see the Maple supplements to [71]) is obtained by placing the four spheres so they mutually touch each other. There are three distinct lines connecting the points where the spheres touch each other. Solving the corresponding algebraic system [15] shows the multiplicity of each line to be four, revealing twelve as the answer for this problem. One deflation suffices to compute all solutions accurately to the full machine precision, even with 16-digit approximations for the algebraic numbers $\sqrt{3}$ and $\sqrt{6}$ appearing as coefficients in the system. Computational results are in the second to last line of Table IV.

These three examples illustrate the motivation of our deflation algorithm: we present a method, designed to handle any kind of general isolated singular solutions of polynomial systems, efficient enough for large problems, and accepting approximate input coefficients.

5.2 Numerical Results

Observe the improved numerical conditioning in Table IV. This observation justifies calling¹ our method a “re-conditioning” method. In Table IV, the first column shows the systems we collected, for system details, see appendices. The dimension is listed under n , m is the multiplicity, and D is the number of deflations needed to restore quadratic convergence. The fifth column shows the decrease in the corank of the Jacobian matrix for all stages in the deflation. The second to last column contains the estimate for the inverse condition number of

¹We are grateful to Erich Kaltofen for this.

TABLE IV
 NUMERICAL RESULTS ON A COLLECTION OF TEST SYSTEMS

System	n	m	D	corank($A(\mathbf{x}^*)$)	Inverse Condition#	#Digits
simple [69]	2	3	1	2 \rightarrow 0	1.0e-08 \rightarrow 4.1e-01	8 \rightarrow 24
baker1 [33]	2	2	1	1 \rightarrow 0	1.7e-08 \rightarrow 3.8e-01	9 \rightarrow 24
cbms1 [74]	3	11	1	3 \rightarrow 0	4.2e-05 \rightarrow 5.0e-01	5 \rightarrow 20
cbms2 [74]	3	8	1	3 \rightarrow 0	1.2e-08 \rightarrow 5.0e-01	8 \rightarrow 18
mth191	3	4	1	2 \rightarrow 0	1.3e-08 \rightarrow 3.5e-02	7 \rightarrow 13
decker1 [10]	2	3	2	1 \rightarrow 1 \rightarrow 0	3.4e-10 \rightarrow 2.6e-02	6 \rightarrow 11
decker2 [8]	2	4	3	1 \rightarrow 1 \rightarrow 1 \rightarrow 0	4.5e-13 \rightarrow 6.9e-03	5 \rightarrow 16
decker3 [9]	2	2	1	1 \rightarrow 0	4.6e-08 \rightarrow 2.5e-02	8 \rightarrow 17
kss3 [34]	10	638	1	9 \rightarrow 0	4.4e-12 \rightarrow 1.4e-02	7 \rightarrow 16
ojika1 [60]	2	3	2	1 \rightarrow 1 \rightarrow 0	9.3e-12 \rightarrow 4.3e-02	5 \rightarrow 12
ojika2 [60]	3	2	1	1 \rightarrow 0	3.3e-08 \rightarrow 7.4e-02	6 \rightarrow 14
ojika3 [63]	3	2	1	1 \rightarrow 0	1.7e-08 \rightarrow 9.2e-03	7 \rightarrow 15
		4	1	2 \rightarrow 0	6.5e-08 \rightarrow 8.0e-02	6 \rightarrow 13
ojika4 [62]	3	3	2	1 \rightarrow 1 \rightarrow 0	1.9e-13 \rightarrow 2.4e-04	6 \rightarrow 11
caprasse [57]	4	4	1	2 \rightarrow 0	1.5e-09 \rightarrow 9.3e-03	8 \rightarrow 15
cyclic9 [3; 4]	9	4	1	2 \rightarrow 0	5.6e-10 \rightarrow 1.8e-03	5 \rightarrow 15
tangents [71]	6	4	1	2 \rightarrow 0	2.6e-08 \rightarrow 2.4e-02	7 \rightarrow 14

$A(\mathbf{x})$ at the start of the deflation to the end of the deflation for $\mathbf{x} \approx \mathbf{x}^*$. The last column lists the increase in the number of correct digits from the initial guess to the final approximation.

One of the interesting examples is taken from [63] and listed as “ojika3” in Table IV. This system has two isolated roots: one with multiplicity two, and the other one has multiplicity four. Both roots need only one deflation, but at the double root, the rank of the Jacobian matrix is two, while the rank is one at the other quadruple root. The program produces two

different deflated systems: one with three multipliers (for the double root) and the other with two multipliers (for the quadruple root).

The high multiplicity 638 of one root of the system `kss3` in Table IV looks spectacular, but since the defining equations are nice quadrics, one simple deflation suffices to compute the multiple root accurately, starting from any approximate root in the cluster of 638 solutions.

5.3 Conclusions

Our modified deflation method works in general, is numerically stable, relatively simple to implement; and perhaps most importantly, a preliminary implementation on a wide class of examples performs quite well.

The doubling of the number of equations by the deflation has been addressed in [43] and [7] for the corank one case. Our method is numerically robust, depending primarily on a reliable determination of the numerical rank of a matrix, a well studied subject in numerical linear algebra (see e.g. [17] or [50]). Nevertheless, an algebra theoretic certificate [5] of the numerical rank might be desirable for a fully automatic computer implementation.

Moreover, we have developed a higher order deflation algorithm that reduces the multiplicity faster than the first-order deflation in Chapter 2.

In our opinion, one of the main benefits of the higher order deflation for the numerical algebraic geometry algorithms is the possibility to regularize the system in a single step. For that one has to determine the minimal order of such a deflation or, even better, construct a sparse ansatz for its deflation operator. Predicting these numerically could be a very challenging task, which should be explored in the future.

APPENDICES

Appendix A

INPUT SYSTEMS (PHC FORMAT)

simple:

x^2 ;

$x*y$;

y^2 .

baker1:

$x_1^2 - x_2$;

$x_1^2 + x_2$.

cbms1:

$x_1^3 - x_2 * x_3$;

$x_2^3 - x_1 * x_3$;

$x_3^3 - x_1 * x_2$.

cbms2:

$x_1^3 - 3 * x_1^2 * x_2 + 3 * x_1 * x_2^2 - x_2^3 - x_3^2$;

$x_3^3 - 3 * x_3^2 * x_1 + 3 * x_3 * x_1^2 - x_1^3 - x_2^2$;

$x_2^3 - 3 * x_2^2 * x_3 + 3 * x_2 * x_3^2 - x_3^3 - x_1^2$.

Appendix A (Continued)

mth191:

$$x^3+y^2+z^2-1;$$

$$x^2+y^3+z^2-1;$$

$$x^2+y^2+z^3-1.$$

decker1:

$$x1^3+x1*x2;$$

$$x2^2+x2.$$

decker2:

$$x1+x2^3;$$

$$x1^2*x2-x2^4.$$

decker3:

$$x1+x2^2;$$

$$1.5*x1*x2+x2^2+x2^3.$$

kss3:

$$x1^2+x1+x2+x3+x4+x5+x6+x7+x8+x9+x10-2*x1-9;$$

$$x2^2+x1+x2+x3+x4+x5+x6+x7+x8+x9+x10-2*x2-9;$$

Appendix A (Continued)

$$x^3 + x^2 + x + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} - 2x^3 - 9;$$

$$x^4 + x^2 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} - 2x^4 - 9;$$

$$x^5 + x^2 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} - 2x^5 - 9;$$

$$x^6 + x^2 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} - 2x^6 - 9;$$

$$x^7 + x^2 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} - 2x^7 - 9;$$

$$x^8 + x^2 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} - 2x^8 - 9;$$

$$x^9 + x^2 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} - 2x^9 - 9;$$

$$x^{10} + x^2 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} - 2x^{10} - 9.$$

ojika1:

$$x^2 + y - 3;$$

$$x + 0.125y^2 - 1.5.$$

ojika2:

$$x^1 + x^2 + x^3 - 1;$$

$$x^1 + x^2 + x^3 - 1;$$

$$x^1 + x^2 + x^3 - 1.$$

ojika3;

$$x^1 + x^2 + x^3 - 1;$$

$$0.2x^1 + 0.5x^2 - x^3 + 0.5x^3 + 0.5;$$

Appendix A (Continued)

$$x_1 + x_2 + 0.5 * x_3^2 - 0.5.$$

ojika4:

$$x_1 + x_3 * x_1^3 + x_1 * x_3 * x_2^2 - x_1 * x_3;$$

$$10 * x_2 - 2 * x_2 * x_3 * x_1^2 - x_3 * x_2^3 - x_2 * x_3;$$

$$-6 * x_3^2 * x_1^4 - 3 * x_1^2 * x_2^2 * x_3^2 - x_3^2 * x_1^2 + 28 * x_3 * x_1^2$$

$$- 3 * x_3^2 * x_2^4 + 2 * x_3^2 * x_2^2 + 7 * x_3 * x_2^2 + x_3^2 - 11 * x_3 + 10.$$

caprasse:

$$y^2 * z + 2 * x * y * t - 2 * x - z;$$

$$-x^3 * z + 4 * x * y^2 * z + 4 * x^2 * y * t + 2 * y^3 * t + 4 * x^2 - 10 * y^2 + 4 * x * z - 10 * y * t + 2;$$

$$2 * y * z * t + x * t^2 - x - 2 * z;$$

$$-x * z^3 + 4 * y * z^2 * t + 4 * x * z * t^2 + 2 * y * t^3 + 4 * x * z + 4 * z^2 - 10 * y * t - 10 * t^2 + 2.$$

cyclic9:

$$z_0 + z_1 + z_2 + z_3 + z_4 + z_5 + z_6 + z_7 + z_8;$$

$$z_0 * z_1 + z_0 * z_8 + z_1 * z_2 + z_2 * z_3 + z_3 * z_4 + z_4 * z_5 + z_5 * z_6 + z_6 * z_7 + z_7 * z_8;$$

$$z_0 * z_1 * z_2 + z_0 * z_1 * z_8 + z_0 * z_7 * z_8 + z_1 * z_2 * z_3 + z_2 * z_3 * z_4 + z_3 * z_4 * z_5 + z_4 * z_5 * z_6 + z_5 * z_6 * z_7 + z_6 * z_7 * z_8;$$

$$z_0 * z_1 * z_2 * z_3 + z_0 * z_1 * z_2 * z_8 + z_0 * z_1 * z_7 * z_8 + z_0 * z_6 * z_7 * z_8 + z_1 * z_2 * z_3 * z_4 + z_2 * z_3 * z_4 * z_5 + z_3 * z_4 * z_5 * z_6 + z_4 * z_5 * z_6 * z_7 + z_5 * z_6 * z_7 * z_8;$$

Appendix A (Continued)

$$z_0*z_1*z_2*z_3*z_4+z_0*z_1*z_2*z_3*z_8+z_0*z_1*z_2*z_7*z_8+z_0*z_1*z_6*z_7*z_8+z_0*z_5*z_6*z_7*z_8$$

$$+z_1*z_2*z_3*z_4*z_5+z_2*z_3*z_4*z_5*z_6+z_3*z_4*z_5*z_6*z_7+z_4*z_5*z_6*z_7*z_8;$$

$$z_0*z_1*z_2*z_3*z_4*z_5+z_0*z_1*z_2*z_3*z_4*z_8+z_0*z_1*z_2*z_3*z_7*z_8+z_0*z_1*z_2*z_6*z_7*z_8+z_0$$

$$*z_1*z_5*z_6*z_7*z_8+z_0*z_4*z_5*z_6*z_7*z_8+z_1*z_2*z_3*z_4*z_5*z_6+z_2*z_3*z_4*z_5*z_6*z_7+z_3*z_4$$

$$*z_5*z_6*z_7*z_8;$$

$$z_0*z_1*z_2*z_3*z_4*z_5*z_6+z_0*z_1*z_2*z_3*z_4*z_5*z_8+z_0*z_1*z_2*z_3*z_4*z_7*z_8+z_0*z_1*z_2*z_3$$

$$*z_6*z_7*z_8+z_0*z_1*z_2*z_5*z_6*z_7*z_8+z_0*z_1*z_4*z_5*z_6*z_7*z_8+z_0*z_3*z_4*z_5*z_6*z_7*z_8+z_1$$

$$*z_2*z_3*z_4*z_5*z_6*z_7+z_2*z_3*z_4*z_5*z_6*z_7*z_8;$$

$$z_0*z_1*z_2*z_3*z_4*z_5*z_6*z_7+z_0*z_1*z_2*z_3*z_4*z_5*z_6*z_8+z_0*z_1*z_2*z_3*z_4*z_5*z_7*z_8+z_0$$

$$*z_1*z_2*z_3*z_4*z_6*z_7*z_8+z_0*z_1*z_2*z_3*z_5*z_6*z_7*z_8+z_0*z_1*z_2*z_4*z_5*z_6*z_7*z_8+z_0*z_1$$

$$*z_3*z_4*z_5*z_6*z_7*z_8+z_0*z_2*z_3*z_4*z_5*z_6*z_7*z_8+z_1*z_2*z_3*z_4*z_5*z_6*z_7*z_8;$$

$$z_0*z_1*z_2*z_3*z_4*z_5*z_6*z_7*z_8-1.$$

tangents:

$$x_0^2+x_1^2+x_2^2-1;$$

$$x_0*x_3+x_1*x_4+x_2*x_5;$$

$$x_3^2+x_4^2+x_5^2-0.25;$$

$$x_3^2+x_4^2-2*x_2*x_4+x_2^2+x_5^2+2*x_1*x_5+x_1^2-0.25;$$

$$x_3^2+1.73205080756888*x_2*x_3+0.75*x_2^2+x_4^2-x_2*x_4+0.25*x_2^2+x_5^2$$

$$-1.73205080756888*x_0*x_5+x_1*x_5+0.75*x_0^2-0.86602540378444*x_0*x_1$$

$$+0.25*x_1^2-0.25;$$

Appendix A (Continued)

$$\begin{aligned} & x^3 - 1.63299316185545x^2 + 0.57735026918963x - 0.6666666666667 \\ & - 0.47140452079103x^2 + 0.0833333333333x^2 + x^4 + 1.63299316185545x^3 \\ & - x^2 + 0.6666666666667x^2 - 0.81649658092773x^2 + 0.25x^2 + x^5 \\ & - 0.57735026918963x^5 + x^5 + 0.0833333333333x^2 - 0.28867513459481x^4 \\ & + 0.25x^2 - 0.25 \end{aligned}$$

CITED LITERATURE

1. Allgower, E. L. and Georg, K.: Introduction to Numerical Continuation Methods, volume 45 of Classics in Applied Mathematics. SIAM, 2003.
2. Bates, D. J., Peterson, C., and Sommese, A. J.: A numerical-symbolic algorithm for computing the multiplicity of a component of an algebraic set. J. Complexity, 22(4):475–489, 2006.
3. Björck, G. and Fröberg, R.: A faster way to count the solutions of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots. J. Symbolic Computation, 12(3):329–336, 1991.
4. Björck, G. and Fröberg, R.: Methods to “divide out” certain solutions from systems of algebraic equations, applied to find all cyclic 8-roots. In Analysis, Algebra and Computers in Math. research, eds, M. Gyllenberg and L. Persson, volume 564 of Lecture Notes in Mathematics, pages 57–70. Dekker, 1994.
5. Blum, L., Cucker, F., Shub, M., and Smale, S.: Complexity and Real Computation. Springer-Verlag, 1998.
6. Cox, D., Little, J., and O’Shea, D.: Using Algebraic Geometry, volume 185 of Graduate Texts in Mathematics. Springer-Verlag, 1998.
7. Dayton, B. H. and Zeng, Z.: Computing the multiplicity structure in solving polynomial systems. In Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation. ACM, 2005., ed. M. Kauers, pages 116–123, 2005.
8. Decker, D. W., Keller, H. B., and Kelley, C. T.: Convergence rates for Newton’s method at singular points. SIAM J. Numer. Anal., 20(2):296–314, 1983.
9. Decker, D. W., Keller, H. B., and Kelley, C. T.: Broyden’s method for a class of problems having singular Jacobian at the root. SIAM J. Numer. Anal., 22:566–574, 1985.
10. Decker, D. W. and Kelley, C. T.: Newton’s method at singular points: II. SIAM J. Numer. Anal., 17(3):465–471, 1980.

11. Dedieu, J. P. and Shub, M.: Newton's method for overdetermined systems of equations. Math. Comp., 69(231):1099–1115, 1999.
12. Demmel, J. W.: Applied Numerical Linear Algebra, volume 45 of Classics in Applied Mathematics. SIAM, 2003.
13. Deufflard, P., Friedler, B., and Kunkel, P.: Efficient numerical path following beyond critical points. SIAM J. Numer. Anal., 24:912–927, 1987.
14. Deufflard, P.: Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms. Springer-Verlag, 2004.
15. Durand, C.: Symbolic and Numerical Techniques for Constraints Solving. Doctoral dissertation, Purdue University, Department of Computer Science, 1998.
16. Faugère, J. C.: A new efficient algorithm for computing gröbner bases (f_4). Journal of Pure and Applied Algebra, 139(1-3):61–88, 1999. Proceedings of MEGA'98, 22–27 June 1998, Saint-Malo, France.
17. Fierro, R. D. and Hansen, P. C.: Utv expansion pack: Special-purpose rank-revealing algorithms. Numerical Algorithms, 40(1):47–66, 2005.
18. Gao, T. and Li, T.-Y.: Mixed volume computation for semi-mixed systems. Discrete Comput. Geom., 29(2):257–277, 2003.
19. Gerald, C. F. and Wheatley, P. O.: Applied Numerical Analysis. Addison-Wesley, 2003.
20. Giusti, M., Lecerf, G., Salvy, B., and Yakoubsohn, J. C.: On location and approximation of clusters of zeroes of analytic functions. Found. Comput. Math., 5(3):257–311, 2005.
21. Giusti, M., Lecerf, G., Salvy, B., and Yakoubsohn, J. C.: On location and approximation of clusters of zeroes: case of embedding dimension one. Found. Comput. Math., 7(1):1–58, 2007.
22. Govaerts, W. J. F.: Numerical Methods for Bifurcations of Dynamical Equilibria. SIAM, 2000.
23. Grayson, D. and Stillman, M.: Macaulay 2, a software system for research in algebraic geometry. Available at <http://www.math.uiuc.edu/Macaulay2/>.

24. Greuel, G. M. and Pfister, G.: Advances and improvements in the theory of standard bases and syzygies. Arch. Math., 66:163–196, 1996.
25. Greuel, G.-M. and Pfister, G.: A Singular Introduction to Commutative Algebra. Springer-Verlag, 2002.
26. Greuel, G.-M., Pfister, G., and Schönemann, H.: SINGULAR 2.0. A Computer Algebra System for Polynomial Computations, Centre for Computer Algebra, University of Kaiserslautern, 2001. <http://www.singular.uni-kl.de>.
27. Griewank, A.: On solving nonlinear equations with simple singularities or nearly singular solutions. SIAM Review, 27(4):537–563, 1985.
28. Griewank, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. SIAM, 2000.
29. Gunji, T., Kim, S., Kojima, M., Takeda, A., Fujisawa, K., and Mizutani, T.: PHoM – a polyhedral homotopy continuation method for polynomial systems. Computing, 73(1):57–77, 2004.
30. Haagerup, U.: Orthogonal maximal abelian *-algebras of the $n \times n$ matrices and cyclic n -roots. In Operator Algebras and Quantum Field Theory, pages 296–322. International Press, 1996.
31. Hazaveh, K., Jeffrey, D. J., Reid, G. J., Watt, S. M., and Wittkopf, A. D.: An exploration of homotopy solving in maple. In Lect. Notes Series on Computing by World Sci., eds, Z. Li and W. Sit., volume 10, pages 145–162, 2003. the Sixth Asian Symp. on Computer Math. (ASCM).
32. Hoffmann, C. M. and Yuan, B.: On spatial constraint solving approaches. In Lecture Notes In Computer Science, volume 2061, pages 1–15. Springer-Verlag, 2000. the Third International Workshop on Automated Deduction in Geometry.
33. Kearfott, R. B. and Dian, J.: Existence verification for higher degree singular zeros of nonlinear systems. SIAM J. Numer. Anal., 41(6):2350–2373, 2003.
34. Kobayashi, H., Suzuki, H., and Sakai, Y.: Numerical calculation of the multiplicity of a solution to algebraic equations. Math. Comp., 67(221):257–270, 1998.
35. Kress, R.: Numerical Analysis, volume 181 of Graduate Texts in Mathematics. Springer.

36. Kunkel, P.: Efficient computation of singular points. IMA J. Numer. Anal., 9:421–433, 1989.
37. Kunkel, P.: A tree-based analysis of a family of augmented systems for the computation of singular points. IMA J. Numer. Anal., 16:501–527, 1996.
38. Kuznetsov, Y. A.: Elements of Applied Bifurcation Theory, volume 112 of Applied Mathematical Sciences. Springer–Verlag, third edition, 2004.
39. Labs, O.: Hypersurfaces with Many Singularities - History, Constructions, Algorithms, Visualization. Doctoral dissertation, Johannes Gutenberg University of Mainz, Department of Physics, Mathematics and Computer Science, 2005.
40. Lecerf, G.: Quadratic Newton iteration for systems with multiplicity. Found. Comput. Math., 2:247–293, 2002.
41. Leykin, A. and Verschelde, J.: A Maple interface to the numerical homotopy algorithms in PHCpack. In Proceedings of the Tenth International Conference on Applications of Computer Algebra (ACA'2004), ed. Q.-N. Tran, pages 139–147, 2004.
42. Leykin, A., Verschelde, J., and Zhao, A.: Newton's method with deflation for isolated singularities of polynomial systems. Theoretical Computer Science, 359(1-3):111–122, 2006.
43. Leykin, A., Verschelde, J., and Zhao, A.: Evaluation of Jacobian matrices for Newton's method with deflation to approximate isolated singular solutions of polynomial systems. In Symbolic-Numeric Computation, eds, D. Wang and L. Zhi, pages 269–278. Birkhauser, 2007.
44. Leykin, A., Verschelde, J., and Zhao, A.: Higher-order deflation for polynomial systems with isolated singular solutions. Appear in the IMA Volume on Algorithms in Algebraic Geometry, 2007.
45. Li, T.-Y.: Numerical solution of multivariate polynomial systems by homotopy continuation methods. Acta Numerica, 6:399–436, 1997.
46. Li, T.-Y.: Numerical solution of polynomial systems by homotopy continuation methods. In Handbook of Numerical Analysis. Volume XI. Special Volume: Foundations of Computational Mathematics, ed. F. Cucker, pages 209–304. North-Holland, 2003.

47. Li, T.-Y. and Li, X.: Finding mixed cells in the mixed volume computation. Found. Comput. Math., 1(2):161–181, 2001.
48. Li, T.-Y. and Wang, X.: Solving real polynomial systems with real homotopies. Math. Comp., 60:669–680, 1993.
49. Li, T.-Y. and Wang, X.: Higher order turning points. Appl. Math. Comput., 64:155–166, 1994.
50. Li, T.-Y. and Zeng, Z.: A rank-revealing method with updating, downdating and applications. SIAM J. Matrix Anal. Appl., 26:918–946, 2005.
51. Macaulay, F. S.: The Algebraic Theory of Modular Systems. Cambridge University Press, 1916. Reissued with an Introduction by Paul Roberts in the Cambridge Mathematical Library 1994.
52. Macdonald, I. G., Pach, J., and Theobald, T.: Common tangents to four unit balls in \mathbb{R}^3 . Discrete and Comp. Geometry, 26(1):1–17, 2001.
53. Marinari, M. G., Moeller, H. M., and Mora, T.: On multiplicities in polynomial system solving. Trans. AMS, 348(8):3283–3321, 1996.
54. Möller, H. M. and Stetter, H. J.: Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. Numer. Math., 70:311–329, 1995.
55. Mora, F.: An algorithm to compute the equations of tangent cones. In Computer Algebra. EUROCAM’82, European Computer Algebra Conference. Marseille, France, April 1982., ed. J. Calmet, volume 144 of Lecture Notes in Computer Science, pages 158–165. Springer-Verlag, 1982.
56. Mora, T.: Gröbner duality and multiple points in linearly general position. Proceedings of the AMS, 125(5):1273–1282, 1997.
57. Moritsugu, S. and Kuriyama, K.: On multiple zeros of systems of algebraic equations. In Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation (ISSAC 1999), ed. S. Dooley, pages 23–30. ACM, 1999.
58. Mourrain, B.: Isolated points, duality and residues. Journal of Pure and Applied Algebra, 117/118:469–493, 1997.

59. Mourrain, B.: Isolated points, duality and residues. J. Pure Appl. Algebra, 117/118:469–493, 1997. Algorithms for algebra (Eindhoven, 1996).
60. Ojika, T.: Modified deflation algorithm for the solution of singular problems. I. A system of nonlinear algebraic equations. J. Math. Anal. Appl., 123:199–221, 1987.
61. Ojika, T.: Modified deflation algorithm for the solution of singular problems. II. Nonlinear multipoint boundary value problems. J. Math. Anal. Appl., 123:222–237, 1987.
62. Ojika, T.: A numerical method for branch points of a system of nonlinear algebraic equations. Applied Numerical Mathematics, 4:419–430, 1988.
63. Ojika, T., Watanabe, S., and Mitsui, T.: Deflation algorithm for the multiple roots of a system of nonlinear equations. J. Math. Anal. Appl., 96:463–479, 1983.
64. Ortega, J. M. and Rheinboldt, W. C.: Iterative Solution of Nonlinear Equations in Several Variables, volume 30 of Classics in Applied Mathematics. SIAM, 2000.
65. Rall, L. B.: Automatic Differentiation: Techniques and Applications, volume 120 of Lecture Notes in Computer Science. Springer, 1981.
66. Sommese, A. J. and Verschelde, J.: Numerical homotopies to compute generic points on positive dimensional algebraic sets. J. of Complexity, 16(3):572–602, 2000.
67. Sommese, A. J., Verschelde, J., and Wampler, C. W.: A method for tracking singular paths with application to the numerical irreducible decomposition. pages 329–345, 2002. In *Algebraic Geometry, a Volume in Memory of Paolo Francia*, edited by M.C. Beltrametti, F. Catanese, C. Ciliberto, A. Lanteri, C. Pedrini. W. de Gruyter.
68. Sommese, A. J. and Wampler, C. W.: Numerical algebraic geometry. 32:749–763, 1996. In *The Mathematics of Numerical Analysis: Real Number Algorithm, Park City, Utah, Summer 1995*, ed. by J. Renegar, M. Shub, and S. Smale.
69. Sommese, A. J. and Wampler, C. W.: Numerical algebraic geometry. In The Mathematics of Numerical Analysis, eds, J. Renegar, M. Shub, and S. Smale, volume 32 of Lectures in Applied Mathematics, pages 749–763. AMS, 1996. Proceedings of the AMS-SIAM Summer Seminar in Applied Mathematics. Park City, Utah, July 17–August 11, 1995, Park City, Utah.

70. Sommese, A. J. and Wampler, C. W.: The Numerical Solution of Systems of Polynomials arising in Engineering and Science. World Scientific Press, 2005.
71. Sottile, F. and Theobald, T.: Lines tangents to $2n - 2$ spheres in \mathbb{R}^n . Trans. Amer. Math. Soc., 354:4815–4829, 2002. <http://www.math.tamu.edu/~sottile/pages/spheres/index.html> offers a nice computer algebra supplement.
72. Stetter, H. J.: Numerical Polynomial Algebra. SIAM, 2004.
73. Stetter, H. J. and T., T. G.: Singular systems of polynomials. In Proceedings of ISSAC 1998, ed. O. Gloor, pages 9–16, 1998.
74. Sturmfels, B.: Solving Systems of Polynomial Equations. Number 97 in CBMS Regional Conference Series in Mathematics. AMS, 2002.
75. Thallinger, G. T.: Zero behavior in perturbed systems of polynomial equations. 1998. PhD Thesis, Tech. Univ. Vienna.
76. Verschelde, J.: Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. ACM Trans. Math. Softw., 25(2):251–276, 1999. Software available at <http://www.math.uic.edu/~jan>.
77. Verschelde, J.: Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. ACM Trans. Math. Softw., 25(2):251–276, 1999. Software available at <http://www.math.uic.edu/~jan>.
78. Verschelde, J. and Zhao, A.: Newton’s method with deflation for isolated singularities. Poster presented at ISSAC’04, 6 July 2004, Santander, Spain. Available at <http://www.math.uic.edu/~jan/poster.pdf> and at <http://www.math.uic.edu/~azhao1/poster.pdf>.
79. Ypma, T. Y.: Historical development of the Newton-Rapson method. SIAM Review, 37(4):531–551, 1995.
80. Zeng, Z.: Computing multiple roots of inexact polynomials. Mathematics of Computation, 74:869–903, 2005.
81. Zhen, M.: Numerical Bifurcation Analysis For Reaction-Diffusion Equations. Springer-Verlag, 2000.

VITA

NAME: Ailing Zhao

EDUCATION: B.E., Machine Manufacture and Automation, Taiyuan University
of Science and Technology, China, 1993

M.S., Mathematical Computer Science, University of Illinois at
Chicago, Chicago, Illinois, 2001

Ph.D., Mathematical Computer Science, University of Illinois at
Chicago, Chicago, Illinois, 2007

EXPERIENCE: Teaching Assistant, Department of Mathematics, Statistics, and
Computer Science, University of Illinois at Chicago, 2001,
2005-2007

Research Assistant, Department of Mathematics, Statistics, and
Computer Science, University of Illinois at Chicago, 2002-2005

PUBLICATIONS: Newton's Method with Deflation for Isolated Singularities of

Polynomial Systems (with Anton Leykin and Jan Verschelde).

Theoretical Computer Science 359(1-3): 111-122, 2006

Evaluation of Jacobian Matrices for Newton's Method with
Deflation to approximate Isolated Singular Solutions of
Polynomial Systems (with Anton Leykin and Jan Verschelde).

In Symbolic-Numeric Computation, edited by Dongming Wang and
Lihong Zhi. Pages 269-278. Trends in Mathematics. Birkhauser,
2007

Higher-Order Deflation for Polynomial Systems with Isolated
Singular Solutions (with Anton Leykin and Jan Verschelde).

Appear in the IMA Volume on Algorithms in Algebraic Geometry,
2007

PRESENTATIONS: Newton's Method with Deflation for Isolated Singularities of
Polynomial Systems, AMS Special Session on Solving Polynomial
Systems, October 23-24, 2004, Northwestern University,
Evanston, IL

Newton's Method with Deflation for Isolated Singularities of

Polynomial Systems, LNF'04, December 1-3, 2004, Toulouse,
France

Evaluation of Jacobian Matrices for Newton's Method with
Deflation, SNC'05, July 19-21, 2005, Xian, China

Application of Deflation Method, AMS Special Session on
Numerical Solution of Polynomial Systems, April 8-9, 2006,
University of Notre Dame, IN

Newton's Method with Deflation for Isolated Singularities of
Polynomial Systems, AMS National Meetings, January 5-8, 2007,
New Orleans, LA.

POSTERS: Newton's Method with Deflation for Isolated Singularities
(with Jan Verschelde). ISSAC 2004, July 4-7, University of
Cantabria, Santander, Spain

PROFESSIONAL American Mathematical Society

MEMBERSHIP: Society for Industrial and Applied Mathematics

AWARDS: Summer 2005 and Winter 2006 travel grants awarded by WISE
(Women in Science and Engineering) and Department of MSCS

Index

artificial-parameter homotopy, 8

closeness condition, 53
continuation parameter, 10
cyclic 9-roots, 2, 72

deflation matrix, 44
deflation operator, 46
differential functionals, 50
dual space, 51

homotopy continuation, 8, 71

initial ideal, 19
initial support, 52
initial term, 52
isolated solution, 10

local ordering, 18

multiplicity, 10, 16, 21, 59, 61
multiplicity structure, 43, 48

Newton's Method, 5
numeric deflation, 24
numerical rank, 29

order of deflations, 44

reconditioning, 11, 73

singular solution, 10
staircase, 19, 49
standard basis, 19, 49
standard monomial, 19, 49
start system, 8
support, 51
symbolic deflation, 21

target system, 8
truncated deflation matrix, 65

weight vector, 18