Restoring singularity to polynomial roots

A Project
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Masters of Science
Mathematical Sciences

by
Andrew Pitman
July 2020

Acknowledgments

My sincerest thanks and appreciation is for my advisor, Dr. Michael Burr. He's repeatedly shown me more patience and understanding than I deserved. Without him, this project and many other things would be impossible.

I'd like to thank my committee for their flexibility, time, and help in completing this project. Likewise, I wish to acknowledge the administration of the School of Mathematical and Statistical Sciences, for their help in meeting deadlines and getting this project submitted.

I acknowledge the love and support of those close to me. I thank my parents and sister, who have responded to missed phone calls and unanswered messages with words of comfort and offers to help. I also thank my close friends, who have helped me push through at difficult moments while patiently rescheduling many games of D&D.

Clemson University is acknowledged for generous allotment of computational resources on the Palmetto cluster, without which I could not have made the examples in this project.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This project presents an algorithm to recover the multiplicity structure of the roots of a polynomial with perturbed coefficients. We discuss the problem, its applications, and look at an example.

1.1 Root combination

In this section we discuss the problem of combining roots of a polynomial with approximate coefficients.

Definition 1.1. For any polynomial f, the *variety* of f, denoted by $\mathcal{V}(f)$, is the set of solutions to the equation $f(x) = 0$. When $\mathcal{V}(f)$ is zero-dimensional, there are only finitely many solutions, which we call the *roots* of f.

Definition 1.2. Let ρ be the set of distinct roots of the polynomial f. The *multiplicity* of the root r^* of f is denoted $e(r^*, f)$. Multiplicity is implicitly defined by the equation

$$f(x) = p \prod_{r \in \rho} (x - r)^{e(r,f)}$$

where p is the leading coefficient of f. We say that r^* is a singular root of f if $e(r^*, f) > 1$.

Definition 1.3. Let ρ and σ be the sets of distinct roots of the polynomials f and g, respectively, where f and g have the same degree. Suppose there exists a surjective function $\psi : \rho \to \sigma$ such that, for all $s \in \sigma$

$$e(s,g) = \sum_{r \in \psi^{-1}(s)} e(r,f).$$

f has the *same multiplicity structure* as g if ψ is a bijection, but f has *less multiplicity structure* than g otherwise.

Example 1.4. In essence, a polynomial f has less multiplicity structure than g if the multiplicities of the roots of f can be added together to get the multiplicities of the roots of g. Define the fifth-degree polynomial f by

$$f(x) = (x-11)^3(x-12)(x-13)$$

which has the roots $\mathcal{V}(f) = \{11, 12, 13\}$ with multiplicities $e(11,f) = 3$, $e(12,f) = 1$, and $e(13,f) = 1$. Likewise, define the fifth-degree polynomial g by

$$g(x) = (x-24)^3(x-25)^2$$

which has the roots $\mathcal{V}(g) = \{24, 25\}$ with multiplicities $e(24,g) = 3$ and $e(25,g) = 2$. Then f has less multiplicity structure than g since

$$e(24,g) = e(11,f) = 3 \qquad \text{and} \qquad e(25,g) = e(12,f) + e(13,f) = 2.$$

To put these equations in the format of Definition 1.3, we consider the function $\psi : \mathcal{V}(f) \to \mathcal{V}(g)$ given by $\psi(11) = 24$, $\psi(12) = 25$, and $\psi(13) = 25$. Note that $\psi^{-1}(24) = \{11\}$ and $\psi^{-1}(25) = \{12, 13\}$. Thus, ψ is surjective, but not bijective.

Fix a polynomial $f \in \mathbb{R}[x]$, where we consider $\mathbb{R}[x]$ as a subspace of $\mathbb{C}[x]$. Let $\tilde{f} \in \mathbb{R}[x]$ be a polynomial whose coefficients approximate those of f. Note that the roots of a polynomial are sensitive to changes in its coefficients [9, 10]. In fact, $\tilde{f}$ has only nonsingular roots, generically, even when f has singular roots. Conceptually, a singular root r^* of f is replaced under perturbation by $e(r^*, f)$ nonsingular roots of $\tilde{f}$. Our algorithm corrects the multiplicity structure of $\tilde{f}$, producing a polynomial with the multiplicity structure of f. This is done by computing a sequence of polynomials with strictly increasing multiplicity structure while minimizing the change in coefficients between successive polynomials. We see that each polynomial in the sequence combines roots of the previous polynomial.

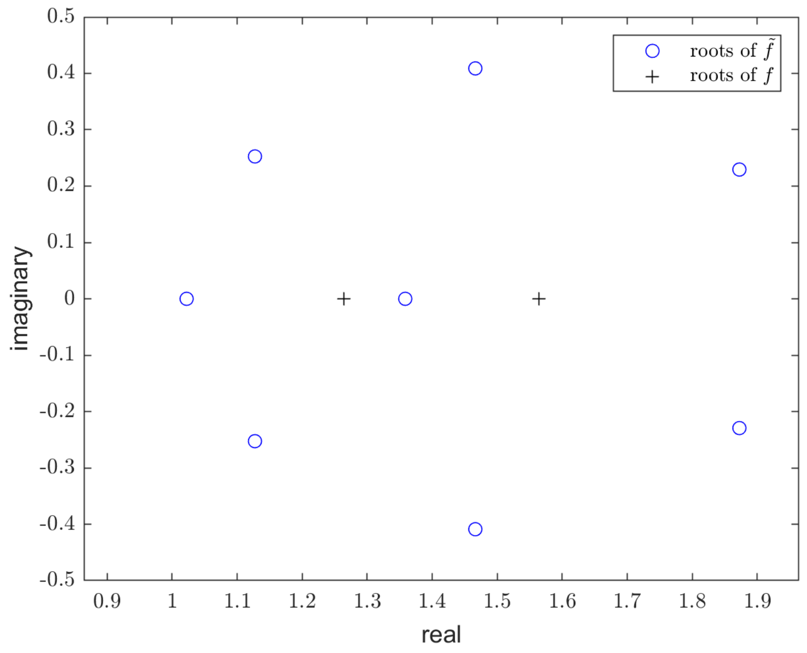


Figure 1.1: The roots of $\tilde{f}$ surround the roots of f. It is unclear which roots of $\tilde{f}$ are associated to which roots of f. This graphic was generated using MATLAB [11].

Example 1.5. Consider the polynomial f with roots $\sqrt{2}+0.15$ and $\sqrt{2}-0.15$ each having multiplicity 4, i.e.,

$$f(x) = (x - \sqrt{2} - 0.15)^4 (x - \sqrt{2} + 0.15)^4.$$

Suppose $\tilde{f}$ is formed by approximating the coefficients of f to 6 significant digits.

$$\tilde{f} = x^8 - 11.3137x^7 + 55.9100x^6 - 157.628x^5 + 277.303x^4 \\ - 311.710x^3 + 218.636x^2 - 87.4892x + 15.2921$$

has roots distributed around $\sqrt{2}$, as shown in Figure 1.1. We see that $\tilde{f}$ has eight nonsingular roots, as opposed to the two singular roots of f. Therefore, $\tilde{f}$ has less multiplicity structure than f. In order to recover the multiplicity structure of f, we compute a sequence $\{f_k\}_k$ of polynomials such that f_{k+1} combines roots of f_k (see Definition 3.6). The sequence starts with $f_0 = \tilde{f}$ and ends with a polynomial having the multiplicity structure of f. Using our software (see the appendices) we generate the sequence $\{\mathcal{V}(f_k)\}_k$ which we record in Table 1.1 and plot in Figure 1.2.

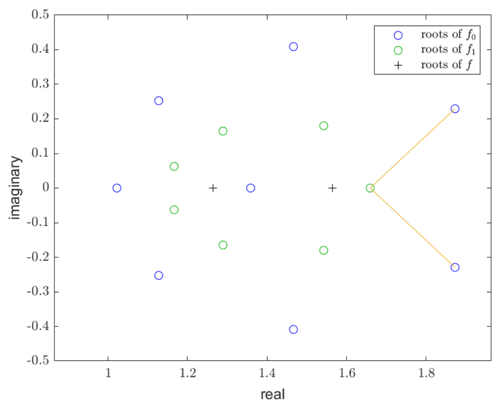
$\boldsymbol{k}$	**Root of $\boldsymbol{f_k}$**	**Multiplicity**
0	1.02224138610754	1
	$1.12736417060518 + .252661805432663i$	1
	$1.46639719146355 + .408887617179349i$	1
	$1.87267330544321 + .229382276735392i$	1
	1.35858927891579	1
	$1.87267330543744 - .229382276741512i$	1
	$1.46639719145683 - .408887617179505i$	1
	$1.12736417061561 - .252661805425774i$	1
1	$1.16626113013897 - .0626020487772903i$	1
	$1.28912209942458 + .164676054207027i$	1
	$1.54233910644068 + .18012166141248i$	1
	$1.16626113013897 + .0626020487772903i$	1
	1.65913699339357	2
	$1.28912209942458 - .164676054207027i$	1
	$1.54233910644068 - .18012166141248i$	1
2	1.16625630649333	2
	1.28912209919162	2
	1.54233910655757	2
	1.65913699333891	2
3	1.17741638353608	2
	1.26514183160039	2
	1.60708003687555	4
4	1.21749259737198	4
	1.61113497412845	4

Table 1.1: We restore the multiplicity structure of f to $\tilde{f}$ by computing the roots of a sequence $\{f_k\}_k$ of polynomials. Imaginary parts of order less than 10^{-10} have been omitted, since these are real to within our error tolerance. These numbers are generated by scripts written in Macaulay2 [6] with Bertini [8] using the package `RootReduce` (see appendices).

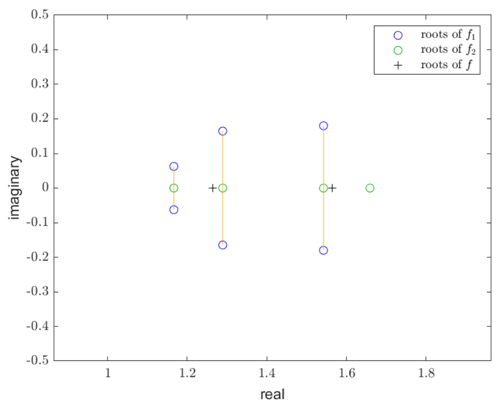
The fourth polynomial f_4 in our sequence has the same multiplicity structure as f, so we call the roots of f_4 the *approximate roots* of f.

$$\begin{aligned} f_4(x) =& (x - 1.21749259737198)^4 (x - 1.61113497412845)^4 \\ \approx& x^8 - 11.3145x^7 + 55.8530x^6 - 157.1107x^5 + 275.4391x^4 \\ & - 308.1796x^3 + 214.9032x^2 - 85.3946x + 14.8045. \end{aligned}$$

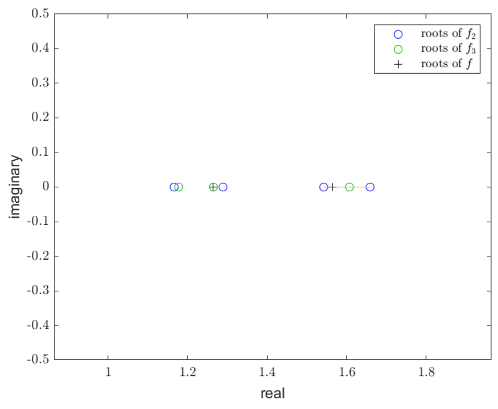
Note that the coefficients of f_4 differ from those of $\tilde{f}$ by a reasonable maximum relative error of approximately 0.031885. Additionally, the roots of f_4 and f are closer together than the roots of $\tilde{f}$ and f. When we combine the roots of $\tilde{f}$, we produce roots which better approximate the roots of f.



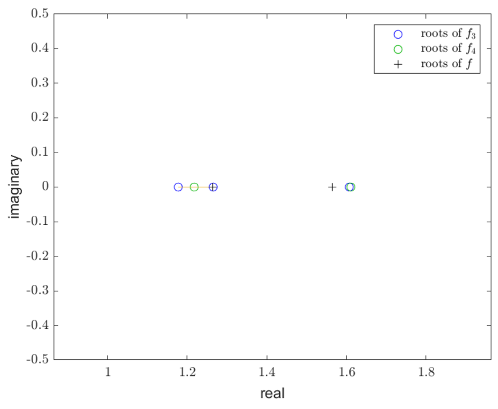
(a) $\mathcal{V}(f_0)$ to $\mathcal{V}(f_1)$



(b) $\mathcal{V}(f_1)$ to $\mathcal{V}(f_2)$



(c) $\mathcal{V}(f_2)$ to $\mathcal{V}(f_3)$



(d) $\mathcal{V}(f_3)$ to $\mathcal{V}(f_4)$

Figure 1.2: These plots depict the roots of $\tilde{f}$ iteratively combining. We use blue circles to represent the roots of f_k and green circles to represent the roots of f_{k+1}. A root of f_{k+1} whose multiplicity is a sum of the multiplicities of roots of f_k is connected to those roots by yellow line segments. Notice that, in Figure 1.2b, 3 pairs of roots combine at once. This graphic was generated using MATLAB [11].

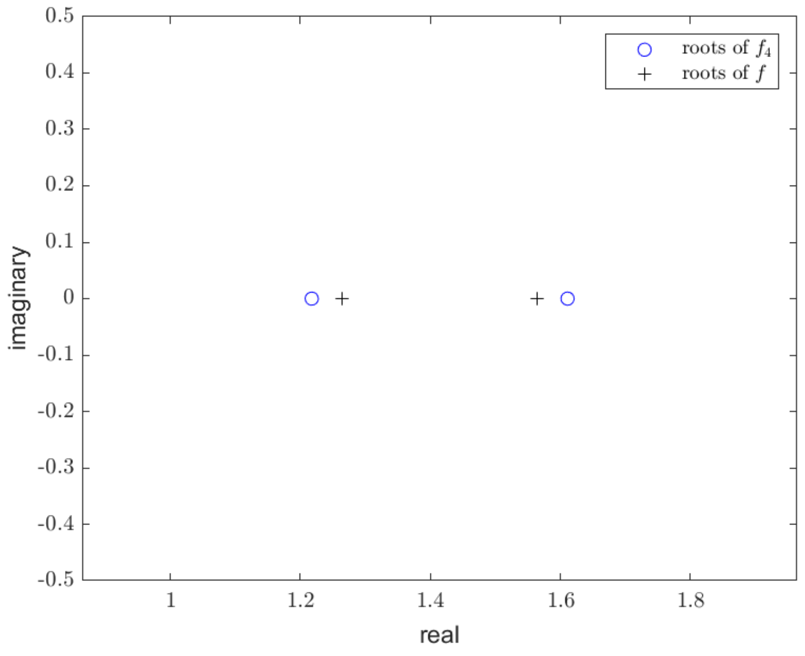


Figure 1.3: The approximate roots of f plotted with the actual roots of f. This graphic was generated using MATLAB [11].

1.2 Application: the approximate GCD

In this section, we describe the application of root combination to the approximate GCD problem. Finding the approximate GCD of two polynomials is an important problem in computational algebra with many applications [12]. We can use our root combination algorithm to work towards a solution of the approximate GCD problem for univariate polynomials.

Problem 1.6 (Approximate GCD problem)**.** Given two approximations $\tilde{f}$ and $\tilde{g}$ of two polynomials f and g, approximate the greatest common divisor of f and g.

Note that the GCD of two polynomials f and g can be constructed from their roots. To solve for the approximate GCD of two polynomials, we identify roots of $\tilde{f}$ with roots of $\tilde{g}$. In practice, roots are often identified with one another based on proximity. However, we recall that the roots of a polynomial are sensitive to changes in the coefficients. This sensitivity is particularly strong for singular roots [9]. Small errors in the approximate coefficients of the polynomials $\tilde{f}$ and $\tilde{g}$ may increase the distance between the roots that should be identified, reducing the accuracy of a GCD approximation. Additionally, $\tilde{f}$ and $\tilde{g}$ do not have the multiplicity structures of f and g generically, e.g., a root of $\tilde{f}$ may be identified with multiple roots of $\tilde{g}$.

Our algorithm restores the multiplicity structures of f and g, e.g., so that a root of $\tilde{f}$ may only be identified with one root of $\tilde{g}$.

Example 1.7. Recall from Example 1.5 that $f(x) = (x - \sqrt{2} - 0.15)^4(x - \sqrt{2} + 0.15)^4$ and consider the polynomial

$$g(x) = (x - \sqrt{2} - 0.15)^4.$$

Note that g divides f. We let $\tilde{g}$ be an approximation to the coefficients of g to 6 significant digits, the resulting polynomial

$$\tilde{g} = x^4 - 6.2569x^3 + 14.6806x^2 - 15.3090x + 5.9867$$

has roots distributed around $\sqrt{2} + 0.15$ as shown in Figure 1.4. The plot of the roots of $\tilde{f}$ and $\tilde{g}$, in Figure 1.4, shows that the roots of $\tilde{g}$ are distributed over a smaller region of the complex plane than the roots of $\tilde{f}$. As a result, it is not immediately obvious how the roots of $\tilde{g}$ should be identified with roots of $\tilde{f}$. We combine the roots of $\tilde{g}$ to restore the multiplicity structure of g. Let $\{g_k\}_k$ be

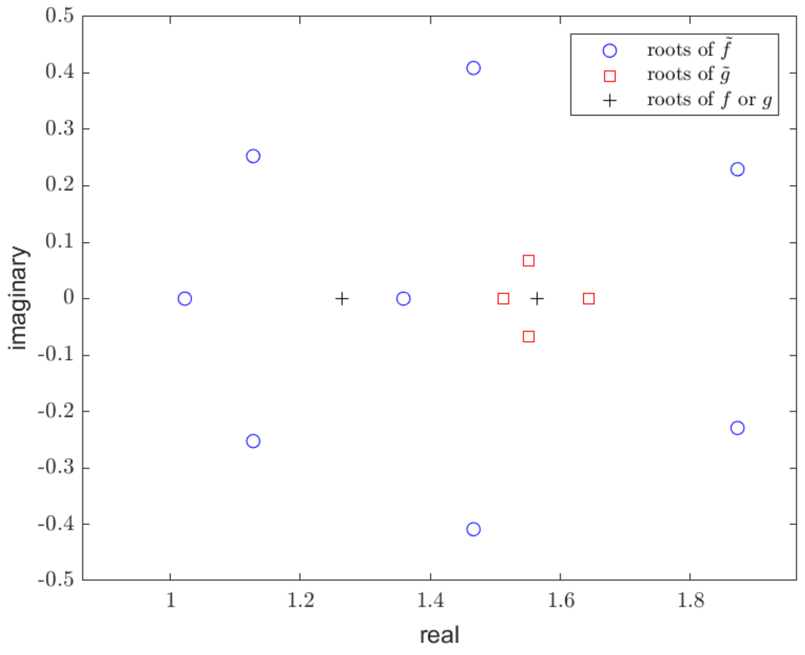


Figure 1.4: The roots of $\tilde{f}$ and $\tilde{g}$ surrounding the roots of f. The roots of $\tilde{f}$ are distributed over a larger area than the roots of $\tilde{g}$ and it is hard to tell which roots of $\tilde{f}$ correspond to which roots of $\tilde{g}$. It is easier to tell that all of the roots of $\tilde{g}$ correspond to the same root of f. This graphic was generated using MATLAB [11].

the sequence of polynomials such that g_{k+1} combines roots of g_k starting with $g_0 = \tilde{g}$ and ending with a polynomial that has only one root. The roots of $\{g_k\}_k$ are recorded in Table 1.2 and plotted in Figure 1.5.

As a result, we perturb the coefficients of $\tilde{g}$ to compute a polynomial g_2 with the multiplicity structure of g given by

$$g_2(x) = (x - 1.56421090658432)^4$$
$$\approx x^4 - 6.2568x^3 + 14.6805x^2 - 15.3090x + 5.9866.$$

We call the root 1.56421090658432 of g_2 the approximate root of g. Note that the coefficients of g_2 differ from those of $\tilde{g}$ by a low relative error of approximately 6.79×10^{-6} and the root of g is much closer to the root of g_2 than to the roots of $\tilde{g}$. Combining the roots of $\tilde{g}$ produces a root which better approximates the root of g.

k	Root of g_k	Multiplicity
0	1.51239824780835	1
	$1.55049553323509 + .0668992571665445i$	1
	1.64351068573018	1
	$1.55049553323386 - .0668992571760467i$	1
1	$1.56420013135082 - .00422998423754113i$	3
	$1.56415375214466 + .0126897469578536i$	1
2	1.56421090658432	4

Table 1.2: We restore the multiplicity structure of g to $\tilde{g}$ by computing the roots of a sequence $\{g_k\}_k$ of polynomials. Imaginary parts smaller than 10^{-10} have been omitted, since these are real to within our error tolerance. These numbers are generated from scripts written in Macaulay2 [6] with Bertini [8] using the method `rootReduce` (see appendices).

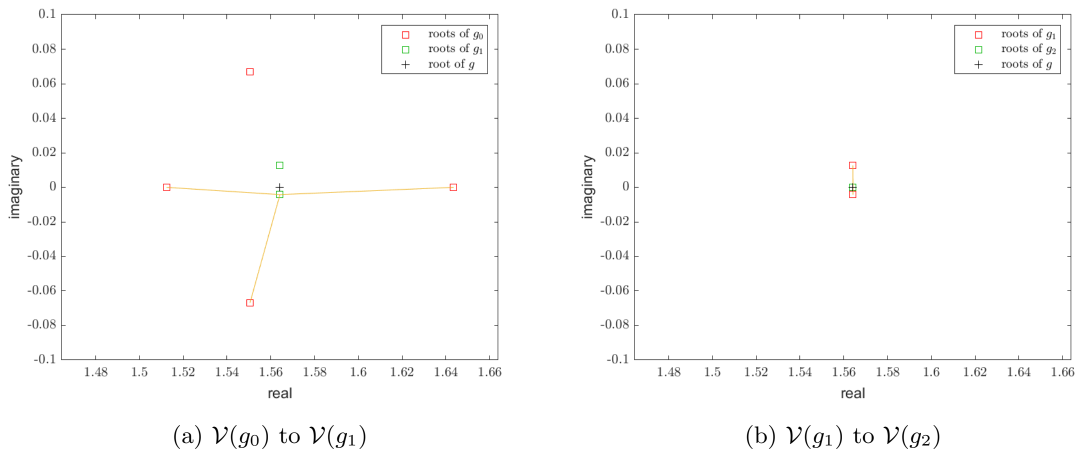


Figure 1.5: These plots show roots of $\tilde{g}$ iteratively combining. We use red squares to represent the roots of g_k and green squares to represent the roots of g_{k+1}. A root of g_{k+1} whose multiplicity is a sum of the multiplicities of roots of g_k is connected to those roots by yellow line segments. Note that, in the first step, three roots combine into a single root. This graphic was generated using MATLAB [11].

Recalling the results of Example 1.5, the approximate roots of f and g are

$$\mathcal{V}(f_4) = \{1.21749259737198, 1.61113497412845\}$$

$$\text{and} \quad \mathcal{V}(g_2) = \{1.564210906584 32\},$$

respectively. We plot the approximate roots of f and g together in Figure 1.6. We identify the approximate root 1.61113497412845 of f with the approximate root 1.56421090658432 of g, since they have the same multiplicity and are closer to each other than they are to the other approximate root 1.21749259737198 of f. It is fortunate that the identified roots have the same multiplicity,

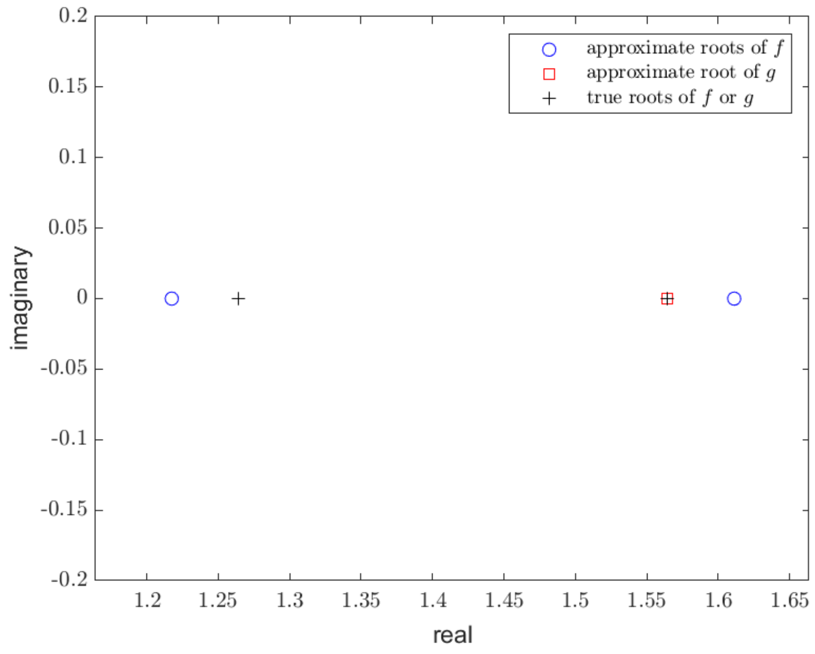


Figure 1.6: A plot of the approximate roots of f and g, computed by iteratively combining the roots of $\tilde{f}$ and $\tilde{g}$, respectively. We see that one of the approximate roots of f is much closer to the approximate root of g than the other. This graphic was generated using MATLAB [11].

as this is not guaranteed. We conclude that the approximate GCD of f and g is a fourth degree polynomial whose only root is near 1.56421090658432 and 1.61113497412845.

1.3 Notation

Throughout this project, we make frequent use of some notational conventions. The symbols used to denote the inputs of the homotopies described in this project are given in Table 1.3. We

	Roots	**Parameters**	**Langrange multipliers**	**Time**
Variable	r	p	ℓ	t
Particular value	ρ^*	π^*	λ^*	t^*
Trackable path	ρ	π	λ	
Starting point	ϱ	ϖ		
Length of vector	R	P		

Table 1.3: Notational conventions for homotopy inputs. By *variable*, we mean a symbol denoting an input to a function but not denoting a particular value of the input. Note that a particular value is not necessarily a point on a trackable path, but is a fixed value in the space of inputs.

denote systems of polynomials by uppercase letters, such as F and G. We typically denote individual polynomials by lowercase letters such as f and g. The symbol ∇ denotes both the gradient operator and the Jacobian operator. If ∇ carries a subscript, then it represents the gradient or Jacobian with respect to the subscripted variables. For example, $\nabla_r H$ represents the of the Jacobian of the homotopy H with respect the the variables $r_1, \ldots, r_R$. We use the Euclidean norm, denoted by $\|\cdot\|$, exclusively. Using $\mathbb{R}$ as a subscript denotes the intersection of the subscripted set with the real numbers; e.g., $\mathcal{V}_{\mathbb{R}}(F) = \mathcal{V}(F) \cap \mathbb{R}^R$.

Chapter 2

Background

We restore the root structure of an approximated polynomial using *homotopy continuation*. We first review homotopy continuation, and how it deforms the roots of a given polynomial. We follow the approach as presented in [7].

2.1 Homotopy continuation

In this section, we introduce homotopy continuation as a tool to compute the solutions to a system of polynomial equations. We seek to compute the roots of a system of polynomials $F : \mathbb{C}^R \to \mathbb{C}^R$. Directly calculating the roots of a polynomial system can be difficult [4]. Instead, we compute them indirectly by deforming the known roots of a different polynomial system. We start with a polynomial system G and its roots $\mathcal{V}(G)$. We deform $\mathcal{V}(G)$ to $\mathcal{V}(F)$ using *homotopy continuation*. Since we do not know $\mathcal{V}(F)$, we have to be clever in order to define such a smooth deformation. We start by focusing on the polynomial systems F and G directly, instead of focusing on their roots, by defining a smooth deformation H between F and G.

Definition 2.1. Two polynomial systems $G : \mathbb{C}^R \to \mathbb{C}^R$ and $F : \mathbb{C}^R \to \mathbb{C}^R$ are (polynomially) *homotopic* if there exists a polynomial system $H : \mathbb{C}^R \times [0, 1] \to \mathbb{C}^R$ such that

$$H(r, 1) = G(r) \quad \text{and} \quad H(r, 0) = F(r).$$

We call G and F the *starting system* and the *target system* of H, respectively, and we say that

$H(r,t)$ is a *homotopy* between G and F with time parameter t [5].

We compute $\mathcal{V}(F)$ by tracking the roots of the system $H|_t$ as the time parameter t changes from 1 to 0. We note that as t changes, the system $H(r,t)$ deforms from $G(r)$ to $F(r)$, and therefore, the variety $\mathcal{V}(H|_t)$ deforms from $\mathcal{V}(G)$ to $\mathcal{V}(F)$.

Definition 2.2. Let H be a homotopy between the polynomial systems G and F and let ϱ be a nonsingular isolated root of G. A *trackable path* with *starting point* ϱ is a smooth map $\rho : (0,1] \to \mathbb{C}^R$ such that $\rho(1) = \varrho$ and $\rho(t^*)$ is a nonsingular isolated root of $H|_{t^*}$ for all $t^* \in (0,1]$. We define the *endpoint* of ρ to be $\rho(0) = \lim_{t\to 0} \rho(t)$ provided the limit exists.

To track a path means to compute a sequence of points approximating the path. Trackability ensures that this sequence can be computed to the endpoint of a path without encountering a singularity or diverging to infinity. By following a trackable path starting from a root of G, we find that either the endpoint of the path does not exist or the endpoint is a root of F [7].

Example 2.3. Let $F : \mathbb{C}^R \to \mathbb{C}^R$ be a square system of polynomials and let $\varrho \in \mathbb{C}^R$ be a nonsingular root of $F(r) - F(\varrho)$. Then, for $\gamma \in \mathbb{C}$, the homotopy $H^{(\gamma)}$ is given by

$$H^{(\gamma)}(r,t) = (1-t)F(r) + \gamma t(F(r) - F(\varrho)).$$

$H^{(\gamma)}$ has a trackable path from ϱ to a point in $\mathcal{V}(F)$, generically [2]. Note that the target system of $H^{(\gamma)}(r,t)$ is F, so the endpoint of any trackable path is in $\mathcal{V}(F)$. Since ϱ is a nonsingular root of the starting system $H^{(\gamma)}(r,1) = \gamma\,(F(r) - F(\varrho))$, a choice of generic γ ensures the existence of a trackable path starting at ϱ and having an endpoint in $\mathcal{V}(F)$ [2].

Note that for $\gamma = 1$, $H^{(\gamma)}$ is just

$$H^{(1)} = F(r) - tF(\varrho).$$

Therefore, replacing the homotopy $H^{(1)}$ by $H^{(\gamma)}$ admits a trackable path starting at ϱ and ending in $\mathcal{V}(F)$ for generic $\gamma \in \mathbb{C}$. We call this replacement the γ-*trick* [2].

2.2 Numerical path tracking

In this section we apply explicit computation to our implicit definitions by describing a numerical path tracking method. Let ρ be a trackable path of the homotopy H between G and F with endpoint $\rho(0) \in \mathcal{V}(F)$. Unfortunately, we cannot simply plug 0 into ρ and get a root of F, as we do not have an explicit expression for ρ to evaluate. Instead, we generate a finite sequence of approximations to the values of $\rho(t)$ as time t decreases, arriving at an approximation of $\rho(0)$.

Conceptually, we follow a trackable path ρ from time $t = 1$ to $t = 0$, i.e., from $\mathcal{V}(G)$ to $\mathcal{V}(F)$. Computationally, we replace the time variable t by a discrete sequence $\{t_i\}_i$ that decreases from $t_0 = 1$ to 0. For each i, we approximate $\rho(t_{i+1})$ by z_{i+1} by computing the step $\Delta z = z_{i+1} - z_i$. The result is a sequence $\{z_i\}_i$ whose final element approximates $\rho(0)$, a root of F.

In order to compute a step Δz, we employ a predictor-corrector method which predicts z_{i+1} using the local properties of H at z_i and takes Newton steps from that prediction toward a root of H [2]. In practice, the prediction is carried out using a fifth-order Runge-Kutta-Fehlberg method [2, 3]. If the time-steps $\Delta t = t_{i+1} - t_i$ are sufficiently small and computations are sufficiently precise, then the trackability of ρ ensures that each z_i is closer to $\rho(t_i)$ than to any other point in $\mathcal{V}(H(t_i))$ and, therefore, z_{i+1} does not jump to another path.

Example 2.4. For simplicity, we illustrate a predictor step using a first order approximation. Let us consider an arbitrary step in the process, starting at time t_i and approximation z_i. To compute z_{i+1}, we first calculate the derivative of ρ at t_i using implicit differentiation on H. Note that $H(\rho(t), t) = 0$, so we find that

$$\frac{d\rho}{dt} = -\left(\nabla_r H\right)^{-1} \frac{\partial H}{\partial t}.$$

We make a first order prediction of Δz by plugging z_i into this formula and taking a step in the resulting direction

$$\tilde{z}_{i+1} = z_i + \left(t_{i+1} - t_i\right) \left(\nabla_r H\right)^{-1} \left.\frac{\partial H}{\partial t}\right|_{z_i, t_i}.$$

We compute a pair of Newton steps to step from $\tilde{z}_{i+1}$ to z_{i+1}, which is always closer to $\rho(t_{i+1})$ [2].

2.3 Gradient descent homotopies

Although root combination is presented as a problem of computing a polynomial, it can be expressed as a problem of finding the nearest point on a variety to a given starting point. In this section, we develop *gradient descent homotopies* in order to find the solutions of a system of polynomial equations nearest to a given starting point. Standard homotopy continuation does not impose extra conditions on the solutions to a system of polynomial equations. In order to impose conditions, we encode them as polynomial equations, which are then added to the system. Finding the roots of the augmented system then ensures that our conditions are met.

2.3.1 Real critical points

Let $F(r) : \mathbb{C}^R \to \mathbb{C}^m$ be a system of m polynomials with real coefficients and let $R \geq m$ so that $\mathcal{V}(F)$ is generically nonempty. We choose a point $\varrho \in \mathbb{R}^R$ to be our starting point. Note that $F(r) = 0$ may have many solutions, and in fact $\mathcal{V}(F)$ has components of dimension $R - m$, generically. We seek to find a point in $\mathcal{V}_{\mathbb{R}}(F)$ minimizing the distance between ϱ and $\mathcal{V}(F)$.

Problem 2.5.

$$\begin{aligned} &\min \ \ \|r - \varrho\|^2 \\ &\text{S.t.} \ \ F(r) = 0 \end{aligned}$$

However, we do not directly obtain a real solution of Problem 2.5 using homotopy continuation because the problem cannot be stated as a system of polynomial equations. Instead, we use homotopy continuation to find *real critical points* of Problem 2.5, and test them to find a solution.

Definition 2.6. A *real critical point* of Problem 2.5 is a point $\rho^* \in \mathcal{V}_{\mathbb{R}}(F)$ such that either

1. $\rho^* - \varrho$ is in the normal space of $\mathcal{V}(F)$ at ρ^* or
2. ρ^* is a singular point of $\mathcal{V}(F)$.

Proposition 2.7. All real solutions of Problem 2.5 are also real critical points of Problem 2.5.

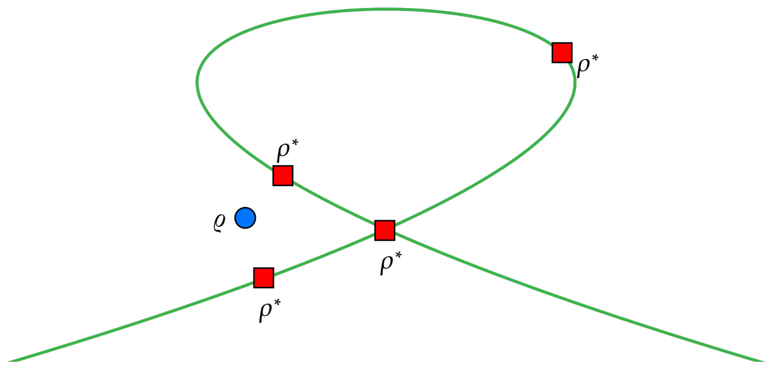


Figure 2.1: A visualization of real critical points of Problem 2.5, represented by red squares, lying on the variety $\mathcal{V}(x^2 + y^3 - 5y^2)$ which is plotted in green. There are four real critical points, one of which is the singular point of the variety. The starting point ϱ is represented by a blue circle to the left of the singular point. This figure was created using the Mathcha Editor [1].

Proof. If ρ^* is a real singular point of $\mathcal{V}(F)$, then ρ^* is also a real critical point of Problem 2.5 by definition.

Therefore, assume that ρ^* is a nonsingular real solution of Problem 2.5. We use Lagrange multipliers to show that ρ^* is a real critical point of Problem 2.5. Consider the Lagrange function of Problem 2.5, which is given by

$$\mathcal{L}(r, \ell) = \|r - \varrho\|^2 + \ell_1 F_1(r) + \cdots + \ell_m F_m(r).$$

The gradient of $\mathcal{L}$ is given by

$$\nabla\mathcal{L}(r, \ell) = \begin{bmatrix} 2(r - \varrho) + \ell_1 \nabla F_1(r) + \cdots + \ell_m \nabla F_m(r) \\ F(r) \end{bmatrix}$$

By the Lagrange multiplier theorem, there exists $\lambda^* \in \mathbb{C}^m$ such that $\nabla\mathcal{L}(\rho^*, \lambda^*) = 0$. Note that λ^* is a particular value of ℓ, as discussed in Table 1.3. Therefore, $\rho^* \in \mathcal{V}_{\mathbb{R}}(F)$ since $F(\rho^*) = 0$ and

$$\rho^* - \varrho = -\frac{1}{2}\left(\lambda_1^* \nabla F_1(\rho^*) + \cdots + \lambda_m^* \nabla F_m(\rho^*)\right).$$

Note that the vectors ∇F_k span the normal space of $\mathcal{V}(F)$ at nonsingular points of the variety.

Therefore, $\rho^* - \varrho$ lies in the normal space of $\mathcal{V}(F)$. Thus, ρ^* is a real critical point of Problem 2.5. Therefore, every real solution of Problem 2.5 is a real critical point. $\square$

To define a trackable path to the solution of Problem 2.5, we first construct a target system whose roots are precisely the real critical points of Problem 2.5. We obtain this target system by augmenting the gradient $\nabla\mathcal{L}$ of the Lagrange function discussed in the proof of Proposition 2.7. We recall from the proof of Proposition 2.7 that for any $(\rho^*, \lambda^*) \in \mathcal{V}_{\mathbb{R}}(\nabla\mathcal{L})$, ρ^* is a real critical point of Problem 2.5. However, the converse does not hold, as real singular points of $\mathcal{V}(f)$ are not roots of $\nabla\mathcal{L}$, generically. We modify $\nabla\mathcal{L}$ to construct L by adding an additional Lagrange multiplier ℓ_0 to the system, defining L by

$$L(r, \ell) = \begin{bmatrix} F(r) \\ \ell_0(r - \varrho) + \ell_1 \nabla F_1(r) + \cdots + \ell_m \nabla F_m(r) \end{bmatrix}.$$

If ρ^* is a singular point of $\mathcal{V}(F)$, then there exists a particular value of ℓ given by $\lambda^* \neq 0$ such that $L(\rho^*, \lambda^*) = 0$, since

$$L(r, \ell)|_{l_0 = 0} = \begin{bmatrix} F(r) \\ \ell_1 \nabla F_1(r) + \cdots + \ell_m \nabla F_m(r) \end{bmatrix}.$$

In fact, for any real critical point ρ^* of Problem 2.5, there exists λ^* such that $L(\rho^*, \lambda^*) = 0$. We have constructed L so that its zeros include all real critical points. However, we have gone too far; for any root ρ^* of F, $L(\rho^*, 0) = 0$. Therefore, we must augment L so that it has no roots when ℓ takes on the value 0. We select α from $\mathbb{C}^{m+1}$ and define $L^{(\alpha)}$ by

$$L^{(\alpha)}(r, \ell) = \begin{bmatrix} F(r) \\ \ell_0(r - \varrho) + \ell_1 \nabla F_1(r) + \cdots + \ell_m \nabla F_m(r) \\ \ell_0 + \alpha_1 \ell_1 + \cdots + \alpha_m \ell_m - \alpha_0 \end{bmatrix}.$$

Since α_0 is nonzero generically, $L^{(\alpha)}(r, 0)$ has no roots. In fact, the equation

$$\ell_0 + \alpha_1 \ell_1 + \cdots + \alpha_m \ell_m - \alpha_0 = 0$$

restricts ℓ to an affine patch of the complex projective space $\mathbb{P}^m$. This prevents ℓ from scaling; i.e., if $L^{(\alpha)}(\rho^*, \lambda^*) = 0$, then for any $\beta \in \mathbb{C}$ such that $\beta \neq 1$, $L^{(\alpha)}(\rho^*, \beta\lambda^*) \neq 0$. Note that $L^{(\alpha)}$ is a square system of $R+m+1$ inputs and polynomials, so $\mathcal{V}(L^{(\alpha)})$ is zero-dimensional generically. In fact, for a generic $\alpha \in \mathbb{C}^{m+1}$, ρ^* is a real critical points of Problem 2.5 if and only if there exists $\lambda^* \in \mathbb{C}^{m+1}$ such that $L^{(\alpha)}(\rho^*, \lambda^*) = 0$ [7].

We construct a homotopy by combining the target system $L^{(\alpha)}$ and the starting point $(\varrho, (\alpha_0, 0, \dots, 0))$ with the homotopy $H^{(1)}$ from Example 2.3. With this target system and starting point the resulting homotopy H is given by

$$\begin{aligned} H(r,\ell,t) &= L^{(\alpha)}(r,\ell) - tL^{(\alpha)}(\varrho, (\alpha_0, 0, \dots, 0)) \\ &= \begin{bmatrix} F(r) \\ \ell_0(r-\varrho) + \ell_1 \nabla F_1(r) + \cdots + \ell_m \nabla F_m(r) \\ \ell_0 + \alpha_1 \ell_1 + \cdots + \alpha_m \ell_m - \alpha_0 \end{bmatrix} - t \begin{bmatrix} F(\varrho) \\ 0 \\ 0 \end{bmatrix}. \end{aligned}$$

We see that our choice of starting point is a common root of the polynomials in $L^{(\alpha)}$ which depend on ℓ, so H is very similar in form to $L^{(\alpha)}$. Therefore, our *gradient descent homotopy* is

$$H(r,\ell,t) = \begin{bmatrix} F(r) - tF(\varrho) \\ \ell_0(r-\varrho) + \ell_1 \nabla F_1(r) + \cdots + \ell_m \nabla F_m(r) \\ \ell_0 + \alpha_1 \ell_1 + \cdots + \alpha_m \ell_m - \alpha_0 \end{bmatrix}. \tag{2.8}$$

For generic $\alpha \in \mathbb{C}^{m+1}$, if ϱ is a nonsingular root of $F(r) - F(\varrho)$, then, by the gamma trick mentioned in Example 2.3, a trackable path (ρ, λ) of H exists which starts at $(\varrho, (\alpha_0, 0, \dots, 0))$ and ends at a real critical point of Problem 2.5.

Chapter 3

Root combination

3.1 Solving for parameters with gradient descent homotopies

In this section, we specialize Problem 2.5 to minimize a change in only the parameters of a polynomial system, rather than a change in all of the variables. We modify our gradient descent homotopy to find real critical points of the specialized problem. This section summarizes and adapts the work in [7] to the problem of combining roots.

3.1.1 Variables

We replace the polynomial sytem $F(r)$ from Problem 2.5 with a family of polynomial systems $F(r, p)$ parameterized by $p \in \mathbb{C}^P$ with real coefficients, and we require that $R + P \geq m$ rather than $R \geq m$. Let $\varpi \in \mathbb{C}^P$ be our starting value for p. We replace r in Problem 2.5 with the vector

$$\begin{bmatrix} r \\ p \end{bmatrix}.$$

Then, Problem 2.5 becomes

$$\begin{aligned} &\min \|r - \varrho\|^2 + \|p - \varpi\|^2 \\ &\text{S.t. } F(r, p) = 0 \end{aligned}$$

and the gradient descent homotopy H from Equation 2.8 becomes

$$H(r,p,\ell,t) = \begin{bmatrix} F(r,p) - tF(\varrho,\varpi) \\ \ell_0 \begin{bmatrix} r-\varrho \\ p-\varpi \end{bmatrix} + \ell_1 \begin{bmatrix} \nabla_r F_1(r,p) \\ \nabla_p F_1(r,p) \end{bmatrix} + \cdots + \ell_m \begin{bmatrix} \nabla_r F_m(r,p) \\ \nabla_p F_m(r,p) \end{bmatrix} \\ \ell_0 + \alpha_1\ell_1 + \cdots + \alpha_m\ell_m - \alpha_0 \end{bmatrix}.$$

Since p is the parameter defining the system F, a deformation of p is a deformation of F. By modifying the objective function of Problem 2.5, we can describe a minimization problem in a space of polynomial systems rather than a space of all inputs.

3.1.2 Parameter continuation

Consider the problem of finding the system of equations $F(r,\pi^*)$ nearest to $F(r,\varpi)$ such that $\mathcal{V}(F)$ is nonempty; i.e.,

Problem 3.1.

$$\begin{aligned} &\min \|p-\varpi\|^2 \\ &\text{S.t. } F(r,p) = 0 \end{aligned}$$

Problem 3.1 restricts the value of r in order to minimize the magnitude of $p-\varpi$, but we observe that the magnitude of $r-\varrho$ is not part of the objective function.

Definition 3.2. A point $\pi^* \in \mathbb{R}^P$ is a *real critical point* of Problem 3.1 if there exists $\rho^* \in \mathbb{C}^R$ such that $F(\rho^*,\pi^*) = 0$ and either,

1. $\pi^* - \varpi$ is in the normal space of $\mathcal{V}(F|_{\rho^*})$ at π^* or

2. (ρ^*,π^*) is a singular point of $\mathcal{V}(F)$.

In order to find the real critical points of Problem 3.1, we build a homotopy H' which performs gradient descent in the space of parameters $\mathbb{C}^P$. The homotopy H' tracks the input variables (r,p) along the trackable path (ρ,π) with starting point (ϱ,ϖ). As we track p along the path π, we track the system of polynomials $F(r,p)$ along the path $F(r,\pi)$. The homotopy which tracks r and p to real critical points of Problem 3.1 is given by

$$H'(r,p,\ell,t) = \begin{bmatrix} F(r,p) - tF(\varrho,\varpi) \\ \ell_0 \begin{bmatrix} 0 \\ p-\varpi \end{bmatrix} + \ell_1 \begin{bmatrix} \nabla_r F_1(r,p) \\ \nabla_p F_1(r,p) \end{bmatrix} + \cdots + \ell_m \begin{bmatrix} \nabla_r F_m(r,p) \\ \nabla_p F_m(r,p) \end{bmatrix} \\ \ell_0 + \alpha_1\ell_1 + \cdots + \alpha_m\ell_m - \alpha_0 \end{bmatrix}. \tag{3.3}$$

The only difference between H and H' is the vector multiplied by ℓ_0. In both H and H', this vector is the gradient of the objective function of the problem, up to a constant factor. In particular, for Problem 3.1, $\nabla_r\|p-\varpi\|^2 = 0$ and $\nabla_p\|p-\varpi\|^2 = 2(p-\varpi)$, giving us the term

$$\ell_0 \begin{bmatrix} 0 \\ p-\varpi \end{bmatrix}.$$

This change in homotopies forces $p-\varpi$ to move in the direction of gradient descent [7].

3.1.3 Splitting off a condition

We consider a special case of Problem 3.1 by adding conditions to F. let $\mathcal{F}(r,p) : \mathbb{C}^R \times \mathbb{C}^P \to \mathbb{C}^{m-1}$ and $g : \mathbb{C}^R \to \mathbb{C}$ be polynomials and consider

$$F(r,p) = \begin{bmatrix} \mathcal{F}(r,p) \\ g(r) \end{bmatrix}$$

where $\mathcal{F}(\varrho,\varpi) = 0$. Note that ϱ is a root of $\mathcal{F}(r,\varpi)$ but it is not necessarily a root of g. Then Problem 3.1 becomes:

Problem 3.4.

$$\begin{aligned} \min\ & \|p-\varpi\|^2 \\ \text{S.t. } & \mathcal{F}(r,p) = 0 \\ & g(r) = 0. \end{aligned}$$

We consider the homotopy H' from Equation 3.3 and observe that, in Problem 3.4,

$$F(r,p) - tF(\varrho,\varpi) = \begin{bmatrix} \mathcal{F}(r,p) - t\mathcal{F}(\varrho,\varpi) \\ g(r) - tg(\varrho) \end{bmatrix} = \begin{bmatrix} \mathcal{F}(r,p) \\ g(r) - tg(\varrho) \end{bmatrix}.$$

Therefore, in Problem 3.4, our gradient descent homotopy H' is

$$H'(r,p,\ell,t) = \begin{bmatrix} \mathcal{F}(r,p) \\ g(r) - tg(\varrho) \\ \ell_0 \begin{bmatrix} 0 \\ p - \varpi \end{bmatrix} + \ell_1 \begin{bmatrix} \nabla_r \mathcal{F}_1(r,p) \\ \nabla_p \mathcal{F}_1(r,p) \end{bmatrix} + \cdots + \ell_{m-1} \begin{bmatrix} \nabla_r \mathcal{F}_{m-1}(r,p) \\ \nabla_p \mathcal{F}_{m-1}(r,p) \end{bmatrix} + \ell_m \begin{bmatrix} \nabla g(r) \\ 0 \end{bmatrix} \\ \ell_0 + \alpha_1 \ell_1 + \cdots + \alpha_m \ell_m - \alpha_0 \end{bmatrix}.$$

We recall that (ρ,π) is a trackable path of H'. Therefore, $\rho(t^*)$ is a root of $F(r,\pi(t^*))$ for all $t^* \in (0,1]$. In particular, ρ is a path starting at the root ϱ of $\mathcal{F}(r,\varpi)$ and ending at the root $\rho(0)$ of $g(r)$ and $F(r,\pi(0))$. $F(r,\pi)$ is likewise a path starting at the system $\mathcal{F}(r,\varpi)$ and ending at the system $\mathcal{F}(r,\pi(0))$.

3.2 Combining roots

In this section, we show how to combine the roots of a polynomial using a gradient descent homotopy.

3.2.1 Set up of the problem

Let $f(x,p) : \mathbb{C} \times \mathbb{C}^P \to \mathbb{C}$ be a family of univariate polynomials in x parameterized by $p \in \mathbb{C}^P$, a vector of the coefficients of the polynomials. In other words $f(x,p)$ is given by

$$f(x,p) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{P-1} x^{P-1}.$$

We use homotopy continuation to deform p from our *starting coefficients* $\varpi \in \mathbb{R}^P$ to some target coefficients $\pi(0) \in \mathbb{R}^P$ as the parameter t changes from 1 to 0. Our goal for this project is to find

$\pi(0) \in \mathbb{R}^P$ such that $f(x, \pi(0))$ has greater multiplicity structure than $f(x, \varpi)$ while minimizing the change in the coefficients of f.

Problem 3.5.

$$\begin{aligned} &\min \ \|p - \varpi\|^2 \\ &\text{S.t. } f(x,p) \text{ has greater multiplicity structure than } f(x,\varpi). \end{aligned}$$

Definition 3.6. We say that $f(x, \pi^*)$ *combines* roots of $f(x, \varpi)$ if π^* is a real critical point of Problem 3.5.

In order to use homotopy continuation to combine the roots of $f(x, \varpi)$, we restate Problem 3.5 in the form of Problem 3.4 by defining $\mathcal{F}$ and g. For our convenience, we introduce new notation for the multiplicity of roots of polynomials in the parameterized family $f(x, p)$ (see Definition 1.2 for the old notation).

Definition 3.7. Let $\pi^* \in \mathbb{C}^P$. Then the *multiplicity* of the root ρ^* of $f(x, \pi^*)$ is given by $e(\rho^*, \pi^*)$; i.e.,

$$e(\rho^*, \pi^*) = e(\rho^*, f|_{\pi^*}).$$

Let $\varrho \in R$ be a vector of the distinct roots of $f(x, \varpi)$. We use the variable r to represent the roots of the parameterized family $f(x, p)$. We define $\mathcal{F}(r, p)$ as the system of polynomials

$$\frac{\partial^i}{\partial x^i} f(r_j, p)$$

for all $0 \leq i < e(\varrho_j, \varpi)$. Let (ρ^*, π^*) be a root of $\mathcal{F}$. Then, for any $1 \leq j \leq R$ and any $0 \leq i < e(\varrho_j, \varpi)$,

$$\frac{\partial^i}{\partial x^i} f(\rho_j^*, \pi^*) = 0. \tag{3.8}$$

Therefore, ρ_j^* is a root of $f(x, \pi^*)$ with multiplicity at least $e(\varrho_j, \varpi)$ for all j. Therefore, $f(x, \pi^*)$ has at least the multiplicity structure of $f(x, \varpi)$ and the roots of $f(x, \pi^*)$ are the entries of ρ^*. On the other hand, suppose that $f(x, \pi^*)$ has at least the multiplicity structure of $f(x, \varpi)$. Then, for all $1 \leq j \leq R$, there exists a root ρ_j^* of $f(x, \pi^*)$ such that Equation 3.8 holds for all $0 \leq i < e(\varrho_j, \varpi)$. Note that $\mathcal{F}(\varrho, \varpi) = 0$, as in Problem 3.4. We define $g(r)$ as the polynomial

$$g(r) = \prod_{i<j} (r_i - r_j).$$

Note that ρ^* is a root of g if and only if $\rho_i^* = \rho_j^*$ for some $i \neq j$; i.e., ρ^* has fewer than R distinct elements. Therefore, $f(x, \pi^*)$ has greater multiplicity structure than $f(x, \varpi)$ if and only if there exists $\rho^* \in \mathbb{C}^R$ such that $\mathcal{F}(\rho^*, \pi^*) = 0$ and $g(\rho^*) = 0$. Thus, Problem 3.5 can be written

Problem 3.9.

$$\begin{aligned} \min \ & \|p - \varpi\|^2 \\ \text{S.t. } & \mathcal{F}(r, p) = 0 \\ & g(r) = 0. \end{aligned}$$

Since Problem 3.9 is a special case of Problem 3.4, it has the gradient descent homotopy

$$H'(r, p, \ell, t) = \begin{bmatrix} \mathcal{F}(r, p) \\ g(r) - tg(\varrho) \\ \ell_0 \begin{bmatrix} 0 \\ p - \varpi \end{bmatrix} + \ell_1 \begin{bmatrix} \nabla_r \mathcal{F}_1(r, p)^T \\ \nabla_p \mathcal{F}_1(r, p)^T \end{bmatrix} + \cdots + \ell_{P-1} \begin{bmatrix} \nabla_r \mathcal{F}_{P-1}(r, p)^T \\ \nabla_p \mathcal{F}_{P-1}(r, p)^T \end{bmatrix} + \ell_P \begin{bmatrix} \nabla g(r) \\ 0 \end{bmatrix} \\ \ell_0 + \alpha_1 \ell_1 + \cdots + \alpha_P \ell_P - \alpha_0 \end{bmatrix}. \tag{3.10}$$

Suppose (ρ, π, λ) is the trackable path of H' with the starting point $(\varrho, \varpi, (\alpha_0, 0, \ldots, 0))$. Then $\pi(0)$ is a real critical point of Problem 3.9. Therefore, $f(x, \pi(0))$ combines roots of $f(x, \varpi)$ and $\mathcal{V}(f|_{\pi(0)}) = \{\rho_1(0), \ldots, \rho_R(0)\}$.

Chapter 4

Pseudocode

We use the notation `a` $\leftrightarrow a$ to indicate that a variable `a` in the pseudocode corresponds to a symbol a in Section 3.2. We implement the homotopy $H'(r, p, \lambda, t)$ from Equation 3.10 in the following way: We define a method called `rootReduce` which takes the option `Sparse`, a polynomial `f`, and optionally the list of roots `Roots` of the function `f`. `rootReduce` attempts to return a polynomial `fS0` which combines roots of `f`.

The Boolean variable `Sparse` allows us to decide if the algorithm maintains sparsity of monomials. That is, only monomials that appear in the input polynomial `f` appear in the output polynomial if `Sparse` is set to true. In this case, `rootReduce` solves a modified version of Problem 3.9 in which $f(x, p)$ has the sparsity of $f(x, \varpi)$; i.e., ϖ has only nonzero entries and $f(x, p) = p_0x^{j_0} + \cdots + p_Px^{j_P}$ where $j_i \geq i$. `Sparse` is set to false by default.

The list of points `Roots` $\leftrightarrow \varrho$ represents the roots of the input polynomial. These points must include their multiplicity data. When `Roots` is null, `rootReduce` finds the roots of `f` without user input. The algorithm follows:

1. x, the indeterminate:

   (a) check that `f` $\leftrightarrow f(x, \varpi)$ is univariate and nonconstant. $f(x, p)$ is a family of polynomials parameterized by p, which is a vector of the coefficients of $f(x, p)$. ϖ is our starting parameter, so $f(x, \varpi)$ is our starting polynomial, whose roots we wish to combine.

   (b) Assign `x` $\leftrightarrow x$ to be the indeterminate variable of `f`, so that we have `f` $\in \mathbb{R}[$`x`$]$.

2. Roots of $f(x, \varpi)$:

(a) We assign the list of the roots of `f` to `yList` $\leftrightarrow \varrho$. Note that $\varrho = \mathcal{V}(f|_{\varpi})$, the roots of our starting polynomial. If the roots are not given as input, then we compute them with Bertini using the function `bertiniZeroDimSolve`. Each root should include its coordinates and multiplicity.

(b) If `f` only has one root, then return `f`.

The goal of `RootReduce` is to reduce the number of roots of `f` by combining roots. So once there's only one left, we're done because there is nothing to combine.

(c) Separate the multiplicities of `yList` into a separate list `multList` $\leftrightarrow \{e(\varrho_j, \varpi)\}_j$. From Definition 3.7 $\{e(\varrho_j, \varpi)\}_j$ is the list of multiplicities of our starting polynomial $f(x, \varpi)$. Define `nRoots` to be the largest index of `yList`. Note we have `nRoots` $\leftrightarrow R - 1$, since we have zero-based array indexing. R is the number of roots of $f(x, \varpi)$, and thus the size of r. `nRoots` is thus the largest index of the list, but is one less than the size of the list.

(d) Define the integer `nLagrange` $\leftrightarrow P$ by

$$\texttt{nLagrange} := 1 + \sum_{k=0}^{\texttt{nRoots}} \texttt{multList}_k.$$

We call this variable `nLagrange` because it corresponds to the Lagrange multiplier system in the homotopy. P, here, is one greater than the degree of $f(x, \varpi)$. Note that

$$P - 1 = \sum_{k=1}^{R} e(\varrho_k, \varpi).$$

3. Terms and Coefficients:

(a) Assign `monomVec` to be the row vector of all monomials of `f`, and assign `qVec` to be the vector of all coefficients of `f`. `qVec` $\leftrightarrow \varpi$, since ϖ is the vector of parameters (and thus coefficients) of $f(x, \varpi)$. Note that if `Sparse` is set to true, and thus we maintain sparsity of monomials, `qVec` does not include zeros and `monomVec` only contains monomials corresponding to coefficients found in `qVec`. Otherwise, if we do not maintain sparsity, i.e., `Sparse` is set to false, then `monomVec` includes all the monomials $1, \mathtt{x}, \mathtt{x}^2, \ldots, \mathtt{x}^{P-1}$.

(b) Define `nCoef` to be the number of entries in `qVec` minus one. Thus, `qVec` is given by the vector

$$\mathtt{qVec} = \begin{bmatrix} \mathtt{qVec}_0 \\ \vdots \\ \mathtt{qVec}_{\mathtt{nCoef}} \end{bmatrix}.$$

`nCoef` is the largest index of `qVec`. Thus, in the non-sparse case, $\mathtt{nCoef} \leftrightarrow P-1$ is the degree of `f` since we have

$$\mathtt{monomVec} = \begin{bmatrix} 1 & \mathtt{x} & \mathtt{x}^2 & \cdots & \mathtt{x}^{\mathtt{nCoef}} \end{bmatrix}$$

and

$$\begin{aligned} \mathtt{f} &= \mathtt{qVec}_0 + \mathtt{qVec}_1\mathtt{x} + \cdots + \mathtt{qVec}_{\mathtt{nCoef}}\mathtt{x}^{\mathtt{nCoef}} \\ &= \mathtt{qVec}_0\mathtt{monomVec}_0 + \mathtt{qVec}_1\mathtt{monomVec}_1 + \cdots + \mathtt{qVec}_{\mathtt{nCoef}}\mathtt{monomVec}_{\mathtt{nCoef}} \\ &= \mathtt{monomVec} \cdot \mathtt{qVec} \end{aligned}$$

Note that in the sparse case we still have $\mathtt{f} = \mathtt{monomVec}\,\mathtt{qVec}$.

(c) For the rest of this section, we assume that `Sparse` is set to false because this case corresponds to Problem 3.9. The case where `Sparse` is set to true does not correspond to a form of root combination discussed in this project.

(d) Define the new ring

$$\mathtt{R} := \mathbb{C}[\mathtt{x}, \mathtt{p}_0, \ldots, \mathtt{p}_{\mathtt{nCoef}}]$$

where $\mathtt{p}_k \leftrightarrow p_k$ and $\mathtt{R} \leftrightarrow \mathbb{C}[x,p]$. p is a vector of indeterminates representing the parameters and coefficients of $f(x,p)$. $\mathbb{C}[x,p]$ is the polynomial ring containing the parameterized family $f(x,p)$.

(e) Make the vector

$$\mathtt{pVec} = \begin{bmatrix} \mathtt{p}_0 \\ \vdots \\ \mathtt{p}_{\mathtt{nCoef}} \end{bmatrix}.$$

The assignment of $\texttt{pVec} \leftrightarrow p$ allows us to define a parameterized version of $\texttt{f}$ as

$$\begin{bmatrix} \texttt{fp} \end{bmatrix} := \texttt{monomVec}\,\texttt{pVec}$$

and therefore

$$\texttt{fp} = \texttt{p}_0 + \texttt{p}_1\texttt{x} + \cdots + \texttt{p}_{\texttt{nCoef}}\texttt{x}^{\texttt{nCoef}}.$$

$\texttt{fp} \leftrightarrow f(x,p)$. $f(x,p)$ is the parameterized family of polynomials that we use in Section 3.2. Our starting polynomial $f(x,\varpi)$ is a member of this family.

4. The ring for the homotopy:

   Define a polynomial ring that contains all of the homotopy inputs as indeterminates.

$$\texttt{RHom} := \mathbb{C}[\texttt{r}_0,\ldots,\texttt{r}_{\texttt{nRoots}},\texttt{p}_0,\ldots,\texttt{p}_{\texttt{nCoef}},\texttt{l}_0,\ldots,\texttt{l}_{\texttt{nLagrange}},\texttt{timeVar},\texttt{pathVar}].$$

   Here, $\texttt{r}_k \leftrightarrow r_k$, $\texttt{l}_k \leftrightarrow \ell_k$, $\texttt{timeVar} \leftrightarrow t$, and $\texttt{pathVar} \leftrightarrow t$. r is the vector of indeterminants used to represent the roots of $f(x,p)$ in the homotopy H'. ℓ is the vector of indeterminate Lagrange coefficients we use to ensure that our output consists of real critical points when we construct H'. t is the time variable in H'.

5. The polynomial $g(r)$:

   Define $\texttt{g} \leftrightarrow g(r) - tg(\varrho)$ by

$$\texttt{g} = \prod_{i<j}(\texttt{r}_i - \texttt{r}_j) - \texttt{timeVar}\prod_{i<j}(\texttt{yList}_i - \texttt{yList}_j).$$

   $g(r) = \prod_{i<j}(r_i - r_j)$ is a polynomial function such that $g(r) = 0$ if and only if two of the entries of r are equivalent.

6. The system $F(r,p)$:

   (a) For each $k \in \{0,\ldots,\texttt{nRoots}\}$, define $\texttt{fr}_k \leftrightarrow f(r_k,p)$ by

$$\texttt{fr}_k := \texttt{f}_{\texttt{p}}(\texttt{r}_k,\texttt{p}_0,\ldots,\texttt{p}_{\texttt{nCoefs}}) \in \texttt{RHom}.$$

$f(r_k, p)$ is the result of substituting the indeterminate x for the indeterminate r_k in the polynomial family $f(x, p)$. Note that r_k represents a root of $f(x, p)$, so substituting related values for r_k and p into $f(r_k, p)$ should give 0.

(b) Define the function `multiDiff` by

$$\texttt{multiDiff}(x, f, n) = \left(f, \frac{\partial f}{\partial x}, \frac{\partial^2 f}{\partial x^2}, \ldots, \frac{\partial^n f}{\partial x^n}\right)$$

(c) Define $\texttt{F} \leftrightarrow \mathcal{F}(r, p)$ as the vector obtained from concatenating the sequences

$$\texttt{multiDiff}(\texttt{r}_k, \texttt{fr}_k, \texttt{multList}_k - 1)$$

from $k = 0$ to `nRoots`.

$$\mathcal{F}(r, p) = \left(\frac{\partial^i}{\partial x^i} f(r_j, p)\right)_{ij}$$

is a system that equals 0 when r is the set of roots of $f(x, p)$ with at least the multiplicity structure of $f(x, \varpi)$.

7. The homotopy $H'(r, p, \ell, t)$:

(a) Now we put the homotopy together one piece at a time, starting at the top. We assign

$$\texttt{Htop} := \begin{bmatrix} \texttt{F} \\ \texttt{g} \end{bmatrix}.$$

(b) We assign

$$\texttt{LagrangeMat} := \begin{bmatrix} 0 & \mathcal{J}_{\texttt{r}}\texttt{F} & \nabla_{\texttt{r}}\texttt{g} \\ \begin{bmatrix} \texttt{p}_0 \\ \vdots \\ \texttt{p}_{\texttt{nCoef}} \end{bmatrix} - \texttt{qVec} & \mathcal{J}_{\texttt{p}}\texttt{F} & \nabla_{\texttt{p}}\texttt{g} \end{bmatrix}$$

`LagrangeMat` corresponds to the matrix which has rank less than or equal to P exactly at real critical points of Problem 3.9.

(c) We assign `lambdaVec` $\leftrightarrow \ell$ as

$$\mathtt{lambdaVec} := \begin{bmatrix} \mathtt{l}_0 \\ \vdots \\ \mathtt{l}_{\mathtt{nLagrange}} \end{bmatrix},$$

(d) We assign

$$\mathtt{Hmid} := \mathtt{LagrangeMat}\;\mathtt{lambdaVec}$$
$$= \begin{bmatrix} \mathtt{l}_0 0 & + & \sum_{k=1}^{\mathtt{nLagrange}-1} \mathtt{l}_k \nabla_{\mathtt{r}} \mathtt{F}_{k-1} & + & \mathtt{l}_{\mathtt{nLagrange}} \nabla_{\mathtt{r}} \mathtt{g} \\ \mathtt{l}_0(\mathtt{p} - \mathtt{qVec}) & + & \sum_{k=1}^{\mathtt{nLagrange}-1} \mathtt{l}_k \nabla_{\mathtt{p}} \mathtt{F}_{k-1} & + & \mathtt{l}_{\mathtt{nLagrange}} \nabla_{\mathtt{p}} \mathtt{g} \end{bmatrix}.$$

(e) We assign

$$\mathtt{a} := \begin{bmatrix} 1 & \mathtt{a}_1 & \mathtt{a}_2 & \cdots & \mathtt{a}_{\mathtt{nLagrange}-1} & \mathtt{a}_{\mathtt{nLagrange}} & \mathtt{a}_0 \end{bmatrix}$$

randomly generating the $\mathtt{a}_k \leftrightarrow \alpha_k$ in $\mathbb{C}$, and defining $\mathtt{a}_0$ in the process. Note that $\alpha \in \mathbb{C}^{P+1}$ is just a vector of complex numbers.

(f) We assign

$$\mathtt{Hbottom} := \mathtt{a} \begin{bmatrix} \mathtt{lambdaVec} \\ -1 \end{bmatrix} = \begin{bmatrix} \mathtt{l}_0 + \mathtt{a}_1 \mathtt{l}_1 + \cdots + \mathtt{a}_{\mathtt{nLagrange}} \mathtt{l}_{\mathtt{nLagrange}} - \mathtt{a}_0 \end{bmatrix}$$

(g) Now we can put the whole homotopy together, we assign `H` $\leftrightarrow H'(r, p, \lambda, t)$ (see Equation

3.10) as

$$\mathtt{H} := \begin{bmatrix} \mathtt{Htop} \\ \mathtt{Hmid} \\ \mathtt{Hbottom} \end{bmatrix}$$

$$= \begin{bmatrix} \mathtt{F}^T \\ \mathtt{g} \\ \mathtt{l}_0 0 \quad + \sum_{k=1}^{\mathtt{nLagrange}-1} \mathtt{l}_k \nabla_{\mathtt{r}} \mathtt{F}_{k-1} + \mathtt{l}_{\mathtt{nLagrange}} \nabla_{\mathtt{r}} \mathtt{g} \\ \mathtt{l}_0(\mathtt{p} - \mathtt{qVec}) + \sum_{k=1}^{\mathtt{nLagrange}-1} \mathtt{l}_k \nabla_{\mathtt{p}} \mathtt{F}_{k-1} + \mathtt{l}_{\mathtt{nLagrange}} \nabla_{\mathtt{p}} \mathtt{g} \\ \mathtt{l}_0 + \mathtt{a}_1 \mathtt{l}_1 + \cdots + \mathtt{a}_{\mathtt{nLagrange}} \mathtt{l}_{\mathtt{nLagrange}} - \mathtt{a}_0 \end{bmatrix}.$$

H' is the gradient descent homotopy at the core of `RootReduce`. If we follow the trackable path (ρ, π, λ) of H' to its endpoint, we get a polynomial $f(x, \pi(0))$ which combines the roots of $f(x, \varpi)$, solving Problem 3.9.

8. Combining roots:

   (a) Define `qList` as the list of entries in `qVec`.

   (b) Next we assign the start system to a list `S1` $\leftrightarrow (\varrho, \varpi, (\alpha_0, 0, \ldots, 0))$. This is the starting point of the path ρ, π, λ of the homotopy in Equation 3.10.

   $$\mathtt{S1} := \mathtt{yList} \oplus \mathtt{qList} \oplus \{a_0, 0, \ldots, 0\}$$

   where $\oplus$ denotes concatenation and there are `nLagrange` zeros at the end of the list.

   (c) We use the function `bertiniUserHomotopy` to compute

   $$\mathtt{S0} = \{\mathtt{r}, \mathtt{p}, \mathtt{lambdaVec}\} \quad \text{at} \quad \mathtt{pathVar} \equiv \mathtt{timeVar} = 0$$

   where `S0` $\leftrightarrow (\rho(0), \pi(o), \lambda(0))$. $(\rho(0), \pi(o), \lambda(0))$ is the endpoint of our trackable path. Thus, at this step, we have combined two of the roots of `f`.

9. Output:

   (a) Define `rS0` $\leftrightarrow \rho(0)$ as the coordinates of the roots recorded in `S0`. $\rho(0)$ is the vector of roots of $f(x, \pi(0))$.

(b) Define the polynomial `fS0` $\leftrightarrow f(x, \pi(0))$:

$$\texttt{fS0} := \prod_{k=0}^{\texttt{nRoots}+1} (x - \texttt{rS0}_k)^{\texttt{multlist}_k}$$

$f(x, \pi(0))$ is the polynomial which combines the roots of $f(x, \varpi)$.

(c) Output the triple (`fS0`, `rS0`, `S0`).

Appendices

Appendix A The RootReduce package

RootReduce is a Macaulay2 package containing the method rootReduce, which combines the roots of a polynomial. rootReduce takes a polynomial f as input and outputs a polynomial fS0 which combines the roots of f. rootReduce also outputs the roots rS0 of fS0, with multiplicity, and the full endpoint S0 of the homotopy's trackable path. This package was written in Macaulay2 [6] using the Bertini package [8].

```
newPackage(
    "RootReduce",
    Version =>"0.1",
    Date => "October 24, 2019",
    Authors =>{{Name => "Andrew Pitman",
        Email => "apitman@g.clemson.edu"}},
    Headline => "Reduce the number of roots of your polynomial",
    DebuggingMode => false
    )

export{"rootReduce",
    "Roots", -- list of points. option to specify roots
    "Sparse" -- boolean. option to maintain sparsity of terms
    }

    needsPackage("Bertini");

    -- here we create the method which reduces the number of roots of a
    polynomial by altering its coefficients
        -- if Sparse is set to true, then rootReduce will return a
    polynomial with the same set of monomials (with nonzero coefficients) as
     its input
        -- eventually we will add options to modify the homotopy used in
    rootReduce
        -- if specified, Roots should be of the form {(r_1,m_1),(r_2,m_2)
    ,...,(r_n,m_n)} where r_k is a root and m_k is its multiplicity
```

```

rootReduce = method(Options=>{Sparse=>false,Roots=>null});

-- rootReduce needs to act on a ring element regardless of the ring, so we only define the method when it acts on some ring element

rootReduce RingElement := o -> f ->(

    --------------------------
    -- x: the indeterminate --
    --------------------------

    -- throw an error if f is multivariate or constant
    if #support(f) > 1 then error "f needs to be univariate";
    if #support(f) == 0 then error "f is constant";

    -- get the variable used for f
    x := first support f;

    ----------------
    -- roots of f --
    ----------------

    -- if Roots is specified, we expect it to be a list of points
    yList := {};
    if class o.Roots =!= Nothing then yList = o.Roots

    -- otherwise, get the roots of f at the starting parameters, yielding a list of points
    else yList = bertiniZeroDimSolve({f},AffVariableGroup => {x});

    -- if f already has only one root, return f
    if #yList == 1 then return f;

    -- get the multiplicities of the roots and put them in a list
        -- presumably this is unnessary as we can get this information
from the points, but I'm writing the code just in case it turns out to
be super convenient
    multList := apply(yList, u -> u.Multiplicity);

    -- record the number of roots minus 1 (because we will label a 0th
root)
    nRoots := #yList - 1;

    -- record number of lagrange multipliers that we will use minus 1
    nLagrange := sum multList + 1;

    ----------------------------
    -- terms and coefficients --
    ----------------------------

    -- get monomials and coefficients of f
        -- if Sparse (option in this method) is set to true, we only get
 nonzero coefficients and their monomials. Otherwise, we get all
possible monomials and coefficients up to the degree of f
    (monomVec, qVec) := coefficients(f, Monomials=> (if o.Sparse ==
false then apply(first degree f + 1, n-> x^n) else null));

    -- save the number of parameters minus 1 (since we have a parameter
0) to nCoef
    nCoef := numRows qVec - 1;

    -- define p_0 through p_nCoef as symbols within the method (this can
 be done by defining p as a symbol)
    p := symbol p;

    -- define a ring that includes the support of f along with variables
 for the coefficients (parameters) of f
    R := CC[x,p_0..p_nCoef];
    use R;

    -- update the monomials to the new ring R
    monomVec = substitute(monomVec,R);

    -- make a list and a vector of the parameters
    pList := toList (p_0..p_nCoef);
    pVec := transpose matrix {pList};

    -- make a new polynomial fp in R which is formed by replacing the
coefficients of f with parameters (indeterminates in R)
    fp := (monomVec*pVec)_(0,0);

    -------------------------------------
    -- RHom: the ring for the homotopy --
    -------------------------------------

    r := symbol r;
    l := symbol l;
    timeVar := symbol timeVar;
    pathVar := symbol pathVar;
    -- make a variable for each root in a new ring that we will use for
the homotopy
    RHom := CC[ r_0..r_nRoots, p_0..p_nCoef, l_0..l_nLagrange, timeVar,
pathVar ];
    use RHom;

    ---------------------------------------------------------
    -- g: the function that ensures that roots reduce in number --
    ---------------------------------------------------------

    -- indices is the sequence of all pairs (i,j) such that 0 <= i < j <= nRoots.
    indicesMaker := (a,b)->apply(a+1..b,u->(a,u));
    indices := splice apply(0..nRoots-1, u->indicesMaker(u,nRoots));
    g := product( indices, (i,j)->(r_i-r_j)) - timeVar*product(indices, (i,j)->(yList#i#Coordinates#0 - yList#j#Coordinates#0) );

    -- ----------------------------------------------------------------------
    -- F: the function which ensures root variables remain roots as the parameters change --
    -- ----------------------------------------------------------------------

    -- make a polynomial fr_k for each root_k, which is just fp but with indeterminate r_k instead of x
    fMap := i -> map(RHom,R,{r_i,p_0..p_nCoef});
    fr := apply(nRoots+1, j -> (fMap j) fp);

    -- multiDiff takes the 0th through nth proper derivative of f with respect to x and puts all of them in a sequence
    multiDiff := (u,v,n) -> sequence(v) | accumulate( (a,b)->diff(b,a), v, n:u );

    -- F will be directly inserted. It is responsible for making sure that f remains 0 at every root, and that some derivatives of f remain zero at roots of certain multiplicities
    F := splice apply( nRoots+1, i -> ( multiDiff( r_i, fr_i, yList#i#Multiplicity-1 ) ) );

    ---------------------
    -- H: the homotopy --
    ---------------------

    -- First we form the top of the homotopy
    Htop := transpose( matrix{F} | matrix{{g}} );

    -- next we form the middle of the homotopy

    -- LagrangeMat is the matrix whose critical point we'd like to reach
    LagrangeMat := submatrix(jacobian matrix{ F | {g}},{0..nCoef+nRoots
+1},);

    -- we have to form the first column, (0, p-q)^T, of the Lagrange
matrix and then concatenate it on.
    qVec = substitute(qVec, RHom);
    LagrangeCol := matrix{{p_0..p_nCoef}} - transpose qVec;
    LagrangeCol = transpose( matrix{{(nRoots+1):0_RHom}} | LagrangeCol )
;
    LagrangeMat = LagrangeCol | LagrangeMat;

    -- form a vector for the l_k's
    lambdaVec := transpose matrix{{l_0..l_nLagrange}};

    -- the middle of the homotopy in matrix form is Hmid
    Hmid := LagrangeMat * lambdaVec;

    -- next we form the bottom of the homotopy
    setRandomSeed currentTime();
    a := matrix{{1_RHom}} | matrix fillMatrix(mutableMatrix(CC,1,
nLagrange+1));
    Hbottom := a*(lambdaVec || matrix{{-1_RHom}});

    -- get the starting value for l_0
    a0 := last first entries a;

    -- finally we put the top, middle, and bottom together to get the whole homotopy in matrix form
    Hmatrix := Htop || Hmid || Hbottom;

    -- convert the homotopy to list form
    H := first entries transpose (Htop || Hmid || Hbottom);
    -- next we form the Bertini Command

    ---------------------
    -- combining roots --
    ---------------------

    -- create a simple list of our starting parameters
    qList := first entries transpose qVec;

    -- place the starting system in a variable --r,p,l,a
    S1 := {apply(yList, u->u#Coordinates),qList,a0,toList(nLagrange:0)};
    S1 = flatten flatten S1;


    -- COMPUTE the actual homotopy
    S0 := bertiniUserHomotopy(pathVar,{timeVar=>pathVar},H,{S1});

    ------------
    -- output --
    ------------

    -- now we need to automatically identify the roots that have combined.

    -- output the new polynomial, its roots, and the bertini output
    use CC[x];
```

```
        cS0 := take(S0#0#Coordinates, nRoots+1);
        fS0 := product apply(cS0, multList, (y,z)-> (x - realPart y)^z);
        rS0 := apply(cS0, multList, (y,z)->point {{y},Multiplicity=>z});
        return (fS0,rS0,S0);

        )

beginDocumentation()
multidoc ///
    Node
        Key
            RootReduce
        Headline
            Reduce the number of roots of your polynomial
        Description
            Text
                {\em RootReduce} contains the function {\em rootReduce}.
    Node
        Key
            (rootReduce,RingElement)
            rootReduce
        Headline
             {\em rootReduce} can reduce the number of roots of a univariate
     polynomial with minimial change to its coefficients.
        Usage
            rootReduce f
        Inputs
            f:
        Outputs
            :
                a polynomial of the same degree as {\tt f}, but with fewer
    roots
    Node
```

```
        Key
            Roots
        Headline
            Optional list of points representing the roots of {\tt f}
        Usage
            Roots => {point{{-1.5},Multiplicity=>2},point{{1.0},Multiplicity
    =>3}}
        Caveat
            Multiplicity must be specified for each point
    Node
        Key
            Sparse
        Headline
            option to maintain sparsity of terms of {\tt f}. {\tt false} by
    default.
///

TEST ///
    R = CC[x,y]; rootList = {{-2,1},{-1,1},{1,1},{1.8,1},{2.2,1}};
    rootPoints = apply(rootList, z->point{{z#0},Multiplicity=>z#1}); peek
    rootPoints; p1 = product apply(rootList, r->(y-r#0)^(r#1)); peek p1; p2
    = rootReduce(p1)
///

end --
```

Appendix B   Example computation files

B.1   Directory structure

In order to produce Examples 1.5 and 1.7, I used scripts written in bash and Macaulay2 [6]. The directory for those scripts is depicted in Table B.1. The directory `combining_roots` contains the child directories `input` and `output`, as well as the scripts `run.pbs` and `run.sh`, which are the main scripts used to run all other scripts. The directory `input` contains the script `combine.m2` as well as the input files for `combine.m2`, which have the `.i` extension. The directory `output` contains the files `run.o` and `run.e`, which contain the output and error messages produced by the scripts in this file. `output` also contains the output files produced by `combine.m2`, which have the `.o` extension.

B.2   run.pbs

The script `run.pbs` runs `combine.m2` as a job on the Palmetto Cluster. It sends standard output to `run.o` and standard error to `run.e`.

```
#PBS -N reduce
#PBS -l select=1:ncpus=8:ngpus=2:gpu_model=k20:mem=125gb,walltime=10:00:00
#PBS -o /scratch1/apitman/output/run.o
#PBS -e /scratch1/apitman/output/run.e

echo "RUNNING reduce.m2"
M2 /scratch1/apitman/input/combine.m2
echo "run.pbs FINISHED"
```

combining_roots	input	combine.m2
		*.i
	output	run.o
		run.e
		*.o
	run.pbs	
	run.sh	

Table B.1: This is a representation of the directory for `combining_roots`, which contains the scripts used to produce Examples 1.5 and 1.7.

B.3 run.sh

The script `run.sh` runs `combine.m2` on a local machine while writing standard output to `run.o` and standard error to `run.e` in order to mimic the behavior of `run.pbs`.

```
echo "RUNNING reduce.m2"
M2 ./input/combine.m2 > ./output/run.o 2> ./output/run.e
echo "run.pbs FINISHED"
```

B.4 combine.m2

The script `combine.m2` is written in Macaulay2 [6] with the Bertini package [8]. It takes a set of roots and multiplicities from the file `filename.i`, combines those roots using the method `rootReduce`, and outputs the combined roots to `filename.o`. We then edit the output by hand to produce the next input file.

```
-- combine.m2
-- Andrew Pitman

stderr << endl << "SCRIPT RUNNING" << endl << endl;

-- EDIT FILENAME HERE
s = "filename"; -- string specifying input/output filenames


printingPrecision = 0; -- print maximum number of digits
needsPackage "Bertini"; -- provides the "point" data type
needsPackage "RootReduce"; -- package for combining roots
R := CC[x]; -- the ring of complex univariate polynomials

stderr << "LOADING INPUT" << endl << endl;
InLines := lines get openIn("./"|s|".i"); -- read string from input file
varrho := value InLines#0; -- assign roots of starting polynomial
stderr << endl << endl << endl << "ROOTS varrho: " << endl << endl << peek
    varrho << endl << endl << endl;
```

```

fVarpi := product apply(varrho, a -> (x - a#Coordinates#0)^(a#Multiplicity))
    ; -- build the starting polynomial
stderr << "POLYNOMIAL fVarpi: " << endl << endl << peek fVarpi << endl <<
    endl << endl;

(fpi0, rho0, bert0) := rootReduce(fVarpi, Roots=>varrho); -- combine roots;
    save roots as rho0
stderr << "ROOTS COMBINED" << endl << endl;

strrho0 := apply(rho0, a->("point {{"|toString(a#Coordinates#0)|"},
    Multiplicity => "|toString(a#Multiplicity)|"}")); -- turn rho0 into a
    string that Macaulay2 can interpret later
out := openOut("../output/"|s|".o");
out << toString(strrho0); -- write rho0 to output file
close out;
stderr << s|".o WRITTEN" << endl;

quit();
```

B.5 Example input and output

B.5.1 v401.i

The file `v401.i` is one of the input files used by `combine.m2` to produce the approximate roots of g in Example 1.7. It is only one line long, and contains a list of roots with multiplicity.

```
{point { {1.51239824780835-7.92976795338518e-13*ii}, Multiplicity => 1},
    point { {1.55049553323509+.0668992571665445*ii}, Multiplicity => 1},
    point { {1.64351068573018+9.21082654592453e-13*ii}, Multiplicity => 1},
    point { {1.55049553323386-.0668992571760467*ii}, Multiplicity => 1}}
```

B.5.2 `v401.o`

The file `v401.o` is one of the output files generated by `combine.m2` in order to produce the approximate roots of g in Example 1.7. It is only one line long, and contains a list of roots with multiplicity.

```
{point {{1.56420013135082-.00422998423754113*ii}, Multiplicity => 1}, point
    {{1.56415375214466+.0126897469578536*ii}, Multiplicity => 1}, point
    {{1.56420013135082-.00422998423754057*ii}, Multiplicity => 1}, point
    {{1.56420013135082-.00422998423754328*ii}, Multiplicity => 1}}
```

B.5.3 `v401.i`

Three of the roots listed in `v401.o` are identical to within an acceptable tolerance, which means that those roots have combined. We rename the file `401.o` to `402.i` and edit it by hand so that those three roots, all of multiplicity one, are replaced by single root with a multiplicity of three.

```
{point {{1.56420013135082-.00422998423754113*ii}, Multiplicity => 3}, point
    {{1.56415375214466+.0126897469578536*ii}, Multiplicity => 1}}
```

Bibliography

[1] Mathcha editor. `https://www.mathcha.io/`. Accessed: 2020-07-16.

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

[3] Daniel J. Bates, Jonathan D. Hauenstein, and Andrew J. Sommese. Efficient path tracking methods. *Numerical Algorithms*, 58:451 – 459, 2011.

[4] Prashant Batra. Newton's method and the computational complexity of the fundamental theorem of algebra. *Electronic Notes in Theoretical Computer Science*, 202:201 – 218, 2008. Proceedings of the Fourth International Conference on Computability and Complexity in Analysis (CCA 2007).

[5] M. Gemignani. *Elementary Topology.* Dover, 1972.

[6] Daniel R. Grayson and Michael E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at `https://faculty.math.illinois.edu/Macaulay2/`.

[7] Z. Griffin and J. Hauenstein. Real solutions to systems of polynomial equations and parameter continuation. *Advances in Geometry*, 15(2):173–187, 2015.

[8] Elizabeth Gross, Jose Israel Rodriguez, Dan Bates, and Anton Leykin. Bertini: Interface to Bertini. Version 2.1.2.3. A *Macaulay2* package available at `https://github.com/Macaulay2/M2/tree/master/M2/Macaulay2/packages`.

[9] David Granville Hough. *Explaining and ameliorating the ill condition of zeros of polynomials.* PhD thesis, University of California, Berkeley, May 1977.

[10] W. Kahan. Conserving confluence curbs ill-condition. Technical Report 6, University of California, Berkeley, August 1972.

[11] MATLAB. *9.7.0.1261785 (R2019b).* The MathWorks Inc., Natick, Massachusetts, 2020.

[12] Zhonggang Zeng. The approximate gcd of inexact polynomials. 2004.