2018

Pseudo-Companion Matrices for Polynomial Systems

Melinda Kleczynski
Michigan Technological University, mkleczyn@mtu.edu

PSEUDO-COMPANION MATRICES FOR POLYNOMIAL SYSTEMS

By

Melinda Kleczynski

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Mathematical Sciences

MICHIGAN TECHNOLOGICAL UNIVERSITY

2018

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Mathematical Sciences.

Department of Mathematical Sciences

Thesis Advisor:	*Dr. Allan A. Struthers*
Committee Member:	*Dr. Benjamin W. Ong*
Committee Member:	*Dr. Cécile M. Piret*
Department Chair:	*Dr. Mark S. Gockenbach*

Contents

List of Figures

List of Tables

Abstract

Roots of a scalar polynomial in one variable are frequently found by computing the eigenvalues of the standard companion matrix. In this exploratory work, we introduce the pseudo-companion matrix for finding roots of multivariable polynomial systems. In some cases, a perturbation of the polynomial system is used for the matrix construction, yielding approximate roots of the original polynomial system. The coordinates of the roots, or their approximations, are obtained from the eigenvectors of this matrix. In this thesis, we describe the process of constructing the pseudo-companion matrix and computing the polynomial roots using illustrative examples.

1 Introduction

Numerical algorithms have previously been developed to find polynomial roots by determining the eigenvalues of a companion matrix. This is, for example, the strategy employed by the MATLAB roots function [10]. Currently companion matrices are only used for finding roots of polynomials of one variable, not for finding roots of multivariable polynomial systems.

The problem of finding a companion matrix can be framed as an inverse problem. The direct problem is "given a matrix, find its characteristic polynomial." The inverse problem is "given a polynomial, find a matrix which has that polynomial as its characteristic polynomial." The companion matrix is the solution to this inverse problem [22]. There is more than one type of companion matrix [5]; we will not discuss all of them.

The term companion matrix was introduced as a translation of the German term Begleitmatrix [13, p. 20]. The following established companion matrix addresses the case of one polynomial in one variable. Without loss of generality, we can assume the coefficient of the highest degree term is 1. For the polynomial $p : \mathbb{C} \to \mathbb{C}$ defined by $p(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{n-1} x^{n-1} + x^n$, an $n \times n$ companion matrix is given by

$$A = \begin{pmatrix} 0 & 1 & 0 & ... & 0 & 0 \\ 0 & 0 & 1 & ... & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & ... & 0 & 1 \\ -a_0 & -a_1 & -a_2 & ... & -a_{n-2} & -a_{n-1} \end{pmatrix}$$

where for each distinct root λ of the polynomial, there is an eigenvalue λ and corresponding eigenvector of the form $< 1, \lambda, \lambda^2, ..., \lambda^{n-1} >$ [4].

Consider the polynomial system $\vec{p} : \mathbb{C}^n \to \mathbb{C}^n$ where $\vec{p} =< p_1, p_2, ..., p_n >$ and $p_1, p_2, ..., p_n$ are polynomials of $x_1, x_2, ..., x_n$. The goal is to construct a pseudo-companion matrix such that the eigenvectors of the matrix will include information about the roots of the polynomial system or approximations of the roots. If a set of points, some of which are approximate solutions, has been found, methods exist for identifying those which will converge via Newton's method to the roots of the system of polynomials [9]. Using Newton's method to refine results has precedent in other polynomial solvers [3].

Attention is restricted in this thesis to polynomial systems which have some solution to $\vec{p} = \vec{0}$, and whose solutions are given by isolated points. The number of such solutions may be up to the product of the degrees of the individual polynomials making up the system [17, p. 228]. This type of solution set is referred to as zero-dimensional (as opposed to a positive-dimensional solution set) [3]. Existing methods utilizing matrices to solve multivariable polynomial systems include those based on ring representations [22] and resultants [14].

A standard automated method for finding roots of a polynomial system is polynomial homotopy continuation. Available implementations include Bertini [3] and PHCpack [20]. For MATLAB users, BertiniLab is an interface for using MATLAB to run Bertini [1]. Similarly, PHClab can be used to run PHCpack [7]. We frequently compare the results of our algorithm to the results obtained using Bertini$^{\text{TM}}$ v1.5.1; we use Bertini for comparison because it is readily available for use with multiple operating systems [2]. The materials accompanying PHCpack include a collection of polynomial test problems, some of which are used as examples throughout this work [20, 21]. Mathematica added polynomial homotopy continuation capability as

of Mathematica 10, improving speed for certain systems. Previously Mathematica used a Gröbner basis method [23].

The general concept of polynomial homotopy continuation is as follows. Suppose we would like to solve the polynomial system $\vec{f}(\vec{x}) = \vec{0}$. Create a new polynomial system $\vec{g}(\vec{x})$ which is similar to $\vec{f}(\vec{x})$ where you know or can readily find the solutions of $\vec{g}(\vec{x}) = \vec{0}$. Deform $\vec{g}(\vec{x})$ to $\vec{f}(\vec{x})$ and track the solutions. Parallelization can be easily utilized [3].

The paths taken by the solutions may be primarily non-real, but both endpoints may be real [3]. This serves as a barrier to attempting to only compute the real solutions (or some other desirable subset of the solutions).

2 Notation and Definitions

- Let n be the number of variables and the number of polynomials; by assumption these are equal.

- Consider $\vec{x} =< x_1, x_2, ..., x_n > \in \mathbb{C}^n$. Define the **monomial** $\vec{x}^{\vec{\alpha}}$ to be $x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot ... \cdot x_n^{\alpha_n}$ where $\vec{\alpha} =< \alpha_1, \alpha_2, ..., \alpha_n >$ and $\alpha_1, \alpha_2, ..., \alpha_n$ are nonnegative integers [17, p. 4].

- $|\vec{\alpha}| = \alpha_1 + \alpha_2 + ... + \alpha_n$ is the **degree** [17, p. 4]. The degree of a polynomial is the largest degree of any of its monomials.

- Let B be a basis of monomials. A basis B of monomials is called **closed** if $\vec{x}^{\vec{\alpha}} \in B \Rightarrow \vec{x}^{\vec{\alpha}'} \in B$ for all $\vec{\alpha}'$ such that $\vec{x}^{\vec{\alpha}'}$ divides $\vec{x}^{\vec{\alpha}}$ [17, p. 55].

- Basis elements are written using the standard mathematical font, for example x. Coordinates of the roots are written using blackboard bold, for example $\mathbb{x}$.

- The **border set** S corresponding to a closed basis B consists of the monomials which are not contained in B but which can be produced as the product of an element of B and a degree one monomial [17, p. 58].

- A **replacement monomial** is a univariate monomial which is a member of the border set and which is replaced by a linear combination of basis elements using the original or modified polynomial system.

- Multiplication by a degree one monomial x_i will be denoted $\boldsymbol{\vec{x}^{\vec{\alpha}}} \xrightarrow{*\boldsymbol{x_i}} \boldsymbol{\vec{x}^{\vec{\beta}}}$ where $\vec{x}^{\vec{\alpha}} \in B$ and either $\vec{x}^{\vec{\beta}} \in B$ or $\vec{x}^{\vec{\beta}} \in S$.

- $\boldsymbol{M_{x_i}}$ denotes the matrix for multiplication by x_i.

- In the case of two variables x_1 and x_2, the monomials can be illustrated graphically. The origin represents $x_1^0 x_2^0 = 1$. Moving to the right corresponds to increasing the power of x_1 and moving up corresponds to increasing the power of x_2 [17, p. 56]. Suppose, for example, that a basis B is given by $\{1,\ x_1,\ x_2,\ x_1^2,\ x_1x_2,\ x_2^2,\ x_1x_2^2\}$. The border set S is $\{x_1^3,\ x_1^2x_2,\ x_2^3,\ x_1^2x_2^2,\ x_1x_2^3\}$. These are depicted in Figure 2.1.

- A **rectangular basis with bound** $\boldsymbol{< d_1, d_2, ..., d_n >}$ is $\{\vec{x}^{\vec{\alpha}} \,|\, \alpha_i \leq d_i \text{ for } i = 1, 2, ..., n\}$.

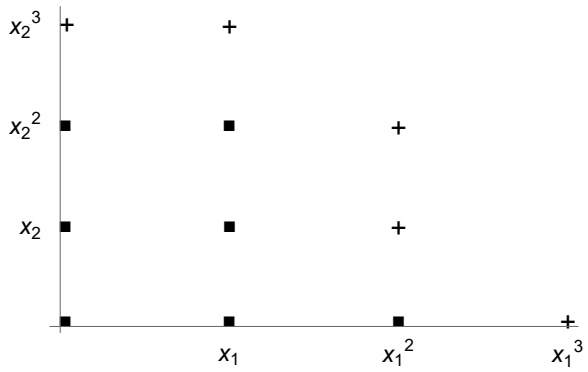

Figure 2.1: Example of a 2 variable basis (■) and the corresponding border set (+).

- Monomials are listed in order of ascending degree. Within monomials of equal degree, monomials are then ordered based on variable name, for example

$$\{..., x_1^3, x_1^2x_2, x_1^2x_3, ...\}$$

This is referred to as **degree lexicographic order** [18]. Note that this order is only used for the purpose of consistently listing the monomials in a given basis. No preference is given to any of the variables in the computations.

- If a perturbation is introduced, then the original polynomial system will be denoted $\vec{p} =< p_1, p_2, ..., p_n >$ and the new, modified polynomial system will be denoted $\boldsymbol{\vec{q} =< q_1, q_2, ..., q_n >}$.

- A term added to a polynomial is called a **perturbation term** and takes the form $\epsilon\, x_i^{d_i+1}$ where ϵ is small and d_i is the degree of p_i.

- The **standard companion matrix** refers to the one variable companion matrix described in Section 1.

- Throughout, $\boldsymbol{\mu}$ is defined as $\frac{1}{\epsilon}$.

3 Objective

We intend to construct pseudo-companion matrices to find the roots of polynomial systems. The Central Theorem, described by Stetter, shows that for a suitable basis and multiplication matrices, roots of a zero-dimensional polynomial system can be determined from the eigenvectors [17, p. 52]. We would like a simple algorithm to construct a pseudo-companion matrix without complicated polynomial computations, such as a Gröbner basis type computation.

The most novel aspect of our method is the modification of the polynomial system in some cases with the inclusion of additional terms to allow for the matrix construction. These additional terms will have a coefficient of ϵ which is taken to be small. Computationally, the system is not monitored as ϵ approaches zero. Rather, ϵ is set to some small number to find approximate potential roots, which are then refined using Newton's method.

The matrix is constructed so that for each root $\vec{x} = (x_1, x_2, ..., x_n)$ of the polynomial system, there is an eigenvector which is equal to a scalar multiple of $< 1, x_1, x_2, ..., x_n, ... >$. The details of the eigensystem computations are not the focus of this investigation, and in the following examples are performed using standard routines in Mathematica. After the eigenvectors have been determined, normalizing by the first element of the eigenvector produces the desired form. The coordinates of the root are found in elements 2 through $n + 1$ of the eigenvector. Subsequent elements of the eigenvector correspond to higher order monomials in the basis.

4 Matrix Construction

4.1 Standard Companion Matrix

For the standard companion matrix, the ith column was associated with λ^{i-1} where $x = \lambda$ is a root of the polynomial $p(x)$. The standard companion matrix applies to one polynomial in one variable, so here p and x are both scalar valued. In this manner, associate the rows and the columns of the matrix with increasing powers of x as shown:

$$\begin{array}{c} \\ 1 \\ x \\ \\ x^{n-2} \\ x^{n-1} \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & x & x^2 & & x^{n-2} & x^{n-1} \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & ... & 0 & 0 \\ 0 & 0 & 1 & ... & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & ... & 0 & 1 \\ -a_0 & -a_1 & -a_2 & ... & -a_{n-2} & -a_{n-1} \end{pmatrix} \end{array}$$

$\{1, x, ..., x^{n-2}, x^{n-1}\}$ forms a basis of monomials. The basis should be closed. In other words, every power of x, from 0 to $n-1$, should be included in the basis even if one of the corresponding coefficients in the polynomial happens to be zero. The matrix embodies multiplication by x. For example, in the first row, the element 1 is multiplied by x to obtain x. In the final row, x^{n-1} is multiplied by x to obtain x^n, which is not a basis element itself but can be expressed in terms of the other basis elements using $p(x) = a_0 + a_1x + a_2x^2 + ... + a_{n-1}x^{n-1} + x^n = 0$ when x is a root. After multiplication by x, every element of the basis can still be represented using

elements of the basis, either directly or using $x^n = -a_0 - a_1 x - a_2 x^2 - ... - a_{n-1} x^{n-1}$.

4.2 Pseudo-Companion Matrix

This construction becomes more difficult in the multivariable case. There is more than one variable by which the basis elements could be multiplied. Additionally, it can no longer be assumed that there is a single highest degree term in each polynomial.

The first step of building the pseudo-companion matrix is to obtain polynomials of the appropriate form. We will call such polynomials **pre-companion polynomials**. Every scalar polynomial in one variable is a pre-companion polynomial, so no such distinction is necessary for the standard companion matrix. In some polynomial systems, some or all of the polynomials will already be pre-companion polynomials. We begin by successively identifying these pre-companion polynomials, which must satisfy the following:

1. The polynomial contains a univariate term whose degree is strictly greater than the degrees of the other terms in that polynomial

and

2. The variable in the univariate highest degree term did not already appear in the univariate highest degree term of any previously identified pre-companion polynomial

In other words, a pre-companion polynomial has the form

$$p_i = \mathbb{p}_i + x_i^{d_i+1}$$

where $d_i + 1$ is strictly greater than the degree of $\mathbb{p}_i$. Reordering the polynomials and/or renaming the variables allows x_i to be the variable appearing in the univariate

highest degree term of p_i. Since our objective is only to find the roots, we can assume without loss of generality that the coefficient of $x_i^{d_i+1}$ is 1.

If p_k is any polynomial which is not already a pre-companion polynomial, then a perturbation term $\epsilon\, x_k^{d_k+1}$ is added to that polynomial, where ϵ is small and d_k is the degree of p_k. With these added terms, we obtain a new polynomial system $\vec{q}$ in which each polynomial is now a pre-companion polynomial. The following illustrates the form a system of four polynomials in four variables would take if two of the polynomials started out as pre-companion polynomials with x_1 and x_2 appearing in the univariate terms:

$$\begin{aligned} q_1 &= p_1 = \mathbb{p}_1 + x_1^{d_1+1} \\ q_2 &= p_2 = \mathbb{p}_2 + x_2^{d_2+1} \\ q_3 &= p_3 + \epsilon\, x_3^{d_3+1} \\ q_4 &= p_4 + \epsilon\, x_4^{d_4+1} \end{aligned}$$

Here degree($\mathbb{p}_1$) $\leq d_1$, degree($\mathbb{p}_2$) $\leq d_2$, degree(p_3) $= d_3$, and degree(p_4) $= d_4$.

Continue using the system of four polynomials as an example. We will set $\vec{q} = \vec{0}$ to build the pseudo-companion matrix. Then we have

$$\begin{aligned} x_1^{d_1+1} &= -\mathbb{p}_1 \\ x_2^{d_2+1} &= -\mathbb{p}_2 \\ x_3^{d_3+1} &= -\mu p_3 \\ x_4^{d_4+1} &= -\mu p_4 \end{aligned}$$

where $\mu = \frac{1}{\epsilon}$. We will call $x_1^{d_1+1}$, $x_2^{d_2+1}$, $x_3^{d_3+1}$, and $x_4^{d_4+1}$ replacement monomials because we will replace them with $-\mathbb{p}_1$, $-\mathbb{p}_2$, $-\mu p_3$, and $-\mu p_4$, respectively.

For i from 1 to n, we will represent multiplication by x_i as a matrix M_{x_i}. The basis B will be monomials of the form $x_1^{\alpha_1}x_2^{\alpha_2}x_3^{\alpha_3}x_4^{\alpha_4}$. The border set S consists of the monomials which are not contained in B but which can be produced as the product of an element of B and one of the x_i. The key is that we want a square matrix. So once we choose a basis, every element of the border set must be expressible as a linear combination of basis elements. This can be accomplished using our replacements as long as we pick an appropriate basis. The basis should be closed (if a monomial is contained in the basis, then so are all of its divisors). Elements of the border set must have the individual degree of some x_i be at least $d_i + 1$ so that a replacement can be made. The requirements can be met by taking $B = \{\vec{x}^{\vec{\alpha}} \,|\, \alpha_i \leq d_i \text{ for } i = 1, 2, ..., n\}$, which we will refer to as a rectangular basis with bound $< d_1, d_2, ..., d_n >$ (in this example, $n = 4$). We call it a rectangular basis because it would look rectangular if plotted as in Figure 2.1.

After multiplying the basis elements by a given variable x_i, some of the results will be elements of the border set. This will occur for basis elements whose degree of x_i is d_i; then multiplication by x_i increases the degree to $d_i + 1$. Then a replacement will be used. The replacement will decrease the degree of x_i, but it could do so at the expense of increasing the degree of some other variable x_j. If the resulting degree of x_j is larger than d_j, then the result can not be expressed using only basis elements, but a second replacement could be performed. This process must terminate, since the replacements are decreasing in degree.

M_{x_i} is the matrix for multiplication by x_i, and the equations generating the replacements used to construct M_{x_i} are satisfied at the roots of the modified polynomial system $\vec{q}$. Randomly generate coefficients c_i; these can be real or complex. The pseudo-companion matrix is $A = \sum_{i=1}^{n} c_i M_{x_i}$. Let $\vec{v}$ be a vector whose elements are the basis monomials evaluated at the roots of $\vec{q}$, and let $\mathbb{x}_\mathbb{i}$ be the x_i coordinate of

that root. Then

$$A\vec{v} = \left(\sum_{i=1}^{n} c_i M_{x_i}\right)\vec{v} = \sum_{i=1}^{n} c_i \left(M_{x_i}\vec{v}\right) = \sum_{i=1}^{n} c_i \left(\mathbb{x}_i\vec{v}\right) = \left(\sum_{i=1}^{n} c_i \mathbb{x}_i\right)\vec{v}$$

We see that $\vec{v}$ is an eigenvector of the pseudo-companion matrix. Since any scalar multiple of an eigenvector is also an eigenvector, we need a way to scale $\vec{v}$ and obtain the root. Using degree lexicographic order, the first basis element is 1, so we scale the eigenvector so the first element of the eigenvector is 1. Then the values of the degree one monomials give the coordinates of the root of $\vec{q}$. If any eigenvalue has an eigenspace with dimension greater than 1, then there is no expectation that the eigenvectors returned by a numerical computation have the required structure. The purpose of the randomization is to avoid repeated eigenvalues. For problems where a coordinate of a root never takes the same value in two distinct roots, a single multiplication matrix could be used. Problems of interest may have some structure such that this is not true, so in general using a random linear combination is recommended.

4.3 Merit Functions

We have discussed replacement monomials which have degree strictly larger than the degrees of the other terms in the respective polynomials. Then when a replacement is made, the degree decreases. Since the degree is strictly decreasing as replacements are made, eventually the border set elements can be expressed using only the basis elements. This can be generalized by replacing the degree with any merit function, as long as the merit function decreases at each iteration and a sufficiently small merit function guarantees that you have reached the basis elements. The key is that the replacement process must successfully terminate.

5 Examples

5.1 Intersection of Two Curves

Determine the points of intersection of the two curves given by $-1 + 2y + x^2 = 0$ and $4 - 5x + 6y + y^2 = 0$, shown in Figure 5.1. The system of polynomials has four solutions, two of which correspond to the real points of intersection of the curves. These solutions, found using a standard solver, are

$$\begin{aligned} x_1 &= -2.52369 - 1.16200i, & y_1 &= -2.00939 - 2.93254i \\ x_2 &= -2.52369 + 1.16200i, & y_2 &= -2.00939 + 2.93254i \\ x_3 &= 0.907464, & y_3 &= 0.0882549 \\ x_4 &= 4.13992, & y_4 &= -8.06948 \end{aligned}$$

We will now use a pseudo-companion matrix to solve the polynomial system.

Each individual polynomial has a highest degree term. The highest degree terms are univariate, with a different variable in each of these terms. Using the two polynomials, we can use replacements $x^2 \to 1 - 2y$ and $y^2 \to -4 + 5x - 6y$. Then x^2 and y^2 can be part of the border set. Any monomial for which the degree of each variable is less than 2 must be included in the basis, so the basis elements are $1,\ x,\ y$, and xy. The monomials which are not included in the basis, but which can be obtained by multiplying a basis element by x or y, are $x^2,\ y^2,\ x^2y$, and xy^2. This is the border set. The basis and border set are shown graphically in Figure 5.2.

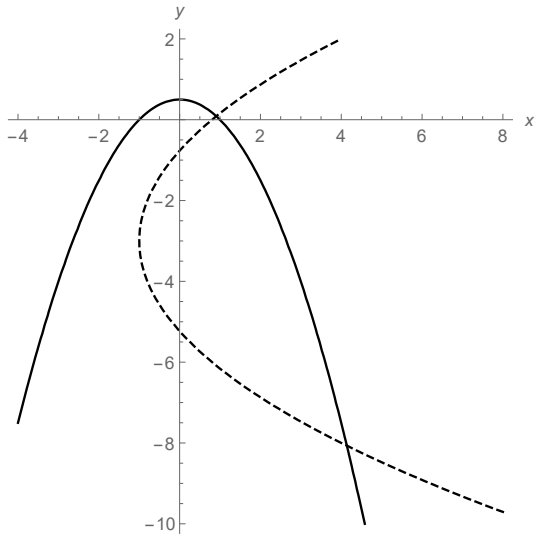

Figure 5.1: The curves $-1 + 2y + x^2 = 0$ (solid) and $4 - 5x + 6y + y^2 = 0$ (dashed).

Multiplying all elements of the basis by x yields the following:

$$1 \xrightarrow{*x} x$$

$$x \xrightarrow{*x} x^2$$

$$y \xrightarrow{*x} xy$$

$$xy \xrightarrow{*x} x^2y$$

Using the replacements, we have $x^2 = 1 - 2y$ and $x^2y = (1 - 2y)y = y - 2y^2 = y - 2(-4 + 5x - 6y) = y + 8 - 10x + 12y = 8 - 10x + 13y$. The updated multiplication results are:

$$1 \xrightarrow{*x} x$$

$$x \xrightarrow{*x} 1 - 2y$$

$$y \xrightarrow{*x} xy$$

$$xy \xrightarrow{*x} 8 - 10x + 13y$$

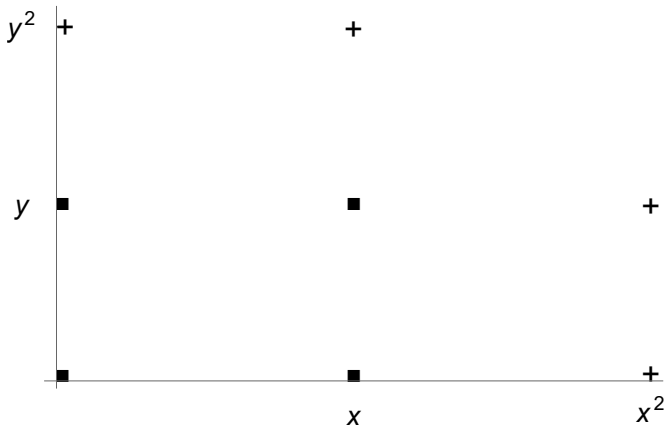

Figure 5.2: Basis (■) and border set (+) corresponding to the polynomial system $-1 + 2y + x^2 = 0$ and $4 - 5x + 6y + y^2 = 0$.

Having obtained the results of multiplying all basis elements by x, we can encode these results in a multiplication matrix for x by associating each row and column of the matrix with a basis element. This construction relies on the fact that we were able to use the replacements to express border set elements in terms of the basis elements. The multiplication matrix for x is

$$M_x = \begin{array}{c} \\ 1 \\ x \\ y \\ xy \end{array} \begin{array}{c} \begin{array}{cccc} 1 & x & y & xy \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 \\ 8 & -10 & 13 & 0 \end{pmatrix} \end{array}$$

Now how can we obtain the roots of the original polynomial system? Suppose that $(\mathrm{x},\ \mathrm{y})$ is a root. Then $\mathrm{x}^2 = 1 - 2\mathrm{y}$ and $\mathrm{y}^2 = -4 + 5\mathrm{x} - 6\mathrm{y}$. Let $\vec{v} =< 1, \mathrm{x}, \mathrm{y}, \mathrm{xy} >$.

Then

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 \\ 8 & -10 & 13 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ \mathrm{x} \\ \mathrm{y} \\ \mathrm{xy} \end{pmatrix} = \begin{pmatrix} \mathrm{x} \\ 1 - 2\mathrm{y} \\ \mathrm{xy} \\ 8 - 10\mathrm{x} + 13\mathrm{y} \end{pmatrix} = \begin{pmatrix} \mathrm{x} \\ \mathrm{x}^2 \\ \mathrm{xy} \\ \mathrm{x}^2\mathrm{y} \end{pmatrix} = \mathrm{x} \begin{pmatrix} 1 \\ \mathrm{x} \\ \mathrm{y} \\ \mathrm{xy} \end{pmatrix}$$

In other words $M_x\vec{v} = \mathrm{x}\vec{v}$. So x is an eigenvalue of M_x and $\vec{v}$ is the corresponding eigenvector. This can be verified numerically by finding the eigenvalues and eigenvectors of M_x. The eigenvalues are $-2.52369 - 1.16200i,\ -2.52369 + 1.16200i,\ 0.907464$, and 4.13992. As expected, these are the x coordinates of the roots. The eigenvectors should be scaled so the first element is 1. Then the elements of the eigenvectors will be the basis elements evaluated at each of the roots. The scaled eigenvectors are:

$$< 1,\ -2.52369 - 1.16200i,\ -2.00939 - 2.93254i,\ 1.66346 + 9.73574i >$$

$$< 1,\ -2.52369 + 1.16200i,\ -2.00939 + 2.93254i,\ 1.66346 - 9.73574i >$$

$$< 1,\ 0.907464,\ 0.0882549,\ 0.0800881 >$$

$$< 1,\ 4.13992,\ -8.06948,\ -33.4070 >$$

The elements are as expected. Since the eigenvalues contain only the x coordinates of the roots, the roots will be obtained from the eigenvectors instead. Simply take the second and third elements of each eigenvector.

This structure is not unique to the M_x matrix. M_y, the multiplication matrix for y, is constructed in the same manner and has corresponding properties.

Multiply all elements of the basis by y:

$$1 \xrightarrow{*y} y$$
$$x \xrightarrow{*y} xy$$
$$y \xrightarrow{*y} y^2$$
$$xy \xrightarrow{*y} xy^2$$

Using the same replacements as before, we have $y^2 = -4 + 5x - 6y$ and $xy^2 = x(-4+5x-6y) = -4x+5x^2-6xy = -4x+5(1-2y)-6xy = -4x+5-10y-6xy = 5-4x-10y-6xy$. The updated multiplication results are:

$$1 \xrightarrow{*y} y$$
$$x \xrightarrow{*y} xy$$
$$y \xrightarrow{*y} -4 + 5x - 6y$$
$$xy \xrightarrow{*y} 5 - 4x - 10y - 6xy$$

Writing these in matrix form, we have:

$$M_y = \begin{array}{c} \\ 1 \\ x \\ y \\ xy \end{array} \begin{array}{c} \begin{array}{cccc} 1 & x & y & xy \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -4 & 5 & -6 & 0 \\ 5 & -4 & -10 & -6 \end{pmatrix} \end{array}$$

The eigenvalues of this matrix are -8.06948, $-2.00939 + 2.93254i$, $-2.00939 - 2.93254i$, and 0.0882549. These are the y coordinates of the roots. The scaled eigen-

vectors are:

$$< 1,\ 4.13992,\ -8.06948,\ -33.4070 >$$

$$< 1,\ -2.52369 + 1.16200i,\ -2.00939 + 2.93254i,\ 1.66346 - 9.73574i >$$

$$< 1,\ -2.52369 - 1.16200i,\ -2.00939 - 2.93254i,\ 1.66346 + 9.73574i >$$

$$< 1,\ 0.907464,\ 0.0882549,\ 0.0800881 >$$

Note that M_x and M_y have the same eigenvectors, so the roots could be obtained from either. In this example, both M_x and M_y are pseudo-companion matrices. In later examples we will see that it is sometimes necessary to use a randomized linear combination of the multiplication matrices as the pseudo-companion matrix.

5.2 Intersection of Two Curves (Real Solutions)

Typically the roots will be obtained from the eigenvectors of the pseudo-companion matrix. In limited situations, the roots can be obtained from the eigenvalues. We will discuss an example of such a situation here, before proceeding by using the eigenvectors in subsequent examples.

Determine the points of intersection of the two curves given by $5 + x - y^2 = 0$ and $-6 - xy + x^2 = 0$, shown in Figure 5.3. The system of polynomials has four solutions, all of which correspond to real points of intersection of the curves. These solutions

are

$$x_1 = -3.20850, \qquad y_1 = -1.33847$$

$$x_2 = -1.70459, \qquad y_2 = 1.81533$$

$$x_3 = 1.48726, \qquad y_3 = -2.54701$$

$$x_4 = 4.42583, \qquad y_4 = 3.07015$$

We intend to find these solutions using a pseudo-companion matrix.

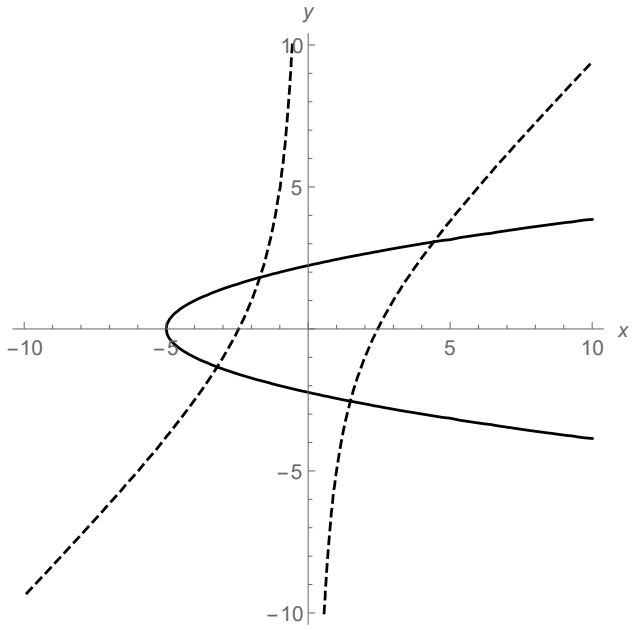

Figure 5.3: The curves $5 + x - y^2 = 0$ (solid) and $-6 - xy + x^2 = 0$ (dashed).

There is no longer a single highest degree term in each polynomial, because the second polynomial has two terms of degree two. A pseudo-companion matrix can still be constructed without modifying the original polynomial system.

In the first polynomial, y^2 is a natural choice for a replacement monomial. In the second polynomial, we will take the univariate term of degree two, namely x^2, to be the replacement monomial. Then the replacements are $y^2 \to 5 + x$ and $x^2 \to 6 + xy$.

We must include xy in the basis. To obtain a closed basis including xy, take $1,\ x,\ y,$ and xy as the basis elements. This basis will be sufficient for the construction of the multiplication matrices.

Multiplying all the basis elements by x gives the following:

$$\begin{aligned} 1 &\xrightarrow{*x} x \\ x &\xrightarrow{*x} x^2 \\ y &\xrightarrow{*x} xy \\ xy &\xrightarrow{*x} x^2y \end{aligned}$$

Using the replacements, $x^2 = 6+xy$ and $x^2y = (6+xy)y = 6y+xy^2 = 6y+x(5+x) = 6y + 5x + x^2 = 6y + 5x + 6 + xy = 6 + 5x + 6y + xy$. After the replacements, the multiplication results are:

$$\begin{aligned} 1 &\xrightarrow{*x} x \\ x &\xrightarrow{*x} 6 + xy \\ y &\xrightarrow{*x} xy \\ xy &\xrightarrow{*x} 6 + 5x + 6y + xy \end{aligned}$$

The multiplication matrix for x is

$$M_x = \begin{array}{c} \\ 1 \\ x \\ y \\ xy \end{array} \begin{array}{c} \begin{array}{cccc} 1 & x & y & xy \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 6 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 6 & 5 & 6 & 1 \end{pmatrix} \end{array}$$

Repeat this process for multiplication by y.

$$1 \xrightarrow{*y} y$$
$$x \xrightarrow{*y} xy$$
$$y \xrightarrow{*y} y^2$$
$$xy \xrightarrow{*y} xy^2$$

$y^2 = 5 + x$ and $xy^2 = x(5 + x) = 5x + x^2 = 5x + 6 + xy = 6 + 5x + xy$.

$$1 \xrightarrow{*y} y$$
$$x \xrightarrow{*y} xy$$
$$y \xrightarrow{*y} 5 + x$$
$$xy \xrightarrow{*y} 6 + 5x + xy$$

Then the multiplication matrix for y is:

$$M_y = \begin{array}{c} \\ 1 \\ x \\ y \\ xy \end{array} \begin{array}{c} \begin{array}{cccc} 1 & x & y & xy \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 5 & 1 & 0 & 0 \\ 6 & 5 & 0 & 1 \end{pmatrix} \end{array}$$

For a single polynomial, the roots are given by the eigenvalues of the standard companion matrix. For most polynomial systems, the eigenvectors of the pseudo-companion matrix must be used to find the roots. A multivariable solution simply cannot be contained in a single eigenvalue. The current problem is an example of a situation in which the eigenvalues of the pseudo-companion matrix are sufficient to

give the roots of a polynomial system. The key is that there are only two variables and all the solutions are real. Then the pseudo-companion matrix can be constructed so that the x and y coordinates of the roots are the real and imaginary parts of the eigenvalues. This can be accomplished by taking the pseudo-companion matrix to be equal to the linear combination $M_x + iM_y$.

For example, for the current problem, the eigenvalues of $M_x + iM_y$ are $-3.20850 - 1.33847i$, $-1.70459 + 1.81533i$, $1.48726 - 2.54701i$, and $4.42583 + 3.07015i$. The real and imaginary parts are the coordinates of the roots, as desired.

5.3 Multivariable Replacement Monomials

Consider the construction of the multiplication matrix for the variable x_i. Let r be the maximum integer such that x_i^r is an element of the basis. Then x_i^{r+1} is an element of the border set. This is expressible as a linear combination of basis elements if x_i^{r+1} is one of the replacement monomials. In general we construct all the multiplication matrices and the replacement monomials are univariate.

This example illustrates that exceptions to these guidelines are possible. The following polynomial system was created for use in this example. In general, polynomial systems of interest did not have this structure, so we chose to note this construction in an example but not pursue it further.

Consider the polynomial system

$$x^2 + y + z - 7 = 0$$
$$xy - 2z - 5 = 0$$
$$xz - x + 2 = 0$$

which has solutions

$$
\begin{array}{lll}
x_1 = 0.369651 - 0.719866i, & y_1 = 7.51053 + 2.73077i, & z_1 = -0.128967 - 2.19857i \\
x_2 = 0.369651 + 0.719866i, & y_2 = 7.51053 - 2.73077i, & z_2 = -0.128967 + 2.19857i \\
x_3 = 2.12933, & y_3 = 2.40520, & z_3 = 0.0607387 \\
x_4 = -2.86864, & y_4 = -2.92627, & z_4 = 1.69720
\end{array}
$$

Each polynomial has a highest degree term, which we will use as the replacement monomial, yielding replacements $x^2 \to -y - z + 7$, $xy \to 2z + 5$, and $xz \to x - 2$. It seems that we must include 1, x, y, and z in our basis. If these are our only replacements, we can not construct multiplication matrices for y or z. If we consider only multiplication by x, then the elements in the border set are x^2, xy, and xz. These are the same as the replacement monomials, so all the border set elements can be expressed in terms of the basis elements. The multiplication matrix for x is

$$
M_x = \begin{array}{c} \\ 1 \\ x \\ y \\ z \end{array}
\begin{array}{c}
\begin{array}{cccc} 1 & x & y & z \end{array} \\
\left(\begin{array}{cccc}
0 & 1 & 0 & 0 \\
7 & 0 & -1 & -1 \\
5 & 0 & 0 & 2 \\
-2 & 1 & 0 & 0
\end{array} \right)
\end{array}
$$

The scaled eigenvectors of this matrix are

$$< 1,\ 0.369651 - 0.719866i,\ 7.51053 + 2.73077i,\ -0.128967 - 2.19857i >$$

$$< 1,\ 0.369651 + 0.719866i,\ 7.51053 - 2.73077i,\ -0.128967 + 2.19857i >$$

$$< 1,\ 2.12933,\ 2.40520,\ 0.0607387 >$$

$$< 1,\ -2.86864,\ -2.92627,\ 1.69720 >$$

Elements 2 through 4 of each eigenvector give the coordinates of the roots.

5.4 PHCpack Test Problem "Rediff3"

The following polynomial system is from the set of test problems accompanying PHCpack [21].

$$\begin{aligned} -2x_1 + x_2 + \alpha x_1(1 - x_1) &= 0 \\ x_1 - 2x_2 + x_3 + \alpha x_2(1 - x_2) &= 0 \\ x_2 - 2x_3 + \alpha x_3(1 - x_3) &= 0 \end{aligned}$$

In the test problem $\alpha = 0.835634534$. In general α is a parameter which could take some other positive value. For this problem we will build the pseudo-companion matrix using an unassigned parameter.

Rewrite this as

$$\begin{aligned} -\alpha x_1^2 + (\alpha - 2)x_1 + x_2 &= 0 \\ -\alpha x_2^2 + x_1 + (\alpha - 2)x_2 + x_3 &= 0 \\ -\alpha x_3^2 + x_2 + (\alpha - 2)x_3 &= 0 \end{aligned}$$

Then we have replacements

$$
\begin{aligned}
x_1^2 &\to \frac{(\alpha-2)}{\alpha}x_1 + \frac{1}{\alpha}x_2 \\
x_2^2 &\to \frac{1}{\alpha}x_1 + \frac{(\alpha-2)}{\alpha}x_2 + \frac{1}{\alpha}x_3 \\
x_3^2 &\to \frac{1}{\alpha}x_2 + \frac{(\alpha-2)}{\alpha}x_3
\end{aligned}
$$

The monomials in the border set should have the degree of one of the variables equal to 2 so a replacement can be used. Then the basis elements consist of monomials for which the individual degree of each variable is either zero or one. The number of basis elements is then $2^3 = 8$. Using this basis and these replacements we get the following multiplication matrices M_{x_1}, M_{x_2}, and M_{x_3}:

$$
M_{x_1} = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1-\frac{2}{\alpha} & \frac{1}{\alpha} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & \frac{1}{\alpha^2} & \frac{1}{\alpha}-\frac{2}{\alpha^2} & \frac{1}{\alpha^2} & 1-\frac{2}{\alpha} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1-\frac{2}{\alpha} & \frac{1}{\alpha} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & \frac{1}{\alpha^3} & \frac{1}{\alpha^2}-\frac{2}{\alpha^3} & 0 & \frac{1}{\alpha^2} & \frac{1}{\alpha}-\frac{2}{\alpha^2} & 1-\frac{2}{\alpha}
\end{pmatrix}
$$

$$M_{x_2} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\alpha} & 1-\frac{2}{\alpha} & \frac{1}{\alpha} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\alpha}-\frac{2}{\alpha^2} & \frac{1}{\alpha^2} & 0 & 1-\frac{2}{\alpha} & \frac{1}{\alpha} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{\alpha^2} & \frac{1}{\alpha}-\frac{2}{\alpha^2} & 0 & \frac{1}{\alpha} & 1-\frac{2}{\alpha} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\alpha^2} & \frac{2}{\alpha}-\frac{4}{\alpha^2} & \frac{1}{\alpha^2} & 1-\frac{2}{\alpha} \end{pmatrix}$$

$$M_{x_3} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{\alpha} & 1-\frac{2}{\alpha} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \frac{1}{\alpha} & 1-\frac{2}{\alpha} & 0 & 0 \\ 0 & \frac{1}{\alpha^2} & \frac{1}{\alpha}-\frac{2}{\alpha^2} & \frac{1}{\alpha^2} & 0 & 0 & 1-\frac{2}{\alpha} & 0 \\ 0 & \frac{1}{\alpha^2}-\frac{2}{\alpha^3} & \frac{1}{\alpha^3} & 0 & \frac{1}{\alpha}-\frac{2}{\alpha^2} & \frac{1}{\alpha^2} & 0 & 1-\frac{2}{\alpha} \end{pmatrix}$$

The first column of each of these matrices contains all zeros because no constant term appears in the polynomial system. This will result in a zero eigenvalue, which corresponds to the trivial solution of the system.

We can generate randomized coefficients to make a linear combination of the three multiplication matrices. The pseudo-companion matrix can be constructed without assigning a specific value of α. This value should be given prior to finding the eigenvectors. Obtaining the randomized coefficients and setting $\alpha = 0.835634534$,

we have the pseudo-companion matrix

$$\begin{pmatrix}
0 & 0.398795 & 0.123007 & 0.504388 & 0 & 0 & 0 & 0 \\
0 & -0.555677 & 0.477236 & 0 & 0.123007 & 0.504388 & 0 & 0 \\
0 & 0.147202 & -0.171396 & 0.147202 & 0.398795 & 0 & 0.504388 & 0 \\
0 & 0 & 0.603599 & -0.702809 & 0 & 0.398795 & 0.123007 & 0 \\
0 & 0.365997 & -0.488821 & 0.571106 & -0.727073 & 0.147202 & 0 & 0.504388 \\
0 & 0 & 0 & 0 & 0.603599 & -1.25849 & 0.477236 & 0.123007 \\
0 & 0.722324 & -0.664893 & 0.517214 & 0 & 0.147202 & -0.874206 & 0.398795 \\
0 & -1.00648 & 1.54784 & -0.795774 & -0.664893 & 0.883211 & -0.488821 & -1.42988
\end{pmatrix}$$

Entries 2 through 4 of the 8 scaled eigenvectors give the coordinates of the roots.

$\vec{x}_1 = < -1.51955 - 1.43763i,\ -1.56689 + 1.97705i,\ -1.51955 - 1.43763i >$

$\vec{x}_2 = < -1.51955 + 1.43763i,\ -1.56689 - 1.97705i,\ -1.51955 + 1.43763i >$

$\vec{x}_3 = < -1.40330 - 0.920669i,\ -0.696695 + 1.08723i,\ 0.00990452 + 0.920669i >$

$\vec{x}_4 = < -1.40330 + 0.920669i,\ -0.696695 - 1.08723i,\ 0.00990452 - 0.920669i >$

$\vec{x}_5 = < 0,\ 0,\ 0 >$

$\vec{x}_6 = < 0.00990452 - 0.920669i,\ -0.696695 - 1.08723i,\ -1.40330 + 0.920669i >$

$\vec{x}_7 = < 0.00990452 + 0.920669i,\ -0.696695 + 1.08723i,\ -1.40330 - 0.920669i >$

$\vec{x}_8 = < 0.252318,\ 0.346990,\ 0.252318 >$

5.5 PHCpack Test Problem "Mickey"

The following polynomial system is from the set of test problems accompanying PHCpack [21].

$$x^2 + 4y^2 - 4 = 0$$
$$2y^2 - x = 0$$

This is a simple system with which we can illustrate the use of an alternate degree bound for characterizing replacements. We will also explore some requirements for forming the linear combination of multiplication matrices to obtain the pseudo-companion matrix.

We have been choosing replacements which cause a decrease in degree. Here we will use a different merit function. Let d_x be the highest degree of x and let d_y be the highest degree of y. Then require that the replacements decrease a modified degree bound which is a function of d_x and d_y. For this problem, let the modified degree bound be given by $3d_x + 2d_y$. This is not the same as decreasing the degree with every replacement, but it will result in the eventual decrease of the degree so that the border set elements can ultimately be represented in terms of the basis elements.

Let the first replacement be $x^2 \to -4y^2 + 4$. Before the replacement, $3d_x + 2d_y = 3 \cdot 2 + 2 \cdot 0 = 6$. After the replacement, $3d_x + 2d_y = 3 \cdot 0 + 2 \cdot 2 = 4$. Let the second replacement be $y^2 \to \frac{1}{2}x$. Before the replacement, $3d_x + 2d_y = 3 \cdot 0 + 2 \cdot 2 = 4$. After the replacement, $3d_x + 2d_y = 3 \cdot 1 + 2 \cdot 0 = 3$. Every replacement decreases the modified degree bound, so we can construct the pseudo-companion matrix.

Elements of the border set should have a degree of x or y equal to 2 for one of the replacements to be usable. Take the basis to be monomials in which the degree of

each individual element is either zero or one. Namely the basis elements are 1, x, y, and xy. Since there are four elements in the basis, the pseudo-companion matrix will have four rows and columns. The pseudo-companion matrix can only encode one root per eigenvector, and this polynomial system has four roots, so this basis size is optimal.

We start by constructing multiplication matrices for x and y. Multiplying all elements of the basis by x yields the following:

$$1 \xrightarrow{*x} x$$
$$x \xrightarrow{*x} x^2$$
$$y \xrightarrow{*x} xy$$
$$xy \xrightarrow{*x} x^2y$$

Using our replacements, we have that $x^2 = -4y^2 + 4 = -4(\frac{1}{2}x) + 4 = -2x + 4$ and $x^2y = (-2x + 4)y = -2xy + 4y$. Updating our results, we have:

$$1 \xrightarrow{*x} x$$
$$x \xrightarrow{*x} 4 - 2x$$
$$y \xrightarrow{*x} xy$$
$$xy \xrightarrow{*x} 4y - 2xy$$

Then the multiplication matrix for x is

$$M_x = \begin{array}{c} \\ 1 \\ x \\ y \\ xy \end{array} \begin{array}{c} \begin{array}{cccc} 1 & x & y & xy \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 4 & -2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 4 & -2 \end{pmatrix} \end{array}$$

Now make the multiplication matrix for y. Multiplying all elements of the basis by y yields the following:

$$1 \xrightarrow{*y} y$$
$$x \xrightarrow{*y} xy$$
$$y \xrightarrow{*y} y^2$$
$$xy \xrightarrow{*y} xy^2$$

Using our replacements again gives $y^2 = \frac{1}{2}x$ and $xy^2 = x(\frac{1}{2}x) = \frac{1}{2}x^2 = \frac{1}{2}(-2x + 4) = -x + 2$. The updated multiplication results are

$$1 \xrightarrow{*y} y$$
$$x \xrightarrow{*y} xy$$
$$y \xrightarrow{*y} \frac{1}{2}x$$
$$xy \xrightarrow{*y} 2 - x$$

Then the multiplication matrix for y is

$$M_y = \begin{array}{c} \\ 1 \\ x \\ y \\ xy \end{array} \begin{array}{c} \begin{array}{cccc} 1 & x & y & xy \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{1}{2} & 0 & 0 \\ 2 & -1 & 0 & 0 \end{pmatrix} \end{array}$$

For reference, the following are the decimal values of the roots of the polynomial system:

$$\begin{aligned} x_1 &= -3.23607, & y_1 &= 1.27202i \\ x_2 &= -3.23607, & y_2 &= -1.27202i \\ x_3 &= 1.23607, & y_3 &= 0.786151 \\ x_4 &= 1.23607, & y_4 &= -0.786151 \end{aligned}$$

The eigenvectors of the multiplication matrix M_x are

$$\begin{aligned} &< -0.295242,\ 0.955423,\ 0,\ 0 > \\ &< 0,\ 0,\ -0.295242,\ 0.955423 > \\ &< 0.628960,\ 0.777438,\ 0,\ 0 > \\ &< 0,\ 0,\ 0.628960,\ 0.777438 > \end{aligned}$$

Our procedure includes scaling each eigenvector of the pseudo-companion matrix so that the first element is 1. That would be impossible for the second and fourth eigenvectors since the first element is 0. The matrix M_x on its own does not constitute a pseudo-companion matrix. The eigenvalues of M_x are $-3.23607,\ -3.23607,\ 1.23607,$ and 1.23607. These are the x coordinates of the roots.

The scaled eigenvectors of the multiplication matrix M_y are

$$< 1,\ -3.23607,\ 1.27202i,\ -4.11634i >$$
$$< 1,\ -3.23607,\ -1.27202i,\ 4.11634i >$$
$$< 1,\ 1.23607,\ 0.786151,\ 0.971737 >$$
$$< 1,\ 1.23607,\ -0.786151,\ -0.971737 >$$

None of the eigenvectors of M_y have 0 as their first element. After scaling, the second and third elements give the x and y coordinates of each root. Thus the multiplication matrix M_y on its own is a pseudo-companion matrix. The eigenvalues of M_y are the y coordinates of each root: $1.27202i,\ -1.27202i,\ 0.786151$, and -0.786151.

Our general idea is to construct the pseudo-companion matrix as a linear combination of the multiplication matrices. The scaled eigenvectors of the matrix $M_x + M_y$ are

$$< 1,\ -3.23607,\ 1.27202i,\ -4.11634i >$$
$$< 1,\ -3.23607,\ -1.27202i,\ 4.11634i >$$
$$< 1,\ 1.23607,\ 0.786151,\ 0.971737 >$$
$$< 1,\ 1.23607,\ -0.786151,\ -0.971737 >$$

The matrix $M_x + M_y$ also successfully provides the roots of the polynomial system. The eigenvalues of $M_x + M_y$ are the sums of the coordinates of the roots: $-3.23607 + 1.27202i,\ -3.23607 - 1.27202i,\ 2.02222$, and 0.449917.

We have seen that M_y and $M_x + M_y$ encode the roots, while M_x does not. We would like to know why some linear combinations of the multiplication matrices are successful while other linear combinations fail. Investigating the differences among

these matrices, we notice that M_y and $M_x + M_y$ have four distinct eigenvalues while M_x has only two distinct eigenvalues. Based on this, we suspect that the pseudo-companion matrix requires distinct eigenvalues. Test this hypothesis by intentionally constructing another linear combination of M_x and M_y with fewer than four distinct eigenvalues. We are able to do this, since the eigenvalues are linear combinations of the coordinates of the roots, with the coefficients being the same as the coefficients in the linear combination of the multiplication matrices.

We solve for two coefficients a and b for which $ax_2 + by_2 = ax_4 + by_4$. This equality is satisfied when $a = (0.175789 - 0.284432i)b$. We will take $a = 0.175789 - 0.284432i$ and $b = 1$. Then the eigenvalues of $aM_x + bM_y$ are $-0.568864 + 2.19246i,\ 1.00344 - 0.351578i,\ -0.568864 - 0.351578i$, and $-0.568864 - 0.351578i$. Two eigenvalues are the same, as desired. Now find the eigenvectors. After scaling we obtain

$$< 1,\ -3.23607,\ 1.27202i,\ -4.11634i >$$

$$< 1,\ 1.23607,\ 0.786151,\ 0.971737 >$$

$$< 1,\ 0,\ -0.568864 - 0.351578i,\ -0.703155 + 1.13773i >$$

$$< 1,\ 13.7541 - 23.0345i,\ 3.56508 + 7.60973i,\ -24.8937 - 6.51698i >$$

The first two eigenvectors have x and y coordinates of roots as their second and third elements. The other two eigenvectors do not. This is further evidence that repeated eigenvalues are problematic.

In general this should be avoidable by using randomized coefficients in the linear combination of the multiplication matrices. This will be explored further in the "Wright" example. In an automated setting where any type of polynomial system could be provided, randomization should be incorporated. Alternatively, if randomization is not used, it should be verified that all eigenvalues are distinct before pro-

ceeding.

Of course, it was also observed that in some cases the pseudo-companion matrix can be taken to be equal to a single multiplication matrix (M_y in this example). This could decrease the computation time. The actual time saved would depend on the computing setup, since all the multiplication matrices could be built at once if parallelization is available. If we have preexisting knowledge that some coordinate of the root should be different for each root, then we can use a single multiplication matrix as the pseudo-companion matrix.

5.6 PHCpack Test Problem "Wright"

The following is from the test problems accompanying PHCpack [21]. The specific problem is from [24].

$$x_1^2 - x_1 + x_2 + x_3 + x_4 + x_5 - 10 = 0$$

$$x_2^2 + x_1 - x_2 + x_3 + x_4 + x_5 - 10 = 0$$

$$x_3^2 + x_1 + x_2 - x_3 + x_4 + x_5 - 10 = 0$$

$$x_4^2 + x_1 + x_2 + x_3 - x_4 + x_5 - 10 = 0$$

$$x_5^2 + x_1 + x_2 + x_3 + x_4 - x_5 - 10 = 0$$

The solutions are permutations of

$$(2,\ 2,\ 2,\ 2,\ 2)$$

$$(-5,\ -5,\ -5,\ -5,\ -5)$$

$$(-1,\ -1,\ 3,\ 3,\ 3)$$

$$(-2,\ -2,\ -2,\ 4,\ 4)$$

$$(-a,\ 2+a,\ 2+a,\ 2+a,\ 2+a)$$

$$(5+a,\ -3-a,\ -3-a,\ -3-a,\ -3-a)$$

where $a = \frac{-5+\sqrt{33}}{2}$.

No perturbation terms are required to construct the pseudo-companion matrix. The highest degree term in each polynomial is univariate and has a degree strictly larger than all other terms in the polynomial. The replacement monomials will be $x_1^2,\ x_2^2,\ x_3^2,\ x_4^2$ and x_5^2. The basis will consist of all monomials for which each individual degree is 0 or 1. The complete list is

$1,$

$x_1,\ x_2,\ x_3,\ x_4,\ x_5,$

$x_1x_2,\ x_1x_3,\ x_1x_4,\ x_1x_5,\ x_2x_3,\ x_2x_4,\ x_2x_5,\ x_3x_4,\ x_3x_5,\ x_4x_5,$

$x_1x_2x_3,\ x_1x_2x_4,\ x_1x_2x_5,\ x_1x_3x_4,\ x_1x_3x_5,\ x_1x_4x_5,\ x_2x_3x_4,\ x_2x_3x_5,\ x_2x_4x_5,\ x_3x_4x_5,$

$x_1x_2x_3x_4,\ x_1x_2x_3x_5,\ x_1x_2x_4x_5,\ x_1x_3x_4x_5,\ x_2x_3x_4x_5,$

$x_1x_2x_3x_4x_5$

This basis contains 32 elements, which will result in a 32 by 32 matrix. There are 32 roots to the polynomial system, so no smaller basis can be used. Our choice of

construction seems optimal.

The pseudo-companion matrix is constructed as a linear combination of multiplication matrices (one for each variable). For this polynomial system, the success of the pseudo-companion matrix method depends on the coefficients used. Various sets of coefficients, the eigenvalues of the resulting pseudo-companion matrix, and the corresponding norms of the differences between the actual roots and the roots computed from the matrix are shown in Figure 5.4. Coefficients of (1, 0, 0, 0, 0), (1, 1, 1, 1, 1), and (1, 2, 3, 4, 5) result in failure of the method while randomized coefficients, either real or complex, result in success. Perturbation is not the reason for the issue, since this polynomial system required no perturbation terms for the matrix construction. This demonstrates that randomization of the coefficients is necessary for some polynomial systems. This polynomial system had a special structure which may have contributed to the problem, but the implications are important because many polynomial systems of interest could have some special structure.

```
Coefficients:  {1, 0, 0, 0, 0}
Eigenvalues:   {5.37228, -5., 4., 4., 4., 4., -3.37228, -3.37228,
  -3.37228, -3.37228, 3., 3., 3., 3., 3., 3., 2.37228, 2.37228, 2.37228,
  2.37228, -2., -2., -2., -2., 2., -2., -2., -1., -1., -1., -1., -0.372281}
Maximum norm:  7.98717×10^14

Coefficients:  {1, 1, 1, 1, 1}
Eigenvalues:   {-25., 10., 9.11684, 9.11684, 9.11684, 9.11684,
  9.11684, -8.11684, -8.11684, -8.11684, -8.11684, -8.11684, 7., 7.,
  7., 7., 7., 7., 7., 7., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2.}
Maximum norm:  7.02554×10^15

Coefficients:  {1, 2, 3, 4, 5}
Eigenvalues:
 {-75., -41.8397, -33.0951, 33., 32.8397, 30.0951, 30., 29., 27.3505, 25., 25.,
  24.606, -24.3505, 24., 21.8614, 21., 21., 18., 17., 17., -15.606, 13.,
  -12., 12., 12., 9., -6.86141, -6., 6., 6., 1.91636×10^-15, 1.91636×10^-15}
Maximum norm:  2.68179×10^15

Coefficients:  {-0.892495, 0.19142, 0.93218, 0.793989, 0.845965}
Eigenvalues:
 {-14.1142, -9.3553, 8.41748, -7.94857, 6.92675, 6.88819, 6.61489, 6.09761,
  6.0072, 5.7993, 5.45444, -4.63585, -4.33315, -4.0213, 3.91331, 3.74212,
  -3.50401, 2.99948, 2.48219, 2.25953, 2.17034, 2.11687, 1.88025, 1.84176,
  1.67154, -1.4994, 1.46364, -1.2915, 1.11878, 1.08786, -0.94664, 0.633351}
Maximum norm:  8.63107×10^-13

Coefficients:  {-0.0902376 - 0.199044 i, -0.535705 + 0.512448 i,
  0.0342581 + 0.0984042 i, -0.416437 + 0.952095 i, -0.14843 - 0.0893559 i}
Eigenvalues:   {5.78276 - 6.37274 i, -3.39975 + 6.23816 i,
  3.11113 - 6.03868 i, 2.60226 - 5.07951 i, -2.51498 + 4.97724 i,
  4.19979 - 3.43763 i, -3.24574 + 4.2262 i, -3.01297 + 3.78745 i,
  0.881098 - 4.27949 i, -2.496 + 3.56987 i, 0.258658 + 4.02752 i,
  -2.33629 + 3.26883 i, -2.83769 + 2.75351 i, 0.0200298 + 3.7539 i,
  1.97723 - 3.15293 i, -2.3131 + 2.54909 i, 1.62807 - 2.49481 i,
  -1.0761 + 2.62734 i, -0.965883 + 2.57003 i, -0.733114 + 2.13128 i,
  -0.726944 + 1.96921 i, 0.338914 - 2.03453 i, -1.27339 + 1.61714 i,
  -1.46387 + 1.38024 i, -1.94094 - 0.378355 i, -1.79171 - 0.0105447 i,
  -1.44296 + 0.811436 i, -1.60073 + 0.4105 i, -1.44255 - 0.668671 i,
  -0.695579 + 1.11602 i, -1.21019 + 0.372685 i, -0.78429 + 0.182998 i}
Maximum norm:  1.68416×10^-13
```

Figure 5.4: Output for varying coefficients of the multiplication matrices during the pseudo-companion matrix build for PHCpack test problem wright.

5.7 Use of a Perturbation

Consider the following polynomial system.

$$p_1 = 1 + 2x + 3x^2 + 4xy$$
$$p_2 = 5 + 6xy + 7y^2$$

Both polynomials lack strictly highest degree terms. We will introduce a perturbation to obtain this form. Define new polynomials

$$q_1 = 1 + 2x + 3x^2 + 4xy + \epsilon\, x^3$$
$$q_2 = 5 + 6xy + 7y^2 + \epsilon\, y^3$$

The degree of the added terms, in this case 3, is one greater than the degree of the polynomials.

The set of monomials appearing in $\vec{p}$ is $\{1, x, x^2, xy, y^2\}$; the basis will include these. The added monomials x^3 and y^3 will be replaced using

$$x^3 = -\mu - 2\mu x - 3\mu x^2 - 4\mu xy$$
$$y^3 = -5\mu - 6\mu xy - 7\mu y^2$$

where $\mu = \frac{1}{\epsilon}$.

In order to use these substitutions for every element of the border set, every monomial $x^{\alpha_1} y^{\alpha_2}$ in the border set should have $\alpha_1 \geq 3$ or $\alpha_2 \geq 3$ (both could be true). In order to achieve this, use a rectangular basis with bound $< 2, 2 >$. Explicitly, the basis elements are $1,\ x,\ y,\ x^2,\ xy,\ y^2,\ x^2y,\ xy^2,$ and x^2y^2.

Determine the effect of multiplying each basis element by x and y, using substitutions if needed to express each result as a linear combination of basis elements. For multiplication by x we have

$$
\begin{aligned}
1 &\xrightarrow{*x} x \\
x &\xrightarrow{*x} x^2 \\
y &\xrightarrow{*x} xy \\
x^2 &\xrightarrow{*x} -\mu - 2\mu x - 3\mu x^2 - 4\mu xy \\
xy &\xrightarrow{*x} x^2 y \\
y^2 &\xrightarrow{*x} xy^2 \\
x^2 y &\xrightarrow{*x} -\mu y - 2\mu xy - 3\mu x^2 y - 4\mu xy^2 \\
xy^2 &\xrightarrow{*x} x^2 y^2 \\
x^2 y^2 &\xrightarrow{*x} 20\mu^2 x - \mu y^2 + 24\mu^2 x^2 y + (28\mu^2 - 2\mu) xy^2 - 3\mu x^2 y^2
\end{aligned}
$$

and for multiplication by y we have

$$
\begin{aligned}
1 &\xrightarrow{*y} y \\
x &\xrightarrow{*y} xy \\
y &\xrightarrow{*y} y^2 \\
x^2 &\xrightarrow{*y} x^2y \\
xy &\xrightarrow{*y} xy^2 \\
y^2 &\xrightarrow{*y} -5\mu - 6\mu xy - 7\mu y^2 \\
x^2y &\xrightarrow{*y} x^2y^2 \\
xy^2 &\xrightarrow{*y} -5\mu x - 6\mu x^2y - 7\mu xy^2 \\
x^2y^2 &\xrightarrow{*y} 6\mu^2 y - 5\mu x^2 + 12\mu^2 xy + 18\mu^2 x^2y + 24\mu^2 xy^2 - 7\mu x^2y^2
\end{aligned}
$$

These determine the matrices M_x and M_y.

$$
M_x = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
-\mu & -2\mu & 0 & -3\mu & -4\mu & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & -\mu & 0 & -2\mu & 0 & -3\mu & -4\mu & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 20\mu^2 & 0 & 0 & 0 & -\mu & 24\mu^2 & 28\mu^2 - 2\mu & -3\mu
\end{pmatrix}
$$

$$M_y = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -5\mu & 0 & 0 & 0 & -6\mu & -7\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -5\mu & 0 & 0 & 0 & 0 & -6\mu & -7\mu & 0 \\ 0 & 0 & 6\mu^2 & -5\mu & 12\mu^2 & 0 & 18\mu^2 & 24\mu^2 & -7\mu \end{pmatrix}$$

The pseudo-companion matrix is a linear combination of these. The coefficients in the linear combination are randomized.

ϵ is taken to be a small number, in this case 10^{-3}, and then the eigensystem is computed. The polynomial system $\vec{q}$ has more roots than the polynomial system $\vec{p}$. The computation produces some eigenvalues with large magnitude; the eigenvectors associated with these do not correspond to roots of $\vec{p}$ and are disregarded. The remaining four eigenvectors are scaled so the first element is one, and the second and third elements of these eigenvectors are approximations of the x and y coordinates of the roots of $\vec{p}$. The computed approximations are:

$$< -2.08430,\ 1.18208 >$$

$$< -0.108885 - 0.217562i,\ 0.0415760 - 0.755769i >$$

$$< -0.108885 + 0.217562i,\ 0.0415760 + 0.755769i >$$

$$< 6.34639,\ -5.30926 >$$

These are roots of $\vec{q}$.

The roots of $\vec{p}$ as computed by NSolve in Mathematica are:

$$
\begin{aligned}
&x \to -2.08494, && y \to 1.18361 \\
&x \to -0.108878 - 0.217564i, && y \to 0.0415373 - 0.755775i \\
&x \to -0.108878 + 0.217564i, && y \to 0.0415373 + 0.755775i \\
&x \to 6.30270, && y \to -5.26669
\end{aligned}
$$

Plots of the curves are shown in Figures 5.5 through 5.8. The real solutions can be visualized on these plots. The system $\vec{q} = \vec{0}$ is close to the system $\vec{p} = \vec{0}$ when $|x|$ and $|y|$ are relatively small. The system $\vec{q} = \vec{0}$ has additional solutions with large norm.

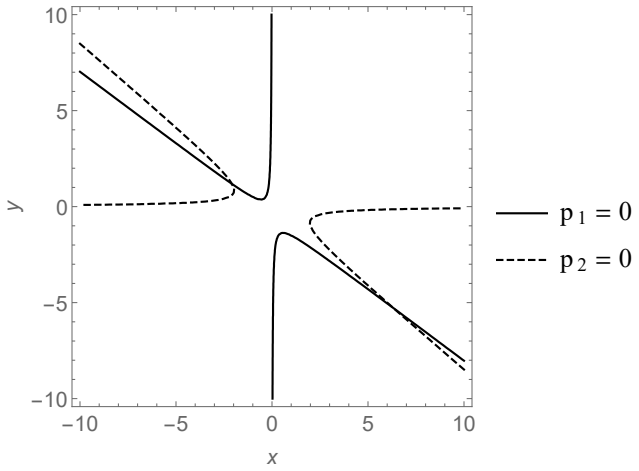

Figure 5.5: The curves $p_1 = 0$ and $p_2 = 0$ where $p_1 = 1 + 2x + 3x^2 + 4xy$ and $p_2 = 5 + 6xy + 7y^2$.

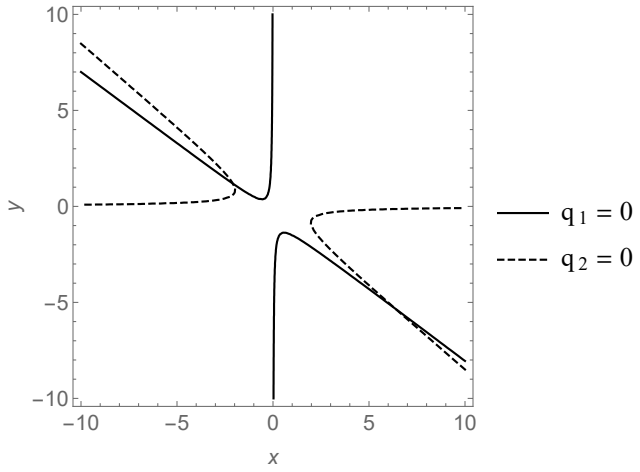

Figure 5.6: The system $\vec{q} = \vec{0}$ is close to $\vec{p} = \vec{0}$ when $|x|$ and $|y|$ are small.

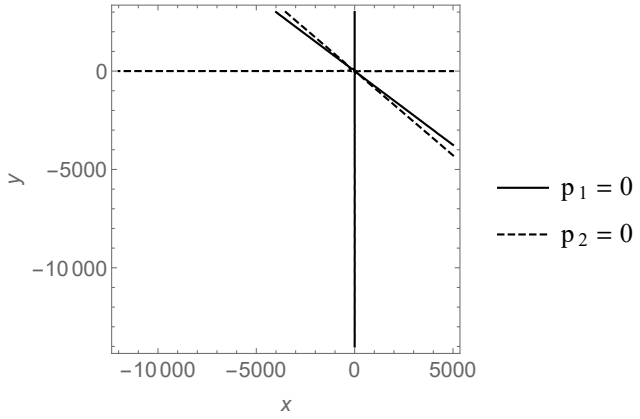

Figure 5.7: The curves $p_1 = 0$ and $p_2 = 0$ where $p_1 = 1 + 2x + 3x^2 + 4xy$ and $p_2 = 5 + 6xy + 7y^2$.

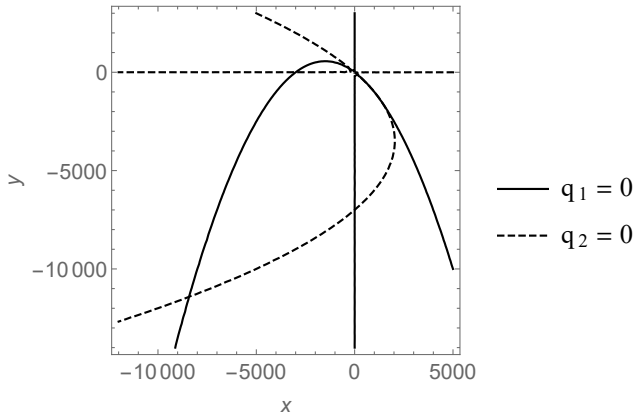

Figure 5.8: The system $\vec{q} = \vec{0}$ has additional solutions with large norm.

5.8 Multiple Steady State Solutions for a Reaction- Diffusion Model

We will approximate steady state solutions of a reaction-diffusion model as a means of constructing polynomial systems of increasing size. Each polynomial system will be qualitatively similar, but the number of polynomials and variables will increase. We can then examine the corresponding increase in run times.

When we construct the pseudo-companion matrix, only half of the replacements will decrease the degree, but the structure of the problem still allows the replacement process to terminate. Half of the polynomials can be used in their original form, while the other half require the addition of a perturbation term.

Consider the Turing reaction-diffusion model for pattern formation [19], specifically the Gierer-Meinhardt system [6]. We will use the form of the model and parameters given in [16]. This system is given by

$$u_t = r\left(1 + \frac{u^2}{v}\right) - \mu u + D_u \nabla^2 u \tag{5.1a}$$

$$v_t = ru^2 - \nu v + D_v \nabla^2 v \tag{5.1b}$$

where

$$r = 0.03,\ \mu = 0.25,\ D_u = 0.000027,\ \nu = 2.0,\ D_v = 0.027$$

with zero flux boundary conditions. $u(t, x)$ and $v(t, x)$ are concentrations of an activator and inhibitor. We will consider this model for one spatial dimension.

The constant steady state is $u^* = 8.12,\ v^* = 0.989$. Nonconstant steady state solutions are of greater interest. One way to look for additional steady state solutions is to vary the initial conditions and solve the partial differential equation system over

an extended time interval, so that a steady state solution is attained (time-marching). This approach can produce multiple stable steady states for this system, but provides no insight for unstable steady state solutions.

We are interested in finding multiple steady state solutions. To systematically search for both stable and unstable steady state solutions, we will discretize the steady state problem using finite differences, and solve the corresponding system of polynomials. This process is utilized in [8]. Although higher order discretizations exist, we use a second order central difference method to be consistent with [8].

In equations 5.1a and 5.1b, set $u_t = v_t = 0$. Multiply equation 5.1a by v to get a system of polynomials. Subtract equation 5.1b from equation 5.1a to eliminate the u^2 term in equation 5.1a. Divide equation 5.1b by r. The resulting equations are

$$0 = (r + \nu)v - \mu uv + D_u(\nabla^2 u)v - D_v\nabla^2 v \tag{5.2a}$$

$$0 = u^2 - \frac{\nu}{r}v + \frac{D_v}{r}\nabla^2 v \tag{5.2b}$$

We can take advantage of the u^2 term in equation 5.2b, but we will need to add a term to equation 5.2a. Although equation 5.2a contains terms of degree 2, it is sufficient to add a term of the form ϵv^2. Using the replacement for v^2 will not decrease the degree overall, but it will decrease the degree of v and increase the degree of u; then the replacement for u^2 can be used, which will decrease the degree. Thus the replacement process is guaranteed to terminate.

Take equally spaced grid points x_j where $x_0 = 0$, $x_N = 1$, and $h = \frac{1}{N}$. In the Mathematica implementation, N is replaced by M since N is already used by the Mathematica program. At x_j $(j = 1, ..., N-1)$, $\nabla^2 u$ is replaced by the second order centered approximation $\frac{1}{h^2}[u_{j-1} - 2u_j + u_{j+1}]$. The boundary conditions become $\frac{1}{h}(\frac{3}{2}u_0 - 2u_1 + \frac{1}{2}u_2) = 0$ and $\frac{1}{h}(\frac{3}{2}u_N - 2u_{N-1} + \frac{1}{2}u_{N-2}) = 0$. The same approximations

are made for v. For additional information on finite difference approximations, see [12]. Adding a $-\epsilon v^2$ term to equation 5.2a (ϵ was taken to be 2^{-4}) and using these discretizations results in the system of polynomials

$$\begin{aligned} 0 &= -\epsilon v_j^2 + (r+\nu)v_j - \mu u_j v_j + D_u \left(\frac{1}{h^2}[u_{j-1} - 2u_j + u_{j+1}]\right) v_j \\ &\qquad - D_v \left(\frac{1}{h^2}[v_{j-1} - 2v_j + v_{j+1}]\right) \\ 0 &= u_j^2 - \frac{\nu}{r} v_j + \frac{D_v}{r}\left(\frac{1}{h^2}[v_{j-1} - 2v_j + v_{j+1}]\right) \end{aligned}$$

for $j = 1, ..., N-1$. This gives replacements of

$$\begin{aligned} v_j^2 &= \frac{1}{\epsilon}\Bigg[(r+\nu)v_j - \mu u_j v_j + D_u \left(\frac{1}{h^2}[u_{j-1} - 2u_j + u_{j+1}]\right) v_j \\ &\qquad - D_v \left(\frac{1}{h^2}[v_{j-1} - 2v_j + v_{j+1}]\right)\Bigg] \\ u_j^2 &= \frac{\nu}{r} v_j - \frac{D_v}{r}\left(\frac{1}{h^2}[v_{j-1} - 2v_j + v_{j+1}]\right) \end{aligned}$$

for $j = 1, ..., N-1$. The boundary equations are used to eliminate $u_0,\ u_N,\ v_0,$ and v_N from the equations. This results in a system of $2N-2$ polynomial equations in $2N-2$ unknowns. The pseudo-companion matrix size is $2^{2N-2} \times 2^{2N-2}$.

The number of grid points must be chosen prior to matrix construction, but all parameter values and ϵ may be left as symbolic quantities at this stage. If needed, the symbolic matrix could be built once and subsequent steps could be performed using multiple sets of parameters. Note that it takes longer to build the matrix with arbitrary parameters than with specific numerical values. Run times (real time in seconds) are shown in Table 5.1.

Biologically meaningful solutions should be real and nonnegative. We screen for solutions meeting these criteria. The eigenvalues are linear combinations of the co-

N	Symbolic matrix	Numerical matrix	Dense eigensystem	Bertini
4	5.3751	0.51758	0.0030028	1.6634
5	146.59	8.4115	0.099112	12.355
6	4492.3	163.45	2.4658	107.10

Table 5.1: Run times on a laptop computer (real time in seconds) for the steady state solutions problem.

ordinates of the roots. We have chosen the coefficients in this linear combination to be nonnegative and real. Then the eigenvalues corresponding to meaningful solutions are positive and real (we are not interested in solutions which are identically equal to zero). First we can screen solutions based on eigenvalues. Then screen the solutions by checking each coordinate individually.

The addition of the perturbation term means that the pseudo-companion matrix eigenvectors encode approximations of the roots. The results are run through a few iterations of Newton's method as a refinement step. Our code limits the number of iterations to at most 6; some randomizations may require more iterations. Since ours is a new method, we also found solutions using Bertini for comparison. When running Bertini, all the default settings were used. The boundary conditions were used to eliminate $u_0,\ u_N,\ v_0$, and v_N from the equations prior to running Bertini, and the u_j and v_j variables comprised two variable groups. We first verified that our matrix method and Bertini found the same number of biologically meaningful roots. We then verified that the 2-norm of the difference between each root found by our matrix method and the corresponding root found by Bertini was less than 10^{-14}.

The results are shown in Figures 5.9 through 5.14. The psuedo-companion matrix successfully finds the roots of the polynomial system. The constant steady state is among the results.

Each additional grid point increases the number of polynomials and variables by 2. As grid points are added, the time required to solve the polynomial system increases

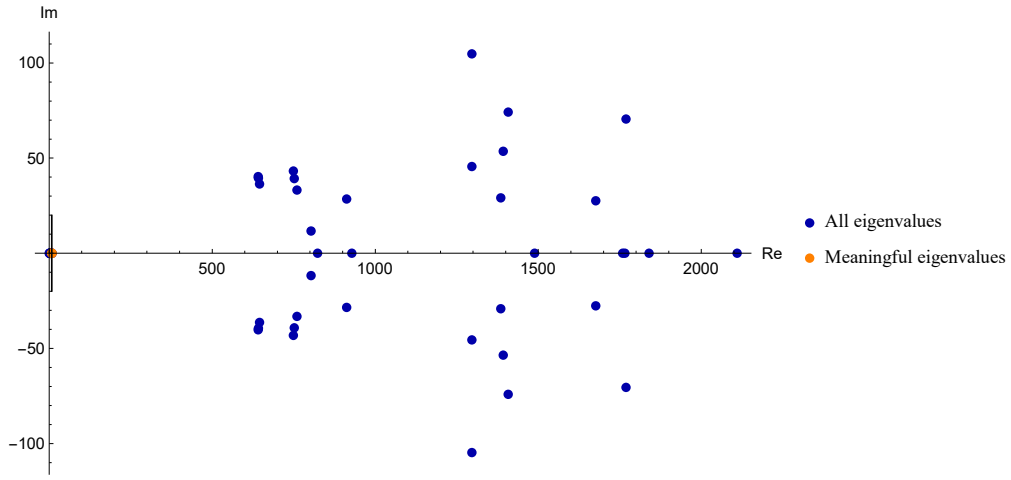

Figure 5.9: Eigenvalues for discretized steady states on grid points 0 through 4.

exponentially. Note that we could make the computations faster by building the pseudo-companion matrix with decimal values for the parameters instead of building it symbolically (see table 5.1). We could also search for eigenvalues in a region instead of finding the entire eigensystem.

Our pseudo-companion matrix method successfully found roots of systems of 6, 8, and 10 quadratics. For the two smaller polynomial systems, the sum of the matrix build time and the eigensystem computation time (performed in Mathematica) was less than the Bertini run time (due to variations in the Bertini run times, Bertini was run 5 times for each polynomial system and the average was reported). For the largest polynomial system, Bertini was faster. For our psuedo-companion matrix method, the majority of the time for the largest polynomial system was spent on the eigensystem computation. Note that the full eigensystem was computed using dense matrix methods. Alternative eigensolvers could be considered to utilize the distribution of the eigenvalues for solutions of interest, but that is beyond the scope of this work.

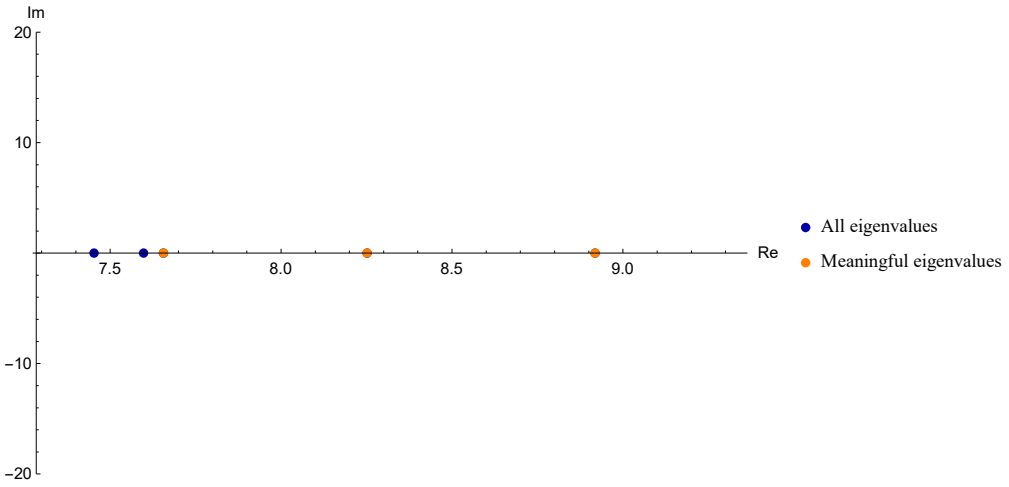

Figure 5.10: Closeup of meaningful eigenvalues for discretized steady states on grid points 0 through 4.

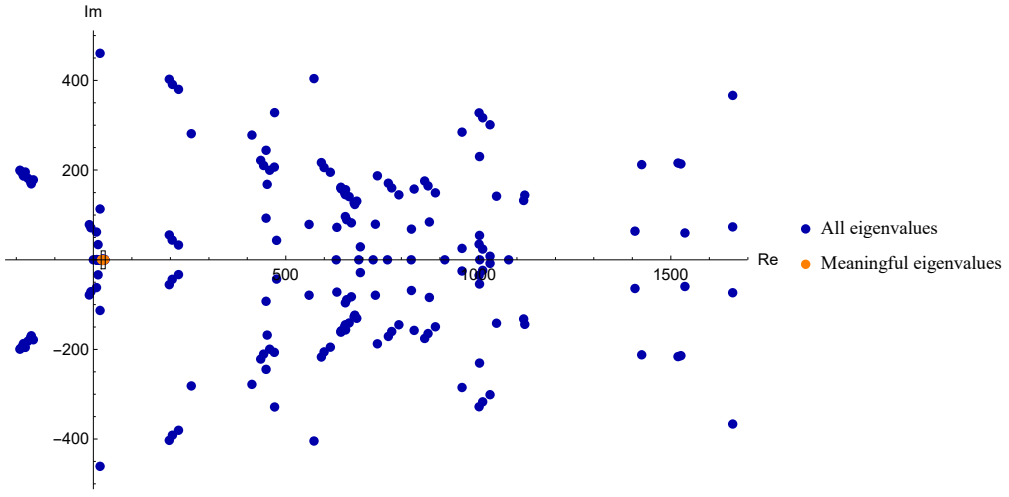

Figure 5.11: Eigenvalues for discretized steady states on grid points 0 through 5.

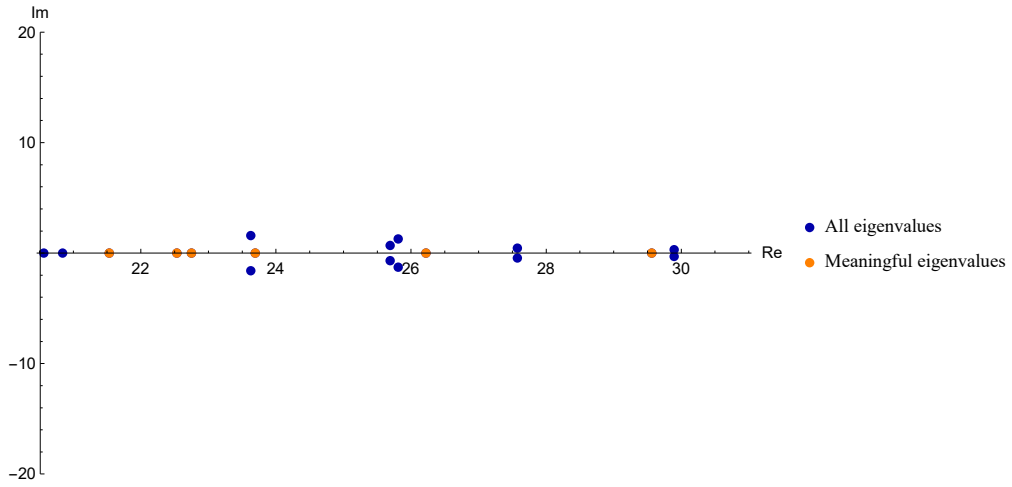

Figure 5.12: Closeup of meaningful eigenvalues for discretized steady states on grid points 0 through 5.

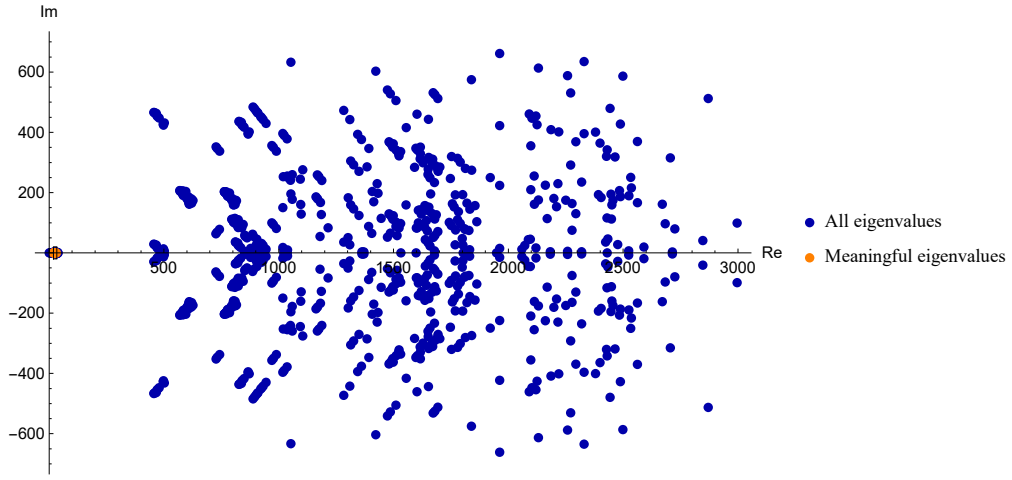

Figure 5.13: Eigenvalues for discretized steady states on grid points 0 through 6.

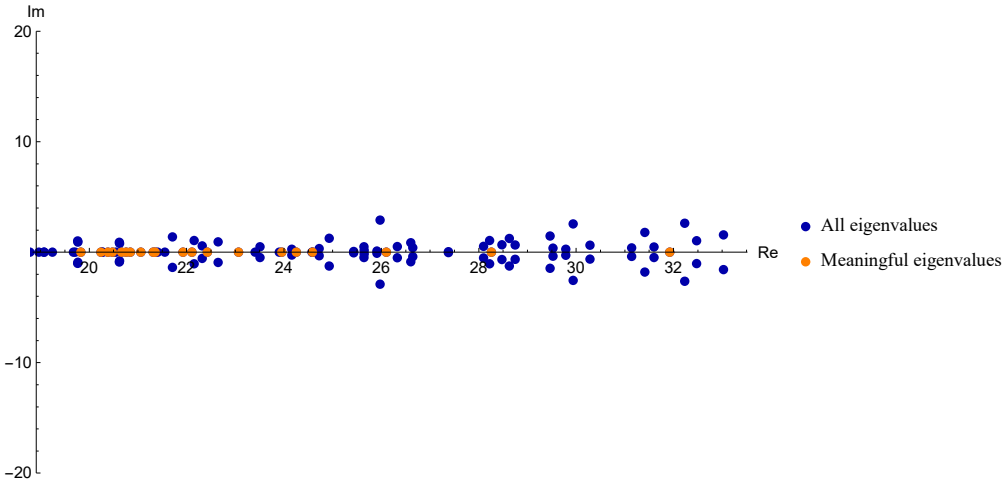

Figure 5.14: Closeup of meaningful eigenvalues for discretized steady states on grid points 0 through 6.

6 Effect of Perturbation Terms

When perturbation terms are added, the modified polynomial system $\vec{q}$ has higher degree than the original polynomial system $\vec{p}$. Thus $\vec{q}$ would be expected to have more roots than $\vec{p}$. The additional "spurious" roots are those roots of $\vec{q}$ which do not correspond to roots of $\vec{p}$. The roots of $\vec{q}$ which do correspond to roots of $\vec{p}$ will be used as approximations of the roots of $\vec{p}$. We will examine the behavior of both types of roots as ϵ approaches 0, or in other words as μ gets larger. Recall that $\epsilon = 0$ produces the original polynomial system. ϵ must be taken to be nonzero in order to construct the pseudo-companion matrix.

Earlier examples in this work were chosen because their structure illustrated some aspect of constructing the pseudo-companion matrix. Now we will select some examples from the PHCpack test problems which do not seem to have any special structure which can be exploited by our method. These polynomial systems will require the addition of a perturbation term to each individual polynomial. The degree of each perturbation term will be one more than the degree of the polynomial to which it is added.

A complete assessment of our method would include all the PHCpack test problems. For this exploratory work, we will only examine a few of them. Note that with our available computing and time resources we have selected some of the relatively smaller problems.

For visualization purposes the eigenvalues of the pseudo-companion matrices are plotted. The target solutions are depicted on these plots as linear combinations of the coordinates of the roots. The coefficients in these linear combinations are the

same as the coefficients in the linear combinations of the multiplication matrices used to construct the pseudo-companion matrices. The code is in Appendix B. 32 digit precision is used in the Mathematica implementations in anticipation of large variations in the size of the entries of the matrices.

6.1 PHCpack Test Problem "Noon3"

We begin by exploring the behavior of the PHCpack test problem noon3 [21]:

$$x_1 x_2^2 + x_1 x_3^2 - 1.1 x_1 + 1$$
$$x_2 x_1^2 + x_2 x_3^2 - 1.1 x_2 + 1$$
$$x_3 x_1^2 + x_3 x_2^2 - 1.1 x_3 + 1$$

There are 21 roots. The size of the pseudo-companion matrix is 64×64. Results are shown in Figures 6.1 through 6.4. We show results for $\mu = 2^6$ and $\mu = 2^{12}$. Increasing μ increases the magnitude of the spurious eigenvalues and improves the estimation of the roots of the original system. By $\mu = 2^{12}$, we have achieved good estimations and good separation between the spurious and meaningful eigenvalues. In practice, we may be able to use lower values of μ if we are willing to run more iterations of Newton's method and screen out spurious roots.

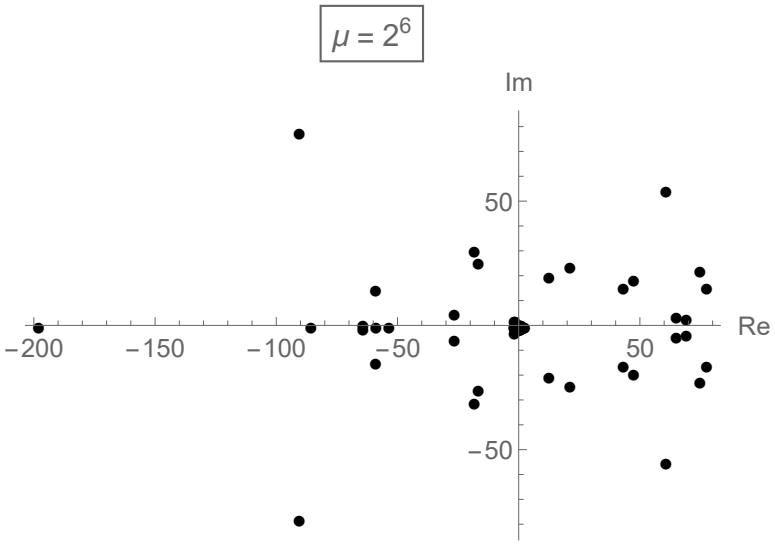

Figure 6.1: All eigenvalues for PHCpack test problem noon3 with $\mu = 2^6$.

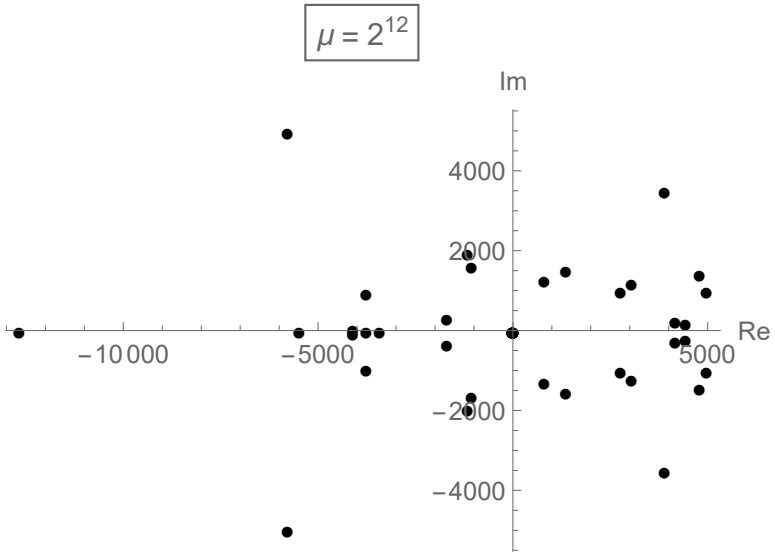

Figure 6.2: All eigenvalues for PHCpack test problem noon3 with $\mu = 2^{12}$.

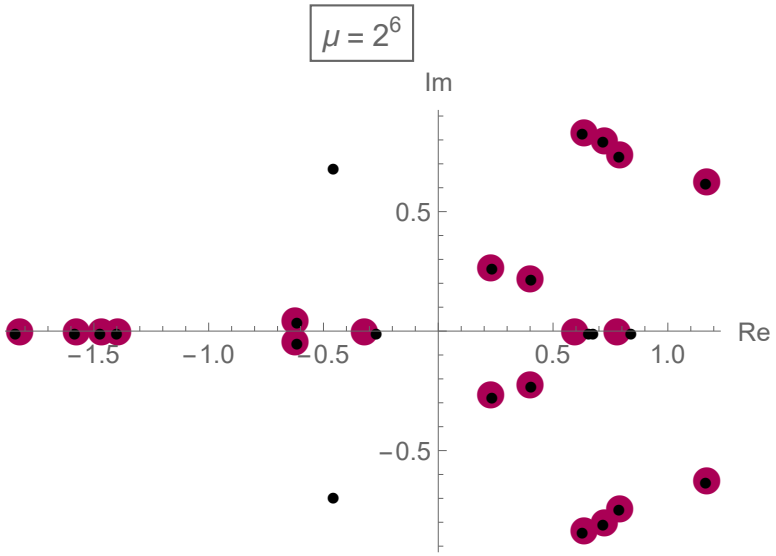

Figure 6.3: Eigenvalues (black) and target solutions (magenta) for PHCpack test problem noon3 with $\mu = 2^6$.

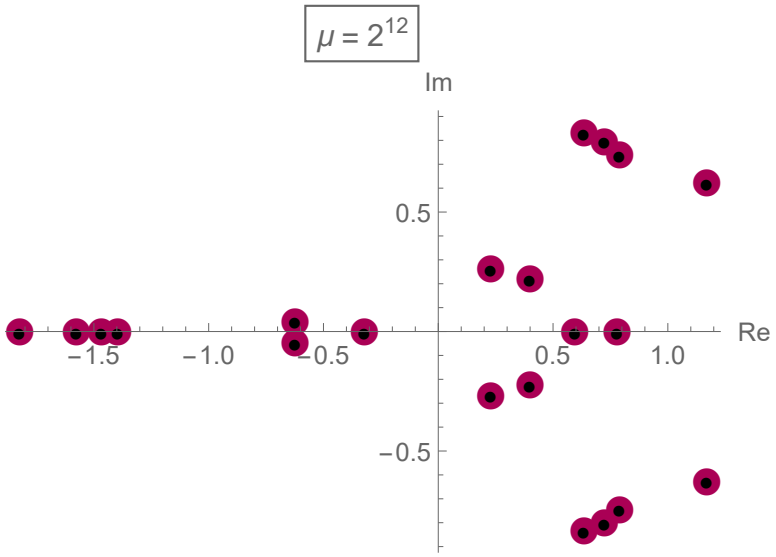

Figure 6.4: Eigenvalues (black) and target solutions (magenta) for PHCpack test problem noon3 with $\mu = 2^{12}$.

6.2 PHCpack Test Problem "Chandra4"

Now consider the PHCpack test problem chandra4 [21]:

$$
\begin{aligned}
&8H_1 - 0.51234H_1\left(1 + \frac{1}{2}H_1 + \frac{1}{3}H_2 + \frac{1}{4}H_3\right) - 8 \\
&8H_2 - 0.51234H_2\left(\frac{1}{2} + \frac{1}{3}H_1 + \frac{1}{4}H_2 + \frac{1}{5}H_3\right) - 8 \\
&8H_3 - 0.51234H_3\left(\frac{1}{3} + \frac{1}{4}H_1 + \frac{1}{5}H_2 + \frac{1}{6}H_3\right) - 8 \\
&8H_4 - 0.51234H_4\left(\frac{1}{4} + \frac{1}{5}H_1 + \frac{1}{6}H_2 + \frac{1}{7}H_3\right) - 8
\end{aligned}
$$

The system has 8 solutions. The size of the pseudo-companion matrix is 81×81. For this polynomial system, we increase μ to as high as 2^{24} to obtain convincing results on the plot. In the first set of computations, real coefficients are used in the linear combination of the multiplication matrices to obtain the pseudo-companion matrix (see results in Figures 6.5 through 6.12). In the second set of computations, complex coefficients are used (see Figures 6.13 through 6.20). Again, we continue until $\mu = 2^{24}$. In Figures 6.19 and 6.20, it is apparent that the high magnitude roots present in this system are resolved for higher values of μ.

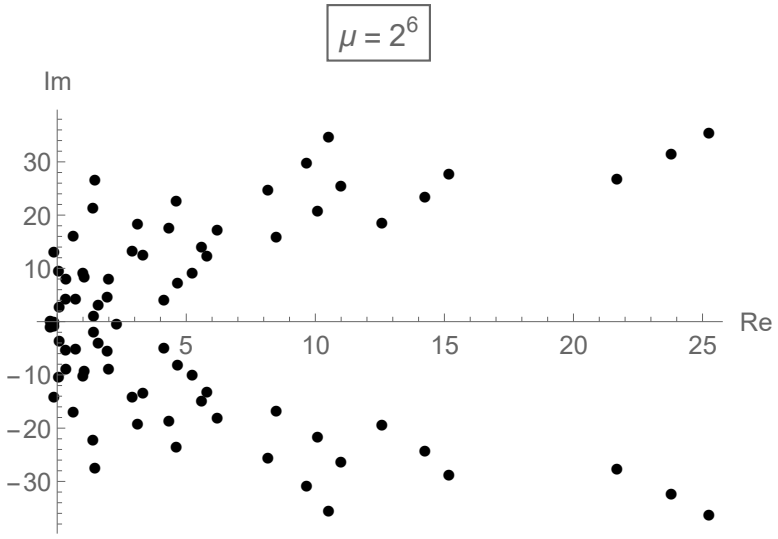

Figure 6.5: All eigenvalues for PHCpack test problem chandra4 with $\mu = 2^6$ (real randomization).

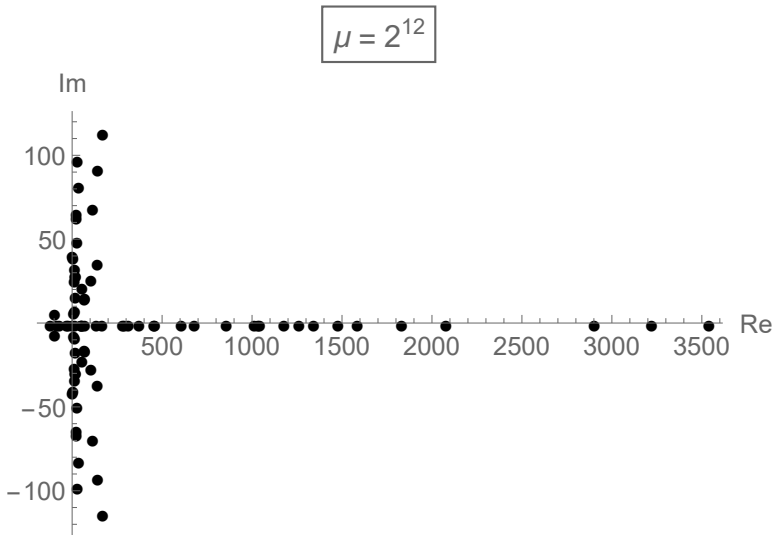

Figure 6.6: All eigenvalues for PHCpack test problem chandra4 with $\mu = 2^{12}$ (real randomization).

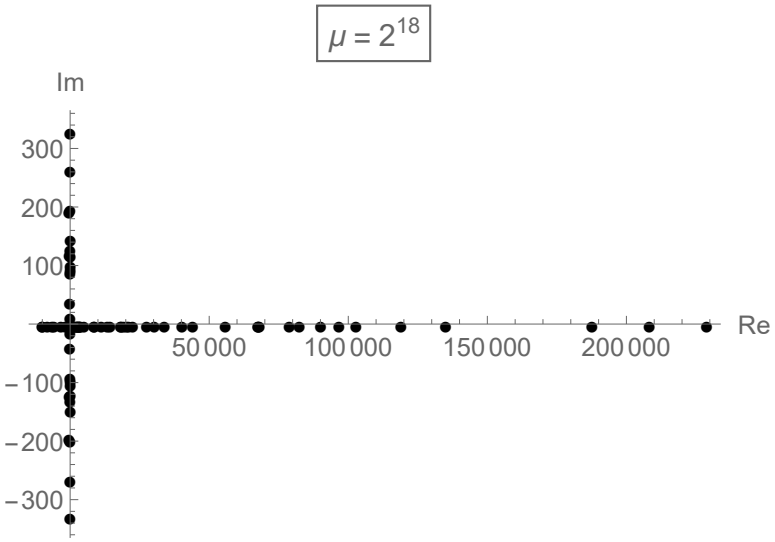

Figure 6.7: All eigenvalues for PHCpack test problem chandra4 with $\mu = 2^{18}$ (real randomization).

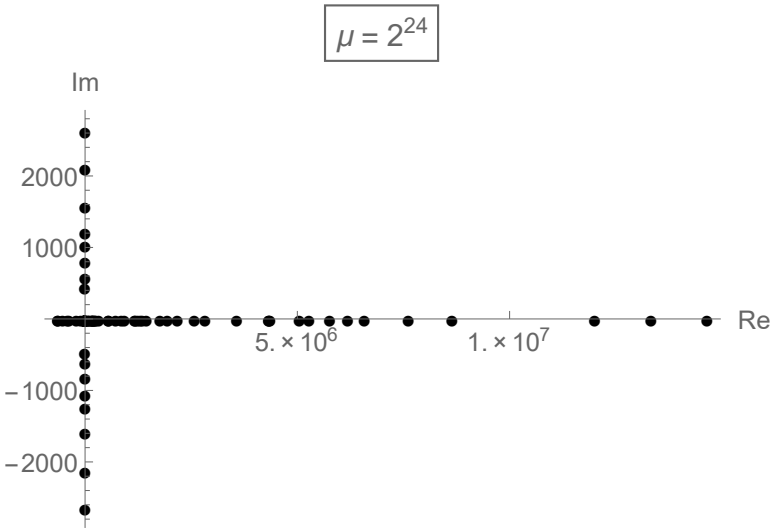

Figure 6.8: All eigenvalues for PHCpack test problem chandra4 with $\mu = 2^{24}$ (real randomization).

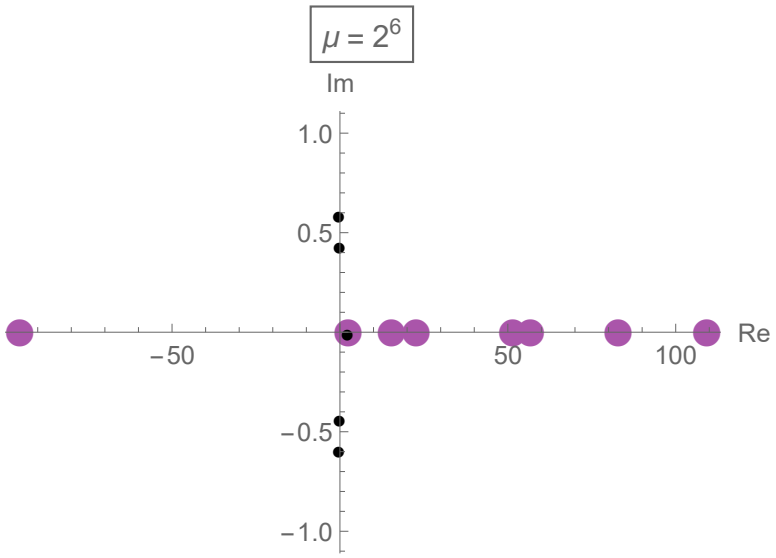

Figure 6.9: Eigenvalues (black) and target solutions (purple) for PHCpack test problem chandra4 with $\mu = 2^6$ (real randomization).

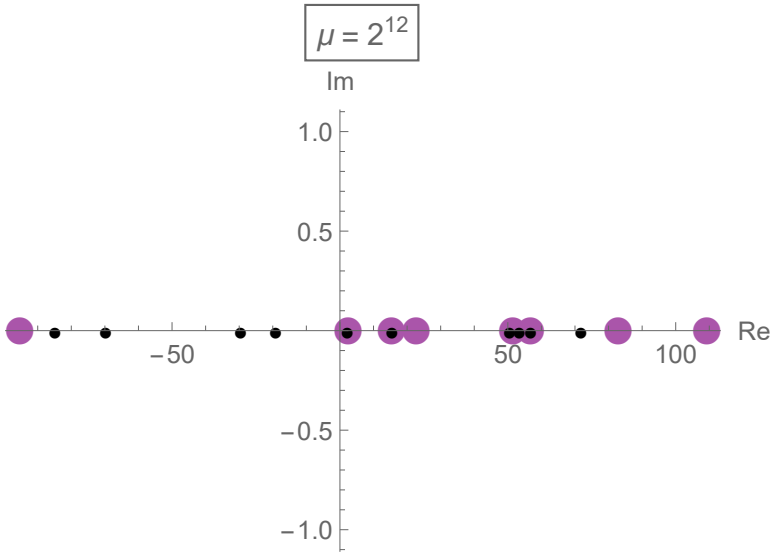

Figure 6.10: Eigenvalues (black) and target solutions (purple) for PHCpack test problem chandra4 with $\mu = 2^{12}$ (real randomization).

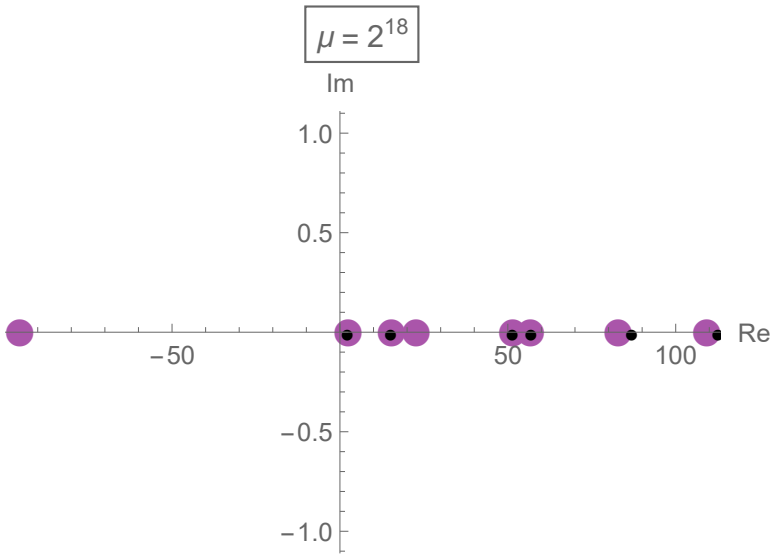

Figure 6.11: Eigenvalues (black) and target solutions (purple) for PHCpack test problem chandra4 with $\mu = 2^{18}$ (real randomization).

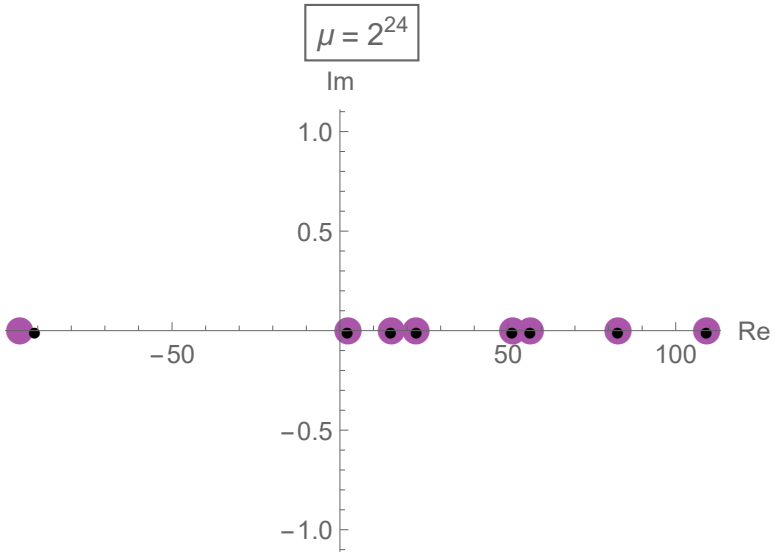

Figure 6.12: Eigenvalues (black) and target solutions (purple) for PHCpack test problem chandra4 with $\mu = 2^{24}$ (real randomization).

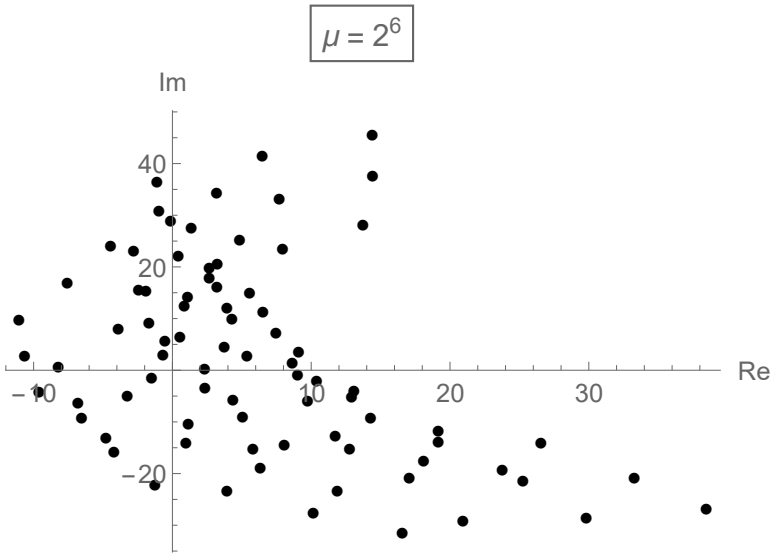

Figure 6.13: All eigenvalues for PHCpack test problem chandra4 with $\mu = 2^6$ (complex randomization).

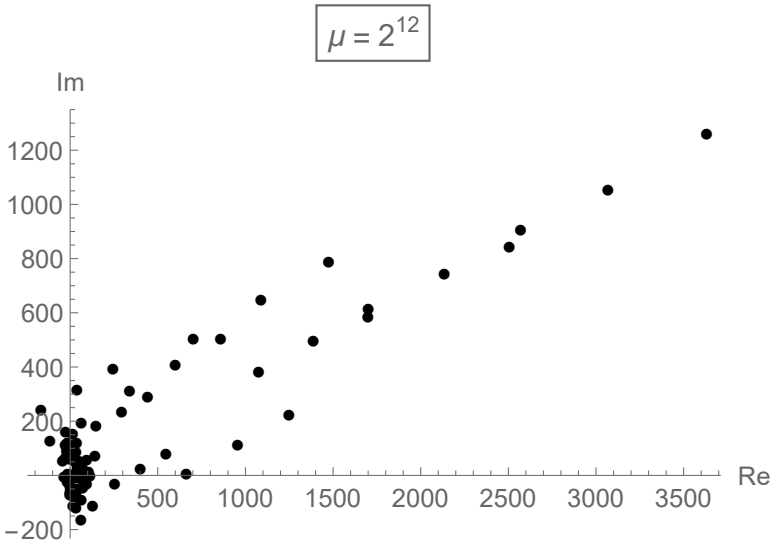

Figure 6.14: All eigenvalues for PHCpack test problem chandra4 with $\mu = 2^{12}$ (complex randomization).

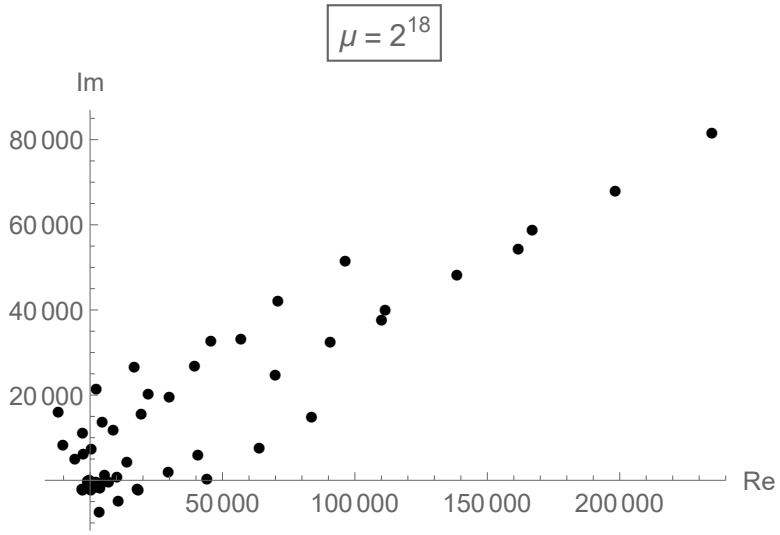

Figure 6.15: All eigenvalues for PHCpack test problem chandra4 with $\mu = 2^{18}$ (complex randomization).

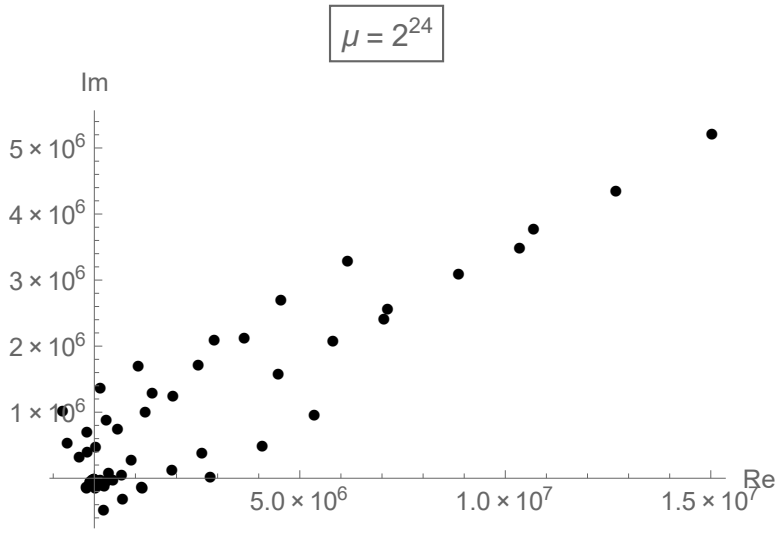

Figure 6.16: All eigenvalues for PHCpack test problem chandra4 with $\mu = 2^{24}$ (complex randomization).

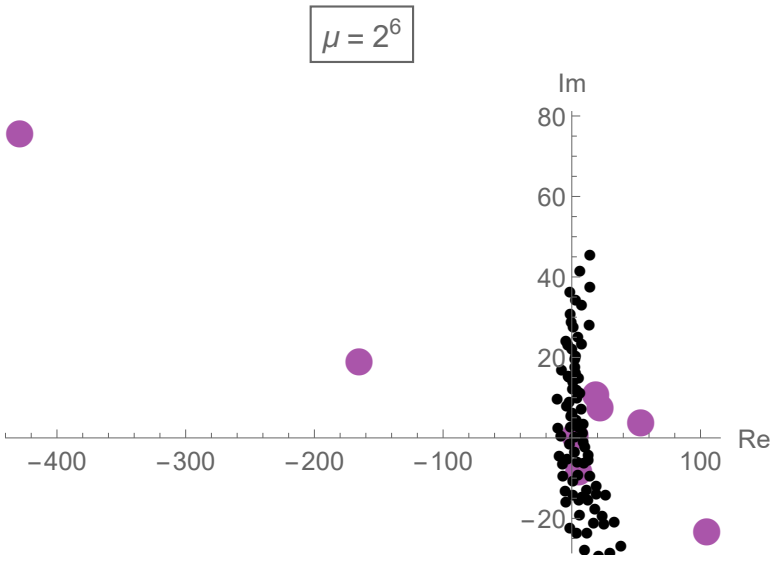

Figure 6.17: Eigenvalues (black) and target solutions (purple) for PHCpack test problem chandra4 with $\mu = 2^6$ (complex randomization).

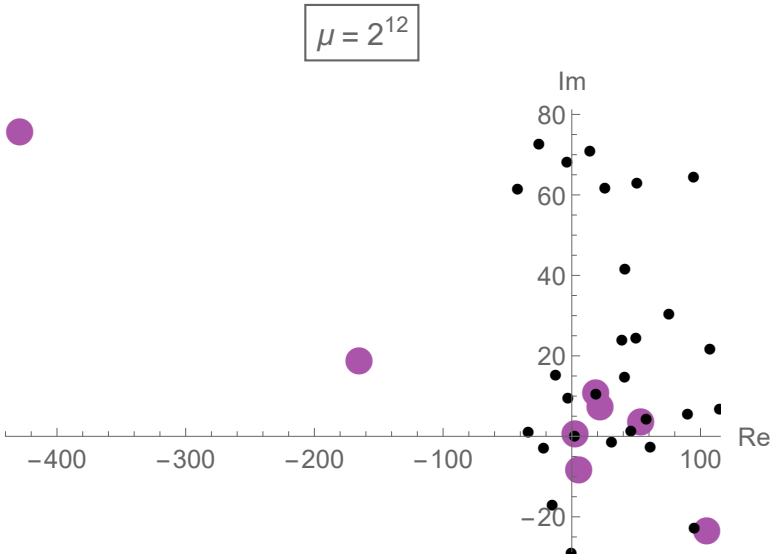

Figure 6.18: Eigenvalues (black) and target solutions (purple) for PHCpack test problem chandra4 with $\mu = 2^{12}$ (complex randomization).

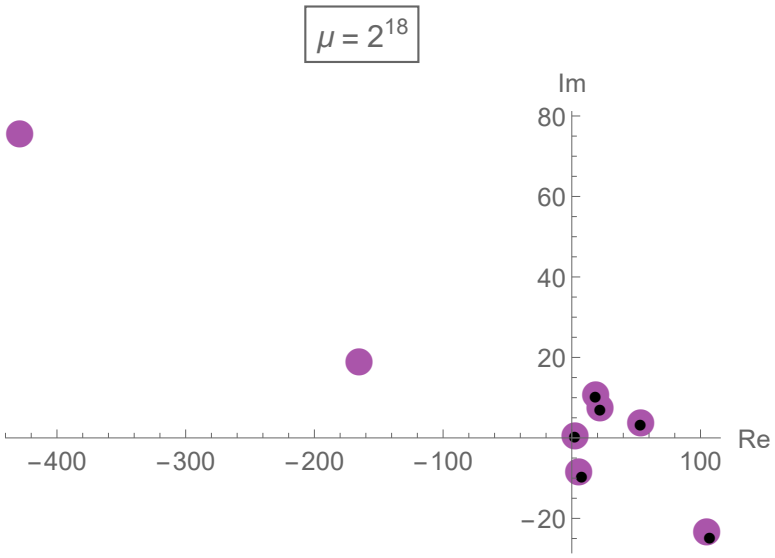

Figure 6.19: Eigenvalues (black) and target solutions (purple) for PHCpack test problem chandra4 with $\mu = 2^{18}$ (complex randomization).

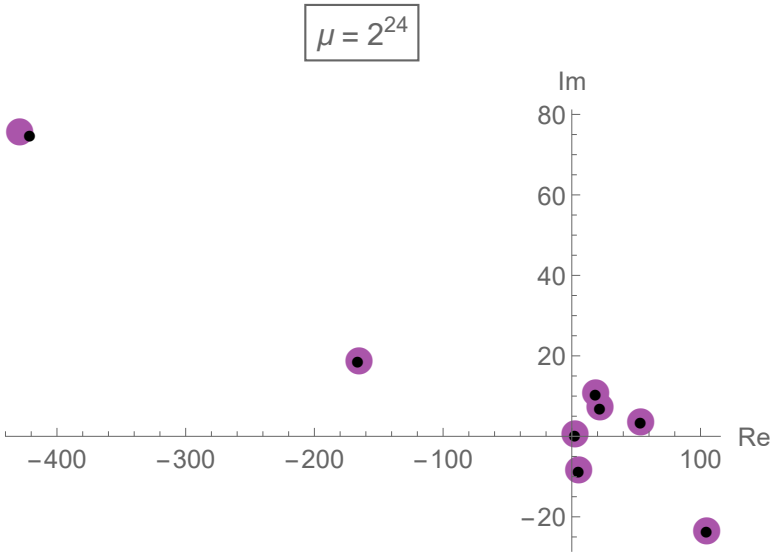

Figure 6.20: Eigenvalues (black) and target solutions (purple) for PHCpack test problem chandra4 with $\mu = 2^{24}$ (complex randomization).

6.3 General Case

The original polynomial system is $\vec{p}$ and the polynomial system with added perturbation terms is $\vec{q}$. Let $\vec{\mathbb{x}}$ be a root of $\vec{p}$. Let $\vec{\mathcal{X}}(\epsilon)$ be a root of $\vec{q}$. Then

$$\vec{q}(\vec{\mathcal{X}}(\epsilon)) = \vec{0}$$

or equivalently

$$\vec{p}(\vec{\mathcal{X}}(\epsilon)) + \epsilon \begin{pmatrix} \mathcal{X}_1(\epsilon)^{d_1+1} \\ \mathcal{X}_2(\epsilon)^{d_2+1} \\ \vdots \\ \mathcal{X}_n(\epsilon)^{d_n+1} \end{pmatrix} = \vec{0}$$

Set

$$\vec{r}(\vec{\mathcal{X}}(\epsilon)) = \begin{pmatrix} \mathcal{X}_1(\epsilon)^{d_1+1} \\ \mathcal{X}_2(\epsilon)^{d_2+1} \\ \vdots \\ \mathcal{X}_n(\epsilon)^{d_n+1} \end{pmatrix}$$

to get

$$\vec{p}(\vec{\mathcal{X}}(\epsilon)) + \epsilon\, \vec{r}(\vec{\mathcal{X}}(\epsilon)) = \vec{0}$$

Now take the partial derivative with respect to ϵ to obtain

$$\vec{p}\,'(\vec{\mathcal{X}}(\epsilon))\vec{\mathcal{X}}'(\epsilon) + \vec{r}(\vec{\mathcal{X}}(\epsilon)) + \epsilon\, \vec{r}\,'(\vec{\mathcal{X}}(\epsilon))\vec{\mathcal{X}}'(\epsilon) = \vec{0}$$

We also have that $\vec{\mathcal{X}}(0) = \vec{\mathbb{x}}$, so setting $\epsilon = 0$ we have

$$\vec{p}\,'(\vec{\mathbb{x}})\vec{\mathcal{X}}'(0) + \vec{r}(\vec{\mathbb{x}}) = \vec{0}$$

Assuming the relevant matrix inverse exists, then

$$\vec{\mathcal{X}}'(0) = -[\vec{p}\,'(\vec{\mathbb{x}})]^{-1}\vec{r}(\vec{\mathbb{x}})$$

If $\vec{\mathcal{X}}(\epsilon)$ is approximated by $\vec{\mathbb{x}} + \epsilon\vec{\mathcal{X}}'(0)$, then $\vec{\mathcal{X}}(\epsilon) \to \vec{\mathbb{x}}$ as $\epsilon \to 0$ because $\vec{\mathcal{X}}'(0)$ does not depend on ϵ.

This approach could not be used for a system such as

$$p_1 = x^2 + y^2 - 1$$
$$p_2 = x - 1$$

which has a root of $x = 1$, $y = 0$. Replacing x with $\mathcal{X}(\epsilon)$ and y with $\mathcal{Y}(\epsilon)$, we have

$$\mathcal{X}(\epsilon)^2 + \mathcal{Y}(\epsilon)^2 - 1$$
$$\mathcal{X}(\epsilon) - 1$$

Take the partial derivative with respect to ϵ:

$$2\mathcal{X}(\epsilon)\mathcal{X}'(\epsilon) + 2\mathcal{Y}(\epsilon)\mathcal{Y}'(\epsilon)$$
$$\mathcal{X}'(\epsilon)$$

Set $\epsilon = 0$, noting that $\mathcal{X}(0) = 1$ and $\mathcal{Y}(0) = 0$. We are left with

$$2\mathcal{X}'(0)$$

$$\mathcal{X}'(0)$$

In matrix form,

$$\begin{pmatrix} 2 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \mathcal{X}'(0) \\ \mathcal{Y}'(0) \end{pmatrix}$$

This matrix is not invertible.

Discussion of the Implicit Function Theorem can be found in [15].

6.4 Convergence Order

Consider the polynomial system

$$p_1 = -18 + 6x + 15y - 5xy - 3y^2 + xy^2$$
$$p_2 = 42 - 18x - 42y + 15xy + 9y^2 - 3xy^2$$

The only solution is $x = 3,\ y = 4$. Although each polynomial has a highest degree term, neither of these terms are univariate, so a perturbation term will be added to each polynomial. Namely, we will have

$$q_1 = p_1 + \epsilon\, x^4$$
$$q_2 = p_2 + \epsilon\, y^4$$

We will use this example to observe the convergence order of the approximations

of the root as ϵ becomes smaller. At each step, ϵ is multiplied by 10^{-1}. The error is computed as the 2-norm of the difference between the computed approximate root and the actual root. The convergence order at each step is computed as

$$\ln\left(\frac{\text{previous error}}{\text{new error}}\right) \Big/ \ln(10)$$

When performing the computations, we used 32 digit precision.

The results are shown in Table 6.1. We observed first order convergence. We were able to obtain close estimates of the root by taking ϵ sufficiently small.

We can also use this example as a specific case of the discussion in the "General Case" section. Again, we will consider the polynomial system

$$p_1 = -18 + 6x + 15y - 5xy - 3y^2 + xy^2$$
$$p_2 = 42 - 18x - 42y + 15xy + 9y^2 - 3xy^2$$

with solution $x = 3,\ y = 4$. We add perturbation terms to obtain

$$q_1 = p_1 + \epsilon\, x^4$$
$$q_2 = p_2 + \epsilon\, y^4$$

Let $\mathcal{X}(\epsilon)$ and $\mathcal{Y}(\epsilon)$ be the coordinates of the root of $\vec{q}$ approximating the root of $\vec{p}$. We have, setting $\vec{q}$ equal to zero,

$$-18 + 6\mathcal{X}(\epsilon) + 15\mathcal{Y}(\epsilon) - 5\mathcal{X}(\epsilon)\mathcal{Y}(\epsilon) - 3\mathcal{Y}(\epsilon)^2 + \mathcal{X}(\epsilon)\mathcal{Y}(\epsilon)^2 + \epsilon\,\mathcal{X}(\epsilon)^4 = 0$$
$$42 - 18\mathcal{X}(\epsilon) - 42\mathcal{Y}(\epsilon) + 15\mathcal{X}(\epsilon)\mathcal{Y}(\epsilon) + 9\mathcal{Y}(\epsilon)^2 - 3\mathcal{X}(\epsilon)\mathcal{Y}(\epsilon)^2 + \epsilon\,\mathcal{Y}(\epsilon)^4 = 0$$

ϵ	$x(\epsilon)$	$y(\epsilon)$	Error	Order
10^{-3}	2.951787172558878	3.850787372344666	$1.56808434 \cdot 10^{-1}$	—
10^{-4}	2.995870928954492	3.983550672759212	$1.69596461 \cdot 10^{-2}$	0.96595263
10^{-5}	2.999594208322072	3.998338521744426	$1.71031485 \cdot 10^{-3}$	0.99634072
10^{-6}	2.999959492082346	3.999833685232742	$1.71176789 \cdot 10^{-4}$	0.99963119
10^{-7}	2.999995949920823	3.999983366852343	$1.71191338 \cdot 10^{-5}$	0.99996309
10^{-8}	2.999999594999208	3.999998336668523	$1.71192793 \cdot 10^{-6}$	0.99999631
10^{-9}	2.999999959499992	3.999999833666685	$1.71192939 \cdot 10^{-7}$	0.99999963
10^{-10}	2.999999995950000	3.999999983366667	$1.71192953 \cdot 10^{-8}$	0.99999996
10^{-11}	2.999999999595000	3.999999998336667	$1.71192955 \cdot 10^{-9}$	1.0000000
10^{-12}	2.999999999959500	3.999999999833667	$1.71192955 \cdot 10^{-10}$	1.0000000
10^{-13}	2.999999999995950	3.999999999983367	$1.71192955 \cdot 10^{-11}$	1.0000000
10^{-14}	2.999999999999595	3.999999999998337	$1.71192955 \cdot 10^{-12}$	1.0000000
10^{-15}	2.999999999999959	3.999999999999834	$1.71192955 \cdot 10^{-13}$	1.0000000
10^{-16}	2.999999999999996	3.999999999999983	$1.71192976 \cdot 10^{-14}$	0.99999995
10^{-17}	3.000000000000000	3.999999999999998	$1.71193700 \cdot 10^{-15}$	0.99999816

Table 6.1: Numerical results of using a pseudo-companion matrix for the system $p_1 = -18 + 6x + 15y - 5xy - 3y^2 + xy^2$ and $p_2 = 42 - 18x - 42y + 15xy + 9y^2 - 3xy^2$ whose only solution is $x = 3,\ y = 4$.

Take the partial derivative with respect to ϵ.

$$\begin{aligned}
&6\mathcal{X}'(\epsilon) + 15\mathcal{Y}'(\epsilon) - 5\mathcal{X}'(\epsilon)\mathcal{Y}(\epsilon) - 5\mathcal{X}(\epsilon)\mathcal{Y}'(\epsilon) - 6\mathcal{Y}(\epsilon)\mathcal{Y}'(\epsilon) \\
&\qquad + \mathcal{X}'(\epsilon)\mathcal{Y}(\epsilon)^2 + 2\mathcal{X}(\epsilon)\mathcal{Y}(\epsilon)\mathcal{Y}'(\epsilon) + \mathcal{X}(\epsilon)^4 + 4\epsilon\,\mathcal{X}(\epsilon)^3\mathcal{X}'(\epsilon) = 0 \\
&-18\mathcal{X}'(\epsilon) - 42\mathcal{Y}'(\epsilon) + 15\mathcal{X}'(\epsilon)\mathcal{Y}(\epsilon) + 15\mathcal{X}(\epsilon)\mathcal{Y}'(\epsilon) + 18\mathcal{Y}(\epsilon)\mathcal{Y}'(\epsilon) \\
&\qquad - 3\mathcal{X}'(\epsilon)\mathcal{Y}(\epsilon)^2 - 6\mathcal{X}(\epsilon)\mathcal{Y}(\epsilon)\mathcal{Y}'(\epsilon) + \mathcal{Y}(\epsilon)^4 + 4\epsilon\,\mathcal{Y}(\epsilon)^3\mathcal{Y}'(\epsilon) = 0
\end{aligned}$$

Set $\epsilon = 0$ and use the fact that $\mathcal{X}(0) = 3$ and $\mathcal{Y}(0) = 4$.

$$\begin{aligned}
&6\mathcal{X}'(0) + 15\mathcal{Y}'(0) - 5\mathcal{X}'(0)(4) - 5(3)\mathcal{Y}'(0) - 6(4)\mathcal{Y}'(0) \\
&\qquad + \mathcal{X}'(0)(4)^2 + 2(3)(4)\mathcal{Y}'(0) + (3)^4 = 0 \\
&-18\mathcal{X}'(0) - 42\mathcal{Y}'(0) + 15\mathcal{X}'(0)(4) + 15(3)\mathcal{Y}'(0) + 18(4)\mathcal{Y}'(0) \\
&\qquad - 3\mathcal{X}'(0)(4)^2 - 6(3)(4)\mathcal{Y}'(0) + (4)^4 = 0
\end{aligned}$$

Simplifying, we have

$$\begin{aligned}
6\mathcal{X}'(0) + 15\mathcal{Y}'(0) - 20\mathcal{X}'(0) - 15\mathcal{Y}'(0) - 24\mathcal{Y}'(0) + 16\mathcal{X}'(0) + 24\mathcal{Y}'(0) + 81 = 0 \\
-18\mathcal{X}'(0) - 42\mathcal{Y}'(0) + 60\mathcal{X}'(0) + 45\mathcal{Y}'(0) + 72\mathcal{Y}'(0) - 48\mathcal{X}'(0) - 72\mathcal{Y}'(0) + 256 = 0
\end{aligned}$$

or

$$\begin{aligned}
2\mathcal{X}'(0) + 0\mathcal{Y}'(0) + 81 = 0 \\
-6\mathcal{X}'(0) + 3\mathcal{Y}'(0) + 256 = 0
\end{aligned}$$

Writing this in matrix form,

$$\begin{pmatrix} 2 & 0 \\ -6 & 3 \end{pmatrix} \begin{pmatrix} \mathcal{X}'(0) \\ \mathcal{Y}'(0) \end{pmatrix} + \begin{pmatrix} 81 \\ 256 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Solve to get

$$\begin{pmatrix} \mathcal{X}'(0) \\ \mathcal{Y}'(0) \end{pmatrix} = - \begin{pmatrix} 2 & 0 \\ -6 & 3 \end{pmatrix}^{-1} \begin{pmatrix} 81 \\ 256 \end{pmatrix} = \begin{pmatrix} -81/2 \\ -499/3 \end{pmatrix}$$

We expect that

$$\begin{pmatrix} \mathcal{X}(\epsilon) \\ \mathcal{Y}(\epsilon) \end{pmatrix} \approx \begin{pmatrix} 3 \\ 4 \end{pmatrix} + \epsilon \begin{pmatrix} -81/2 \\ -499/3 \end{pmatrix}$$

We can now evaluate this approximation for some specific values of ϵ. For example,

$$\begin{pmatrix} \mathcal{X}(10^{-3}) \\ \mathcal{Y}(10^{-3}) \end{pmatrix} \approx \begin{pmatrix} 2.95950 \\ 3.83367 \end{pmatrix}$$

and

$$\begin{pmatrix} \mathcal{X}(10^{-4}) \\ \mathcal{Y}(10^{-4}) \end{pmatrix} \approx \begin{pmatrix} 2.99595 \\ 3.98337 \end{pmatrix}$$

These are consistent with the actual results we obtained.

7 Linear Subsystems

Performing linear eliminations as a preliminary step decreases the size of the matrix, but it is not necessary. We will examine two test problems for which the polynomial system contains a linear polynomial. We will compare the results from using the linear equation to eliminate one of the variables from the polynomial system before building the pseudo-companion matrix with the results from treating the linear polynomial like any other polynomial and adding a perturbation term. We will use the same methodology for the examples in this section that we used in Section 6.

This also helps address the larger question of how the presentation of the problem affects the results. The method should still work even if the polynomial system is given in an alternate but equivalent form. Of course, the size of the basis and therefore the matrix could change. Suppose, for example, that we are constructing a pseudo-companion matrix for a polynomial system whose first polynomial is linear and contains x_1 (any polynomial system with a linear polynomial could be made to take this form by reordering the polynomials and renaming the variables). If we added a perturbation term to this first polynomial, it would have degree 2. Then the basis would be rectangular with bound $< 1, d_2, ..., d_n >$. Monomials in the basis would have the degree of the first variable equal to 0 or 1. If instead we used the first (linear) polynomial to eliminate x_1 from the system, then the basis would be rectangular with bound $< d_2, ..., d_n >$. Now monomials in this basis would not contain x_1. We could think of the degree of x_1 as always being 0 in this basis. Eliminating x_1 reduced the choices for the exponent of x_1 from 0 and 1 to only 0. This halves the size of the basis.

7.1 PHCpack Test Problem "Eco5"

We will begin by examining the polynomial system eco5 as given in the PHCpack materials [21]:

$$(x_1 + x_1x_2 + x_2x_3 + x_3x_4)x_5 - 1$$

$$(x_2 + x_1x_3 + x_2x_4)x_5 - 2$$

$$(x_3 + x_1x_4)x_5 - 3$$

$$x_4x_5 - 4$$

$$x_1 + x_2 + x_3 + x_4 + 1$$

The system has 8 solutions. We will first use the linear polynomial to eliminate an arbitrary variable from the system. Here we eliminate x_1. The pseudo-companion matrix size is 192×192. The results are shown in Figures 7.1, 7.3, and 7.5. In the second set of results, we proceed without any linear eliminations and simply add a perturbation term to all five polynomials. The resulting pseudo-companion matrix is a 384×384 matrix. The results are shown in Figures 7.2, 7.4, and 7.6. Performing a linear elimination reduces the size of the matrix and the number of spurious eigenvalues. Satisfactory results are attained by $\mu = 2^{18}$ regardless of the treatment of the linear polynomial.

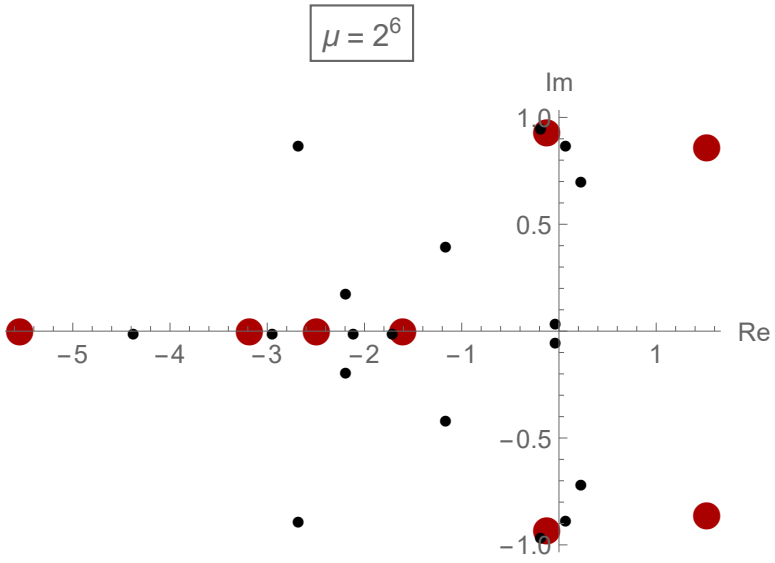

Figure 7.1: Eigenvalues (black) and target solutions (red) for PHCpack test problem eco5 with $\mu = 2^6$ (eliminate x_1).

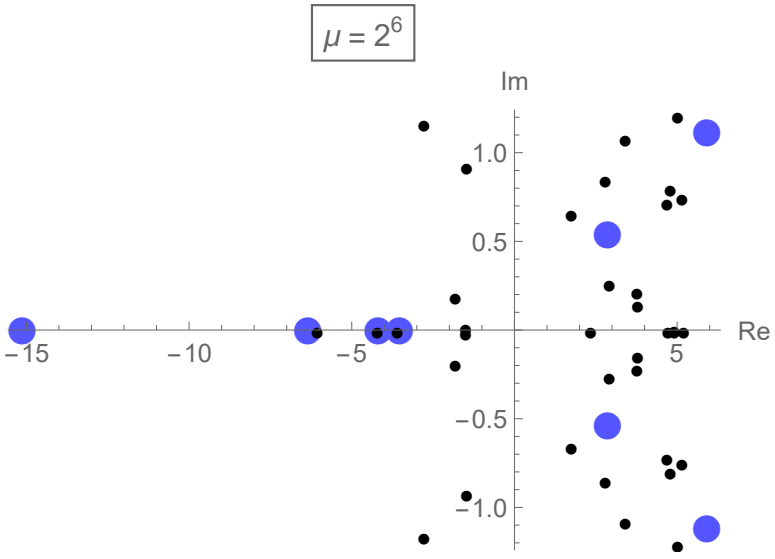

Figure 7.2: Eigenvalues (black) and target solutions (blue) for PHCpack test problem eco5 with $\mu = 2^6$ (no linear eliminations).

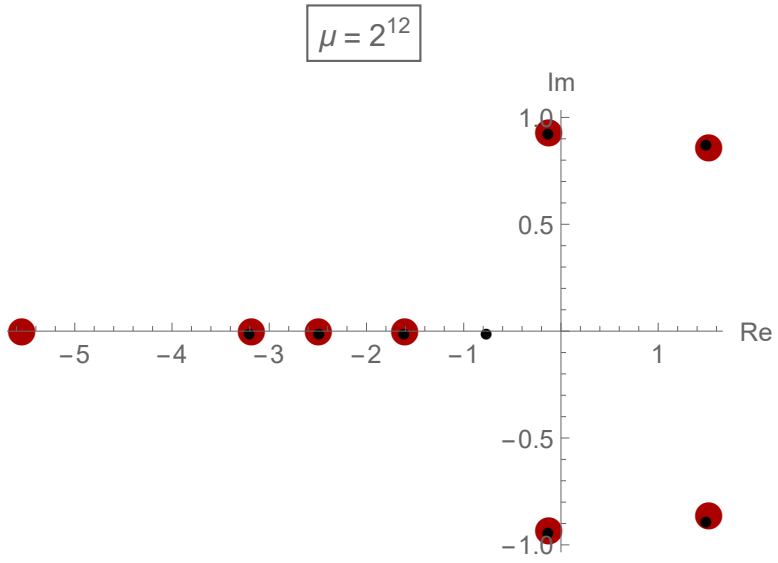

Figure 7.3: Eigenvalues (black) and target solutions (red) for PHCpack test problem eco5 with $\mu = 2^{12}$ (eliminate x_1).

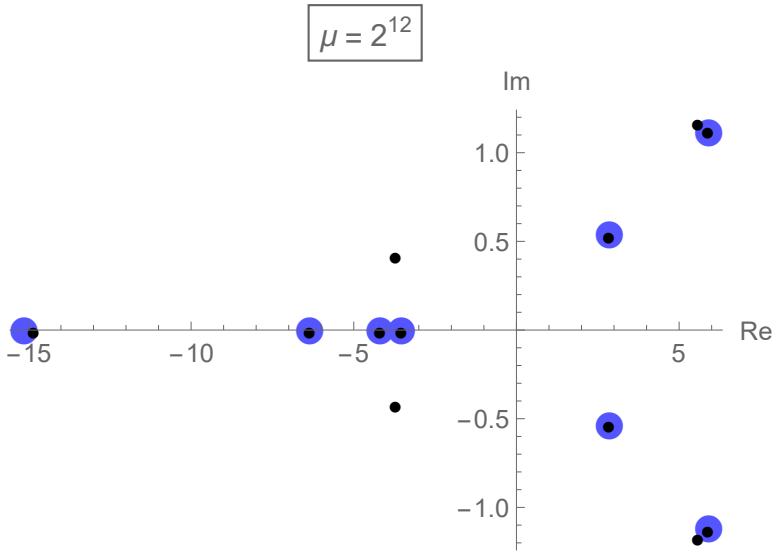

Figure 7.4: Eigenvalues (black) and target solutions (blue) for PHCpack test problem eco5 with $\mu = 2^{12}$ (no linear eliminations).

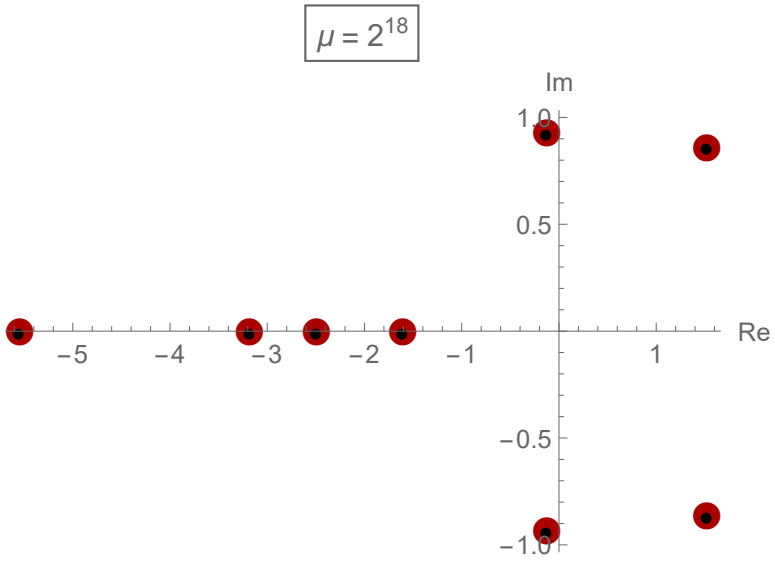

Figure 7.5: Eigenvalues (black) and target solutions (red) for PHCpack test problem eco5 with $\mu = 2^{18}$ (eliminate x_1).

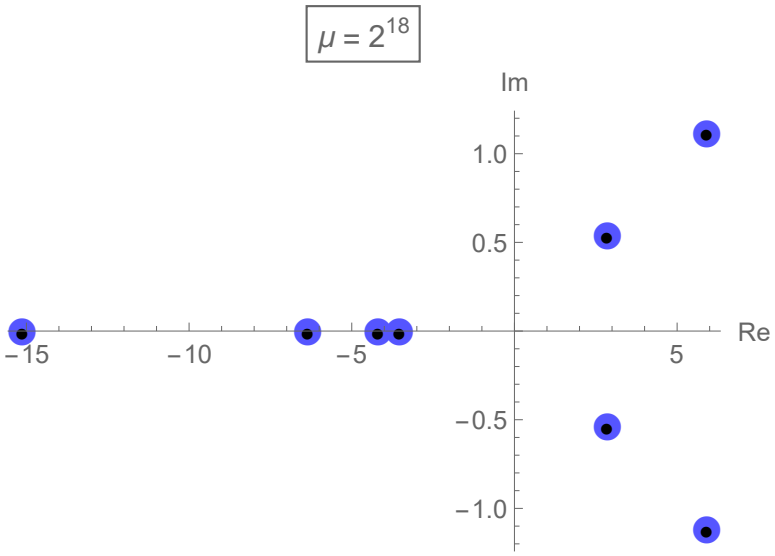

Figure 7.6: Eigenvalues (black) and target solutions (blue) for PHCpack test problem eco5 with $\mu = 2^{18}$ (no linear eliminations).

7.2 PHCpack Test Problem "Gaukwa2"

We will again use a PHCpack problem, this time the system gaukwa2 [21]:

$$w_1 + w_2 - 0.998250904334731 + 0.0591196413630250i$$
$$w_1x_1 + w_2x_2 - 0.892749639148806 + 0.450553084330444i$$
$$w_1x_1^2 + w_2x_2^2 + 0.160088552022675 + 0.987102657027770i$$
$$w_1x_1^3 + w_2x_2^3 - 0.725369971319578 + 0.688359211972815i$$

For the first set of results, we use the linear equation to eliminate w_1 from the polynomial system before proceeding with the pseudo-companion matrix build. The pseudo-companion matrix size is 60×60. For the second set of results, we do not alter the system. Now the pseudo-companion matrix size is 120×120. The results for increasing values of μ are shown in Figures 7.7 through 7.14. With both approaches, suitable results are achieved around $\mu = 2^{18}$ to $\mu = 2^{24}$. Using a linear polynomial to eliminate a variable from the system decreases the size of the matrix, but otherwise the results are comparable. Our results to date suggest that, other than matrix size, the performance of the pseudo-companion matrix method is relatively independent of the form in which the polynomial system is provided.

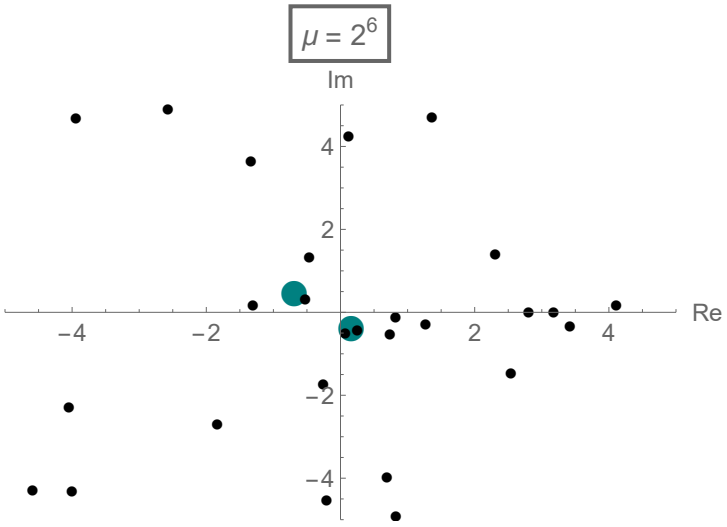

Figure 7.7: Eigenvalues (black) and target solutions (green) for PHCpack test problem gaukwa2 with $\mu = 2^6$ (eliminate w_1).

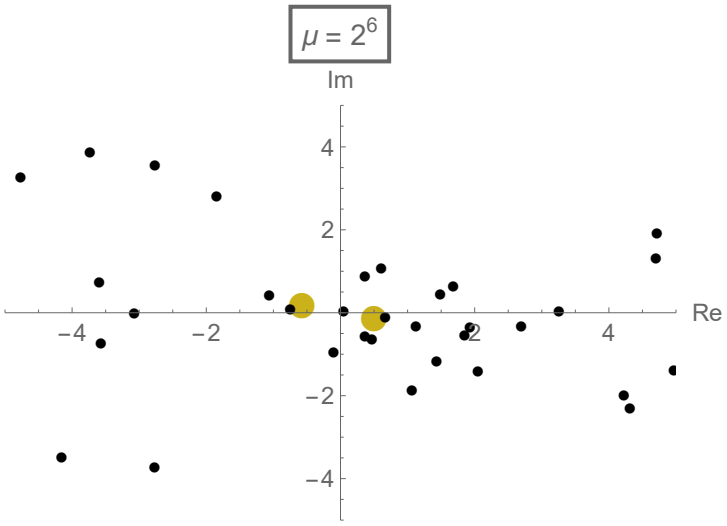

Figure 7.8: Eigenvalues (black) and target solutions (gold) for PHCpack test problem gaukwa2 with $\mu = 2^6$ (no linear eliminations).

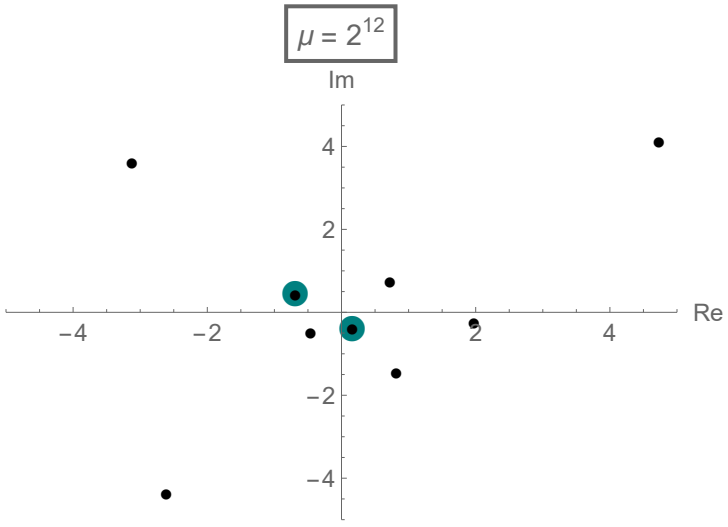

Figure 7.9: Eigenvalues (black) and target solutions (green) for PHCpack test problem gaukwa2 with $\mu = 2^{12}$ (eliminate w_1).

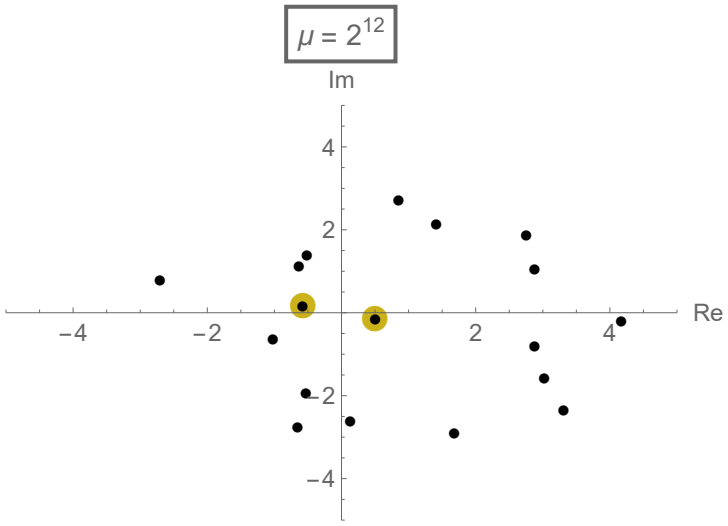

Figure 7.10: Eigenvalues (black) and target solutions (gold) for PHCpack test problem gaukwa2 with $\mu = 2^{12}$ (no linear eliminations).

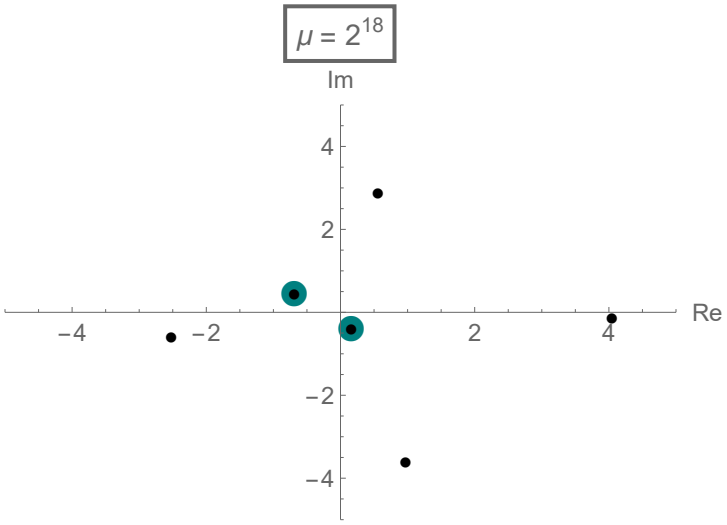

Figure 7.11: Eigenvalues (black) and target solutions (green) for PHCpack test problem gaukwa2 with $\mu = 2^{18}$ (eliminate w_1).

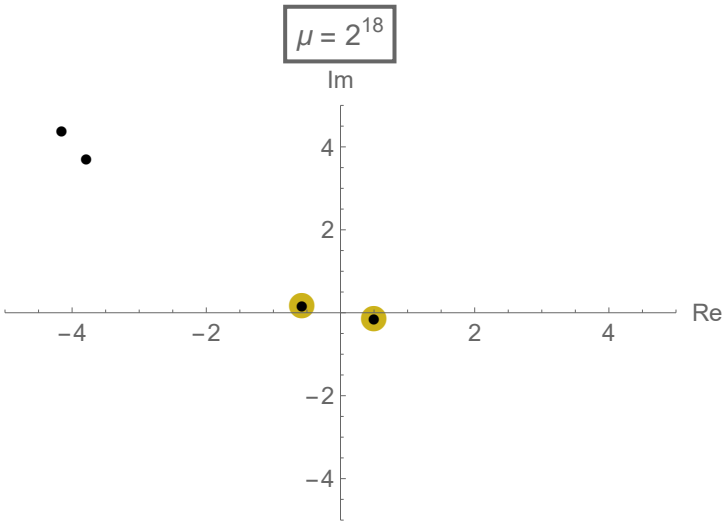

Figure 7.12: Eigenvalues (black) and target solutions (gold) for PHCpack test problem gaukwa2 with $\mu = 2^{18}$ (no linear eliminations).

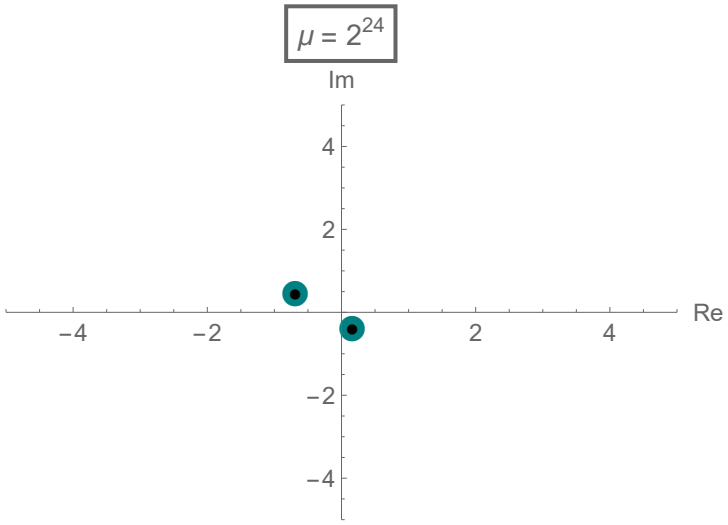

Figure 7.13: Eigenvalues (black) and target solutions (green) for PHCpack test problem gaukwa2 with $\mu = 2^{24}$ (eliminate w_1).

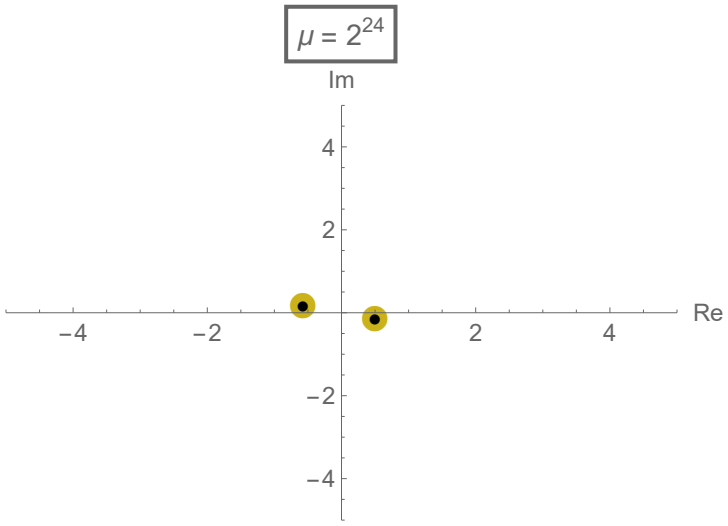

Figure 7.14: Eigenvalues (black) and target solutions (gold) for PHCpack test problem gaukwa2 with $\mu = 2^{24}$ (no linear eliminations).

8 Conclusion and Future Work

We have introduced the pseudo-companion matrix for finding roots of polynomial systems. In some cases, a perturbation of the polynomial system is used for the matrix construction. We have explored the process of constructing the matrix and the results of the computation through a series of examples.

At this stage, the code was written to be straightforward. We have not attempted to make the pseudo-companion matrix code as fast as possible. This would be useful for comparing the pseudo-companion matrix method for root finding to existing methods.

Our interest in our new method relates to the possibility of only finding eigenvalues in a specified region of the complex plane, for example using a non-Hermitian FEAST method [11]. For this introductory work we have used the standard eigensolvers.

The eigenvectors for our problem have a particular structure. This is even present in the single variable case, in that the standard companion matrix has eigenvectors of the form $< 1, \lambda, \lambda^2, ..., \lambda^{n-1} >$. This is of less interest when finding the roots of a single variable polynomial, since the roots are given by the eigenvalues and the eigenvectors do not need to be computed. In the multivariable case the eigenvectors are necessary for obtaining the coordinates of the roots. This raises the question of whether this structure can be utilized to improve the efficiency of the eigensystem computation. There is certainly additional information available which the current eigensolver is not taking in to account.

9 References

[1] Daniel J. Bates, Andrew J. Newell, and Matthew Niemerg. "BertiniLab: A MATLAB interface for solving systems of polynomial equations". In: *Numerical Algorithms* 71 (1 Jan. 2016), pp. 229–244. ISSN: 1572-9265. DOI: 10.1007/s11075-015-0014-6. URL: https://doi.org/10.1007/s11075-015-0014-6.

[2] Daniel J. Bates et al. *Bertini: Software for Numerical Algebraic Geometry*. Available at bertini.nd.edu with permanent doi: dx.doi.org/10.7274/R0H41PB5. 2013.

[3] Daniel J. Bates et al. *Numerically Solving Polynomial Systems with Bertini*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2013.

[4] Louis Brand. "The Companion Matrix and Its Properties". In: *The American Mathematical Monthly* 71.6 (1964), pp. 629–634. URL: http://www.jstor.org/stable/2312322.

[5] Miroslav Fiedler. "A note on companion matrices". In: *Linear Algebra and its Applications* 372 (2003), pp. 325–331. DOI: https://doi.org/10.1016/S0024-3795(03)00548-2. URL: http://www.sciencedirect.com/science/article/pii/S0024379503005482.

[6] A. Gierer and H. Meinhardt. "A theory of biological pattern formation". In: *Kybernetik* 12.1 (Dec. 1972), pp. 30–39. DOI: https://doi.org/10.1007/BF00289234. URL: https://link.springer.com/article/10.1007/BF00289234.

[7] Yun Guan and Jan Verschelde. "PHClab: A MATLAB/Octave Interface to PHCpack". In: *Software for Algebraic Geometry*. Ed. by Michael Stillman, Jan Verschelde, and Nobuki Takayama. New York, NY: Springer New York, 2008, pp. 15–32. ISBN: 978-0-387-78133-4. DOI: 10.1007/978-0-387-78133-4_2. URL: https://doi.org/10.1007/978-0-387-78133-4_2.

[8] Wenrui Hao et al. "Multiple stable steady states of a reaction-diffusion model on zebrafish dorsal-ventral patterning". In: *Discrete and Continuous Dynamical Systems - Series S* 4.6 (2011), pp. 1413–1428. ISSN: 1937-1632. DOI: 10.3934/dcdss.2011.4.1413. URL: http://aimsciences.org/journals/displayArticlesnew.jsp?paperID=5810.

[9] Jonathan D. Hauenstein and Frank Sottile. "Algorithm 921: alphaCertified: Certifying Solutions to Polynomial Systems". In: *ACM Transactions on Mathematical Software* 38.4 (Aug. 2012), 28:1–28:20. DOI: 10.1145/2331130.2331136. URL: http://doi.acm.org/10.1145/2331130.2331136.

[10] The MathWorks Inc. *Polynomial roots - MATLAB roots*. 2017. URL: https://www.mathworks.com/help/matlab/ref/roots.html (visited on 09/21/2017).

[11] James Kestyn, Eric Polizzi, and Ping Tak Peter Tang. "Feast Eigensolver for Non-Hermitian Problems". In: *SIAM Journal on Scientific Computing* 38.5 (2016), S772–S799. DOI: 10.1137/15M1026572. URL: https://doi.org/10.1137/15M1026572.

[12] Randall LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007. ISBN: 9780898716290.

[13] Cyrus Colton MacDuffee. *The Theory of Matrices*. Corrected Reprint of First Edition. New York, NY, USA: Chelsea Publishing Company, 1946.

[14] Dinesh Manocha. "Computing Selected Solutions of Polynomial Equations". In: *Proceedings of the International Symposium on Symbolic and Algebraic Computation*. ISSAC '94. Oxford, United Kingdom: ACM, 1994, pp. 1–8. ISBN: 0-89791-638-7. DOI: 10.1145/190347.190349. URL: http://doi.acm.org/10.1145/190347.190349.

[15] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. 2nd ed. Springer Series in Operations Research and Financial Engineering. Springer-Verlag New York, 2006. ISBN: 978-0-387-30303-1.

[16] Karen M. Page, Philip K. Maini, and Nicholas A.M. Monk. "Complex pattern formation in reaction-diffusion systems with spatially varying parameters". In: *Physica D* 202.1 (2005), pp. 95–115. DOI: https://doi.org/10.1016/j.physd.2005.01.022. URL: http://www.sciencedirect.com/science/article/pii/S0167278905000527.

[17] Hans J. Stetter. *Numerical Polynomial Algebra*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2004.

[18] Bernd Sturmfels. "What is a Gröbner Basis?" In: *Notices of the American Mathematical Society* 52.10 (Nov. 2005), pp. 1199–1200. URL: http://www.ams.org/journals/notices/200510/what-is.pdf.

[19] A. M. Turing. "The Chemical Basis of Morphogenesis". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 237.641 (Aug. 1952), pp. 37–72. DOI: 10.1098/rstb.1952.0012. URL: http://rstb.royalsocietypublishing.org/content/237/641/37.

[20] Jan Verschelde. "Algorithm 795: PHCpack: A General-Purpose Solver for Polynomial Systems by Homotopy Continuation". In: *ACM Transactions on Mathematical Software* 25.2 (June 1999), pp. 251–276. DOI: 10.1145/317275.317286. URL: http://doi.acm.org/10.1145/317275.317286.

[21] Jan Verschelde. *The Test Database of Polynomial Systems*. 1999. URL: http://homepages.math.uic.edu/~jan/PHCpack/node10.html.

[22] Michael Peretzian Williams. "Solving Polynomial Equations Using Linear Algebra". In: *Johns Hopkins APL Technical Digest* 28.4 (2010), pp. 354–363. URL: http://www.jhuapl.edu/techdigest/TD/td2804/Williams.pdf.

[23] Wolfram. *High-Performance Numeric Solution of Polynomial Systems*. URL: https://www.wolfram.com/mathematica/new-in-10/enhanced-algebraic-computation/high-performance-numeric-solution-of-polynomial-sy.html (visited on 09/21/2017).

[24] Alden H. Wright. "Finding all solutions to a system of polynomial equations". In: *Mathematics of Computation* 44.169 (Jan. 1985), pp. 125–133. DOI: https://doi.org/10.1090/S0025-5718-1985-0771035-4. URL: http://www.ams.org/journals/mcom/1985-44-169/S0025-5718-1985-0771035-4/.

A Steady State Solutions Code

Pseudo-companion matrix subfunctions

```
BuildBasis[varCount0_] :=
  Module[{varCount = varCount0, list},
   list = Tuples[{0, 1}, varCount];
   Flatten[Map[Reverse, SplitBy[SortBy[list, Total], Total]], 1]
  ];

(* convert from {1,2,0} to xy^2 *)
MonomialView[basis0_, vars0_] :=
  Module[{basis = basis0, vars = vars0, mview},
   mview = Table[
     FromCoefficientRules[{Rule[basis[[i]], 1]}, vars], {i, Length[basis]}];
   mview
  ];

Replacer[element0_, vars0_, replacements0_] :=
  Module[{element = element0, vars = vars0, replacements = replacements0},
   While[Max[Exponent[element, vars]] > 1,
    element = Expand[element /. replacements]];
   element
  ];

SparseIndices[multiplied0_, row0_, spots0_, vars0_] :=
 Module[{multiplied = multiplied0,
   row = row0, vars = vars0, spots = spots0, crules, elements},
  crules = CoefficientRules[multiplied, vars];
  elements = crules /. spots;
  Table[{row, elements[[i, 1]]} → elements[[i, 2]], {i, Length[elements]}]
 ]
```

```
BuildMatrix[variable0_, spots0_, vars0_, basis0_, replacements0_] :=
  Module[{variable = variable0, spots = spots0, vars = vars0,
    basis = basis0, replacements = replacements0, multiplied, matrix},
   multiplied = basis * variable;
   multiplied =
    Table[Replacer[multiplied[[i]], vars, replacements], {i, Length[multiplied]}];
   matrix = Table[SparseIndices[multiplied[[i]], i, spots, vars],
     {i, Length[multiplied]}];
   matrix = SparseArray[Flatten[matrix]];
   matrix
  ];
```

Define the polynomial system Q

(used for refinement of matrix roots or verifying results using an outside polynomial solver)

```
BuildQ[M0_] :=
 Module[{M = M0, Q},
  h = 1/M;
  Q = Table[0, 2 M + 2];
  Do[
   Q[[j]] = (r + b) v[j] - a u[j] v[j] +
     c 1/h^2 (u[j - 1] - 2 u[j] + u[j + 1]) v[j] - d/h^2 (v[j - 1] - 2 v[j] + v[j + 1]);
   Q[[j + M - 1]] = 1/r (b v[j] - d 1/h^2 (v[j - 1] - 2 v[j] + v[j + 1])) - u[j]^2
   , {j, 1, M - 1}];
  Q[[2 M - 1]] = 3 u[0] - 4 u[1] + u[2];
  Q[[2 M]] = 3 u[M] - 4 u[M - 1] + u[M - 2];
  Q[[2 M + 1]] = 3 v[0] - 4 v[1] + v[2];
  Q[[2 M + 2]] = 3 v[M] - 4 v[M - 1] + v[M - 2];
  Q
 ]
```

Define the modified polynomial system P
(used for the pseudo-companion matrix)

```
BuildP[M0_] :=
 Module[{M = M0, P},
  h = 1/M;
  P = Table[0, 2 M - 2];
  Do[
   P[[j]] = (r + b) v[j] - a u[j] v[j] +
     c 1/h^2 (u[j - 1] - 2 u[j] + u[j + 1]) v[j] - d/h^2 (v[j - 1] - 2 v[j] + v[j + 1]);
   P[[j + M - 1]] = 1/r (b v[j] - d 1/h^2 (v[j - 1] - 2 v[j] + v[j + 1]))
   , {j, 1, M - 1}];
  P = P /. {u[0] → (4/3 u[1] - 1/3 u[2]), u[M] → (4/3 u[M - 1] - 1/3 u[M - 2]),
     v[0] → (4/3 v[1] - 1/3 v[2]), v[M] → (4/3 v[M - 1] - 1/3 v[M - 2])};
  P
 ]
```

Pseudo-companion matrix build

```
PseudoCompanionMatrix[M0_, P0_] :=
 Module[{M = M0, P = P0, vars, listBasis,
   basis, spots, replacements, coefficients, matrices},
  vars = Join[Table[u[i], {i, M - 1}], Table[v[i], {i, M - 1}]];
  listBasis = BuildBasis[2 M - 2];
  basis = MonomialView[listBasis, vars];
  spots = Table[listBasis[[i]] → i, {i, 2^(2 M-2)}];
  replacements = Join[Table[v[j]^2 → Expand[1/ϵ P[[j]]], {j, M - 1}],
    Table[u[j]^2 → Expand[ P[[j + M - 1]]], {j, M - 1}]];
  coefficients = Table[RandomReal[], 2 M - 2];
  matrices =
   Table[BuildMatrix[vars[[i]], spots, vars, basis, replacements], {i, 2 M - 2}];
  Sum[coefficients[[i]] * matrices[[i]], {i, 2 M - 2}]
 ]
```

Enter numerical values for parameters

```
Numericize[] :=
 Module[{a, b, c, d, r, ϵ},
  a = 0.25;
  b = 2.0;
  c = 0.001 * 0.027;
  d = 0.027;
  r = 0.03;
  {a, b, c, d, r} = Rationalize[{a, b, c, d, r}];
  ϵ = 1/2^4;
  {a, b, c, d, r, ϵ}
 ]
```

Input eigenvectors of pseudo-companion matrix. Extract roots, screen, and refine.

```
EigensystemToRoots[evals0_, evecs0_, M0_, ϵ0_, Q0_] :=
 Module[{evals = evals0, evecs = evecs0, M = M0, ϵ = ϵ0, Q = Q0, realEvecs,
   realEvals, sensibleRoots, sensibleEvals, root, sols, solEvals, u0, uM,
   v0, vM, Qvars, refinedSols, point, posRefinedSols, posRefinedEvals},

  (* keep eigenvectors whose eigenvalues are real and positive *)
  realEvecs = {};
  realEvals = {};
  Do[
   If[0 < Re[evals[[i]]] && Abs[Im[evals[[i]]]] < 0.01, {realEvecs =
      Join[realEvecs, {evecs[[i]]}], realEvals = Join[realEvals, {evals[[i]]}]}]
   , {i, Length[evals]}];

  (* scale the eigenvector and take the relevant entries *)
  (* keep positive real solutions (with some tolerance) *)
  sensibleRoots = {};
  sensibleEvals = {};
  Do[
   root = realEvecs[[i]] / realEvecs[[i, 1]];
   root = root[[2 ;; 2 M - 1]];
   If[Min[Re[root]] > -0.01,
    {sensibleRoots = Join[sensibleRoots, {root}], sensibleEvals =
      Join[sensibleEvals, {realEvals[[i]]}]}], {i, Length[realEvecs]}];

  (* add the boundary points back in;
  keep solutions with positive boundary points *)
  sols = {};
  solEvals = {};
  Do[
   root = sensibleRoots[[i]];
   u0 = (4/3 root[[1]] - 1/3 root[[2]]);
   uM = (4/3 root[[M - 1]] - 1/3 root[[M - 2]]);
   v0 = (4/3 root[[M]] - 1/3 root[[M + 1]]);
```

```
  vM = (4/3 root[[2 M - 2]] - 1/3 root[[2 M - 3]]);
  If[Min[Re[{u0, uM, v0, vM}]] > -0.01,
   {root = Join[{u0}, root[[1 ;; M - 1]], {uM, v0}, root[[M ;; 2 M - 2]], {vM}],
    sols = Join[sols, {root}], solEvals = Join[solEvals, {sensibleEvals[[i]]}]}]
  , {i, Length[sensibleRoots]}];

 (* refine the solutions *)
 Qvars = Join[Table[u[i], {i, 0, M}], Table[v[i], {i, 0, M}]];
 refinedSols = Table[0, {Length[sols]}];
 Do[
  point = Table[{Qvars[[j]], sols[[i, j]]}, {j, 2 M + 2}];
  refinedSols[[i]] = Qvars /. FindRoot[Q == 0, point, MaxIterations → 6];
  , {i, Length[sols]}];
 refinedSols = Chop[refinedSols];

 posRefinedSols = {};
 posRefinedEvals = {};
 Do[
  If[Min[Re[refinedSols[[i]]]] ≥ 0,
   {posRefinedSols = Join[posRefinedSols, {refinedSols[[i]]}],
    posRefinedEvals = Join[posRefinedEvals, {sensibleEvals[[i]]}]}]
  , {i, Length[refinedSols]}];
 posRefinedEvals = Chop[posRefinedEvals];

 {posRefinedSols, posRefinedEvals}
]
```

Get positive real roots from Bertini output file

```
ReadBertini[M0_] :=
 Module[
  {M = M0, BertiniOutput, numSolsBertini, rootBertini, posRealRootsBertini},

  BertiniOutput = OpenRead[StringJoin[
     "C:\\BertiniRuns\\M", ToString[M], "\\Output\\finite_solutions"]];
  numSolsBertini = Read[BertiniOutput, Number];
  Do[
   rootBertini[j] =
    Table[Read[BertiniOutput, Number] + I Read[BertiniOutput, Number], {i, 2 M + 2}]
   , {j, numSolsBertini}];
  Close[BertiniOutput];

  posRealRootsBertini = {};
  Do[
   If[Max[Abs[Im[rootBertini[j]]]] < 10^-12 &&
     Min[Re[rootBertini[j]]] > -10^-12 && Norm[rootBertini[j]] > 10^-12,
    posRealRootsBertini = Join[posRealRootsBertini, {rootBertini[j]}]]
   , {j, numSolsBertini}];

  Chop[posRealRootsBertini]

 ]
```

Run example

```
RunSteadyState[M0_] :=
 Module[{M = M0, P, start, A, matrixBuildTime, matrixBuildTimeNumerical,
   parameterNames, evaluateParameters, evals, evecs, eigensystemTime,
   Q, refinedSols, refinedEvals, posRealRootsBertini, xmin, xmax},

  Clear[a, b, c, d, r, ϵ];

  Print["\n\nN:  ", M, "\n"];

  P = BuildP[M];

  start = SessionTime[];
  A = PseudoCompanionMatrix[M, P];
  matrixBuildTime = SessionTime[] - start;
  Print["Symbolic matrix build time:         ", matrixBuildTime];

  {a, b, c, d, r, ϵ} = Numericize[];
  start = SessionTime[];
  A = PseudoCompanionMatrix[M, P];
  matrixBuildTimeNumerical = SessionTime[] - start;
  Print["Numerical matrix build time:          ", matrixBuildTimeNumerical];

  A = Normal[A];
  A = N[A, 16];

  start = SessionTime[];
  {evals, evecs} = Eigensystem[A];
  eigensystemTime = SessionTime[] - start;
  Print["Full eigensystem computation time:  ", eigensystemTime, "\n"];

  Q = BuildQ[M];
  {refinedSols, refinedEvals} = EigensystemToRoots[evals, evecs, M, ϵ, Q];

  xmin = 0.95 Min[refinedEvals];
  xmax = 1.05 Max[refinedEvals];

  Print[
```

```
  Show[ListPlot[{Table[{Re[evals[[i]]], Im[evals[[i]]]}, {i, Length[evals]}], Table[
      {Re[refinedEvals[[i]]], Im[refinedEvals[[i]]]}, {i, Length[refinedEvals]}]},
    PlotLegends → {Text[Style["All eigenvalues", 14]],
      Text[Style["Meaningful eigenvalues", 14]]}, ImageSize → Large,
    PlotRange → All, AxesLabel → {"Re", "Im"}, LabelStyle → Medium, PlotStyle →
     {{PointSize[Large], Darker[Blue]}, {PointSize[Large], Orange}}], Graphics[
    Line[{{xmin, -20}, {xmax, -20}, {xmax, 20}, {xmin, 20}, {xmin, -20}}]]]];
 Print[ListPlot[{Table[{Re[evals[[i]]], Im[evals[[i]]]}, {i, Length[evals]}], Table[
     {Re[refinedEvals[[i]]], Im[refinedEvals[[i]]]}, {i, Length[refinedEvals]}]},
   PlotLegends → {Text[Style["All eigenvalues", 14]],
     Text[Style["Meaningful eigenvalues", 14]]},
   ImageSize → Large, PlotRange → {{xmin, xmax}, {-20, 20}},
   AxesLabel → {"Re", "Im"}, LabelStyle → Medium,
   PlotStyle → {{PointSize[Large], Darker[Blue]}, {PointSize[Large], Orange}}]];

 posRealRootsBertini = ReadBertini[M];

 Print[
  "\nNorms of differences between roots found by the pseudo-companion matrix
    method and roots found by Bertini:"];
 If[Length[refinedSols] == Length[posRealRootsBertini],
  {refinedSols = Sort[refinedSols],
   posRealRootsBertini = Sort[posRealRootsBertini],
   Print[Table[Norm[refinedSols[[i]] - posRealRootsBertini[[i]]],
     {i, Length[refinedSols]}]]},
  Print["Error: unequal number of roots found"]];

 {refinedSols, matrixBuildTime, eigensystemTime}
]
```

B Perturbation Terms Code

Pseudo-companion matrix subfunctions

```
DefaultReplacements[vars0_, P0_] :=
  Module[{vars = vars0, P = P0, n, coeffrules, totalDegrees,
    highestTotalDegrees, replacementDegrees, cap, replacements},

   n = Length[vars];
   P = Expand[P];
   coeffrules = CoefficientRules[P, vars];
   coeffrules = coeffrules[[All, All, 1]];
   totalDegrees = Table[Map[Total, coeffrules[[i]]], {i, Length[coeffrules]}];
   highestTotalDegrees = Map[Max, totalDegrees];
   replacementDegrees = Map[# + 1 &, highestTotalDegrees];
   cap = Total[replacementDegrees] - n;
   replacements = {};
   Do[
    Do[
     replacements =
      Join[replacements, {vars[[i]]^replacementDegrees[[i]]+j → Expand[-μ vars[[i]]^j P[[i]]]}]
     , {j, 0, cap - replacementDegrees[[i]]}]
    , {i, n}];

   {replacementDegrees, replacements}
  ];
```

```
BuildBasis[replacementDegrees0_] :=
  Module[{n, replacementDegrees = replacementDegrees0,
    letters, letter, iterators, basis, maxTotal, sortedBasis},

   n = Length[replacementDegrees];
   letters = Table[letter[i], {i, n}];
   iterators = Table[{letters[[i]], replacementDegrees[[i]] - 1, 0, -1}, {i, n}];
   basis = Table[letters, ##] & @@ iterators;
   basis = Flatten[basis, n - 1];

   maxTotal = Max[Map[Total, basis]];
   sortedBasis = {};
   Do[
    Do[
     If[Total[basis[[j]]] == i, sortedBasis = Join[sortedBasis, {basis[[j]]}]]
     , {j, Length[basis]}]
    , {i, 0, maxTotal}];

   sortedBasis
  ];
(* convert from {1,2,0} to xy^2 *)
MonomialView[basis0_, vars0_] :=
  Module[{basis = basis0, vars = vars0, mview},
   mview = Table[
     FromCoefficientRules[{Rule[basis[[i]], 1]}, vars], {i, Length[basis]}];
   mview
  ];
SparseIndices[multiplied0_, row0_, spots0_, vars0_] :=
 Module[{multiplied = multiplied0,
   row = row0, vars = vars0, spots = spots0, crules, elements},
  crules = CoefficientRules[multiplied, vars];
  elements = crules /. spots;
  Table[{row, elements[[i, 1]]} → elements[[i, 2]], {i, Length[elements]}]
 ]
```

```
Replacer[element0_, vars0_, replacementDegrees0_, replacements0_] :=
  Module[{element = element0, vars = vars0,
    replacementDegrees = replacementDegrees0, replacements = replacements0},
   While[Max[Exponent[element, vars] - replacementDegrees] ≥ 0,
    element = Expand[element /. replacements]];
   element
  ];

BuildMatrix[variable0_, spots0_, vars0_,
   basis0_, replacementDegrees0_, replacements0_] :=
  Module[{variable = variable0, spots = spots0, vars = vars0,
    basis = basis0, replacementDegrees = replacementDegrees0,
    replacements = replacements0, multiplied, matrix},
   multiplied = basis * variable;
   multiplied = Table[Replacer[multiplied[[i]], vars,
      replacementDegrees, replacements], {i, Length[multiplied]}];
   matrix = Table[SparseIndices[multiplied[[i]], i, spots, vars],
     {i, Length[multiplied]}];
   matrix = SparseArray[Flatten[matrix]];
   matrix
  ];
```

Pseudo-companion matrix build

```
PseudoCompanionMatrix[vars0_, P0_, replacementDegrees0_, replacements0_] :=
 Module[{vars = vars0, P = P0,
   replacementDegrees = replacementDegrees0, replacements = replacements0,
   n, listBasis, basis, spots, coefficients, matrices, A},
  n = Length[vars];
  listBasis = BuildBasis[replacementDegrees];
  basis = MonomialView[listBasis, vars];
  spots = Table[listBasis[[i]] → i, {i, Length[listBasis]}];
  coefficients = Table[RandomReal[], n];
  matrices = Table[BuildMatrix[vars[[i]], spots,
     vars, basis, replacementDegrees, replacements], {i, n}];

  A = Sum[coefficients[[i]] * matrices[[i]], {i, n}];
  {A, coefficients}
 ]
```

Read roots from Bertini output file

```
ReadBertini[example0_, n0_] :=
 Module[{example = example0, n = n0, BertiniOutput, numSolsBertini, rootsBertini},

  BertiniOutput = OpenRead[StringJoin[
     "C:\\BertiniRuns\\PHCpack", example, "\\Output\\finite_solutions"]];
  numSolsBertini = Read[BertiniOutput, Number];
  rootsBertini = Table[0, numSolsBertini];
  Do[
   rootsBertini[[j]] =
    Table[Read[BertiniOutput, Number] + I Read[BertiniOutput, Number], {i, n}]
   , {j, numSolsBertini}];
  Close[BertiniOutput];

  Chop[rootsBertini]
 ]
```