ON THE COMPUTATION OF EIGENVALUES, SPECTRAL BOUNDS, AND HESSENBERG FORM FOR MATRIX POLYNOMIALS

By

THOMAS ROBERT CAMERON

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
Department of Mathematics

JULY 2016

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of THOMAS ROBERT CAMERON find it satisfactory and recommend that it be accepted.

Michael Tsatsomeros, Ph.D., Chair

David S. Watkins, Ph.D.

Judith McDonald, Ph.D.

Acknowledgements

To my parents,

for your unwavering love.

To my wife,

for your sacrificial spirit.

To my son,

for your unadulterated joy.

ON THE COMPUTATION OF EIGENVALUES, SPECTRAL BOUNDS, AND HESSENBERG FORM FOR MATRIX POLYNOMIALS

Abstract

by Thomas Robert Cameron, Ph.D.
Washington State University
July 2016

Chair: Michael Tsatsomeros

In this dissertation we focus on root-finding methods, such as Laguerre's method, for solving the polynomial eigenvalue problem. Serious consideration is given to the initial conditions and stopping criteria. Cost efficient and accurate strategies for computing eigenvectors, backward error, and condition estimates are given. Applications for both Hessenberg and tridiagonal structure are provided, and it is shown that significant computational savings can be made from both structures.

Surprising results concerning the spectral bounds for unitary matrix polynomials are presented. In addition, a constructive proof is provided for the result that every square matrix polynomial can be reduced to an upper Hessenberg matrix whose entries are rational functions and in special cases polynomials. The determinant of the matrix polynomial is preserved under this transformation, and sufficient conditions are provided for which the Smith form is preserved.

Contents

List of Tables

Abbreviations

EAM	**E**hrlich-**A**berth **M**ethod
GELMPEP	**L**aguerre's **M**ethod applied to the **G**eneral **M**atrix **P**olynomial **E**igenvalue **P**roblem
GTLMPEP	**L**aguerre's **M**ethod applied to the **T**ridiagonal **M**atrix **P**olynomial **E**igenvalue **P**roblem
HSLMPEP	**L**aguerre's **M**ethod applied to the **H**essenberg **M**atrix **P**olynomial **E**igenvalue **P**roblem
LAPACK	**L**inear **A**lgebra **PACK**age
LM	**L**aguerre's **M**ethod
NP	**N**ewton **P**olygon
NR	**N**umerical **R**ange
SLMPEP	**L**aguerre's **M**ethod applied to the **S**calar **P**olynomial **E**igenvalue **P**roblem

Symbols

$\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{m\times n}$	set of real numbers, vectors, and matrices
$\mathbb{C}$, $\mathbb{C}^n$, $\mathbb{C}^{m\times n}$	set of complex numbers, vectors, and matrices
$\|\cdot\|_2$	vector or matrix 2-norm
A^T, A^*	transpose and conjugate transpose
$\mathrm{Nul}(A)$	null space
$\mathrm{span}\{v_1,\ldots,v_n\}$	space spanned by vectors
$\dim(\mathcal{U})$	dimension of a vector space
I	identity matrix
$P(z)$	matrix polynomial
$P^*(z)$	conjugate transpose of matrix polynomial
$\det P(z)$	determinant of matrix polynomial

Chapter 1

Introduction

1.1 Motivation

The *polynomial eigenvalue problem* is the class of problems for finding the eigenvalues, and often eigenvectors, of a matrix polynomial. Special cases the polynomial eigenvalue problem include the problem of finding the roots of a scalar polynomial and the linear eigenvalue problem. The polynomial eigenvalue problem has received a great deal of attention recently. Notable contributions have been made by Dario Bini, Nicholas Higham, Steven Mackey, Volker Mehrmann, Francoise Tisseur, David Watkins, and others. However, as noted by Volker Mehrmann, more research is urgently needed [25]. In this dissertation we will develop methods for solving the polynomial eigenvalue problem for small to medium size matrix polynomials. There are many applications that fall into this category; including applications from ordinary and partial differential equations, control theory, population models, and many more [3, 24]. While we do not focus on large sparse problems directly, there exist methods for solving these problems which rely on solving smaller projected problems [25].

Generally speaking, there are two ways to go about solving the polynomial eigenvalue problem. One can linearize the problem and then rely on the bounty of methods developed for the linear eigenvalue problem, or one can look to attack the problem directly. Many of the popular methods for solving the polynomial eigenvalue problem rely on a linearization of the matrix polynomial. However, there has been a recent push to develop methods that avoid the linearization of the matrix polynomial; our research is focused on such methods.

1.1.1 Linearization based Methods

The standard eigenvalue problem ($Ax = \lambda x$) and generalized eigenvalue problem ($Ax = \lambda Bx$) can be solved by applying a sequence of similarity and equivalence transformations, respectively. The result is that the matrix A can be reduced to Schur form and the matrix pencil (A, B) can be reduced to generalized Schur form. More importantly, the transformations can be done through a product of unitary matrices, thus the transformations are numerically stable. Algorithms to perform this transformation include the implicitly shifted QR and QZ, and were originally developed by John Francis in the late 1950's. For large sparse problems there are Arnoldi and Lanczos methods, implicit restarts, Jacobi-Davidson methods, and many more. A good introduction to these methods can be found in [37, 38].

As a consequence of the plethora of methods for solving the linear eigenvalue problem, it is natural to take a polynomial eigenvalue problem and linearize it. Let $P(z) = \sum_{i=0}^{d} A_i z^i$ be a $(n \times n)$ matrix polynomial of degree d, such that $\det P(z) \neq 0$ for all $z \in \mathbb{C}$. An $(nd \times nd)$ linear matrix polynomial $A + zB$ is called a *linearization* of $P(z)$ if

$$\begin{bmatrix} P(z) & & \\ & & \\ & & I_{n(d-1)} \end{bmatrix} = E(z)(A + Bz)F(z),$$

for some $(nd \times nd)$ matrix polynomials $E(z)$ and $F(z)$ with constant nonzero determinants[16, Ch. 7.2]. As an example of a linearization of $P(z)$ consider the *companion polynomial* defined by

$$C(z) = \begin{bmatrix} I & & & \\ & I & & \\ & & \ddots & \\ & & & A_d \end{bmatrix} z + \begin{bmatrix} 0 & -I & & \\ & & \ddots & \\ & & & -I \\ A_0 & A_1 & \cdots & A_d \end{bmatrix}.$$

The linearization is not unique; in fact, any equivalence transformation of $C(z)$ will result in yet another linearization of $P(z)$.

For our purposes we will keep the idea behind linearization based methods simple. One takes a linearization of the matrix polynomial $P(z)$ and uses any of the methods designed for the linear eigenvalue problem to compute the eigenvalues and eigenvectors.

1.1.2 Polynomial based Methods

Polynomial based methods are those designed to attack the matrix polynomial directly. There is no analogue of the generalized Schur decomposition for nonlinear matrix polynomials; that is, the matrix polynomial $P(z)$ can not be reduced to triangular form via unitary or strict equivalence transformations [33]. Therefore, other methods must be considered. In this dissertation we will discuss root-finding methods, which work by computing the roots of the polynomial $\det P(z)$. There are many other polynomial based methods; including factorization [23], inverse iteration [25], and invariant pairs [4].

1.2 Preliminary material

In this section we will motivate the development of matrix polynomials through the solution of differential equations. Consider the system of n simultaneous homogeneous differential equations of order d given by

$$A_d \frac{d^d}{dt^d}(u_0) + A_{d-1} \frac{d^{d-1}}{dt^{d-1}}(u_0) + \cdots + A_0 u_0 = 0, \tag{1.1}$$

where A_d, A_{d-1}, ..., A_0 are $(n \times n)$ real or complex matrices. The system in (1.1) may now be reduced to to a system of first-order equations by introducing the new variables

$$\begin{aligned} u_1 &= \frac{d}{dt}(u_0) \\ u_2 &= \frac{d}{dt}(u_1) \\ &\vdots \\ u_d &= \frac{d}{dt}(u_{d-1}) \end{aligned}$$

Now the system in (1.1) takes on the form

$$\begin{bmatrix} & I & & \\ & & \ddots & \\ & & & I \\ -A_0 & -A_1 & \cdots & -A_{d-1} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{d-1} \end{bmatrix} = \begin{bmatrix} I & & & \\ & \ddots & & \\ & & I & \\ & & & A_d \end{bmatrix} \frac{d}{dt} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{d-1} \end{bmatrix} \tag{1.2}$$

We will refer to the process of reducing the system (1.1) to the system (1.2) as a linearization. The reason for this becomes clear when we introduce a matrix polynomial. An $(n \times n)$ matrix polynomial of degree d is defined as

$$P(z) = \sum_{i=0}^{d} A_i z^i, \tag{1.3}$$

where $A_i \in \mathbb{C}^{n\times n}$ for $i = 0, 1, \ldots, d$, and $A_d \neq 0$. An $(nd \times nd)$ linear matrix polynomial $A + Bz$ is called a linearization of $P(z)$ if

$$\begin{bmatrix} P(z) & & \\ & & \\ & & I_{n(d-1)} \end{bmatrix} = E(z)(A + Bz)F(z),$$

for some $(nd \times nd)$ matrix polynomials $E(z)$ and $F(z)$ whose determinants are nonzero constants. It is clear that $\det P(z)$ is a constant multiple of $\det(A + Bz)$; therefore, the eigenvalues of $P(z)$ are equal, including multiplicity, to the eigenvalues of its linearization $A + Bz$.

The homogeneous differential equation in (1.1) can be written as $P\left(\frac{d}{dt}\right) u_0(t) = 0$. Moreover, if we let

$$A = \begin{bmatrix} & I & & \\ & & \ddots & \\ & & & I \\ -A_0 & -A_1 & \cdots & -A_{d-1} \end{bmatrix}, \; B = \begin{bmatrix} I & & & \\ & \ddots & & \\ & & I & \\ & & & A_d \end{bmatrix}$$

then $Bz - A$ is a linearization of $P(z)$. Just as we can obtain the eigenvalues of $P(z)$ from the eigenvalues of $Bz-A$, we can obtain solutions to (1.1) from the system (1.2). This popular approach to solving differential equations is discussed in [40, Ch. 1], and is analogous to the linearization based methods for solving the polynomial eigenvalue problem outlined in Section 1.1.1.

This is also the point at which this dissertation and the mainstream techniques will differ. We prefer the elegance of the matrix polynomial over the convenience of its linearization. The remainder of this section is devoted to developing the preliminary theory behind matrix polynomials. This includes, but is not limited to, the Smith form and Jordan chains.

1.2.1 Smith Form

Every matrix polynomial admits the representation

$$P(z) = E(z)D(z)F(z)$$

where $D(z)$ is a diagonal matrix whose diagonal entries are monic scalar polynomials $d_i(z)$ such that $d_i(z)$ is divisible by $d_{i-1}(z)$; $E(z)$ and $F(z)$ are matrix polynomials with constant nonzero determinants [16, S1.1]. The proof of this result relies on elementary row and column operations which are stored in the matrix polynomials $E(z)$ and $F(z)$, respectively. The matrix polynomial $D(z)$ is referred to as the *Smith form* of $P(z)$. The nonzero diagonal entries of $D(z)$ are called the *invariant polynomials* of $P(z)$. Each invariant polynomial can be expressed as a product of linear factors

$$d_i(z) = (z - \lambda_{i1})^{\alpha_{i1}} \cdots (z - \lambda_{i,k_i})^{\alpha_{i,k_i}},$$

for $i = 1, \ldots, r$, where r is the number of invariant polynomials. The factors $(z - \lambda_{ij})^{\alpha_{ij}}$, for $j = 1, \ldots, k_i$ and $i = 1, \ldots, r$ are known as the *elementary divisors* of $P(z)$. The matrix polynomial $P(z)$ is *regular* if $\det P(z) \neq 0$, this is equivalent to the number of invariant polynomials $r = n$.

For every $\lambda \in \mathbb{C}$, a regular matrix polynomial $P(z)$ admits the following representation, known as the *local Smith form*

$$P(z) = E_\lambda(z) \begin{bmatrix} (z-\lambda)^{k_1} & & 0 \\ & \ddots & \\ 0 & & (z-\lambda)^{k_n} \end{bmatrix} F_\lambda(z)$$

where the determinant of $E_\lambda(\lambda)$ and $F_\lambda(\lambda)$ are nonzero, and $k_1 \leq \cdots \leq k_n$ are nonnegative integers called *partial multiplicities* of $P(z)$ corresponding to λ[16, S1.5]. The nonzero partial multiplicities coincide with the degrees of the elementary divisors of $P(z)$ corresponding to λ.

1.2.2 Jordan Chains

Associated with every matrix polynomial $P(z)$ are the *reversal polynomial*

$$\mathrm{rev}P(z) = z^d P\left(z^{-1}\right) \tag{1.4}$$

and the homogeneous differential equation

$$P\left(\frac{d}{dt}\right) u(t) = 0, \tag{1.5}$$

where $u(t)$ is an n-dimensional vector-valued function to be found. Experience tells us that we seek a solution to (1.5) of the form

$$u(t) = v(t)e^{\lambda t} \tag{1.6}$$

where $\lambda \in \mathbb{C}$; $v(t)$ is an n-dimensional vector valued polynomial of the form

$$v(t) = \frac{t^k}{k!}x_0 + \frac{t^{k-1}}{(k-1)!}x_1 + \cdots + x_k \tag{1.7}$$

with $x_j \in \mathbb{C}^n$ for $j = 0, 1, \ldots, k$ and $x_0 \neq 0$.

Proposition 1.1 ([16, Ch. 1.4])**.** *The vector function $u(t)$ given by (1.6) is a solution to (1.5) if and only if*

$$\sum_{i=0}^{j} P^{(i)}(\lambda)x_{j-i} = 0, \tag{1.8}$$

for $j = 0, 1, \ldots, k$.

The sequence of n-dimensional vectors $x_0, x_1, \ldots, x_k$ ($x_0 \neq 0$) that satisfy (1.8) are called a *Jordan chain* of length $k+1$ for $P(z)$ corresponding to the complex number $\lambda \in \mathbb{C}$. The leading vector x_0 is an *eigenvector*, and the subsequent vectors $x_1, \ldots, x_k$ are known as *generalized eigenvectors*. A $\lambda \in \mathbb{C}$ for which a Jordan chain exists is called an *eigenvalue* of $P(z)$. Moreover, $P(z)$ has an *infinite eigenvalue*, $\lambda = \infty$, if a Jordan chains exists for the matrix polynomial $\mathrm{rev}P(z)$ corresponding to the zero eigenvalue.

A *root polynomial* of $P(z)$ corresponding to $\lambda \in \mathbb{C}$ is defined as an n-dimensional vector polynomial $\phi(z)$, such that $\phi(\lambda) \neq 0$ and $P(\lambda)\phi(\lambda) = 0$. The multiplicity of λ as a root of $P(z)\phi(z)$ is called the *order* of the root polynomial $\phi(z)$. We can write the root polynomial as

$$\phi(z) = \sum_{i=0}^{j} (z-\lambda)^i x_i.$$

Moreover, the vectors $x_0, x_1, \ldots, x_{k-1}$, where k is the order of $\phi(z)$, form a Jordan chain of $P(z)$ corresponding to λ. The converse also holds; that is, if the vectors $x_0, x_1, \ldots, x_{k-1}$ form a Jordan

chain of $P(z)$ corresponding to λ, then the vector polynomial of the form

$$\phi(z) = \sum_{i=0}^{k-1} (z-\lambda)^i x_i + (z-\lambda)^k \psi(z),$$

where $\psi(z)$ is some vector polynomial, is a root polynomial of $P(z)$ of order k corresponding to λ. These root polynomials are useful in the proof of the following proposition.

Proposition 1.2 ([16, Ch. 1.5])**.** *Let $P(z)$ be an $(n \times n)$ matrix polynomial and let $A(z)$ and $B(z)$ be $(n \times n)$ matrix polynomials such that $A(\lambda)$ and $B(\lambda)$ are nonsingular for some $\lambda \in \mathbb{C}$. Then $x_0, x_1, \ldots, x_k$ form a Jordan chain of the matrix polynomial $A(z)P(z)B(z)$ corresponding to λ if and only if the vectors*

$$z_j = \sum_{i=0}^{j} \frac{1}{i!} B^i(z) x_{j-i}, \tag{1.9}$$

for $j = 0, 1, \ldots, k$ form a Jordan chain of $P(z)$ corresponding to λ.

It follows from the previous proposition that the matrix polynomial $P(z)$ and $A(z)P(z)$ have the same set of Jordan chains corresponding to λ, if $\det A(\lambda) \neq 0$.

We now consider the construction of a canonical set of Jordan chains of a regular matrix polynomial $P(z)$, corresponding to a fixed eigenvalue $\lambda \in \mathbb{C}$. First, define

$$\phi_1(z) = \sum_{j=0}^{k_1-1} (z-\lambda)^j x_j^{(1)}$$

to be a root polynomial with the largest order k_1. Note that k_1 does not exceed the multiplicity of λ as a root of $\det P(z)$. Next, define

$$\phi_2(z) = \sum_{j=0}^{k_2-1} (z-\lambda)^j x_j^{(2)}$$

to be a root polynomial with the largest order among all root polynomials whose eigenvector is not a scalar multiple of $x_0^{(1)}$. Note that $k_2 \leq k_1$. If $\phi_1(z), \ldots, \phi_{s-1}(z)$ are already chosen, then let

$$\phi_s(z) = \sum_{j=0}^{k_s-1} (z-\lambda)^j x_j^{(s)}$$

be a root polynomial with the largest order k_s among all root polynomials whose eigenvector is not in the span of the eigenvectors $x_0^{(1)}, \ldots, x_0^{(s-1)}$. We continue this process until the set $\ker P(\lambda)$ has been exhausted. Therefore, r root polynomials are constructed, where $r = \dim \ker P(\lambda)$. In this case, the Jordan chains

$$x_0^{(1)}, \ldots, x_{k_1-1}^{(1)},\ x_0^{(2)}, \ldots, x_{k_2-1}^{(2)},\ x_0^{(r)}, \ldots, x_{k_r-1}^{(r)} \tag{1.10}$$

are said to form a *canonical set* of Jordan chains of $P(z)$ corresponding to λ. Note that the canonical set is not unique; however, the numbers $k_1, \ldots, k_r$ are uniquely defined.

Proposition 1.3 ([16, Ch. 1.6])**.** *Let $P(z)$ be a matrix polynomial. Then the lengths $k_1, \ldots, k_r$ of the Jordan chains in a canonical set of Jordan chains of $P(z)$ corresponding to λ, are exactly the nonzero partial multiplicities of $P(z)$ at λ.*

Since the nonzero partial multiplicities coincide with the degrees of the elementary divisors of $P(z)$, it follows from the previous proposition that if $\lambda \in \mathbb{C}$ is an eigenvalue of $P(z)$ then the lengths of the Jordan chains in a canonical set of Jordan chains of $P(z)$ corresponding to λ are equal to the degrees of the elementary divisors associated with λ.

The following set of vectors

$$y_0^{(i)}, \ldots, y_{s_i-1}^{(i)},\ i = 1, \ldots, q, \tag{1.11}$$

form a canonical set of Jordan chains for $P(z)$ corresponding to $\lambda = \infty$, if they form a canonical set of Jordan chains for the matrix polynomial $\mathrm{rev}P(z)$ corresponding to the zero eigenvalue.

We can write the canonical set of Jordan chains (1.10) in matrix form

$$X = \left[x_0^{(1)} \cdots x_{k_1-1}^{(1)} \cdots x_0^{(r)} \cdots x_{k_r-1}^{(r)}\right],$$

$$J = \mathrm{diag}(J_1, \ldots, J_r),$$

where J_i is a *Jordan block* of size k_i with eigenvalue λ and J is a *Jordan matrix*. The pair of matrices (X, J) is called a *Jordan pair* of $P(z)$ corresponding to λ. The *algebraic multiplicity* of an eigenvalue λ is the size of J, and the *geometric multiplicity* of λ is the number of Jordan blocks. The following theorem provides a characterization of a Jordan pair, where

$$\mathrm{col}\left(XJ^j\right)_{j=0}^{d-1} = \begin{bmatrix} X \\ XJ \\ \vdots \\ XJ^{d-1} \end{bmatrix}.$$

Theorem 1.4 ([16, Ch. 7.1])**.** *Let $P(z)$ be an $(n \times n)$ matrix polynomial of degree d. Let (X, J) be a pair of matrices, where X is a $(n \times \mu)$ matrix and J is a $(\mu \times \mu)$ Jordan matrix with unique eigenvalue λ. Then the following conditions are both necessary and sufficient in order that (X, J) be a Jordan pair of $P(z)$:*

(i) $\det P(z)$ *has a zero λ of multiplicity μ*

(ii) $\mathrm{rank}\ \mathrm{col}\left(XJ^j\right)_{j=0}^{d-1} = \mu$

(iii) $A_0X + A_1XJ + \cdots + A_dXJ^d = 0$

Similarly, we can write the canonical set of Jordan chains (1.11) in matrix form

$$X_\infty = \left[y_0^{(1)} \cdots y_{s_1-1}^{(1)} \cdots y_0^{(q)} \cdots y_{s_q-1}^{(q)} \right],$$

$$J_\infty = \operatorname{diag}(J_{\infty 1}, \ldots, J_{\infty q}),$$

where $J_{\infty j}$ is a Jordan block of size s_j with the zero eigenvalue and J_∞ is a Jordan matrix. The pair of matrices (X_∞, J_∞) are called an *infinite Jordan pair* of $P(z)$. The following theorem provides a characterization of an infinite Jordan pair.

Theorem 1.5 ([16, Ch. 7.1])**.** *Let $P(z)$ be an $(n \times n)$ matrix polynomial of degree d. Let (X, J) be a pair of matrices, where X is a $(n \times \mu)$ matrix and J is a $(\mu \times \mu)$ Jordan matrix with unique eigenvalue $\lambda = 0$. Then the following conditions are both necessary and sufficient in order that (X, J) be an infinite Jordan pair of $P(z)$:*

(i) $\det \operatorname{rev} P(z)$ *has a zero at* $\lambda = 0$ *of multiplicity* μ

(ii) $\operatorname{rank} \operatorname{col} \left(X J^j \right)_{j=0}^{d-1} = \mu$

(iii) $A_0 X J^d + A_1 X J^{d-1} + \cdots + A_d X = 0$

Both Theorem 1.4 and Theorem 1.5 provide insight on possible ways to solve the polynomial eigenvalue problem. Using part (i), one can use an a root-finding algorithm to compute the eigenvalues of a matrix polynomial and then use the null space of $P(\lambda)$ to obtain the corresponding eigenvectors. One could also use parts (ii)-(iii) to obtain the eigenvalues and eigenvectors of a matrix polynomial [4].

1.3 Outline of the Thesis

The outline of this thesis is as follows. In Chapter 2 root-finding methods for solving the matrix polynomial eigenvalue problem are introduced. Specifically, Laguerre's method is developed for use on the matrix polynomial eigenvalue problem and comparisons are made to the Ehrlich-Aberth method. Several strategies for initial estimates to eigenvalues are compared and robust methods for computing eigenvectors, backward error, and condition estimates are described. In addition, surprising results regarding the spectral bounds of unitary matrix polynomials are provided.

In Chapter 3 and Chapter 4 it is shown that significant computational savings can be made when Hessenberg and tridiagonal structures are considered. Applications for both Hessenberg and tridiagonal matrix polynomials are provided in Appendix B. In Chapter 5 we provide a constructive proof of the result that every square matrix polynomial can be reduced to an upper Hessenberg matrix, whose entries are rational functions, and in special cases polynomials. We conclude with Chapter 6 where we summarize our results and discuss directions of future research and open problems.

Chapter 2

Root-Finding Methods

Let $P(z)$ be a regular $(n \times n)$ matrix polynomial of degree d. In this chapter we will develop a numerical method for computing the eigenvalues and eigenvectors of $P(z)$. We say that the eigenvalues of $P(z)$ are *simple* if each eigenvalue is distinct; the eigenvalues of $P(z)$ are *semi-simple* if their algebraic and geometric multiplicities are equal; the eigenvalues of $P(z)$ are *defective* if their algebraic and geometric multiplicities are not equal.

From Theorem 1.4 we know that $\lambda \in \mathbb{C}$ is an eigenvalue of $P(z)$ if and only if λ is a root of $\det P(z)$. We define *root-finding methods* by a class of methods that solve the polynomial eigenvalue problem by finding the roots of $\det P(z)$. In this class of root-finding methods are Newton's method, the Ehrlich-Aberth method, Laguerre's method, and many more. The Ehrlich-Aberth method has already been developed to compute the eigenvalues of a matrix polynomial [7].

In Section 2.1 Laguerre's method is developed for computing the eigenvalues of a matrix polynomial. Accurate and efficient methods for computing eigenvectors, backward error, condition numbers, and a priori estimates to zero and infinite eigenvalues will be provided. Throughout this development,

we will provide several comparisons and give ample reason why we prefer Laguerre's method over the Ehrlich-Aberth method for solving the polynomial eigenvalue problem. In Section 2.2 several strategies for computing initial estimates to the eigenvalues are discussed. A brief theoretical background on Pellet's theorem for matrix polynomials and results on the spectral bounds for unitary matrix polynomials are provided. In Section 2.3 numerical experiments are given to provide a comparison between the Ehrlich-Aberth method and Laguerre's method, as well as a cost analysis of our method.

2.1 Laguerre's Method for the Polynomial Eigenvalue Problem

In this section a numerical method is presented for computing the eigenvalues of a matrix polynomial. This method will be referred to as LMPEP, an acronym for Laguerre's Method applied to the Polynomial Eigenvalue Problem. FORTRAN code for this numerical method is available at `http://www.d.umn.edu/~camer133/code/code.xhtml`. Consistent with LAPACK subroutines, our method has the following naming scheme: XYLMPEP, where X indicates the data type and Y indicates the matrix type. For example, if the matrix polynomial has general double precision coefficients, then the routine used should be DGELMPEP. Throughout Chapters 2-4 we will reference LAPACK subroutines which are used in LMPEP, for more information on these routines see [2] or visit `http://www.netlib.org/lapack/explore-html/index.html`.

A method originating with Edmond Laguerre [19] was designed for polynomials with real zeros. When these are simple zeros this method is known to give strong convergence from any starting value. Laguerre's method has strong virtues, including guaranteed global convergence when all roots are real [1]. In practice, it has been noted that the complex iterations seem as powerful as the real one's [28]. Laguerre's method is used by Numerical Recipes (zroots), and a modified

Laguerre method is used by the NAG F77 Library (C02AFF) to compute the roots of a polynomial. Moreover, Laguerre's method has been applied to the linear eigenvalue problem, both in the monic [28] and non-monic [14] cases. Now, we will apply Laguerre's method to the polynomial eigenvalue problem.

Since $P(z)$ is assumed to be regular, the polynomial $p(z) = \det P(z)$ has at most $N_1 \leq nd$ roots, where $N_1 + N_2 = nd$ and N_2 is the number of infinite eigenvalues. Given an approximation $\lambda \in \mathbb{C}$ to one of the roots of $p(z)$, Laguerre's method uses $p(\lambda)$, $p^{'}(\lambda)$, and $p^{''}(\lambda)$ to obtain a better approximation. Following the development in [28], the derivatives of $\log p(\lambda)$ are defined as

$$S_1(\lambda) = \frac{p^{'}(\lambda)}{p(\lambda)} = \sum_{i=1}^{N_1} \frac{1}{\lambda - r_i}, \tag{2.1}$$

where $r_1, \ldots, r_{N_1}$ are the roots of $p(z)$, and

$$S_2(\lambda) = -\left(\frac{p^{'}(\lambda)}{p(\lambda)}\right)^{'} = \sum_{i=1}^{N_1} \frac{1}{(\lambda - r_i)^2}. \tag{2.2}$$

Then the next approximation is given by

$$\hat{\lambda} = \lambda - \frac{N_1}{S_1 \pm \sqrt{(N_1 - 1)(N_1 S_2 - S_1^2)}}, \tag{2.3}$$

where the sign of the square root is chosen to maximize the magnitude of the denominator. We call $\hat{\lambda}$ the *Laguerre iterate* of λ, and the difference $\hat{\lambda} - \lambda$ is the *Laguerre correction* term.

Theorem 2.1 provides a manageable way to compute (2.1)-(2.3), which avoids the inherent possibility of harmful overflow in the computation of the determinant. The inverse of $P(\lambda)$ appears in this theorem as a symbolic reference to the matrix equation $P(\lambda)X(\lambda) = P^{'}(\lambda)$, which must be

solved, it should not be taken as a literal suggestion to compute the inverse. Moreover, only the diagonal entries of $X(\lambda)^2$ are needed to compute its trace. This is significantly less expensive then computing the matrix product.

Theorem 2.1 ([11]). *Let $\lambda \in \mathbb{C}$ such that $P(\lambda)$ is nonsingular. Then*

$$\frac{p^{'}(\lambda)}{p(\lambda)} = \text{trace}\left(P^{-1}(\lambda)P^{'}(\lambda)\right),$$
$$-\left(\frac{p^{'}(\lambda)}{p(\lambda)}\right)^{'} = \text{trace}\left(\left[P^{-1}(\lambda)P^{'}(\lambda)\right]^2\right) - \text{trace}\left(P^{-1}(\lambda)P^{''}(\lambda)\right)$$

Proof. A proof of the first equation can be found in [7], and relies on Jacobi's formula. For the second equation, define $X(\lambda) = P^{-1}P^{'}(\lambda)$. Then we have $-\left(\frac{p^{'}(\lambda)}{p(\lambda)}\right)^{'} = -\text{trace}\left(X^{'}(\lambda)\right)$. Note that

$$X^{'}(\lambda) = \left(P^{-1}(\lambda)\right)^{'} P^{'}(\lambda) + P^{-1}(\lambda)P^{''}(\lambda),$$

where $\left(P^{-1}(\lambda)\right)^{'} P^{'}(\lambda) = -X(\lambda)^2$. Therefore,

$$-\left(\frac{p^{'}(\lambda)}{p(\lambda)}\right)^{'} = -\text{trace}\left(-X(\lambda)^2 + P^{-1}(\lambda)P^{''}(\lambda)\right)$$

and the result follows. □

In order to compute all the roots of $p(z)$, it is important to avoid unnecessary multiple convergence to the same root. To this end, suppose that the roots $r_1, \ldots, r_k$ have been found, then update (2.1) and (2.2) by subtracting $\sum_{i=1}^{k} \frac{1}{\lambda - r_i}$ and $\sum_{i=1}^{k} \frac{1}{(\lambda - r_i)^2}$, respectively. LMPEP will begin with initial estimates to the eigenvalues of $P(z)$, see Section 2.2. Then, proceeding one at a time, Theorem 2.1 is used to compute the Laguerre correction term and update each approximation until the necessary

stopping criteria has been fulfilled, see Section 2.1.1. Locally, if the root is simple, convergence is cubic; otherwise it is linear.

Laguerre's method experiences cubic convergence one eigenvalue at a time, while the Ehrlich-Aberth method relies on updating every remaining eigenvalue approximation on each iteration in order to obtain cubic convergence. Therefore, the use of Laguerre's method allows for local cubic convergence while taking advantage of eigenvalues which come in complex conjugate pairs. This is not a readily available trait of the Ehrlich-Aberth method. While each Laguerre iteration requires more calculation then the Ehrlich-Aberth iteration, the reduction in the number of iterations to obtain convergence with Laguerre's method more then compensates for the extra calculation, especially when the coefficients of the polynomial are real. A similar observation is made in [28] and numerical evidence is recorded in Section 2.3.1.

2.1.1 Stopping Criteria

An approximate eigenvalue λ to the matrix polynomial $P(z)$ can be updated using Theorem 2.1 to compute (2.1)-(2.3). This computation requires solving the matrix equations

$$P(\lambda)X_1(\lambda) = P^{'}(\lambda),\ P(\lambda)X_2(\lambda) = P^{''}(\lambda).$$

The classical approach to this problem is to first obtain an LU decomposition of $P(\lambda)$ and then use forward and backward substitution to solve for $X_1(\lambda)$ and $X_2(\lambda)$. Moreover, the LU decomposition of $P(\lambda)$ allows for cheap estimates to the backward error of λ to be obtained.

Definition 2.2. For the right eigenpair of the matrix polynomial, the normwise backward error is defined by

$$\eta\left(\lambda,\, x\right) = \min\left\{\epsilon : \left[P(z) + \triangle P(z)\right] x = 0,\, \left\|\triangle A_i\right\|_2 \leq \epsilon \left\|A_i\right\|_2,\, i = 0,\, 1,\, \ldots,\, d\right\}, \tag{2.4}$$

where $\triangle P(z) = \sum_{i=0}^{d} z^i \triangle A_i$.

This definition of the normwise backward error is concerned with a relative measurement of the perturbation in the coefficients of $P(z)$. A similar definition can be given for left eigenpairs. The following theorem gives a cost efficient upper bound on the backward error of an eigenvalue, when the eigenvector has not been computed.

Theorem 2.3 ([11])**.** *If $\lambda \in \mathbb{C}$ is not an eigenvalue of $P(z)$, then the backward error $\eta(\lambda)$ is bounded above by*

$$\frac{\left\|b\right\|_2}{\alpha \left\|P(\lambda)^{-1} b\right\|_2},$$

where $\alpha = \sum_{i=0}^{d} |\lambda|^i \left\|A_i\right\|_2$, and $b \in \mathbb{C}^n$ is any nonzero vector.

Proof. Let $P(z)$ be an $(n \times n)$ matrix polynomial of degree d, and consider the approximate eigenvalue λ. Given a corresponding approximate right eigenvector x, the normwise backward error of this eigenpair is defined in (2.4). By [34, Lemma 3] if an approximate eigenvector has not been computed, then an appropriate measure of the backward error is

$$\eta(\lambda) = \frac{1}{\alpha \left\|P(\lambda)^{-1}\right\|_2},$$

where $\alpha = \sum_{i=0}^{d} |\lambda|^i \|A_i\|_2$. Moreover, given any nonzero vector $b \in \mathbb{C}^n$,

$$\left\|P(\lambda)^{-1}\right\|_2 \geq \frac{\left\|P(\lambda)^{-1}b\right\|_2}{\|b\|_2}$$

and the result follows. □

Once the LU decomposition of $P(\lambda)$ has been obtained solving the matrix equation $P(\lambda)x = b$ is relatively inexpensive. Moreover, the norm of the coefficients of $P(z)$ only need to be computed once, at the beginning of LMPEP. We note that in practice we have observed comparable performance when the 2-norm in α is replaced by the 1-norm.

Given an approximate eigenvalue λ if the following inequality holds

$$\frac{\|b\|_2}{\alpha \|P(\lambda)^{-1}b\|_2} < \epsilon, \tag{2.5}$$

where $b \in \mathbb{C}^n$ is any nonzero vector and ϵ is unit roundoff that depends on the data type, then we stop updating λ. The vector b is selected at random using the LAPACK subroutine XLARNV. This stopping criterion ensures that the backward error in our eigenvalue approximation is small enough. It is useful to add the stopping criterion

$$\left|\hat{\lambda} - \lambda\right| < \tau, \tag{2.6}$$

where $\hat{\lambda}$ is the Laguerre iterate of λ and τ is as predetermined tolerance that depends on unit roundoff ϵ and the moduli of the approximate eigenvalues. If either (2.4) or (2.5) hold, or some maximum number of iterations has been reached, then LMPEP will cease to update the approximate eigenvalue λ. The algorithm for computing the Laguerre correction term, checking stopping criteria,

and updating each eigenvalue approximation will be denoted XGELCORR, and is summarized in Appendix A.

2.1.2 Eigenvectors

Let λ be an approximate eigenvalue of the matrix polynomial $P(z)$ and let the matrices Q and R denote the QR decomposition of $P(\lambda)$. The matrix R has full rank if and only if $P(\lambda)$ has full rank, and the matrix R is rank deficient if and only if at least one of its eigenvalues is zero. In practice the diagonal entries of R will be contaminated by rounding errors and will not be exactly zero. Let τ be some tolerance and $i \in \{1, \ldots, n\}$ be the first index, if there is one, such that $|R(i,i)| < \tau$ holds. Then define the right eigenvector x by

$$x(1:i-1) = -R(1:i-1, 1:i-1)^{-1} R(1:i-1, i),\ x(i) = 1,\ x(i+1:n) = 0. \tag{2.7}$$

From this definition, it follows that $\|Rx\|_2 < \tau$. Therefore,

$$\|P(\lambda)x\|_2 = \|QRx\|_2 < \tau.$$

Given a right eigenpair approximation (λ, x) the goal is to have the normwise backward error be as small as possible. To this end, define the tolerance $\tau = \epsilon\alpha$, where ϵ is unit roundoff depending on the data type and $\alpha = \sum_{i=0}^{d} |\lambda|^i \|A_i\|_2$. Then, if $\|P(\lambda)x\|_2 < \tau$, it follows that the backward error (2.9) is less then ϵ. LMPEP will return the normalized vector $\dfrac{x}{\|x\|_2}$ as the approximate right eigenvector.

A left eigenvector y satisfies $y^*P(\lambda) = 0$ if and only if $R^*Q^*y = 0$. Let $\hat{y} = Q^*y$ and $L = R^*$, then we want to solve $L\hat{y} = 0$. To this end, let $i \in \{1, \ldots, n\}$ be the last index, if there is one, such that

$|L(i,i)| < \tau$. Then define $\hat{y}$ by

$$\hat{y}(1:i-1) = 0,\ \hat{y}(i) = 1,\ \hat{y}(i+1:n) = -L(i+1:n, i+1:n)^{-1} L(i+1:n, i). \tag{2.8}$$

LMPEP will return the normalized vector $\dfrac{Q\hat{y}}{\|Q\hat{y}\|_2}$ as the approximate left eigenvector. Note when computing the right and left eigenvectors the index i is chosen so that the matrix equations (2.7) and (2.8) avoid division by a number smaller then τ.

If $|R(i,i)| < \tau$ for some $i \in \{1, \ldots, n\}$ then the approach described above works well, but this is not guaranteed. Suppose $|R(i,i)| \geq \tau$ for all $i = 1, \ldots, n$, then any potential rank deficiency in the matrix R is not revealed from the diagonal entries of R. A natural second choice is to apply inverse iteration to compute approximate left and right eigenvectors corresponding to the smallest eigenvalue of $P(\lambda)$.

However, it is more reliable to compute the singular vectors corresponding to the smallest singular value of $P(\lambda)$. To this end, we apply inverse iteration to $P(\lambda)^* P(\lambda) = R^* R$ and $P(\lambda) P^*(\lambda) = QRR^*Q^*$ to find the right and left singular vectors corresponding the smallest singular value, respectively.

Let $i \in \{1, \ldots, n\}$ denote the index that minimizes $|R(i,i)|$. Then define x and y as in (2.7) and (2.8) to be initial estimates for inverse iteration applied to R^*R and QRR^*Q^*, respectively. With these initial estimates in hand, we only need a small number of iterations to obtain good right and left eigenvector approximations.

2.1.3 Condition Numbers and Backward Error

Once an approximate eigenvalue λ and corresponding right x and left y eigenvectors have been computed, it is important that LMPEP provide error and condition estimates. By [34, Theorem 1] the normwise backward error for the right eigenpair can be computed as

$$\eta(\lambda, x) = \frac{\|r\|_2}{\alpha \|x\|_2}, \tag{2.9}$$

where $\alpha = \sum_{i=0}^{d} |\lambda|^i \|A_i\|_2$ and $r = P(\lambda)x$. Similar results can be obtained for left eigenpairs. Moreover, by [34, Theorem 5], the normwise condition number of a nonzero simple eigenvalue is given by

$$\kappa(\lambda, P) = \frac{\alpha \|x\|_2 \|y\|_2}{|\lambda| \, |y^* P'(\lambda) x|}. \tag{2.10}$$

For simple zero and infinite eigenvalues, LMPEP will report $\dfrac{1}{|y^* x|}$, where x and y are normalized right and left eigenvectors corresponding to the zero eigenvalues of the matrices A_0 and A_d, respectively.

In summary, once an approximate eigenvalue and corresponding right and left eigenvectors have been computed, LMPEP will return backward error (2.9) and condition number estimates (2.10). The algorithm for computing eigenvectors, backward error, and condition estimates is denoted XGEEVC, and is summarized in Appendix A.

2.2 Initial Estimates

The performance of all root-finding methods is greatly influenced by the quality of the initial estimates. In this section we will review the theoretical background for Rouché and Pellet theorems

for matrix polynomials. The approach we take in developing this material draws closely from the work done in [26]. Pellet's theorem will be used to provide general bounds for the spectrum of a matrix polynomial and surprising results on the location of the spectrum for unitary matrix polynomials. We conclude with a comparison of two strategies for initial estimates for eigenvalues of a matrix polynomial. Throughout this section, let H denote a separable Hilbert space, G an open connected subset of $\mathbb{C}$, and $\mathcal{L}(H)$ the space of bounded linear operators from H to H.

Definition 2.4. A bounded linear operator $A \in \mathcal{L}(H)$ is called *Fredholm* if the range (A) is closed and both the $\dim(\ker A)$ and $\dim(H/\text{range}(A))$ are finite. If H is finite dimensional, then A is always Fredholm.

Let $W : G \to \mathcal{L}(H)$ be an analytic operator function. Then W is said to be *normal* with respect to a curve $\gamma \subset G$, if $W(z)$ is invertible for all $z \in \gamma$ and W is Fredholm on the set of points enclosed by γ.

Theorem 2.5 ([15, p. 205])**.** *Let $W, S : G \to \mathcal{L}(H)$ be analytic operator functions. Assume that W is normal with respect to the simple closed curve $\gamma \subseteq G$. If $\left\|W(z)^{-1}S(z)\right\| < 1$ for all $z \in \gamma$, then $W + S$ is also normal with respect to γ. Moreover, $W + S$ and W have the same number of eigenvalues inside γ, counting multiplicities.*

Note that the norm used in Theorem 2.5 can be any norm induced by a norm on H, and it is easy to see that this result holds for analytic matrix valued functions. We state this result now as a reference for the remainder of this section.

Theorem 2.6 ([26, Theorem 3.2])**.** *Let $A, B : G \to \mathbb{C}^{n\times n}$ be analytic matrix-valued functions. Assume that $A(z)$ is invertible on the simple closed curve $\gamma \subseteq G$. If $\left\|A(z)^{-1}B(z)\right\| < 1$ for all $z \in \gamma$, then $A + B$ and A have the same number of eigenvalues inside γ, counting multiplicities.*

When $n = 1$, Theorem 2.6 is equivalent to the classical Rouché theorem (see [31, p. 391]), which has many applications. For example, Rouché's theorem has been used to prove the fundamental theorem of algebra and provide bounds on the roots of a complex scalar polynomial. With regards to the latter, A.L. Cauchy provided upper and lower bounds in [12] . A more general approach was taken up by M.A. Pellet in [29], this approach was refined by M. Marden in [22], and a generalization of Pellet's theorem was presented by B. Levinger in [21].

Theorem 2.6 yields Pellet and Cauchy bounds for the eigenvalues of a matrix polynomial, which can be found in [26]. We present a detailed proof of these results which was influenced by J.L. Walsh in [35].

Theorem 2.7 (Pellet's Theorem for Matrix Polynomials)**.** *Let $P(z)$ be an $(n\times n)$ matrix polynomial of degree $d \geq 2$, where $A_0 \neq 0$. For each $k \in \{0, 1, \ldots, d\}$ such that A_k is nonsingular, consider the equation*

$$\left\|A_k^{-1}\right\|^{-1} r^k = \sum_{i\neq k} \|A_i\|\, r^i, \tag{2.11}$$

where r is a real positive number and $\|\cdot\|$ is any induced matrix norm.

(i) If $k = 0$ there exists one real positive solution r, and $P(z)$ has no eigenvalues of moduli less than r.

(ii) If $0 < k < d$ there are either no real positive solutions or two real positive solutions $r_1 \leq r_2$. In the latter case, $P(z)$ has no eigenvalues in the annulus $\mathcal{A}_{r_1,r_2}(0) = \{z \in \mathbb{C} : r_1 < |z| < r_2\}$.

(iii) If $k = d$, then there exists one real positive solution R, and $P(z)$ has no eigenvalues of moduli greater than R.

Proof. For each $k \in \{0, 1, \ldots, d\}$ such that A_k is nonsingular, define $F : \mathbb{R} \to \mathbb{R}$ by

$$F(x) = \|A_0\| + \cdots + \|A_{k-1}\| \, x^{k-1} - \left\|A_k^{-1}\right\|^{-1} x^k + \cdots + \|A_d\| \, x^d,$$

and $A(z) = A_k z^k$ and $B(z) = \sum_{i \neq k} A_i z^i$. Let $k = 0$, then $F(x)$ has one sign change and by Descarte's rule of signs (DRS) $F(x)$ has one real positive root r. Since $F(0) < 0$ and $F(r) = 0$, it follows that $F(x) < 0$ for all $x \in [0, r)$ and $\|B(z)\| < \left\|A(z)^{-1}\right\|^{-1}$ for all $|z| < r$. Let $\epsilon > 0$ and define $\gamma = (r - \epsilon) \exp i\theta$, where $0 \leq \theta < 2\pi$ and $i = \sqrt{-1}$. Then $\left\|A(z)^{-1}B(z)\right\| < 1$ for all $z \in \gamma$. Since this holds for all $\epsilon > 0$, by Theorem 2.6, $P(z)$ has no eigenvalues of moduli less than r.

Let $0 < k < d$, then $F(x)$ has either no real positive roots or two real positive roots (DRS). Suppose that $r_1 < r_2$ are two real positive roots of $F(x)$, then $F(x) < 0$ for all $x \in (r_1, r_2)$ and $\|B(z)\| < \left\|A^{-1}(z)\right\|^{-1}$ for all $r_1 < |z| < r_2$. Let $\epsilon > 0$, define $\gamma_1 = (r_1 + \epsilon) \exp i\theta$ and $\gamma_2 = (r_2 - \epsilon) \exp i\theta$, where $0 \leq \theta < 2\pi$ and $i = \sqrt{-1}$. Then $\left\|A(z)^{-1}B(z)\right\| < 1$ for all $z \in \gamma_1$ and $z \in \gamma_2$. So, Theorem 2.6 implies that $P(z)$ has kn eigenvalues in $B_{r_2}(0)$ and $\overline{B_{r_1}(0)}$. Since $\overline{B_{r_1}(0)} \subseteq B_{r_2}(0)$, it follows that $P(z)$ has no eigenvalues inside the annulus $\mathcal{A}_{r_1, r_2}(0)$.

Let $k = d$, then $F(x)$ has on real positive root R (DRS). Moreover, $F(x) < 0$ for all $x \in (R, \infty)$ and $\|B(z)\| < \left\|A(z)^{-1}\right\|^{-1}$ for all $|z| > R$. Let $\epsilon > 0$ and define $\gamma = (R + \epsilon) \exp i\theta$, where $0 \leq \theta < 2\pi$ and $i = \sqrt{-1}$. Then, $\left\|A(z)^{-1}B(z)\right\| < 1$ for all $z \in \gamma$ and for all $\epsilon > 0$. Therefore, by Theorem 2.6, $P(z)$ has nd eigenvalues inside $\overline{B_R(0)}$. Since $\det P(z)$ is a polynomial of degree nd, it follows that $P(z)$ has no eigenvalues of moduli greater than R. □

2.2.1 Spectral Bounds for Unitary Matrix Polynomials

It is well known that the eigenvalues of any unitary matrix lie on the unit circle. Moreover, since unitary matrices from a group under multiplication

$$G(n) = \left\{ A \in \mathbb{C}^{n\times n} : A^*A = I \right\}$$

it follows that the eigenvalues of all linear matrix polynomials $A + Bz$, where A, $B \in G(n)$ also lie on the unit circle. One example will suffice to show that this pattern does not continue for higher degree unitary matrix polynomials.

Example 2.1. *Let $P(z) = z^2I + zI - I$, where I is the identity matrix. Then the eigenvalues of $P(z)$ satisfy the equation $z^2 + z - 1 = 0$; therefore,*

$$\sigma(P) = \left\{ \frac{-1 \pm \sqrt{5}}{2} \right\}.$$

We say that an $(n \times n)$ matrix polynomial $P(z)$ of degree d is unitary, if the coefficient matrices satisfy $A_i \in G(n)$ for all $i = 0, 1, \ldots, d$. We denote the set of all $(n \times n)$ unitary matrix polynomials by

$$U(n) = \left\{ P(z) = \sum_{i=0}^{d} A_i z^i : A_i \in G(n),\, i = 0, 1, \ldots, d,\, d \in \mathbb{N} \right\},$$

and define the family of all unitary matrix polynomials by

$$\mathcal{U} = \bigcup_{n\in\mathbb{N}} U(n)$$

Corresponding to each $P(z) \in \mathcal{U}$ there is a spectrum $\sigma(P)$. Let $\sigma_{\mathcal{U}} = \{\sigma(P) : P(z) \in \mathcal{U}\}$ and $|\sigma_{\mathcal{U}}| = \{|\sigma_{\mathcal{U}}| : P(z) \in \mathcal{U}\}$, where $|\sigma(P)|$ denotes the set of moduli of the eigenvalues of $P(z)$.

Theorem 2.8 ([9, Theorem 3.2]). *For all $\lambda \in \sigma_{\mathcal{U}}$, it follows that*

$$\frac{1}{2} < |\lambda| < 2.$$

Proof. Define $u : \mathbb{R} \to \mathbb{R}$ by

$$u(x) = x^d - x^{d-1} - \cdots - 1.$$

Since the spectral norm of a unitary matrix is 1, it follows from Theorem 2.7 that the one real positive root of $u(x)$ is an upper bound on the moduli of the eigenvalues of any $P(z) \in \mathcal{U}$. Note that $2^d > 2^{d-1} + \cdots + 2^0$ for all positive integers d. Therefore, $u(2) > 0$ and $u(1) < 0$. Since u is continuous, the intermediate value theorem implies that there exists an $R \in (1, 2)$ such that $u(R) = 0$. Moreover, the moduli of any eigenvalue of $P(z)$ is bounded above by R and therefore bounded above by 2.

Define $l : \mathbb{R} \to \mathbb{R}$ by

$$l(x) = x^d + \cdots + x - 1.$$

By Theorem 2.7, the one real positive root of $l(x)$ is a lower bound on the moduli of the eigenvalues of any $P(z) \in \mathcal{U}$. Note that $\sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i = 1$; therefore, $l\left(\frac{1}{2}\right) < 0$ and $l(1) > 0$. By the intermediate value theorem there exists an $r \in (\frac{1}{2}, 1)$ such that $l(r) = 0$. The moduli of any eigenvalue of $P(z)$ is bounded below by r and therefore bounded below by $\frac{1}{2}$. □

Recall the matrix polynomial used in Example 2.1; we can use matrix polynomials of this form to prove that the bounds on the set $|\sigma_{\mathcal{U}}|$ given in Theorem 2.8 are optimal. This is, the upper bound is the least upper bound and the lower bound is the greatest lower bound.

Theorem 2.9 ([9, Theorem 3.3]). $\sup |\sigma_{\mathcal{U}}| = 2$ *and* $\inf |\sigma_{\mathcal{U}}| = \frac{1}{2}$.

Proof. Given Theorem 2.8 it suffices to prove that there is no number less than 2 or greater than $\frac{1}{2}$ which serves an as upper or lower bound to $|\sigma_{\mathcal{U}}|$, respectively. Suppose that $\frac{1}{2} < r < 1$ and note that $\sum_{i=1}^{\infty} r^i > 1$. Therefore, there exists a $d \in \mathbb{N}$ such that $\sum_{i=1}^{d} r^i > 1$. Define $P(z) = -I + \sum_{i=1}^{d} z^i I$, which is clearly an element of $\mathcal{U}$. Moreover, $P(z)$ has a real positive eigenvalue that is also the one real positive root of the polynomial $l(x) = x^d + \cdots + x - 1$. Since $l(r) > 0$ and $l(\frac{1}{2}) < 0$, there exists a $\lambda \in (\frac{1}{2}, r)$ such that $l(\lambda) = 0$. Therefore, r cannot be a lower bound of $|\sigma_{\mathcal{U}}|$. Now, define the matrix polynomial $P(z) = z^d I - \sum_{i=0}^{d-1} z^i I$, which is clearly an element of $\mathcal{U}$. Moreover, $P(z)$ has a real positive eigenvalue which is also the one real positive root of the polynomial $u(x) = x^d - \left(x^{d-1} + \cdots + 1\right)$. If $R \in (1, 2)$ is an upper bound of $|\sigma_{\mathcal{U}}|$, then $R^d \geq R^{d-1} + \cdots + 1$; otherwise, there would exist a $\lambda \in (R, 2)$ such that $u(\lambda) = 0$. Note that

$$R^{d-1} + \cdots + 1 = \frac{1 - R^d}{1 - R}.$$

Therefore, if $R \in (1, 2)$ is an upper bound of $|\sigma_{\mathcal{U}}|$, then $R^d \geq \frac{1-R^d}{1-R}$. Since $(1 - R) < 0$, it follows that $R^d(2 - R) \leq 1$ and

$$(2 - R) \leq \frac{1}{R^d}.$$

The above inequality must hold for all $d \in \mathbb{N}$ in order for R to be an upper bound of $|\sigma_{\mathcal{U}}|$. Since this is not possible, it follows that R cannot be an upper bound of $|\sigma_{\mathcal{U}}|$. □

2.2.2 Strategies for Computing Initial Estimates

We begin by noting that the bounds obtained in Theorem 2.7 can be sharpened if we replace (2.11) with

$$r^k = \sum_{i \neq k} \left\| A_k^{-1} A_i \right\| r^i. \tag{2.12}$$

To see that the bounds obtained will be sharper. Suppose that r is a real positive solution to (2.11), then

$$r^k = \sum_{i \neq k} \left\|A_k^{-1}\right\| \|A_i\| \, r^i \geq \sum_{i \neq k} \left\|A_k^{-1} A_i\right\| r^i.$$

Therefore, real positive solutions to (2.11) are always greater than or equal to the real positive solutions of (2.12).

Let $k_0, k_1, \ldots, k_q$ be values of k such that A_k is nonsingular and there exists real positive solution(s) $s_{k_i} \leq t_{k_i}$ to (2.12). Then $t_{k_i-1} \leq s_{k_i}$ for $i = 1, \ldots, q$ and there are $n(k_i - k_{i-1})$ eigenvalues of $P(z)$ in the annulus $\overline{\mathcal{A}(t_{k_i-1}, s_{k_i})}$. If for $k = 0$ or $k = d$ the matrix A_k is singular, then $t_0 = 0$ or $s_d = \infty$, respectively.

For the scalar case, $n = 1$, there is a cheap alternative to solving (2.12). Consider the scalar polynomial $w(x) = \sum_{i=0}^{d} a_i x^i$, where $a_0, a_d \neq 0$. The *Newton polygon* associated with this polynomial is the upper convex hull of the discrete set $\{(i, \ln |a_i|) : i = 0, 1, \ldots, d\}$. Let $k_0, \ldots, k_q$ denote the abscissas of the vertices of the Newton polygon, such that $0 = k_0 < k_1 < \cdots < k_q = d$, and define the radii

$$r_i = \left| \frac{a_{k_{i-1}}}{a_{k_i}} \right|^{\frac{1}{k_i - k_{i-1}}}, \quad i = 1, \ldots, q.$$

Then, $(k_i - k_{i-1})$ initial estimates to the roots of $w(x)$ are placed on circles centered at 0 with radius r_i for $i = 1, \ldots, q$. In [8, Theorem 1.2] it is shown that these estimates lie within the Pellet bounds for the scalar polynomial $w(x)$.

In [7, 8] this Newton Polygon technique is generalized for a specific class of matrix polynomials. More recently, this Newton polygon technique has been extended to general matrix polynomials [27]. In practice the idea is simple. Let $w(x) = \sum_{i=0}^{d} \|A_i\| \, x^i$. Then, $n(k_i - k_{i-1})$ initial estimates to the eigenvalues of $P(z)$ are placed on circles centered at zero with radius r_i for $i = 1, \ldots, q$.

Here the k_i and r_i are defined as above with reference to the Newton polygon associated with the polynomial $w(x)$.

The efficiency of the Newton polygon approach has been well established in the scalar case [5], and LMPEP will adopt this approach when $n = 1$. However, for $n > 1$ we have chosen to use another approach based on the numerical range.

Definition 2.10. The numerical range of the matrix polynomial $P(z)$ is defined as the set

$$W(P) = \{\lambda \in \mathbb{C} : x^* P(\lambda) x = 0, x \in \mathbb{C}^n, \|x\|_2 = 1\}.$$

To highlight the advantages of using the numerical range to obtain initial estimates to the eigenvalues of $P(z)$ we state and prove the following simple result.

Theorem 2.11. *If $P(z)$ is a hyperbolic quadratic matrix polynomial, then $W(P) \subseteq \mathbb{R}$.*

Proof. The quadratic matrix polynomial $P(z) = A_0 + A_1 z + A_2 z^2$ is said to be *hyperbolic*, if the following quadratic functions

$$a(x) = x^* A_2 x,\ b(x) = x^* A_1 x,\ c(x) = x^* A_0 x$$

satisfy $b(x)^2 > 4a(x)c(x)$ for all nonzero $x \in \mathbb{C}^n$. It follows that the roots of $x^* P(z) x$ are all real. □

Therefore, if $P(z)$ is hyperbolic, then initial estimates from the numerical range will lie on the real line, and initial estimates from the Newton polygon will be placed on circles in the complex plane. The implication is that elements from the numerical range, even random elements, adhere to the structure of the spectrum, in general, better than points on a circle in the complex plane.

This argument is difficult to quantify, since the numerical range of a matrix polynomial is not, in general, compact or convex. However, when the matrix polynomial is self adjoint there is a close relationship between the numerical range and the spectrum of the matrix polynomial [16, Ch. 10.6], [20]. For the moment we rely on numerical experiments in Section 2.3 to make our point, and consider further analysis of the use of the numerical range for initial estimates as a direction of future research.

2.2.3 Zero and Infinite Eigenvalues

LMPEP will begin with an a priori identification of zero and infinite eigenvalues, along with corresponding eigenvectors, backward error, and condition estimates. Then either the Newton polygon or numerical range technique is used to provide initial estimates for the remaining finite eigenvalues.

We assume that the zero and infinite eigenvalues are semi-simple, and return any linearly independent right and left eigenvectors corresponding to the zero eigenvalues of A_0 and A_d. If the zero or infinite eigenvalues are defective, then the missed eigenvalues can be found later using an appropriate root-finding algorithm, such as Laguerre's method.

If semi-simple, then the number of zero and infinite eigenvalues is equal to the dimension of $\mathrm{Nul}\,(A_0)$ and $\mathrm{Nul}\,(A_d)$, respectively. To estimate the dimension of the null space, we compute the QR factorization with column pivoting of both A_0 and A_d. Let $QR = A_i E$, for $i = 0$ or d, where

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix},$$

R_{11} is $k_1 \times k_1$, R_{22} is $k_2 \times k_2$, $k_1 + k_2 = n$, and E is a permutation matrix. Our aim is to determine an index k_1 such that R_{11} is well conditioned and R_{22} is negligible. If $k_1 < n$, then the matrix A_0

or A_d is rank deficient, and there are k_2 zero or infinite eigenvalues, respectively. Corresponding right and left eigenvectors can be computed as follows. For $i = k_1 + 1 : n$, let

$$x(k_1 + 1 : i - 1) = 0,\ x(i) = 1,\ x(i + 1 : n) = 0,$$

$$x(1 : k_1) = -R(1 : k_1, 1 : k_1)^{-1} R(1 : k_1, i).$$

LMPEP will return $\dfrac{Px}{\|Px\|_2}$ as an approximate right eigenvector. In addition, the last k_2 columns of the matrix Q are returned as approximate left eigenvectors. Backward error and condition estimates will be given as discussed in Section 2.1.3.

Once the zero and infinite eigenpairs have been computed, LMPEP will use the columns of the matrix Q, from the QR factorization of A_0 to compute elements of the numerical range. Specifically, the LAPACK subroutine DLARNV is used to obtain a random element of unit length from the column space of Q, which will be denoted x. Then elements of the numerical range are computed by finding the roots of the polynomial

$$x^* A_0 x + x^* A_1 x z + \cdots + x^* A_d x z^d.$$

The roots of the polynomial are computed using SLMPEP, see Section 2.3.3, and then sorted from smallest to largest moduli. Let *dze* denote the dimension of the zero eigenspace and *die* denote the dimension of the infinite eigenspace. Then the first *dze* approximations from the numerical range are replaced by 0 and the last *die* approximations are replaced by ∞. The algorithm for computing zero and infinite eigenvalues a priori and initial estimates for the remaining finite eigenvalues will be denoted by XGESTART, and is summarized in Appendix A.

2.3 Numerical Results

In this section, we provide numerical experiments to verify the claims made throughout Chapter 2. Specifically, we will justify our decision to use Laguerre's method over the Ehrlich-Aberth method, our decision to use elements from the numerical range over the Newton polygon approach for initial estimates, and we will provide a cost analysis of GELMPEP. Experiments that verify the accuracy of GELMPEP when applied to specific application problems can be found in Appendix B.

All code has been written in FORTRAN 95 and is available at `http://www.d.umn.edu/~camer133/code/code.xhtml`. The numerical experiments in this section have been run on a machine with CPU Intel Core i5 1.80GHZ and system Ubuntu 14.04 LTS.

2.3.1 Comparison of EAM vs. LM and NP vs. NR

In this section we provide numerical evidence to justify our choice of using Laguerre's method over the Ehrlich-Aberth method and the use of elements from the numerical range over the Newton polygon approach for initial estimates.

In the following experiments, the coefficients of a quadratic matrix polynomial are generated using the subroutine DLAGGE from LAPACK for sizes $n = 10, 20, 40, 80, 160$. The Ehrlich-Aberth method is used to compute eigenvalue approximations for the generated quadratic matrix polynomial. Over 5 trials, the average number of iterations (AVG IT), elapsed system time (ET), and maximum backward error (MAX BE) are recorded and compared for starting values from both the Newton polygon (NP) approach and the numerical range (NR). The average number of iterations are recorded as the total number of iterations divided by the total number of eigenvalues. The elapsed system time is recorded using the SYSTEM CLOCK subroutine, standard with FORTRAN

95 and latter. Eigenvectors are not computed in this trial, so an upper bound on the backward error is computed for each approximate eigenvalue using the results from Theorem 2.3. The results for the Ehrlich-Aberth method are displayed below.

EAM-NR: Quad. Mat. Poly.				EAM-NP: Quad. Mat. Poly.			
n	AVG IT	ET	MAX BE	n	AVG IT	ET	MAX BE
10	9.3	0.006	3.1E-016	10	7.5	0.005	9.2E-017
20	10.7	0.035	2.1E-016	20	9.5	0.030	6.9E-017
40	12.1	0.219	1.8E-016	40	12.3	0.248	7.8E-017
80	15.1	2.326	2.9E-015	80	17.0	2.390	9.5E-017
160	19.5	24.95	1.5E-015	160	27.2	36.17	1.1E-016

TABLE 2.1: Ehrlich-Aberth Method and Initial Estimate Strategies: Varying Size

Elements from the numerical range provided a slight advantage for the Ehrlich-Aberth method, for the large size problem. The maximum backward error was included in these trials to verify that the eigenvalues obtained are meaningful approximations. Note that the average number of iterations per eigenvalue increased dramatically as the size n increased. This phenomena has been observed in [7, 8]. The results for Laguerre's method are displayed in Table 2.2.

LM-NR: Quad. Mat. Poly.				LM-NP: Quad. Mat. Poly.			
n	AVG IT	ET	MAX BE	n	AVG IT	ET	MAX BE
10	3.1	0.003	1.9E-016	10	3.2	0.003	5.9E-017
20	3.2	0.013	9.9E-017	20	3.5	0.014	6.4E-017
40	3.2	0.072	3.8E-016	40	3.7	0.091	8.4E-017
80	3.3	0.570	1.6E-016	80	4.2	0.819	1.1E-016
160	3.4	5.382	4.1E-016	160	4.7	9.005	1.0E-016

TABLE 2.2: Laguerre's Method and Initial Estimate strategies: Varying Size

From Table 2.2, we can see Laguerre's method outperforms the Ehrlich-Aberth method, independent of which approximations are used. Most importantly, when using the numerical range to obtain initial estimates, the average number of iterations remains around 3 to 4 eigenvalues per iteration.

When the Ehrlich-Aberth method was proposed for solving the polynomial eigenvalue problem, it was not intended to be used for large size problems, rather large degree problems. Therefore, we repeat the above experiments on 10×10 matrix polynomials of degree 10, 20, 40, 80, 160.

EAM-NR: 10×10 Mat. Poly.				EAM-NP: 10×10 Mat. Poly.			
n	AVG IT	ET	MAX BE	n	AVG IT	ET	MAX BE
10	9.1	0.044	1.2E-016	10	8.6	0.040	6.3E-017
20	8.9	0.127	6.9E-016	20	8.6	0.128	6.5E-017
40	8.7	0.419	2.3E-016	40	8.9	0.416	7.5E-017
80	8.7	1.422	2.1E-016	80	9.1	1.539	7.6E-017
160	8.6	5.207	6.1E-016	160	9.7	5.974	1.1E-016

TABLE 2.3: Ehrlich-Aberth Method and Initial Estimate Strategies: Varying Degree

LM-NR: 10×10 Mat. Poly.				LM-NP: 10×10 Mat. Poly.			
n	AVG IT	ET	MAX BE	n	AVG IT	ET	MAX BE
10	3.1	0.021	7.7E-016	10	3.3	0.021	6.0E-017
20	3.1	0.059	1.1E-016	20	3.6	0.076	1.2E-016
40	3.1	0.198	3.1E-016	40	3.7	0.243	6.9E-017
80	3.2	0.740	1.1E-016	80	4.2	0.996	8.7E-016
160	3.1	2.666	1.4E-016	160	5.8	5.353	5.4E-015

TABLE 2.4: Laguerre's Method and Initial Estimate Strategies: Varying Degree

Again, Laguerre's method outperforms the Ehrlich-Aberth method, in the trials recorded in Table 2.3 and 2.4. Moreover, we see that the numerical range provides reliable initial estimates for both root-finding algorithms, and improves performance over the Newton polygon approach.

2.3.2 Cost Analysis of GELMPEP

In this section we will provide a cost analysis for GELMPEP: Laguerre's method applied to general matrix polynomials. Pseudo code for algorithms referenced in this section is available in Appendix A.

GELMPEP will begin by calling XGESTART, Algorithm 2, which can be run in $O(n^3 + nd^2)$ time. Once the starting values have been computed, GELMPEP will continue by updating each eigenvalue approximation using XGELCORR, Algorithm 3, in $O(n^3 + dn^2)$ time. Once the approximate eigenvalues have been computed, GELMPEP will compute corresponding left and right eigenvectors, backward error, and condition estimates. This is done by calling XGEEVC, Algorithm 4, in $O(dn^4 + d^2n^3)$ time.

We assume that the number of Laguerre iterations required to obtain convergence for all approximate eigenvalues is $O(nd)$. Therefore, GELMPEP can compute all approximate eigenvalues, eigenvectors, backward error, and condition estimates in $O(dn^4 + d^2n^3)$ time. Table 2.5 provides a set of experiments to verify our cost analysis. In these experiments both eigenvalues and eigenvectors are recorded; therefore, backward error can be computed using (2.8). The maximum right (MAX BEX) and left (MAX BEY) backward error is included to verify that the approximations obtained are meaningful.

GELMPEP: Quad. Mat. Poly.					
n	AVG IT	ET	ADJ RA	MAX BEX	MAX BEY
10	2.9	0.008		8.1E-017	1.5E-016
20	3.2	0.026	2.9	6.1E-017	1.7E-016
40	3.5	0.113	4.0	6.8E-017	2.2E-016
80	3.3	0.680	6.3	8.4E-017	2.3E-016
160	3.8	6.682	8.6	7.8E-017	2.3E-016
320	3.8	73.33	10.9	6.0E-017	2.3E-016

TABLE 2.5: GELMPEP Cost Analysis: Varying Size

We introduced the measurement adjusted ratio (ADJ RA) in the previous experiment. This measures the ratio of elapsed time when doubling the size (n) or degree (d), while accounting for any change in the average number of iterations (AVG IT). Specifically, it is the ratio of (ET/AVG IT) for successive trials. The above experiment shows that when doubling the size of the matrix polynomial the elapsed time for GELMPEP increases by a factor which we suspect will eventually

approach 16. We also, expect that when doubling the degree of the matrix polynomial the elapsed time for GELMPEP should increase by an asymptotic factor of 4, the experiments recorded in Table 2.6 confirm this.

GELMPEP: 10×10 Mat. Poly.					
d	AVG IT	ET	ADJ RA	MAX BEX	MAX BEY
10	3.1	0.094		6.0E-017	2.0E-017
20	2.9	0.315	3.6	8.5E-017	1.9E-016
40	3.3	0.784	2.1	8.5E-017	2.2E-016
80	3.2	0.801	1.0	1.0E-016	2.3E-016
160	3.3	3.097	3.8	1.0E-016	2.2E-016
320	3.1	11.10	3.7	1.1E-016	2.2E-016

TABLE 2.6: GELMPEP Cost Analysis: Varying Degree

2.3.3 Scalar Polynomials

In this section we will outline LMPEP for a scalar polynomial defined by

$$p(z) = \sum_{i=0}^{d} a_i z^i,$$

where a_i are real or complex numbers. The scalar problem is an important special case of the polynomial eigenvalue problem, and is used when computing elements of the numerical range.

In the scalar case, the initial estimates to the roots of $p(z)$ are computed via the Newton polygon approach as outlined in Section 2.2.2, and can be done in $O(d)$ time. Then each root approximation can be updated using the Laguerre correction in $O(d)$ time. The backward error of each root approximation can be computed using

$$\eta(\lambda) = \frac{|p(\lambda)|}{\alpha},$$

where $\alpha = \sum_{i=0}^{d} |\lambda|^i |a_i|$. Given an approximate root λ, if the inequality $\eta(\lambda) < \epsilon$ holds, where ϵ is unit roundoff that depends on the data type, then we stop updating λ. We add the stopping criterion

$$\left|\hat{\lambda} - \lambda\right| < \tau,$$

where $\hat{\lambda}$ is the Laguerre iterate and τ is a predetermined tolerance that depends on unit roundoff ϵ and the moduli of the approximate roots. Once either of the stopping criteria is met, or some maximum number of iterations has been reached, updates to the approximate eigenvalue will cease.

Assuming that the number of Laguerre iterations required to obtain convergence for all approximate roots is $O(d)$, then all root approximations can be computed in $O(d^2)$ time. The algorithm for computing the roots of a scalar polynomial will be denoted XSLM and is summarized in Algorithm 1. In Table 2.7 are a set of experiments to verify our cost analysis.

The coefficients of a scalar polynomial are generated using subroutine DLARNV from LAPACK for degree $d = 50$, 100, 200, 400, 800, 1600. SLMPEP is used to compute root approximations and backward error. The average number of iterations (AVG IT), elapsed system time (ET), adjusted ratio (ADJ RA), and maximum backward error (MAX BE) are recorded.

SLMPEP: Scal. Poly.				
d	AVG IT	ET	ADJ RA	MAX BE
50	2.7	0.001		1.8E-016
100	2.6	0.004	4.1	6.7E-016
200	2.5	0.014	3.7	7.3E-016
400	2.6	0.035	2.4	9.5E-016
800	2.6	0.074	2.1	3.2E-015
1600	2.6	0.304	4.1	6.7E-015

TABLE 2.7: SLMPEP Cost Analysis

Chapter 3

LMPEP for Hessenberg Matrix Polynomials

One way to make LMPEP faster is to take advantage of any structure present in the coefficient matrices of $P(z)$. In this section we consider the case where the coefficient matrices are in Hessenberg form. We are motivated to consider Hessenberg structure, because to our knowledge it has not been fully utilized in the context of solving matrix polynomial eigenvalue problems. While it is possible to reduce any matrix polynomial to Hessenberg form, see Section 5, no numerically stable algorithm to perform this reduction exists at this time. However, there are applications where this Hessenberg form arises naturally in the matrix polynomial; consider the bilby problem in [3]. In Section 3.1 we introduce a generalization of Hyman's method for matrix polynomials. In Section 3.2 we use Hyman's method to obtain an efficient computation of the Laguerre correction term. The reduction in the cost of computing the eigenvectors is also discussed. In Section 3.3 we provide numerical experiments which verify the cost analysis of our method applied to Hessenberg matrix polynomials.

3.1 Hyman's Method

Hyman's method, a method for evaluating the characteristic polynomial and its derivatives at a point $\lambda \in \mathbb{C}$, is attributed to a conference presentation given by M.A. Hyman of the Naval Ordnance Laboratory in 1957 [39]. The backward stability of Hyman's method has been shown in [39]. Hyman's method has been used with standard matrices [28] and matrix pencils [14]. Now, Hyman's method will be used for matrix polynomials and ultimately used to compute (2.1)-(2.3) for Hessenberg matrix polynomials.

It suffices to consider upper Hessenberg matrix polynomials, since $P(\lambda)x = 0$ if and only if $x^*P^*(\overline{\lambda}) = 0$. Therefore, the right and left eigenvectors of $P(z)$ corresponding to the eigenvalue λ are the conjugate transpose of the left and right eigenvectors of $P^*(z)$ corresponding to the eigenvalue $\overline{\lambda}$, respectively. In this section, we denote an $(n \times n)$ upper Hessenberg matrix polynomial of degree d by

$$P(z) = \begin{bmatrix} p_{11}(z) & p_{12}(z) & \cdots & p_{1n}(z) \\ p_{21}(z) & p_{22}(z) & & p_{2n}(z) \\ & \ddots & \ddots & \vdots \\ & & p_{n,n-1}(z) & p_{nn}(z) \end{bmatrix},$$

where $p_{ij}(z)$ is a scalar polynomial of degree at most d. The matrix polynomial $P(z)$ has the same determinant as

$$\begin{bmatrix} p_{11}(z) & p_{12}(z) & \cdots & b(z) \\ p_{21}(z) & p_{22}(z) & & 0 \\ & \ddots & \ddots & \vdots \\ & & p_{n,n-1}(z) & 0 \end{bmatrix},$$

provided that

$$P(z)\begin{bmatrix} x_1(z) \\ \vdots \\ x_{n-1}(z) \\ 1 \end{bmatrix} = \begin{bmatrix} b(z) \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{3.1}$$

Now, let $p(z) = \det P(z)$, then we have

$$p(z) = (-1)^{n-1} b(z) q(z),$$

where $q(z) = \prod_{j=1}^{n-1} p_{j+1,j}(z)$. Taking first and second derivatives of (3.1) we obtain

$$P(z)\begin{bmatrix} x_1^{'}(z) \\ \vdots \\ x_{n-1}^{'}(z) \\ 0 \end{bmatrix} = \begin{bmatrix} b^{'}(z) \\ 0 \\ \vdots \\ 0 \end{bmatrix} - P^{'}(z)\begin{bmatrix} x_1(z) \\ \vdots \\ x_{n-1}(z) \\ 1 \end{bmatrix}, \tag{3.2}$$

$$P(z)\begin{bmatrix} x_1^{''}(z) \\ \vdots \\ x_{n-1}^{''}(z) \\ 0 \end{bmatrix} = \begin{bmatrix} b^{''}(z) \\ 0 \\ \vdots \\ 0 \end{bmatrix} - 2P^{'}(z)\begin{bmatrix} x_1^{'}(z) \\ \vdots \\ x_{n-1}^{'}(z) \\ 0 \end{bmatrix} - P^{''}(z)\begin{bmatrix} x_1(z) \\ \vdots \\ x_{n-1}(z) \\ 1 \end{bmatrix}. \tag{3.3}$$

Taking the first and second derivatives of $p(z)$ we obtain

$$p^{'}(z) = (-1)^{n-1}\left(b^{'}(z)q(z) + b(z)q^{'}(z)\right),$$

$$p^{''}(z) = (-1)^{n-1}\left(b^{''}(z)q(z) + 2b^{'}(z)q^{'}(z) + b(z)q^{''}(z)\right).$$

Given $\lambda \in \mathbb{C}$, (3.1) can be solved in $O(n^2)$ time, by taking advantage of the upper Hessenberg structure in $P(\lambda)$. The solution from (3.1) is then used in the matrix equation (3.2), which can be solved in $O(n^2)$ time. Finally, the solutions from (3.1) and (3.2) are used in the matrix equation (3.3), which can be solved in $O(n^2)$ time. If $P(\lambda)$ is not properly upper Hessenberg, one of the subdiagonal entries is zero, then solving (3.1)-(3.3) will require division by zero. This problem can be remedied by replacing any zero elements along the subdiagonal with ϵ (machine epsilon) [39, ch. 3].

3.2 Eigenvalue and Eigenvector computation

Using Hyman's method and (3.1)-(3.3), an efficient computation of (2.1)-(2.3) can be obtained by noting the following

$$\frac{p^{'}(\lambda)}{p(\lambda)} = \frac{b^{'}(\lambda) + b(\lambda)\left(\frac{q^{'}(\lambda)}{q(\lambda)}\right)}{b(\lambda)}, \tag{3.4}$$

$$-\left(\frac{p^{'}(\lambda)}{p(\lambda)}\right)^{'} = \left(\frac{p^{'}(\lambda)}{p(\lambda)}\right)^{2} - \left(\frac{b^{''}(\lambda) + 2b^{'}(\lambda)\left(\frac{q^{'}(\lambda)}{q(\lambda)}\right) + b(\lambda)\left(\frac{q^{''}(\lambda)}{q(\lambda)}\right)}{b(\lambda)}\right). \tag{3.5}$$

Equation (3.4) requires the additional computation of

$$\frac{q^{'}(\lambda)}{q(\lambda)} = \sum_{j=1}^{n-1} \frac{p^{'}_{j+1,j}(\lambda)}{p_{j+1,j}(\lambda)}.$$

Moreover, (3.5) requires the additional computation of

$$\frac{q^{''}(\lambda)}{q(\lambda)} = \left(\frac{q^{'}(\lambda)}{q(\lambda)}\right)^{'} + \left(\frac{q^{'}(\lambda)}{q(\lambda)}\right)^{2},$$

where $\left(\frac{q'(\lambda)}{q(\lambda)}\right)' = \sum_{j=1}^{n-1}\left(\frac{p''_{j+1,j}(\lambda)}{p_{j+1,j}(\lambda)} - \left(\frac{p'_{j+1,j}(\lambda)}{p_{j+1,j}(\lambda)}\right)^2\right)$. In summary, (3.4)-(3.5) provide an efficient method for computing the Laguerre correction term. These computations can be done in $O(n^2)$ time and are backward stable.

Stopping criteria is still computed using (2.4)-(2.5) and Theorem 2.3. The LU factorization of an upper Hessenberg matrix can be done in $O(n^2)$ time, taking advantage of the fact that each column has at most one row operation to be performed. The algorithm for computing the Laguerre correction term, checking stopping criteria, and updating each eigenvalue approximation will be denoted XHSLCORR, and is summarized in Appendix A.

Once an approximate eigenvalue is found, corresponding right and left eigenvectors are computed using the QR factorization of $P(\lambda)$ as discussed in Section 2.1.2. Moreover, the QR factorization of an upper Hessenberg matrix can be done in $O(n^2)$ time using plane rotations. The algorithm for computing right and left eigenvectors, backward error, and condition estimates will be denoted XHSEVC.

3.3 Numerical Results

In this section, we provide numerical experiments to verify the claims made throughout Chapter 3. Specifically, we provide a cost analysis of HSLMPEP and provide numerical experiments to justify our claims. Experiments that verify the accuracy of HSLMPEP when applied to specific application problems can be found in Appendix B.

All code has been written in FORTRAN 95 and is available at `http://www.d.umn.edu/~camer133/code/code.xhtml`. The numerical experiments in the section have been run on a machine with CPU Intel Core i5 1.80GHZ and system Ubuntu 14.04 LTS.

3.3.1 Cost Analysis of HSLMPEP

In this section we will provide a cost analysis for HSLMPEP: Laguerre's method applied to Hessenberg matrix polynomials. Pseudo code for algorithms referenced in this section is available in Appendix A.

The initial estimates for HSLMPEP are computed as they are in GELMPEP, see Section 2.2.3 and 2.3.2. Therefore, the starting values for HSLMPEP cost approximately $O(n^3 + nd^2)$. Once the starting values have been computed, HSLMPEP will continue by updating each eigenvalue approximation using XHSLCORR, Algorithm 5. Once the approximate eigenvalues have been computed, HSLMPEP will compute corresponding left and right eigenvectors, backward error, and condition estimates. This is done by calling XHSEVC which can be done in approximately $O(dn^3 + d^2n^3)$ time. We have not included pseudo code for XHSEVC since it is identical to XGEEVC, with the exception that the QR factorization can be done in $O(n^2)$ time.

Again, we assume that the number of Laguerre iterations required to obtain convergence for all approximate eigenvalues is $O(nd)$. Therefore, all eigenvalues of a Hessenberg matrix polynomial can be computed in approximately $O(dn^3+d^2n^3)$ time. In Table 3.1 and 3.2 are a set of experiments to verify our cost analysis.

The coefficients of a quadratic matrix polynomial are generated using the subroutine DLAGGE from LAPACK for sizes n = 10, 20, 40, 80, 160, 320. HSLMPEP is used to compute eigenvalue approximations, corresponding left and right eigenvectors, backward error, and condition estimates. The average number of iterations (AVG IT), elapsed system time (ET), maximum right and left backward error (MAX BEX and MAX BEY), and the adjusted ratio (ADJ RA) are recorded.

HSLMPEP: Quad. Mat. Poly.					
n	AVG IT	ET	ADJ RA	MAX BEX	MAX BEY
10	3.4	0.005		6.8E-017	2.2E-016
20	3.2	0.014	2.9	7.3E-017	1.6E-016
40	3.9	0.075	4.4	5.3E-017	1.8E-016
80	4.8	0.419	4.5	4.1E-017	2.1E-016
160	5.2	3.190	7.0	3.7E-017	2.1E-016
320	6.7	32.96	7.9	3.7E-017	2.1E-016

TABLE 3.1: HSLMPEP Cost Analysis: Varying Size

HSLMPEP: 10×10 Mat. Poly.					
d	AVG IT	ET	ADJ RA	MAX BEX	MAX BEY
10	3.0	0.023		9.0E-017	2.1E-016
20	3.1	0.064	2.7	5.5E-017	1.8E-016
40	2.8	0.175	3.0	4.9E-017	2.1E-016
80	2.9	0.457	2.4	8.6E-017	2.1E-016
160	2.8	1.734	3.9	9.2E-017	2.2E-016
320	2.8	6.130	3.5	1.4E-016	2.2E-016

TABLE 3.2: HSLMPEP Cost Analysis: Varying Degree

Table 3.1 verifies that when doubling the size of the matrix polynomial the elapsed time for HSLMPEP increases by a factor which approaches 8. We also expect that when doubling the degree of the matrix polynomial the elapsed time for HSLMPEP should increase by an asymptotic factor of 4, the results in Table 3.2 confirm this.

Chapter 4

LMPEP for Tridiagonal Matrix Polynomials

Additional computational and storage savings can be obtained if the structure of the tridiagonal matrix polynomial is utilized. Efficient computation for the eigenvalues of a tridiagonal matrix polynomial has been considered both for linear [6] and quadratic [30] matrix polynomials. We are motivated to consider tridiagonal structure, since there are applications where this form arises naturally in matrix polynomials; consider the spring problem in [3]. In addition, previous developments have focused on the computation of only the eigenvalues of specific degree matrix polynomials; LMPEP can be efficiently applied to any degree tridiagonal matrix polynomial, and will include eigenvector computations as well as backward error and condition estimates. In Section 4.1 we revisit Hyman's method for tridiagonal matrix polynomials. In Section 4.2 we outline the methods involved for an efficient computation of the eigenvectors. In Section 4.3 we discuss the a priori identification of zero and infinite eigenvalues. In Section 4.4 We present numerical results which verify the cost analysis of our method applied to tridiagonal matrix polynomials.

4.1 Hyman's Method Revisited

In this section we revisit Hyman's method, first introduced in Section 3.1, with our focus on tridiagonal matrix polynomials. Let the $(n \times n)$ tridiagonal matrix polynomial of degree d be denoted by

$$P(z) = \begin{bmatrix} p_{11}(z) & p_{12}(z) & & & \\ p_{21}(z) & p_{22}(z) & \ddots & & \\ & \ddots & \ddots & p_{n-1,n}(z) & \\ & & p_{n,n-1}(z) & p_{nn}(z) & \end{bmatrix},$$

where $p_{ij}(z)$ is a scalar polynomial of degree at most d. When storing the matrix polynomial it suffices to store the coefficients in three arrays p_{dl}, p_d, p_{du} denoting the lower diagonal, diagonal, and upper diagonal coefficients of $P(z)$. Therefore, the tridiagonal matrix polynomial requires $O(nd)$ storage. This is synonymous with how LAPACK stores general tridiagonal matrices in routines such as DGTSV. Moreover, the tridiagonal structure enables us to solve (3.1-3.3) in $O(n)$ time. Hence, there are significant storage and computational savings for Hyman's method when applied to the tridiagonal matrix polynomial.

4.2 Computing Eigenvectors

Once an approximate eigenvalue λ has been computed, the QR factorization of $P(\lambda)$ is used to find corresponding left and right eigenvector approximations. The tridiagonal structure allows the QR factorization in $O(n)$ time, yet to our knowledge no LAPACK routine is available for this task, so we wrote our own, see XGTQR in Appendix A. The algorithm XGTQR performs the QR factorization using plane rotations on a tridiagonal matrix represented by adl, ad, adu. The plane

rotations which form the matrix Q are returned in arrays c and s, and the upper triangular matrix R is returned as $rd = ad$, $rdu = adu$, $rdu2 = adl$.

With the QR factorization in hand, an efficient way to compute (2.6-2.7) is needed. To this end, XGTRS solves the system $Rx = b$ (where R is stored by XGTQR), and XGTQM performs the multiplication $Q^{\text{trans}}x$ (where Q is stored by XGTQR and tans=N, T, or H). Therefore, with Q and R stored by XGTQR, we can compute (2.6)-(2.7) in $O(n)$ time.

4.3 Zero and Infinite Eigenvalues

In section 2.2.3 we assumed that the zero and infinite eigenvalues were semi-simple. Then the QR factorization with column pivoting is used to obtain an approximate basis for the null space of A_0 and A_d, respectively. If the zero or infinite eigenvalues were defective, we picked up any missed eigenvalues using the root-finding algorithm powered by Laguerre's method.

In order to take full advantage of the tridiagonal structure, we settle for a QR factorization, with no column pivoting, of the coefficient matrices A_0 and A_d. The dimension of the nullspace for either A_0 or A_d is then approximated using the upper triangular matrix R. If the dimension is nonzero then left and right eigenvectors are approximated using both matrices Q and R.

Example 4.1. *Let*

$$P(z) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} + z \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} + z^2 I,$$

where I is the 3×3 identity matrix. There are no infinite eigenvalues and the geometric multiplicity of the zero eigenvalue is 1. The corresponding right and left eigenvectors are

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \; y = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Example 4.1 provides insight into the complications that can arise when using the QR factorization without column pivoting. With column pivoting, the matrix R from the QR factorization of A_0, would have only one main diagonal entry equal to zero. Without column pivoting the matrix R has three main diagonal entries equal to zero. Therefore, it is important to check whether a zero main diagonal entry actually corresponds to a free variable in the matrix equation $Ax = 0$.

Once the zero and infinite eigenpairs have been computed, elements from the numerical range are computed to provide initial estimates for the remaining finite eigenvalues. The columns of the matrix Q, from the QR factorization of A_0, are used to compute elements from the numerical range. The LAPACK subroutine DLARNV is used to obtain a random element of the column space of Q which will be denoted x. Then the products

$$x^* A_i x, \;\; i = 0, 1, \ldots, d$$

can be computed in $O(n(d+1))$ time. Finally, SLMPEP is used to compute the roots of the polynomial with these coefficients. The algorithm for computing zero and infinite eigenvalues a priori and initial estimates for the remaining finite eigenvalues will be denoted by XGTSTART, and is summarized in Algorithm 9.

4.4 Numerical Results

In this section, we provide numerical experiments to verify the claims made throughout Chapter 4. Specifically, we provide a cost analysis of GTLMPEP and provide numerical experiments to justify our claims. Experiments that verify the accuracy of GTLMPEP when applied to specific application problems can be found in Appendix B.

All code has been written in FORTRAN 95 and is available at `http://www.d.umn.edu/~camer133/code/code.xhtml`. The numerical experiments in the section have been run on a machine with CPU Intel Core i5 1.80GHZ and system Ubuntu 14.04 LTS.

4.4.1 Cost Analysis of GTLMPEP

In this section we will provide a cost analysis for GTLMPEP: Laguerre's method applied to tridiagonal matrix polynomials. Pseudo code for algorithms referenced in this section is available in Appendix A.

The initial estimates for GTLMPEP are computed using XGTSTART, Algorithm 9, in $O(n^2d+nd^2)$ time. Once the starting values have been computed, GTLMPEP will continue by updating each eigenvalue approximation using XGTLCORR. We have not included pseudo code for XGTLCORR since it is identical to XHSLCORR, with the exception that Hyman's method can be done in $O(n)$ time. Once the approximate eigenvalues have been computed, GTLMPEP will compute corresponding left and right eigenvectors, backward error, and condition estimates. This is done by calling XGTEVC, Algorithm 10, in $O(dn^2)$ time.

Assuming that the number of Laguerre iterations required to obtain convergence for all approximate eigenvalues is $O(nd)$, GTLMPEP can compute all eigenvalue approximations in $O(n^2d+nd^2)$ time. Below we provide a set of experiments to verify our cost analysis.

The coefficients of a quadratic matrix polynomial are generated using the subroutine DLAGGE from LAPACK for sizes n = 10, 20, 40, 80, 160, 320. HSLMPEP is used to compute eigenvalue approximations, corresponding left and right eigenvectors, backward error, and condition estimates. The average number of iterations (AVG IT), elapsed system time (ET), maximum right and left backward error (MAX BEX and MAX BEY), and the adjusted ratio (ADJ RA) are recorded.

GTLMPEP: Quad. Mat. Poly.					
n	AVG IT	ET	ADJ RA	MAX BEX	MAX BEY
10	2.9	0.003		4.8E-017	1.1E-016
20	3.9	0.005	1.2	3.9E-017	1.7E-016
40	3.4	0.011	2.4	7.9E-017	2.2E-016
80	4.2	0.041	3.0	8.6E-017	1.8E-016
160	5.3	0.164	3.1	9.8E-017	1.7E-016
320	6.6	0.566	2.7	1.2E-016	2.1E-016
640	8.2	2.552	3.6	1.4E-016	1.9E-016

TABLE 4.1: GTLMPEP Cost Analysis: Arraying Size

Table 4.1 shows that when doubling the size of the matrix polynomial the elapsed time for GTLMPEP increases by an asymptotic factor of 4. We also expect that when doubling the degree of the matrix polynomial the elapsed time for GTLMPEP should increase by an asymptotic factor of 4, the results in Table 4.2 confirm this.

GTLMPEP: 10×10 Mat. Poly.					
d	AVG IT	ET	ADJ RA	MAX BEX	MAX BEY
10	3.1	0.010		5.8E-017	2.0E-016
20	3.2	0.027	2.6	1.0E-016	2.0E-016
40	3.3	0.074	2.6	6.5E-017	2.2E-016
80	3.0	0.187	2.7	1.1E-016	2.2E-016
160	3.3	0.584	2.8	1.5E-016	2.2E-016
320	3.1	2.060	3.6	2.7E-016	2.7E-016
640	3.1	8.004	3.8	2.6E-016	2.7E-016

TABLE 4.2: GTLMPEP Cost Analysis: Varying Degree

Chapter 5

On The Reduction of Matrix Polynomials to Hessenberg Form

Recently there has been a push to develop methods for finding the eigenvalues and eigenvectors of a matrix polynomial that are similar to the methods that have been used on real and complex matrices. For example, in [18], the authors develop Arnoldi type methods that operate on the coefficient matrices, and effectively project a large problem onto a smaller problem. In [32, 33] it is shown that any matrix polynomial can be reduced to triangular or quasi-triangular form, while preserving the degree and the finite and infinite elementary divisors of the matrix polynomial; the constructive proof requires information on the finite and infinite elementary divisors of the original matrix polynomial. Starting with the elementary divisors of a matrix polynomial is not practical if our end goal is to solve for the eigenvalues of the matrix polynomial. However, in [33], the authors also discuss using structure preserving similarity transformations acting on the linearization of the matrix polynomial, in order to obtain a triangular form of the matrix polynomial. This is truly

exciting research, a reliable process for reducing a matrix polynomial to triangular form while preserving the eigenvalues would have obvious benefits.

In this section we will show that every matrix polynomial can be reduced to an upper Hessenberg matrix whose entries are rational functions and in special cases polynomials. Recall that every matrix polynomial $P(z)$ admits the representation

$$P(z) = E(z)D(z)F(z),$$

where $D(z)$ is a diagonal matrix polynomial whose diagonal entries are the invariant polynomials of $P(z)$; $E(z)$ and $F(z)$ are matrix polynomials with constant nonzero determinants. This is known as the Smith form of the matrix polynomial $P(z)$, see Section 1.2.1. This Smith form is attained through elementary row and column operations which are stored in the matrix polynomials $E(z)$ and $F(z)$. With this result in mind, it is not surprising that we can reduce a matrix polynomial to upper Hessenberg form.

Our approach to obtaining an upper Hessenberg form of a matrix polynomial is unique, in that we choose to think of a matrix polynomial as a linear operator from $\mathbb{F}^n$ to $\mathbb{F}^n$, where $\mathbb{F}$ is the field of rational functions. In Section 5.1 we define a pseudo inner product on $\mathbb{F}^n \times \mathbb{F}^n$, and R-similarity of matrices in $\mathbb{F}^{n\times n}$. Armed with this pseudo inner product, we are able to apply Arnoldi and Krylov subspace methods to matrix polynomials. For a concise review on these methods see [36]. In Section 3, we use these familiar techniques to provide an elegant and constructive proof of the result that every square matrix polynomial is similar to an upper Hessenberg matrix A in $\mathbb{F}^{n\times n}$. We note that these results can be generalized for non-square matrix polynomials and matrix rationals. The practical importance of this reduction is that using Hyman's method [14, 39] and the Ehrlich-Aberth method [7, 8, 30], we can develop a numerical method for computing the eigenvalues of

A, as was done in Chapters 2 and 3. The theoretical importance is that in proving this result we will show that many familiar techniques from linear algebra can be applied directly to matrix polynomials, without using a linearization, as long as we work over the field of rational functions. In Section 5.3 we show that, in general, R-similarity transformations in $\mathbb{F}^{n\times n}$ do not preserve the Smith form of the original matrix polynomial. After identifying the problem, we provide sufficient conditions for which the Smith form is preserved under such transformations.

5.1 The Field of Rational Functions

A matrix polynomial of size $(n \times n)$ and degree d is defined by

$$P(z) = \sum_{k=0}^{d} A_k z^k, \tag{5.1.1}$$

where $A_k \in \mathbb{C}^{n\times n}$ and $A_d \neq 0$. Let $\mathbb{F}$ denote the field of rational functions over $\mathbb{C}$ and $\mathbb{F}^n$ denote the vector space of all n-tuples whose entries are rational functions with complex coefficients. It will be useful to denote the matrix polynomial as $P = [p_{ij}]_{i,j=1}^{n}$, where p_{ij} is a scalar polynomial in z, whose k^{th} coefficient is given by the ij^{th} entry of the k^{th} coefficient matrix in (5.1.1). A matrix polynomial can be viewed as a linear transformation $P : \mathbb{F}^n \to \mathbb{F}^n$. In this sense, the matrix polynomial in (5.1.1) has been written in the standard basis. The definitions of linear independence and spanning set carry over naturally from $\mathbb{C}^n$ to $\mathbb{F}^n$. We are interested in representing a matrix polynomial P with respect to other bases for $\mathbb{F}^n$, with this in mind we note the following definition.

Definition 5.1. We say that two matrices A, $B \in \mathbb{F}^{n\times n}$ are *R-similar*, if there exists a matrix $S \in \mathbb{F}^{n\times n}$, such that $\det S$ is a nonzero rational function and $AS = SB$.

Let $S \in \mathbb{F}^{n\times n}$ and consider the elementary transformations of interchanging two rows, and adding to some row another row multiplied by a rational function. With this in mind, it is not hard to show that the matrix S has linearly independent columns if and only if $\det S$ is a nonzero rational function.

In Section 5.2 we will use the Arnoldi method to show that every matrix polynomial is similar to an upper Hessenberg matrix A in $\mathbb{F}^{n\times n}$.

Definition 5.2. Define the *pseudo inner product* $\langle \cdot, \cdot \rangle : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$ by

$$\langle u, v \rangle = u_1 \cdot \overline{v_1} + \cdots + u_n \cdot \overline{v_n},$$

where $(\cdot)$ and $(+)$ are multiplication and addition of rational functions. The complex conjugate is taken over the coefficients of v_i, for $i = 1, \ldots, n$.

The function in Definition 5.2 is not a traditional inner product, since the output is a rational function. However, the following theorem will show that the pseudo inner product satisfies linearity in the first argument, and $\langle u, u \rangle = 0$ if and only if $u = 0$.

Theorem 5.3 ([10, Theorem 2.3])**.** *Let $u, v, z \in \mathbb{F}^n$ and $c \in \mathbb{F}$. Then*

$$\langle cu, v \rangle = c \langle u, v \rangle,$$

$$\langle u + v, z \rangle = \langle u, z \rangle + \langle v, z \rangle,$$

$$\langle u, u \rangle = 0$$

if and only if $u = 0$.

Proof. Since $\mathbb{F}^n$ is a vector space over the field of rational functions $\mathbb{F}$ we have

$$\langle cu,\, v\rangle = cu_1\bar{v}_1 + \cdots + cu_n\bar{v}_n = c\,\langle u,\, v\rangle\,,$$

and

$$\langle u+v,\, z\rangle = (u_1+v_1)\cdot\bar{z}_1 + \cdots + (u_n+v_n)\cdot\bar{z}_n$$

$$= (u_1\cdot\bar{z}_1 + \cdots + u_n\cdot\bar{z}_n) + (v_1\cdot\bar{z}_1 + \cdots + v_n\cdot\bar{z}_n) = \langle u,\, z\rangle + \langle v,\, z\rangle\,.$$

If $u = 0$, then it is clear that $\langle u,\, u\rangle = 0$. To prove the converse, first note that each element of u can be expressed as $u_i = \frac{p_i}{q_i}$, where p_i is a polynomial of degree d_i. Let $d = \max\limits_{1\le i\le n} d_i$, then write $p_i = a_{i,d}z^d + \cdots + a_{i,1}z + a_{i,0}$. Therefore, u can be written as

$$u = \begin{bmatrix} \frac{a_{1,d}z^d}{q_1} \\ \vdots \\ \frac{a_{n,d}z^d}{q_n} \end{bmatrix} + \cdots + \begin{bmatrix} \frac{a_{1,0}}{q_1} \\ \vdots \\ \frac{a_{n,0}}{q_n} \end{bmatrix} := v_d + \cdots + v_0.$$

By linearity of the inner product $\langle u,\, u\rangle = \langle v_0,\, v_0\rangle + \langle v_0,\, v_1\rangle + \cdots + \langle v_d,\, v_d\rangle$, where

$$\langle v_i,\, v_i\rangle = z^{2i}\left(\frac{|a_{1,i}|^2}{|q_1|^2} + \cdots + \frac{|a_{n,i}|^2}{|q_n|^2}\right),$$

for $i = 0,\, 1,\, \ldots,\, d$.

If $\langle u,\, u\rangle = 0$, then the highest degree term $\langle v_d,\, v_d\rangle = 0$. But this implies that $\max\limits_{1\le i\le n} d_i < d$, and we can apply the same argument to v_i for $i = d-1,\, \ldots,\, 1,\, 0$. Therefore, $\langle v_i,\, v_i\rangle = 0$ for $i = 0,\, 1,\, \ldots,\, d$ and it follows that $u = 0$. □

In addition to the properties in Theorem 5.3, the pseudo inner product satisfies conjugate symmetry, if the complex conjugate is taken over the coefficients of the rational function. We proceed to define orthogonality in $\mathbb{F}^n$ with respect to our pseudo inner product.

Definition 5.4. Two vectors $u, v \in \mathbb{F}^n$ are orthogonal, if $\langle u, v \rangle = 0$.

The properties that we have proven our inner product possesses are all we need to obtain upper Hessenberg form. This is because the Arnoldi process, which we will employ, is truly the Gram-Schmidt process in disguise. Consider the following example.

Example 5.1. *Let* $v_1(z) = \begin{bmatrix} 1 \\ \\ i \end{bmatrix}$ *and* $v_2(z) = \begin{bmatrix} iz \\ \\ 1 \end{bmatrix}$. *With the inner product in definition 5.2, we can use the Gram-Schmidt process to find an orthogonal basis for* $\mathbb{F}^2$. *Let* $u_1(z) = v_1(z)$ *and*

$$u_2(z) = v_2(z) - \frac{\langle v_2, u_1 \rangle}{\langle u_1, u_1 \rangle} u_1(z) = \begin{bmatrix} iz \\ \\ 1 \end{bmatrix} - \frac{i(z-1)}{2} \begin{bmatrix} 1 \\ \\ i \end{bmatrix} = \begin{bmatrix} \frac{i}{2}(z+1) \\ \\ \frac{1}{2}(z+1) \end{bmatrix}.$$

One can easily verify that $\langle u_1, u_2 \rangle = 0$.

In general, we may have a set of linearly independent vectors $\{v_1, \ldots, v_n\}$ in $\mathbb{F}^n$. Let $u_1 = v_1$ and proceed to define

$$u_{k+1} = v_{k+1} - \sum_{i=1}^{k} \frac{\langle v_{k+1}, u_i \rangle}{\langle u_i, u_i \rangle} u_i,$$

for $k = 1, \ldots, n-1$. Suppose we have created a set of orthogonal vectors $\{u_1, \ldots, u_k\}$ in $\mathbb{F}^n$. By linearity of our inner product

$$\langle u_{k+1}, u_j \rangle = \langle v_{k+1}, u_j \rangle - \sum_{i=1}^{k} \frac{\langle v_{k+1}, u_i \rangle}{\langle u_i, u_i \rangle} \langle u_i, u_j \rangle,$$

for $j = 1, \ldots, k$. Since $\langle u_i, u_j \rangle = 0$ for all $i \neq j$, it follows that $\langle u_{k+1}, u_j \rangle = 0$ for $j = 1, \ldots, k$. Therefore, with the properties proved in Theorem 5.3, we can be sure that the Gram-Schmidt process under our inner product will produce orthogonal vectors in $\mathbb{F}^n$.

5.2 Arnoldi and Krylov Subspace Methods

In Section 5.1 we saw that we could build an orthogonal basis for $\mathbb{F}^n$ by using n linearly independent vectors in $\mathbb{F}^n$ and the Gram-Schmidt process. In this section we will give a practical method for finding an orthogonal basis for a Krylov subspace of $\mathbb{F}^n$.

5.2.1 The Arnoldi Method

Let P be an $(n \times n)$ matrix polynomial, and let $v \in \mathbb{F}^n$ be a nonzero vector. Then the sequence of vectors

$$v, Pv, P^2v, \ldots, \tag{5.2.1}$$

is known as a *Krylov sequence* generated by the vector v and matrix polynomial P. Since the vector space $\mathbb{F}^n$ is an n-dimensional space over the field $\mathbb{F}$, it follows that the above Krylov sequence can produce at most n linearly independent vectors. Suppose that the vectors

$$v, Pv, \ldots, P^{k-1}v, \quad k \leq n$$

are linearly independent. Then these vectors form a basis for the k-dimensional Krylov subspace

$$K_k(P, v) = \text{span}\left\{v, Pv, \ldots, P^{k-1}v\right\}.$$

We are after an orthogonal basis for the Krylov subspace $K_k(P, v)$. To this end, consider the Arnoldi method which starts with the nonzero vector $v_1 = v$ and then produces

$$\hat{v}_{j+1} = Pv_j - \sum_{i=1}^{j} \frac{\langle Pv_j, v_i \rangle}{\langle v_i, v_i \rangle} v_i, \qquad v_{j+1} = \frac{1}{\beta_{j+1}} \hat{v}_{j+1} \tag{5.2.2}$$

for $j = 1, \ldots, k-1$. We will refer to $\beta_{j+1} \in \mathbb{F}$ as a *scaling factor*. We have some freedom in how we choose our scaling factor, but for now we let $\beta_{j+1} = 1$. We will consider other scaling factors and how they affect our transformation in Section 5.3. Note that the vectors that (5.2.2) produces are proportional, with respect to our scaling factor, to the vectors we would obtain from applying the Gram-Schmidt process to the vectors $v, Pv, \ldots, P^{k-1}v$. Therefore,

$$\operatorname{span}\left\{v, Pv, \ldots, P^j v\right\} = \operatorname{span}\left\{v_1, v_2, \ldots, v_{j+1}\right\},$$

for $j = 1, \ldots, k-1$.

The Arnoldi process combined with our inner product has provided us a way of computing an orthogonal basis for the Krylov subspace $K_k(P, v)$. Let k be the largest number of linearly independent vectors that the Krylov sequence in (5.2.1) can produce. If $k = n$, then the Arnoldi process will produce an orthogonal basis for $\mathbb{F}^n$, and the matrix representation of P with respect to this basis will be upper Hessenberg.

Example 5.2. *Let* $P(z) = \begin{bmatrix} z^2 & 1 & z \\ 1 & z^2 & 1 \\ z & 1 & z^2 \end{bmatrix}$ *and* $v_1(z) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$. *Applying the Arnoldi process, with scalar factor* 1, *we obtain*

$$v_2(z) = P(z)v_1(z) - z^2 v_1(z) = \begin{bmatrix} 0 \\ 1 \\ z \end{bmatrix},$$

$$v_3(z) = P(z)v_2(z) - \frac{z\left(z^3+z+2\right)}{z^2+1}v_2(z) - \left(z^2+1\right)v_1(z) = \begin{bmatrix} 0 \\ \frac{z\left(z^2-1\right)}{z^2+1} \\ \frac{1-z^2}{z^2+1} \end{bmatrix},$$

$$v_4(z) = P(z)v_3(z) - \frac{z\left(z^3+z-2\right)}{z^2+1}v_3(z) - \frac{\left(z^2-1\right)^2}{\left(z^2+1\right)^2}v_2(z) - 0v_1(z) = 0.$$

Therefore, we have

$$V(z) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{z\left(z^2-1\right)}{z^2+1} \\ 0 & z & \frac{1-z^2}{z^2+1} \end{bmatrix},\ A(z) = \begin{bmatrix} z^2 & z^2+1 & 0 \\ 1 & \frac{z\left(z^3+z+2\right)}{z^2+1} & \frac{\left(z^2-1\right)^2}{\left(z^2+1\right)^2} \\ 0 & 1 & \frac{z\left(z^3+z-2\right)}{z^2+1} \end{bmatrix},$$

such that $P(z)V(z) = V(z)A(z)$.

The matrix $V(z)$ in Example 5.2 has orthogonal columns, and it follows from Definition 5.1 that $P(z)$ is R-similar to $A(z)$. Therefore, the fact that $\det P(z) = \det A(z)$ is not surprising. We generalize this result in the following theorem.

Theorem 5.5 ([10, Theorem 3.2])**.** *Suppose that $P(z)$, $A(z) \in \mathbb{F}^{n\times n}$ are R-similar. Then*

$$\det\left(z^{\alpha}P(z^{\beta})\right) = \det\left(z^{\alpha}A(z^{\beta})\right)$$

for all α, $\beta \in \mathbb{Z}$.

Proof. If $P(z)$ is R-similar to $A(z)$, then there exists a $V(z) \in \mathbb{F}^{n\times n}$ with linearly independent columns such that $P(z)V(z) = V(z)A(z)$. Therefore, $\det V(z)$ is a nonzero rational function, and it follows that $\det P(z) = \det A(z)$. Moreover, by multiplicativity of the determinant,

$$\det\left(z^{\alpha}P(z^{\beta})\right) = \det\left(z^{\alpha}A(z^{\beta})\right).$$

□

The finite eigenvalues of $P(z)$ are the roots of $\det P(z)$. Therefore, by Theorem 5.5, the finite eigenvalues of $P(z)$ and $A(z)$ are equal. The infinite eigenvalues of $P(z)$ are defined as the zero valued roots of $\det\left(z^{d}P(z^{-1})\right)$. If we let $\alpha = d$ and $\beta = -1$, then Theorem 5.5 implies that

$$\det\left(z^{d}P(z^{-1})\right) = \det\left(z^{d}A(z^{-1})\right),$$

and we can obtain the infinite eigenvalues of $P(z)$ from $A(z)$.

Example 5.3. *Let* $P(z) = \begin{bmatrix} z^2 & z & i \\ z & z^2 & z \\ -i & z & z \end{bmatrix}$. *Applying the Arnoldi process we find*

$$V(z) = \begin{bmatrix} 0 & z & \frac{1}{2}z(z^2 - z + 2i) \\ 1 & 0 & 0 \\ 0 & z & -\frac{1}{2}z(z^2 - z + 2i) \end{bmatrix}, \; A(z) = \begin{bmatrix} z^2 & 2z^2 & 0 \\ 1 & \frac{1}{2}z(z+1) & \frac{1}{4}(z^4 - 2z^3 + z^2 + 4) \\ 0 & 1 & \frac{1}{2}z(z+1) \end{bmatrix},$$

such that $P(z)V(z) = V(z)A(z)$. *The finite eigenvalues of* $P(z)$ *are the roots of*

$$\det A(z) = z^5 - z^4 - z^3 - z^2.$$

The infinite eigenvalues of $P(z)$ *are the zero valued roots of*

$$\det\left(z^2 A(z^{-1})\right) = z - z^2 - z^3 - z^4.$$

Since there is one zero valued root of the above polynomial, $P(z)$ *has one infinite eigenvalue.*

There are several important points from Example 5.3 to make. First, we are able to obtain the finite and infinite eigenvalues of $P(z)$ from $A(z)$. Second, $P(z)$ was self-adjoint, a matrix polynomial is self-adjoint if all of its coefficient matrices are self-adjoint, and the resulting matrix rational was tridiagonal. Finally, the elements of $A(z)$ were all polynomials. It seems that there are starting vectors for the Arnoldi process which force the elements of $A(z)$ to be polynomials. For example, if the starting vector in Example 5.2 were $v = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$, then the elements of $A(z)$ would have been polynomials. Existence and identification of these starting vectors for any matrix polynomial is a topic of future research.

Theorem 5.6 ([10, Theorem 3.4])**.** *Let $P(z)$ be an $(n \times n)$ matrix polynomial and $v(z) \in \mathbb{F}^n$ be a nonzero vector such that the vectors*

$$v(z),\, P(z)v(z),\, \ldots,\, P^{n-1}(z)v(z)$$

are linearly independent. Then the Arnoldi process outlined in (5.2.2) will produce an orthogonal basis, $v_1, v_2, \ldots, v_n$, for $\mathbb{F}^n$. Moreover, the matrix representation of $P(z)$ with respect to this basis is upper Hessenberg, and if $P(z)$ is self-adjoint then the matrix representation is tridiagonal.

Proof. Let $V(z) = \begin{bmatrix} v_1(z) & \cdots & v_n(z) \end{bmatrix}$ and define $A(z) = [a_{ij}(z)]_{i,j=1}^n$, where

$$a_{ij}(z) = \begin{cases} \frac{\langle P(z)v_j(z), v_i(z)\rangle}{\langle v_i(z), v_i(z)\rangle} & \text{if } j \geq i \\ \beta_{j+1} & \text{if } j = i-1 \\ 0 & \text{if } j < i-1 \end{cases}.$$

Then the columns of $V(z)$ are linearly independent and (5.2.2) assures that $P(z)V(z) = V(z)A(z)$.

Suppose now that $P(z)$ is self-adjoint, and consider the generalized Fourier coefficients

$$a_{ij}(z) = \frac{\langle P(z)v_j(z),\, v_i(z)\rangle}{\langle v_i(z),\, v_i(z)\rangle},$$

when $j > i$. Since $P(z)$ is self-adjoint, we have $\langle P(z)v_j(z),\, v_i(z)\rangle = \langle v_j(z),\, P(z)v_i(z)\rangle$. Moreover,

$$P(z)v_i(z) = \sum_{k=1}^{i+1} a_{ki}(z)v_k(z).$$

Therefore, $a_{ij}(z) = 0$ whenever $j > i+1$, and it follows that $A(z)$ is tridiagonal. □

Let $P(z)$ be an $n \times n$ matrix polynomial and $v(z)$ be a nonzero vector for which Theorem 5.6 holds. Then we can use the Arnoldi process to find an upper Hessenberg matrix $A(z) \in \mathbb{F}^{n\times n}$ that is R-similar to $P(z)$, and by Theorem 5.5 we can obtain the finite and infinite eigenvalues of $P(z)$ from $A(z)$. If the matrix polynomial $P(z)$ is self-adjoint, then (5.2.2) becomes the three term recurrence.

$$\beta_{j+1}v_{j+1} = \left(P - \frac{\langle Pv_j,\, v_j\rangle}{\langle v_j,\, v_j\rangle}I\right)v_j - \frac{\langle Pv_j,\, v_{j-1}\rangle}{\langle v_{j-1},\, v_{j-1}\rangle}v_{j-1}. \tag{5.2.3}$$

5.2.2 Invariant Subspaces

We are not guaranteed to be able to form an n-dimensional Krylov subspace from every starting vector v. It may happen that k orthogonal vectors have been produced, where $k < n$, and applying (5.2.2) gives $v_{k+1} = 0$. In this case, the Krylov subspace $K_k(P,\, v)$ is a subspace of $\mathbb{F}^n$ which is invariant under multiplication by P. That is, $PK_k(P, v) \subseteq K_k(P, v)$. We begin with an example of a one dimensional invariant subspace.

Example 5.4. *Let* $P(z) = \begin{bmatrix} z^2+1 & 0 & 1 \\ 0 & z^2+1 & 0 \\ 1 & 0 & z^2+1 \end{bmatrix}$ *and* $v = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$. *Then*

$$P(z)v = (z^2+1)v.$$

It follows that $S = span\{v\}$ *forms a* 1 *dimensional subspace of* $\mathbb{F}^n$, *which is invariant under multiplication by* $P(z)$. *Moreover,* v *is an eigenvector corresponding to the eigenvalues* $\lambda = \pm i$.

Example 5.4 provides important insight into the eigenvalue problem for matrix polynomials. We have chosen to define finite eigenvalues as the roots of $\det P(z)$, as was done in [7, 8, 16, 30] and the references therein. However, if we consider the matrix polynomial $P(z)$ from Example 5.4 and let

$f(z) = z^2 + 1$, then there is a sense in which $f(z)$ is an eigenvalue of $P(z)$. This idea gains traction when one considers the matrix polynomial $P(z)$ as a linear operator from $\mathbb{F}^n$ to $\mathbb{F}^n$. Moreover, the roots of $f(z)$ are zeros of $\det P(z)$, so there seems to be a correlation between the two competing definitions of an eigenvalue of $P(z)$. Studying these definitions and their relationships is a topic of future research.

For our current discussion, what is important is that invariant subspaces can arise while performing the Arnoldi process, as the following example illustrates.

Example 5.5. *Let* $P(z) = \begin{bmatrix} z^2 & z & 1 & 0 \\ z & z^2 & z & 1 \\ 1 & z & z^2 & z \\ 0 & 1 & z & z^2 \end{bmatrix}$, $v_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$, $v_2 = \begin{bmatrix} -\frac{z}{2} \\ \frac{z}{2} \\ \frac{z}{2} \\ -\frac{z}{2} \end{bmatrix}$. *Then* $S = span\{v_1, v_2\}$ *is a* $2-$*dimensional subspace of* $\mathbb{F}^n$ *which is invariant under multiplication by* $P(z)$. *In fact, we found this invariant subspace by applying the Arnoldi process to starting vector* v_1. *Therefore, we have* $V(z) = \begin{bmatrix} v_1(z) & v_2(z) \end{bmatrix}$ *and*

$$A(z) = \begin{bmatrix} 1 + \frac{3}{2}z + z^2 & \frac{z^2}{4} \\ 1 & -1 - \frac{z}{2} + z^2 \end{bmatrix},$$

such that $P(z)V(z) = V(z)A(z)$.

Example 5.5 highlights the fact that coming across an invariant subspace while performing the Arnoldi process is a good thing. Since $\det A(z)$ divides $\det P(z)$, see Theorem 5.7, it follows that all the finite eigenvalues of $A(z)$ are also finite eigenvalues of $P(z)$.

5.2.3 Continue the Arnoldi Process

Suppose that after $k < n$ steps of the Arnoldi process outlined in Section 5.2.1, we find that $v_{k+1} = 0$. Then the Krylov subspace $K_k(P, v)$ is $k-$dimensional subspace of $\mathbb{F}^n$ which is invariant under multiplication by P. We can continue the Arnoldi process, but not with the vector $v_{k+1} = 0$. So, we choose v_{k+1} to be a nonzero vector that is orthogonal to each vector in $K_k(P, v)$ and continue on. In this way we can complete our task of finding an orthogonal basis of $\mathbb{F}^n$ such that the matrix representation of P with respect to this basis is upper Hessenberg.

Example 5.6. *Here we continue Example 5.5. We choose v_3 to be orthogonal to v_1 and v_2 and the continue the Arnoldi process to obtain*

$$V(z) = \begin{bmatrix} 1 & -\frac{z}{2} & \frac{1}{2} & 0 \\ 1 & \frac{z}{2} & 0 & \frac{1}{2}(z-1) \\ 1 & \frac{z}{2} & 0 & -\frac{1}{2}(z-1) \\ 1 & -\frac{z}{2} & -\frac{1}{2} & 0 \end{bmatrix}, \; A(z) = \begin{bmatrix} z^2 + \frac{3z}{2} + 1 & \frac{z^2}{4} & 0 & 0 \\ 1 & z^2 - \frac{z}{2} - 1 & 0 & 0 \\ 0 & 0 & z^2 & (z-1)^2 \\ 0 & 0 & 1 & z(z-1) \end{bmatrix}.$$

Note that in Example 5.6 we effectively split the problem of finding the eigenvalues of $P(z)$ into two smaller problems. This is known as deflation and we summarize this result in the following theorem.

Theorem 5.7 ([10, Theorem 3.8])**.** *Let $P(z)$ be a $(n \times n)$ matrix polynomial. Let $V_1(z) \in \mathbb{F}^{n \times k}$ be a matrix whose columns form a basis for a $k-$dimensional subspace which is invariant under multiplication by $P(z)$. Let $V_2(z) \in \mathbb{F}^{n \times (n-k)}$ be a matrix whose columns are additional vectors, such that the columns of $V(z) = \begin{bmatrix} V_1(z) & V_2(z) \end{bmatrix}$ form a basis of $\mathbb{F}^n$. Let $A(z) \in \mathbb{F}^{n \times n}$ be a matrix such that $P(z)V(z) = V(z)A(z)$, then*

$$A(z) = \begin{bmatrix} A_{11}(z) & A_{12}(z) \\ 0 & A_{22}(z) \end{bmatrix},$$

where $A_{11}(z) \in \mathbb{F}^{k\times k}$ *and* $A_{22}(z) \in \mathbb{F}^{(n-k)\times(n-k)}$. *Moreover,* $\det\left(z^{\alpha}P(z^{\beta})\right) = \det\left(z^{\alpha}A(z^{\beta})\right)$ *for all* α, $\beta \in \mathbb{Z}$.

Proof. The form of $A(z)$ follows readily from the fact that the columns of $V_1(z)$ form a basis for a $k-$dimensional invariant subspace which is invariant under multiplication by $P(z)$. The fact that $\det\left(z^{\alpha}P(z^{\beta})\right) = \det\left(z^{\alpha}A(z^{\beta})\right)$ for all α, $\beta \in \mathbb{Z}$, follows from Theorem 5.5. □

Theorem 5.7 can easily be extended to multiple invariant subspaces. Moreover, if the vectors in $V(z)$ come from the Arnoldi process described in Section 5.2.1, then $A_{11}(z)$ and $A_{22}(z)$ will be upper Hessenberg.

In conclusion, every $(n \times n)$ matrix polynomial $P(z)$ is R-similar to an upper Hessenberg $A(z) \in \mathbb{F}^{n\times n}$. Moreover, by Theorem 5.5, the finite eigenvalues of $P(z)$ and $A(z)$ are equal and we can obtain the infinite eigenvalues of $P(z)$ from $A(z)$. Areas of future research include developing efficient numerical methods for computing $A(z)$, and methods for computing subspaces of $\mathbb{F}^n$, which are invariant under multiplication by $P(z)$, directly.

5.3 Scaling Factors and Smith Form

In Section 5.2 we proved that every $(n \times n)$ matrix polynomial $P(z)$ is R-similar to an upper Hessenberg $A(z)$ in $\mathbb{F}^{n\times n}$. We know that the finite eigenvalues of $P(z)$ are preserved under this transformation. In this section we assume that $A(z)$ is an upper Hessenberg matrix polynomial. Example 5.7 will show that, in general, the Smith form of $A(z)$ and $P(z)$ are not the same. However, the problems that cause the Smith form to not be preserved seem to be alleviated by a proper choice of scaling factor β_{j+1}, introduced in (5.2.2). Finally, we provide sufficient conditions under which two similar matrix polynomials have equivalent Smith forms.

Example 5.7. *Let* $P(z) = \begin{bmatrix} z^2 & z & 1 \\ z & z^2 & z \\ 1 & z & z \end{bmatrix}$. *Using the Arnoldi process outlined in (5.2.2), with a scaling factor of* 1*, we find*

$$V(z) = \begin{bmatrix} 0 & z & \frac{1}{2}z^2(z-1) \\ 1 & 0 & 0 \\ 0 & z & -\frac{1}{2}z^2(z-1) \end{bmatrix}, A(z) = \begin{bmatrix} z^2 & 2z^2 & 0 \\ 1 & \frac{1}{2}(z^2+z+2) & \frac{1}{4}z^2(z-1)^2 \\ 0 & 1 & \frac{1}{2}(z+2)(z-1) \end{bmatrix},$$

such that $P(z)V(z) = V(z)A(z)$*. The Smith Form of* $P(z)$ *and* $A(z)$ *are*

$$P(z) \sim \begin{bmatrix} 1 & 0 & 0 \\ 0 & z-1 & 0 \\ 0 & 0 & z^2(z-1)(z+1) \end{bmatrix}, A(z) \sim \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & z^2(z-1)^2(z+1) \end{bmatrix}.$$

Moreover, $P(z)$ *has only one infinite eigenvalue, while* $A(z)$ *has seven infinite eigenvalues.*

Example 5.7 shows that, in general, similarity transformations in $\mathbb{F}^{n\times n}$ do not preserve the Smith form of a matrix polynomial. It is important to note that $\det V(z) = 0$ when $z = 0$ and $z = 1$. Moreover, the elementary divisors of $P(z)$ and $A(z)$, corresponding to $z = \infty$ and $z = 1$, are different. We can use our scaling factor β_{j+1} to simplify the columns of $V(z)$, so that this is no longer a problem.

Example 5.8. *Let* $P(z) = \begin{bmatrix} z^2 & z & 1 \\ z & z^2 & z \\ 1 & z & z \end{bmatrix}$ *and* $v_1(z) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$. *Using the Arnoldi process, with scaling factor* β_{j+1} *defined to be the greatest common divisor among the entries of* $\hat{v}_{j+1}$ *in (5.2.2),*

we find

$$V(z) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & -1 \end{bmatrix}, A(z) = \begin{bmatrix} z^2 & 2z & 0 \\ z & \frac{1}{2}(z^2+z+2) & \frac{z}{2}(z-1) \\ 0 & \frac{z}{2}(z-1) & \frac{1}{2}(z+2)(z-1) \end{bmatrix},$$

such that $P(z)V(z) = V(z)A(z)$. *Since* $\det V(z)$ *is a nonzero constant, we know that the Smith form of* $A(z)$ *and* $P(z)$ *are the same 1.2.1. In fact, both the finite and infinite elementary divisors of* $P(z)$ *and* $A(z)$ *are equal.*

Whether or not it is always possible to choose a scaling factor β_{j+1} in (5.2.2), so that the matrix $A(z)$ has the same Smith form as the original matrix polynomial $P(z)$ is a topic of future research. In what follows we will provide sufficient conditions under which the Smith form of two similar matrix polynomials are equivalent.

Recall that every matrix polynomial $P(z)$ admits the representation

$$P(z) = E(z)D(z)F(z).$$

It is common to refer to the diagonal matrix polynomial $D(z)$ as the Smith form of $P(z)$. The diagonal entries of $D(z)$ are the invariant polynomials. Moreover, if we represent each invariant polynomial as a product of linear factors

$$d_i(z) = (z-\lambda_{i1})^{\alpha_{i1}} \cdots (z-\lambda_{i,k_i})^{\alpha_{i,k_i}},$$

then the factors $(z-\lambda_{ij})^{\alpha_{ij}}$, $j = 1, \ldots, k_i$, $i = 1, \ldots, r$, are called the elementary divisors of $P(z)$. The invariant polynomials and elementary divisors are closely related to the Jordan chains of a

matrix polynomial. In what follows we will use several results from Section 1.2 to prove our main result, Theorem 5.11.

Lemma 5.8 (Proposition 1.2)**.** *Let $P(z)$ be an $(n \times n)$ matrix polynomial and let $S(z)$ and $V(z)$ be $n \times n$ matrix polynomials such that $S(\lambda)$ and $V(\lambda)$ are nonsingular for some $\lambda \in \mathbb{C}$. Then $y_0, \ldots, y_k$ is a Jordan chain of the matrix polynomial $S(z)P(z)V(z)$ corresponding to λ if and only if the vectors*

$$z_j = \sum_{i=0}^{j} \frac{1}{i!} V^{(i)}(\lambda) y_{j-i}, \quad j = 0, 1, \ldots, k$$

form a Jordan chain of $P(z)$ corresponding to λ.

From Lemma 5.8, it follows that the matrix polynomials $A(z)$ and $V(z)A(z)$ have the same set of Jordan chains corresponding to $\lambda \in \mathbb{C}$, if $\det V(\lambda) \neq 0$. We will use this fact in the following theorem.

Theorem 5.9 ([10, Theorem 4.4])**.** *Suppose $P(z)V(z) = V(z)A(z)$, where $P(z)$, $V(z)$, $A(z)$ are $(n \times n)$ matrix polynomials. The length of the Jordan chains of $A(z)$ corresponding to λ are equal to the lengths of the Jordan chains of $P(z)$ corresponding to λ, if* $\det V(\lambda) \neq 0$.

Proof. Let $y_0, \ldots, y_k$ be a Jordan chain of $A(z)$ corresponding to $\lambda \in \mathbb{C}$, where $\det V(\lambda) \neq 0$. By Lemma 5.8, $y_0, \ldots, y_k$ is a Jordan chain of $V(z)A(z)$ corresponding to λ. Therefore, $y_0, \ldots, y_k$ is a Jordan chain of $P(z)V(z)$ corresponding to λ. Again by Lemma 5.8, it follows that the vectors

$$z_j = \sum_{i=0}^{j} \frac{1}{i!} V^{(i)}(\lambda) y_{j-i}, \quad j = 0, 1, \ldots, k$$

form a Jordan chain of $P(z)$ corresponding to λ. □

As a consequence of Theorem 5.9 we have the following result.

Corollary 5.10 ([10, Corollary 4.5]). *Let $P(z)V(z) = V(z)A(z)$, where $P(z)$, $V(z)$, $A(z)$ are $(n \times n)$ matrix polynomials. The degree of the elementary divisors of $A(z)$ corresponding to λ are equal to the degree of the elementary divisors of $P(z)$ corresponding to λ, if $\det V(\lambda) \neq 0$.*

Proof. It follows from Theorem 5.9 that the lengths $k_1, \ldots, k_r$ of the Jordan chains of $A(z)$, corresponding to λ, are equal to the lengths of Jordan chains of $P(z)$, corresponding to λ. Therefore, by Proposition 1.3, the nonzero partial multiplicities of $A(z)$ at λ are equal to the nonzero partial multiplicities of $P(z)$ at λ. Since the nonzero partial multiplicities coincide with the degrees of the elementary divisors corresponding to λ, the result follows. □

One can use the number of invariant polynomials and the elementary divisors to uniquely determine the Smith form of a matrix polynomial. We will use this fact to prove the following.

Theorem 5.11 ([10, Theorem 4.6]). *Let $P(z)$ be a regular $(n \times n)$ matrix polynomial. Let $A(z)$ be a matrix polynomial that is R-similar to $P(z)$. Then the Smith form of $P(z)$ and $A(z)$ are equivalent, if none of the finite eigenvalues of $P(z)$ are also eigenvalues of $V(z)$.*

Proof. Since $P(z)$ is regular, we know that $\det P(z)$ is nonzero and therefore $P(z)$ has n invariant polynomials. The matrix polynomial $A(z)$ is R-similar to $P(z)$, therefore $A(z)$ also has n invariant polynomials. Let $\lambda \in \mathbb{C}$ be an eigenvalue of $A(z)$, then λ is also an eigenvalue of $P(z)$. If the $\det V(\lambda)$ is nonzero, then it follows from Corollary 5.10 that the elementary divisors of $A(z)$ and $P(z)$ corresponding to λ are the same. Therefore, if none of the finite eigenvalues of $P(z)$ are also eigenvalues of $V(z)$, then $A(z)$ and $P(z)$ have the same number of invariant polynomials and the same elementary divisors. It follows that the Smith form of $P(z)$ and $A(z)$ are equivalent. □

Chapter 6

Conclusions

Matrix polynomials are an important area of research which deserve attention from a large range of mathematicians. Insightful generalizations can be made when moving from the general eigenvalue problem to the matrix polynomial eigenvalue problem. Moreover, there is an urgent need for numerical algorithms which solve the polynomial eigenvalue problem, for a large range of matrix polynomials, and provide condition and error estimates. In this dissertation we have provided an algorithm which meets these qualifications.

In Chapter 2 we developed a root-finding algorithm, powered by Laguerre's method, which computes the eigenvalues of a matrix polynomial. Careful consideration was given to initial estimates and a priori identification of zero and infinite eigenvalues. Moreover, an efficient method for computing right and left eigenvectors, backward error, and condition estimates was given. Several examples were provided where the accuracy of our algorithm outperformed the competition. Research on the use of the reversal polynomial to prevent harmful overflow for large degree matrix polynomials is in progress. Future research includes further analysis of the use of the numerical range for initial estimates, and a generalization of Descartes' rule of signs for matrix polynomials.

In Chapters 3 and 4 we specialized our root finding algorithm for Hessenberg and tridiagonal structure. This was done by introducing Hyman's method and generalizing it for use on matrix polynomials. This generalization of Hyman's' method allows us to efficiently compute the Laguerre correction term of both Hessenberg and tridiagonal matrix polynomials. Numerical experiments were provided to confirm the efficiency and accuracy of these algorithms. Moreover, specific applications were provided and solved, for both Hessenberg and tridiagonal structures. Further research includes the consideration of other structures that appear naturally in matrix polynomials.

In Chapter 5 we introduced a pseudo inner product suitable for the vector space $\mathbb{F}^n$, where $\mathbb{F}$ is the field of rational functions. Armed with this pseudo inner product, we defined orthogonality of vectors in $\mathbb{F}^n$ and showed that we can apply the Arnoldi method directly to matrix polynomials. This allowed us to construct an upper Hessenberg matrix in $\mathbb{F}^{n\times n}$ that is similar to the original matrix polynomial. We proved that this transformation preserves the determinant, and we saw scenarios in which the finite and infinite elementary divisors are also preserved. Future research includes developing numerical methods for computing this transformation efficiently, finding optimal starting vectors for the Arnoldi process, and determining if we can always do this transformation in such a way that preserves the Smith form of the original matrix polynomial.

Appendix A

Algorithms

In this appendix pseudo-code is provided for algorithms described throughout the dissertation. All variables used in the pseudo-code are described in the corresponding sections of the dissertation. For a more detailed reference of each algorithm, see the source code at `http://www.d.umn.edu/~camer133/code/code.xhtml`.

Algorithm 1 XSLM

Require: $a_0, a_1, \ldots, a_d$

Compute upper convex hull of set $\{i, \log|a_i|\}$.
Compute the abscissas of the vertices $k_0 = 0 < k_1 < \cdots < k_q = d$
for $i = 0$ **to** q **do**
Place $k_i - k_{i-1}$ initial estimates on circles centered at 0 with radius

$$r_i = \left| \frac{a_{k_{i-1}}}{a_{k_i}} \right|^{\frac{1}{k_i - k_{i-1}}}$$

end for
For each root approximation λ
while stopping criteria is unmet **do**
$\lambda \leftarrow \lambda - \frac{d}{S_1 \pm \sqrt{(d-1)(d*S_2 - S_1^2}}$
end while
return λ and $\eta(\lambda) = \frac{|p(\lambda)|}{\alpha}$

Algorithm 2 XGESTART

Require: A_0, A_1, $\ldots$, A_d, and n, d, τ
  Compute Q, R, P from the QR factorization with column pivoting of A_d
  Compute jlo, the lowest index such that $|R(j,j))| < \tau$
  Set $die = n - jlo + 1$
  for $j = jlo$ **to** n **do**
    Set $x(jlo : j-1) = 0$, $x(j) = 1$, $x(j+1 : n) = 0$
    Solve $R(1 : jlo-1, 1 : jlo-1)x(1 : jlo-1) = -R(1 : jlo-1, j)$
    return $x \leftarrow \frac{Px}{\|Px\|_2}$
    return $y \leftarrow$ jth column of Q
    return $\eta(\lambda, x) = \frac{\|A_d x\|_2}{\|A_d\|}$, $\eta(\lambda, y) = \frac{\|y^* A_d\|}{\|A_d\|}$, $\kappa(\lambda, P) = \frac{1}{|y^* x|}$
  end for
  Repeat for A_0 and dze
  for $i = 1$ **to** n **do**
    Set random vector x
    $x \leftarrow Qx$
    Use XSLM to compute roots of polynomial with coefficients x^*A_0x, x^*A_1x, $\ldots$, x^*A_dx
  end for
  return *die* and *dze* infinite and zero eigenpairs
  return root approximations for remaining finite eigenvalues

Algorithm 3 XGELCORR

Require: A_0, A_1, $\ldots$, A_d, and n, d, k, $\lambda, \epsilon,\ r_1, \ldots, r_k$
  Set dd=n*d-k
  Compute LU factorization of $P(\lambda)$
  Use LU factorization to compute backward error upper bound $\eta(\lambda)$
  if $\eta(\lambda) < \epsilon$ **then**
    return λ
  else
    Solve $P(\lambda)X_1 = P^{'}(\lambda)$ and $P(\lambda)X_2 = P^{''}(\lambda)$
    Compute $s_1 = \text{trace}\,(X_1)$ and $s_2 = \text{trace}\left(X_1^2\right) - \text{trace}\,(X_2)$
    $s_1 \leftarrow s_1 - \sum_{i=1}^{k} \frac{1}{\lambda - r_i}$
    $s_2 \leftarrow s_2 - \sum_{i=1}^{k} \frac{1}{(\lambda - r_i)^2}$
    return $\lambda \leftarrow \lambda - \frac{dd}{s_1 \pm \sqrt{(dd-1)(dd*s_2 - s_1^2)}}$
  end if

Algorithm 4 XGEEVC

Require: A_0, A_1, ..., A_d and λ, τ, α

Compute Q, R from the QR factorization of $P(\lambda)$

if $|R_{i,i}| < \tau$ for some $i \in \{1, \ldots, n\}$ **then**

Set i to first index for which $|R_{i,i}| < \tau$

Set $x(i) = 1$, $x(i+1:n) = 0$

Solve $R(1:i-1, 1:i-1)x(1:i-1) = -R(1:i-1, i)$

$x \leftarrow \frac{x}{\|x\|_2}$

Set i to last index for which $|R_{i,i}| < \tau$

Set $y(i) = 1$, $y(1:i-1) = 0$

Solve $R^*(i+1:n, i+1:n)y(i+1:n) = -R^*(i, i+1:n)$

$y \leftarrow \frac{Qy}{\|Qy\|_2}$

else

Set i to index which minimizes $|R(i,i)|$

Set $x(i) = 1$, $x(i+1:n) = 0$

Solve $R(1:i-1, 1:i-1)x(1:i-1) = -R(1:i-1, i)$

$x \leftarrow \frac{x}{\|x\|_2}$

Set $y(i) = 1$, $y(1:i-1) = 0$

Solve $R^*(i+1:n, i+1:n)y(i+1:n) = -R^*(i, i+1:n)$

$y \leftarrow \frac{Qy}{\|Qy\|_2}$

for $it = 1$ **to** 3 **do**

Solve $R^*R\hat{x} = x$

Solve $QRR^*Q^*\hat{y} = y$

$x \leftarrow \frac{\hat{x}}{\|\hat{x}\|_2}$

$y \leftarrow \frac{\hat{y}}{\|\hat{y}\|_2}$

end for

end if

return x, y

return $\eta(\lambda, x) = \frac{\|P(\lambda)x\|_2}{\alpha}$, $\eta(\lambda, y) = \frac{\|y^*P(\lambda)\|}{\alpha}$, $\kappa(\lambda, P) = \frac{\alpha}{|\lambda|\left|y^*P'(\lambda)x\right|}$

Algorithm 5 XHSLCORR

Require: A_0, A_1, ..., A_d, and n, d, k, $\lambda, \epsilon,\ r_1, \ldots, r_k$

Set $dd = n * d - k$

Compute LU factorization of $P(\lambda)$

Use LU factorization to compute backward error upper bound $\eta(\lambda)$

if $\eta(\lambda) < \epsilon$ **then**

return λ

else

Compute $A = P(\lambda)$, $B = P^{'}(\lambda)$, $C = P^{''}(\lambda)$

Solve $A(2:n, 1:n-1)x(1:n-1) = -A(2:n, n)$

Solve $A(2:n, 1:n-1)x^{'}(1:n-1) = -B(2:n, 1:n)x(1:n)$

Solve $A(2:n, 1:n-1)x^{''}(1:n-1) = -2B(2:n, 1:n)x^{'}(1:n) - C(2:n, 1:n)x(1:n)$

$b \leftarrow A(1, 1:n)x(1:n)$

$b^{'} \leftarrow A(1, 1:n-1)x^{'}(1:n-1) + B(1, 1:n)x(1:n)$

$b^{''} \leftarrow A(1, 1:n-1)x^{''}(1:n-1) + 2B(1, 1:n-1)x^{'}(1:n-1) + C(1, 1:n)x(1:n)$

$y_1 = \sum_{j=1}^{n-1} \frac{B(j+1,j)}{A(j+1,j)}$

$y_2 = \sum_{j=1}^{n-1} \left(\frac{C(j+1,j)}{A(j+1,j)} - \left(\frac{B(j+1,j)}{A(j+1,j)} \right)^2 \right) + y_1^2$

$s_1 = \frac{b^{'} + by_1}{b}$

$s_2 = y_1^2 - \frac{b^{''} + 2b^{'}y_1 + by_2}{b}$

$s_1 \leftarrow s_1 - \sum_{i=1}^{k} \frac{1}{\lambda - r_i}$

$s_2 \leftarrow s_2 - \sum_{i=1}^{k} \frac{1}{(\lambda - r_i)^2}$

return $\lambda \leftarrow \lambda - \frac{dd}{s_1 \pm \sqrt{(dd-1)(dd*s_2 - s_1^2)}}$

end if

Algorithm 6 XGTQR

Require: pdl, pd, pdu, λ

Set $adl = pdl(\lambda)$, $ad = pd(\lambda)$, $adu = pdu(\lambda)$

for $i = 1$ **to** $n - 1$ **do**

Set $r = \sqrt{|ad(j)|^2 + |adl(j)|^2}$

Set $c(j) = adj(j)/r$, $s(j) = adl(j)/r$

$rd(j) \leftarrow r$

$$\begin{bmatrix} adu(j) \\ ad(j+1) \end{bmatrix} \leftarrow \begin{bmatrix} \bar{c} & \bar{s} \\ -s & c \end{bmatrix} \begin{bmatrix} adu(j) \\ ad(j+1) \end{bmatrix}$$

$$\begin{bmatrix} adl(j) \\ adu(j+1) \end{bmatrix} \leftarrow \begin{bmatrix} \bar{c} & \bar{s} \\ -s & c \end{bmatrix} \begin{bmatrix} 0 \\ adu(j+1) \end{bmatrix}$$

end for

return ad, adu, adl, c, s

Algorithm 7 XGTQM

Require: c, s, x, trans
  if trans $=' N'$ **then**
    for i=n-1 **to** 1 **do**
$$\begin{bmatrix} x(i) \\ x(i+1) \end{bmatrix} \leftarrow \begin{bmatrix} c & -\bar{s} \\ s & \bar{c} \end{bmatrix} \begin{bmatrix} x(i) \\ x(i+1) \end{bmatrix}$$
    end for
  else
    for i=1 **to** n-1 **do**
$$\begin{bmatrix} x(i) \\ x(i+1) \end{bmatrix} \leftarrow \begin{bmatrix} \bar{c} & \bar{s} \\ -s & c \end{bmatrix} \begin{bmatrix} x(i) \\ x(i+1) \end{bmatrix}$$
    end for
  end if
  return x

Algorithm 8 XGTRS

Require: rd, rdu, $rdu2$, b, trans
  if trans $=' N'$ **then**
    for j=n **to** 1 **do**
      $b(j) \leftarrow b(j)/rd(j)$
$$\begin{bmatrix} b(j-1) \\ b(j-2) \end{bmatrix} \leftarrow \begin{bmatrix} b(j-1) \\ b(j-2) \end{bmatrix} - b(j) \begin{bmatrix} rdu2(j-2) \\ rdu(j-1) \end{bmatrix}$$
    end for
  else
    for j=1 **to** n **do**
      $b(j) \leftarrow b(j)/\overline{rd(j)}$
$$\begin{bmatrix} b(j+1) \\ b(j+2) \end{bmatrix} \leftarrow \begin{bmatrix} b(j+1) \\ b(j+2) \end{bmatrix} - b(j) \begin{bmatrix} \overline{rdu(j)} \\ \overline{rdu(j)} \end{bmatrix}$$
    end for
  end if

Algorithm 9 XGTSTART

Require: pdl, pd, pdu and n, d, τ
  Use XGTQR to compute Q, R from the QR factorization of A_d.
  Set $die = 0$
  for $i = 1$ **to** n **do**
    if $|ad(i)| < \tau$ **then**
      Check if x_i is free in equation $Rx = 0$
      if x_i is free **then**
        Compute corresponding right eigenvector
        $die \leftarrow die + 1$
      end if
    end if
  end for
  for $i = n$ **to** 1 **do**
    if $|ad(i)| < \tau$ **then**
      Check if y_i is free in equation $R^*y = 0$
      If free, compute corresponding left eigenvector
    end if
  end for
  Repeat for A_0 and dze
  for $i = 1$ **to** n **do**
    Set random vector x
    Use XGTQM to compute $x \leftarrow Qx$
    Use XSLM to compute roots of polynomials with coefficients x^*A_0x, x^*A_1x, ..., x^*A_dx
  end for
  return die and dze infinite and zero eigenpairs
  return root approximations for remaining finite eigenvalues

Algorithm 10 XGTEVC

Require: *pdl*, *pd*, *pdu* and λ, τ, α

Use XGTQR to compute Q, R from QR factorization of $P(\lambda)$

if $|ad(i)| < \tau$ for some $i \in \{1, \ldots, n\}$ **then**

Set i to first index for which $|ad(i)| < \tau$

Set $x(i) = 1$, $x(i+1:n) = 0$

Use XGTRS to solve $R(1:i-1, 1:i-1)x(1:i-1) = -R(1:i-1, i)$

$x \leftarrow \frac{x}{\|x\|_2}$

Set i to last index for which $|ad(i)| < \tau$

Set $y(i) = 1$, $y(1:i-1) = 0$

Use XGTRS to solve $R^*(i+1:n, i+1:n) = -R^*(i, i+1:n)$

Use XGTQM to compute $y \leftarrow Qy$

$y \leftarrow \frac{y}{\|y\|_2}$

else

Set i to index which minimizes $|ad(i)|$

Set $x(i) = 1$, $x(i+1:n) = 0$

Use XGTRS to solve $R(1:i-1, 1:i-1)x(1:i-1) = -R(1:i-1, i)$

$x \leftarrow \frac{x}{\|x\|_2}$

Set $y(i) = 1$, $y(1:i-1) = 0$

Solve $R^*(i+1:n, i+1:n)y(i+1:n) = -R^*(i, i+1:n)$

Use XGTQM to compute $y \leftarrow Qy$

$y \leftarrow \frac{y}{\|y\|_2}$

for $it = 1$ **to** 3 **do**

USE XGTRS and XGTQM to solve

$R^*R\hat{x} = x$ and $QRR^*Q^*\hat{y} = y$

$x \leftarrow \frac{x}{\|x\|_2}$

$y \leftarrow \frac{y}{\|y\|_2}$

end for

end if

return x, y

return $\eta(\lambda, x) = \frac{\|P(\lambda)x\|_2}{\alpha}$, $\eta(\lambda, y) = \frac{\|y^*P(\lambda)\|}{\alpha}$, $\kappa(\lambda, P) = \frac{\alpha}{|\lambda|\left|y^*P'(\lambda)x\right|}$

Appendix B

Applications

In the following experiments we will solve many of the polynomial eigenvalue problems provided in NLEVP [3]. For all parameter dependent problems the default values are chosen. The letters next to the problem name, if any, indicates the structure detected by LMPEP and the corresponding version of LMPEP that is called. The maximum right backward error is recorded for LMPEP and compared to the linearization based methods QUADEIG [17] and POLYEIG [13, 24, 34]. If the degree of the matrix polynomial is greater then 2, then QUADEIG is not applicable and therefore not recorded.

Table B.1 shows that LMPEP provides competitive results in comparison to the accuracy of the standard linearization method as implemented by QUADEIG and POLYEIG. We make special note of the problems damped-beam, orr-sommerfeld, pdde-stability, plasma-drift, wiresaw1, and wiresaw2; in all of these problems LMPEP was several orders of magnitude better then the competition.

MAX BEX for LMPEP, QUADEIG, POLYEIG			
Problem	LMPEP	QUADEIG	POLYEIG
bicycle(GT)	1.1E-016	6.0E-017	1.5E-015
bilby(HS)	1.2E-016	7.2E-016	3.3E-016
butterfly	1.3E-016		3.8E-015
cd-player	1.1E-016	1.8E-015	4.3E-010
closed-loop(GT)	1.1E-016	1.0E-015	1.7E-016
damped-beam	4.6E-017	5.3E-015	2.2E-008
dirac	1.6E-016	2.5E-015	8.1E-015
gen-hyper2	8.4E-017	5.5E-016	8.8E-016
hospital	1.2E-016	1.4E-015	1.3E-012
intersection	2.0E-017	1.6E-016	4.6E-017
metal-strip	1.6E-016	1.2E-015	6.7E-014
mobile-manipulator	1.1E-018	5.8E-017	4.6E-019
omnicam1	6.8E-017	9.1E-017	1.2E-015
omnicam2	8.4E-017	4.5E-017	5.9E-017
orr-sommerfeld	7.8E-017		3.6E-005
pdde-stability	2.1E-016	1.5E-013	4.8E-013
plasma-drift	1.4E-015		7.7E-013
power-plant	8.5E-017	3.6E-016	1.2E-008
relative-pose-5pt	3.5E-017		1.2E-016
relative-pose-6pt	4.6E-017	5.4E-016	8.1E-016
shaft	1.3E-016	7.6E-015	8.0E-008
sign1	2.2E-016	3.6E-015	8.5E-016
sign2	2.0E-016	5.5E-015	2.3E-015
sleeper	4.3E-016	7.9E-016	7.1E-015
speaker-box	2.1E-017	4.5E-016	5.6E-011
spring(GT)	8.1E-017	8.1E-016	8.5E-016
spring-dashpot	6.2E-017	1.7E-016	5.1E-016
wing	2.0E-017	1.4E-016	3.6E-015
wiresaw1	5.3E-017	1.3E-015	4.3E-014
wiresaw2	4.8E-017	3.1E-015	8.8E-014

TABLE B.1: Accuracy Comparisons

Bibliography

[1] Forman S. Acton, *Numerical Methods that Work*, Harper and Row, New York, 1970.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Cruz Du, A. Geenbaum, S. Hammarling, A. Mckenney, S. Ostrouchov, and D. Sorensen, *LAPACK user's guide*, SIAM, Philadelphia, PA, 1992.

[3] Timo Betcke, Nicholas J. Higham, Volker Mehrmann, Christian Schröder, and Francoise Tisseur, *NLEVP: a collection of nonlinear eigenvalue problems*, Trans. Math. Software **39** (2013), no. 4, 28.

[4] Timo Betcke and Daniel Kressner, *Perturbation, computation and refinement of invariant pairs for matrix polynomials*, Linear Algebra Appl. **435** (2009), no. 3, 514–536.

[5] Dario A. Bini, *Numerical computation of polynomial zeros by means of Aberths method*, Numer. Algorithms **13** (1996), no. 2, 179–200.

[6] Dario A. Bini, Luca Gemignani, and Francoise Tisseur, *The Ehrlich-Aberth method for the nonsymmetric tridiagonal eigenvalue problem*, SIAM J. Matrix Anal. Appl. **27** (2005), no. 1, 153–175.

[7] Dario A. Bini and Vanni Noferini, *Solving polynomial eigenvalue problem by means of the Ehrlich-Aberth method*, Linear Algebra Appl. **439** (2013), no. 4, 1130–1149.

[8] Dario A. Bini, Vanni Noferini, and Meisam Sharify, *Locating the eigenvalues of matrix polynomials*, SIAM J. Matrix Anal. Appl. **34** (2013), no. 1, 1708–1727.

[9] Thomas R. Cameron, *Spectral bounds for matrix polynomials with unitary coefficients*, Electronic Journal of Linear Algebra **30** (2015), 585–591.

[10] ———, *On the reduction of matrix polynomials to Hessenberg form*, Electronic Journal of Linear Algebra **31** (2016), 321–334.

[11] ———, *On the application of laguerre's method to the polynomial eigenvalue problem*, Work in progress (2016-17).

[12] A.L. Cauchy, *Exercises de Mathématiques. Seconde Année*, 1827.

[13] Jean-Pierre Dedieu and Francoise Tisseur, Linear Algebra Appl. **358**, 71–94.

[14] John Gary, *Hyman's method applied to the general eigenvalue problem*, Mathematics of Computation **19** (1965), 314–316.

[15] I. Gohberg, S. Goldberg, and M.A. Kaashoek, *Classes of Linear Operators Vol. 1*, Springer Basel AG, Basel Switzerland, 1990.

[16] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*, SIAM, Philadelphia, PA, 2009.

[17] Sven J. Hammerling, Christopher J. Munro, and Tisseur Francoise, *An algorithm for the complete solution of quadratic eigenvalue problem*, Transactions on Mathematical Software **39** (2013), 19.

[18] L. Hoffnung, Li Ren-Cang, and Qiang Ye, *Krylov type subspace methods for matrix polynomials*, Linear Algebra Appl. **415** (2006), 52–81.

[19] E. Laguerre, *Oeuvres de Laguerre*, Paris Authier-Villars, 1898.

[20] P. Lancaster and P. Psarrakos, *The numerical range of self-adjoint quadratic matrix polynomials*, SIAM J. Matrix Anal. Appl. **23** (2001), 615–631.

[21] B.W. Levinger, *A generalization of Pellet's theorem concerning the zeros of a polynomial*, Proceedings of the AMS **18** (1967), 767–774.

[22] M. Marden, *A refinement of Pellet's theorem*, Bulletin of the AMS **54** (1948), 550–557.

[23] J Maroulas and P. Psarrakos, *On factorization of matrix polynomials*, Linear Algebra Appl. **304** (2000), 131–139.

[24] Karl Meerbergen and Francoise Tisseur, *The quadratic eigenvalue problem*, SIAM Review **43** (2001), no. 2, 235–286.

[25] Volker Mehrmann and Heinrich Voss, *Nonlinear eigenvalue problems: a challenge for modern eigenvalue methods*, Gamm-Mitt. **27** (2004), no. 2, 121–152.

[26] Aaron Melman, *Generalization and variations of Pellet's theorem for matrix polynomials*, Linear Algebra Appl. **439** (2013), 1550–1567.

[27] Vanni Noferini, Meisam Sharify, and Francoise Tisseur, *Tropical roots as approximations to eigenvalues of matrix polynomials*, SIAM J. Matrix Anal. Appl. **36** (2015), 138–157.

[28] Beresford Parlett, *Laguerre's method applied to the matrix eigenvalue problem*, Mathematics of Computation **18** (1964), 464–485.

[29] M.A. Pellet, *Sur un mode de séperation des racines des équations et la formule de lagrange*, Bulletin des Sciences Mathématiques **5** (1881), 393–395.

[30] B. Plestenjak, *Numerical methods for the tridiagonal hyperbolic quadratic eigenvalue problem*, SIAM J. Matrix Anal. Appl. **28** (2006), 1157–1172.

[31] R. Remmert, *Theory of Complex Functions*, Springer-Verlag, New York, 1991.

[32] L. Taslaman, F. Tisseur, and I. Zaballa, *Triangularizing matrix polynomials*, Linear Algebra Appl. **439** (2013), 1679–1699.

[33] F. Tisseur and I. Zaballa, *Triangularizing quadratic matrix polynomials*, SIAM J. Matrix Anal. Appl. **34** (2013), 312–337.

[34] Francoise Tisseur, *Backward error and condition of polynomial eigenvalue problem*, Linear Algebra Appl. **309** (2000), 339–361.

[35] J.L. Walsh, *On Pellet's theorem concerning the roots of a polynomial*, Annals of Mathematics **18** (1964), 464–485.

[36] David S. Watkins, *Some perspectives on the eigenvalue problem*, SIAM Review **35** (1993), 430–471.

[37] ———, *The Matrix Eigenvalue Problem*, SIAM, Philadelphia, PA, 2007.

[38] ———, *Fundamentals of Matrix Computations*, John Wiley and Sons, Inc., Hoboken, New Jersey, 2010.

[39] J.H. Wilkinson, *Rounding Errors in Algebraic Processes*, Prenctice-Hall, New Jersey, 1963.

[40] ———, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, England, 1965.