July 2013

Eigenvalue Methods for Interpolation Bases

Piers W. Lawrence
The University of Western Ontario

Supervisor
Corless, Robert M.
The University of Western Ontario

Graduate Program in Applied Mathematics

A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy

Eigenvalue Methods for Interpolation Bases

(Thesis format: Integrated-Article)

by

Piers William Lawrence

Graduate Program in Applied Mathematics

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

This thesis investigates eigenvalue techniques for the location of roots of polynomials expressed in the Lagrange basis. Polynomial approximations to functions arise in almost all areas of computational mathematics, since polynomial expressions can be manipulated in ways that the original function cannot. Polynomials are most often expressed in the monomial basis; however, in many applications polynomials are constructed by interpolating data at a series of points. The roots of such polynomial interpolants can be found by computing the eigenvalues of a generalized companion matrix pair constructed directly from the values of the interpolant. This affords the opportunity to work with polynomials expressed directly in the interpolation basis in which they were posed, avoiding the often ill-conditioned transformation between bases.

Working within this framework, this thesis demonstrates that computing the roots of polynomials via these companion matrices is numerically stable, and the matrices involved can be reduced in such a way as to significantly lower the number of operations required to obtain the roots.

Through examination of these various techniques, this thesis offers insight into the speed, stability, and accuracy of rootfinding algorithms for polynomials expressed in alternative bases.

Keywords: polynomial interpolation, Lagrange interpolation, barycentric formula, generalized companion matrices, polynomial roots, eigenvalue problem, stability, backward error, semiseparable matrices, nonlinear eigenvalue problem

Co-Authorship Statement

Chapters 3 and 4 were co-authored with Rob Corless, and have been submitted for publication. Rob Corless supervised the research, and provided key guidance for the formulation and proof of the main theorem of Chapter 3. Rob Corless also reviewed and revised earlier drafts of both chapters.

Acknowledgements

Although there may have been many opportunities for this thesis not to have come to fruition, it has, and I am grateful to many people who have made this possible.

First of all, had it not been for Rob Corless, none of this would have been possible. It was his visit to the University of Canterbury, at the precise moment I was contemplating pursuing a PhD, and his inspiring talk on barycentric Hermite interpolation which started the ball rolling. I would like to thank him for the many discussions we have had, and his patience explaining the intricacies of backward error analysis.

From the Department, I would like to thank Colin Denniston for always having an open door, and a willingness to discuss any topic of mathematics. I also thank him for providing the funds for our departmental morning teas, making the department feel a little more like home. I also thank David Jeffrey for sharing many humorous anecdotes, and his expert editorial advice.

Further abroad, I would like to thank Georges Klein, for the many stimulating discussions about barycentric interpolation, and for inviting me to visit Fribourg to work on rootfinding problems relating to Floater-Hormann interpolation.

From the University of Canterbury, I thank Chris Hann for showing me the path into mathematics and into research—I may well have become an engineer otherwise. I thank Bob Broughton and Mark Hickman for their stellar teaching and mentorship, and for sparking my interest in numerical linear algebra and numerical computation.

Many of my graduate student colleagues have had a hand in making this

experience more interesting, not least of all James Marshall, for the many discussions on any topic you wish to name. I thank Frances, Mona, Melissa, Alex, Walid, Bernard, and Anna for providing good companionship. I also thank Dave and Aimee for the excellent BBQ, and for exposing me to the beauty of the cottage in Canada.

To my parents, I thank them for supporting me in my endeavours into mathematics, even though these pursuits have taken me out of middle earth to the other side of the world.

Above all I thank Trish for her constant and unwavering love and support. She has always been there for me, and without her, this thesis surely would not have come into being. And thanks to Doug, for always being there for us.

Contents

[1] A version of this chapter has been submitted to the SIAM Journal of Matrix Analysis and Application for publication.

[2]A version of this chapter has been submitted to Numerical Algorithms for publication.
[3]A version of this chapter has been submitted to Linear Algebra and its Applications for publication.

List of Tables

List of Figures

List of Algorithms

List of Notation

$\text{adj}(\mathbf{A})$	The adjugate of $\mathbf{A}$: the transpose of the cofactor matrix of $\mathbf{A}$.
$[z^n](p(z))$	Coefficient of z^n of the polynomial $p(z)$.
$\|\mathbf{A}\|_F$	Frobenius norm of $\mathbf{A}$ defined by $$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} \|a_{ij}\|^2}$$
$\|(\mathbf{A},\mathbf{B})\|_F$	Frobenius norm of the matrix pair$(\mathbf{A},\mathbf{B})$ defined by $$\|(\mathbf{A},\mathbf{B})\|_F = \sqrt{\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2}\,.$$
$\eta_{(\mathbf{A},\mathbf{B})}(\lambda,\mathbf{z})$	Backward error of an approximate eigenpair $(\lambda,\mathbf{z})$ of the linearization $(\mathbf{A},\mathbf{B})$.
$\eta_P(\lambda,\mathbf{x})$	Backward error of an approximate eigenpair $(\lambda,\mathbf{x})$ of a matrix polynomial $\mathbf{P}(z)$.

Chapter 1

Introduction

1.1 Motivation

Solving univariate polynomial equations is an essential part of numerical analysis, and for a great many other areas in mathematics. It is well known that the accuracy of computing the roots of univariate polynomials is strongly affected by the basis in which the polynomial is originally expressed. Yet, most classical polynomial rootfinders require that polynomials be converted to the monomial basis first, before calling the rootfinder. Significant exceptions to this approach include the rootfinder used in the software package CHEBFUN [26] which is based upon the Chebyshev Colleague matrix [14, 22]; the algorithm proposed by Day and Romero [10] for orthogonal polynomial bases; and the Bernstein-Bézier package of Farouki and Rajan [12]. Conversion to another basis cannot be expected to improve the numerical conditioning of the polynomial, and indeed such conversion usually makes it much worse—exponentially worse, in the degree [15]. Therefore, conversion between bases should be avoided.

In this thesis, we explore the computation of the roots of polynomials via eigenvalue methods. Instead of solving the polynomial equation directly, we *linearize* the polynomial, which is to say the polynomial is transformed into a generalized eigenvalue problem whose eigenvalues are exactly the roots of the polynomial. For example, in the monomial basis, the roots of a polynomial

may be computed via the Frobenius companion matrix constructed from the polynomial coefficients. Working in the barycentric Lagrange basis has significant accuracy advantages [4, 9]. According to Higham [17], the first form of the barycentric interpolation formula is backward stable, and the second barycentric form is forward stable for any set of nodes with a small Lebesgue constant. The particular linearization we investigate is the generalized companion matrix pair first proposed by Corless [6], constructed from the coefficients of the polynomial in the Lagrange basis.

Forming the linearization of a polynomial has the effect of squaring the number of values involved: the generalized eigenvalue problem involves $O(n^2)$ entries. Furthermore, the complexity of computing the eigenvalues (via the QZ algorithm) is an $O(n^3)$ process. As pointed out by Moler [21], many people are willing to pay the $O(n^3)$ cost of this approach simply because of its convenience. Here we give a proof of the stability, and lower the cost.

In the literature to date, this particular linearization has been used in practice before any rigorous analysis of its stability has been conducted. Accordingly, one of the primary objectives of this thesis is to examine the stability properties of this linearization. Most studies investigating the stability properties of computing roots of polynomials via linearization consider only the monomial basis. The advantages of using the Lagrange basis for computations are significant, and thus there is a need to address the corresponding stability properties of rootfinding via linearization for the Lagrange basis.

To discuss the stability properties we first introduce the notion of condition numbers and backward errors, further details can be found in [8]. Consider the situation where we wish to compute the value of a function $y = f(x)$ for some given x. Lets say we have an algorithm that computes a solution $\widehat{y}$ approximating y. We ask the question: for what input data have we actually solved our problem? Or in other words for what perturbation δx is $\widehat{y} = f(x + \delta x)$ satisfied? The backward error is then just the smallest $|\delta x|$ for which the equation $\widehat{y} = f(x + \delta x)$ is satisfied. The forward error is measured by the difference between the approximate solution and the true solution of the problem, that is, $|\widehat{y} - y|$. The backward and forward errors are related

through the condition number, in general we have the relationship

$$\text{Forward error} \leq \text{Condition number} \times \text{Backward error}\,. \tag{1.1}$$

Thus, an algorithm is called backward stable if the backward error is small, because the backward error and condition number provide a bound on the forward error. If the problem which we are solving is well-conditioned, and the small backward error is small, then the forward error will also be small.

Furthermore, we are motivated by applications that give rise to polynomial eigenvalue problems, or matrix polynomials. We find significant discussion of the quadratic eigenvalue problem (QEP) in [24], a problem which has received much attention in recent years because of the vast variety of application areas that give rise to QEPs, including dynamical analysis of mechanical systems, acoustics, and linear stability of flows in fluid mechanics, to name a few. Similarly, higher order polynomial eigenvalue problems also arise in a number of application areas. It is not entirely clear that the monomial basis is the best basis for expressing matrix polynomials, and the Lagrange basis may yet prove to be superior for formulating and solving high-order matrix polynomials.

Born out of the need to produce accurate and stable numerical methods for solving the polynomial eigenvalue problem, this thesis addresses the matter of the numerical stability of computing eigenpairs of matrix polynomials expressed in the Lagrange basis. We demonstrate the conditions under which eigenpairs, computed via linearization of matrix polynomials, are numerically stable. We also show, through numerical examples, that by applying blockwise balancing to the linearization we are able to compute eigenpairs with small backward errors, and give computable and intelligible upper bounds for the backward errors.

One might ask *why not just use Newton's method to compute the roots?* The answer to this is that we wish to compute all of the roots, not just a selection. The eigenvalue techniques discussed in this thesis are robust and convenient for just this task. Therefore, we start there and modify the them for efficiency, instead of creating a special-purpose method.

1.2 Outline

Chapter 2 investigates the structured reduction of the linearization of scalar polynomials expressed in the Lagrange basis. The linearization we investigate has only $O(n)$ nonzero entries, and is a low-rank perturbation of a diagonal matrix. It is our goal to retain some sort of structure of the linearization during the solution process. Normally the eigenvalue problem is reduced first to upper Hessenberg form to reduce the complexity of carrying out QZ iterations. For real interpolation nodes, the Hessenberg form also has low-rank structure, being a rank one perturbation of a symmetric tridiagonal matrix. Thus, we should be able to transform the original linearization to another low-rank formulation. We show that such a transformation may be achieved in only $O(n^2)$ operations, by applying structured methods. Furthermore, once we have reduced the matrix to tridiagonal plus rank one form, there exist a variety of $O(n^2)$ methods to apply the QR algorithm to the matrix. Hence, we could obtain an $O(n^2)$ algorithm for locating all n roots of the original polynomial.

In Chapter 3, we investigate the stability of computing roots of polynomials via linearization. In the monomial basis, the conditions under which the Frobenius companion matrix produces accurate and stable roots of the polynomial has been established in [11]. For the Lagrange basis, the corresponding analysis is lacking, and filling this gap in the literature is one of the contributions that this thesis offers. We obtain bounds on the backward error that are easily computable and intelligible. Through a range of numerical experiments, we illustrate that the backward error is small, and that the bound is often only around one order of magnitude larger than the actual backward error. We also develop a balancing strategy for the linearization, to improve the accuracy of the computed eigenvalues.

In Chapter 4, we investigate the stability of computing eigenvalues and eigenvectors of matrix polynomials expressed in the Lagrange basis via linearization. We establish the conditions under which the linearization produces eigenpairs with small backward errors, and provide computable bounds for the backward errors. With the aid of a block-wise balancing strategy,

similar to that for the scalar case, we are able to compute eigenpairs with excellent backward errors for a range of numerical examples, and our bounds are approximately one order of magnitude larger.

In sum, this thesis offers new algorithms for the structured reduction of linearizations of polynomials in the Lagrange basis, and undertakes the necessary analysis to establish the numerical stability of computing roots and eigenvalues of polynomials and matrix polynomials.

1.3 A Brief Literature Review

The literature on polynomial rootfinding is vast, and we do not aim to be exhaustive. Much of literature on solving polynomial equations can be found in the electronic bibliographies authored by McNamee [18, 19]. Also the book [20] by the same author contains an extensive overview of methods for solving univariate polynomials. Various matrix methods for solving univariate polynomial equations have been proposed. Much of the framework for working in the Lagrange basis has been described in [1, 6]. The necessary extensions to matrix polynomials can be found in [2, 7]. Both of these references provide a good grounding for what has been done so far in the Lagrange basis, as well as the book [8] which has many discussions about alternative bases, as well as significant discussion on backward error. Other matrix approaches have been proposed in the Lagrange basis, these may be found in [5] and the references therein. For the Chebyshev basis the appropriate results can be found in [14, 22]. For matrix methods for other orthogonal polynomial bases, many results may be found in [10].

For a good introduction to barycentric Lagrange interpolation, we point to the landmark paper by Berrut and Trefethen [4]. We also point out Trefethen's book on approximation theory [25] as a good introduction to polynomial interpolation. There have been a number of recent papers discussing the various properties of the barycentric Lagrange formulation. For a good overview, we suggest [3, 17, 27].

For matrix polynomials, a good introduction can be found in [13], and for

many backward stability results (in the Monomial basis), we refer to [16, 23].

Bibliography

[1] A. Amiraslani, R. M. Corless, L. Gonzalez-Vega, and A. Shakoori, *Polynomial algebra by values*, Tech. Rep. TR-04-01, Ontario Research Centre for Computer Algebra, http://www.orcca.on.ca/TechReports, January 2004.

[2] A. Amiraslani, R. M. Corless, and P. Lancaster, *Linearization of matrix polynomials expressed in polynomial bases*, IMA Journal of Numerical Analysis, 29 (2009), pp. 141–157.

[3] J.-P. Berrut, R. Baltensperger, and H. D. Mittelmann, *Recent developments in barycentric rational interpolation*, in Trends and applications in constructive approximation, Springer, 2005, pp. 27–51.

[4] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Review, 46 (2004), pp. 501–517.

[5] D. Bini, L. Gemignani, and V. Pan, *Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations*, Numerische Mathematik, 100 (2005), pp. 373–408.

[6] R. M. Corless, *Generalized companion matrices in the Lagrange basis*, in Proceedings EACA, L. Gonzalez-Vega and T. Recio, eds., June 2004, pp. 317–322.

[7] ——, *On a generalized companion matrix pencil for matrix polynomials expressed in the Lagrange basis*, in Symbolic-Numeric Computation, D. Wang and L. Zhi, eds., Trends in Mathematics, Birkhuser Basel, 2007, pp. 1–15.

[8] R. M. Corless and N. Fillion, *A graduate survey of numerical methods*, Forthcoming, 2014, (2014).

[9] R. M. Corless and S. M. Watt, *Bernstein bases are optimal, but, sometimes, Lagrange bases are better*, in Proceedings of SYNASC, Timisoara, MIRTON Press, September 2004, pp. 141–153.

[10] D. Day and L. Romero, *Roots of polynomials expressed in terms of orthogonal polynomials*, SIAM Journal on Numerical Analysis, 43 (2005), p. 1969.

[11] A. Edelman and H. Murakami, *Polynomial roots from companion matrix eigenvalues*, Mathematics of Computation, 64 (1995), pp. 763–776.

[12] R. T. Farouki and V. T. Rajan, *Algorithms for polynomials in Bernstein form*, Comput. Aided Geom. Des., 5 (1988), pp. 1–26.

[13] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix polynomials*, SIAM, 2009.

[14] I. J. Good, *The Colleague matrix, a Chebyshev analogue of the companion matrix*, The Quarterly Journal of Mathematics, 12 (1961), pp. 61–68.

[15] T. Hermann, *On the stability of polynomial transformations between Taylor, Bernstein and Hermite forms*, Numerical Algorithms, 13 (1996), pp. 307–320.

[16] N. Higham, R. Li, and F. Tisseur, *Backward error of polynomial eigenproblems solved by linearization*, SIAM Journal on Matrix Analysis and Applications, 29 (2008), pp. 1218–1241.

[17] N. J. Higham, *The numerical stability of barycentric Lagrange interpolation*, IMA Journal of Numerical Analysis, 24 (2004), pp. 547–556.

[18] J. M. McNamee, *A bibliography on roots of polynomials*, Journal of Computational and Applied Mathematics, 47 (1993), pp. 391–394.

[19] ——, *A 2002 update of the supplementary bibliography on roots of polynomials*, J. Comput. Appl. Math., 142 (2002), pp. 433–434.

[20] J. M. McNamee, *Numerical methods for roots of polynomials. Part I*, Elsevier Science, 2007.

[21] C. Moler, *Cleves corner: Roots—of polynomials, that is*, The MathWorks Newsletter, 5 (1991), pp. 8–9.

[22] W. Specht, *Die Lage der Nullstellen eines Polynoms. III*, Mathematische Nachrichten, 16 (1957), pp. 369–389.

[23] F. Tisseur, *Backward error and condition of polynomial eigenvalue problems*, Linear Algebra and its Applications, 309 (2000), pp. 339–361.

[24] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Review, 43 (2001), pp. 235–286.

[25] L. N. Trefethen, *Approximation theory and approximation practice*, Society for Industrial and Applied Mathematics, 2012.

[26] L. N. Trefethen et al., *Chebfun Version 4.2*, The Chebfun Development Team, 2011. `http://www.maths.ox.ac.uk/chebfun/`.

[27] M. Webb, L. N. Trefethen, and P. Gonnet, *Stability of barycentric interpolation formulas for extrapolation*, SIAM Journal on Scientific Computing, 34 (2012), pp. A3009–A3015.

Chapter 2

Fast Reduction of Generalized Companion Matrix Pairs for Barycentric Lagrange Interpolants[1]

2.1 Introduction

For a polynomial $p(z)$ expressed in the monomial basis, it is well known that one can find the roots of $p(z)$ by computing the eigenvalues of a certain companion matrix constructed from its coefficients. For polynomials expressed in other bases (such as the Chebyshev basis, the Lagrange basis, or other orthogonal polynomial bases), generalizations of the companion matrix exist, constructed from the appropriate coefficients; see for example [4, 5, 10, 21].

In this article we consider polynomial interpolants in the Lagrange basis, expressed in barycentric form. Berrut and Trefethen [2] present a comprehensive review of such polynomial interpolants.

Given a set of $n + 1$ distinct interpolation nodes $\{x_0, \ldots, x_n\}$, with corre-

[1] A version of this chapter has been submitted to the SIAM Journal of Matrix Analysis and Application for publication.

sponding values $\{f_0, \ldots, f_n\}$, the barycentric weights w_j are defined by

$$w_j = \prod_{\substack{k=0 \\ k \neq j}}^{n} (x_j - x_k)^{-1}, \quad 0 \leq j \leq n \,. \tag{2.1}$$

The unique polynomial of degree less than or equal to n interpolating the data f_j at x_j is

$$p(z) = \prod_{i=0}^{n} (z - x_i) \sum_{j=0}^{n} \frac{w_j}{(z - x_j)} f_j \,. \tag{2.2}$$

Equation (2.2) is known as the "first form of the barycentric interpolation formula" [20] or the "modified Lagrange formula" [12]. The "second (true) form of the barycentric formula" [20] is

$$p(z) = \frac{\displaystyle\sum_{j=0}^{n} \frac{w_j}{(z - x_j)} f_j}{\displaystyle\sum_{j=0}^{n} \frac{w_j}{(z - x_j)}} \,, \tag{2.3}$$

which is constructed by dividing Equation (2.2) by the interpolant of the constant function 1 at the same nodes, and by cancelling the factor $\prod_{i=0}^{n} (z - x_i)$.

As was first shown in [4], the roots of the interpolating polynomial $p(z)$, as defined by (2.2), are exactly the generalized eigenvalues of the matrix pair

$$(\mathbf{A}, \mathbf{B}) = \left(\begin{bmatrix} 0 & -\mathbf{f}^T \\ \mathbf{w} & \mathbf{D} \end{bmatrix}, \begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix} \right) \,, \tag{2.4}$$

where $\mathbf{w} = \begin{bmatrix} w_0 & \cdots & w_n \end{bmatrix}^T$, $\mathbf{f}^T = \begin{bmatrix} f_0 & \cdots & f_n \end{bmatrix}$, and

$$\mathbf{D} = \begin{bmatrix} x_0 & & \\ & \ddots & \\ & & x_n \end{bmatrix} \,. \tag{2.5}$$

This can be shown by applying Schur's determinant formula: if $z \neq x_i$ for all

i, $0 \leq i \leq n$, then

$$\det(z\mathbf{B} - \mathbf{A}) = \det \begin{bmatrix} 0 & \mathbf{f}^T \\ -\mathbf{w} & z\mathbf{I} - \mathbf{D} \end{bmatrix} \tag{2.6}$$

$$= \det(z\mathbf{I} - \mathbf{D}) \det\left(0 + \mathbf{f}^T (z\mathbf{I} - \mathbf{D})^{-1} \mathbf{w}\right) \tag{2.7}$$

$$= \prod_{i=0}^{n} (z - x_i) \sum_{j=0}^{n} \frac{w_j f_j}{(z - x_j)} = p(z)\,, \tag{2.8}$$

and then by continuity at $z = x_i$ we have $\det(z\mathbf{B} - \mathbf{A}) = p(z)$. Thus, the eigenvalues of the matrix pair $(\mathbf{A}, \mathbf{B})$ are exactly the roots of $p(z)$. Furthermore, computing the roots of polynomial interpolants via the eigenvalues of the matrix pair $(\mathbf{A}, \mathbf{B})$ is numerically stable, as shown in Chapter 3 and in [14].

Remark 1. *While the degree of the polynomial interpolant $p(z)$ is less than or equal to n, the dimensions of the matrices $\mathbf{A}$ and $\mathbf{B}$ are $n + 2$ by $n + 2$. This formulation gives rise to two spurious infinite eigenvalues. We will show how to deflate these infinite eigenvalues in §2.3.3.*

The second form of the barycentric interpolation formula has a remarkable feature [1]: the interpolating property is satisfied independently of the choice of weights w_j, as long as they are all nonzero. Let us define some arbitrary nonzero weights u_j, we may write a rational interpolant as a quotient of polynomials interpolating the values $u_k f_k / w_k$ and u_k / w_k at the nodes x_k, where the w_k's are the weights given in (2.1). The rational function is then given by

$$r(z) = \frac{\displaystyle\prod_{i=0}^{n} (z - x_i) \sum_{j=0}^{n} \frac{w_j}{(z - x_j)} \left(\frac{u_j f_j}{w_j}\right)}{\displaystyle\prod_{i=0}^{n} (z - x_i) \sum_{j=0}^{n} \frac{w_j}{(z - x_j)} \left(\frac{u_j}{w_j}\right)}\,, \tag{2.9}$$

or

$$r(z) = \frac{\displaystyle\sum_{j=0}^{n} \frac{u_j}{(z - x_j)} f_j}{\displaystyle\sum_{j=0}^{n} \frac{u_j}{(z - x_j)}}, \tag{2.10}$$

which is exactly the second barycentric formula for the weights u_k. The choice of weights (2.1) forces the second form of the barycentric interpolation formula to be a polynomial, but for other choices of weights it is a rational function. For example, for interpolation nodes on the real line, if we let the barycentric weights be equal to $w_i = (-1)^i$ for all i, $0 \leq i \leq n$ (as suggested by Berrut [1]), we obtain a rational interpolant guaranteed to have no poles in $\mathbb{R}$. The eigenvalues of $(\mathbf{A}, \mathbf{B})$ give the roots of the numerator of the rational interpolant, and letting $f_j = 1$ for all j, $0 \leq j \leq n$, we may also compute the poles.

Through numerical experimentation we found that for real interpolation nodes, the initial reduction of $(\mathbf{A}, \mathbf{B})$ to Hessenberg-triangular form seemed always to reduce $\mathbf{A}$ to a symmetric tridiagonal plus rank-one matrix, and left $\mathbf{B}$ unchanged. We will now show that this is always the case.

Theorem 1. *For real interpolation nodes x_j, arbitrary barycentric weights w_j, and arbitrary values f_j, there exists a unitary matrix $\mathbf{Q}$ such that the matrix pair $(\mathbf{Q}^*\mathbf{A}\mathbf{Q}, \mathbf{Q}^*\mathbf{B}\mathbf{Q}) = (\mathbf{T} + \mathbf{e}_1\mathbf{c}^T, \mathbf{B})$ is in Hessenberg-triangular form, and $\mathbf{T}$ is a symmetric tridiagonal matrix.*

Proof. Theorem 3.3.1 of [29, p.138] states that there exists a unitary matrix $\mathbf{Q}$ whose first column is equal to $\mathbf{e}_1$ such that $\mathbf{H} = \mathbf{Q}^*\mathbf{A}\mathbf{Q}$ is upper Hessenberg. Partition $\mathbf{Q}$ as

$$\mathbf{Q} = \begin{bmatrix} 1 & \\ & \mathbf{Q}_1 \end{bmatrix}, \tag{2.11}$$

Explicitly, $\mathbf{H}$ is

$$\mathbf{H} = \mathbf{Q}^*\mathbf{A}\mathbf{Q} = \begin{bmatrix} 0 & -\mathbf{f}^T\mathbf{Q}_1 \\ \mathbf{Q}_1^*\mathbf{w} & \mathbf{Q}_1^*\mathbf{D}\mathbf{Q}_1 \end{bmatrix}. \tag{2.12}$$

The interpolation nodes $x_0, \ldots, x_n$ are all real. Thus, $\mathbf{T}_1 = \mathbf{Q}_1^*\mathbf{D}\mathbf{Q}_1$ is symmetric and upper Hessenberg, and therefore symmetric tridiagonal. Furthermore,

since $\mathbf{H}$ is upper Hessenberg then $\mathbf{Q}_1^*\mathbf{w} = t_0\mathbf{e}_1$. Let

$$\mathbf{T} = \begin{bmatrix} 0 & t_0\mathbf{e}_1^T \\ t_0\mathbf{e}_1 & \mathbf{T}_1 \end{bmatrix}, \tag{2.13}$$

and

$$\mathbf{c}^T = \begin{bmatrix} 0 & -t_0\mathbf{e}_1^T - \mathbf{f}^T\mathbf{Q}_1 \end{bmatrix}. \tag{2.14}$$

Then we can rewrite (2.12) as

$$\mathbf{Q}^*\mathbf{A}\mathbf{Q} = \mathbf{T} + \mathbf{e}_1\mathbf{c}^T. \tag{2.15}$$

Multiplying $\mathbf{B}$ on the left by $\mathbf{Q}^*$, and on the right by $\mathbf{Q}$, yields

$$\mathbf{Q}^*\mathbf{B}\mathbf{Q} = \begin{bmatrix} 1 & \\ & \mathbf{Q}_1^* \end{bmatrix}\begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix}\begin{bmatrix} 1 & \\ & \mathbf{Q}_1 \end{bmatrix} = \begin{bmatrix} 0 & \\ & \mathbf{Q}_1^*\mathbf{Q}_1 \end{bmatrix} = \mathbf{B}. \tag{2.16}$$

Thus, $(\mathbf{Q}^*\mathbf{A}\mathbf{Q}, \mathbf{Q}^*\mathbf{B}\mathbf{Q}) = (\mathbf{T} + \mathbf{e}_1\mathbf{c}^T, \mathbf{B})$ is in Hessenberg-triangular form. □

2.2 Fast Reduction to Hessenberg form

2.2.1 Lanczos Based Reduction

We have shown that the matrix pair $(\mathbf{A}, \mathbf{B})$ can be reduced to the matrix pair $(\mathbf{T} + \mathbf{e}_1\mathbf{c}^T, \mathbf{B})$ via unitary similarity transformations. However, the cost of the standard reduction algorithm using Givens rotations to reduce the matrix pair $(\mathbf{A}, \mathbf{B})$ to Hessenberg-triangular form is about $5n^3$ floating point operations [9]. This reduction also introduces nonzero entries, on the order of machine precision (and polynomial in the size of the matrix), to the upper triangular part of $\mathbf{B}$, which could in turn lead to errors being propagated later on in the QZ iterations.

We will now show how the reduction might be performed in $O(n^2)$ operations, by making use of the structure of the input matrix $\mathbf{A}$. We wish to

construct a unitary matrix $\mathbf{Q}$ of the form in Equation (2.11) such that

$$\mathbf{Q}^* \mathbf{A} \mathbf{Q} = \mathbf{T} + \mathbf{e}_1 \mathbf{c}^T \,. \tag{2.17}$$

To determine such a matrix $\mathbf{Q}$, partition $\mathbf{Q}_1$ as

$$\mathbf{Q}_1 = \begin{bmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \cdots & \mathbf{q}_n \end{bmatrix} \,. \tag{2.18}$$

The first column of (2.17) requires that $\mathbf{Q}_1^* \mathbf{w} = t_0 \mathbf{e}_1$, and hence we may immediately identify that

$$\mathbf{q}_0 = \frac{\mathbf{w}}{t_0} \,. \tag{2.19}$$

The matrix $\mathbf{Q}_1$ is unitary, so we require that $t_0 = \|\mathbf{w}\|_2$. Let

$$\mathbf{T}_1 = \begin{bmatrix} d_0 & t_1 & & & \\ t_1 & d_1 & \ddots & & \\ & \ddots & \ddots & t_{n-1} & \\ & & t_{n-1} & d_{n-1} & t_n \\ & & & t_n & d_n \end{bmatrix} \,, \tag{2.20}$$

then form $\mathbf{D}\mathbf{Q}_1 = \mathbf{Q}_1 \mathbf{T}_1$:

$$\begin{bmatrix} \mathbf{D}\mathbf{q}_0 & \mathbf{D}\mathbf{q}_1 & \cdots & \mathbf{D}\mathbf{q}_n \end{bmatrix} = \begin{bmatrix} d_0\mathbf{q}_0 + t_1\mathbf{q}_1 & t_1\mathbf{q}_0 + d_1\mathbf{q}_1 + t_2\mathbf{q}_2 & \cdots & t_n\mathbf{q}_{n-1} + d_n\mathbf{q}_n \end{bmatrix} \,, \tag{2.21}$$

the first column of which gives the equation

$$\mathbf{D}\mathbf{q}_0 = d_0\mathbf{q}_0 + t_1\mathbf{q}_1 \,. \tag{2.22}$$

Multiplying on the left by $\mathbf{q}_0^*$ and using the orthogonality of $\mathbf{q}_0$ and $\mathbf{q}_1$ identifies

$$d_0 = \mathbf{q}_0^* \mathbf{D} \mathbf{q}_0 \,. \tag{2.23}$$

The vector $\mathbf{q}_1$ is then given by

$$\mathbf{q}_1 = \frac{1}{t_1} \left(\mathbf{D} - d_0 \mathbf{I} \right) \mathbf{q}_0 \, , \tag{2.24}$$

and has unit length, which requires that $t_1 = \| \left(\mathbf{D} - d_0 \mathbf{I} \right) \mathbf{q}_0 \|_2$. The ith column of Equation (2.21) for $1 \leq i \leq n-1$ is

$$\mathbf{D}\mathbf{q}_i = t_i \mathbf{q}_{i-1} + d_i \mathbf{q}_i + t_{i+1} \mathbf{q}_{i+1} \, . \tag{2.25}$$

Multiplying on the left by $\mathbf{q}_i^*$ and using the orthogonality of the $\mathbf{q}_i$'s identifies

$$d_i = \mathbf{q}_i^* \mathbf{D} \mathbf{q}_i \, . \tag{2.26}$$

The vector $\mathbf{q}_{i+1}$ is given by

$$\mathbf{q}_{i+1} = \frac{1}{t_{i+1}} \left(\left(\mathbf{D} - d_i \mathbf{I} \right) \mathbf{q}_i - t_i \mathbf{q}_{i-1} \right) \, , \tag{2.27}$$

and has unit length, which requires that $t_{i+1} = \| \left(\mathbf{D} - d_i \mathbf{I} \right) \mathbf{q}_i - t_i \mathbf{q}_{i-1} \|_2$. The last column of (2.21) is

$$\mathbf{D}\mathbf{q}_n = t_n \mathbf{q}_n + d_n \mathbf{q}_n \, . \tag{2.28}$$

Multiplying on the left by $\mathbf{q}_n^*$ finally identifies

$$d_n = \mathbf{q}_n^* \mathbf{D} \mathbf{q}_n \, . \tag{2.29}$$

Algorithm 1 in the appendix shows the reduction explicitly. The total cost of the algorithm is approximately $9n^2$ floating point operations, which is a considerable reduction in cost compared with the standard Hessenberg-triangular reduction algorithm, or even compared to reducing $\mathbf{A}$ to Hessenberg form via elementary reflectors (the cost of which is still $O(n^3)$).

Remark 2. *Algorithm 1 is equivalent to the symmetric Lanczos process applied to the matrix* $\mathbf{D}$ *with starting vector* $\mathbf{w}$*. The reduction process generates an orthonormal basis* $\mathbf{q}_0, \cdots, \mathbf{q}_n$ *for the Krylov subspace* $\mathcal{K}_{n+1}(\mathbf{D}, \mathbf{w}) = \operatorname{span} \{ \mathbf{w}, \mathbf{D}\mathbf{w}, \cdots, \mathbf{D}^n \mathbf{w} \}$*. One of the potential difficulties which can arise*

when applying the symmetric Lanczos process [29, p. 372], and hence also when applying the reduction algorithm which we have described, is that in floating point arithmetic the orthogonality of the vectors $\mathbf{q}_i$ *is gradually lost. The remedy for this is to reorthogonalize the vector* $\mathbf{q}_{i+1}$ *against* $\mathbf{q}_0, \ldots, \mathbf{q}_i$ *at each step. This reorthogonalization increases the operation count to* $O(n^3)$ *which defeats the purpose of using this reduction algorithm in the first place.*

We will now prove some facts about the vectors $\mathbf{q}_0, \ldots, \mathbf{q}_n$ that are produced by Algorithm 1.

Lemma 1. *The set of vectors* $\{\mathbf{w}, \mathbf{D}\mathbf{w}, \ldots, \mathbf{D}^k\mathbf{w}\}$ *are linearly independent for all* k, $0 \le k \le n$, *as long as all of the nodes* x_i *are distinct and no* w_i *is zero.*

Proof. Form the matrix $\mathbf{V} = \begin{bmatrix} \mathbf{w} & \mathbf{D}\mathbf{w} & \cdots & \mathbf{D}^n\mathbf{w} \end{bmatrix}$, which can be written as

$$\mathbf{V} = \begin{bmatrix} w_0 & w_0 x_0 & \cdots & w_0 x_0^n \\ w_1 & w_1 x_1 & \cdots & w_1 x_1^n \\ \vdots & \vdots & \ddots & \vdots \\ w_n & w_n x_n & \cdots & w_n x_n^n \end{bmatrix} = \begin{bmatrix} w_0 & & \\ & \ddots & \\ & & w_n \end{bmatrix} \begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^n \end{bmatrix} . \tag{2.30}$$

The determinant of $\mathbf{V}$ is

$$\det \mathbf{V} = \prod_{i=0}^{n} w_i \prod_{0 \le j < k \le n} (x_k - x_j) , \tag{2.31}$$

which is nonzero as long as no w_i is equal to zero, and the x_i's are distinct. Thus, the set of vectors $\{\mathbf{w}, \mathbf{D}\mathbf{w}, \ldots, \mathbf{D}^n\mathbf{w}\}$ are linearly independent, and consequently the subsets $\{\mathbf{w}, \mathbf{D}\mathbf{w}, \ldots, \mathbf{D}^k\mathbf{w}\}$, for all k, $0 \le k \le n$, are all also linearly independent. □

Theorem 2. *Suppose* $\mathbf{w}$, $\mathbf{D}\mathbf{w}, \ldots, \mathbf{D}^n\mathbf{w}$ *are linearly independent, and the vectors* $\mathbf{q}_0, \cdots, \mathbf{q}_n$ *are generated by Algorithm 1. Then*

1. *The vectors* $\mathbf{q}_0, \cdots, \mathbf{q}_j$ *span the Krylov subspace* $\mathcal{K}_{j+1}(\mathbf{D}, \mathbf{w})$*, that is*

$$\operatorname{span}\{\mathbf{q}_0, \cdots, \mathbf{q}_j\} = \mathcal{K}_{j+1}(\mathbf{D}, \mathbf{w}) \quad 0 \le j \le n . \tag{2.32}$$

2. *The subdiagonal elements of* $\mathbf{T}$ *are all strictly positive, and hence* $\mathbf{T}$ *is properly upper Hessenberg.*

Proof. Lemma 1 showed that $\mathbf{w}, \mathbf{Dw}, \ldots, \mathbf{D}^n\mathbf{w}$ are linearly independent. The vectors $\mathbf{q}_0, \ldots, \mathbf{q}_n$ are generated by the symmetric Lanczos process, which is a special case of the Arnoldi process. The result follows from Theorem 6.3.9 of [28, p. 436]. □

2.2.2 Givens Rotation Based Reduction

Since the reduction process described in §2.2.1 has the potential for losing orthogonality of the transformation matrix $\mathbf{Q}_1$ [29, p. 373], we investigate other reduction algorithms that take advantage of the structure of $\mathbf{A}$.

The standard Hessenberg reduction routines in LAPACK and MATLAB (`_GEHRD` and `hess`, respectively) use a sequence of elementary reflectors to reduce the matrix. The first elementary reflector in this sequence annihilates the lower n entries of the first column of $\mathbf{A}$. This elementary reflector will also fill in most of the zero elements in the trailing submatrix, which must then be annihilated to reduce the matrix to Hessenberg form.

When we do annihilate elements in the first column of $\mathbf{A}$, the diagonal structure of the trailing submatrix will be disturbed. If we are to have any hope of lowering the operation count, then we will need to ensure that the trailing submatrix retains its symmetric tridiagonal structure.

To achieve this goal, it is clear that we should apply Givens rotations, annihilating entries of the first column of $\mathbf{A}$ one by one, and then returning the trailing submatrix to tridiagonal form.

Let $\mathbf{G}_k$ be a Givens rotation matrix and $\mathbf{A}_i = \mathbf{G}_i^* \cdots \mathbf{G}_1^* \mathbf{A} \mathbf{G}_1 \cdots \mathbf{G}_i$ be the matrix resulting from applying a sequence of i Givens rotations to the matrix $\mathbf{A}$. The first Givens rotation in the reduction $\mathbf{G}_1$ should act on the $(n+1, n+2)$

plane to annihilate the $(n+2,1)$ element of $\mathbf{A}$, yielding

$$\mathbf{A}_1 = \mathbf{G}_1^* \mathbf{A} \mathbf{G}_1 = \begin{bmatrix} 0 & f_0 & \cdots & f_{n-2} & \times & \times \\ w_0 & x_0 & & & & \\ \vdots & & \ddots & & & \\ w_{n-2} & & & x_{n-2} & & \\ \times & & & & \times & \times \\ 0 & & & & \times & \times \end{bmatrix}. \tag{2.33}$$

The $\times$ symbol specifies where elements of the matrix have been modified under this transformation. After this first Givens rotation, the matrix is still in the form we desire (the trailing 2 by 2 matrix is symmetric tridiagonal), so we push on. The next Givens rotation $\mathbf{G}_2$ will act on the $(n, n+1)$ plane to annihilate the $(n+1,1)$ element of $\mathbf{A}_1$, resulting in the matrix

$$\mathbf{A}_2 = \mathbf{G}_2^* \mathbf{G}_1^* \mathbf{A} \mathbf{G}_1 \mathbf{G}_2 = \begin{bmatrix} 0 & f_0 & \cdots & f_{n-3} & \times & \times & \times \\ w_0 & x_0 & & & & & \\ \vdots & & \ddots & & & & \\ w_{n-3} & & & x_{n-3} & & & \\ \times & & & & \times & \times & \times \\ 0 & & & & \times & \times & \times \\ 0 & & & & \times & \times & \times \end{bmatrix}. \tag{2.34}$$

Note again that the trailing 3×3 submatrix will be symmetric. Since we are aiming to reduce the matrix to a symmetric tridiagonal plus rank-one matrix, we should now apply a Givens rotation acting on the $(n+1, n+2)$ plane to eliminate the $(n+2, n)$ element of $\mathbf{A}_2$. This transformation does not disturb any of the zeros that we have just created in the first column. The resulting

matrix is

$$\mathbf{A}_3 = \begin{bmatrix} 0 & f_0 & \cdots & f_{n-3} & \times & \times & \times \\ w_0 & x_0 & & & & & \\ \vdots & & \ddots & & & & \\ w_{n-3} & & & x_{n-3} & & & \\ \times & & & & \times & \times & \\ 0 & & & & \times & \times & \times \\ 0 & & & & & \times & \times \end{bmatrix}. \tag{2.35}$$

We can now continue to reduce the first column of $\mathbf{A}_3$ by applying a Givens rotation $\mathbf{G}_4$, acting on the $(n-1, n)$ plane. This annihilates the $(n, 1)$ element of $\mathbf{A}_3$. The resulting matrix is now

$$\mathbf{A}_4 = \begin{bmatrix} 0 & f_0 & \cdots & f_{n-4} & \times & \times & \times & \times \\ w_0 & x_0 & & & & & & \\ \vdots & & \ddots & & & & & \\ w_{n-4} & & & x_{n-4} & & & & \\ \times & & & & \times & \times & \times & \\ 0 & & & & \times & \times & \times & \\ 0 & & & & \times & \times & \times & \times \\ 0 & & & & & & \times & \times \end{bmatrix}. \tag{2.36}$$

Applying another Givens rotation acting on the $(n, n+1)$ plane to annihilate the $(n+1, n-1)$ element yields

$$\mathbf{A}_5 = \begin{bmatrix} 0 & f_0 & \cdots & f_{n-4} & \times & \times & \times & \times \\ w_0 & x_0 & & & & & & \\ \vdots & & \ddots & & & & & \\ w_{n-4} & & & x_{n-4} & & & & \\ \times & & & & \times & \times & & \\ 0 & & & & \times & \times & \times & \times \\ 0 & & & & & \times & \times & \times \\ 0 & & & & & \times & \times & \times \end{bmatrix}. \tag{2.37}$$

We must now apply another Givens rotation acting on the $(n+1, n+2)$ plane in order to annihilate the $(n+2, n)$ element of $\mathbf{A}_5$, reducing the trailing submatrix to symmetric tridiagonal form:

$$\mathbf{A}_6 = \begin{bmatrix} 0 & f_0 & \cdots & f_{n-4} & \times & \times & \times & \times \\ w_0 & x_0 & & & & & & \\ \vdots & & \ddots & & & & & \\ w_{n-4} & & & x_{n-4} & & & & \\ \times & & & & \times & \times & & \\ 0 & & & & \times & \times & \times & \\ 0 & & & & & \times & \times & \times \\ 0 & & & & & & \times & \times \end{bmatrix}. \tag{2.38}$$

It should now be evident what will happen in the rest of the reduction: when we annihilate an element of the first column of $\mathbf{A}_i$, a bulge will be introduced into the top of the trailing submatrix. This bulge can then be chased out of the matrix without modifying any elements of the first column. We alternate between annihilating elements of the first column and chasing bulges out of the matrix until the matrix has been reduced to symmetric tridiagonal plus rank-one:

$$\mathbf{A}_{\frac{n(n+1)}{2}} = \begin{bmatrix} 0 & \times & \times & \times & \cdots & \times & \times \\ \times & \times & \times & & & & \\ & \times & \times & \times & & & \\ & & \times & \times & \ddots & & \\ & & & \ddots & \ddots & \times & \\ & & & & \times & \times & \times \\ & & & & & \times & \times \end{bmatrix}. \tag{2.39}$$

The cost of this reduction requires $n(n+1)/2$ Givens rotations to reduce the matrix to tridiagonal form, which ordinarily would lead to an $O(n^3)$ algorithm for the reduction. However, because of the structure of the trailing submatrix, we need only modify 9 elements of the matrix when annihilating an element of the first column, and 8 elements when chasing the bulge out of

the matrix. Hence, the total cost of the reduction is still $O(n^2)$, giving a considerable reduction in cost compared to the standard reduction via elementary Householder transformations.

Remark 3. *When performing the reduction algorithm described in this section, we need never actually form the full matrix. The whole algorithm can be implemented by modifying only* 4 *vectors:* $\mathbf{w}$, $\mathbf{f}$, *the diagonal elements* $\mathbf{d}$, *and the subdiagonal elements* $\mathbf{t}$. *This is shown in Algorithm 2 in the appendix.*

Lemma 2. *The reduction algorithm described in this section results in essentially the same matrix as Algorithm 1 proposed in §2.2.1. That is, there exists a unitary diagonal matrix* $\widehat{\mathbf{D}}$ *such that* $\mathbf{Q}_L = \mathbf{Q}_G\widehat{\mathbf{D}}$ *and* $\mathbf{H}_L = \widehat{\mathbf{D}}^{-1}\mathbf{H}_G\widehat{\mathbf{D}}$, *where* $\mathbf{Q}_L$ *and* $\mathbf{H}_L$ *are the unitary matrix and Hessenberg matrix resulting from Algorithm 1, and* $\mathbf{Q}_G$ *and* $\mathbf{H}_G$ *are the unitary matrix and Hessenberg matrix resulting from the Givens reduction described in this section.*

Proof. Theorem 5.7.24 of [28, p. 382] shows that such a $\widehat{\mathbf{D}}$ exists, as the first columns of $\mathbf{Q}_L$ and $\mathbf{Q}_G$ are both equal to $\mathbf{e}_1$. This same theorem also proves that $\mathbf{H}_G$ is also properly upper Hessenberg. □

2.3 Corollaries, Implications, and Discussion

2.3.1 Complex Nodes

The reduction processes proposed in §2.2.1 and §2.2.2 both require that the interpolation nodes x_k are real. In many applications, this restriction should not present a significant burden. However, since we would like these methods to be as general as possible, we will now discuss the changes that need to be made to the algorithms to extend them to complex interpolation nodes.

For complex interpolation nodes, we cannot find a unitary matrix $\mathbf{Q}_1$ and a symmetric tridiagonal matrix $\mathbf{T}_1$ such that $\mathbf{D}\mathbf{Q}_1 = \mathbf{Q}_1\mathbf{T}_1$ (we would require $\mathbf{D} = \mathbf{D}^*$). Thus, we will have to relax the conditions on the transformation matrix $\mathbf{Q}_1$ if we are to find a structured Hessenberg matrix in this case. If

we specify that $\mathbf{Q}$ be complex orthogonal (that is, $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$) instead of unitary, the matrix $\mathbf{A}$ can still be reduced to a tridiagonal plus rank-one matrix $\mathbf{Q}^T\mathbf{A}\mathbf{Q} = \mathbf{T} + \mathbf{e}_1\mathbf{c}^T$; however, now $\mathbf{T}$ is a complex symmetric tridiagonal matrix. This use of non-unitary similarity transformations preserves the spectrum. However, the condition number of the transformed matrix may become larger, and hence the accuracy of the computed eigenvalues may be worse. This is the price that we will have to pay in order to reduce $\mathbf{A}$ to a structured form.

In light of this, the reduction algorithm presented in §2.2.1 was equivalent to the symmetric Lanczos process. Thus, for complex interpolation nodes the reduction transforms to the complex symmetric Lanczos process; see, for example, [8].

To convert the reduction algorithm presented in §2.2.2 to work for complex interpolation nodes, we need to apply complex orthogonal Givens-like matrices [16] in place of the unitary Givens rotations to retain the complex symmetric structure of the trailing submatrix when annihilating elements of the matrix.

2.3.2 Zero Leading Coefficients in the Monomial Basis

Throughout this chapter, we have not specified that the leading coefficients (in the monomial basis) of the polynomial interpolant $p(z)$ are nonzero. If the exact degree of $p(z)$ is $n-m$, then the matrix pair $(\mathbf{A}, \mathbf{B})$ will have $m+2$ infinite eigenvalues in total. In this section, we will give some tools to determine if the leading coefficients are indeed zero (or if they are very small).

For notational convenience the term $[z^n](p(z))$ means the coefficient of z^n of the polynomial $p(z)$, as described in [11]. Thus, if $p(z)$ has degree n, then $[z^n](p(z))$ is the leading coefficient of $p(z)$ in the monomial basis. Throughout this thesis, unless otherwise specified, the term leading coefficient refers to the leading coefficient with respect to the monomial basis.

If the first m leading coefficients (in the monomial basis) of $p(z)$ are all zero, then we may reconstruct the same unique polynomial interpolant by

removing up to m of the interpolation nodes. Furthermore, when we remove an interpolation node x_k, the barycentric weights do not have to be recomputed: we may simply update the existing barycentric formula by multiplying each weight w_ℓ by $(x_\ell - x_k)$, and dividing the formula by $(z - x_k)$. If we remove a set of j interpolation nodes $\{x_k | k \in K_j\}$, where $K_j = \{k_1, \cdots, k_j\}$ is a set of unique integers, then we may restate the barycentric interpolation formula as

$$p(z) = \prod_{\substack{i=0 \\ i \notin K_j}}^{n} (z - x_i) \sum_{\substack{\ell=0 \\ \ell \notin K_j}}^{n} \left(\prod_{k \in K_j} (x_\ell - x_k) \frac{w_\ell f_\ell}{(z - x_\ell)} \right), \tag{2.40}$$

for all j, $0 \leq j \leq m-1$.

We will now state a theorem which gives a useful formula for the first nonzero leading coefficient in the monomial basis of the polynomial interpolant $p(z)$.

Theorem 3. *If the leading coefficients* $[z^{n-j}](p(z)) = 0$ *for all* j, $0 \leq j \leq m-1$, *then* $[z^{n-m}](p(z)) = \mathbf{f}^T \mathbf{D}^m \mathbf{w}$. *That is, if the first* m *leading coefficients (in the monomial basis) of* $p(z)$ *are all zero, then the* $(m+1)$*th leading coefficient is* $\mathbf{f}^T \mathbf{D}^m \mathbf{w}$.

Proof. We use induction on m. The barycentric formula (2.2) can be written as

$$p(z) = \sum_{\ell=0}^{n} \left(\prod_{\substack{k=0 \\ k \neq \ell}}^{n} (z - x_k) \right) w_\ell f_\ell \,, \tag{2.41}$$

whose leading coefficient is

$$[z^n](p(z)) = \sum_{\ell=0}^{n} w_\ell f_\ell = \mathbf{f}^T \mathbf{w} \,. \tag{2.42}$$

The next leading coefficient is given by

$$[z^{n-1}](p(z)) = -\sum_{\ell=0}^{n} w_\ell f_\ell \sum_{\substack{k=0 \\ k\neq\ell}}^{n} x_k \tag{2.43}$$

$$= -\sum_{\ell=0}^{n} w_\ell f_\ell \left(-x_\ell + \sum_{k=0}^{n} x_k \right) \tag{2.44}$$

$$= \sum_{\ell=0}^{n} w_\ell f_\ell x_\ell - \left(\sum_{j=0}^{n} w_j f_j \right) \sum_{k=0}^{n} x_k \tag{2.45}$$

$$= \mathbf{f}^T \mathbf{D}\mathbf{w} - \mathbf{f}^T \mathbf{w} \sum_{k=0}^{n} x_k \,. \tag{2.46}$$

Thus, if the leading coefficient $[z^n](p(z)) = 0 = \mathbf{f}^T\mathbf{w}$, then $[z^{n-1}](p(z)) = \mathbf{f}^T\mathbf{D}\mathbf{w}$, which will serve as our basis of induction.

Now assume that the theorem is true for all m with $1 \leq m \leq M-1$. Thus, if $[z^{n-j}](p(z)) = 0$ for all j, $0 \leq j \leq M-1$, then $[z^{n-M}](p(z)) = \mathbf{f}^T\mathbf{D}^M\mathbf{w}$. Now suppose that additionally $[z^{n-M}](p(z)) = 0$. The $(M+2)$nd leading coefficient of $p(z)$ can be obtained from (2.40):

$$\left[z^{n-(M+1)}\right](p(z)) = \sum_{\ell=0}^{n} q_{m+1}(x_\ell) w_\ell f_\ell \tag{2.47}$$

$$= \mathbf{f}^T q_{m+1}(\mathbf{D})\mathbf{w} \,, \tag{2.48}$$

where $q_j(z)$ is the monic polynomial

$$q_j(z) = \prod_{k\in K_j} (z - x_k) \,. \tag{2.49}$$

Expanding $q_{M+1}(\mathbf{D})$ in the monomial basis yields

$$q_{M+1}(\mathbf{D}) = \mathbf{D}^{M+1} + b_M\mathbf{D}^M + \cdots + b_0\mathbf{I} \,, \tag{2.50}$$

where all of the coefficients b_j are expressions in terms of x_k for $k \in K_{M+1}$. Substituting this expansion into (2.48) and expanding the resulting expression,

we obtain

$$\left[z^{n-(M+1)}\right](p(z)) = \mathbf{f}^T\mathbf{D}^{M+1}\mathbf{w} + b_M\mathbf{f}^T\mathbf{D}^M\mathbf{w} + \cdots + b_0\mathbf{f}^T\mathbf{w}\,. \tag{2.51}$$

From the induction hypothesis, all of the terms $\mathbf{f}^T\mathbf{D}^j\mathbf{w} = 0$ for all j, $0 \le j \le M$, and hence (2.51) reduces to

$$\left[z^{n-(M+1)}\right](p(z)) = \mathbf{f}^T\mathbf{D}^{M+1}\mathbf{w}\,. \tag{2.52}$$

Thus, we have proved that, if $[z^{n-j}](p(z)) = 0$ for all j, $0 \le j \le m-1$, then the first (and only the first) nonzero leading coefficient is $[z^{n-m}](p(z)) = \mathbf{f}^T\mathbf{D}^m\mathbf{w}$. □

We will now show how Theorem 3 can be applied to the reduction algorithm proposed in §2.2.1, so that we may determine some more information about the vector $\mathbf{c}_1$ produced from the reduction algorithm.

Corollary 1. *For the reduction process described in §2.2.1, if* $[z^{n-j}](p(z)) = 0$ *for all* j, $0 \le j \le m-1$, *then* $c_0 = -\|\mathbf{w}\|$ *and* $c_k = 0$ *for all* k, $1 \le k \le m-1$.

Proof. If $[z^{n-j}](p(z)) = 0$ for all j, $0 \le j \le m-1$, then Theorem 3 implies that

$$\mathbf{f}^T\mathbf{D}^j\mathbf{w} = 0 \tag{2.53}$$

for all j, $0 \le j \le m-1$. The first m columns of $\mathbf{c}_1^T$ are

$$\begin{bmatrix} c_0 & \cdots & c_{m-1} \end{bmatrix} = -\mathbf{f}^T \begin{bmatrix} \mathbf{q}_0 & \cdots & \mathbf{q}_{m-1} \end{bmatrix} - \|\mathbf{w}\|_2\mathbf{e}_1^T\,. \tag{2.54}$$

Theorem 2 established that the vectors $\mathbf{q}_0, \ldots, \mathbf{q}_{m-1}$ span the Krylov subspace $\mathcal{K}_m(\mathbf{D}, \mathbf{w}) = \text{span}\{\mathbf{w}, \mathbf{D}\mathbf{w}, \ldots, \mathbf{D}^{m-1}\mathbf{w}\}$, and the conditions (2.53) imply that the vector $\mathbf{f}$ is contained in the orthogonal complement $\mathcal{K}_m(\mathbf{D}, \mathbf{w})^\perp$. Thus, (2.54) reduces to

$$\begin{bmatrix} c_0 & c_1 & \cdots & c_{m-1} \end{bmatrix} = -\|\mathbf{w}\|\mathbf{e}_1^T\,, \tag{2.55}$$

and hence $c_0 = -\|\mathbf{w}\|$ and $c_k = 0$ for all k, $1 \le k \le m-1$. □

2.3.3 Deflation of Infinite Eigenvalues

The matrices in the pair $(\mathbf{A}, \mathbf{B})$ have dimension $(n+2) \times (n+2)$, whereas the degree of the polynomial $p(z)$ is only n; this formulation necessarily gives rise to two spurious infinite eigenvalues. If the characteristic polynomial of a matrix pair is not identically zero (indicating that the pair is singular, see, for example, [13]), infinite eigenvalues can be deflated from a matrix pair by transforming the pair to the form

$$\left(\widehat{\mathbf{A}}, \widehat{\mathbf{B}}\right) = \left(\begin{bmatrix} \mathbf{R}_\infty & \times \\ & \mathbf{H}_f \end{bmatrix}, \begin{bmatrix} \mathbf{J}_\infty & \times \\ & \mathbf{T}_f \end{bmatrix} \right) , \tag{2.56}$$

where $\mathbf{R}_\infty$ and $\mathbf{J}_\infty$ are both upper triangular. $\mathbf{R}_\infty$ is nonsingular, and $\mathbf{J}_\infty$ has only zero entries on the diagonal. $\mathbf{H}_f$ is upper Hessenberg, and $\mathbf{T}_f$ is upper triangular with nonzero diagonal entries. The matrix pair $(\widehat{\mathbf{A}}, \widehat{\mathbf{B}})$ is no longer properly upper Hessenberg-triangular, so we may split the eigenvalue problem into two parts: the infinite eigenvalues are the eigenvalues of the pair $(\mathbf{R}_\infty, \mathbf{J}_\infty)$, while the finite eigenvalues are the eigenvalues of the pair $(\mathbf{H}_f, \mathbf{T}_f)$. For the matrix pair $(\mathbf{A}, \mathbf{B})$, the dimensions of the matrices $\mathbf{R}_\infty$ and $\mathbf{J}_\infty$ will be $m+2$, where m is the number of zero leading coefficients of the interpolant $p(z)$. Reduction of the matrix pair to the form $(\widehat{\mathbf{A}}, \widehat{\mathbf{B}})$ is usually carried out by first reducing $(\mathbf{A}, \mathbf{B})$ to Hessenberg-triangular form, and then applying QZ iterations to force the subdiagonal elements to zero.

For the reduced matrix pair $(\mathbf{H}, \mathbf{B})$ obtained from either of the reduction algorithms proposed in §2.2.1 and §2.2.2, the reduction to the form $(\widehat{\mathbf{A}}, \widehat{\mathbf{B}})$ can be achieved by applying $m+2$ Givens rotations to the left of the matrix pair $(\mathbf{H}, \mathbf{B})$. Furthermore, the matrix $\mathbf{T}_f$ in (2.56) is a diagonal matrix. Thus, we may easily convert the generalized eigenvalue problem for the finite eigenvalues into a standard eigenvalue problem, provided that $\mathbf{T}_f$ is well conditioned.

Using either of the reduction algorithms described in §2.2.1 or §2.2.2, we can reduce the matrix pair $(\mathbf{A}, \mathbf{B})$ to the pair

$$(\mathbf{H}, \mathbf{B}) = (\mathbf{T} + \mathbf{e}_1 \mathbf{c}^T, \mathbf{B}) , \tag{2.57}$$

where $\mathbf{T}$ is a symmetric tridiagonal matrix. We then partition $\mathbf{H}$ as

$$\begin{bmatrix} 0 & t_0 + c_0 & \mathbf{c}_2^T \\ t_0 & d_0 & t_1\mathbf{e}_1 \\ \mathbf{0} & t_1\mathbf{e}_1 & \mathbf{T}_2 \end{bmatrix}, \tag{2.58}$$

where $\mathbf{T}_2$ is the trailing n by n submatrix of $\mathbf{T}$ and $\mathbf{c}_2^T$ is the vector of the last n elements of $\mathbf{c}^T$. To annihilate the $(2,1)$ entry of $\mathbf{H}$ we apply a permutation matrix $\mathbf{G}_1$ which swaps the first two rows. Applying $\mathbf{G}_1^*$ to the left of $\mathbf{H}$ and $\mathbf{B}$ yields the equivalent pair

$$(\mathbf{G}_1^*\mathbf{H}, \mathbf{G}_1^*\mathbf{B}) = \left(\begin{bmatrix} t_0 & d_0 & t_1\mathbf{e}_1 \\ 0 & t_0 + c_0 & \mathbf{c}_2^T \\ & t_1\mathbf{e}_1 & \mathbf{T}_2 \end{bmatrix}, \begin{bmatrix} 0 & 1 & \\ & 0 & \\ & & \mathbf{I} \end{bmatrix} \right). \tag{2.59}$$

Now that $\mathbf{G}_1^*\mathbf{H}$ is no longer properly upper Hessenberg and $\mathbf{G}_1^*\mathbf{B}$ is still upper triangular, we may deflate one of the infinite eigenvalues since the $(1,1)$ element of $\mathbf{G}_1^*\mathbf{B}$ is zero (indicating an infinite eigenvalue). Thus, we can delete the first row and column of each of these matrices, and operate on the matrix pair

$$(\mathbf{H}_1, \mathbf{B}_1) = \left(\begin{bmatrix} t_0 + c_0 & \mathbf{c}_2^T \\ t_1\mathbf{e}_1 & \mathbf{T}_2 \end{bmatrix}, \begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix} \right). \tag{2.60}$$

Remark 4. *If the first m leading coefficients of $p(z)$ are zero, Corollary 1 implies that $t_0 + c_0 = 0$ and $c_k = 0$ for all k, $1 \le k \le m-1$. Thus, we can apply a series of permutation matrices, swapping the first two rows and then deflating an infinite eigenvalue from the matrix pair by deleting the first row and column, until we obtain the matrix pair $(\mathbf{H}_m, \mathbf{B}_m)$, where*

$$\mathbf{H}_m = \begin{bmatrix} c_m & c_{m+1} & c_{m+2} & \cdots & c_n \\ t_{m+1} & d_{m+1} & t_{m+2} & & \\ & t_{m+2} & d_{m+2} & \ddots & \\ & & \ddots & \ddots & t_n \\ & & & t_n & d_n \end{bmatrix}, \quad \mathbf{B}_m = \begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix}. \tag{2.61}$$

Assuming that $t_0 + c_0 \neq 0$ (or $c_m \neq 0$ in light of Remark 4), we can annihilate the $(2,1)$ element of $\mathbf{H}_1$, by applying a unitary Givens rotation $\mathbf{G}_2$ that acts on the first two rows of $\mathbf{H}_1$, where

$$\mathbf{G}_2^* = \begin{bmatrix} \overline{h} & \overline{g} & \\ -g & h & \\ & & \mathbf{I} \end{bmatrix}, \tag{2.62}$$

and where h and g are suitably chosen to annihilate the $(2,1)$ element of $\mathbf{H}_1$. Applying $\mathbf{G}_2^*$ to the left of $\mathbf{H}_1$ yields

$$\mathbf{G}_2^*\mathbf{H}_1 = \begin{bmatrix} \overline{h}(t_0+c_0)+\overline{g}t_1 & \overline{h}c_1+\overline{g}d_1 & \overline{h}c_2+\overline{g}t_2 & \overline{h}c_3 & \cdots & \overline{h}c_n \\ 0 & -gc_1+hd_1 & -gc_2+ht_2 & -gc_3 & \cdots & -gc_n \\ & t_2 & d_2 & t_3 & & \\ & & t_3 & d_3 & \ddots & \\ & & & \ddots & \ddots & t_n \\ & & & & t_n & d_n \end{bmatrix}, \tag{2.63}$$

and applying $\mathbf{G}_2^*$ to the left of $\mathbf{B}_1$ yields

$$\mathbf{G}_2^*\mathbf{B}_1 = \begin{bmatrix} 0 & \overline{g} & \\ & h & \\ & & \mathbf{I} \end{bmatrix}. \tag{2.64}$$

Since $\mathbf{G}_2^*\mathbf{H}_1$ is no longer properly upper Hessenberg and $\mathbf{G}_2^*\mathbf{B}_1$ is upper triangular with the $(1,1)$ element being zero we may deflate the second spurious infinite eigenvalue by deleting the first row and column of these matrices. This

yields the matrix pair $(\mathbf{H}_2, \mathbf{B}_2)$, where

$$\mathbf{H}_2 = \begin{bmatrix} -gc_1 + hd_1 & -gc_2 + ht_2 & -gc_3 & \cdots & -gc_n \\ t_2 & d_2 & t_3 & & \\ & t_3 & d_3 & \ddots & \\ & & \ddots & \ddots & t_n \\ & & & t_n & d_n \end{bmatrix}, \tag{2.65}$$

and

$$\mathbf{B}_2 = \begin{bmatrix} h & \\ & \mathbf{I} \end{bmatrix}. \tag{2.66}$$

As long as h is nonzero (which it will be, since $t_0 + c_0 \neq 0$), we can convert this generalized eigenvalue problem into a standard eigenvalue problem by multiplying on the left by $\mathbf{B}_2^{-1}$. Furthermore, if we define $\beta = -g/h$, then the standard eigenvalue problem is

$$\mathbf{H}_s = \mathbf{T}_2 + \beta \mathbf{e}_1 \mathbf{c}_2^T . \tag{2.67}$$

Remark 5. *The question arises: what should we do if h is very small but nonzero? This would be the case if the leading coefficient of the interpolating polynomial is very close to zero. One possible answer would be to work directly with the generalized eigenvalue problem $(\mathbf{H}_2, \mathbf{B}_2)$ and regard the accuracy of the resulting (very large) eigenvalue as being dubious. Another option would be to monitor the size of $t_0 + c_0$ or c_j, and explicitly set these values to zero, resulting in the deflation of an infinite eigenvalue.*

Remark 6. *Here we are concerned primarily with the initial reduction of the matrix pair $(\mathbf{A}, \mathbf{B})$ to a standard eigenvalue problem with tridiagonal plus rank-one form. Fast algorithms do exist to perform the QR algorithm on such structured matrices, as shown in [3, 25, 26], and we believe that these methods should be very competitive once the matrix pair is reduced.*

2.3.4 Connection to the Chebyshev Colleague Matrix

For polynomials expressed by their values at Chebyshev points of the second kind ($x_i = \cos(j\pi/n)$, $0 \le j \le n$), one can solve the rootfinding problem by first converting the polynomial to a Chebyshev series. This can be done via the discrete cosine transform or the fast Fourier transform [22, 23]. The roots of the polynomial expressed as a finite Chebyshev series

$$p(z) = \sum_{k=0}^{n} a_k T_k(z) \tag{2.68}$$

can be found by computing the eigenvalues of the colleague matrix, discovered independently by Specht [21] and by Good [10]. The colleague matrix is a tridiagonal plus rank-one matrix $\mathbf{C} = \mathbf{T} + \mathbf{e}_1\mathbf{c}^T$, where the tridiagonal part $\mathbf{T}$ arises from the recurrence relation defining the Chebyshev polynomials. The coefficients of the Chebyshev expansion appear in the rank-one part of $\mathbf{C}$. There are many different forms in which the colleague matrix can be stated, and here we present one form with symmetric tridiagonal part:

$$\mathbf{C} = \begin{bmatrix} 0 & \frac{1}{2} & & & \\ \frac{1}{2} & \ddots & \ddots & & \\ & \ddots & 0 & \frac{1}{2} & \\ & & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ & & & \frac{1}{\sqrt{2}} & 0 \end{bmatrix} - \frac{1}{2a_n}\mathbf{e}_1 \begin{bmatrix} a_{n-1} & \cdots & a_1 & \sqrt{2}a_0 \end{bmatrix}. \tag{2.69}$$

The basic idea to show that the eigenvalues of the Colleague matrix (2.69)

are the roots of (2.68) is to multiply $\mathbf{C}$ on the left by the vector

$$\mathbf{v} = \begin{bmatrix} T_{n-1}(z) \\ \vdots \\ T_1(z) \\ \frac{1}{\sqrt{2}} T_0(z) \end{bmatrix} , \tag{2.70}$$

and utilize the three term recurrence relation for Chebyshev polynomials. The first row of $\mathbf{Cv} = z\mathbf{v}$ is satisfied exactly when $p(z) = 0$, and the others are satisfied by the three term recurrence relation.

In §2.2.1, §2.2.2, and §2.3.3, we were able to construct two nonsingular matrices $\mathbf{U}$ and $\mathbf{V}$ such that

$$(\mathbf{UAV}, \mathbf{UBV}) = \left(\begin{bmatrix} \mathbf{R}_\infty & \times \\ & \mathbf{H}_s \end{bmatrix}, \begin{bmatrix} \mathbf{J}_\infty & \times \\ & \mathbf{I} \end{bmatrix} \right) , \tag{2.71}$$

where the pair $(\mathbf{R}_\infty, \mathbf{J}_\infty)$ is upper triangular, and contains the $m + 2$ infinite eigenvalues of the pair $(\mathbf{A}, \mathbf{B})$. The eigenvalues of the upper Hessenberg matrix $\mathbf{H}_s$ are the finite eigenvalues of the pair $(\mathbf{A}, \mathbf{B})$, and hence are both the roots of $p(z)$ and the eigenvalues of $\mathbf{C}$. Thus, there exists two nonsingular matrices $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$ such that

$$\left(\widehat{\mathbf{U}}\mathbf{A}\widehat{\mathbf{V}}, \widehat{\mathbf{U}}\mathbf{B}\widehat{\mathbf{V}}\right) = \left(\begin{bmatrix} \mathbf{R}_\infty & \times \\ & \mathbf{C} \end{bmatrix}, \begin{bmatrix} \mathbf{J}_\infty & \times \\ & \mathbf{I} \end{bmatrix} \right) . \tag{2.72}$$

For Chebyshev points of the second kind, the barycentric weights w_ℓ are defined by [19, 30]

$$w_\ell = \frac{2^{n-1}}{n} (-1)^\ell \delta_\ell \, , \quad \delta_\ell = \begin{cases} 1/2 & \text{if } \ell = 0 \text{ or } \ell = n \\ 1 & \text{otherwise} \end{cases} . \tag{2.73}$$

Owing to their magnitude for large n, there is the risk of overflow in floating-point computations, so usually [2, 12, 30] the weights are multiplied by $n/2^{n-1}$

to give

$$\hat{w}_\ell = (-1)^\ell \delta_\ell \, . \tag{2.74}$$

This rescaling is possible because, for the second form of the barycentric interpolation formula (2.3), w_ℓ appears in both the numerator and denominator so this factor cancels. For the companion matrix pair $(\mathbf{A}, \mathbf{B})$, rescaling the barycentric weights w_ℓ does not change the eigenvalues. This is because the $(1,1)$ element of matrix $\mathbf{B}$ is zero, so we may rescale the first row or column of $\mathbf{A}$ independently.

We would ideally like to find a nonsingular diagonal matrix $\mathbf{S}$ such that when we reduce the matrix pair $(\mathbf{S}^{-1}\mathbf{A}\mathbf{S}, \mathbf{S}^{-1}\mathbf{B}\mathbf{S})$ to the form (2.71), and deflate the infinite eigenvalues, we have exactly $\mathbf{H}_s = \mathbf{C}$.

We begin again with the matrix pair $(\mathbf{A}, \mathbf{B})$, where

$$\mathbf{A} = \begin{bmatrix} 0 & -\mathbf{f}^T \\ \widehat{\mathbf{w}} & \mathbf{D} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix} , \tag{2.75}$$

and the vector of scaled barycentric weights is $\widehat{\mathbf{w}} = \begin{bmatrix} \hat{w}_0 & \cdots & \hat{w}_n \end{bmatrix}$.

We then define the matrix $\mathbf{S}_1 = \operatorname{diag}\left(\left[\sqrt{n}/\sqrt{2}, \sqrt{n}, \sqrt{n}, \cdots, \sqrt{n}, \sqrt{n}/\sqrt{2}\right]\right)$, for which $\|\mathbf{S}_1^{-1}\widehat{\mathbf{w}}\|_2 = 1$. Define

$$\mathbf{S} = \begin{bmatrix} 1 & \\ & \mathbf{S}_1 \end{bmatrix} \tag{2.76}$$

and form

$$\hat{\mathbf{A}} = \mathbf{S}^{-1}\mathbf{A}\mathbf{S} \, . \tag{2.77}$$

Note also that $\mathbf{S}^{-1}\mathbf{B}\mathbf{S} = \mathbf{B}$. Applying either of the reduction algorithms described in §2.2.1 and in §2.2.2 to $\hat{\mathbf{A}}$ produces a symmetric tridiagonal plus

rank-one matrix with tridiagonal part

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & & & & & \\ 1 & 0 & \frac{1}{\sqrt{2}} & & & & \\ & \frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & & & \\ & & \frac{1}{2} & \ddots & \ddots & & \\ & & & \ddots & 0 & \frac{1}{2} & \\ & & & & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ & & & & & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}. \tag{2.78}$$

After deflating the spurious infinite eigenvalues and reducing the generalized eigenvalue problem to a standard one, we obtain exactly the Chebyshev colleague matrix $\mathbf{H}_s = \mathbf{C}$.

2.3.5 Balancing

It is standard practice to balance a matrix before reducing it to Hessenberg form and computing its eigenvalues. For the standard eigenvalue problem, balancing is the default option in MATLAB's `eig` routine, and also in LAPACK's `_GEEV` routine. The balancing strategy used in these routines, as outlined in [17, 18], aims to improve the norm of the input matrix by applying diagonal similarity transformations, where the diagonal elements are restricted to exact powers of the radix employed.

For a generalized eigenvalue problem, similar algorithms exist to balance the matrix pair $(\mathbf{A}, \mathbf{B})$; full details of the algorithms can be found in [27] and in [15]. Both of these algorithms determine two diagonal matrices $\mathbf{D}_L$ and $\mathbf{D}_R$ such that the condition number of the eigenvalues of the equivalent matrix pair $(\mathbf{D}_L\mathbf{A}\mathbf{D}_R, \mathbf{D}_L\mathbf{B}\mathbf{D}_R)$ is improved. However, the algorithms described in §2.2.1 and §2.2.2 rely upon the fact that $\mathbf{Q}^*\mathbf{B}\mathbf{Q} = \mathbf{B}$. So we must have $\mathbf{D}_L\mathbf{B}\mathbf{D}_R = \mathbf{B}$, requiring that $\mathbf{D}_L = \mathbf{D}_R^{-1}$ except in the first diagonal entry,

which can be arbitrary. Hence, we may balance the matrix pair $(\mathbf{A}, \mathbf{B})$ by applying standard balancing to the matrix $\mathbf{A}$.

We are also in a unique situation in that the first row and column of $\mathbf{A}$ can be scaled independently without modifying $\mathbf{B}$, since the $(1,1)$ element is zero. For example, before finding a diagonal scaling transformation to balance the matrix $\mathbf{A}$, we could scale the first row and column to have unit norm; this will avoid difficulties where the first row and column have very different magnitudes.

2.4 Numerical Experiments

In this section, we will investigate the accuracy and stability of the two reduction algorithms proposed in §2.2.1 and §2.2.2, as well as that of the deflation procedure proposed in §2.3.3. Furthermore, we will investigate the application of Theorem 3 and Corollary 1 numerically.

We also compare the algorithms presented here to the algorithm described in [6]. That algorithm reduces a quasiseparable matrix to Hessenberg form in $O(n^2)$ operations.

2.4.1 Chebyshev Polynomials of the First Kind

We will first give an example for which the Hessenberg reduction of the matrix pair $(\mathbf{A}, \mathbf{B})$ is known. We interpolate Chebyshev polynomials of the first kind $T_n(z)$ at the roots of $T_{n+1}(z)$. At these nodes, the reduced matrix $\mathbf{Q}^*\mathbf{A}\mathbf{Q} = \mathbf{T} + \mathbf{e}_1\mathbf{c}^T$ has tridiagonal part $\mathbf{T}$ with entries $d_i = 0$, $0 \leq i \leq n$, and

$$t_i = \begin{cases} \|\mathbf{w}\|_2 & \text{if } i = 0 \\ 1/\sqrt{2} & \text{if } i = n \\ 1/2 & \text{otherwise} \end{cases} . \tag{2.79}$$

Furthermore, if we interpolate the Chebyshev polynomial $T_n(z)$ at these nodes, we will be able to symmetrize the matrix $\mathbf{A}$ by scaling the first column. The

Hessenberg reduction of $\mathbf{A}$ will then produce a symmetric tridiagonal matrix, and we will be able to directly test the accuracy of the reduction processes.

The interpolation nodes are taken to be Chebyshev nodes of the first kind, that is, the roots of $T_{n+1}(z)$, which are given by

$$x_j = \cos \frac{(2j+1)\pi}{2n+2}\,, \quad 0 \le j \le n\,. \tag{2.80}$$

At these nodes, $T_n(z)$ takes on the values

$$f_j = (-1)^j \sin \frac{(2j+1)\pi}{2n+2}\,, \quad 0 \le j \le n\,, \tag{2.81}$$

and the barycentric weights are

$$w_j = \frac{2^n}{(n+1)}(-1)^j \sin \frac{(2j+1)\pi}{2n+2}\,, \quad 0 \le j \le n\,. \tag{2.82}$$

Because $w_j = 2^n/(n+1)f_j$, we are able to symmetrize $\mathbf{A}$ by scaling the barycentric weights so that $\mathbf{w} = -\mathbf{f}$. Thus, the matrix pair

$$(\mathbf{A}, \mathbf{B}) = \left(\begin{bmatrix} 0 & -\mathbf{f}^T \\ -\mathbf{f} & \mathbf{D} \end{bmatrix}, \begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix} \right) \tag{2.83}$$

has eigenvalues which are exactly the roots of $T_n(z)$, and will reduce to a symmetric tridiagonal matrix. The characteristic polynomial of the scaled pair $(\mathbf{A}, \mathbf{B})$ is now

$$\det(z\mathbf{B} - \mathbf{A}) = -\frac{(n+1)}{2^n} T_n(z)\,. \tag{2.84}$$

For each n ranging from 1 to 100, we measured the accuracy of reducing the matrix pair to Hessenberg form by computing the maximum error in the computed subdiagonal entries and the exact subdiagonal entries given in (2.79). Figure 2.1 reports the maximum error for each value of n. We also measured the maximum forward error in the computed eigenvalues, as shown in Figure 2.2. Four reduction algorithms are compared: the standard Hessenberg-

triangular reduction, the Lanczos type reduction of §2.2.1, the Givens type reduction of §2.2.2, and the quasiseparable matrix algorithm proposed in [6].

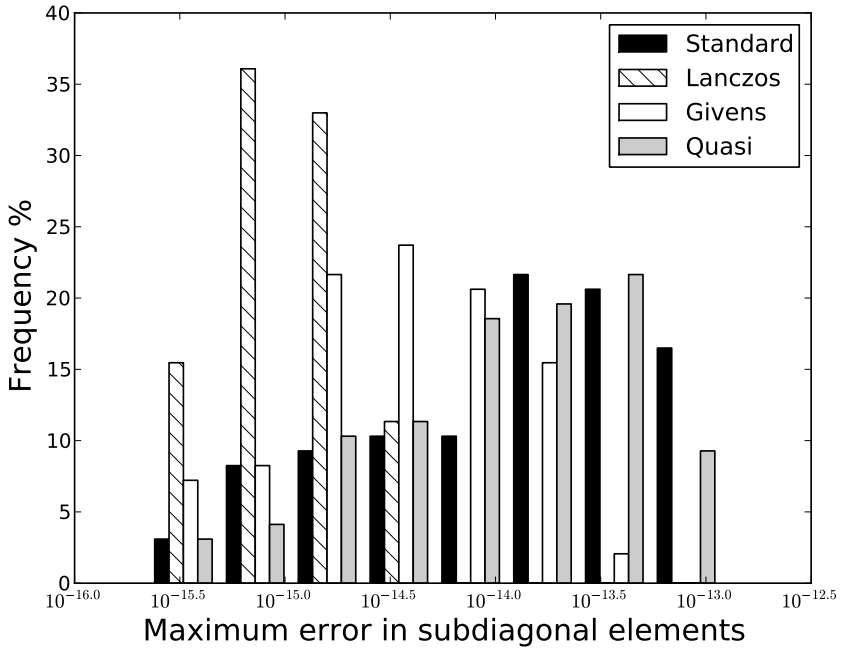


Figure 2.1: Distribution of maximum error in the subdiagonal entries of $\mathbf{T}$.

The standard reduction algorithm is not able to make use of the symmetry of $\mathbf{A}$. This leads to rounding errors propagating into the upper triangular part of $\mathbf{B}$. The error in the computed subdiagonal elements of the Hessenberg matrix and the maximum forward error are the worst of the four algorithms.

The Lanczos type reduction is the most accurate, which was somewhat surprising given that the transformation matrix could lose orthogonality. It seems that for this particular set of nodes, the method is fairly well behaved. Orthogonality of the transformation matrix is lost at a linear rate, that is $\|\mathbf{Q}^*\mathbf{Q}-\mathbf{I}\|_2 \propto n$. For $n = 100$ we have $\|\mathbf{Q}^*\mathbf{Q}-\mathbf{I}\|_2 \approx 10^{-14}$. We do not expect this to be the case for arbitrary sets of nodes. However, it is surprising that even for equispaced nodes, orthogonality appears to be lost at the same rate.

The Givens type reduction performs approximately halfway in between the

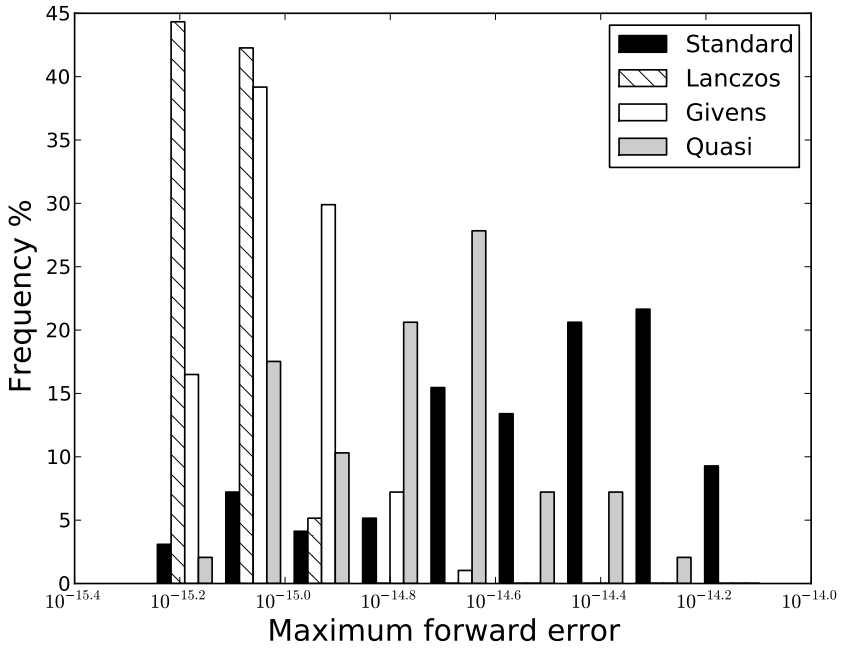


Figure 2.2: Maximum forward error in the computed roots from each of the four reduction processes.

standard reduction and the Lanczos type reduction. We also compared the Givens type reduction with LAPACK's `DSYTRD`, which uses elementary reflectors to decompose the matrix and is also able to take advantage of the symmetry of the matrix. The errors in the computed subdiagonal entries are comparable for both reduction methods. However, `DSYTRD` uses $O(n^3)$ operations to perform the reduction.

The quasiseparable matrix algorithm of [6] produced subdiagonal elements of around the same accuracy as the standard reduction. The computed eigenvalues were more accurate than the standard reduction, but were about half an order of magnitude larger, on average, than those of either the Givens type or Lanczos type reductions. We should note however, the algorithm can be applied to a much more general class of matrices than the one considered here.

2.4.2 Scaled Wilkinson Polynomial

Here we investigate the accuracy of computing the roots of a poorly conditioned polynomial, investigated by Wilkinson in [31]. We modify Wilkinson's polynomial to have roots equispaced in the interval $[0, 1]$: the polynomial we investigate is defined by

$$p(z) = \prod_{\ell=1}^{20} \left(z - \frac{\ell}{21} \right). \tag{2.85}$$

We sampled the polynomial at a range of interpolation nodes: equispaced nodes, Chebyshev nodes, and Legendre nodes. We choose 21 nodes of each distribution in the interval $[1/(2n), 1-1/(2n)]$. We applied diagonal balancing to the matrix $\mathbf{A}$, equalizing the row and column norms, and then scaled the first row and column to have unit norm. We reduced the matrix pair using the standard, Lanczos type, Givens type, and quasiseparable matrix reductions, and then deflated the spurious infinite eigenvalues to obtain the matrices $\mathbf{H}_S$, $\mathbf{H}_L$, $\mathbf{H}_G$, and $\mathbf{H}_Q$, respectively. Table 2.1 shows the maximum error between the true roots of $p(z)$ and the computed eigenvalues of $\mathbf{H}_S$, $\mathbf{H}_L$, $\mathbf{H}_G$, and $\mathbf{H}_Q$. We see that equispaced points are well suited for this particular problem, as they interlace the roots and hence the Hessenberg reduction will give a tridiagonal matrix which is symmetric except for the first two off-diagonal entries. For all three of the node distributions, both the Givens type and the quasiseparable matrix reductions are only a small factor more or less accurate than the standard reduction.

The accuracy of the Lanczos type reduction shows some degradation, and this is due to a slight loss of numerical orthogonality of the vectors $\mathbf{q}_0, \ldots, \mathbf{q}_n$ produced in the reduction process. This loss of orthogonality is due to the initial balancing performed on the matrix. If balancing is not used, the transformation matrix does not lose orthogonality. However, the computed eigenvalues are roughly one to two orders of magnitude less accurate than those computed from the balanced matrix. By computing $\|\mathbf{Q}_L^* \mathbf{Q}_L - \mathbf{I}\|_2$, as shown in Table 2.2, we see that the vectors produced by the reduction algorithm do

not give a numerically orthogonal transformation matrix $\mathbf{Q}_L$. Thus, we would not recommend the use of such an algorithm for arbitrary node distributions and weights.

Table 2.1: Maximum error in computed eigenvalues for Wilkinson's polynomial, sampled at different node distributions.

Point distribution	Standard	Lanczos	Givens	Quasi
Chebyshev	3.29×10^{-14}	2.37×10^{-12}	2.43×10^{-14}	6.25×10^{-14}
Equispaced	1.78×10^{-15}	5.65×10^{-13}	2.33×10^{-15}	1.33×10^{-15}
Legendre	1.67×10^{-14}	3.13×10^{-12}	1.05×10^{-14}	1.13×10^{-14}

Table 2.2: Measure of loss of orthogonality of vectors $\mathbf{q}_0, \ldots, \mathbf{q}_n$ produced from the Lanczos type reduction process of §2.2.1.

Point distribution	$\|\mathbf{Q}_L^* \mathbf{Q}_L - \mathbf{I}\|_2$
Chebyshev	3.37×10^{-12}
Equispaced	1.96×10^{-11}
Legendre	2.75×10^{-12}

Both the standard reduction algorithm and the Givens type reduction algorithm produce numerically orthogonal transformation matrices. This explains the increased accuracy of both of these methods over the Lanczos type reduction algorithm.

2.4.3 Polynomials with Zero Leading Coefficients in the Monomial Basis

We will now investigate the numerical application of Theorem 3 to detect when the leading coefficients (in the monomial basis) of the interpolant are zero. If

the first m leading coefficients are all equal to zero, then the values

$$\mathbf{f}^T\mathbf{D}^k\mathbf{w}\,, \quad 0 \le k \le m-1 \tag{2.86}$$

will all be equal to zero, and the first nonzero leading coefficient will be equal to $\mathbf{f}^T\mathbf{D}^m\mathbf{w}$. For these experiments, no scaling is used for either the values or the barycentric weights.

First we give a very simple example: we will interpolate the polynomial

$$f(z) = z^2 + 4z + 1 \tag{2.87}$$

at 7 Chebyshev points of the second kind. Chebyshev points are convenient for this example, but other points could be used, and produce similar results. Because the first 4 leading coefficients of the interpolant are zero, by Theorem 3 we may compute the first 5 leading coefficients using $\mathbf{f}^T\mathbf{D}^k\mathbf{w}$ for $0 \le k \le 4$, as shown in Table 2.3. The first four leading coefficients are all very close to machine epsilon ($\varepsilon_M \approx 2.2 \times 10^{-16}$), all being less than 32 times larger than machine epsilon. The first nonzero leading coefficient, which should be equal to 1, differs from 1 by less than 16 times machine epsilon. This example

Table 2.3: Leading coefficients of the degree six interpolant to (2.87).

$\mathbf{f}^T\mathbf{w}$	$\mathbf{f}^T\mathbf{D}\mathbf{w}$	$\mathbf{f}^T\mathbf{D}^2\mathbf{w}$	$\mathbf{f}^T\mathbf{D}^3\mathbf{w}$	$\mathbf{f}^T\mathbf{D}^4\mathbf{w}$
3.55×10^{-15}	3.55×10^{-15}	1.78×10^{-15}	7.11×10^{-15}	1.00

illustrates that for small problems, and polynomials with leading coefficients which are not close to zero, we can accurately determine the first nonzero leading coefficient.

However, in applications such as the MATLAB package CHEBFUN [24], polynomial interpolants are constructed to approximate functions accurate to machine precision. To monitor the accuracy of the interpolant, the Chebyshev expansion coefficients are computed from the values of the interpolant via the fast Fourier transform, and once the magnitude of these coefficients falls below a certain tolerance, the approximation is deemed to be accurate enough. In

this case, detecting the first nonzero leading coefficient of the interpolant using Theorem 3 may be very difficult, since the magnitude of the nonzero leading coefficient will necessarily be small.

We contrive the following example to illustrate just this point. Define the polynomial

$$f(z) = 10^{-12}T_9(z) + 10^{-10}T_8(z) + 10^{-8}T_7(z) + 10^{-6}T_6(z) + 10^{-4}T_5(z) + 10^{-2}T_4(z) + T_3(z) + 3T_2(z) - 2T_1(z) - T_0(z)\,, \quad (2.88)$$

where $T_k(z)$ is the k^{th} Chebyshev polynomial of the first kind. We then sampled this polynomial at 12 Chebyshev points of the second kind, and computed $\mathbf{f}^T\mathbf{D}^k\mathbf{w}$ for $0 \leq k \leq 2$ as shown in Table 2.4. The leading coefficients in the monomial basis which should be zero are not very accurate, although in this case the difference between the exact leading coefficient in the monomial basis 2.56×10^{-10} and the computed leading coefficient is approximately 10^{-13}. However, using this technique to determine the first nonzero leading coefficient may be unsuitable for applications where it is known that the first leading coefficient is very small. We should note here that although Theorem 3 may

Table 2.4: Leading coefficients of the degree 11 interpolant to (2.88).

$\mathbf{f}^T\mathbf{w}$	$\mathbf{f}^T\mathbf{D}\mathbf{w}$	$\mathbf{f}^T\mathbf{D}^2\mathbf{w}$
-4.83×10^{-13}	-4.83×10^{-13}	2.56×10^{-10}

not give numerically useful results, Corollary 1 may be better suited to numerically determining the first nonzero leading coefficient of an interpolant. The downside is that we then obtain an $O(n^2)$ algorithm, instead of an $O(kn)$ one from forming $\mathbf{f}^T\mathbf{D}^k\mathbf{w}$. For the previous example, we reduce the matrix $\mathbf{A}$ using the Givens type reduction algorithm, and look at the first three elements of the output vector $\mathbf{c}_1$. The first coefficient c_0 should be $-\|\mathbf{w}\|$, and agrees well with this value numerically. The second coefficient c_1 is indeed very small, as we would hope. The value of the first nonzero leading coefficient will now not be expressed in the monomial basis, or the Chebyshev basis, but rather in

Table 2.5: Elements of $\mathbf{c}_1$ produced by the Givens type reduction process.

$c_0 + \|\mathbf{w}\|$	c_1	c_2
-5.68×10^{-14}	1.55×10^{-15}	-2.46×10^{-12}

an orthogonal polynomial basis defined by the three term recurrence relation derived from the tridiagonal part $\mathbf{T}$ of the reduced matrix. However, one could determine that this is indeed a nonzero value.

What we have illustrated through these examples is that if the first nonzero leading coefficient of the interpolant is very much different from zero, then computing it through the formula $\mathbf{f}^T\mathbf{D}^m\mathbf{w}$ will be fairly effective. However, if the first nonzero leading coefficient is close to zero, then this formula may be inaccurate, and one should instead use the coefficients $\mathbf{c}_1$ produced by one of the reduction algorithms of §2.2.1 or §2.2.2 to determine the first nonzero leading coefficient.

2.4.4 Barycentric Rational Interpolation

An example of severe loss of orthogonality of the vectors $\mathbf{q}_0, \ldots, \mathbf{q}_n$ produced by the Lanczos type reduction of §2.2.1 occurs for barycentric rational interpolants described by Floater and Hormann [7]. In the simplest case, which was previously discovered by Berrut [1], for an arbitrary set of nodes one can prescribe the barycentric weights w_j to be

$$w_j = (-1)^j, \quad 0 \le j \le n\,. \tag{2.89}$$

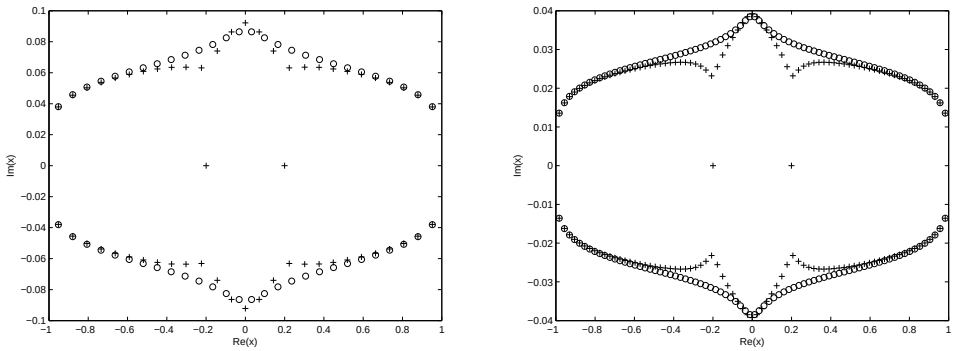
As discussed in §2.1, the second form of the barycentric formula (2.3) interpolates the values f_j, as long as the weights w_j are not zero. The choice of weights (2.89) guarantees that the rational interpolant does not have poles in $\mathbb{R}$. The second barycentric formula is the quotient of two polynomials expressed in the first form of the barycentric formula. Thus, we can compute the roots and poles of the rational interpolant by forming two matrix pairs: one for the roots, from the barycentric weights w_j and the values f_j, and one

for the poles, from the barycentric weights w_j and the values $f_j = 1$.

As an example, let us interpolate the function

$$f(z) = \frac{1}{(1 + 25z^2)} - \frac{1}{2} \tag{2.90}$$

at equispaced points on the interval $[-1, 1]$, using the weights given in (2.89). We construct the two matrix pairs corresponding to the numerator and denominator polynomials of the rational interpolant. We then reduce them to tridiagonal plus rank-one form using the Givens type reduction of §2.2.2, and deflate the two spurious infinite eigenvalues. The roots and poles of the rational interpolant are shown in Figure 2.3. Interestingly, at the extremities of the interval, the roots and poles of the interpolant become very close, indicating that there are common factors of the numerator and denominator polynomials. Table 2.6 shows the interpolation error and the error between the two real



(a) $n = 56$ (b) $n = 156$

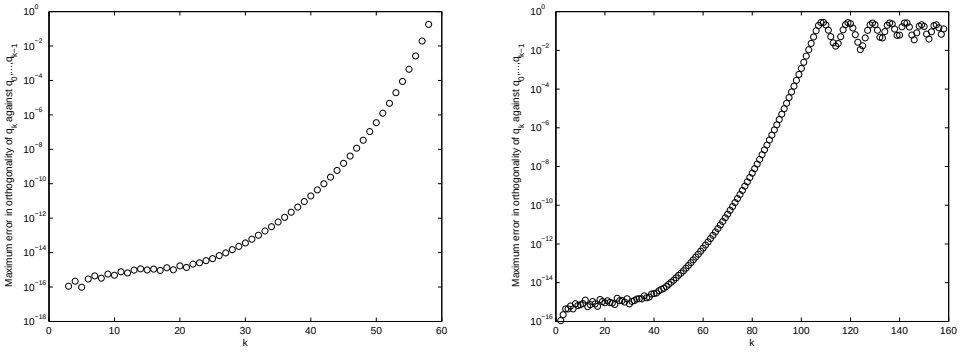
Figure 2.3: Roots (+) and poles (∘) of the rational interpolant.

roots of $f(z)$ and the root approximations generated from the eigenvalue computation for two choices of n. There is good agreement between the computed roots and the roots of the original function, especially considering the size of the interpolation error.

Next, we decompose the matrix pair $(\mathbf{A}, \mathbf{B})$ using the Lanczos type reduction of §2.2.1. Figure 2.4 illustrates the degradation in the orthogonality

Table 2.6: Accuracy of interpolant and eigenvalue computation.

n	$\max_{x\in[-1,1]} \lvert f(x) - p(x)\rvert$	Error in root approximation
56	1.05×10^{-3}	4.05×10^{-4}
156	3.86×10^{-4}	1.49×10^{-4}



(a) $n = 57$ (b) $n = 157$

Figure 2.4: Error in orthogonality of vectors $\mathbf{q}_0, \ldots, \mathbf{q}_n$: $\max_{0\le i\le k-1} |\mathbf{q}_i^* \mathbf{q}_k|$.

of the vector $\mathbf{q}_k$ against $\mathbf{q}_0, \ldots \mathbf{q}_{k-1}$ produced by the reduction process. For both values of n, there is a fairly rapid degradation of the orthogonality after only a small number of steps. By the time the algorithm has produced $\mathbf{q}_n$, all orthogonality of the vectors has been lost. The transformation matrix $\mathbf{Q}_L$ is formed using only the barycentric weights and the interpolation nodes. Thus, for this particular choice of nodes and weights, the Lanczos type algorithm is not suitable, and the Givens type reduction should be used instead.

2.5 Concluding Remarks

In this chapter, we have described two algorithms to reduce the matrix pair $(\mathbf{A}, \mathbf{B})$ to Hessenberg-triangular form, and to deflate at least two spurious infinite eigenvalues from the matrix pair so that it can be converted to a standard eigenvalue problem. The matrix pair is reduced in such a way that

the resulting standard eigenvalue problem has tridiagonal plus rank-one form. In addition to reducing the number of entries in the matrix being filled in, both reduction algorithms lower the cost of the Hessenberg-triangular reduction from $O(n^3)$ to $O(n^2)$. By numerical experimentation, we have shown that for particular choices of interpolation node distributions, the algorithms are accurate, despite the above-mentioned limitations of the Lanczos reduction algorithm.

Acknowledgement

The author thanks Luca Gemignani for providing an implementation of the quasiseparable matrix algorithm described in [6].

Bibliography

[1] J.-P. Berrut, *Rational functions for guaranteed and experimentally well-conditioned global interpolation*, Computers & Mathematics with Applications, 15 (1988), pp. 1–16.

[2] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Review, 46 (2004), pp. 501–517.

[3] D. Bini, L. Gemignani, and V. Pan, *Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations*, Numerische Mathematik, 100 (2005), pp. 373–408.

[4] R. M. Corless, *Generalized companion matrices in the Lagrange basis*, in Proceedings EACA, L. Gonzalez-Vega and T. Recio, eds., June 2004, pp. 317–322.

[5] D. Day and L. Romero, *Roots of polynomials expressed in terms of orthogonal polynomials*, SIAM Journal on Numerical Analysis, 43 (2005), p. 1969.

[6] Y. Eidelman, I. Gohberg, and L. Gemignani, *On the fast reduction of a quasiseparable matrix to Hessenberg and tridiagonal forms*, Linear Algebra and its Applications, 420 (2007), pp. 86–101.

[7] M. Floater and K. Hormann, *Barycentric rational interpolation with no poles and high rates of approximation*, Numerische Mathematik, 107 (2007), pp. 315–331.

[8] R. W. Freund, *Lanczos method for complex symmetric eigenproblems (Section 7.11)*, in Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., SIAM, Philadelphia, 2000, pp. 216–220.

[9] G. Golub and C. Van Loan, *Matrix computations*, Johns Hopkins Univ Pr, 3rd ed., 1996.

[10] I. J. Good, *The Colleague matrix, a Chebyshev analogue of the companion matrix*, The Quarterly Journal of Mathematics, 12 (1961), pp. 61–68.

[11] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd ed., 1994.

[12] N. J. Higham, *The numerical stability of barycentric Lagrange interpolation*, IMA Journal of Numerical Analysis, 24 (2004), pp. 547–556.

[13] B. Kågström, *Singular matrix pencils (Section 8.7)*, in Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., SIAM, Philadelphia, 2000, pp. 260–277.

[14] P. W. Lawrence and R. M. Corless, *Stability of rootfinding for barycentric Lagrange interpolants*. Submitted.

[15] D. Lemonnier and P. Van Dooren, *Balancing regular matrix pencils*, SIAM Journal on Matrix Analysis and Applications, 28 (2006), pp. 253–263.

[16] D. Mackey, N. Mackey, and F. Tisseur, *Structured tools for structured matrices*, Electron. J. Linear Algebra, 10 (2003), pp. 106–145.

[17] E. E. Osborne, *On pre-conditioning of matrices*, J. ACM, 7 (1960), pp. 338–345.

[18] B. Parlett and C. Reinsch, *Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numerische Mathematik, 13 (1969), pp. 293–304.

[19] M. Riesz, *Eine trigonometrische Interpolationsformel und einige Ungleichungen für Polynome*, Jahresbericht der Deutschen Mathematiker-Vereinigung, 23 (1914), pp. 354–368.

[20] H. Rutishauser, *Vorlesungen über Numerische Mathematik*, vol. 1, Birkhäuser, Basel, Stuttgart, 1976. English translation, *Lectures on Numerical Mathematics*, Walter Gautschi, ed., Birkhäuser, Boston, 1990.

[21] W. Specht, *Die Lage der Nullstellen eines Polynoms. III*, Mathematische Nachrichten, 16 (1957), pp. 369–389.

[22] L. N. Trefethen, *Spectral methods in MATLAB*, vol. 10, Society for Industrial and Applied Mathematics, 2000.

[23] L. N. Trefethen and Z. Battles, *An extension of MATLAB to continuous functions and operators*, SIAM J. Sci. Comput., 25 (2004), pp. 1743–1770.

[24] L. N. Trefethen et al., *Chebfun Version 4.2*, The Chebfun Development Team, 2011. `http://www.maths.ox.ac.uk/chebfun/`.

[25] M. Van Barel, R. Vandebril, P. Van Dooren, and K. Frederix, *Implicit double shift QR-algorithm for companion matrices*, Numerische Mathematik, 116 (2010), pp. 177–212.

[26] R. Vandebril, M. Van Barel, and N. Mastronardi, *A multiple shift QR-step for structured rank matrices*, Journal of Computational and Applied Mathematics, 233 (2010), pp. 1326–1344.

[27] R. C. Ward, *Balancing the generalized eigenvalue problem*, SIAM Journal on Scientific and Statistical Computing, 2 (1981), pp. 141–152.

[28] D. Watkins, *Fundamentals of matrix computations*, John Wiley and Sons, 2 ed., 2002.

[29] ——, *The matrix eigenvalue problem: GR and Krylov subspace methods*, SIAM, 2007.

[30] M. Webb, L. N. Trefethen, and P. Gonnet, *Stability of barycentric interpolation formulas for extrapolation*, SIAM Journal on Scientific Computing, 34 (2012), pp. A3009–A3015.

[31] J. Wilkinson, *The perfidious polynomial*, Studies in numerical analysis, 24 (1984), pp. 1–28.

Chapter 3

Stability of Rootfinding for Barycentric Lagrange Interpolants[1]

3.1 Introduction

This chapter establishes the numerical stability of rootfinding via the eigenvalues of a generalized companion matrix pair, for polynomials expressed in a Lagrange basis. This process is related to Lagrange interpolants expressed in barycentric form, as described by Berrut and Trefethen [5]. These representations are interesting in part because they are often so very well conditioned [8]. It has been shown in [1, 6, 7] that the roots of polynomials expressed in this basis can be found via the eigenvalues of a generalized companion matrix pair. We have previously suggested that computing the roots of interpolants in this manner is numerically stable [15]. However, apart from our own brief discussion in that paper, no in-depth analysis of the stability of this approach has been published to date, and hence we perform such an investigation here.

As seems to be the case for many different generalized companion matrices, the matrix pair discussed here came into use well before it was rigorously shown to be numerically stable. For the monomial basis, Edelman and Murakami [10] demonstrated the backward stability of the Frobenius com-

[1] A version of this chapter has been submitted to Numerical Algorithms for publication.

panion matrix, which forms the basis of MATLAB's `roots` command. This was experimentally and algorithmically discussed by Toh and Trefethen [21], and by Moler [18]. Similarly, for polynomials expressed as Chebyshev expansions [13, 20] and other orthogonal bases [9], use preceded analysis.

The interpolating polynomials we are investigating are defined as follows: given a set of $n+1$ distinct nodes $\{x_0, \cdots, x_n\}$, define the nodal polynomial $\ell(z)$ by

$$\ell(z) = \prod_{i=0}^{n} (z - x_i) \,, \tag{3.1}$$

and define the barycentric weights w_j by

$$w_j = \prod_{k \neq j} (x_j - x_k)^{-1} \,, \quad 0 \leq j \leq n \,. \tag{3.2}$$

The unique polynomial $p(z)$ of degree less than or equal to n interpolating a set of values $\{f_0, \cdots, f_n\}$ at the nodes $\{x_0, \ldots, x_n\}$ is given by the first form of the barycentric interpolation formula [5]

$$p(z) = \ell(z) \sum_{j=0}^{n} \frac{w_j f_j}{(z - x_j)} \,. \tag{3.3}$$

A generalized companion matrix pair for this interpolant can be written as

$$(\mathbf{A}, \mathbf{B}) = \left(\begin{bmatrix} 0 & -\mathbf{f}^T \\ \mathbf{w} & \mathbf{D} \end{bmatrix}, \begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix} \right) \,, \tag{3.4}$$

where $\mathbf{w}^T = \begin{bmatrix} w_0 & \cdots & w_n \end{bmatrix}$, $\mathbf{f}^T = \begin{bmatrix} f_0 & \cdots & f_n \end{bmatrix}$, and

$$\mathbf{D} = \begin{bmatrix} x_0 & & \\ & \ddots & \\ & & x_n \end{bmatrix} \,. \tag{3.5}$$

It was shown in [7], using the Schur complement, that (independent of the

ordering of the nodes)

$$\det(z\mathbf{B}-\mathbf{A}) = \det(z\mathbf{I}-\mathbf{D})(\mathbf{f}^T(z\mathbf{I}-\mathbf{D})^{-1}\mathbf{w}) = p(z)\,. \tag{3.6}$$

Thus, the generalized eigenvalues of $(\mathbf{A},\mathbf{B})$ are exactly the roots of the polynomial interpolant $p(z)$. We may also write the determinant as

$$\det(z\mathbf{B}-\mathbf{A}) = \mathbf{f}^T\mathrm{adj}(z\mathbf{I}-\mathbf{D})\mathbf{w}\,. \tag{3.7}$$

Notice that the Lagrange basis polynomials are $\ell_k(z) = \ell(z)w_k/(z-x_k)$, and hence (3.7) is mathematically equal to the usual Lagrange form of the interpolating polynomial.

3.2 Numerical Stability of $(\mathbf{A},\mathbf{B})$

In [10, 14, 16] we find significant discussion of the backward error of companion matrix pairs and other linearizations. These works consider only the monomial basis, and it turns out to be worthwhile to reformulate the analysis for the Lagrange basis.

The deepest part of their works is an explanation of why generic matrix perturbations of $\mathbf{A}$ (and $\mathbf{B}$) are equivalent (up to first order) to a much more restricted class of perturbations of the n $(+1)$ values of the polynomial coefficients. The reason, as deduced by Arnol'd [3] and refined by others, is that perturbations tangent to the set $\{\mathbf{TCT}^{-1} : \det\mathbf{T} \neq 0\}$ (the set of matrices similar to a companion matrix $\mathbf{C}$) do not matter, to first order. The other (normal) direction can be accounted for by small changes in the polynomial coefficients.

The same is true in this case (it is just a different basis) but the computations are remarkably simpler, and the bound we obtain is often smaller.

We first observe that the "coefficients" of the polynomial $p(z)$ in the Lagrange basis are in fact the "values" f_j at distinct nodes x_j. Thus, a good backward stability result would be something like:

The computed generalized eigenvalues of $(\mathbf{A},\mathbf{B})$ *are the exact roots of a*

polynomial $\widetilde{p}(z)$ *which satisfies*

$$\widetilde{p}(x_i) = f_i + \kappa_i \varepsilon + O(\varepsilon^2)\,, \tag{3.8}$$

where κ_i *is a moderate constant.* Now, the computed eigenvalues are the exact eigenvalues of a slightly perturbed pair $(\mathbf{A}+\Delta\mathbf{A}, \mathbf{B}+\Delta\mathbf{B})$, where the perturbations $\Delta\mathbf{A}$ and $\Delta\mathbf{B}$ satisfy $\|(\Delta\mathbf{A}, \Delta\mathbf{B})\|_F \leq \sigma(n)\varepsilon_M \|(\mathbf{A}, \mathbf{B})\|_F$, $\sigma(n)$ being a slowly growing function of n, and ε_M is the machine precision, this will give us a natural ε to take.

One complication is that the degree of the interpolating polynomial $p(z)$ is at most n, whereas the degree of the characteristic polynomial of a slightly perturbed pair $(\mathbf{A}+\Delta\mathbf{A}, \mathbf{B}+\Delta\mathbf{B})$ could be up to $n+2$. In exact arithmetic, we have $[z^{n+2}](\det(z\mathbf{B}-\mathbf{A})) = 0$, and $[z^{n+1}](\det(z\mathbf{B}-\mathbf{A})) = 0$. Hence, the formulation $(\mathbf{A}, \mathbf{B})$ introduces two spurious infinite eigenvalues to the problem. The perturbations $\Delta\mathbf{A}$ and $\Delta\mathbf{B}$ could, in principle, cause the two spurious infinite eigenvalues to become very large finite eigenvalues. However, this complication can be avoided if we ensure that the $(1,1)$ elements of the perturbations $\Delta\mathbf{A}$ and $\Delta\mathbf{B}$ are both equal to zero, as we show in the following lemma.

Lemma 3. *If the* $(1,1)$ *elements of the perturbation matrices* $\Delta\mathbf{A}$ *and* $\Delta\mathbf{B}$ *are both equal to zero, then the degree of* $\det(z(\mathbf{B}+\Delta\mathbf{B}) - (\mathbf{A}+\Delta\mathbf{A}))$ *is at most* n*.*

Proof. The known backward error result [2, §4.11.1.1] guarantees the existence of $\Delta\mathbf{A}$ and $\Delta\mathbf{B}$ such that all computed eigenvalues are the exact eigenvalues of $(\mathbf{A}+\Delta\mathbf{A}, \mathbf{B}+\Delta\mathbf{B})$. If it so happened that perturbations to the $(1,1)$ elements of the pair $\Delta\mathbf{A}_{11}$ and $\Delta\mathbf{B}_{11}$ were both identically zero, then via (3.7) we may write the determinant of $z(\mathbf{B}+\Delta\mathbf{B}) - (\mathbf{A}+\Delta\mathbf{A})$ as

$$\det\begin{bmatrix} 0 & (\mathbf{f}+\Delta\mathbf{f})^T \\ -(\mathbf{w}+\Delta\mathbf{w}) & z(\mathbf{I}+\Delta\mathbf{B}_{22}) - (\mathbf{D}+\Delta\mathbf{A}_{22}) \end{bmatrix} = \\ (\mathbf{f}+\Delta\mathbf{f})^T \mathrm{adj}(z(\mathbf{I}+\Delta\mathbf{B}_{22}) - (\mathbf{D}+\Delta\mathbf{A}_{22}))(\mathbf{w}+\Delta\mathbf{w})\,. \tag{3.9}$$

The degree in z of each minor of $z(\mathbf{I} + \Delta\mathbf{B}_{22}) - (\mathbf{D} + \Delta\mathbf{A}_{22})$ (each being determinants of $n \times n$ matrices) is at most n. Hence, the degree of $\det(z(\mathbf{B} + \Delta\mathbf{B}) - (\mathbf{A} + \Delta\mathbf{A}))$ is at most n.

□

As shown in §2.3.3, the two infinite eigenvalues can be deflated exactly from the matrix. In that section, only real interpolation nodes were considered, but the result is easily extended to arbitrary interpolation nodes, as we now show.

Lemma 4. *The two spurious infinite eigenvalues can be deflated exactly from the matrix pair* $(\mathbf{A}, \mathbf{B})$ *using unitary equivalence transformations.*

Proof. There exists a unitary matrix $\mathbf{Q}$ with $\mathbf{Q}\mathbf{e}_1 = \mathbf{e}_1$ such that $\mathbf{Q}^*\mathbf{A}\mathbf{Q} = \mathbf{H}$ is an upper Hessenberg matrix (see, for example, Theorem 3.3.1 in [24, p. 138]). Hence, $\mathbf{Q} = \mathrm{diag}(1, \mathbf{Q}_1)$, and $\mathbf{Q}^*\mathbf{B}\mathbf{Q} = \mathbf{B}$. The invariance of $\mathbf{B}$ under this similarity transformation ensures that the two spurious infinite eigenvalues can be deflated from the top left hand corner of the matrix pair. Applying this similarity transformation to $\mathbf{A}$ yields

$$\mathbf{Q}^*\mathbf{A}\mathbf{Q} = \begin{bmatrix} 0 & -\mathbf{f}^T\mathbf{Q}_1 \\ \mathbf{Q}_1^*\mathbf{w} & \mathbf{Q}_1^*\mathbf{D}\mathbf{Q}_1 \end{bmatrix} = \begin{bmatrix} 0 & -\mathbf{f}^T\mathbf{Q}_1 \\ h_{2,1}\mathbf{e}_1 & \widehat{\mathbf{H}}_1 \end{bmatrix}, \tag{3.10}$$

where $\widehat{\mathbf{H}}_1$ is upper Hessenberg.

The two spurious infinite eigenvalues can be deflated exactly from the matrix by applying two Givens rotations to the left of the pair $\mathbf{Q}^*(\mathbf{A}, \mathbf{B})\mathbf{Q}$. The first of which (say $\mathbf{G}_1$), swaps the first two rows of the pair. This introduces another zero on the diagonal of $\mathbf{B}$, which remains upper triangular. The matrix $\mathbf{G}_1\mathbf{Q}^*\mathbf{A}\mathbf{Q}$ is still upper Hessenberg, but is no longer properly upper Hessenberg. Thus, we may deflate the first spurious infinite eigenvalue by deleting the first row and column of these matrices, yielding the pair $(\mathbf{H}_1, \mathbf{B}_1)$.

The $(1,1)$ element of $\mathbf{H}_1$ is given by $\widehat{h}_{1,1} = -\mathbf{f}^T\mathbf{Q}_1\mathbf{e}_1$, and since $\mathbf{Q}_1^*\mathbf{w} = h_{2,1}\mathbf{e}_1$, the first column of $\mathbf{Q}_1$ is proportional to $\mathbf{w}$, so that $\widehat{h}_{1,1} = -\mathbf{f}^T\mathbf{w}/h_{2,1}$. From the barycentric formula (3.3), the leading coefficient of $p(z)$ is $\mathbf{f}^T\mathbf{w}$, and thus we have two cases to treat.

If $\mathbf{f}^T\mathbf{w}$ is zero, we can apply a Givens rotation, $\mathbf{G}_2$, swapping the first two rows of the pair $(\mathbf{H}_1, \mathbf{B}_1)$, and then deflate the second spurious infinite eigenvalue by deleting the first row and column of $\mathbf{G}_2(\mathbf{H}_1, \mathbf{B}_1)$. (In fact, we may continue this process until a nonzero $(1,1)$ element of the deflated pencil is encountered, which will deflate any other infinite eigenvalues that might be present.)

If $\mathbf{f}^T\mathbf{w} \neq 0$ (or the (k,k) element is nonzero), we can apply a Givens rotation, $\mathbf{G}_2$, to the left of the pair $(\mathbf{H}_1, \mathbf{B}_1)$, annihilating the $(2,1)$ element of $\mathbf{H}_1$, which will now no longer be properly upper Hessenberg. This rotation does not disturb the upper triangular structure of $\mathbf{B}_1$. Thus, we may deflate the second spurious infinite eigenvalue by deleting the first row and column of the pair $\mathbf{G}_2(\mathbf{H}_1, \mathbf{B}_1)$, yielding the pair $(\mathbf{H}_2, \mathbf{B}_2)$. □

Remark 7. *In other words, because the* $(1,1)$ *elements of* $\mathbf{A}$ *and* $\mathbf{B}$ *are both zero,* $\mathbf{G}_1$ *must swap the first two rows of the pair* $(\mathbf{A}, \mathbf{B})$*, introducing a second zero on the diagonal of* $\mathbf{B}$*. This ensures that the two spurious infinite eigenvalues are deflated exactly from the pair.*

Remark 8. *Once we have deflated the two spurious infinite eigenvalues, the generalized eigenvalue problem* $(\mathbf{H}_2, \mathbf{B}_2)$ *may be converted to a standard eigenvalue problem (provided that* $\mathbf{B}_2$ *is well-conditioned) by multiplying on the left by* $\mathbf{B}_2^{-1}$*.*

Remark 9. *If the first* m *leading coefficients of* $p(z)$ *are all zero, then a sequence of Givens rotations (swapping the first two rows of the pencil and deleting the first row and column) will deflate the* m *additional infinite eigenvalues, as shown in §2.3.3.*

The previous two lemmas constrain the structure of the perturbations $\Delta\mathbf{A}$ and $\Delta\mathbf{B}$. If we convert the generalized eigenvalue problem to a standard one, we might as well eliminate the perturbation $\Delta\mathbf{B}$ altogether. However, if we keep the perturbations $\Delta\mathbf{B}$, then our results will also apply to algorithms which allow perturbations in $\mathbf{B}$, and hence we keep them for this work.

Lemma 5. *Suppose that the computed generalized eigenvalues of* $(\mathbf{A}, \mathbf{B})$*, the set* $\{\lambda_1, \ldots, \lambda_n, \infty, \infty\}$*, are the exact eigenvalues of the matrix pair*

$$(\mathbf{A} + \Delta\mathbf{A}, \mathbf{B} + \Delta\mathbf{B}) \ , \tag{3.11}$$

where the perturbations satisfy $\|(\Delta\mathbf{A}, \Delta\mathbf{B})\|_F \leq \sigma(n)\varepsilon_M \|(\mathbf{A}, \mathbf{B})\|_F$ *(as guaranteed by* LAPACK *[2, §4.11.1.1]), and* ε_M *is the machine precision. Take* $\varepsilon = \sigma(n)\varepsilon_M \|(\mathbf{A}, \mathbf{B})\|_F$*, and define* $(\Delta\mathbf{A}, \Delta\mathbf{B}) = \varepsilon(\mathbf{E}, \mathbf{F})$*, where* $\|(\mathbf{E}, \mathbf{F})\|_F \leq 1$*. The computed generalized eigenvalues* λ_j *are the exact roots of a polynomial* $\widetilde{p}(z)$ *of degree at most* n*, satisfying*

$$\widetilde{p}(x_i) = f_i + \kappa_i \varepsilon + O(\varepsilon^2) \ , \tag{3.12}$$

where $\kappa_i = \operatorname{tr}\left(\operatorname{adj}(x_i\mathbf{B} - \mathbf{A})(x_i\mathbf{F} - \mathbf{E})\right)$.

Proof. The algorithms for the computing the generalized eigenvalues of a nonsymmetric matrix pair are normwise backward stable [2, §4.11.1.1]. The computed eigenvalues are the exact eigenvalues of a perturbed matrix pair $(\mathbf{A} + \Delta\mathbf{A}, \mathbf{B} + \Delta\mathbf{B})$, where the perturbations satisfy

$$\|(\Delta\mathbf{A}, \Delta\mathbf{B})\|_F \leq \sigma(n)\varepsilon_M \|(\mathbf{A}, \mathbf{B})\|_F \ , \tag{3.13}$$

where ε_M is the machine precision. Define the polynomial $\widetilde{p}(z)$ to be the characteristic polynomial of the perturbed pair $(\mathbf{A} + \Delta\mathbf{A}, \mathbf{B} + \Delta\mathbf{B})$, that is,

$$\widetilde{p}(z) = \det\left(z(\mathbf{B} + \Delta\mathbf{B}) - (\mathbf{A} + \Delta\mathbf{A})\right) . \tag{3.14}$$

Recall that for square matrices

$$\frac{d}{d\varepsilon} \det\left(A(\varepsilon)\right) = \operatorname{tr}\left(\operatorname{adj}(\mathbf{A}(\varepsilon)) \frac{d\mathbf{A}(\varepsilon)}{d\varepsilon}\right) \tag{3.15}$$

(see, for example, [12]).

Define $\varepsilon = \sigma(n)\varepsilon_M \|(\mathbf{A}, \mathbf{B})\|_F$ and $(\Delta\mathbf{A}, \Delta\mathbf{B}) = \varepsilon(\mathbf{E}, \mathbf{F})$. Then expand

$\widetilde{p}(x_i)$ about $\varepsilon = 0$:

$$\begin{aligned}\widetilde{p}(x_i) &= \det\left(x_i\mathbf{B} - \mathbf{A} + \varepsilon(x_i\mathbf{F} - \mathbf{E})\right) \\ &= \det\left(x_i\mathbf{B} - \mathbf{A}\right) + \mathrm{tr}(\mathrm{adj}(x_i\mathbf{B} - \mathbf{A} + 0(x_i\mathbf{F} - \mathbf{E}))(x_i\mathbf{F} - \mathbf{E})) \cdot \varepsilon + O(\varepsilon^2) \\ &= f_i + \mathrm{tr}\left(\mathrm{adj}(x_i\mathbf{B} - \mathbf{A})(x_i\mathbf{F} - \mathbf{E})\right) \cdot \varepsilon + O(\varepsilon^2)\,. \end{aligned} \tag{3.16}$$

Thus, $\kappa_i = \mathrm{tr}\left(\mathrm{adj}(x_i\mathbf{B} - \mathbf{A})(x_i\mathbf{F} - \mathbf{E})\right)$. □

The κ_i's in Lemma 5 are essentially the ratio of the backward error in the coefficients of the polynomial $p(z)$ to the backward error of the generalized eigenvalue problem. Since we may compute the backward error of the generalized eigenvalue problem easily, then obtaining a bound on the κ_i's allows us to compute useful information about the structured backward errors induced in the coefficients of the polynomial. The following theorem obtains such a useful bound.

Theorem 4. *The κ_i's in Lemma 5 satisfy*

$$|\kappa_i| \leq (|x_i| + 1)\|\mathrm{adj}(x_i\mathbf{B} - \mathbf{A})\|_F\,, \tag{3.17}$$

and

$$\|\mathrm{adj}(x_i\mathbf{B} - \mathbf{A})\|_F = \left|\prod_{\substack{j=0\\ j\neq i}}^{n}(x_i - x_j)\right| \sqrt{|f_i|^2 + |w_i|^2 + \sum_{\substack{j=0\\ j\neq i}}^{n}\frac{(|f_iw_i|^2 + |f_jw_i|^2 + |f_iw_j|^2)}{|x_i - x_j|^2} + \left|\sum_{\substack{j=0\\ j\neq i}}^{n}\frac{w_jf_j}{x_i - x_j}\right|^2}\,. \tag{3.18}$$

Proof. For all $\mathbf{X} \in \mathbb{C}^{n\times m}$, $\mathbf{Y} \in \mathbb{C}^{m\times n}$ we have $|\mathrm{tr}(\mathbf{XY})| \leq \|\mathbf{X}\|_F\|\mathbf{Y}\|_F$ (this is just the Cauchy-Schwartz inequality, see [24, p. 50]). Applying this to κ_i, we obtain

$$|\kappa_i| \leq (|x_i|\|\mathbf{F}\|_F + \|\mathbf{E}\|_F)\|\mathrm{adj}(x_i\mathbf{B} - \mathbf{A})\|_F\,, \tag{3.19}$$

and since $\|\mathbf{E}\|_F \leq 1$ and $\|\mathbf{F}\|_F \leq 1$, this reduces to

$$|\kappa_i| \leq (|x_i| + 1)\|\mathrm{adj}(x_i\mathbf{B} - \mathbf{A})\|_F \,. \tag{3.20}$$

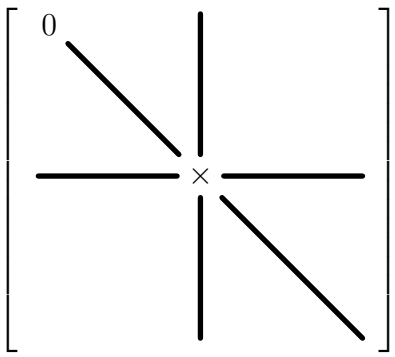
To derive the explicit bound (3.18) we must first determine $\mathrm{adj}(x_i\mathbf{B}-\mathbf{A})$. If x_i is not an eigenvalue of $(\mathbf{A}, \mathbf{B})$, this turns out to be remarkably simple:

$$\mathrm{adj}(x_i\mathbf{B}-\mathbf{A}) = \prod_{\substack{j=0 \\ j\neq i}}^{n} (x_i - x_j) \begin{bmatrix} 0 & -f_i\mathbf{e}_i^T \\ w_i\mathbf{e}_i & \Big(f_i\mathbf{I} - \mathbf{e}_i\mathbf{f}^T\Big)(x_i\mathbf{I} - \mathbf{D})^\dagger\Big(w_i\mathbf{I} - \mathbf{w}\mathbf{e}_i^T\Big) \end{bmatrix}, \tag{3.21}$$

where $(x_i\mathbf{I} - \mathbf{D})^\dagger$ is the Moore-Penrose pseudoinverse given by

$$(x_i\mathbf{I} - \mathbf{D})^\dagger = \begin{bmatrix} \frac{1}{(x_i - x_0)} & & & & & & \\ & \ddots & & & & & \\ & & \frac{1}{(x_i - x_{i-1})} & & & & \\ & & & 0 & & & \\ & & & & \frac{1}{(x_i - x_{i+1})} & & \\ & & & & & \ddots & \\ & & & & & & \frac{1}{(x_i - x_n)} \end{bmatrix}. \tag{3.22}$$

Notice that the trailing submatrix $\Big(f_i\mathbf{I} - \mathbf{e}_i\mathbf{f}^T\Big)(x_i\mathbf{I} - \mathbf{D})^\dagger\Big(w_i\mathbf{I} - \mathbf{w}\mathbf{e}_i^T\Big)$ is zero except for the diagonal entries and the i^{th} row and column, that is, $\mathrm{adj}(x_i\mathbf{B}-\mathbf{A})$ has the following structure:



. (3.23)

To show that (3.21) is indeed the adjugate of $(x_i\mathbf{B}-\mathbf{A})$, we follow [11]: assuming that $(x_i\mathbf{B}-\mathbf{A})$ is nonsingular,

$$\mathrm{adj}(x_i\mathbf{B}-\mathbf{A}) = \begin{bmatrix} \det(x_i\mathbf{I}-\mathbf{D}) & -\mathbf{f}^T\mathrm{adj}(x_i\mathbf{I}-\mathbf{D}) \\ \mathrm{adj}(x_i\mathbf{I}-\mathbf{D})\mathbf{w} & \mathrm{adj}(x_i\mathbf{I}-\mathbf{D}) - \mathrm{adj}(x_i\mathbf{I}-\mathbf{D}-\mathbf{w}\mathbf{f}^T) \end{bmatrix} . \tag{3.24}$$

The matrix $(x_i\mathbf{I}-\mathbf{D})$ is singular, the left and right null spaces are spanned by $\mathbf{e}_i$, and computing the (i,i) minor of $(x_i\mathbf{I}-\mathbf{D})$ yields

$$\mathrm{adj}(x_i\mathbf{I}-\mathbf{D}) = \prod_{\substack{j=0\\ j\neq i}}^{n}(x_i-x_j)\mathbf{e}_i\mathbf{e}_i^T . \tag{3.25}$$

It is shown in [11] that

$$\mathrm{adj}(x_i\mathbf{I}-\mathbf{D}) - \mathrm{adj}(x_i\mathbf{I}-\mathbf{D}-\mathbf{w}\mathbf{f}^T) = \prod_{\substack{j=0\\ j\neq i}}^{n}(x_i-x_j)\left(f_i\mathbf{I}-\mathbf{e}_i\mathbf{f}^T\right)(x_i\mathbf{I}-\mathbf{D})^{\dagger}\left(w_i\mathbf{I}-\mathbf{w}\mathbf{e}_i^T\right) , \tag{3.26}$$

and this completes the construction of the adjugate. Taking the Frobenius norm of (3.21), we obtain

$$\|\mathrm{adj}(x_i\mathbf{B}-\mathbf{A})\|_F^2 = \left|\prod_{\substack{j=0\\ j\neq i}}^{n}(x_i-x_j)\right|^2 \left(|f_i|^2+|w_i|^2+|f_iw_i|^2\|(x_i\mathbf{I}-\mathbf{D})^{\dagger}\|_F^2+|w_i|^2\|\mathbf{f}^T(x_i\mathbf{I}-\mathbf{D})^{\dagger}\|_F^2 + |f_i|^2\|(x_i\mathbf{I}-\mathbf{D})^{\dagger}\mathbf{w}\|_F^2+|\mathbf{f}^T(x_i\mathbf{I}-\mathbf{D})^{\dagger}\mathbf{w}|^2\right) . \tag{3.27}$$

Then, using the pseudoinverse (3.22), we arrive at

$$\|\mathrm{adj}(x_i\mathbf{B}-\mathbf{A})\|_F = \left|\prod_{\substack{j=0\\ j\neq i}}^{n}(x_i-x_j)\right| \sqrt{|f_i|^2+|w_i|^2+\sum_{\substack{j=0\\ j\neq i}}^{n}\frac{(|f_iw_i|^2+|f_jw_i|^2+|f_iw_j|^2)}{|x_i-x_j|^2}+\left|\sum_{\substack{j=0\\ j\neq i}}^{n}\frac{w_jf_j}{x_i-x_j}\right|^2}. \tag{3.28}$$

□

Notice that in the bound (3.18), the factor in front of the square root is exactly $1/|w_i|$. We choose not to simplify this term, because we will be able to use the same bound if we scale or balance the matrix. Taking this factor into account, we see that κ_i is proportional to $\max_j |f_j|$, to $\max_j |w_j|/|w_i|$, and to $\max_i 1/|w_i|$, and inversely proportional to $\min_{i\neq j}|x_i-x_j|$. Thus, we expect to encounter difficulties when the barycentric weights vary greatly. Equispaced points, for example, have barycentric weights which vary by exponentially large factors. Chebyshev points, on the other hand, have barycentric weights which differ only by a factor of two. However, they have an asymptotic density of $(1-x^2)^{-1/2}$, and thus for nodes near the boundary of the interval, the distance x_i-x_j can become $O(n^{-2})$; this is usually not significant.

Remark 10. *The κ_i are proportional to $\|\mathbf{f}\|$, but ε is proportional to $\|(\mathbf{A},\mathbf{B})\|_F$, which at first sight also varies as $\|\mathbf{f}\|$ and so it appears that $\kappa_i \propto \|\mathbf{f}\|^2$. This is not so, because we may scale $\mathbf{A}$ so that $\|(\mathbf{A},\mathbf{B})\|_F = O(\|\mathbf{D}\|_F)$. We will now discuss this scaling issue, together with the matter of balancing.*

Remark 11. *As pointed out by Berrut in [4], the second form of the barycentric formula can represent rational interpolants. The interpolation property holds as long as the barycentric weights are nonzero. We can compute the roots and poles of rational interpolants via the companion matrix pair for the numerator and denominator of the rational interpolant. Thus, we may also compute the backward error bound for the roots and poles.*

3.3 Scaling and Balancing $(\mathbf{A}, \mathbf{B})$

Before solving a standard or generalized eigenvalue problem, it is recommended that the input matrix be balanced first. The purpose of balancing is to improve the conditioning of the eigenvalues. For the standard eigenvalue problem, this is achieved by applying a diagonal similarity transformation to the matrix to bring it closer to a normal matrix. This balancing is discussed by Parlett and Reinsch [19] where they describe an iterative scheme to balance the matrix where the scaling factors are a power of the radix. For the generalized eigenvalue problem, two different strategies have been proposed to balance the matrix. The first, described by Ward [23], produces diagonal scaling matrices which bring the magnitude of each of the elements of the matrix pair is as close to unity as possible. The second, described by Lemonnier and Van Dooren [17], produces diagonal scaling matrices which aim to bring the matrix pair closer to a normal pair.

The pair $(\mathbf{A}, \mathbf{B})$ is, in a sense, close to a standard eigenvalue problem. Furthermore, $\mathbf{B}$ is already a normal matrix. To bring the pair closer to a normal matrix, we need only to balance $\mathbf{A}$ in the standard sense. Balancing $\mathbf{A}$ amounts to solving the optimization problem

$$\inf_{\mathbf{S}} \|\mathbf{S}^{-1}\mathbf{A}\mathbf{S}\|_F \,, \tag{3.29}$$

where $\mathbf{S} = \mathrm{diag}\begin{bmatrix} s_{-1} & s_0 & \cdots & s_n \end{bmatrix}$ is real diagonal matrix with positive entries. It is easy to see that the minimum is attained when $s_{-1} = 1$, and

$$s_j = \begin{cases} \sqrt{|w_j|/|f_j|} & \text{if } f_j \neq 0 \\ 1 & \text{otherwise} \end{cases}, \quad 0 \leq j \leq n \,. \tag{3.30}$$

The balanced matrix then satisfies

$$\|\mathbf{e}_i^T(\mathbf{S}^{-1}\mathbf{A}\mathbf{S})\|_2 = \|(\mathbf{S}^{-1}\mathbf{A}\mathbf{S})\mathbf{e}_i\|_2, \quad 1 \leq i \leq n+2 \,. \tag{3.31}$$

The $(1,1)$ elements of $\mathbf{A}$ and $\mathbf{B}$ are both zero. Hence, we may scale the

first row and the first column of $\mathbf{A}$ independently. This is equivalent to scaling the polynomial, that is, $p_\alpha(z) = p(z)/\alpha$, for which we have the choice of absorbing the scaling factor α into either $\mathbf{f}$, or $\mathbf{w}$, or both. This also holds for the balanced matrix, and thus we may also arbitrarily scale the first row and column of $\mathbf{S}^{-1}\mathbf{AS}$ independently. For example, we could scale the first row and column to have unit norm; this has the effect of reducing the norm $\|(\mathbf{A}, \mathbf{B})\|_F$ in the error bound even further.

3.4 Numerical Examples

We consider here a number of examples which illustrate the backward error bound described in Theorem 4. In that theorem, we defined $\varepsilon = \sigma(n)\varepsilon_M\|(\mathbf{A}, \mathbf{B})\|$, but did not specify anything about the function $\sigma(n)$. In the following experiments, we choose $\sigma(n) = \sqrt{n}$. This seems to be a good experimental fit, but we admit that we have no theoretical reason for choosing this.

In these experiments, we first balance $\mathbf{A}$ using the optimal diagonal balancing matrix $\mathbf{S}$ described in (3.30), that is, we form $\mathbf{S}^{-1}\mathbf{AS}$. We then scale the first row and column of $\mathbf{S}^{-1}\mathbf{AS}$ to have unit norm, that is, we apply two matrices $\mathbf{S}_\ell$ and $\mathbf{S}_r$ defined by

$$\mathbf{S}_\ell = \begin{bmatrix} s_\ell & \\ & \mathbf{I} \end{bmatrix}, \quad \mathbf{S}_r = \begin{bmatrix} s_r & \\ & \mathbf{I} \end{bmatrix}, \tag{3.32}$$

to the left and right of $\mathbf{S}^{-1}\mathbf{AS}$, yielding the balanced and scaled matrix

$$\widehat{\mathbf{A}} = \mathbf{S}_\ell^{-1}\mathbf{S}^{-1}\mathbf{AS}\mathbf{S}_r^{-1}\,. \tag{3.33}$$

The characteristic polynomial of $z\mathbf{B} - \widehat{\mathbf{A}}$ no longer takes on the value f_i at x_i. Instead, it takes on the value

$$\widetilde{p}(x_i) = \det\left(x_i\mathbf{B} - \widehat{\mathbf{A}}\right) = \frac{f_i}{s_\ell s_r}\,, \tag{3.34}$$

where s_ℓ and s_r are the factors used in (3.32) to scale the first row and column

of $\mathbf{S}^{-1}\mathbf{AS}$. Since we compute the eigenvalues of the pair $(\widehat{\mathbf{A}}, \mathbf{B})$, we need to multiply the error bound by $s_\ell s_r$, and furthermore, to make it a relative error bound, we divide by the norm of the original values $\mathbf{f}$. Thus, the relative backward error is

$$\frac{|s_\ell s_r \widetilde{p}(x_i) - f_i|}{\|\mathbf{f}\|_2} , \tag{3.35}$$

where

$$\widetilde{p}(z) = \prod_{i=1}^{n} (z - \lambda_i) , \tag{3.36}$$

is constructed from the computed eigenvalues $\{\lambda_i, \ldots, \lambda_n\}$. The relative backward error bound is now

$$\frac{s_\ell s_r |\widehat{\kappa}_i| \sqrt{n} \varepsilon_M \|(\widehat{\mathbf{A}}, \mathbf{B})\|_F}{\|\mathbf{f}\|_2} , \tag{3.37}$$

where the $\widehat{\kappa}_i$'s are computed from the entries of the balanced and scaled matrix $\widehat{\mathbf{A}}$ defined in (3.33).

3.4.1 Test problems from Edelman and Murakami

In this section we provide a comparison of our bound to the test problems given in [10] (who take the examples from [21]). We make a slight modification to those polynomials, so that we may interpolate them at the 21st roots of unity. This amounts to transforming the polynomials so that the roots are inside the unit circle (or nearby). The polynomials which we investigate are:

1. The scaled Wilkinson polynomial: $p(z) = \prod_{i=1}^{20} (z - i/21)$,
2. the monic polynomial with zeros equally spaced in the interval $[-2.1, 1.9]$,
3. $p(z) = \sum_{k=0}^{20} \frac{z^k}{k!}$,
4. the Bernoulli polynomial of degree 20, with its argument scaled by 3 so that the roots are within the unit circle, i.e. $B_{20}(3x)$,

5. $p(z) = \sum_{k=0}^{20} z^k$,

6. the monic polynomial with zeros 2^{-20}, 2^{-20},2^{-19},...,2^{-1},

7. the Chebyshev polynomial of degree 20,

8. the monic polynomial with zeros equally spaced on a sin curve, viz.,

   $(2\pi/19(k+0.5) + i\sin(2\pi/19(k+0.5)))/4$, $k = -10, -9, -8, \ldots, 9$.

For each of these polynomials, we compute the roots via the scaled and balanced pair $(\widehat{\mathbf{A}}, \mathbf{B})$ (using $\widehat{\mathbf{A}}$ from (3.33)), and compute the relative backward error (3.35), the relative error bound (3.37). Table 3.1 shows the results of these computations. The information is organized into three columns for each polynomial. The first column is the logarithm (base 10) of the observed relative backward error for each coefficient f_i. The second column is the logarithm (base 10) predicted relative backward error. The third column is the pessimism index [10], which is the logarithm (base 10) of the ratio of the observed relative backward error to the predicted relative backward error. The pessimism index gives us an indication of how many orders of magnitude larger the backward error is than the actual backward error. Thus, a pessimism index close to zero means that the bound describes the backward error very well.

For all of the test polynomials the observed backward error is close to machine precision. The bound on the error is approximately one to two orders of magnitude greater than the observed error. Furthermore, when the bound is more pessimistic, the observed backward error is much less than the machine precision.

Table 3.1: Relative backward error in coefficients (log base 10) for eight different degree-20 polynomials.

(1)			(2)			(3)			(4)			(5)			(6)			(7)			(8)		
−24	−19	−5	−17	−16	−2	−15	−13	−2	−18	−15	−2	−15	−13	−2	−16	−14	−2	−20	−17	−3	−16	−14	−1
−23	−18	−5	−16	−15	−1	−15	−13	−2	−17	−15	−2	−∞	−13	−∞	−15	−14	−2	−18	−15	−3	−16	−15	−1
−20	−17	−3	−15	−14	−1	−15	−13	−2	−17	−15	−2	−∞	−14	−∞	−15	−14	−2	−16	−14	−2	−16	−14	−2
−19	−16	−3	−14	−14	−0	−15	−13	−2	−16	−14	−2	−∞	−14	−∞	−15	−14	−2	−16	−14	−2	−15	−14	−1
−17	−15	−2	−14	−13	−0	−15	−13	−2	−15	−14	−2	−∞	−14	−∞	−15	−14	−2	−15	−13	−2	−15	−14	−2
−16	−15	−1	−13	−13	−0	−15	−13	−2	−15	−13	−1	−∞	−14	−∞	−15	−13	−2	−15	−13	−2	−15	−13	−1
−15	−14	−1	−14	−13	−0	−16	−14	−2	−15	−13	−1	−∞	−14	−∞	−15	−13	−2	−16	−13	−3	−14	−13	−1
−15	−14	−1	−14	−14	−0	−16	−14	−2	−15	−13	−2	−∞	−14	−∞	−15	−13	−2	−15	−13	−2	−14	−13	−1
−14	−13	−1	−15	−14	−1	−15	−14	−2	−15	−13	−2	−∞	−14	−∞	−15	−13	−1	−16	−14	−2	−15	−13	−1
−14	−13	−1	−16	−15	−1	−16	−14	−2	−15	−13	−2	−∞	−14	−∞	−15	−13	−2	−17	−15	−2	−15	−14	−1
−14	−13	−1	−17	−15	−2	−16	−14	−2	−15	−13	−2	−∞	−14	−∞	−15	−13	−2	−19	−16	−3	−16	−14	−1
−14	−13	−1	−17	−15	−2	−16	−14	−2	−15	−13	−2	−∞	−14	−∞	−15	−13	−2	−18	−16	−2	−16	−15	−1
−14	−13	−1	−16	−15	−1	−16	−14	−2	−15	−13	−2	−∞	−14	−∞	−15	−13	−2	−17	−15	−2	−16	−14	−2
−14	−13	−1	−15	−14	−1	−15	−14	−2	−15	−13	−2	−∞	−14	−∞	−15	−13	−2	−16	−14	−2	−16	−14	−2
−15	−14	−1	−14	−14	−0	−15	−14	−2	−15	−13	−2	−∞	−14	−∞	−15	−13	−2	−15	−13	−1	−15	−14	−2
−15	−14	−1	−14	−13	−0	−16	−14	−2	−15	−13	−2	−∞	−14	−∞	−16	−13	−3	−15	−13	−2	−15	−13	−1
−16	−15	−1	−13	−13	−0	−16	−13	−2	−15	−13	−1	−∞	−14	−∞	−16	−13	−2	−15	−13	−2	−15	−13	−1
−17	−15	−2	−14	−13	−0	−15	−13	−2	−15	−14	−2	−∞	−14	−∞	−15	−14	−2	−16	−13	−2	−15	−13	−2
−18	−16	−2	−14	−14	−0	−15	−13	−2	−16	−14	−1	−∞	−14	−∞	−16	−14	−2	−15	−14	−2	−15	−13	−2
−20	−17	−3	−15	−14	−1	−15	−13	−2	−17	−15	−2	−∞	−14	−∞	−15	−14	−2	−16	−14	−2	−16	−14	−2
−22	−18	−3	−16	−15	−1	−15	−13	−2	−17	−15	−2	−∞	−13	−∞	−16	−14	−2	−17	−15	−2	−16	−14	−2

For each of the polynomials, we also present the maximum relative backward error, and the associated bound, shown in Table 3.2. We see that the maximum error is only a small multiple of machine precision, and also that the bound is not very pessimistic (approximately an order of magnitude).

Table 3.2: Maximum observed backward error and bound.

Polynomial	Error	Bound	Pessimism index
1	1.99×10^{-14}	7.80×10^{-14}	-0.6
2	4.12×10^{-14}	6.06×10^{-14}	-0.2
3	9.96×10^{-16}	6.51×10^{-14}	-1.8
4	2.39×10^{-15}	4.50×10^{-14}	-1.3
5	6.97×10^{-16}	8.78×10^{-14}	-2.1
6	1.98×10^{-15}	6.08×10^{-14}	-1.5
7	1.74×10^{-15}	6.53×10^{-14}	-1.6
8	4.36×10^{-15}	6.91×10^{-14}	-1.2

We should note that if we interpolate the original polynomials given [10] on this set of nodes, the condition number for evaluation is very large. This

is the reason why the polynomials were modified so that the roots were closer to the interpolation nodes.

3.4.2 Chebyshev Polynomials of the First Kind

Our next experiment will test the bound for a particularly well-conditioned family of polynomials: Chebyshev polynomials of the first kind $T_n(x)$ interpolated at their extreme points. At the Chebyshev points of the first kind

$$x_j = \cos(j\pi/n), \quad 0 \le j \le n\,, \tag{3.38}$$

the Chebyshev polynomial of degree n, $T_n(x)$, takes on the values

$$f_j = T_n(x_j) = (-1)^j, \quad 0 \le j \le n\,. \tag{3.39}$$

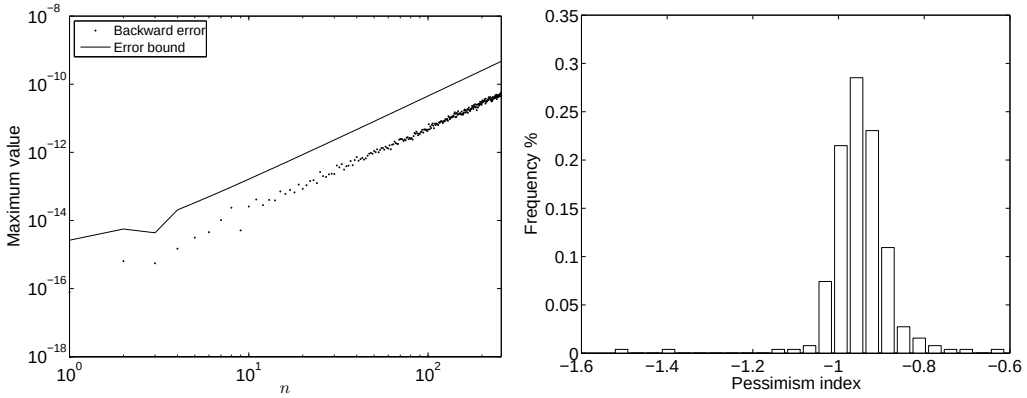
The barycentric weights are given by

$$w_j = \frac{2^{n-1}}{n}(-1)^j\,, \quad 1 \le j \le n-1 \tag{3.40}$$

and half of these values when $j = 0$ and $j = n$.

For each polynomial $T_n(x)$, we compute the roots via the balanced and scaled pair $(\widehat{\mathbf{A}}, \mathbf{B})$ (using $\widehat{\mathbf{A}}$ from (3.33)), then compute the relative backward error (3.35), and the bound (3.37) for n ranging from 1 to 256.

Figure 3.1 shows the maximum relative error and the associated error bound, as well as the associated pessimism index. The maximum relative error grows like $O(n^{2.5})$, and we see that the bound is only pessimistic by approximately one order of magnitude. The figure on the left shows the associated pessimism index for the maximum error bound. Agreement is excellent, the bound is typically one order of magnitude greater than the actual error.



(a) Maximum relative backward error (b) Associated pessimism index

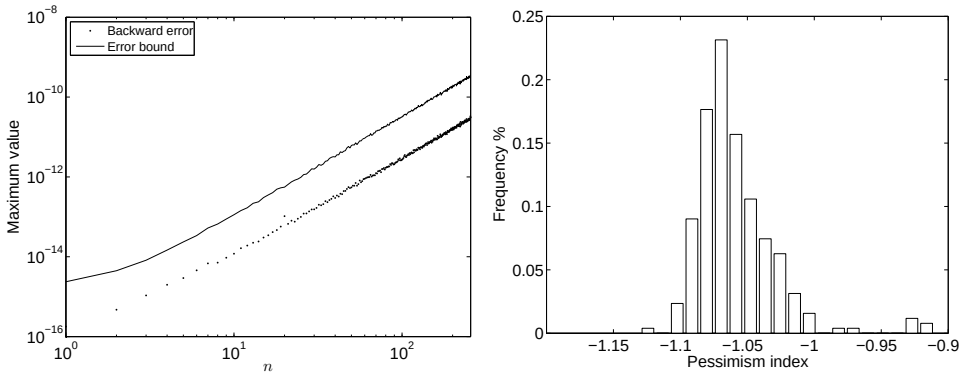
Figure 3.1: Chebyshev polynomials interpolated at their extreme points.

3.4.3 Polynomials taking on random values on a Chebyshev grid

In this experiment, we interpolate at Chebyshev nodes of the second kind. This time however, we interpolate random normally distributed values with mean zero and standard deviation of 10. We compute the maximum relative backward error (3.35) and the associated bound (3.37) for each of 100 different degree n polynomials, then take the mean of these values, for n varying from 1 to 256.

Figure 3.2(a) shows the mean relative backward error and bound as a function of the degree. Again, we see that the backward error grows like $O(n^{2.5})$. The error bound is roughly one order of magnitude larger than the backward error, as illustrated by Figure 3.2(b).

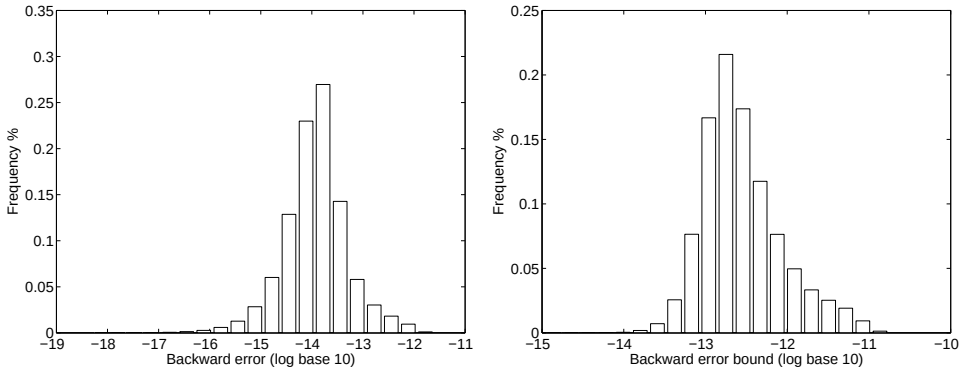
To further illustrate the distribution of the relative backward error and backward error bound, we repeat the experiment with a fixed degree $n = 50$. We take 10000 polynomials taking on random normally distributed values, at Chebyshev points of the first kind, with mean zero and standard deviation 10. Figures 3.3(a) and 3.3(b) show the distribution of the backward error as well as the distribution of the bound. This illustrates that on average we obtain a small backward error. Furthermore, as illustrated in Figure 3.4, the



(a) Mean relative backward error (b) Associated pessimism index

Figure 3.2: Polynomials taking on random normally distributed values at Chebyshev nodes of the first kind.

error bound is on average only one to two orders of magnitude larger than the observed backward error.



(a) Relative backward error (b) Error bound

Figure 3.3: Backward error and bound for 10000 degree 50 polynomials taking on random normally distributed values at Chebyshev points of the first kind.

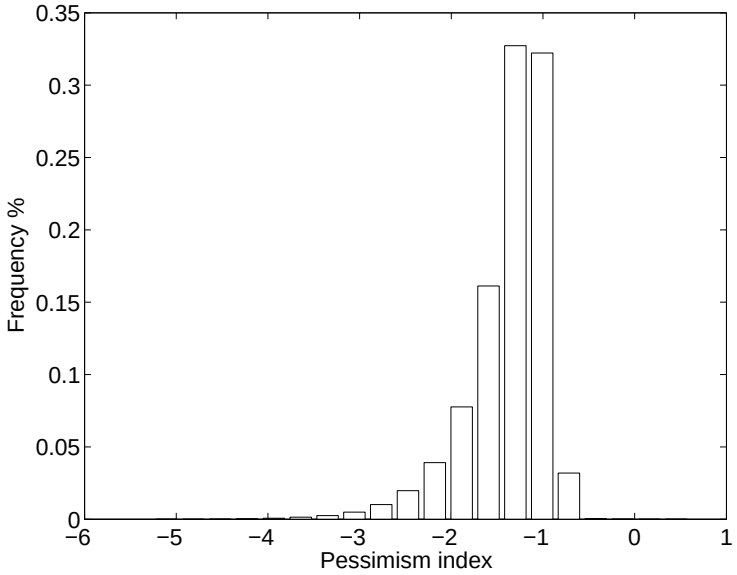


Figure 3.4: Pessimism index for 10000 degree 50 polynomials taking on random normally distributed values at Chebyshev points of the first kind.

3.4.4 Wilkinson polynomial

We investigate the effectiveness of balancing the companion matrix for the scaled Wilkinson polynomial

$$p(z) = \prod_{k=0}^{20} (z - k/21) \,. \tag{3.41}$$

We interpolate $p(z)$ at 21 equispaced points $x_k = (k+1/2)/21$, $0 \le k \le 20$. Then compute the eigenvalues for three different pairs: unscaled, scaled, and balanced.

As discussed by Berrut and Trefethen [5], to improve the stability of computing the barycentric weights (3.2), each difference $x_i - x_j$ in (3.2) should first be multiplied by C^{-1}, where $C = (b-a)/4$ is the *capacity* of the interval $[a, b]$ [22, p. 92]. In this case the capacity is $C = 1/4$, and thus multiplying each factor $x_i - x_j$ by C^{-1} has the effect of rescaling the weights by a factor of C^{20}, that is $\widehat{w}_j = w_j/4^{20}$.

For the unscaled pair, we use the scaled weights $\widehat{w}_j$ and the original function

values. For the scaled pair, we scale the weights and values so they have unit 2−norm. For the balanced pair, we balance the weights and values via (3.30), then scale the first row and column to have unit 2−norm.

From these three pairs, we compute the maximum relative backward error (3.35), the associated error bound (3.35), and the forward error in the computed roots. The results are shown in Table 3.3. The unscaled pair performs poorly, this may be partly explained by the difference in the norms of the first row and column ($\|\mathbf{f}\|_2 \approx 1.6 \times 10^{-9}$ and $\|\widehat{\mathbf{w}}\|_2 \approx 14.6$). Once we normalize the first row and column, the errors reduce significantly. Balancing the pair produces the greatest reduction in the error, the relative forward error is approximately 35 times machine precision.

Table 3.3: Wilkinson polynomial interpolated at equispaced points.

	Backward error	Error bound	Forward error
Unscaled	8.65×10^{-7}	1.09×10^{-5}	1.69×10^{-3}
Scaled	4.07×10^{-14}	7.08×10^{-9}	1.06×10^{-12}
Balanced	9.81×10^{-14}	3.69×10^{-13}	2.66×10^{-15}

The relative error bound also predicts the backward error quite well for the unscaled pair, and the balanced pair. However, for the scaled pair, the bound overestimates the backward error by about 5 orders of magnitude. Between the scaled and balanced pairs, we see a reduction in the forward error of three orders of magnitude. Hence, balancing the pair reduces the condition number significantly.

We should also point out that the roots of $p(z)$ interlace the points x_k, and thus are quite well suited for computing the roots. We repeated the analysis using Chebyshev points of the first kind on the interval $[0, 1]$, the results of which are shown in Table 3.4.

Table 3.4: Wilkinson polynomial interpolated at Chebyshev points.

	Backward error	Error bound	Forward error
Unscaled	5.93×10^{-10}	7.63×10^{-7}	2.29×10^{-8}
Scaled	8.48×10^{-14}	3.00×10^{-12}	3.79×10^{-11}
Balanced	9.88×10^{-14}	2.50×10^{-12}	5.03×10^{-12}

We do not see much of a difference in the backward error between the scaled and balanced pairs, but the forward error has been reduced by an order of magnitude.

3.4.5 Wilkinson filter example

In [25], we find a polynomial rootfinding problem that is interesting to attack using Lagrange interpolation. The discussion there begins "As a second example, we give a polynomial expression which arose in filter design. The zeros were required of the function $p(z)$ defined by"

$$p(z) = \prod_{i=1}^{7} \left(z^2 + A_i z + B_i\right) - k \prod_{i=1}^{6} (z + C_i)^2 \,, \tag{3.42}$$

with the data values as given below:

$$\mathbf{A} = \begin{bmatrix} 2.008402247 \\ 1.974225110 \\ 1.872661356 \\ 1.714140938 \\ 1.583160527 \\ 1.512571776 \\ 1.485030592 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 1.008426206 \\ 0.9749050168 \\ 0.8791058345 \\ 0.7375810928 \\ 0.6279419845 \\ 0.5722302977 \\ 0.5513324340 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 0 \\ 0.7015884551 \\ 0.6711668301 \\ 0.5892018711 \\ 1.084755941 \\ 1.032359024 \end{bmatrix}$$

and $k = 1.380 \times 10^{-8}$. When expanded into the monomial basis centred at 0, Wilkinson claims that this polynomial is very ill-conditioned: "The explicit polynomial $p(z)$ is so ill-conditioned that the double-precision Bairstow pro-

gramme gave only 2 correct figures in several of the factors and the use of the treble-precision section was essential." He later observes that if $p(z)$ is expanded into the shifted monomial basis centred at $z = -0.85$, it's not so badly conditioned.

We interpolate $p(z)$ at the roots of

$$\prod_{i=1}^{7} \left(z^2 + A_i z + B_i\right) \ , \tag{3.43}$$

and at one extra point: the mean of the roots of (3.43), giving 15 interpolation nodes in total. We compute the eigenvalues using the balanced and scaled pair $(\widehat{\mathbf{A}}, \mathbf{B})$ (using $\widehat{\mathbf{A}}$ from (3.33)). The relative backward error (3.35), backward error bound (3.37), and pessimism index are shown in Table 3.5. We see that again, for this problem the backward error is excellent, and the bound agrees within an order of magnitude of the actual error.

Table 3.5: Backward error and bound for Wilkinson's filter example.

	Backward Error	Error bound	Pessimism index
f_0	2.55×10^{-14}	3.39×10^{-13}	-1.12
f_1	8.76×10^{-16}	1.39×10^{-14}	-1.20
f_2	2.36×10^{-16}	1.39×10^{-14}	-1.77
f_3	2.73×10^{-15}	5.41×10^{-14}	-1.30
f_4	1.59×10^{-15}	5.41×10^{-14}	-1.53
f_5	1.69×10^{-14}	2.37×10^{-13}	-1.15
f_6	1.66×10^{-14}	2.37×10^{-13}	-1.15
f_7	5.91×10^{-14}	9.26×10^{-13}	-1.20
f_8	6.50×10^{-14}	9.26×10^{-13}	-1.15
f_9	4.07×10^{-14}	1.26×10^{-12}	-1.49
f_{10}	1.26×10^{-13}	1.26×10^{-12}	-1.00
f_{11}	3.98×10^{-14}	5.09×10^{-13}	-1.11
f_{12}	2.26×10^{-14}	5.09×10^{-13}	-1.35
f_{13}	1.64×10^{-14}	1.32×10^{-13}	-0.90
f_{14}	2.12×10^{-14}	1.32×10^{-13}	-0.79

3.5 Concluding Remarks

In this chapter, we have shown that computing the roots of interpolants via the eigenvalues of a companion matrix pair $(\mathbf{A}, \mathbf{B})$ is normwise backward stable. We have proposed a bound on the backward error, which is typically only an order of magnitude larger than the actual error. Through a number of numerical experiments, we have shown that the roots of the interpolating polynomial can be found accurately and in a stable manner. The first-order analysis presented here suggests that the forward error of the roots of polynomials computed by this method will usually be small, because the condition number of polynomial evaluation and rootfinding is usually small in these bases.

Bibliography

[1] A. Amiraslani, R. M. Corless, L. Gonzalez-Vega, and A. Shakoori, *Polynomial algebra by values*, Tech. Rep. TR-04-01, Ontario Research Centre for Computer Algebra, http://www.orcca.on.ca/TechReports, January 2004.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, PA, third ed., 1999.

[3] V. I. Arnold, *On matrices depending on parameters*, Russian Mathematical Surveys, 26 (1971), pp. 29–43.

[4] J.-P. Berrut, *Rational functions for guaranteed and experimentally well-conditioned global interpolation*, Computers & Mathematics with Applications, 15 (1988), pp. 1–16.

[5] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Review, 46 (2004), pp. 501–517.

[6] R. M. Corless, *Generalized companion matrices in the Lagrange basis*, in Proceedings EACA, L. Gonzalez-Vega and T. Recio, eds., June 2004, pp. 317–322.

[7] ——, *On a generalized companion matrix pencil for matrix polynomials expressed in the Lagrange basis*, in Symbolic-Numeric Computation, D. Wang and L. Zhi, eds., Trends in Mathematics, Birkhuser Basel, 2007, pp. 1–15.

[8] R. M. Corless and S. M. Watt, *Bernstein bases are optimal, but, sometimes, Lagrange bases are better*, in Proceedings of SYNASC, Timisoara, MIRTON Press, September 2004, pp. 141–153.

[9] D. Day and L. Romero, *Roots of polynomials expressed in terms of orthogonal polynomials*, SIAM Journal on Numerical Analysis, 43 (2005), p. 1969.

[10] A. Edelman and H. Murakami, *Polynomial roots from companion matrix eigenvalues*, Mathematics of Computation, 64 (1995), pp. 763–776.

[11] L. Elsner and P. Rózsa, *On eigenvectors and adjoints of modified matrices*, Linear and Multilinear Algebra, 10 (1981), pp. 235–247.

[12] M. A. Golberg, *The derivative of a determinant*, American Mathematical Monthly, 79 (1972), pp. 1124–1126.

[13] I. J. Good, *The Colleague matrix, a Chebyshev analogue of the companion matrix*, The Quarterly Journal of Mathematics, 12 (1961), pp. 61–68.

[14] N. Higham, R. Li, and F. Tisseur, *Backward error of polynomial eigenproblems solved by linearization*, SIAM Journal on Matrix Analysis and Applications, 29 (2008), pp. 1218–1241.

[15] P. W. Lawrence and R. M. Corless, *Numerical stability of barycentric Hermite root-finding*, in Proceedings of the 2011 International Workshop on Symbolic-Numeric Computation, ACM, 2012, pp. 147–148.

[16] D. Lemonnier and P. Van Dooren, *Optimal scaling of block companion pencils*, in International Symposium on Mathematical Theory of Networks and Systems, Leuven, Belgium, 2004.

[17] D. Lemonnier and P. Van Dooren, *Balancing regular matrix pencils*, SIAM Journal on Matrix Analysis and Applications, 28 (2006), pp. 253–263.

[18] C. Moler, *Cleves corner: Roots—of polynomials, that is*, The MathWorks Newsletter, 5 (1991), pp. 8–9.

[19] B. Parlett and C. Reinsch, *Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numerische Mathematik, 13 (1969), pp. 293–304.

[20] W. Specht, *Die Lage der Nullstellen eines Polynoms. III*, Mathematische Nachrichten, 16 (1957), pp. 369–389.

[21] K.-C. Toh and L. N. Trefethen, *Pseudozeros of polynomials and pseudospectra of companion matrices*, Numerische Mathematik, 68 (1994), pp. 403–425.

[22] L. N. Trefethen, *Approximation theory and approximation practice*, Society for Industrial and Applied Mathematics, 2012.

[23] R. C. Ward, *Balancing the generalized eigenvalue problem*, SIAM Journal on Scientific and Statistical Computing, 2 (1981), pp. 141–152.

[24] D. Watkins, *The matrix eigenvalue problem: GR and Krylov subspace methods*, SIAM, 2007.

[25] J. H. Wilkinson, *The evaluation of the zeros of ill-conditioned polynomials. Part II*, Numerische Mathematik, 1 (1959), pp. 167–180.

Chapter 4

Backward Stability of Polynomial Eigenvalue Problems Expressed in the Lagrange Basis[1]

4.1 Introduction

The standard approach to solving the polynomial eigenvalue problem is to linearize, which is to say the problem is transformed into an equivalent larger order generalized eigenproblem. For the monomial basis, much work has been done to show the conditions under which linearizations produce small backward errors. In this work, we extend these results to the Lagrange basis. We show that computing the eigensystem of a certain linearization of a matrix polynomial expressed in barycentric Lagrange form is numerically stable, and give a bound for the backward errors. We also show how the linearization may be block balanced, so as to produce eigenvalues and eigenvectors with small backward error.

For an arbitrary basis $\{\phi_0(z), \cdots, \phi_n(z)\}$ spanning $\mathbb{P}^n$, the space of polynomials in z of degree at most n, and $m \times m$ matrices $\mathbf{C}_k$, we may define a

[1] A version of this chapter has been submitted to Linear Algebra and its Applications for publication.

matrix polynomial $\mathbf{P}(z)$ by

$$\mathbf{P}(z) = \sum_{j=0}^{n} \mathbf{C}_j \phi_j(z) \,. \tag{4.1}$$

A pair $(\lambda, \mathbf{x})$ is called a right eigenpair of the matrix polynomial $\mathbf{P}(z)$ if

$$\mathbf{P}(\lambda)\mathbf{x} = \mathbf{0} \,, \tag{4.2}$$

and, analogously, a left eigenpair $(\lambda, \mathbf{y}^*)$ satisfies

$$\mathbf{y}^*\mathbf{P}(\lambda) = \mathbf{0}^* \,. \tag{4.3}$$

In this work, we shall assume that the matrix polynomial is regular, that is, $\det \mathbf{P}(z)$ is not identically zero.

Matrix polynomials are usually expressed in the monomial basis [9, 12, 16], that is,

$$\mathbf{P}(z) = z^n \mathbf{A}_n + z^{n-1}\mathbf{A}_{n-1} + \cdots + \mathbf{A}_0 \,. \tag{4.4}$$

However, there has been growing interest in matrix polynomials expressed in other bases [1, 6, 17], either due to the construction of the matrix polynomials themselves, or in order to take advantage of the properties of the polynomial basis.

In this work, we consider matrix polynomials expressed in barycentric Lagrange form [2]. In the scalar case ($m = 1$), the unique polynomial of degree less than or equal to n, interpolating a set of $n + 1$ values $\{f_0, \ldots, f_n\}$ at the set of nodes $\{x_0, \ldots, x_n\}$, can be written as

$$p(z) = \ell(z) \sum_{j=0}^{n} \frac{w_j f_j}{z - x_j} \,, \tag{4.5}$$

where the nodal polynomial $\ell(z)$ is defined by

$$\ell(z) = \prod_{i=0}^{n} (z - x_i) \,, \tag{4.6}$$

and the barycentric weights w_j are defined by

$$w_j = \prod_{k \neq j} (x_j - x_k)^{-1} , \quad 0 \leq j \leq n . \tag{4.7}$$

It is immediately clear that we may also interpolate a set of matrix values $\{\mathbf{F}_0, \ldots, \mathbf{F}_n\}$ at the nodes $\{x_0, \ldots, x_n\}$ and construct the unique matrix polynomial interpolant

$$\mathbf{P}(z) = \ell(z) \sum_{j=0}^{n} \frac{w_j \mathbf{F}_j}{z - x_j} . \tag{4.8}$$

It is desirable to use the barycentric formulation because it has been shown (in the scalar case) that evaluation is numerically stable [11]. The matrix polynomial interpolant is made up of scalar interpolants of the entries of the matrix polynomial. Thus, it follows that the evaluation of the matrix polynomial via the formula (4.8) is also numerically stable.

As demonstrated in [4], the eigenvalues of $\mathbf{P}(z)$ can be found via the eigenvalues of a generalized companion matrix pair, defined by

$$(\mathbf{A}, \mathbf{B}) = \left(\begin{bmatrix} \mathbf{0} & -\mathbf{F}^T \\ \mathbf{W} & \mathbf{D} \end{bmatrix}, \begin{bmatrix} \mathbf{0} & \\ & \mathbf{I} \end{bmatrix} \right) , \tag{4.9}$$

where $\mathbf{F}^T = \begin{bmatrix} \mathbf{F}_0 & \ldots & \mathbf{F}_n \end{bmatrix}$, $\mathbf{W} = \begin{bmatrix} w_0 & \cdots & w_n \end{bmatrix}^T \otimes \mathbf{I}_m$,

$$\mathbf{D} = \begin{bmatrix} x_0 & & \\ & \ddots & \\ & & x_n \end{bmatrix} \otimes \mathbf{I}_m , \tag{4.10}$$

and $\otimes$ denotes the Kronecker product (see, for example, [18, p. 65]).

Remark 12. *The formulation* $(\mathbf{A}, \mathbf{B})$ *introduces* $2m$ *spurious infinite eigenvalues. We will show in §4.3 that these spurious infinite eigenvalues can be deflated exactly from the pair. Hence, they do not affect the accuracy of the finite (or infinite) eigenvalues.*

4.2 Numerical Stability of Eigenvalues Found Through Linearization

For a set of weights $\alpha_k \geq 0$, not all equal to zero, define the absolute condition number for evaluation of a polynomial [5, 7]:

$$B(z) = \sum_{j=0}^{n} \alpha_j |\phi_j(z)| \,. \tag{4.11}$$

Then define

$$\Delta\mathbf{P}(\lambda) = \sum_{j=0}^{n} \Delta\mathbf{C}_j \phi_j(\lambda) \,, \tag{4.12}$$

where the $\Delta\mathbf{C}_j$'s are small perturbations of the coefficient matrices $\mathbf{C}_j$.

The normwise backward error of a finite approximate right eigenpair $(\lambda, \mathbf{x})$ of a matrix polynomial $\mathbf{P}(z)$, expressed in an arbitrary basis, is defined by

$$\eta_P(\lambda, \mathbf{x}) = \min\{\varepsilon : (\mathbf{P}(\lambda)+\Delta\mathbf{P}(\lambda))\mathbf{x} = \mathbf{0}, \|\Delta\mathbf{C}_j\|_2 \leq \varepsilon\alpha_j, 0 \leq j \leq n\}. \tag{4.13}$$

This is a trivial generalization of the definition for the monomial basis [10].

Similarly, for a left eigenpair $(\lambda, \mathbf{y}^*)$ the backward error is defined by

$$\eta_P(\lambda, \mathbf{y}^*) = \min\{\varepsilon : \mathbf{y}^*(\mathbf{P}(\lambda) + \Delta\mathbf{P}(\lambda)) = \mathbf{0}^*, \ \|\Delta\mathbf{C}_j\|_2 \leq \varepsilon\alpha_j, \ 0 \leq j \leq n\} \,. \tag{4.14}$$

The α_j's determine how the perturbations $\Delta\mathbf{C}_j$ to the coefficients $\mathbf{C}_j$ are measured: setting $\alpha_j = 1$ results in absolute perturbations, whereas $\alpha_j = \|\mathbf{C}_j\|_2$ results in relative perturbations to the coefficients.

For the monomial basis, Tisseur [15] obtained explicit expressions for the backward errors of approximate left and right eigenpairs $(\lambda, \mathbf{y}^*)$ and $(\lambda, \mathbf{x})$, respectively, of $\mathbf{P}(z)$ given by

$$\eta_P(\lambda, \mathbf{x}) = \frac{\|\mathbf{P}(\lambda)\mathbf{x}\|_2}{B_M(\lambda)\|\mathbf{x}\|_2} \,, \tag{4.15}$$

and

$$\eta_P(\lambda, \mathbf{y}^*) = \frac{\|\mathbf{y}^*\mathbf{P}(\lambda)\|_2}{B_M(\lambda)\|\mathbf{y}\|_2}\,, \tag{4.16}$$

where

$$B_M(z) = \sum_{j=0}^{n} \|\mathbf{A}_j\|_2 |z|^j\,. \tag{4.17}$$

The Lagrange basis elements are

$$\phi_k(z) = \ell(z)\frac{w_j}{z - x_j}\,, \tag{4.18}$$

and thus the equivalent result in the Lagrange basis has $B_L(z)$ instead of $B_M(z)$ where

$$B_L(z) = \sum_{j=0}^{n} \|\mathbf{F}_j\|_2 \frac{|\ell(z)w_j|}{|z - x_j|}\,. \tag{4.19}$$

Hence, in the Lagrange basis, the backward errors of an approximate left and right eigenpairs $(\lambda, \mathbf{x})$ and $(\lambda, \mathbf{y}^*)$, respectively, are given by

$$\eta_P(\lambda, \mathbf{x}) = \frac{\|\mathbf{P}(\lambda)\mathbf{x}\|_2}{B_L(\lambda)\|\mathbf{x}\|_2}\,, \tag{4.20}$$

and

$$\eta_P(\lambda, \mathbf{y}^*) = \frac{\|\mathbf{y}^*\mathbf{P}(\lambda)\|_2}{B_L(\lambda)\|\mathbf{y}\|_2}\,. \tag{4.21}$$

Now, for an approximate right eigenpair $(\lambda, \mathbf{z})$ of the linearization $(\mathbf{A}, \mathbf{B})$, the backward error is given by

$$\eta_{(\mathbf{A},\mathbf{B})}(\lambda, \mathbf{z}) = \frac{\|(\lambda\mathbf{B} - \mathbf{A})\mathbf{z}\|_2}{(|\lambda|\|\mathbf{B}\|_2 + \|\mathbf{A}\|_2)\,\|\mathbf{z}\|_2}\,. \tag{4.22}$$

We aim to bound the backward error of $\mathbf{P}(z)$ by the backward error of the linearization $(\mathbf{A}, \mathbf{B})$, and thus we need to relate the eigenvectors of $(\mathbf{A}, \mathbf{B})$ to those of $\mathbf{P}(z)$. As it so happens, one may recover the eigenvector of $\mathbf{P}(z)$ directly from an eigenvector of $(\mathbf{A}, \mathbf{B})$, as we demonstrate in the following lemma.

Lemma 6. *For an eigenvalue $\lambda \neq x_i$, $0 \leq i \leq n$, of the pair $(\mathbf{A}, \mathbf{B})$, the corresponding left and right eigenvectors ($\mathbf{w}^*$ and $\mathbf{z}$, respectively) are given by*

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \frac{w_0}{\lambda - x_0}\mathbf{x} \\ \vdots \\ \frac{w_n}{\lambda - x_n}\mathbf{x} \end{bmatrix}, \tag{4.23}$$

and

$$\mathbf{w}^* = \begin{bmatrix} \mathbf{y}^* & -\mathbf{y}^*\frac{\mathbf{F}_0}{\lambda - x_0} & \cdots & -\mathbf{y}^*\frac{\mathbf{F}_n}{\lambda - x_n} \end{bmatrix}, \tag{4.24}$$

where $\mathbf{y}^*$ and $\mathbf{x}$ are the left and right eigenvectors of $\mathbf{P}(\lambda)$.

Proof. Let $(\lambda, \mathbf{w}^*)$ and $(\lambda, \mathbf{z})$ be the left and right eigenpairs of $(\mathbf{A}, \mathbf{B})$, respectively. Partition $\mathbf{z}$ and $\mathbf{w}$ conformably with the blocks of $\mathbf{A}$:

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_{-1} \\ \mathbf{z}_0 \\ \vdots \\ \mathbf{z}_n \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} \mathbf{w}_{-1} \\ \mathbf{w}_0 \\ \vdots \\ \mathbf{w}_n \end{bmatrix}. \tag{4.25}$$

The right eigenpair $(\lambda, \mathbf{z})$ satisfies $(\lambda\mathbf{B} - \mathbf{A})\mathbf{z} = \mathbf{0}$, or

$$\begin{bmatrix} \mathbf{0} & \mathbf{F}_0 & \cdots & \mathbf{F}_n \\ -w_0\mathbf{I}_m & (\lambda - x_0)\mathbf{I}_m & & \\ \vdots & & \ddots & \\ -w_n\mathbf{I}_m & & & (\lambda - x_n)\mathbf{I}_m \end{bmatrix} \begin{bmatrix} \mathbf{z}_{-1} \\ \mathbf{z}_0 \\ \vdots \\ \mathbf{z}_n \end{bmatrix} = \mathbf{0}\,. \tag{4.26}$$

Thus, we have

$$\sum_{j=0}^{n} \mathbf{F}_j\mathbf{z}_j = \mathbf{0}\,, \tag{4.27}$$

and

$$(\lambda - x_j)\mathbf{z}_j = w_j\mathbf{z}_{-1}\,, \quad 0 \leq j \leq n\,. \tag{4.28}$$

Solving the relations (4.28) for $\mathbf{z}_j$ (assuming that $\lambda \neq x_j$), and substituting

them into (4.27) yields

$$\left(\sum_{j=0}^{n} \frac{w_j \mathbf{F}_j}{\lambda - x_j}\right) \mathbf{z}_{-1} = \mathbf{0}\,. \tag{4.29}$$

This is equal to $\ell(\lambda)^{-1}\mathbf{P}(\lambda)\mathbf{z}_{-1} = \mathbf{0}$, so long as λ is not an interpolation node.

Thus, if $(\lambda, \mathbf{z})$ is an eigenpair of $(\mathbf{A}, \mathbf{B})$, where $\lambda \neq x_i$ for all i, $0 \leq i \leq n$, then $(\lambda, \mathbf{z}_{-1})$ is an eigenpair of $\mathbf{P}(z)$. The left eigenpair $(\lambda, \mathbf{w}^*)$ satisfies $\mathbf{w}^*(\lambda\mathbf{B} - \mathbf{A}) = \mathbf{0}^*$, which yields the relations

$$-\sum_{j=0}^{n} w_j \mathbf{w}_j^* = \mathbf{0}^* \tag{4.30}$$

and

$$(\lambda - x_j)\mathbf{w}_j^* = -\mathbf{w}_{-1}^* \mathbf{F}_j, \quad 0 \leq j \leq n\,, \tag{4.31}$$

which together yield

$$\mathbf{w}_{-1}^* \left(\sum_{j=0}^{n} \frac{w_j \mathbf{F}_j}{\lambda - x_j}\right) = \mathbf{0}^*\,. \tag{4.32}$$

Thus, if $(\lambda, \mathbf{w}^*)$ is a left eigenpair of $(\mathbf{A}, \mathbf{B})$, then $(\lambda, \mathbf{w}_{-1}^*)$ is a left eigenpair of $\mathbf{P}(z)$. □

Remark 13. *If x_i were an eigenvalue of $(\mathbf{A}, \mathbf{B})$, then (4.28) gives $\mathbf{z}_{-1} = \mathbf{0}$. Since the nodes x_j are distinct, we have $\mathbf{z}_j = \mathbf{0}$ for $j \neq i$, $0 \leq j \leq n$. We are left with one equation: $\mathbf{F}_i\mathbf{z}_i = \mathbf{0}$, or $\mathbf{P}(x_i)\mathbf{z}_i = \mathbf{0}$. Similarly, for the left eigenvector of $(\mathbf{A}, \mathbf{B})$, we obtain $\mathbf{w}_i^*\mathbf{P}(x_i) = \mathbf{0}^*$.*

Theorem 5. *We may bound the backward error of an approximate right eigenpair $(\lambda, \mathbf{x})$ of $\mathbf{P}(z)$ by*

$$\eta_P(\lambda, \mathbf{x}) \leq \frac{|\lambda|\|\mathbf{B}\|_2 + \|\mathbf{A}\|_2}{B_L(\lambda)} \cdot \frac{\|\mathbf{G}(\lambda)\|_2\|\mathbf{z}\|_2}{\|\mathbf{x}\|_2} \cdot \eta_{(\mathbf{A},\mathbf{B})}(\lambda, \mathbf{z})\,, \tag{4.33}$$

where $\eta_{(\mathbf{A},\mathbf{B})}(\lambda, \mathbf{z})$ is the backward error of an approximate right eigenpair $(\lambda, \mathbf{z})$

of $(\mathbf{A}, \mathbf{B})$*, and*

$$\mathbf{G}(\lambda) = \ell(\lambda) \begin{bmatrix} \mathbf{I}_m & -\frac{\mathbf{F}_0}{\lambda - x_0} & \cdots & -\frac{\mathbf{F}_n}{\lambda - x_n} \end{bmatrix} . \tag{4.34}$$

Similarly, we may bound the backward error of an approximate left eigenpair $(\lambda, \mathbf{y}^*)$ *of* $\mathbf{P}(z)$ *by*

$$\eta_P(\lambda, \mathbf{y}^*) \leq \frac{|\lambda| \|\mathbf{B}\|_2 + \|\mathbf{A}\|_2}{B_L(\lambda)} \cdot \frac{\|\mathbf{H}(\lambda)\|_2 \|\mathbf{w}\|_2}{\|\mathbf{y}\|_2} \cdot \eta_{(\mathbf{A},\mathbf{B})}(\lambda, \mathbf{w}^*) , \tag{4.35}$$

where

$$\mathbf{H}(\lambda) = \ell(\lambda) \begin{bmatrix} \mathbf{I}_m \\ \frac{w_0}{\lambda - x_0} \mathbf{I}_m \\ \vdots \\ \frac{w_n}{\lambda - x_n} \mathbf{I}_m \end{bmatrix} . \tag{4.36}$$

Proof. Following Higham [10], we aim to find an $m \times mn$ matrix polynomial $\mathbf{G}(\lambda)$, such that

$$\mathbf{G}(\lambda)(\lambda \mathbf{B} - \mathbf{A}) = \mathbf{g}^T \otimes \mathbf{P}(\lambda) \tag{4.37}$$

for some nonzero $\mathbf{g} \in \mathbb{C}^n$. A simple calculation shows that

$$\mathbf{G}(\lambda) = \ell(\lambda) \begin{bmatrix} \mathbf{I}_m & -\frac{\mathbf{F}_0}{\lambda - x_0} & \cdots & -\frac{\mathbf{F}_n}{\lambda - x_n} \end{bmatrix} , \tag{4.38}$$

and $\mathbf{g} = \mathbf{e}_1$. We have

$$\mathbf{G}(\lambda)(\lambda \mathbf{B} - \mathbf{A})\mathbf{z} = (\mathbf{g}^T \otimes \mathbf{P}(\lambda))\mathbf{z} = \mathbf{P}(\lambda)(\mathbf{g}^T \otimes \mathbf{I}_m)\mathbf{z} = \mathbf{P}(\lambda)\mathbf{x} , \tag{4.39}$$

and thus we may recover the right eigenvector of $\mathbf{P}(z)$ from the first m rows of the right eigenvector $\mathbf{z}$ of $(\mathbf{A}, \mathbf{B})$. Similarly, for the left eigenvectors, we aim to find an $nm \times m$ matrix polynomial $\mathbf{H}(\lambda)$, such that

$$(\lambda \mathbf{B} - \mathbf{A})\mathbf{H}(\lambda) = \mathbf{h} \otimes \mathbf{P}(\lambda) . \tag{4.40}$$

It is easy to show that the matrix polynomial $\mathbf{H}(\lambda)$ is given by

$$\mathbf{H}(\lambda) = \ell(\lambda) \begin{bmatrix} \mathbf{I}_m \\ \frac{w_0}{\lambda - x_0}\mathbf{I}_m \\ \vdots \\ \frac{w_n}{\lambda - x_n}\mathbf{I}_m \end{bmatrix}, \tag{4.41}$$

and $\mathbf{h} = \mathbf{e}_1$. We have

$$\mathbf{w}^*(\lambda\mathbf{B} - \mathbf{A})\mathbf{H}(\lambda) = \mathbf{w}^*(\mathbf{h} \otimes \mathbf{P}(\lambda)) = \mathbf{w}^*(\mathbf{h} \otimes \mathbf{I}_m)\mathbf{P}(\lambda) = \mathbf{y}^*\mathbf{P}(\lambda)\,, \tag{4.42}$$

so we may recover the left eigenvector of $\mathbf{P}(z)$ from the first m columns of $\mathbf{w}^*$.

Now, for a right eigenpair $(\lambda, \mathbf{z})$ of $(\mathbf{A}, \mathbf{B})$, we may combine (4.15), (4.22), and (4.39) to obtain the bound

$$\begin{aligned} \eta_P(\lambda, \mathbf{x}) &= \frac{\|\mathbf{G}(\lambda)(\lambda\mathbf{B} - \mathbf{A})\mathbf{z}\|_2}{B_L(\lambda)\|\mathbf{x}\|_2} \leq \frac{\|\mathbf{G}(\lambda)\|_2\|(\lambda\mathbf{B} - \mathbf{A})\mathbf{z}\|_2}{B_L(\lambda)\|\mathbf{x}\|_2} \\ &\leq \frac{|\lambda|\|\mathbf{B}\|_2 + \|\mathbf{A}\|_2}{B_L(\lambda)} \cdot \frac{\|\mathbf{G}(\lambda)\|_2\|\mathbf{z}\|_2}{\|\mathbf{x}\|_2} \cdot \eta_{(\mathbf{A},\mathbf{B})}(\lambda, \mathbf{z})\,. \end{aligned} \tag{4.43}$$

A similar combination of (4.16), (4.22), and (4.42) yields the bound

$$\eta_P(\lambda, \mathbf{y}^*) \leq \frac{|\lambda|\|\mathbf{B}\|_2 + \|\mathbf{A}\|_2}{B_L(\lambda)} \cdot \frac{\|\mathbf{H}(\lambda)\|_2\|\mathbf{w}\|_2}{\|\mathbf{y}\|_2} \cdot \eta_{(\mathbf{A},\mathbf{B})}(\lambda, \mathbf{w}^*)\,. \tag{4.44}$$

□

Theorem 6. *We may bound the ratio of the backward error of an approximate right eigenpair $(\lambda, \mathbf{x})$ of $\mathbf{P}(z)$ to the backward error of an approximate right eigenpair $(\lambda, \mathbf{z})$ of $(\mathbf{A}, \mathbf{B})$ by*

$$\frac{\eta_P(\lambda, \mathbf{x})}{\eta_{(\mathbf{A},\mathbf{B})}(\lambda, \mathbf{z})} \leq \sqrt{m}(|\lambda| + 1)\max\left(1, \|\mathbf{A}\|_2\right)\left(\frac{|\ell(\lambda)|}{\min_j \|\mathbf{F}_j\|_2} + \frac{\sqrt{m}}{\min_k |w_k|}\right) \cdot \frac{\|\mathbf{z}\|_2}{\|\mathbf{x}\|_2}\,. \tag{4.45}$$

Similarly, for an approximate left eigenpair,

$$\frac{\eta_P(\lambda, \mathbf{y}^*)}{\eta_{(\mathbf{A},\mathbf{B})}(\lambda, \mathbf{w}^*)} \leq (|\lambda| + 1) \max(1, \|\mathbf{A}\|_2) \frac{(|\ell(\lambda)| + 1)}{\min_j \|\mathbf{F}_j\|_2} \cdot \frac{\|\mathbf{w}\|_2}{\|\mathbf{y}\|_2} \,. \tag{4.46}$$

Proof. For an approximate right eigenpair $(\lambda, \mathbf{z})$ of $(\mathbf{A}, \mathbf{B})$, we may bound $B_L(\lambda)$ from below by

$$B_L(\lambda) = |\ell(\lambda)| \sum_{j=0}^{n} \frac{\|\mathbf{F}_j\|_2 |w_j|}{|\lambda - x_j|} \geq \min_k \|\mathbf{F}_k\|_2 \sum_{j=0}^{n} |\ell_j(\lambda)| \geq \min_k \|\mathbf{F}_k\|_2 \,, \tag{4.47}$$

where the $\ell_j(\lambda)$ are the Lagrange polynomials. We obtain a bound on $\mathbf{G}(\lambda)$, as follows:

$$\begin{aligned} \|\mathbf{G}(\lambda)\|_2 \leq \sqrt{m} \|\mathbf{G}(\lambda)\|_\infty &\leq \sqrt{m} |\ell(\lambda)| \left(1 + \sum_{j=0}^{n} \frac{\|\mathbf{F}_j\|_\infty}{|\lambda - x_j|} \right) \\ &\leq \sqrt{m} |\ell(\lambda)| \left(1 + \sqrt{m} \sum_{j=0}^{n} \frac{\|\mathbf{F}_j\|_2}{|\lambda - x_j|} \right) \\ &\leq \sqrt{m} \left(|\ell(\lambda)| + \sqrt{m} \frac{B_L(\lambda)}{\min_k |w_k|} \right) \,, \end{aligned} \tag{4.48}$$

and dividing this by $B_L(\lambda)$, we obtain the bound

$$\frac{\|\mathbf{G}(\lambda)\|_2}{B_L(z)} \leq \sqrt{m} \left(\frac{|\ell(\lambda)|}{\min_j \|\mathbf{F}_j\|_2} + \frac{\sqrt{m}}{\min_k |w_k|} \right) \,. \tag{4.49}$$

The bound (4.45) simply follows by rearranging (4.33).

For an approximate left eigenpair, we may bound the $\|\mathbf{H}(\lambda)\|_2$ as follows:

$$\begin{aligned} \|\mathbf{H}(\lambda)\|_2 \leq \sqrt{m} \|\mathbf{H}(\lambda)\|_1 &= \sqrt{m} |\ell(\lambda)| \left(1 + \sum_{j=0}^{n} \frac{|w_j|}{|\lambda - x_j|} \right) \\ &\leq \sqrt{m} \left(|\ell(\lambda)| + \frac{B_L(\lambda)}{\min_j \|\mathbf{F}_j\|_2} \right) \,. \end{aligned} \tag{4.50}$$

Dividing by $B_L(\lambda)$, we obtain the bound

$$\frac{\|\mathbf{H}(\lambda)\|_2}{B_L(\lambda)} \leq \frac{|\ell(\lambda)|+1}{\min_j \|\mathbf{F}_j\|_2}\,. \tag{4.51}$$

□

Remark 14. *The bound derived in the last theorem is quite crude. However, it does point out the primary drivers in the backward error bound. The term* $\|\mathbf{A}\|_2$ *contains terms involving the barycentric weights* $|w_j|$*, the nodes* $|x_i|$*, and the norms of the values* $\|\mathbf{F}_k\|_2$*. Hence, we see that the ratios of the barycentric weights matter, as does the magnitude of the coefficients. The other factor which arises is the distance of the eigenvalue* λ *to the set of interpolation nodes (the* $|\ell(\lambda)|$ *term will be very large). Thus, we can expect to obtain good backward error if the interpolation nodes are near to the eigenvalues.*

Remark 15. *As we will show in §4.4, we may block-wise balance the matrix* $\mathbf{A}$ *to bring the pair* $(\mathbf{A},\mathbf{B})$ *closer to a normal pair. Experimentally, we have observed that with the balancing matrices proposed in §4.4, the ratio* $\|\mathbf{G}(\lambda)\|_2/B_L(\lambda)$ *seems always to be less than one. However, we have no proof of this.*

4.3 Deflation of Spurious Infinite Eigenvalues

As we noted in the introduction, the formulation $(\mathbf{A},\mathbf{B})$ introduces $2m$ spurious infinite eigenvalues. In this section, we show how these may be deflated exactly from the pair by reducing the pair to m-Hessenberg form via block-wise Givens rotations. The reduction to m-Hessenberg form is essentially the same as for the scalar case ($m = 1$) described in Chapter 2, except that the Givens rotations on the right must be applied block-wise to $\mathbf{F}^T$.

We can write the matrix as

$$\mathbf{A} = \begin{bmatrix} 0 & \mathbf{0}^T \\ \widehat{\mathbf{w}} & \widehat{\mathbf{D}} \end{bmatrix} \otimes \mathbf{I}_m + \mathbf{e}_1 \otimes \begin{bmatrix} \mathbf{0} & -\mathbf{F}^T \end{bmatrix}\,, \tag{4.52}$$

where $\widehat{\mathbf{w}} = \begin{bmatrix} w_0 & \cdots & w_n \end{bmatrix}^T$, and $\widehat{\mathbf{D}} = \operatorname{diag}\begin{bmatrix} x_0 & \cdots & x_n \end{bmatrix}$. There exists a unitary matrix $\mathbf{Q}_1$, such that

$$\begin{bmatrix} 0 & \mathbf{0}^T \\ h_0\mathbf{e}_1 & \mathbf{H}_1 \end{bmatrix} = \begin{bmatrix} 1 & \\ & \mathbf{Q}_1^* \end{bmatrix} \begin{bmatrix} 0 & \mathbf{0}^T \\ \widehat{\mathbf{w}} & \widehat{\mathbf{D}} \end{bmatrix} \begin{bmatrix} 1 & \\ & \mathbf{Q}_1 \end{bmatrix}, \tag{4.53}$$

and $\mathbf{H}_1$ is upper Hessenberg. Then define

$$\mathbf{Q} = \begin{bmatrix} \mathbf{I}_m & \\ & \mathbf{Q}_1 \otimes \mathbf{I}_m \end{bmatrix}, \tag{4.54}$$

and form

$$\mathbf{Q}^*\mathbf{A}\mathbf{Q} = \begin{bmatrix} 0 & \mathbf{0}^T \\ h_0\mathbf{e}_1 & \mathbf{H}_1 \end{bmatrix} \otimes \mathbf{I}_m + \mathbf{e}_1 \otimes \begin{bmatrix} \mathbf{0} & -\mathbf{F}^T(\mathbf{Q}_1 \otimes \mathbf{I}_m) \end{bmatrix}, \tag{4.55}$$

which is now an m-Hessenberg matrix. The product $\mathbf{F}^T(\mathbf{Q}_1 \otimes \mathbf{I}_m)$ may be computed quite efficiently; for example, if $\mathbf{Q}_1$ is constructed as a product of Givens rotations, we may apply them block-wise to $\mathbf{F}^T$. In fact, doing so will also retain the structure of the coefficients of the matrix polynomial. Applying $\mathbf{Q}$ to $\mathbf{B}$ yields $\mathbf{Q}^*\mathbf{B}\mathbf{Q} = \mathbf{B}$.

We may now begin to deflate the $2m$ spurious infinite eigenvalues from the pair $(\mathbf{Q}^*\mathbf{A}\mathbf{Q}, \mathbf{B})$ by applying block-wise Givens rotations to the pair. Because the leading $m \times m$ submatrix of $\mathbf{Q}^*\mathbf{A}\mathbf{Q}$ is zero, we may annihilate the $m \times m$ diagonal block $h_0\mathbf{I}_m$ below it by applying the permutation matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_m & \\ \mathbf{I}_m & \mathbf{0} & \\ & & \mathbf{I}_{mn} \end{bmatrix} \tag{4.56}$$

to the pair. This introduces a nilpotent Jordan block to the leading $2m \times 2m$ submatrix of $\mathbf{B}$:

$$\mathbf{P}\mathbf{B} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_m & \\ & \mathbf{0} & \\ & & \mathbf{I}_{nm} \end{bmatrix}. \tag{4.57}$$

The pair $\mathbf{PQ}^*(\mathbf{A},\mathbf{B})\mathbf{Q}$ is no longer unreduced, and we may deflate the first set of m spurious infinite eigenvalues from the pair by deleting the first m rows and columns of the transformed pair. The deflated pair is now

$$(\mathbf{H}_2,\mathbf{B}_2) = \left(\begin{bmatrix} -\mathbf{F}^T(\mathbf{Q}_1 \otimes \mathbf{I}_m) \\ \widehat{\mathbf{H}}_1 \otimes \mathbf{I}_m \end{bmatrix}, \begin{bmatrix} \mathbf{0} & \\ & \mathbf{I}_{nm} \end{bmatrix} \right), \tag{4.58}$$

where $\widehat{\mathbf{H}}_1$ is the matrix containing the last n rows of $\mathbf{H}_1$. Now, let $\widehat{\mathbf{F}} = -\mathbf{F}^T(\mathbf{Q}_1 \otimes \mathbf{I}_m)$, partition $\widehat{\mathbf{F}}$ as

$$\widehat{\mathbf{F}} = \begin{bmatrix} \widehat{\mathbf{F}}_0 & \cdots & \widehat{\mathbf{F}}_n \end{bmatrix}, \tag{4.59}$$

and let

$$\mathbf{U}_0\mathbf{\Sigma}_0\mathbf{V}_0^* = \widehat{\mathbf{F}}_0 \tag{4.60}$$

be the singular value decomposition of $\widehat{\mathbf{F}}_0$. Define the block diagonal matrices $\mathbf{U}$ and $\mathbf{V}$ by

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_0 & & & \\ & \mathbf{V}_0 & & \\ & & \ddots & \\ & & & \mathbf{V}_0 \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} \mathbf{V}_0 & & & \\ & \mathbf{V}_0 & & \\ & & \ddots & \\ & & & \mathbf{V}_0 \end{bmatrix}, \tag{4.61}$$

and form $\mathbf{U}^*(\mathbf{H}_2,\mathbf{B}_2)\mathbf{V}$, which yields

$$\mathbf{U}^*\mathbf{H}_2\mathbf{V} = \begin{bmatrix} \mathbf{\Sigma}_0 & \mathbf{U}_0^*\widehat{\mathbf{F}}_1\mathbf{V}_0 & \cdots & \cdots & \mathbf{U}_0^*\widehat{\mathbf{F}}_n\mathbf{V}_0 \\ h_{1,0}\mathbf{I}_m & h_{1,1}\mathbf{I}_m & \cdots & \cdots & h_{1,n}\mathbf{I}_m \\ & h_{2,1}\mathbf{I}_m & \ddots & & h_{2,n}\mathbf{I}_m \\ & & \ddots & \ddots & \vdots \\ & & & h_{n,n-1}\mathbf{I}_m & h_{n,n}\mathbf{I}_m \end{bmatrix}, \tag{4.62}$$

and $\mathbf{U}^*\mathbf{B}_2\mathbf{V} = \mathbf{B}_2$. We may deflate the second set of m spurious infinite eigenvalues from the pair $\mathbf{U}^*(\mathbf{H}_2,\mathbf{B}_2)\mathbf{V}$ by applying a set of m Givens rotations to the first $2m$ rows, annihilating the block $h_{1,0}\mathbf{I}_m$. This can be formulated as

a block-wise transformation: we construct diagonal $m \times m$ matrices $\mathbf{C}$ and $\mathbf{S}$, with $\mathbf{C}^2 + \mathbf{S}^2 = \mathbf{I}_m$, such that

$$\begin{bmatrix} \mathbf{C} & \mathbf{S} \\ -\mathbf{S} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_0 \\ h_{1,0}\mathbf{I}_m \end{bmatrix} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} . \tag{4.63}$$

Applying this transformation to the first $2m$ rows of $\mathbf{B}_2$ does not disturb the m zeros on the diagonal, nor the upper triangular structure. We do, however, introduce a block equal to $\mathbf{C}$ on the diagonal. If the leading coefficient of $\mathbf{P}(z)$ is nonsingular, then $\mathbf{C}$ will also be nonsingular. We may convert the generalized eigenvalue problem to a standard one by multiplying the second set of m rows by $\mathbf{C}^{-1}$.

After applying the second block Givens rotation matrix, we may deflate the second set of m infinite eigenvalues by deleting the first m rows and columns of the pair. This yields the pair $(\mathbf{H}_3, \mathbf{B}_3)$, where

$$\mathbf{H}_3 = \begin{bmatrix} h_{1,1}\mathbf{C} - \mathbf{S}\mathbf{U}_0^*\widehat{\mathbf{F}}_1\mathbf{V}_0 & \cdots & \cdots & h_{1,n}\mathbf{C} - \mathbf{S}\mathbf{U}_0^*\widehat{\mathbf{F}}_n\mathbf{V}_0 \\ h_{2,1}\mathbf{I}_m & \cdots & \cdots & h_{2,n}\mathbf{I}_m \\ & \ddots & & \vdots \\ & & h_{n,n-1}\mathbf{I}_m & h_{n,n}\mathbf{I}_m \end{bmatrix} , \tag{4.64}$$

and

$$\mathbf{B}_3 = \begin{bmatrix} \mathbf{C} & \\ & \mathbf{I}_{(n-1)m} \end{bmatrix} . \tag{4.65}$$

If the leading coefficient of $\mathbf{P}(z)$ is nonsingular, then we may simply convert the generalized eigenvalue problem to a standard one by multiplying on the left by $\mathbf{B}_3^{-1}$.

Remark 16. *If some of the singular values of $\widehat{\mathbf{F}}_0$ are zero, this indicates that the leading coefficient of the matrix polynomial $\mathbf{P}(z)$ is singular, and hence the pair $(\mathbf{A}, \mathbf{B})$ has other non-spurious infinite eigenvalues in addition to the $2m$ spurious ones.*

Remark 17. *If one or more of the singular values of $\widehat{\mathbf{F}}_0$ are zero, then the corresponding Givens rotations used to annihilate the $h_{1,0}\mathbf{I}_m$ block must permute*

the corresponding rows. Thus, another zero will have been introduced on the diagonal of $\mathbf{B}_3$*. If we were to order the singular values of* $\widehat{\mathbf{F}}_0$ *from smallest to largest, the zeros corresponding to non-spurious infinite eigenvalues would appear at the top left corner of the matrix, and we could deflate those eigenvalues by deleting the first row and column of the matrix pair.*

4.4 Block Scaling and Balancing

In this section, we discuss the issue of balancing the matrix pair $(\mathbf{A}, \mathbf{B})$ to improve the accuracy of the computed eigenvalues. In the scalar case, it was proposed in §3.3 that the matrix pair could be balanced by applying diagonal similarity transformations to the matrix $\mathbf{A}$ in order to bring the pair closer to a normal pair. The optimal diagonal matrix that achieves this is

$$\mathbf{D}_s = \begin{bmatrix} 1 & & & \\ & \sqrt{|w_0|/|f_0|} & & \\ & & \ddots & \\ & & & \sqrt{|w_n|/|f_n|} \end{bmatrix}, \tag{4.66}$$

where the f_j's are the values of the polynomial at the nodes x_j. In the matrix polynomial case, we have much more freedom in the transformations that are used. At the very least, we may apply a block diagonal similarity transformation $(\mathbf{D}_s^{-1} \otimes \mathbf{I}_m)\mathbf{A}(\mathbf{D}_s \otimes \mathbf{I}_m)$ to the matrix, where

$$\mathbf{D}_s = \begin{bmatrix} 1 & & & \\ & \sqrt{|w_0|/\|\mathbf{F}_0\|_2} & & \\ & & \ddots & \\ & & & \sqrt{|w_n|/\|\mathbf{F}_n\|_2} \end{bmatrix}. \tag{4.67}$$

This transformation equalizes the row and column norms of the matrix pair. Additionally, because the leading $m \times m$ submatrix of $\mathbf{B}$ is zero, we may arbitrarily scale the first m rows and columns.

It is not entirely clear what transformations would be best to use on the

matrix. For example, we could equalize all of the rows and columns of the matrix $\mathbf{A}$, rather than the proposed block-wise balancing. However, it not clear that this would lead to optimal results, and perhaps retaining the structure of the matrix is, in some sense, desirable. In this work, we restrict the balancing to the proposed block-wise balancing, and pursue a more rigorous theory of balancing in the future.

4.5 Numerical Examples

In this section, we illustrate the backward error of computing eigenpairs of $\mathbf{P}(z)$ via the linearization $(\mathbf{A}, \mathbf{B})$ through some examples taken from the collection NLEVP [3]. For these examples, we do not know the true eigenvalues or eigenvectors a priori, but we are able to compare them approximately to the results of other computations in the literature.

In all of the following examples, the matrix polynomials are sampled at a set of $n + 1$ interpolation nodes, where n is the degree of the matrix polynomial. The matrix pair $(\mathbf{A}, \mathbf{B})$ is constructed from the samples and computed barycentric weights, and then the pair is balanced using the block-wise balancing proposed in §4.4.

The eigenpairs of $(\mathbf{A}, \mathbf{B})$ are computed via the QZ algorithm in MATLAB, using the function `qz`. We then compute the backward errors for the left and right eigenpairs of $\mathbf{P}(z)$ using expressions (4.21) and (4.20), as well as the upper bounds for the backward error given in (4.33) and in (4.35).

Furthermore, we compute the pessimism index of the error bounds, that is, the log (base 10) of the ratios of the backward errors $\eta_P(\lambda, \mathbf{x})$ and $\eta_P(\lambda, \mathbf{y}^*)$ to the backward error bounds in (4.33) and in (4.35), respectively.

4.5.1 Butterfly

Our first example is a 64×64 quartic matrix polynomial with T-even structure, proposed in [14]. The spectrum has a butterfly shape. In the monomial basis,

the matrix polynomial is

$$\mathbf{P}(z) = z^4\mathbf{A}_4 + z^3\mathbf{A}_3 + z^2\mathbf{A}_2 + z\mathbf{A}_1 + \mathbf{A}_0 \,, \tag{4.68}$$

where $\mathbf{A}_4$, $\mathbf{A}_2$, and $\mathbf{A}_0$ are real symmetric matrices. The matrices $\mathbf{A}_3$ and $\mathbf{A}_1$ are real skew-symmetric.

We sample the matrix polynomial at 5 Chebyshev points of the second kind, on the interval $[-1, 1]$. The computed eigenvalues are shown in Figure 4.1. These show good visual agreement to the eigenvalues computed in [14]. The distribution of the backward errors $\eta_P(\lambda, \mathbf{y}^*)$ and $\eta_P(\lambda, \mathbf{x})$ are shown in

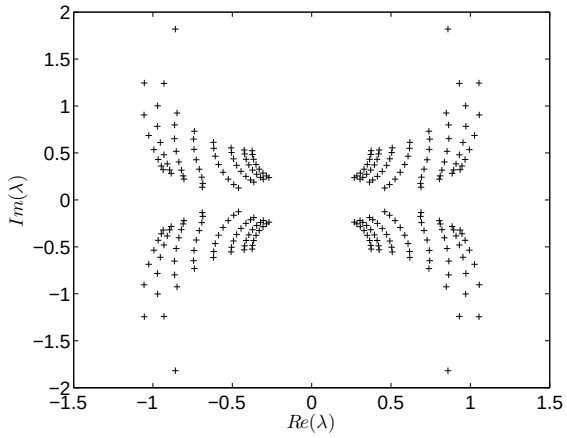
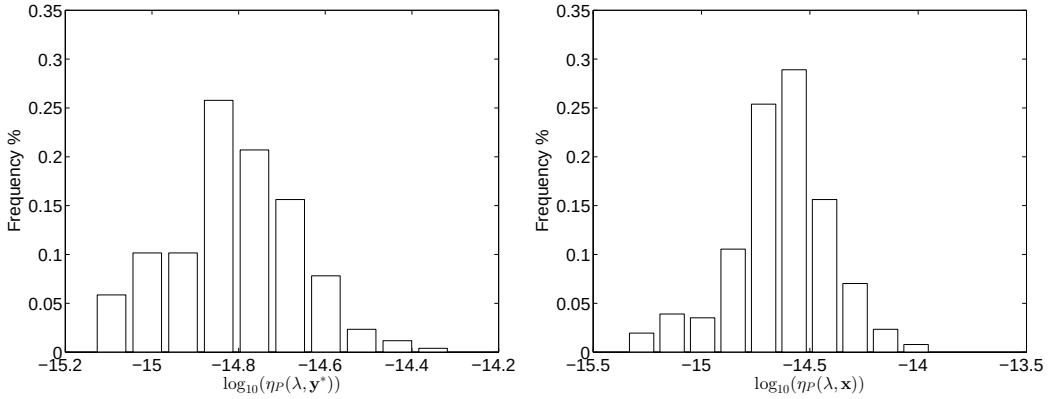


Figure 4.1: Butterfly example, eigenvalue distribution.

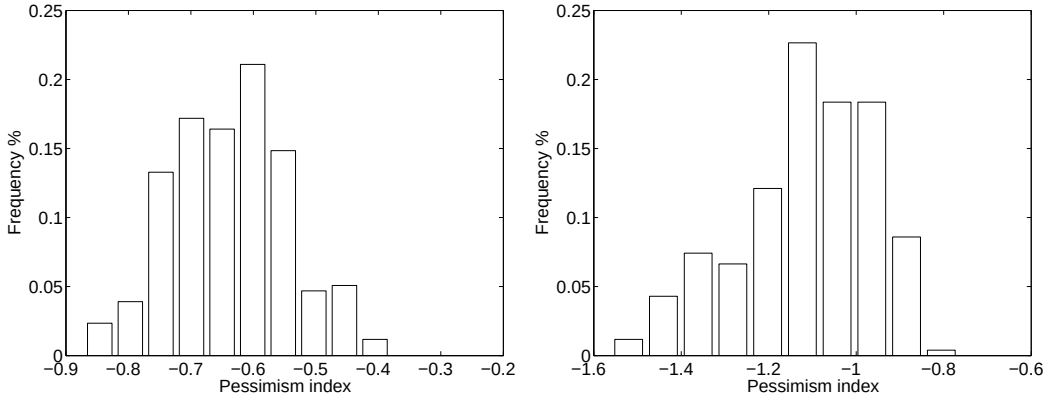
Figure 4.2. The backward errors are excellent, being only a modest multiple of the unit roundoff $u = 2^{-53} \approx 1.1 \times 10^{-16}$.



(a) Left eigenpair (b) Right eigenpair

Figure 4.2: Butterfly example, backward error distributions.

The distribution of the pessimism index is shown in Figure 4.3. For the left eigenpair, the bound exceeds the actual backward error by about 0.6 of an order of magnitude, and for the right eigenpair, a little over one order of magnitude.



(a) Left eigenpair (b) Right eigenpair

Figure 4.3: Butterfly example, pessimism index distributions.

4.5.2 Speaker Enclosure

Our second example is taken from [3]. The matrix polynomial is the quadratic $\mathbf{P}(z) = z^2\mathbf{M} + z\mathbf{C} + \mathbf{K}$, with $\mathbf{M}, \mathbf{C}, \mathbf{K} \in \mathbb{C}^{107\times 107}$, arising from a finite element model of a speaker enclosure. There is a large variation in the norms of the monomial basis coefficients: $\|\mathbf{M}\|_2 = 1$, $\|\mathbf{C}\|_2 = 5.7\times 10^{-2}$, and $\|\mathbf{K}\|_2 = 1\times 10^{7}$.

We interpolate the matrix polynomial at the nodes $\{-i, 0, i\}$. At these nodes $\|\mathbf{F}_j\|_2 \approx 1 \times 10^7$, and so we have already, in a sense, equalized the norms of the coefficients through interpolation. The eigenvalues are shown in Figure 4.4, all of which are purely imaginary. Other choices of interpolations nodes may not give such clean results.

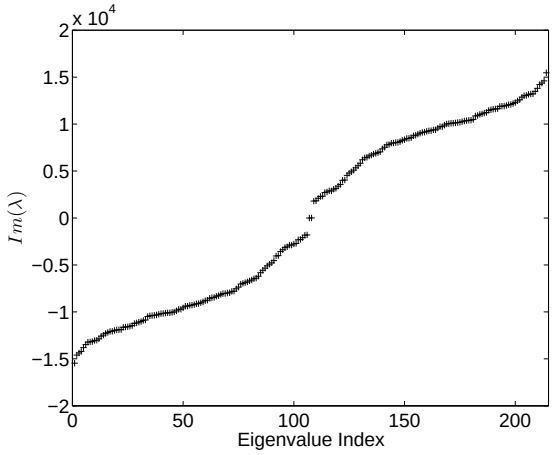
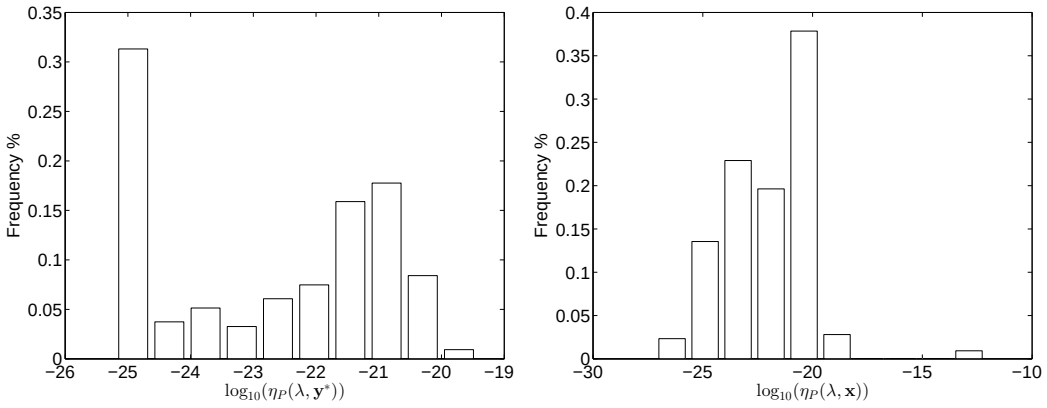


Figure 4.4: Speaker enclosure example, eigenvalue distribution.

Figure 4.5 shows the backward error of the computed eigenpairs of the matrix polynomial. The backward error is truly excellent, being well below the machine precision. We do not show the pessimism index for this problem. The bound overestimates the backward error by at least two orders of magnitude, and thus the backward error is not adequately described by the bound. This may be due to additional structure that the QZ algorithm is able to take advantage of. However, we are not sure exactly what this structure is.



(a) Left eigenpair

(b) Right eigenpair

Figure 4.5: Speaker enclosure example, backward error distributions.

4.5.3 Damped Mass Spring System

Our third example is taken from [10], and also described in [15]. The matrix polynomial is a 100×100 quadratic polynomial $\mathbf{P}(z) = z^2\mathbf{M} + z\mathbf{C} + \mathbf{K}$, where $\mathbf{M} = \mathbf{I}$, $\mathbf{C}$ is tridiagonal with super- and subdiagonal elements all equal to -64 and diagonal elements equal to $128, 192, 192, \ldots, 192, 128$, and $\mathbf{K}$ is tridiagonal with super- and subdiagonal elements all equal to -1 and diagonal elements equal to $2, 3, 3, \ldots, 3, 2$.

The matrix polynomial describes a connected damped mass-spring system. All of the eigenvalues are real and negative, 50 of which range from -320 to -64, and the remaining 50 are all approximately equal to -1.56×10^{-2}.

We interpolate $\mathbf{P}(z)$ at the nodes $\{-320, -150, 0\}$. The eigenvalues of $\mathbf{P}(z)$ are all real, and hence we plot the real part against the index of the eigenvalue, as shown in Figure 4.6. The backward error and pessimism index for the computed eigenpairs are shown in Figures 4.7 and 4.8. We see that, again, the backward errors are very small. Owing to the differing magnitudes of the cluster of eigenvalues near -1.56×10^{-2} and the rest of the spectrum, we see a bimodal distribution of the backward errors, and additionally in the pessimism index. Figure 4.8 illustrates the excellent agreement between the bound and the backward error. For the left eigenpair, the pessimism index of

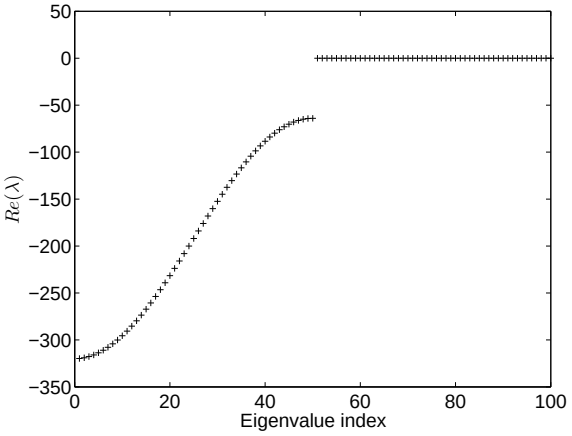
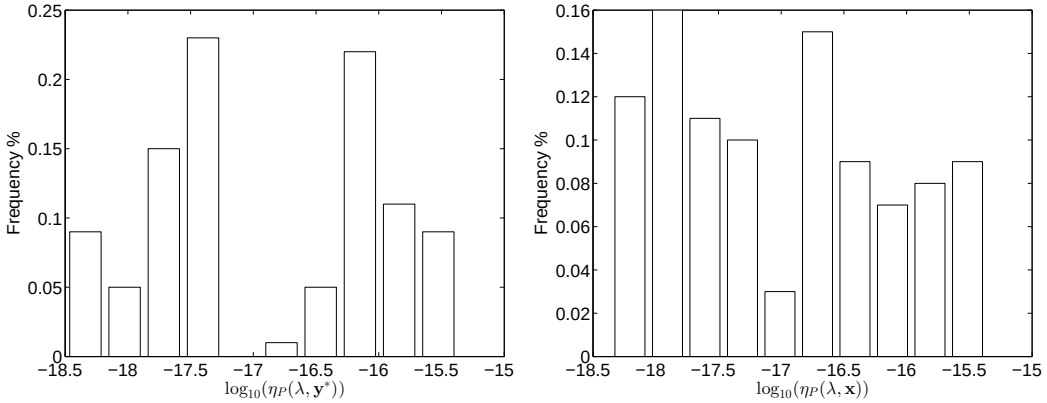


Figure 4.6: Damped mass spring system, eigenvalue distribution.

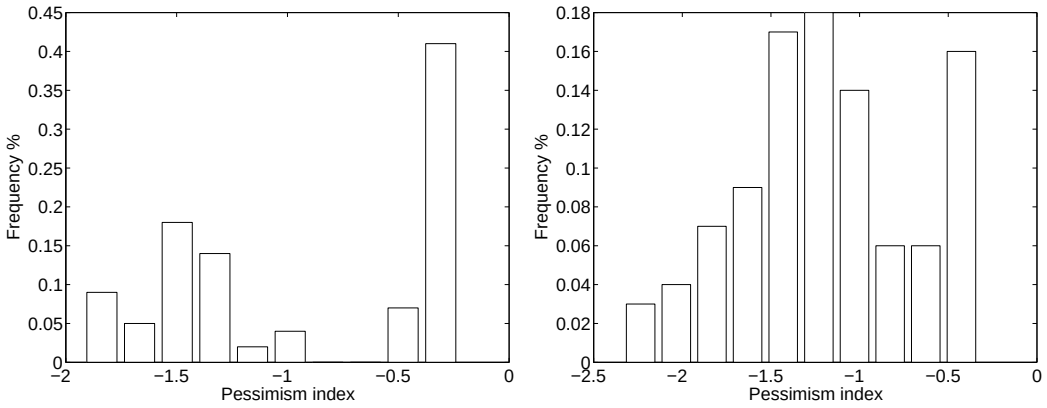
the eigenvalues near -1.56×10^{-2} is close to half an order of magnitude.



(a) Left eigenpair

(b) Right eigenpair

Figure 4.7: Damped mass spring system, backward error distributions.



(a) Left eigenpair (b) Right eigenpair

Figure 4.8: Damped mass spring system, pessimism index distributions.

4.5.4 Damped Gyroscopic system

For our final example, we examine the damped gyroscopic system proposed in [13]. The matrix polynomial is constructed as follows: let $\mathbf{N}$ denote the 10×10 nilpotent matrix having ones on the subdiagonal and zeros elsewhere, and define $\widehat{\mathbf{M}} = (4\mathbf{I}_{10} + \mathbf{N} + \mathbf{N}^T)/6$, $\widehat{\mathbf{G}} = \mathbf{N} - \mathbf{N}^T$, and $\widehat{\mathbf{K}} = \mathbf{N} + \mathbf{N}^T - 2\mathbf{I}_{10}$. Then define the matrices $\mathbf{M}$, $\mathbf{G}$, and $\mathbf{K}$, using the Kronecker product $\otimes$, by

$$\begin{aligned}
\mathbf{M} &= \mathbf{I}_{10} \otimes \widehat{\mathbf{M}} + 1.3\widehat{\mathbf{M}} \otimes \mathbf{I}_{10} \,, \\
\mathbf{G} &= 1.35\mathbf{I}_{10} \otimes \widehat{\mathbf{G}} + 1.1\widehat{\mathbf{G}} \otimes \mathbf{I}_{10} \,, \\
\mathbf{K} &= \mathbf{I}_{10} \otimes \widehat{\mathbf{K}} + 1.2\widehat{\mathbf{K}} \otimes \mathbf{I}_{10} \,,
\end{aligned}$$

and the damping matrix by $\mathbf{D} = \text{tridiag}\{-0.1, 0.2, -0.1\}$. The quadratic matrix polynomial we examine is then defined by

$$\mathbf{P}(z) = z^2\mathbf{M} + z(\mathbf{G} + \mathbf{D}) + \mathbf{K} \,.$$

We interpolate $\mathbf{P}(z)$ at the nodes $\{-1.8, 0, 1.8\}$. In addition to the eigenvalues of $\mathbf{P}(z)$, we also computed the weighted ε-pseudospectrum (see [13]), shown together with the eigenvalues in Figure 4.9. The dotted line represents

where the condition numbers for the Lagrange basis and the monomial basis are equal. Within the dotted line, the condition number of the Lagrange basis is somewhat smaller than that of the monomial basis, and hence we can expect to compute more accurate eigenvalues there.

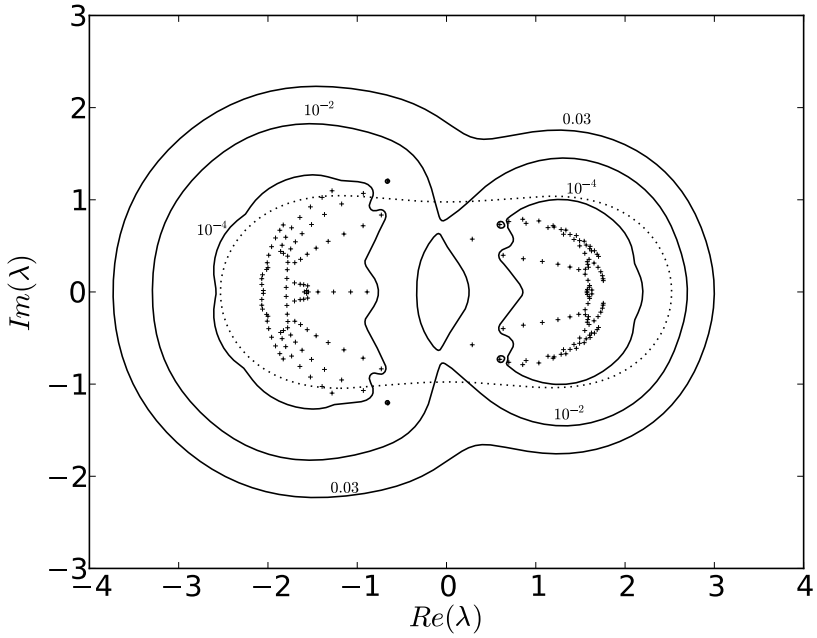


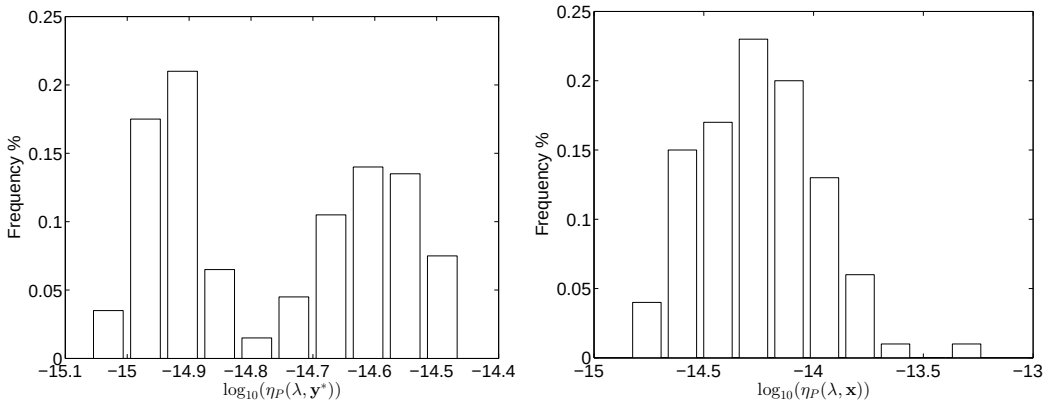
Figure 4.9: Damped gyroscopic system, distributions of eigenvalues and pseudospectra. The dotted line represents the level curve where $B_M(z) = B_L(z)$.

The ε-pseudospectrum in this example gives us an idea of what the conditioning of the problem is like. The outer boundary tells us that if we perturb the coefficients of $\mathbf{P}(z)$ by up to 0.03 in norm, all of the eigenvalues will be contained within the boundary. As we decrease the size of the perturbations, we see that the boundaries get tighter around the computed eigenvalues. In this case, we require fairly large perturbations before the pseudospectral boundaries move a great distance from the eigenvalues, and thus the problem is fairly well conditioned.

Furthermore, because we are able to choose the locations of the nodes,

we are able to ensure that eigenvalues of interest are computed accurately by placing nodes near to the eigenvalues. If nothing is known about the spectrum of $\mathbf{P}(z)$ then we may initially compute the eigenvalues using, for example, Chebyshev nodes on $[-1, 1]$. Then interpolate $\mathbf{P}(z)$ using some of the computed eigenvalues as nodes. This kind of iterative algorithm has been used successfully in the scalar case [8], and we expect to obtain similar results in the matrix polynomial case. For the monomial basis, we have no such flexibility.

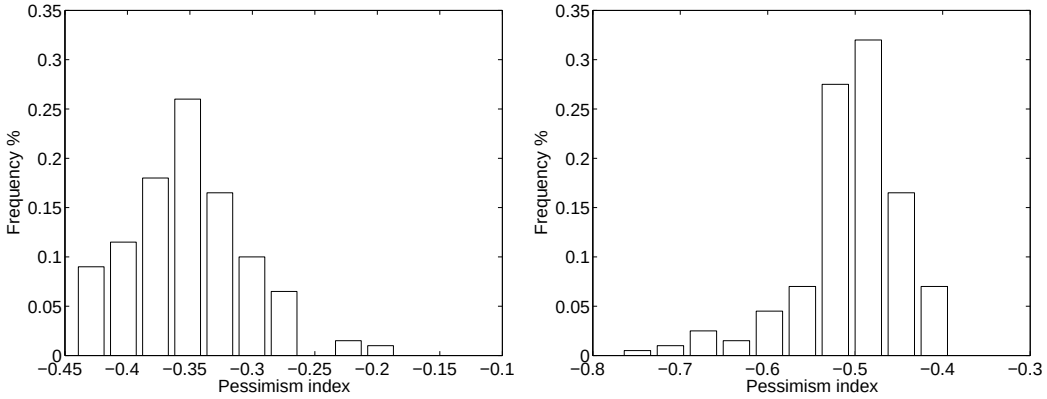
Figures 4.10 and 4.11 show the distributions of the backward error and the pessimism index for the computed eigenpairs. Again, we see that the backward errors are small, and the bound agrees well with the backward error.



(a) Left eigenpair

(b) Right eigenpair

Figure 4.10: Damped gyroscopic system, backward error distributions.



(a) Left eigenpair (b) Right eigenpair

Figure 4.11: Damped gyroscopic system, pessimism index distributions.

4.6 Concluding Remarks

In this chapter, we have derived bounds for the backward error of the eigenvalues of matrix polynomials expressed in the Lagrange basis, computed via the eigenvalues of a generalized companion matrix pair. The backward error will be small, provided that the ratios of the barycentric weights, the norms of the coefficients in the Lagrange basis, and the interpolation nodes are all not too large. We have described a block-wise balancing scheme for the linearization that improves the accuracy of the computed eigenvalues. Numerical experiments show that we obtain small backward errors of the computed eigenvalues.

Bibliography

[1] A. Amiraslani, R. M. Corless, and P. Lancaster, *Linearization of matrix polynomials expressed in polynomial bases*, IMA Journal of Numerical Analysis, 29 (2009), pp. 141–157.

[2] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Review, 46 (2004), pp. 501–517.

[3] T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur, *Nlevp: A collection of nonlinear eigenvalue problems*, ACM Transactions on Mathematical Software (TOMS), 39 (2013), p. 7.

[4] R. M. Corless, *Generalized companion matrices in the Lagrange basis*, in Proceedings EACA, L. Gonzalez-Vega and T. Recio, eds., June 2004, pp. 317–322.

[5] R. M. Corless, N. Rezvani, and A. Amiraslani, *Pseudospectra of matrix polynomials that are expressed in alternative bases*, Mathematics in Computer Science, 1 (2007), pp. 353–374.

[6] C. Effenberger and D. Kressner, *Chebyshev interpolation for nonlinear eigenvalue problems*, BIT Numerical Mathematics, 52 (2012), pp. 933–951.

[7] R. T. Farouki and V. T. Rajan, *On the numerical condition of polynomials in Bernstein form*, Comput. Aided Geom. Des., 4 (1987), pp. 191–216.

[8] S. Fortune, *An iterated eigenvalue algorithm for approximating roots of univariate polynomials*, Journal of Symbolic Computation, 33 (2002), pp. 627–646.

[9] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix polynomials*, SIAM, 2009.

[10] N. Higham, R. Li, and F. Tisseur, *Backward error of polynomial eigenproblems solved by linearization*, SIAM Journal on Matrix Analysis and Applications, 29 (2008), pp. 1218–1241.

[11] N. J. Higham, *The numerical stability of barycentric Lagrange interpolation*, IMA Journal of Numerical Analysis, 24 (2004), pp. 547–556.

[12] P. Lancaster, *Linearization of regular matrix polynomials*, Electron. J. Linear Algebra, 17 (2008), pp. 21–27.

[13] P. Lancaster and P. Psarrakos, *On the pseudospectra of matrix polynomials*, SIAM journal on matrix analysis and applications, 27 (2005), pp. 115–129.

[14] V. Mehrmann and D. S. Watkins, *Polynomial eigenvalue problems with Hamiltonian structure*, Electr. Trans. Num. Anal, 13 (2002), pp. 106–113.

[15] F. Tisseur, *Backward error and condition of polynomial eigenvalue problems*, Linear Algebra and its Applications, 309 (2000), pp. 339–361.

[16] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Review, 43 (2001), pp. 235–286.

[17] A. Townsend, V. Noferini, and Y. Nakatsukasa, *Vector spaces of linearizations for matrix polynomials: A bivariate polynomial approach*. (Submitted).

[18] D. Watkins, *The matrix eigenvalue problem: GR and Krylov subspace methods*, SIAM, 2007.

Chapter 5

Concluding Remarks

In this work, we have been motivated by the fact that computations in the Lagrange basis are often better conditioned than those in the monomial basis. Conversion between polynomial bases is often ill-conditioned, and this motivates us to work entirely in the Lagrange basis, that is, directly with the values of a polynomial interpolant.

In the second chapter, we introduced two fast algorithms which reduce the linearization of a polynomial to a rank one perturbation of a symmetric tridiagonal matrix. This reduction process can be carried out in $O(n^2)$ operations, significantly lowering the cost compared to the standard reduction algorithms. The proposed reduction processes also lower the storage requirements to $O(n)$, rather than the $O(n^2)$ required for the standard reduction, and thus bring us closer to an algorithm requiring only $O(n)$ storage, and $O(n^2)$ cost for computing all of the roots of polynomials.

Since the Lagrange basis is not a degree graded basis, the question of determining the degree of the polynomial was also addressed. If some of the leading coefficients of the interpolating polynomial are zero, then the degree of the polynomial may be determined through simple expressions requiring only $O(mn)$ complexity. Furthermore, if the leading coefficients are zero, then the linearization will have extra infinite eigenvalues in addition to the two spurious ones introduced by the linearization. We have shown that these, too, may be deflated easily from the linearization.

Through numerical experimentation, we tested the accuracy of the reduc-

tion processes. The proposed reduction processes produce accurate Hessenberg matrices, and outperform both the standard reduction process, and the quasiseparable algorithm proposed in [1].

In the third and fourth chapters, we discussed the issue of numerical stability of computing the roots of polynomials and the eigenvalues of matrix polynomials through linearization. We established that the roots computed in this manner are normwise backward stable. Additionally, the balancing strategy proposed ensures that the roots are computed with small backward errors. We formulated upper bounds for the backward errors. For a range of examples, the computations produced roots and eigenvalues with small backward errors, and the bounds were only a small factor larger than the backward errors.

The linearization introduces two spurious infinite eigenvalues to the formulation, and we showed how these eigenvalues could be deflated exactly from the matrix pair, without affecting the accuracy of the finite (or infinite) eigenvalues. This same procedure was also extended to the matrix polynomial case, where the formulation introduces $2m$ spurious infinite eigenvalues. These infinite eigenvalues may also be deflated exactly, and (if the transformation is well conditioned) the generalized eigenvalue problem transformed to a standard eigenvalue problem.

5.1 Future work

As was shown in Chapter 4, the strategy for balancing the linearization produced eigenvalues of matrix polynomials with small backward errors. We believe that there is more work to be done to obtain an optimal balancing strategy similar to that proposed in [3]. It is my intention to investigate the problem of balancing more thoroughly in the near future.

Another consideration worth noting is the matter of the influence of the geometry of the interpolation nodes on the accuracy of the computed eigenvalues. As shown in the pseudospectral plot in Figure 4.9 and the discussion thereafter, we may adapt the interpolation nodes to ensure that the eigen-

values nearby are computed with small backward errors. If prior knowledge of the spectrum is known, this can be utilized to good effect. If not, we may iteratively compute the eigenvalues, adjusting the location of the interpolation nodes to ensure accurate computation of the eigenvalues near to the nodes.

The barycentric form can be extended to Hermite interpolational bases, and linearizations of these polynomials exist. Some work has been done already to extend the work presented here to these bases [2], and we believe that even more elegant expressions than the one presented in that extended abstract may be formed. Unfortunately it appears that the fast reduction techniques do not extend to the linearization of Hermite interpolants, further work is necessary to determine what structure might be retained in a reduction process.

Another natural extension of this work is to multivariate polynomial rootfinding. For example, the software package CHEBFUN [4] has recently been extended to two dimensions, and there is a need for accurate and stable rootfinding methods within this framework. Of course, much work is left to be done to generalize the work presented here to higher dimensions.

For matrix polynomials with structured coefficients, the linearization does not preserve the structure of the coefficients. The deflation process proposed in §4.3 may afford some insight into how structure might be preserved in the solution process. In that deflation process, block wise operations were used, and these operations maintain the structure of the original coefficients. It may be possible to maintain some structure in the coefficients by using block operations in the subsequent QZ (or QR) iterations. The extent to which this may or may not be possible is a matter for future investigation.

Another future direction of research is to combine the fast reduction techniques proposed in Chapter 2 with the fast QR algorithms for tridiagonal plus rank one matrices proposed in [5]. Hence, we would obtain an $O(n^2)$ algorithm for computing all of the roots of scalar polynomials.

It may also be worthwhile investigating more efficient computation of pseudospectra for matrix polynomials, combining the structured reduction of the linearization to obtain a structure which might be more amenable to grid computations.

Bibliography

[1] Y. Eidelman, I. Gohberg, and L. Gemignani, *On the fast reduction of a quasiseparable matrix to Hessenberg and tridiagonal forms*, Linear Algebra and its Applications, 420 (2007), pp. 86–101.

[2] P. W. Lawrence and R. M. Corless, *Numerical stability of barycentric Hermite root-finding*, in Proceedings of the 2011 International Workshop on Symbolic-Numeric Computation, ACM, 2012, pp. 147–148.

[3] D. Lemonnier and P. Van Dooren, *Optimal scaling of block companion pencils*, in International Symposium on Mathematical Theory of Networks and Systems, Leuven, Belgium, 2004.

[4] L. N. Trefethen et al., *Chebfun Version 4.2*, The Chebfun Development Team, 2011. `http://www.maths.ox.ac.uk/chebfun/`.

[5] M. Van Barel, R. Vandebril, P. Van Dooren, and K. Frederix, *Implicit double shift QR-algorithm for companion matrices*, Numerische Mathematik, 116 (2010), pp. 177–212.

Appendix A

Algorithms for Fast Reduction to Hessenberg form

Algorithm 1 Reduction of $\mathbf{A}$ to symmetric tridiagonal plus rank-one form.

$\mathbf{q}_0 \leftarrow \mathbf{w}$
$t_0 \leftarrow \|\mathbf{q}_0\|_2$
$\mathbf{q}_0 \leftarrow \mathbf{q}_0/t_0$
$d_0 \leftarrow \mathbf{q}_0^* \mathbf{D} \mathbf{q}_0$
$\mathbf{q}_1 \leftarrow (\mathbf{D} - d_0 \mathbf{I})\, \mathbf{q}_0$
$t_1 \leftarrow \|\mathbf{q}_1\|_2$
$\mathbf{q}_1 \leftarrow \mathbf{q}_1/t_1$
for $i = 1$ to $n-1$ **do**
  $d_i \leftarrow \mathbf{q}_i^* \mathbf{D} \mathbf{q}_i$
  $\mathbf{q}_{i+1} \leftarrow (\mathbf{D} - d_i \mathbf{I})\, \mathbf{q}_i - t_i \mathbf{q}_{i-1}$
  $t_{i+1} \leftarrow \|\mathbf{q}_{i+1}\|_2$
  $\mathbf{q}_{i+1} \leftarrow \mathbf{q}_{i+1}/t_{i+1}$
end for
$d_n \leftarrow \mathbf{q}_n^* \mathbf{D} \mathbf{q}_n$
$\mathbf{c}^T \leftarrow \begin{bmatrix} 0 & -\mathbf{f}^T \mathbf{Q}_1 - t_0 \mathbf{e}_1^T \end{bmatrix}$
return $\mathbf{t}$, $\mathbf{d}$, $\mathbf{c}$

Algorithm 2 Reduction of $\mathbf{A}$ via Givens rotations.

$\mathbf{t} \leftarrow \mathbf{0}_{(n-1)\times 1}$
for $i = n$ to 1 **do**
  Compute Givens rotation such that

$$\begin{bmatrix} w_{i-1} \\ w_i \end{bmatrix} \leftarrow \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^* \begin{bmatrix} w_{i-1} \\ w_i \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

  Update remainder of matrix:

$$\begin{bmatrix} f_{i-1} & f_i \end{bmatrix} \leftarrow \begin{bmatrix} f_{i-1} & f_i \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$
$$\begin{bmatrix} x_{i-1} & t_i \\ t_i & x_i \end{bmatrix} \leftarrow \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^* \begin{bmatrix} x_{i-1} & 0 \\ 0 & x_i \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

  if $i < n$ **then**
    $b \leftarrow s \cdot t_{i-1}$
    $t_{i+1} \leftarrow c \cdot t_{i+1}$
    for $j = i$ to $n - 1$ **do**
      Compute Givens rotation such that:

$$\begin{bmatrix} t_j \\ 0 \end{bmatrix} \leftarrow \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^* \begin{bmatrix} t_j \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

      Update remainder of matrix:

$$\begin{bmatrix} f_j & f_{j+1} \end{bmatrix} \leftarrow \begin{bmatrix} f_j & f_{j+1} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$
$$\begin{bmatrix} x_j & t_{j+1} \\ t_{j+1} & x_{j+1} \end{bmatrix} \leftarrow \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^* \begin{bmatrix} x_j & t_{j+1} \\ t_{j+1} & x_{j+1} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

      if $j < n - 1$ **then**
        $b \leftarrow s \cdot t_{j+2}$
        $t_{j+2} \leftarrow c \cdot t_{j+2}$
      end if
    end for
  end if
end for
$\mathbf{c} = \begin{bmatrix} 0 & \mathbf{f}^T \end{bmatrix}$
$\mathbf{d} = \begin{bmatrix} 0 & \mathbf{x} \end{bmatrix}$
return $\mathbf{t}$, $\mathbf{d}$, $\mathbf{c}$

Piers W. Lawrence

Education

The University of Western Ontario **2009 - Present**
PhD candidate in Applied Mathematics supervised by Rob Corless.
Thesis title: *"Eigenvalue Methods for Interpolation Bases"*

University of Canterbury **2005 - 2008**
BSc Hons – Mathematics (First class).

Academic Awards

Western Graduate Research Scholarship: *"Eigenvalue Methods for Interpolation Bases"* (2009-Present).

University of Canterbury Summer Scholarship: *"Nonlinear timeseries with measurable double pendulum"* (2008-2009).

University of Canterbury Mathematics and Statistics Department Scholarship (2008).

HRC scholarship, funded from Sir Charles Hercus Health Research Fellowship: *"Parameter estimation techniques for insulin-glucose models"* (2008).

Australian National University Summer Research Scholarship (2007-2008).

HRC scholarship, funded from Sir Charles Hercus Health Research Fellowship: *"Geometric theory and application of integral based parameter identification"* (2007).

University of Canterbury Summer Scholarship: *"Fast algorithms for glucose control protocol testing"* (2006-2007).

Publications

Robert M. Corless and **Piers W. Lawrence**, "The largest roots of the Mandelbrot polynomials." In D. Bailey, H.H. Bauschke, P. Borwein, F. Garvan, M. Thera, J. Vanderwerff and H. Wolkowicz, editors, *Computational and Analytical Mathematics*, Springer Proceedings in Mathematics and Statistics, 2012 (in press).

P.W. Lawrence, R.M. Corless, and D.J. Jeffrey, "Complex Double-Precision Evaluation of the Wright ω Function." *ACM Transactions on Mathematical Software*. **38**(3) 2012.

Piers W. Lawrence and Robert M. Corless, "Numerical Stability of Barycentric Hermite Rootfinding." In Marc Moreno Maza, editor, *Proceedings of the 2011 Internation Workshop on Symbolic-Numeric Computation*, ACM, pp. 147–148, 2011.

Piers Lawrence, Michael Stuart, Richard Brown, Warwick Tucker and Raazesh Sainudiin, "A Mechatronically Measurable Double Pendulum for Machine Interval Experiments." Indian Statistical Institute Technical Report, isibang/ms/2010/11, October 25, 2010.

C.E. Hann, J.G. Chase, M.F. Ypma, J. Elfring, N.H.M. Nor, **P.W. Lawrence** and G.M. Shaw, "The Impact of Parameter Identification Methods on Drug Therapy Control in an Intensive Care Unit." *The Open Medical Informatics Journal*, **2**, pp. 92-104, 2008.

Submitted Publications

P.W. Lawrence, "Fast Reduction of Generalized Companion Matrix Pairs for Barycentric Lagrange Interpolation."

P.W. Lawrence and R.M. Corless, "Stability of Rootfinding for Barycentric Lagrange Interpolants."

P.W. Lawrence and R.M. Corless, "Stability of Matrix Polynomial Rootfinding."

Conference Presentations

"Rootfinding for Polynomial and Rational Interpolants," presented at the 3rd Dolomites Workshop on Constructive Approximation and Applications, Alba di Canazei (Trento, Italy), 2012.

"Eigenstructure of an Arrowhead Matrix Pencil," presented at ICIAM 2011, Vancouver, Canada.

"Numerical Stability of Barycentric Hermite Rootfinding," presented at Symbolic Numeric Computation 2011, San Jose, USA.

"On the Stability of Barycentric Hermite Rootfinding," presented at Southern Ontario Numerical Analysis Day 2011, McMaster University, Hamilton, Canada.

"Event Handling with Barycentric Hermite Interpolation," presented at CAIMS*SCMAI 2010, St John's, Canada.

Posters

"Mandelbrot Polynomials and Matrices" presented at the East Coast Computer Algebra Day (ECCAD) 2012, Oakland University, Rochester, Michigan, USA.

"Computing Roots of Mandelbrot Polynomials" presented at the East Coast Computer Algebra Day (ECCAD) 2011, The University of Waterloo, Waterloo, Canada.

Other Scholarly Activities

Visiting scholar: "Rootfinding with linear barycentric rational interpolation", hosted by Professor Jean-Paul Berrut and Georges Klein at the Universtiy of Fribourg, February 6-15, 2012.

Attended Dobbiaco summer school: "Approximation Theory, Spectral Methods and Chebfun" June 12-17,2011.

Research Experience

Research Assistant — 2009 - Present
Rob Corless (PhD supervisor) — The University of Western Ontario
Implementing algorithms for evaluation and rootfinding of Birkhoff and Hermite interpolants expressed by values. Implementing robust event location for differential algebraic equations in the software package DAETS. Implementing algorithms to deflate infinite eigenvalues from companion matrices associated with barycentric Hermite interpolants. Implementing algorithms to locate k-periodic points in the Mandelbrot set.

Research Assistant — 2008-2009
Raazesh Sainudiin — University of Canterbury
Designing and fabricating a measurable double pendulum apparatus for statistical experiments. Implementing logging software for capturing data from the device.

Research Assistant — 2008
Chris Hann and Geoff Chase — University of Canterbury
Investigating novel methods for parameter estimation of differential equation models. With particular applications in estimating insulin sensitivity from clinical data obtained in Intravenous Glucose Tolerance Testing in humans.

Research Assistant — 2007
Jason Wong — University of Canterbury
Investigation of linear compartmental insulin glucose model for Diabetic patients for the effective dosing of insulin. Development of fast algorithms for forward simulation and parameter reconstruction.

Research Assistant — 2007
Stephen Roberts — Australian National University

Implementation of visualisation tools for ANUGA shallow water wave equation modelling software. With particular application to a case study of water gauges in Lake Merimbula.

Research Assistant 2006
Chris Hann University of Canterbury
Investigation of non linear insulin-glucose models. Development of fast algorithms for model based therapeutics.

Computer Skills

Proficient in: MATLAB, LaTeX, Maple, Python, C/C++ (with skills in parallel programming with OpenMP and MPI), FORTRAN, bash, emacs lisp, HTML.

Teaching Experience

The University of Western Ontario London, Canada
Teaching Assistant 2009 - Present
Third year partial differential equations, second year numerical analysis and graduate numerical analysis.

University of Canterbury Christchurch, New Zealand
Teaching Assistant 2007-2008
First and second year computational mathematics and third year partial differential equations.