THE RAYLEIGH QUOTIENT ITERATION FOR GENERALIZED COMPANION MATRIX PENCILS*

A. Amiraslani[a,1] , D. A. Aruliah[b], and Robert M. Corless[c]

[a] *Department of Mathematics and Statistics, University of Calgary*[2]
Calgary, AB T2N 1N4, Canada (Supported by NSERC grant 12345)
amiram@math.ucalgary.ca

[b] *Faculty of Science, University of Ontario Institute of Technology*
Oshawa, ON L1H 7K4, Canada (Supported by NSERC grant 12345)
Dhavide.Aruliah@uoit.ca

[c] *Ontario Research Centre for Computer Algebra and Department of Applied Mathematics, University of Western Ontario*
London, ON N6A 5B7, Canada
rcorless@uwo.ca

Summary

Experimental observations of univariate rootfinding by generalised companion matrix pencils expressed in the Lagrange basis show that the method can often give accurate answers. This current paper, motivated in part by analogy with the Jenkins-Traub method for polynomial rootfinding, applies a version of the Rayleigh quotient iteration to these generalised companion matrix pencils for efficiency and explores the results for polynomial eigenvalue problems. The particular choice of initial vectors that we use, parameterised by a scalar initial guess for the eigenvalue, guarantees that convergence occurs for almost all initial guesses.

keywords: numerical linear algebra; Rayleigh quotients; matrix polynomials; Lagrange basis.

1. Introduction. There is a thread of recent research into polynomial computation using bases other than the standard monomial power basis. The investigations in [1, 2, 3, 4] specifically focus on computation with polynomials expressed in the Lagrange basis, or, in other words, polynomial computation directly by values. The related works [5, 6] also use the Lagrange basis as an intermediate step in the computation and analysis of polynomial roots.

The motivation for this interest in polynomial computation using alternatives to the power basis is that *conversion between bases can be unstable*. Moreover, the instability increases with the degree [7]. The complications arising from such computations motivate explorations of hybrid symbolic-numeric techniques for polynomial computation that are relevant to researchers interested in computer algebra and numerical analysis.

Recent papers by Berrut & Trefethen [8] and Higham [9] show that working directly in the Lagrange basis is both numerically stable and efficient, more stable and

*This work was supported in part by NSERC

[1]Correspondence to: Amirhossein Amiraslani, Department of Mathematics and Statistics, University of Calgary, 2500 University Dr. NW, Calgary, AB T2N 1N4, Canada

[2]was a post-doctoral fellow at University of Ontario Institute of technology while working on this manuscript.

efficient than had been heretofore credited widely in the numerical analysis community. These recent results strengthen the motivation for examining algorithms for direct manipulation of polynomials by values.

There is also a large body of work on matrix polynomials and their spectra (called polynomial eigenvalues or latent roots or nonlinear eigenvalues in the literature). The classic work [10] gives much of the theory and many applications; recent work includes [11], [12] and [13].

In the present work, we examine the problem of computing polynomial eigenvalues coming from matrix polynomials given by their (matrix) values at certain nodes; that is, the matrix polynomial is assumed to be expressed in the Lagrange basis. The results of this paper complement those of [3] for the case of simple, finite generalised eigenvalues of the scalar companion matrix pencil arising from a polynomial expressed in the Lagrange basis. This turns out to be related to the so-called *secular equation* [14].

Without efficient and stable methods for computing generalised eigenvalues of matrix pencils, this new companion matrix pencil would be a curiosity only. The standard QZ iteration works well, when the companion matrix pencil is well-balanced, but takes $O(n^2)$ storage and $O(n^3)$ time, and one hopes that it would be possible to do better. Motivated by the Jenkins-Traub iteration, which is equivalent to the application of the Rayleigh quotient iteration to the standard (Frobenius form) companion matrix [15], we search for a faster method, essentially undoing the linearization given by the generalised companion matrix pencil. See, for comparison, the classic work [16].

Our ultimate aim is to use an as-yet undiscovered variant of this approach to determine roots of multivariate polynomial systems. Multiplication matrices associated with resultants or Gröbner bases give multivariate polynomial roots using eigenvalue computations. As such, Rayleigh quotient iteration can be interpreted as matrix polynomial evaluation together with small matrix-vector products in the univariate case; we expect this might also work in the multivariate case.

A very recent advance of Bini, Gemignani and Pan [17] shows great promise using a structure-preserving (and thereby fast) variant of the **QR** algorithm to determine eigenvalues of matrices of similar structure to the matrix pencils studied in the current work. Although the work of [17] has as yet neither been extended to matrix polynomials nor been extended to the present companion matrix pencils, there is no reason to doubt that it could be so extended. Whether this approach could be extended to multivariate polynomial systems is less clear.

In this present paper we pursue some specific numerical methods, which, even if the structured **QR** algorithm of [17] can be used in general, may be used to find a few selected polynomial eigenvalues in an efficient manner.

1.1. Potential Applications. The first application that we encountered was a simple scalar rootfinding problem, where the polynomials were given by values. The problem arose in the numerical continuation solution of the nonlinear equations $F(x, \lambda) = 0$ via a predictor-corrector method, where the algorithm for adapting the step-size used interpolation of the numerical solution at different λ-values, parame-

terised by arclength. For certain stiff problems, warning messages were being generated because the conversion to monomial basis was suffering because the sample points were close together, resulting in an ill-conditioned Vandermonde matrix. We believe that the approach of this paper will be more robust and less susceptible to such ill-conditioning (and at the same time, more efficient).

Common matrix polynomial eigenproblems such as vibration problems are usually expressed most naturally in the monomial basis; moreover, typically $s \gg n$ (n is often 2), and so in those cases we do not expect the methods of this paper to be useful.

Other applications are speculative at this point. We expect that there are circumstances where matrix functions are known at sampled times (or have been transformed by the FFT to a situation where the matrix functions are so known), perhaps because they arise by discrete dynamical systems, or by numerical methods applied to continuous systems. It has been observed that rootfinding for scalar analytic functions can be carried out by first approximating the function by its sampled values [3].

We also have some hope that this new family of linearizations of matrix polynomials may have some application to the inverse eigenvalue problem: after all, we have $n + 1$ more parameters to play with (the nodes).

1.2. Outline of the Paper. We begin, in Section 2, by reviewing the standard Rayleigh Quotient Iteration (RQI) and one of its variants for generalised eigenvalue problems. We provide in Section 3 a summary of the polynomial eigenvalue problem and its relation to a block companion matrix as represented in the monomial basis. We introduce in Section 4 the generalised companion matrix pencil relating to the polynomial eigenvalue problem expressed in the Lagrange basis with formulas for the corresponding eigenvectors. In Section 5, we derive two algorithms based on Rayleigh quotients for the computation of eigenvalues and eigenvectors of these generalised companion matrix pencils, that uses an efficient **LU** factorisation of the companion pencil and a procedure for deflation. In Section 6, we give numerical experiments based on implementations of these algorithms, and we present conclusions in Section 7.

Throughout the present work, boldface letters are used to denote vectors and matrices. The superscript $()^H$ denotes the Hermitian (complex-conjugate) transpose of a matrix or vector while the superscript $()^T$ denotes the transpose *without* complex conjugation. Subscripts enclosed in parentheses (e.g., $\lambda_{(k)}$) denote iterates within an iterative algorithm.

2. Rayleigh Quotient Iteration and its variants. The Rayleigh quotient iteration (RQI, [18]) is a well-known iterative method used to determine the eigenvalues of a matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$. Starting with a normalised putative eigenvector $\mathbf{x}_{(0)} \in \mathbb{C}^{N\times 1}$, a sequence of normalised approximate eigenvectors $\{\mathbf{x}_{(k)}\}_{k=0}^{\infty}$ is generated with their associated Rayleigh quotients $\{\lambda_{(k)}\}_{k=0}^{\infty} = \{\mathbf{x}_{(k)}^H \mathbf{A}\mathbf{x}_{(k)}\}_{k=0}^{\infty}$ as shown in Algorithm 2.1. The Rayleigh quotient iteration is well-known to be locally cubically convergent given sufficiently accurate initial data [19, 20], i.e., the sequence $\{\lambda_{(k)}\}_{k=0}^{\infty}$ converges to some eigenvalue λ^* of $\mathbf{A}$ with the vectors $\{\mathbf{x}_{(k)}\}_{k=0}^{\infty}$ converging to the corresponding eigenvector $\mathbf{x}^*$

Algorithm 2.1 (Rayleigh Quotient Iteration).

Input: $\mathbf{A} \in \mathbb{C}^{N\times N}$, $\boldsymbol{\xi}_{(0)} \in \mathbb{C}^{N\times 1}$

for $k = 0, 1, 2, \ldots$
Normalise $\mathbf{x}_{(k)} \leftarrow \|\boldsymbol{\xi}_{(k)}\|_2^{-1}\boldsymbol{\xi}_{(k)}$
Compute $\lambda_{(k)} \leftarrow \mathbf{x}_{(k)}^H \mathbf{A}\mathbf{x}_{(k)}$
Solve $\left[\lambda_{(k)}\mathbf{I} - \mathbf{A}\right]\boldsymbol{\xi}_{(k+1)} = \mathbf{x}_{(k)}$ for $\boldsymbol{\xi}_{(k+1)}$
end for

Variants of Rayleigh quotient iteration include Ostrowski's two-sided RQI [21] for nonsymmetric eigenproblems, Parlett's alternating RQI [22], O'Leary and Stewart's singular-value RQI [23], and Schwetlick and Lösche's EMGRE algorithm [24].

A generalisation of Rayleigh quotient iteration for computing generalised eigenvalues of a matrix pencil $\lambda\mathbf{B} - \mathbf{A}$ is given in Algorithm 2.2 [25, 16]. Within this generalised Rayleigh quotient iteration, given good starting guesses, the sequence $\{\lambda_{(k)}\}_{k=0}^{\infty}$ converges to a generalised eigenvalue λ of the matrix pencil $\lambda\mathbf{B} - \mathbf{A}$ and the sequence of vectors $\{(\mathbf{x}_{(k)}, \mathbf{y}_{(k)}^H)\}_{k=0}^{\infty}$ converge to the corresponding right and left eigenvectors $(\mathbf{x}, \mathbf{y}^H)$.

ALGORITHM 2.2 (Generalised Rayleigh Quotient Iteration).
Input: $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{N\times N}$, $\boldsymbol{\xi}_{(0)}, \boldsymbol{\eta}_{(0)} \in \mathbb{C}^{N\times 1}$
for $k = 0, 1, 2, \ldots$
Normalise $\mathbf{x}_{(k)} \leftarrow \|\boldsymbol{\xi}_{(k)}\|_2^{-1}\boldsymbol{\xi}_{(k)}$
Normalise $\mathbf{y}_{(k)} \leftarrow \|\boldsymbol{\eta}_{(k)}\|_2^{-1}\boldsymbol{\eta}_{(k)}$
Compute $\lambda_{(k)} \leftarrow \mathbf{y}_{(k)}^H \mathbf{A}\mathbf{x}_{(k)}[\mathbf{y}_{(k)}^H \mathbf{B}\mathbf{x}_{(k)}]^{-1}$
Solve $\left[\lambda_{(k)}\mathbf{B} - \mathbf{A}\right]\boldsymbol{\xi}_{(k+1)} = \mathbf{B}\mathbf{x}_{(k)}$ for $\boldsymbol{\xi}_{(k+1)}$
Solve $\left[\lambda_{(k)}\mathbf{B} - \mathbf{A}\right]^H\boldsymbol{\eta}_{(k+1)} = \mathbf{B}^H\mathbf{y}_{(k)}$ for $\boldsymbol{\eta}_{(k+1)}$
end for

3. Matrix Polynomials and Polynomial Eigenvalue Problems. A *matrix polynomial* $\mathbf{P}(z)$ is a polynomial function in a scalar argument $z \in \mathbb{C}$ with $s \times s$ matrix coefficients [10, 26]. Conventionally, $\mathbf{P}(z)$ is expressed relative to a *monomial* or *power basis* $\{1, z, \ldots, z^n\}$, i.e.,

$$\mathbf{P}(z) = \sum_{k=0}^{n} z^k \mathbf{A}_k = \mathbf{A}_0 + z\mathbf{A}_1 + \cdots + z^n\mathbf{A}_n$$

where $\{\mathbf{A}_k\}_{k=0}^n \subset \mathbb{C}^{s\times s}$ are matrix coefficients and $z \in \mathbb{C}$. The theory of matrix polynomials is described extensively in [10].

Given a matrix polynomial $\mathbf{P}(z)$, the *polynomial eigenvalue problem* (also known as the *nonlinear eigenvalue problem*) is as follows (see [19, 27, 10, 28]):

$$\text{Find } \lambda \in \mathbb{C} \text{ such that } \mathbf{P}(\lambda) \text{ is singular.} \tag{3.1}$$

The polynomial eigenvalues are exactly those $\lambda \in \mathbb{C}$ satisfying $\det(\mathbf{P}(\lambda)) = 0$. If $\det(\mathbf{P}(z)) \equiv 0$, the polynomial eigenvalue problem (3.1) is said to be *singular*; otherwise, it is *regular*. Typically, when the monomial basis coefficients $\{\mathbf{A}_k\}_{k=0}^n$ of $\mathbf{P}(z)$ are known *a priori* and when $\mathbf{A}_n$ is nonsingular, the polynomial eigenvalue problem (3.1) is solved by "linearising" and solving an eigenvalue problem for the associated

block companion matrix

$$\mathbf{C} := \begin{bmatrix} \mathbf{0} & & & & -\mathbf{A}_n^{-1}\mathbf{A}_0 \\ \mathbf{I} & \mathbf{0} & & & -\mathbf{A}_n^{-1}\mathbf{A}_1 \\ & \ddots & \ddots & & \vdots \\ & & \mathbf{I} & \mathbf{0} & -\mathbf{A}_n^{-1}\mathbf{A}_{n-2} \\ & & & \mathbf{I} & -\mathbf{A}_n^{-1}\mathbf{A}_{n-1} \end{bmatrix} \in \mathbb{C}^{ns\times ns} \tag{3.2}$$

(see, e.g., [10]). Matlab's `polyeig` routine solves the polynomial eigenvalue problem (3.1) by computing generalised eigenvalues of a related block matrix pencil to avoid explicitly finding $\mathbf{A}_n^{-1}$. In the case $n = 1$, the polynomial eigenvalue problem (3.1) is precisely the generalised eigenvalue problem for the matrix pencil $\lambda\mathbf{A}_1 + \mathbf{A}_0$. In the case $s = 1$, the polynomial eigenvalue problem (3.1) reduces to a standard polynomial root-finding problem with an equivalent eigenvalue problem for a companion matrix.

3.1. Essentially scalar matrix polynomials. To test algorithms for matrix polynomials, it is useful to have a family of matrix polynomials whose exact polynomial eigenvalues are known.

Definition 3.1. *An essentially scalar matrix polynomial is a matrix polynomial* $\mathbf{P}(z)$ *that can be written in the form*

$$\mathbf{P}(z) := p(z\mathbf{A}) \tag{3.3}$$

for some scalar polynomial $p(z)$ *and some* $s \times s$ *matrix* $\mathbf{A}$*.* Clearly, not every matrix polynomial is essentially scalar. If a matrix $\mathbf{A}$ is invertible with known simple eigenvalues, the preceding definition permits the construction of many matrix polynomials with known polynomial eigenvalues.

Proposition 3.2. *Let* $p(z)$ *be a scalar polynomial of degree* n *with distinct roots* ρ_j, $1 \le j \le n$ *and let* $\mathbf{A} \in \mathbb{C}^{s\times s}$ *be an invertible matrix with simple eigenvalues* μ_k, $1 \le k \le s$*. Suppose further that the* ns *quantities* $\lambda_{jk} := \rho_j/\mu_k$ *are all themselves distinct (this is not true in general, even given distinct* ρ_j *and* μ_k*). Then, under these circumstances, the matrix polynomial* $\mathbf{P}(z) := p(z\mathbf{A})$ *is regular and has* ns *distinct eigenvalues* λ_{jk}*.*

Proof. $\mathbf{P}(z)$ is regular because $\mathbf{A}^n$ is nonsingular. For each μ_k there exists an eigenvector $\mathbf{v}_k$ of $\mathbf{A}$. We have $\mathbf{A}\mathbf{v}_k = \mu_k\mathbf{v}_k$ and hence $\mathbf{A}^m\mathbf{v}_k = \mu_k^m\mathbf{v}_k$. Expressing $\mathbf{P}(z)$ in the monomial basis we see $\mathbf{P}(z)\mathbf{v}_k = p(z\mu_k)\mathbf{v}_k$. If $z = \lambda_{jk} = \rho_j/\mu_k$, we have

$$\mathbf{P}(\lambda_{jk})\mathbf{v}_k = p(\rho_j)\mathbf{v}_k = \mathbf{0}.$$

As there are ns such eigenvalues and by construction $P(z)$ is regular, we are done. □

Example 1. Let $p(z) = (z-1)(z-2)(z-3)(z-4)$ and

$$\mathbf{A} := \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}.$$

Then, the matrix polynomial $\mathbf{P}(z) := p(z\mathbf{A})$ expressed in the monomial basis is

$$\mathbf{P}(z) = p(z\mathbf{A}) = z^4\mathbf{A}^4 - 10z^3\mathbf{A}^3 + 35z^2\mathbf{A}^2 - 50z\mathbf{A} + 24\mathbf{I}$$

and its twelve polynomial eigenvalues are

$$-\frac{1}{2}, -1, -\frac{3}{2}, -2, -1 \pm \frac{1}{2}\sqrt{2}, -2 \pm \sqrt{2}, -3 \pm \frac{3}{2}\sqrt{2}, -4 \pm 2\sqrt{2}.$$

EXAMPLE 2. Let $p(z) = z^n - 1$ and

$$\mathbf{A} = \begin{bmatrix} \mu_1 & 1 & & & \\ & \mu_2 & 1 & & \\ & & \mu_3 & \ddots & \\ & & & \ddots & 1 \\ & & & & \mu_s \end{bmatrix}.$$

By choosing μ_k real and larger than one, we get polynomial eigenvalues of $\mathbf{P}(z)$ on concentric circles inside the unit circle (at radii $1/\mu_k$). By choosing μ_k positive but less than 1 we get eigenvalues of $\mathbf{P}(z)$ on concentric circles outside the unit circle.

4. Generalised Companion Matrix Pencils in the Lagrange basis. The approach for solving the polynomial eigenvalue problem (3.1) described in Section 3 is based on the assumption that the matrix polynomial $\mathbf{P}(z)$ is specified by its coefficients $\{\mathbf{A}_k\}_{k=0}^n$ relative to the power basis. Assume instead that the matrix polynomial is given by the values of $\mathbf{P}(z)$ at specific values of z. That is, let $\{(z_k, \mathbf{P}_k)\}_{k=0}^n \subset \mathbb{C} \times \mathbb{C}^{s\times s}$ be a collection of distinct[3] nodes $\{z_k\}_{k=0}^n \subset \mathbb{C}$ with associated numerical matrices $\{\mathbf{P}_k\}_{k=0}^n \subset \mathbb{C}^{s\times s}$. Then, there exists a unique family of s^2 scalar polynomials $\{p_{ij}(z)\}_{i,j=1}^s$ each of degree at most n such that $p_{ij}(z_k) = [\mathbf{P}_k]_{ij}$ $(i, j = 1, \ldots, s; k = 0, \ldots, n)$. This is more concisely written as

$$\mathbf{P}(z_k) = \mathbf{P}_k \in \mathbb{C}^{s\times s} \qquad (k = 0, \ldots, n).$$

Given the data $\{(z_k, \mathbf{P}_k)\}_{k=0}^n$, the corresponding polynomial eigenvalue problem (3.1) can be solved by determining the monomial basis coefficients $\{\mathbf{A}_k\}_{k=0}^n$ and subsequently finding the eigenvalues of the block companion matrix (3.2) (using, e.g., Matlab's `polyeig`). However, we would have to compute the monomial-basis coefficients of s^2 scalar polynomial interpolants with associated Vandermonde systems. This procedure is not generally advisable due to possible sensitivity of the coefficients $\{\mathbf{A}_k\}_{k=0}^n$ to perturbations in the data $\{\mathbf{P}_k\}_{k=0}^n$. In particular, changing the polynomial basis potentially worsens the conditioning of the associated eigenproblem.

For a concrete example, suppose the scalar data $\{p_k\}_{k=0}^n$ are obtained by accurately sampling the Wilkinson polynomial at some reasonably distributed points

[3] As was done for the DPR1 matrix of Smith [6], the theory developed here can be modified for the confluent case (i.e., when some of the nodes z_k are repeated), but we ignore this case in the present work.

$\{z_k\}_{k=0}^n$. The polynomial eigenvalues in this case are the usual roots of the Wilkinson polynomial. The monomial basis coefficients can be computed from the sampled data and the roots can be found from the eigenvalues of the corresponding companion matrix (e.g., with `eig` or `polyeig`). One finds, as expected, that the methods described in Section 5 for determining the roots directly from the data, i.e., avoiding interpolation, give substantially more accurate answers. See [29] for some representative results.

We can work with the data directly to circumvent such difficulties. If a matrix polynomial is specifed by the data $\{(z_k, \mathbf{P}_k)\}_{k=0}^n$ rather than the monomial basis coefficients, it is natural to express $\mathbf{P}(z)$ in the *Lagrange basis* $\{\ell_k(z)\}_{k=0}^n$, where the *Lagrange polynomials* $\ell_k(z)$ are defined by

$$\ell_k(z) = w_k \prod_{\substack{j=0 \\ j\neq k}}^{n} (z - z_j), \qquad (k = 0, \dots n) \tag{4.1a}$$

$$w_k := \prod_{\substack{j=0 \\ j\neq k}}^{n} \frac{1}{(z_k - z_j)} \qquad (k = 0, \dots n). \tag{4.1b}$$

The scalar factors w_k in (4.1b) are the *barycentric weights*[4]. The matrix polynomial $\mathbf{P}(z)$ expressed in the Lagrange basis $\{\ell_k(z)\}_{k=0}^n$ is

$$\mathbf{P}(z) = \sum_{k=0}^{n} \ell_k(z)\mathbf{P}_k. \tag{4.2}$$

Equivalently, $\mathbf{P}(z)$ can be expressed in one of two *barycentric forms*

$$\mathbf{P}(z) \;=\; \ell(z) \sum_{k=0}^{n} \frac{w_k}{z - z_k}\mathbf{P}_k, \quad \text{or} \tag{4.3a}$$

$$\mathbf{P}(z) \;=\; \left(\sum_{j=0}^{n} \frac{w_j}{z - z_j} \right)^{-1} \sum_{k=0}^{n} \frac{w_k}{z - z_k}\mathbf{P}_k, \quad \text{where} \tag{4.3b}$$

$$\ell(z) := (z - z_0)(z - z_1)\cdots(z - z_n). \tag{4.3c}$$

To derive the second barycentric form (4.3b), interpolate the constant polynomial $\mathbf{P}(z) = \mathbf{I}$ and solve for $\ell(z)$ [8].

With the notation (4.1) in hand, we define a generalised companion matrix pencil associated with the matrix polynomial $\mathbf{P}(z)$ specified by data $\{(z_k, \mathbf{P}_k)\}_{k=0}^n$ as in [3].

Definition 4.1. *Given* $\{(z_k, \mathbf{P}_k)\}_{k=0}^n \subset \mathbb{C} \times \mathbb{C}^{s\times s}$, *define the block matrix*

$$\mathbf{Z} := \mathbf{diag}[\, z_0\mathbf{I}, \dots, z_n\mathbf{I}\,] = \begin{bmatrix} z_0\mathbf{I} & & \\ & \ddots & \\ & & z_n\mathbf{I} \end{bmatrix} \in \mathbb{C}^{N\times N}, \tag{4.4a}$$

[4]The notation here differs from that of [8]: what they call L_j we call ℓ_j; we use the symbol $\mathbf{L}$ to denote the lower triangular matrix in $\mathbf{LU}$ factorisation.

where $N := s(n+1)$. *Further, define the block matrices*

$$\boldsymbol{\pi} := \begin{bmatrix}\mathbf{P}_0^H, & \cdots, & \mathbf{P}_n^H\end{bmatrix}^H \in \mathbb{C}^{N\times s} \text{ and} \tag{4.4b}$$

$$\boldsymbol{\omega}^H := \begin{bmatrix}w_0\mathbf{I}, & \ldots, & w_n\mathbf{I}\end{bmatrix} \in \mathbb{C}^{s\times N}. \tag{4.4c}$$

The generalised companion matrix pencil $\lambda\mathbf{C}_1 - \mathbf{C}_0$ *is given by*

$$\mathbf{C}_0 := \begin{bmatrix}\mathbf{Z} & \boldsymbol{\pi}\\ \boldsymbol{\omega}^H & \mathbf{0}\end{bmatrix} \quad \text{and} \tag{4.5a}$$

$$\mathbf{C}_1 := \begin{bmatrix}\mathbf{I} & \\ & \mathbf{0}\end{bmatrix}. \tag{4.5b}$$

In terms of the notation in (4.4), *the pencil* $\mathbf{C}(\lambda) := \lambda\mathbf{C}_1 - \mathbf{C}_0$ *is*

$$\mathbf{C}(\lambda) := \lambda\mathbf{C}_1 - \mathbf{C}_0 = \begin{bmatrix}\mathbf{Q}(\lambda) & -\boldsymbol{\pi}\\ -\boldsymbol{\omega}^H & \mathbf{0}\end{bmatrix}, \text{ where} \tag{4.5c}$$

$$\mathbf{Q}(\lambda) := \lambda\mathbf{I} - \mathbf{Z}. \tag{4.5d}$$

REMARK 4.2. Using the notation in (4.4), the barycentric interpolation formula (4.3a) is

$$\mathbf{P}(z) = \ell(z)\,\boldsymbol{\omega}^H\mathbf{Q}(z)^{-1}\boldsymbol{\pi} \quad \text{for } z \notin \{z_0, z_1, \ldots, z_n\}. \tag{4.6a}$$

The apparent discontinuity as $z \to z_k$ is removable because

$$\mathbf{P}(z_k) = \lim_{\lambda\to z_k} \ell(z)\,\boldsymbol{\omega}^H\mathbf{Q}(z)^{-1}\boldsymbol{\pi} = \mathbf{P}_k \qquad (k = 0, 1, \ldots, n). \tag{4.6b}$$

REMARK 4.3. The block row vector $\boldsymbol{\omega}^H$ defined in (4.4c) can also be written using the Kronecker product as $\boldsymbol{\omega}^H = [w_0, w_1, \ldots, w_n] \otimes \mathbf{I}$. Notice that the corresponding block column vector $\boldsymbol{\omega}$ has blocks scaled by the complex conjugates of the barycentric weights. This distinction is important in the event that the data or the polynomial eigenvalues are complex.

The relationship of the polynomial eigenvalue problem (3.1) to the companion pencil (4.5) is made apparent in Theorem 4.4.

THEOREM 4.4. *For any* $\lambda \in \mathbb{C}$, $\det(\lambda\mathbf{C}_1 - \mathbf{C}_0) \equiv \det(\mathbf{P}(\lambda))$.

Proof. If $\lambda \notin \{z_0, \ldots, z_n\}$, then $\mathbf{Q}(\lambda)$ is of full rank, so, using the barycentric formula (4.6a), we obtain

$$\begin{aligned}
\det(\lambda\mathbf{C}_1 - \mathbf{C}_0) &= \det\left(\begin{bmatrix}\mathbf{Q}(\lambda) & -\boldsymbol{\pi}\\ -\boldsymbol{\omega}^H & \end{bmatrix}\right)\\
&= \det\left(\begin{bmatrix}\mathbf{I} & \\ \boldsymbol{\omega}^H\mathbf{Q}(\lambda)^{-1} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{Q}(\lambda) & -\boldsymbol{\pi}\\ & \boldsymbol{\omega}^H\mathbf{Q}(\lambda)^{-1}\boldsymbol{\pi}\end{bmatrix}\right)\\
&= \det(\mathbf{Q}(\lambda))\ \det(\boldsymbol{\omega}^H\mathbf{Q}(\lambda)^{-1}\boldsymbol{\pi})\\
&= \ell(\lambda)^s\ \det(\boldsymbol{\omega}^H\mathbf{Q}(\lambda)^{-1}\boldsymbol{\pi})\\
&= \det(\mathbf{P}(\lambda)).
\end{aligned}$$

Furthermore, using (4.6b) in the limit as $\lambda \to z_k$, $\det(\lambda \mathbf{C}_1 - \mathbf{C}_0) \to \det(\mathbf{P}_k)$. ∎

As an immediate consequence of Theorem 4.4, any finite polynomial eigenvalue of (3.1) is also a generalised eigenvalue of the matrix pencil $\lambda \mathbf{C}_1 - \mathbf{C}_0$.

The generalised eigenvalue problem associated with the companion matrix pencil (4.5) has an additional s double eigenvalues at infinity. If $\boldsymbol{\omega}^H \boldsymbol{\pi}$ is not singular, these are the only such [30]. In practice, these infinite eigenvalues do not disrupt numerical computation, even if the QZ iteration is used; in the Rayleigh quotient methods discussed here, they are automatically avoided.

REMARK 4.5. Clearly, multiplying $\mathbf{P}(z)$ by any nonzero $S \in \mathbb{C}^{s \times s}$ does not alter the polynomial eigenvalues in (3.1). Hence, we may make the substitutions $\mathbf{P}_k \leftarrow S\,\mathbf{P}_k$ in all the rightmost blocks of $\mathbf{C}_0$ in (4.5) without changing the associated generalised eigenvalues of the pencil $\lambda \mathbf{C}_1 - \mathbf{C}_0$. Similarly, multiplying the barycentric formula (4.6a) by any scalar c implies that the substitutions $w_k \mathbf{I} \leftarrow c w_k\,\mathbf{I}$ in all the bottom blocks of $\mathbf{C}_0$ in (4.5) do not affect the generalised eigenvalues of the pencil.

Finally, we may scale all z_k by a constant, say $Z = \max |z_k|$ and $\widehat{z}_k = z_k / Z$. This induces a scaling of the barycentric weights by Z^n. If $\widehat{\lambda}_{ij}$ are the generalised eigenvalues of the pencil $\lambda \widehat{C}_1 - \widehat{C}_0$ formed with the scaled $\widehat{z}_k$, then the original polynomial eigenvalues λ_{ij} are related by $\widehat{\lambda}_{ij} = \lambda_{ij} / Z$.

We assume henceforth that the values $\{\mathbf{P}_k\}_{k=0}^n$ are suitably scaled for computation *by a person knowledgable about the particular application context* (e.g., by taking $c = (\max_{0 \le k \le n} \|\mathbf{P}_k\|)^{-1} \mathbf{I}$). This corresponds to working with non-monic polynomials in the monomial case. We do not yet know enough about this to recommend any particular automatic scaling in what follows, though we do recommend that the person doing the computation pay attention to this issue. For the RQI methods discussed in this paper, some of these scalings are of lesser importance, but if the standard QZ iteration is used, an incorrect scaling can lead to dramatically wrong answers.

4.1. Eigenvectors of the Generalised Companion Matrix Pencil. If $\lambda \in \mathbb{C}$ is a finite polynomial eigenvalue of the matrix polynomial, the matrix $\mathbf{P}(\lambda)$ is singular. Let $\mathbf{u}$ and $\mathbf{v}^H$ be associated right and left null vectors of $\mathbf{P}(\lambda)$ respectively for the polynomial eigenvalue λ. Then, λ is also a generalised eigenvalue of the pencil $\mathbf{C}(\lambda)$ as Theorem 4.4 implies. Lemma 4.6 explicitly provides the right and left generalised eigenvectors $\boldsymbol{\xi}(\lambda)$ and $\boldsymbol{\eta}^H(\lambda)$ of $\mathbf{C}(\lambda)$ in terms of the polynomial eigenvalue λ and the null vectors $\mathbf{u}$ and $\mathbf{v}^H$ of $\mathbf{P}(\lambda)$.

LEMMA 4.6. *Given the data* $\{(z_k, \mathbf{P}_k)\}_{k=0}^n$, *let* $\mathbf{P}(z)$ *be the associated interpolating matrix polynomial. Assume that* $\lambda \in \mathbb{C} \backslash \{z_0, \ldots, z_n\}$ *is a simple polynomial eigenvalue of the matrix polynomial* $\mathbf{P}(z)$ *that is distinct from the interpolation nodes* $\{z_0, \ldots, z_n\}$. *If the associated right and left null vectors of* $\mathbf{P}(\lambda)$ *are* $\mathbf{u}$ *and* $\mathbf{v}^H$ *respectively, then the corresponding right and left generalised eigenvectors of the generalised companion matrix pencil* $\mathbf{C}(\lambda) = \lambda \mathbf{C}_1 - \mathbf{C}_0$ *from* (4.5) *are respectively* $\boldsymbol{\xi}(\lambda)$ *and* $\boldsymbol{\eta}(\lambda)^H$,

where

$$\boldsymbol{\xi}(\lambda) := \begin{bmatrix} \mathbf{Q}(\lambda)^{-1}\boldsymbol{\pi}\mathbf{u} \\ \mathbf{u} \end{bmatrix} \quad \textit{and} \tag{4.7a}$$

$$\boldsymbol{\eta}^H(\lambda) := \begin{bmatrix} \mathbf{v}^H\boldsymbol{\omega}^H\mathbf{Q}(\lambda)^{-1}, & -\mathbf{v}^H \end{bmatrix}. \tag{4.7b}$$

Proof. Assume that $\lambda \notin \{z_0, \ldots, z_n\}$, so $\ell(\lambda) \neq 0$ and $\mathbf{Q}(\lambda)$ is of full rank. If we define $\boldsymbol{\xi}(\lambda)$ and $\boldsymbol{\eta}(\lambda)$ as in (4.7), we find

$$\begin{aligned} \mathbf{C}(\lambda)\boldsymbol{\xi}(\lambda) &= \begin{bmatrix} \mathbf{Q}(\lambda) & -\boldsymbol{\pi} \\ \boldsymbol{\omega}^H & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{Q}(\lambda)^{-1}\boldsymbol{\pi}\mathbf{u} \\ \mathbf{u} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\omega}^H\mathbf{Q}(\lambda)^{-1}\boldsymbol{\pi}\mathbf{u} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} \\ (\ell(\lambda))^{-1}\,\mathbf{P}(\lambda)\mathbf{u} \end{bmatrix} \qquad \text{(from (4.6a))} \\ &= \mathbf{0}, \end{aligned}$$

because $\mathbf{P}(\lambda)\mathbf{u} = \mathbf{0}$ by hypothesis. A similar calculation yields $\boldsymbol{\eta}^H(\lambda)\mathbf{C}(\lambda) = \mathbf{0}^H$. ∎

Remark 4.7. In the event that one of the nodes z_k happens to be a polynomial eigenvalue of the matrix polynomial $\mathbf{P}(z)$, the formula (4.7a) for the right eigenvector $\boldsymbol{\xi}(z_k)$ cannot be used directly because $\mathbf{Q}(z_k)$ is singular. Instead, the kth block row of the right eigenvector $\boldsymbol{\xi}(z_k)$ in (4.7a) is replaced by

$$\mathbf{P}'(z_k)\mathbf{u} = -\frac{1}{w_k}\sum_{\substack{j=0 \\ j\neq k}}^{n} \frac{w_j\mathbf{P}_j\mathbf{u}}{z_k - z_j}. \tag{4.8}$$

The corresponding left eigenvector in that case is $\boldsymbol{\eta}^H(z_k) = \mathbf{e}_k^H \otimes \mathbf{v}^H$.

5. Algorithms for Generalised Companion Matrix Pencils. We wish to solve the polynomial eigenvalue problem (3.1) represented in the Lagrange basis by applying various methods for solving generalised eigenvalue problems to the generalised companion matrix pencil (4.5c). We describe two techniques using sequences of Rayleigh quotients. The first explicitly uses the structure of the eigenvectors in Lemma 4.6 to update the putative eigendirections. The second uses explicit formulas to solve the linear systems efficiently that determine the updated eigendirections; the formulas are based on the structure of the companion matrix pencil $\mathbf{C}(\lambda)$ in (4.5c).

5.1. Constrained RQI for the Generalised Companion Matrix Pencil. Lemma 4.6 provides the formulas (4.7) relating a simple eigenvalue λ of the generalised companion matrix pencil $\mathbf{C}(\lambda)$ to its corresponding right and left generalised eigenvectors $\boldsymbol{\xi}(\lambda)$ and $\boldsymbol{\eta}^H(\lambda)$ respectively. Thus, this basic form for the generalised eigenvectors can be used as constraints that the tentative eigenvectors must satisfy within an iteration based on Rayleigh quotients. Following [16], Algorithm 5.1 consists of using $\lambda_{(k)}$ from Lemma 4.6 to explicitly update the eigendirections $\boldsymbol{\xi}_{(k)}$ and $\boldsymbol{\eta}_{(k)}$.

The vectors $\mathbf{u}$ and $\mathbf{v}^H$ can be chosen at random; convergence can still be achieved provided that $\mathbf{u}$ and $\mathbf{v}^H$ have nonzero components along the actual null directions of $P(\lambda)$ [16].

ALGORITHM 5.1 (Constrained RQI).

Input: $\lambda_{(0)} \in \mathbb{C}$, $\mathbf{u}, \mathbf{v} \in \mathbb{C}^{s\times 1}$

for $k = 1, 2, \ldots$

Compute $\boldsymbol{\xi}_{(k)} \leftarrow \boldsymbol{\xi}(\lambda_{(k-1)})$ as in (4.7a)

Compute $\boldsymbol{\eta}_{(k)} \leftarrow \boldsymbol{\eta}^H(\lambda_{(k-1)})$ as in (4.7b)

Compute $\lambda_{(k)} \leftarrow \dfrac{\boldsymbol{\eta}_{(k)}^H \mathbf{C}_0 \boldsymbol{\xi}_{(k)}}{\boldsymbol{\eta}_{(k)}^H \mathbf{C}_1 \boldsymbol{\xi}_{(k)}}$

end for

When $s = 1$, $\mathbf{P}(\lambda) = p(\lambda)$ is a scalar polynomial with scalar coefficients $\{p_k\}_{k=0}^n$. In that case, Algorithm 5.1 reduces to the formula

$$\lambda_{(k)} = \lambda_{(k-1)} + \frac{\sum_{j=0}^n (\lambda_{(k-1)} - z_j)^{-1} w_j p_j}{\sum_{j=0}^n (\lambda_{(k-1)} - z_j)^{-2} w_j p_j}, \tag{5.1a}$$

which is Newton's method for the rational function

$$r(z) = \sum_{j=0}^n \frac{w_j p_j}{z - z_j} = \frac{p(z)}{\ell(z)}. \tag{5.1b}$$

This can be compared to the secular equation [14] used in some divide-and-conquer algorithms for eigenvalue problems [31]. The dynamical behaviour of Newton's method is now well understood (see, e.g., [32]). The convergence is quadratic and the basin of attraction of each root usually has fractal boundaries. Assuming that the interpolation nodes $\{z_j\}_{j=0}^n$ do not coincide with the roots of $p(z)$, the zeros of $r(z)$ and $p(z)$ are identical. When a root does coincide with an interpolation node z_j, the corresponding coefficient $p_j = 0$, so the polynomial $p(z)$ deflates trivially.

5.2. Unconstrained RQI for the Generalised Companion Matrix Pencil. The constrained iteration in Algorithm 5.1 explicitly enforces an asymptotically correct structure in the putative generalised eigenvectors $\boldsymbol{\xi}_{(k)}$ and $\boldsymbol{\eta}_{(k)}^H$ before updating the Rayleigh quotients. By contrast, we can implement the generalised Rayleigh quotient iteration in Algorithm 2.2 to specify how the eigendirections should be updated. To solve the linear systems $(\lambda\mathbf{C}_1 - \mathbf{C}_0)\boldsymbol{\xi} = \mathbf{C}_1\mathbf{x}$ and $(\lambda\mathbf{C}_1 - \mathbf{C}_0)^H\boldsymbol{\eta} = \mathbf{C}_1^H\mathbf{y}$ efficiently, we present explicit formulas for the $\mathbf{LU}$ factors and inverse of the pencil $\lambda\mathbf{C}_1 - \mathbf{C}_0$ in Theorem 5.2. Naturally, in computation, the inverse is not used, but the particular formula (5.3) for $\mathbf{C}(\lambda)^{-1}$ is useful for the complexity analysis.

THEOREM 5.2. *Let $\mathbf{C}_0$ and $\mathbf{C}_1$ be given as in (4.5). Assume that λ is not an eigenvalue of the pencil $\mathbf{C}(\lambda) = \lambda\mathbf{C}_1 - \mathbf{C}_0$ and that $\lambda \notin \{z_0, \ldots, z_n\}$. Then, $\mathbf{C}(\lambda) = \mathbf{L}(\lambda)\mathbf{U}(\lambda)$, where*

$$\mathbf{L}(\lambda) = \begin{bmatrix} \mathbf{I} & \\ \boldsymbol{\omega}^T\mathbf{Q}(\lambda)^{-1} & \mathbf{I} \end{bmatrix} \text{ and} \tag{5.2a}$$

$$\mathbf{U}(\lambda) = \begin{bmatrix} \mathbf{Q}(\lambda) & -\boldsymbol{\pi} \\ & \ell(\lambda)^{-1}\mathbf{P}(\lambda) \end{bmatrix}. \tag{5.2b}$$

Furthermore, the matrix $\mathbf{C}(\lambda)^{-1}$ *is given explicitly by*

$$\mathbf{C}(\lambda)^{-1} = \begin{bmatrix} \mathbf{Q}(\lambda)^{-1} & \\ & \mathbf{0} \end{bmatrix} + \ell(\lambda) \begin{bmatrix} \mathbf{Q}(\lambda)^{-1}\boldsymbol{\pi} \\ \mathbf{I} \end{bmatrix} \mathbf{P}(\lambda)^{-1} \begin{bmatrix} -\boldsymbol{\omega}^T\mathbf{Q}(\lambda)^{-1}, & \mathbf{I} \end{bmatrix}. \tag{5.3}$$

Proof. The **LU** factors can be verified by direct computation. Similarly, multiply the expression in (5.3) on the left or right by $(\lambda\mathbf{C}_1 - \mathbf{C}_0)$ and expand out, using the barycentric form (4.6a) to simplify. □

REMARK 5.3. The expression for $\mathbf{C}(\lambda)^{-1}$ in (5.3) is comparable to the formula

$$p(t)(t\mathbf{I} - \mathbf{C})^{-1} = \sum_{k=1}^{n} \mathbf{M}_k t^{k-1} \tag{5.4}$$

shown in [33] (from Gantmacher's classic book [27]). In (5.4), the matrix $\mathbf{C}$ is the companion matrix in (3.2) of the scalar monic polynomial $p(t) = t^n + a_{n-1}t^{n-1} + \cdots + a_0$ as represented in the monomial basis and the centraliser $\mathbf{M}_k = \sum_{j=k}^{n} a_j \mathbf{C}^{j-k}$ resembles a Sylvester matrix.

REMARK 5.4. When $\lambda = z_k \in \{z_0, \ldots, z_n\}$, the formulas for the **LU** factors in (5.3) cannot be used. However, if z_k is not a polynomial eigenvalue of $\mathbf{P}(z)$, then $\mathbf{P}_k = \mathbf{P}(z_k)$ is invertible and the appropriate **LU** factors of $\mathbf{C}(z_k)$ can be constructed. If z_k is a polynomial eigenvalue, then $\mathbf{P}(z_k) = \mathbf{P}_k$ is necessarily singular as is the pencil $\mathbf{C}(z_k)$, so no **LU** factors exist.

The **LU** factors in (5.2) provide efficient means for solving the linear systems $(\lambda\mathbf{C}_1 - \mathbf{C}_0)\boldsymbol{\xi} = \mathbf{C}_1\mathbf{x}$ and $(\lambda\mathbf{C}_1 - \mathbf{C}_0)^H\boldsymbol{\eta} = \mathbf{C}_1\mathbf{y}$. In particular, an immediate corollary of Theoremm 5.2 is that the tentative eigendirections generated in Rayleigh quotient iteration can be updated in $O(n)$ operations.[5] We use the explicit inverse $\mathbf{C}(\lambda)^{-1}$ in (5.3) to establish the desired complexity result.

COROLLARY 5.5. *Under the hypotheses of Lemma 5.2, the linear systems*

$$(\lambda\mathbf{C}_1 - \mathbf{C}_0)\boldsymbol{\xi} = \mathbf{C}_1\mathbf{x} \text{ and } (\lambda\mathbf{C}_1 - \mathbf{C}_0)^H\boldsymbol{\eta} = \mathbf{C}_1^H\mathbf{y}$$

can be solved in $O(ns^2 + s^3)$ *operations using* $O(ns^2)$ *storage.*

The unconstrained generalised Rayleigh quotient iteration proceeds starting from tentative eigendirections $\boldsymbol{\xi}_{(1)}$ and $\boldsymbol{\eta}_{(1)}^H$. In practice, the directions $\boldsymbol{\xi}_{(1)}$ and $\boldsymbol{\eta}_{(1)}^H$ are generated from an initial tentative eigenvalue $\lambda_{(0)}$ and random directions $\mathbf{u}, \mathbf{v} \in \mathbb{C}^{s\times 1}$ and using (4.7) as in the first iteration of Algorithm 5.1. After normalising $\boldsymbol{\xi}_{(1)}$ and $\boldsymbol{\eta}_{(1)}^H$ to yield $\mathbf{x}_{(1)}$ and $\mathbf{y}_{(1)}^H$, the sequence of Rayleigh quotients are given by

$$\lambda_{(k)} = \frac{\mathbf{y}_{(k)}^H\mathbf{C}_0\mathbf{x}_{(k)}}{\mathbf{y}_{(k)}^H\mathbf{C}_1\mathbf{x}_{(k)}} \quad (k = 1, 2, \ldots) \tag{5.5a}$$

[5] That is, we think of a family of problems of constant s and varying n; as n increases, the cost increases linearly with n. Sometimes we include the s factors, as in $O(ns^2)$, which gives a shorthand for another limit, namely a constant n and a growing s. Finally, $O(ns^2 + s^3)$ means $O(ns^2) + O(s^3)$, and in both cases we may think of either $n \to \infty$ or $s \to \infty$, but not both.

After the first iteration, the vectors $\boldsymbol{\xi}_{(k)}$ and $\boldsymbol{\eta}_{(k)}$ are no longer generated from $\lambda_{(k-1)}$ as in (4.7). Instead, the updated eigendirections are determined by solving the linear systems

$$\left[\lambda_{(k)}\mathbf{C}_1 - \mathbf{C}_0\right]\boldsymbol{\xi}_{(k+1)} = \mathbf{C}_1\mathbf{x}_{(k)} \text{ and } \left[\lambda_{(k)}\mathbf{C}_1 - \mathbf{C}_0\right]^H\boldsymbol{\eta}_{(k+1)} = \mathbf{C}_1^H\mathbf{y}_{(k)}$$

using (5.3). The resulting eigendirections are normalised by $\mathbf{x}_{(k)} = \boldsymbol{\xi}_{(k)}/\|\boldsymbol{\xi}_{(k)}\|_2$ and $\mathbf{y}_{(k)} = \boldsymbol{\eta}_{(k)}/\|\boldsymbol{\eta}_{(k)}\|_2$. Algorithm 2.2 is an adaptation of Algorithm 2.2 for the generalised companion matrix pencil (4.5).

ALGORITHM 5.6 (Unconstrained RQI).

Input: $\lambda_{(0)} \in \mathbb{C}$, $\mathbf{u}, \mathbf{v} \in \mathbb{C}^{s\times 1}$
Initialise $\boldsymbol{\xi}_{(1)} \leftarrow \boldsymbol{\xi}(\lambda_{(0)})$ as in (4.7a)
$\boldsymbol{\eta}_{(1)}^H \leftarrow \boldsymbol{\eta}^H(\lambda_{(0)})$ as in (4.7b)
for $k = 1, 2, \ldots$
Normalise $\mathbf{x}_{(k)} \leftarrow \|\boldsymbol{\xi}_{(k)}\|_2{}^{-1}\boldsymbol{\xi}_{(k)}$
Normalise $\mathbf{y}_{(k)}^H \leftarrow \|\boldsymbol{\eta}_{(k)}^H\|_2{}^{-1}\boldsymbol{\eta}_{(k)}^H$
Compute $\lambda_{(k)} \leftarrow \dfrac{\mathbf{y}_{(k)}^H\mathbf{C}_0\mathbf{x}_{(k)}}{\mathbf{y}_{(k)}^H\mathbf{C}_1\mathbf{x}_{(k)}}$
Solve $[\lambda_{(k)}\mathbf{C}_1 - \mathbf{C}_0]\boldsymbol{\xi}_{(k+1)} = \mathbf{x}_{(k)}$ for $\boldsymbol{\xi}_{(k+1)}$
Solve $[\lambda_{(k)}\mathbf{C}_1 - \mathbf{C}_0]^H\boldsymbol{\eta}_{(k+1)} = \mathbf{y}_{(k)}$ for $\boldsymbol{\eta}_{(k+1)}$
end for

The Rayleigh quotients $\lambda_{(k)}$ in (5.5a) can be computed by a more stable formula.

$$\begin{aligned}\lambda_{(k)} &= \frac{\mathbf{y}_{(k)}^H\mathbf{C}_0\mathbf{x}_{(k)}}{\mathbf{y}_{(k)}^H\mathbf{C}_1\mathbf{x}_{(k)}}\\ &= \lambda_{(k-1)} + \frac{\mathbf{y}_{(k)}^H(\mathbf{C}_0 - \lambda_{(k-1)}\mathbf{C}_1)\mathbf{x}_{(k)}}{\mathbf{y}_{(k)}^H\mathbf{C}_1\mathbf{x}_{(k)}}\\ &= \lambda_{(k)} - \frac{\mathbf{y}_{(k)}^H\mathbf{C}_1\mathbf{x}_{(k-1)}}{\mathbf{y}_{(k)}^H\mathbf{C}_1\boldsymbol{\xi}_{(k)}}\end{aligned} \tag{5.5b}$$

In the actual implementation of Algorithm 5.6, we use (5.5b) rather than (5.5a) to calculate the Rayleigh quotients after the first iteration.

5.3. Global Convergence. It is well-known that convergence can fail for the two-sided Rayleigh quotient iteration only if the initial left and right putative eigenvectors are each close to a left eigenvector for one eigenvalue and a right eigenvector for another eigenvalue [22]. In this work, we use the asymptotically correct (known) structures for the left and right eigenvectors of the companion matrix pencil, parameterised by the initial guess for the eigenvalue. If one of these is close to a true eigenvector, then the other must also be close to an eigenvector, corresponding to the same eigenvalue. Thus, for almost all initial guesses, the iteration converges. Ultimately, it converges cubically, as is to be expected.

5.4. Structured Deflation. Consider first the scalar case where $s = 1$, so $\mathbf{P}(z) = p(z)$ is a scalar polynomial of degree n. Suppose we have identified a root λ of $p(z)$ and we wish to reduce the problem to determining the roots of a lower degree polynomial. Then, there exists a polynomial $\widehat{p}(z)$ of degree $n-1$, such that:

$$p(z) = \widehat{p}(z)(z - \lambda) \tag{5.6}$$

Making the substitutions $z = z_j$ into (5.6), we find

$$\widehat{p}(z_j) = \frac{p(z_j)}{z_j - \lambda} \qquad (j = 0, \ldots, n).$$

Moreover, as $\widehat{p}(z)$ is of degree $n-1$, n data points are sufficient to contruct an interpolant, so one data point can be eliminated, say (z_k, p_k). One may need to apply the "Nearest Polynomial of Lower Degree" process [34] in order to counter possible instability issues. The barycentric weights w_j also need to be revised appropriately by the relation

$$\widehat{w}_j = w_j(z_j - z_k) \qquad (j = 0, \ldots, n,\ j \neq k).$$

In summary, given a root λ of $p(z)$, the deflation process is as follows:

1. Eliminate a node, say, the node that is closest to the root, i.e. choose z_k such that $k = \arg\min_j\{|\lambda - z_j| \ : \ j = 0, \cdots, n\}$.
2. Eliminate the corresponding value p_k.
3. Define $\widehat{p}_j = \frac{p_j}{z_j - \lambda}, \quad (j = 0, \ldots, n,\ j \neq k)$.
4. Update $\widehat{w}_j = w_j(z_j - z_k), \quad (j = 0, \ldots, n,\ j \neq k)$.

The new companion matrix pencil is constructed using the updated values $\widehat{p}_j$ and weights $\widehat{w}_j$.

For the $s \times s$ matrix case, the corresponding pencil can be deflated in $s \times s$ blocks [35]. This is most useful if $n \gg s$. The extension of the scalar algorithm to the matrix case is as follows.

THEOREM 5.7. *To deflate an $s \times s$ matrix polynomial $\mathbf{P}(x)$ in the Lagrange basis:*

1. *Use Algorithm 5.1 or 5.6 to find s generalised eigenvalues $\lambda_0, \ldots, \lambda_\ell$ $(\ell \leq s-1)$ (some eigenvalues may be repeated[6]).*
2. *Find s corresponding null vectors $\boldsymbol{u}_0, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_{s-1}$ of $\mathbf{P}(\lambda_j)$ $(j = 0, \ldots, \ell)$.*
3. *Eliminate a node, say, the node that is closest to any root, i.e., choose z_k such that $k = \arg\min_k \{\min_j\{|\lambda_j - z_k| \ : \ j = 0, \ldots, \ell\} \ : \ k = 0, \ldots, n\}$.*
4. *Define the matrices*

$$\widehat{\mathbf{P}}_j := \left(\tfrac{1}{z_j - \lambda_0}\mathbf{P}_j\boldsymbol{u}_0, \tfrac{1}{z_j - \lambda_1}\mathbf{P}_j\boldsymbol{u}_1, \cdots, \tfrac{1}{z_j - \lambda_\ell}\mathbf{P}_j\boldsymbol{u}_{s-1}\right) \quad (j = 0, \ldots, n,\ j \neq k).$$

5. *Update $\widehat{w}_j = w_j(z_j - z_k), \quad (j = 0, \ldots, n,\ j \neq k)$.*

[6] Note that repeated trials may be necessary, or heuristics such as discussed below, in order to find s eigenvalues without deflation.

The new companion matrix pencil is then constructed using the updated coefficients $\widehat{\mathbf{P}}_j$ *and weights* $\widehat{w}_j$*.*

Proof. If $\boldsymbol{u}_0, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_{s-1}$ are the right null vectors of $\mathbf{P}(z)$ corresponding to $\lambda_0, \lambda_1, \cdots, \lambda_{s-1}$ respectively, then there exist vectors $\widehat{\boldsymbol{p}}_0(z), \widehat{\boldsymbol{p}}_1(z), \cdots, \widehat{\boldsymbol{p}}_{s-1}(z)$ such that:

$$\begin{aligned}
&\mathbf{P}(z)\boldsymbol{u}_0 = (z-\lambda_0)\widehat{\boldsymbol{p}}_0(z)\\
&\mathbf{P}(z)\boldsymbol{u}_1 = (z-\lambda_1)\widehat{\boldsymbol{p}}_1(z)\\
&\vdots\\
&\mathbf{P}(z)\boldsymbol{u}_{s-1} = (z-\lambda_{s-1})\widehat{\boldsymbol{p}}_{s-1}(z)
\end{aligned}$$

The above equations can be written in the following matrix form:

$$\mathbf{P}(z)\mathbf{U} = \widehat{\mathbf{P}}(z)\mathbf{D}(z) \tag{5.8}$$

where $\mathbf{U} = [\boldsymbol{u}_0, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_{s-1}]$, $\widehat{\mathbf{P}}(z) = [\widehat{\boldsymbol{p}}_0(z), \widehat{\boldsymbol{p}}_1(z), \cdots, \widehat{\boldsymbol{p}}_{s-1}(z)]$ and $\mathbf{D} = \mathbf{diag}[z - \lambda_0, z-\lambda_1, \cdots, z-\lambda_{s-1}]$. $\mathbf{U}$ is invertible, because at this point, the eigenvalues are supposed to be distinct. Therefore it can be observed that:

$$\widehat{\mathbf{P}}(z) = \mathbf{P}(z)\mathbf{U}\mathbf{D}^{-1}(z) \tag{5.9}$$

This means:

$$\det \widehat{\mathbf{P}}(z) = \frac{\det \mathbf{P}(z) \det \mathbf{U}}{\det \mathbf{D}(z)} \tag{5.10}$$

which means that $\widehat{\mathbf{P}}(z)$ has the same eigenvalues as $\mathbf{P}(z)$ except for the ones eliminated by $\mathbf{D}(z)$. Now (5.9) can be evaluated at the given $n+1$ nodes.

$$\begin{aligned}
&\widehat{\mathbf{P}}_0 = \mathbf{P}_0\mathbf{U}\mathbf{D}^{-1}(z_0)\\
&\widehat{\mathbf{P}}_1 = \mathbf{P}_1\mathbf{U}\mathbf{D}^{-1}(z_1)\\
&\vdots\\
&\widehat{\mathbf{P}}_n = \mathbf{P}_n\mathbf{U}\mathbf{D}^{-1}(z_n)
\end{aligned}$$

and just like the scalar case, after polishing by NPLD [34], one node can be eliminated, say z_k, and the barycentric weights updated as

$$\widehat{w}_j = w_j(z_j - z_k) \quad j \neq k\,.$$

□

5.5. Multiple polynomial eigenvlaues. Multiple polynomial eigenvalues are studied in the monomial case in, for example, [36]. An adaptation of that algorithm, or something equivalent, seems to be necessary here. We first sketch some initial ideas, which will be pursued in a future paper.

Here, if some of the eigenvectors are multiple, say (without loss of generality) λ_0 is multiple of order $m \leq s$. In that case, instead of $\mathbf{D}$, $\mathbf{J} = \mathbf{diag}[z\mathbf{I} - \mathbf{J}_0, z - \lambda_m, \cdots, z - \lambda_{s-1}]$ should be defined where $\mathbf{J}_0$ is the Jordan block corresponding to λ_0. In that case $\mathbf{U} = [\boldsymbol{u}_{00}, \boldsymbol{u}_{01}, \cdots, \boldsymbol{u}_{0,m-1}, \boldsymbol{u}_m, \cdots, \boldsymbol{u}_{s-1}]$, where $\boldsymbol{u}_{0j}, \quad j = 0, \cdots, m-1$ are the generalised null vectors of $\mathbf{P}(z)$ corresponding to λ_0. In that case, $\widehat{\mathbf{P}}$ will be evaluated as follows:

$$\begin{aligned}
\widehat{\mathbf{P}}_0 &= \mathbf{P}_0\mathbf{U}\mathbf{J}^{-1}(z_0)\\
\widehat{\mathbf{P}}_1 &= \mathbf{P}_1\mathbf{U}\mathbf{J}^{-1}(z_1)\\
&\vdots\\
\widehat{\mathbf{P}}_n &= \mathbf{P}_n\mathbf{U}\mathbf{J}^{-1}(z_n)
\end{aligned}$$

We have tried this on a few examples with low multiplicity; even without special care, the computation of multiple polynomial eigenvalues is surprisingly robust. More work is clearly required, however.

6. Numerical Experiments. We report some numerical experiments to illustrate the theoretical results of Section 5. The computations are performed in Matlab or Maple with standard hardware precision.

6.1. Direct vs Indirect Root-Finding. In this part, we want to compare the direct method of root-finding with the indirect one i.e. finding the roots in Lagrange Basis using the generalised companion matrix pencils versus converting the given data to the monomial basis and finding the roots there.

We consider the scaled Wilkinson polynomial $p(z) = \prod_{j=1}^{20}(z - j/21)$ and sample $p(z)$ at points uniformly scattered over $[0, 1]$. We observed that the direct root-finding technique is more accurate than the conversion method (Figure 6.1).

For a matrix polynomial example, we first consider a small matrix polynomial of low degree: a matrix polynomial of size $s = 2$ and degree $n = 2$. The matrix polynomial is

$$\mathbf{P}(z) = \begin{pmatrix} \frac{1}{2} + z^2 & z + 0.8\,iz \\ z & \frac{1}{4} + z^2 \end{pmatrix}$$

and the nodes are $z_0 = -0.24 - 0.41\,i$, $z_1 = 0$, and $z_2 = 0.52 + 0.19\,i$. It turns out that the direct computation of the polynomial eigenvalues and the conversion to the monomial basis has about the same accuracy in this case. The infinity-norm of absolute difference between the computed eigenvalues in both cases is 7.5×10^{-11} (Figure 6.2).

Next, we consider an essentially scalar matrix polynomial of size $s = 4$ and of degree $n = 8$. Let $\mathbf{Q}(z) := z^8\mathbf{T}^8 - \mathbf{I}_4$, where $\mathbf{T}$ is the companion matrix associated

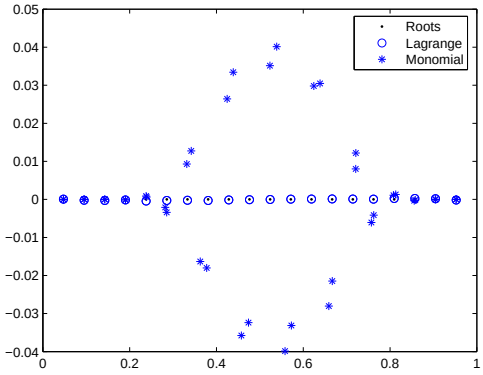

FIG. 6.1. *The roots of the scaled Wilkinson polynomial. Roots found directly through the generalised companion matrix pencil are more accurate than the roots found through conversion to monomial basis. Roots are double.*

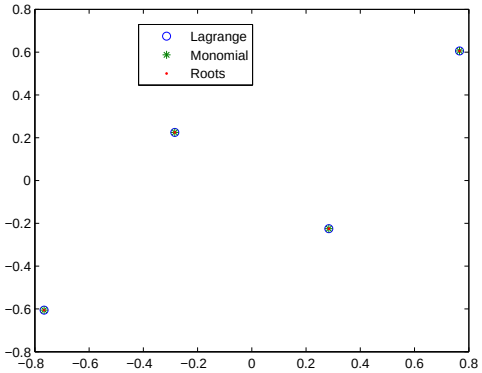

FIG. 6.2. *Small matrix polynomial example. The roots found directly through the generalised companion matrix pencil are about as accurate as the roots found through conversion to monomial basis.*

with the of the Jacobi polynomial $P_4^{0,10}(z)$. From Proposition 3.2, we thus know the polynomial eigenvalues of $\mathbf{Q}(z)$ exactly. The interpolation nodes $\{z_k\}_{k=0}^{8}$ are 0 and the eighth roots of unity. We observe that the computation of the polynomial eigenvalues of $\mathbf{Q}(z)$ directly from the sample data is more accurate than the polynomial eigenvalues computed by first computing the monomial basis coefficients of $\mathbf{Q}(z)$ and applying Matlab's `polyeig` routine. The infinty-norm of the error of the computed eigenvalues is in the former case is 3.12×10^{-6} while it is 6.32×10^{-5} (Figure 6.3).

6.2. Basin of Attraction: Constrained Method. We first considered the polynomial $z^4 - 1$, sampled at the symmetric points $z = \{0, 1+i, 1-i, -1+i, -1-i\}$.

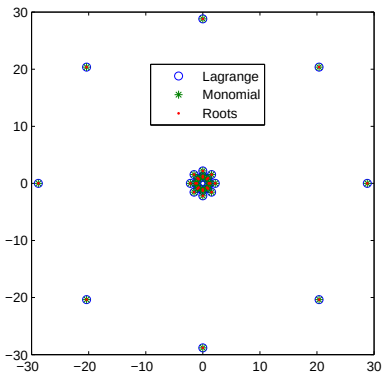

FIG. 6.3. *An essentially scalar matrix polynomial with $s = 4$ and $n = 8$. The roots found directly through the generalised companion matrix pencil are more accurate than the roots found through conversion to monomial basis.*

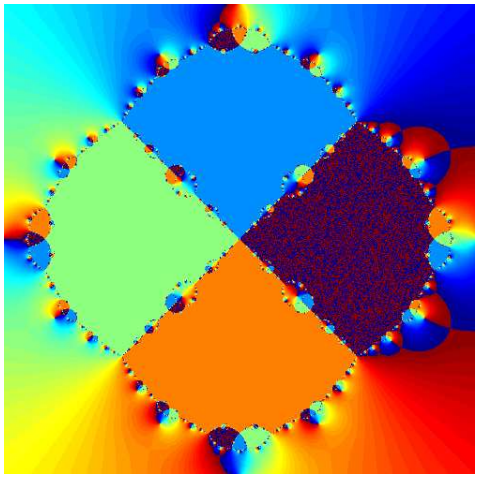

FIG. 6.4. *The basins of attraction of constrained Rayleigh quotient iteration are shown. Note the characteristic fractal nature of the boundary between the basins of attraction.*

We observed that initial guesses exterior to a region defined by these points (something like the convex hull, but with an apparently fractal boundary) do not converge to any root in 30 iterations(Figure 6.4).

We then considered the polynomial $(z-1)^2(z^2+1)$, sampled at the symmetric points $z = \{0, 1+i, 1-i, -1+i, -1-i\}$. We observe a behaviour similar to that above. Therefore, in the multiple-root case for constrained iteration, again initial guesses exterior to a region loosely defined by the nodes do not converge to any root in 30 iterations(Figure 6.5). See [29] for a discussion of the condition number of

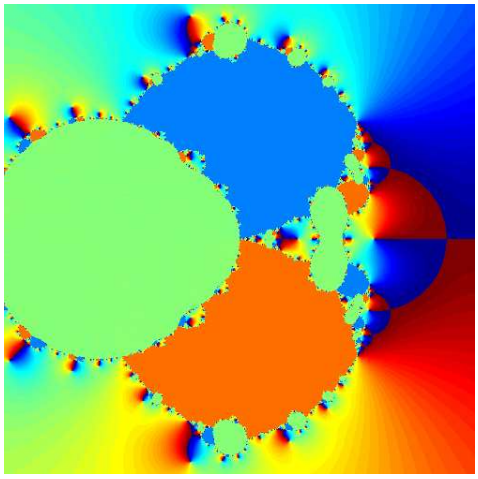

FIG. 6.5. *The basins of attraction of constrained Rayleigh quotient iteration are shown for the multiple root case. Characteristic fractal nature of the boundary between the basins of attraction can be observed again.*

polynomial eigenvalues as affected by the node placement.

6.3. Basin of Attraction: Unconstrained Method. We first considered the same polynomial again, i.e. $z^4 - 1$, sampled at the symmetric points $z = \{0, 1+i, 1-i, -1+i, -1-i\}$. We observed that convergence with the unconstrained method, unlike that of the constrained method, is almost always attained. Convergence is also ultimately cubic, not quadratic, and hence though the cost per iteration is higher—though still only $O(n)$—the number of iterations is quite a bit smaller, leading to an overall faster computation(Figure 6.6).

We then considered the polynomial $(z-1)^2(z^2+1)$, sampled at the symmetric points $z = \{0, 1+i, 1-i, -1+i, -1-i\}$. We observed the same thing as above. Even in the multiple-root case, the convergence is cubic(Figure 6.7).

7. Conclusions. We have shown that two-sided Rayleigh Quotient Iteration is a viable method for the computation of polynomial eigenvalues for matrix polynomials given in the Lagrange basis, that is, by values. The method can be made to be fast, by using the structure of the companion matrix pencil, and is convergent for almost all starting guesses. The method takes $O(ns^2+s^3)$ flops to get one polynomial eigenvalue, using $O(ns^2)$ storage. For ns eigenvalues, the cost is $O(n^2s^4)$.

The method is apparently numerically stable, and deflation can be stabilised by applying the NPLD procedure of [34].

We have also shown that the use of the asymptotically correct form of the eigenvectors *slows* the iteration, making each iteration only quadratically, not cubically, convergent, and destroys the global convergence as well.

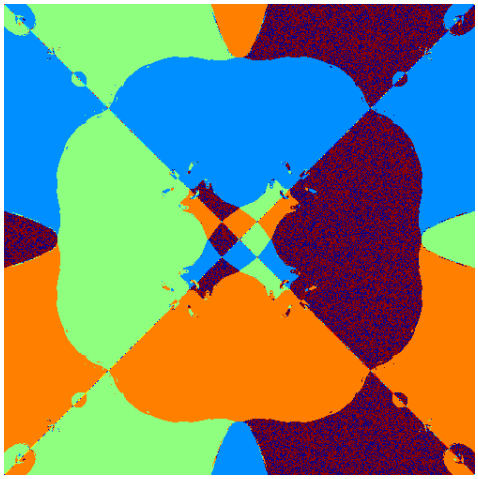

FIG. 6.6. *The basins of attraction of unconstrained Rayleigh quotient iteration are shown. Note that the boundary between the basins of attraction seems rather smoother than in the constrained case.*

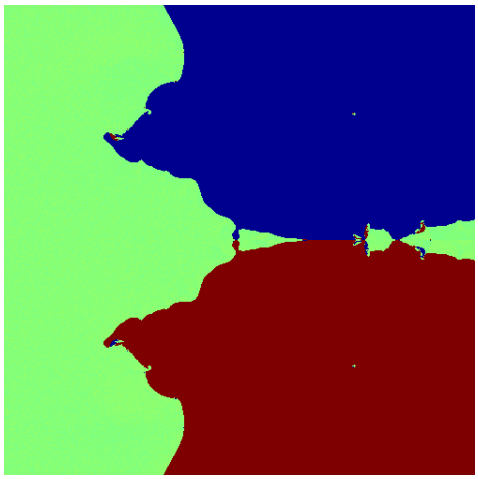

FIG. 6.7. *The basins of attraction of unconstrained Rayleigh quotient iteration are shown for the multiple roots case. The boundary between the basins of attraction seems much smoother than in the constrained case.*

REFERENCES

[1] Amirhossein Amiraslani. Dividing polynomials when you only know their values. In Laureano Gonzalez-Vega and Tomas Recio, editors, *Proceedings EACA*, pages 5–10, June 2004.

[2] Amirhossein Amiraslani, Robert M. Corless, Laureano Gonzalez-Vega, and Azar Shakoori. Polynomial algebra by values. Technical Report TR-04-01, Ontario Research Centre for Computer Algebra, http:// www.orcca.on.ca/ TechReports, January 2004.

[3] Robert M. Corless. Generalized companion matrices in the Lagrange basis. In Laureano

Gonzalez-Vega and Tomas Recio, editors, *Proceedings EACA*, pages 317–322, June 2004.
[4] Azar Shakoori. The Bézout matrix in the Lagrange basis. In Laureano Gonzalez-Vega and Tomas Recio, editors, *Proceedings EACA*, pages 295–299, June 2004.
[5] Steven Fortune. Polynomial root finding using iterated eigenvalue computation. In Bernard Mourrain, editor, *Proc. ISSAC*, pages 121–128, London, Canada, 2001. ACM.
[6] Brian T. Smith. Error bounds for zeros of a polynomial based upon Gerschgorin's theorem. *Journal of the Association for Computing Machinery*, 17(4):661–674, October 1970.
[7] T. Hermann. On the stability of polynomial transformations between Taylor, Bézier, and Hermite forms. *Numerical Algorithms*, 13:307–320, 1996.
[8] Jean-Paul Berrut and Lloyd N. Trefethen. Barycentric Lagrange interpolation. *SIAM Review*, 46(3):501–517, 2004.
[9] Nicholas J. Higham. The numerical stability of barycentric Lagrange interpolation. *IMA Journal of Numerical Analysis*, 24:547–556, 2004.
[10] I. Gohberg, P. Lancaster, and L. Rodman. *Matrix Polynomials*. Academic Press, 1982.
[11] Françoise Tisseur and Nicholas J. Higham. Structured pseudospectra for polynomial eigenvalue problems, with applications. *SIAM J. Matrix Anal. Appl.*, 23(1):187–208, 2001.
[12] Peter Lancaster and P. Psarrakos. On the pseudospectra of matrix polynomials. *SIAM J. Matrix Anal. and Appl.*, 27:115–129, 2005.
[13] Nicholas J. Higham, D. Steven Mackey, and Franǫise Tisseur. The conditioning of linearizations of matrix polynomials. *SIAM J. Matrix Anal. Appl.*, to appear. Revised October 2005.
[14] Gene H. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, 15:318–334, 1973.
[15] K.-C. Toh and Lloyd N. Trefethen. Pseudozeros of polynomials and pseudospectra of companion matrices. *Numerische Mathematik*, 68:403–425, 1994.
[16] P. Lancaster. A generalized Rayleigh quotient iteration for lambda-matrices. *Archive for Rational Mechanics and Analysis*, 8:309–322, 1961.
[17] Dario A. Bini, Luca Gemignani, and Victor Ya. Pan. Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations. *Numer. Math.*, 100:373–408, 2005.
[18] B.N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice Hall, 1980.
[19] James W. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.
[20] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1996.
[21] A. M. Ostrowski. On the convergence of the Rayleigh quotient iteration for the computation of the characteristic roots and vectors. I, II. *Arch. Rational Mech. Anal. 1 (1958), 233–241*, 2:423–428, 1958/1959.
[22] B. N. Parlett. The Rayleigh quotient iteration and some generalizations for nonnormal matrices. *Math. Comp.*, 28:679–693, 1974.
[23] D.P. O'Leary and G.W. Stewart. On the convergence of a new Rayleigh quotient method with applications to large eigenproblems. *Electron. Trans. Numer. Anal.*, 7:182–189 (electronic), 1998.
[24] H. Schwetlick and R. Lösche. A generalized Rayleigh quotient iteration for computing simple eigenvalues of nonnormal matrices. *ZAMM Z. Angew. Math. Mech.*, 80(1):9–25, 2000.
[25] S. H. Crandall. Iterative Procedures Related to Relaxation Methods for Eigenvalue Problems. *Royal Society of London Proceedings Series A*, 207:416–423, 1951.
[26] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.
[27] F.R. Gantmacher. *The Theory of Matrices*, volume 1. Chelsea, 1959.
[28] Françoise Tisseur and K. Meerbergen. The quadratic eigenvalue problem. *SIAM Rev*, 43:235–286, 2001.
[29] Robert M. Corless. On a generalized companion matrix pencil for matrix polynomials expressed in the Lagrange basis. In Dongming Wang and Lihong Zhi, editors, *Symbolic-Numeric Computation*, page to appear. Birkhauser, 2006.
[30] Robert M. Corless. The reducing subspace at infinity for the generalized companion matrix in the Lagrange basis. *in preparation*, 2006.
[31] Ren-cang Li. Solving secular equations stably and efficiently. Technical Report CSD-94-851, University of California at Berkeley, Computer Science Division, 1994. LAPACK working

notes 89, (1993).
[32] Edward R. Vrscay and William J. Gilbert. Extraneous fixed points, basin boundaries and chaotic dynamics for Schröder and König rational iteration functions. *Numerische Mathematik*, 52:1–16, 1988.
[33] Alan Edelman and H. Murakami. Polynomial roots from companion matrix eigenvalues. *Mathematics of Computation*, 64(210):763–776, April 1995.
[34] Robert M. Corless and Nargol Rezvani. The nearest polynomial of lower degree. *submitted*, 2006. http://www.orcca.on.ca/TechReports/2006/TR-06-03.html.
[35] Amirhossein Amiraslani. *Algorithms for Matrices, Polynomials, and Matrix Polynomials*. PhD thesis, University of Western Ontario, London, Canada, May 2006.
[36] Ren-cang Li. Compute multiple nonlinear eigenvalues. *J. Comp. Math.*, 10:1–20, 1992.