SIAM J. MATRIX ANAL. APPL.
Vol. 34, No. 3, pp. 1277–1300

FAST REDUCTION OF GENERALIZED COMPANION MATRIX PAIRS FOR BARYCENTRIC LAGRANGE INTERPOLANTS*

PIERS W. LAWRENCE†

Abstract. For a barycentric Lagrange interpolant $p(z)$, the roots of $p(z)$ are exactly the eigenvalues of a generalized companion matrix pair $(\mathbf{A}, \mathbf{B})$. For real interpolation nodes, the matrix pair $(\mathbf{A}, \mathbf{B})$ can be reduced to a pair $(\mathbf{H}, \mathbf{B})$, where $\mathbf{H}$ has tridiagonal plus rank-one structure. In this paper we propose two fast algorithms for reducing the pair $(\mathbf{A}, \mathbf{B})$ to Hessenberg-triangular form. The matrix pair $(\mathbf{A}, \mathbf{B})$ has two spurious infinite eigenvalues, and if the leading coefficients of the interpolant are zero, there will also be other infinite eigenvalues. We propose tools for detecting when the leading coefficients of $p(z)$ are zero, and describe a procedure to deflate all of the infinite eigenvalues from the reduced matrix pair $(\mathbf{H}, \mathbf{B})$, while still maintaining the tridiagonal plus rank-one structure of the resulting standard eigenvalue problem. Since fast QR algorithms exist for such structured matrices, the complexity of computing the roots of barycentric Lagrange interpolants could be significantly reduced.

Key words. polynomial interpolation, Lagrange interpolation, barycentric formula, generalized companion matrices, polynomial roots, eigenvalue problem, semiseparable matrices

AMS subject classifications. 65H04, 65H17, 65F15, 65D05

DOI. 10.1137/130904508

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, [5, 6, 11, 20].

In this paper we consider polynomial interpolants in the Lagrange basis, expressed in barycentric form. Berrut and Trefethen [3] present a comprehensive review of such polynomial interpolants.

Given a set of $n+1$ distinct interpolation nodes $\{x_0, \ldots, x_n\}$, with corresponding values $\{f_0, \ldots, f_n\}$, the barycentric weights w_j are defined by

$$w_j = \left(\prod_{\substack{k=0 \\ k \neq j}}^{n} (x_j - x_k) \right)^{-1}, \quad 0 \leq j \leq n\,. \tag{1.1}$$

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

$$p(z) = \prod_{i=0}^{n} (z - x_i) \sum_{j=0}^{n} \frac{w_j}{(z - x_j)} f_j\,. \tag{1.2}$$

*Received by the editors January 2, 2013; accepted for publication (in revised form) by M. Van Barel June 13, 2013; published electronically September 3, 2013.
http://www.siam.org/journals/simax/34-3/90450.html

†Department of Applied Mathematics, The University of Western Ontario, London, Ontario N6A 5B7, Canada (plawren@uwo.ca).

Equation (1.2) is known as the "first form of the barycentric interpolation formula" [19] or the "modified Lagrange formula" [12]. The "second (true) form of the barycentric formula" [19] is

$$p(z) = \frac{\displaystyle\sum_{j=0}^{n} \frac{w_j}{(z - x_j)} f_j}{\displaystyle\sum_{j=0}^{n} \frac{w_j}{(z - x_j)}}, \tag{1.3}$$

which is constructed by dividing (1.2) by the interpolant of the constant function 1 at the same nodes, and by canceling the factor $\prod_{i=0}^{n} (z - x_i)$.

As was first shown in [5], the roots of the interpolating polynomial $p(z)$, as defined by (1.2), are exactly the eigenvalues of a generalized companion matrix pair

$$(\mathbf{A}, \mathbf{B}) = \left(\begin{bmatrix} 0 & -\mathbf{f}^T \\ \mathbf{w} & \mathbf{D} \end{bmatrix}, \begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix} \right), \tag{1.4}$$

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

$$\mathbf{D} = \begin{bmatrix} x_0 & & \\ & \ddots & \\ & & x_n \end{bmatrix}. \tag{1.5}$$

This can be shown by applying Schur's determinant formula:

$$\det(z\mathbf{B} - \mathbf{A}) = \det \begin{bmatrix} 0 & \mathbf{f}^T \\ -\mathbf{w} & z\mathbf{I} - \mathbf{D} \end{bmatrix} \tag{1.6}$$

$$= \det(z\mathbf{I} - \mathbf{D}) \det\left(0 + \mathbf{f}^T (z\mathbf{I} - \mathbf{D})^{-1} \mathbf{w}\right) \tag{1.7}$$

$$= \prod_{i=0}^{n} (z - x_i) \sum_{j=0}^{n} \frac{w_j f_j}{(z - x_j)} = p(z). \tag{1.8}$$

Thus, the eigenvalues of the matrix pair $(\mathbf{A}, \mathbf{B})$ are exactly the zeros of $p(z)$. Furthermore, computing the roots of polynomial interpolants via the eigenvalues of the matrix pair $(\mathbf{A}, \mathbf{B})$ is numerically stable [13].

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 section 7.

The second form of the barycentric interpolation formula has a remarkable feature [2]: the interpolating property is satisfied independently of the choice of weights w_j, as long as they are all nonzero. The choice of weights (1.1) 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 [2]), 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.

2. Reduction of $(\mathbf{A}, \mathbf{B})$ to Hessenberg-triangular form. Through numerical experimentation we found that for real interpolation nodes, the initial reduction of $(\mathbf{A}, \mathbf{B})$ to Hessenberg-triangular form always seemed 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 2.1. *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 [27, p. 138] states that there exists a unitary matrix $\mathbf{Q}$ whose first column is proportional to $\mathbf{e}_1$ such that $\mathbf{H} = \mathbf{Q}^*\mathbf{A}\mathbf{Q}$ is upper Hessenberg. Partition $\mathbf{Q}$ as

$$\mathbf{Q} = \begin{bmatrix} 1 & \\ & \mathbf{Q}_1 \end{bmatrix}. \tag{2.1}$$

Explicitly, $\mathbf{H}$ is

$$\mathbf{H} = \mathbf{Q}^*\mathbf{A}\mathbf{Q} = \begin{bmatrix} 0 & -\mathbf{f}^T\mathbf{Q}_1 \\ \mathbf{Q}_1^*\mathbf{w} & \mathbf{Q}_1^*\mathbf{D}\mathbf{Q}_1 \end{bmatrix}. \tag{2.2}$$

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

$$\mathbf{T} = \begin{bmatrix} 0 & t_0\mathbf{e}_1^T \\ t_0\mathbf{e}_1 & \mathbf{T}_1 \end{bmatrix} \tag{2.3}$$

and

$$\mathbf{c}^T = \begin{bmatrix} 0 & -t_0\mathbf{e}_1^T - \mathbf{f}^T\mathbf{Q}_1 \end{bmatrix}. \tag{2.4}$$

Then we can rewrite (2.2) as

$$\mathbf{Q}^*\mathbf{A}\mathbf{Q} = \mathbf{T} + \mathbf{e}_1\mathbf{c}^T. \tag{2.5}$$

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

$$\mathbf{Q}^*\mathbf{B}\mathbf{Q} = \begin{bmatrix} 1 & \\ & \mathbf{Q}_1^* \end{bmatrix} \begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix} \begin{bmatrix} 1 & \\ & \mathbf{Q}_1 \end{bmatrix} = \begin{bmatrix} 0 & \\ & \mathbf{Q}_1^*\mathbf{Q}_1 \end{bmatrix} = \mathbf{B}. \tag{2.6}$$

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. ∎

3. A fast reduction to Hessenberg form. 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 [10]. 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 (2.1) such that

$$\mathbf{Q}^*\mathbf{A}\mathbf{Q} = \mathbf{T} + \mathbf{e}_1\mathbf{c}^T. \tag{3.1}$$

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

$$\mathbf{Q}_1 = \begin{bmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \cdots & \mathbf{q}_n \end{bmatrix} . \tag{3.2}$$

The first column of (3.1) requires that $\mathbf{Q}_1^* \mathbf{w} = t_0 \mathbf{e}_1$, and hence we may immediately identify that

$$\mathbf{q}_0 = \frac{\mathbf{w}}{t_0} . \tag{3.3}$$

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

$$\mathbf{T}_1 = \begin{bmatrix} d_0 & t_1 & & & \\ t_1 & d_1 & \ddots & & \\ & \ddots & \ddots & t_{n-1} & \\ & & t_{n-1} & d_{n-1} & t_n \\ & & & t_n & d_n \end{bmatrix} , \tag{3.4}$$

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

$$\begin{bmatrix} \mathbf{D}\mathbf{q}_0 & \mathbf{D}\mathbf{q}_1 & \cdots & \mathbf{D}\mathbf{q}_n \end{bmatrix} = \begin{bmatrix} d_0\mathbf{q}_0 + t_1\mathbf{q}_1 & t_1\mathbf{q}_0 + d_1\mathbf{q}_1 + t_2\mathbf{q}_2 & \cdots & t_n\mathbf{q}_{n-1} + d_n\mathbf{q}_n \end{bmatrix} , \tag{3.5}$$

the first column of which gives the equation

$$\mathbf{D}\mathbf{q}_0 = d_0\mathbf{q}_0 + t_1\mathbf{q}_1 . \tag{3.6}$$

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

$$d_0 = \mathbf{q}_0^* \mathbf{D}\mathbf{q}_0 . \tag{3.7}$$

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

$$\mathbf{q}_1 = \frac{1}{t_1} \left(\mathbf{D} - d_0\mathbf{I}\right) \mathbf{q}_0 , \tag{3.8}$$

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 (3.5) for $1 \leq i \leq n-1$ is

$$\mathbf{D}\mathbf{q}_i = t_i\mathbf{q}_{i-1} + d_i\mathbf{q}_i + t_{i+1}\mathbf{q}_{i+1} . \tag{3.9}$$

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

$$d_i = \mathbf{q}_i^* \mathbf{D}\mathbf{q}_i . \tag{3.10}$$

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

$$\mathbf{q}_{i+1} = \frac{1}{t_{i+1}} \left(\left(\mathbf{D} - d_i\mathbf{I}\right) \mathbf{q}_i - t_i\mathbf{q}_{i-1} \right) , \tag{3.11}$$

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 (3.5) is

$$\mathbf{D}\mathbf{q}_n = t_n\mathbf{q}_n + d_n\mathbf{q}_n . \tag{3.12}$$

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

$$d_n = \mathbf{q}_n^* \mathbf{D} \mathbf{q}_n \,. \tag{3.13}$$

Algorithm 1 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 $O(n^3)$).

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 $\mathbf{t}$, $\mathbf{d}$, $\mathbf{c}$

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 $\mathbf{q}_0, \ldots, \mathbf{q}_n$ for the Krylov subspace $\mathcal{K}_{n+1}(\mathbf{D}, \mathbf{w}) = \operatorname{span}\{\mathbf{w}, \mathbf{D}\mathbf{w}, \ldots, \mathbf{D}^n \mathbf{w}\}$. One of the potential difficulties which can arise when applying the symmetric Lanczos process [27, 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 3.1. *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

$$\mathbf{V} = \begin{bmatrix} w_0 & w_0 x_0 & \cdots & w_0 x_0^n \\ w_1 & w_1 x_1 & \cdots & w_1 x_1^n \\ \vdots & \vdots & \ddots & \vdots \\ w_n & w_n x_n & \cdots & w_n x_n^n \end{bmatrix} = \begin{bmatrix} w_0 & & & \\ & \ddots & & \\ & & & w_n \end{bmatrix} \begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^n \end{bmatrix}. \tag{3.14}$$

The determinant of $\mathbf{V}$ is

$$\det \mathbf{V} = \prod_{i=0}^{n} w_i \prod_{0 \le j < k \le n} (x_k - x_j)\,, \tag{3.15}$$

which is nonzero as long as no w_i is equal to zero, and the x_is 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 3.2. *Suppose* $\mathbf{w}$, $\mathbf{D}\mathbf{w}, \ldots, \mathbf{D}^n\mathbf{w}$ *are linearly independent, and the vectors* $\mathbf{q}_0, \ldots, \mathbf{q}_n$ *are generated by Algorithm* 1. *Then*

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

$$\operatorname{span}\{\mathbf{q}_0, \ldots, \mathbf{q}_j\} = \mathcal{K}_{j+1}(\mathbf{D}, \mathbf{w}), \quad 0 \le j \le n\,; \tag{3.16}$$

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

Proof. Lemma 3.1 showed that $\mathbf{w}, \mathbf{D}\mathbf{w}, \ldots, \mathbf{D}^n\mathbf{w}$ 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 [26, p. 436]. ∎

4. Alternative reduction via Givens rotations. Since the reduction process described in section 3 has the potential for losing orthogonality of the transformation matrix $\mathbf{Q}_1$ [27, 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 reducing 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 let $\mathbf{A}_i = \mathbf{G}_i^* \cdots \mathbf{G}_1^* \mathbf{A} \mathbf{G}_1 \cdots \mathbf{G}_i$ 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

$$\mathbf{A}_1 = \mathbf{G}_1^* \mathbf{A} \mathbf{G}_1 = \begin{bmatrix} 0 & f_0 & \cdots & f_{n-2} & \times & \times \\ w_0 & x_0 & & & & \\ \vdots & & \ddots & & & \\ w_{n-2} & & & x_{n-2} & & \\ \times & & & & \times & \times \\ 0 & & & & \times & \times \end{bmatrix}. \tag{4.1}$$

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

$$\mathbf{A}_2 = \mathbf{G}_2^* \mathbf{G}_1^* \mathbf{A} \mathbf{G}_1 \mathbf{G}_2 = \begin{bmatrix} 0 & f_0 & \cdots & f_{n-3} & \times & \times & \times \\ w_0 & x_0 & & & & & \\ \vdots & & \ddots & & & & \\ w_{n-3} & & & x_{n-3} & & & \\ \times & & & & \times & \times & \times \\ 0 & & & & \times & \times & \times \\ 0 & & & & \times & \times & \times \end{bmatrix}. \tag{4.2}$$

Note again that the trailing 3 by 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

$$\mathbf{A}_3 = \begin{bmatrix} 0 & f_0 & \cdots & f_{n-3} & \times & \times & \times \\ w_0 & x_0 & & & & & \\ \vdots & & \ddots & & & & \\ w_{n-3} & & & x_{n-3} & & & \\ \times & & & & \times & \times & \\ 0 & & & & \times & \times & \times \\ 0 & & & & & \times & \times \end{bmatrix}. \tag{4.3}$$

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

$$\mathbf{A}_4 = \begin{bmatrix} 0 & f_0 & \cdots & f_{n-4} & \times & \times & \times & \times \\ w_0 & x_0 & & & & & & \\ \vdots & & \ddots & & & & & \\ w_{n-4} & & & x_{n-4} & & & & \\ \times & & & & \times & \times & \times & \\ 0 & & & & \times & \times & \times & \\ 0 & & & & \times & \times & \times & \times \\ 0 & & & & & & \times & \times \end{bmatrix}. \tag{4.4}$$

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

$$\mathbf{A}_5 = \begin{bmatrix} 0 & f_0 & \cdots & f_{n-4} & \times & \times & \times & \times \\ w_0 & x_0 & & & & & & \\ \vdots & & \ddots & & & & & \\ w_{n-4} & & & x_{n-4} & & & & \\ \times & & & & \times & \times & & \\ 0 & & & & \times & \times & \times & \times \\ 0 & & & & & \times & \times & \times \\ 0 & & & & & \times & \times & \times \end{bmatrix}. \tag{4.5}$$

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:

$$\mathbf{A}_6 = \begin{bmatrix} 0 & f_0 & \cdots & f_{n-4} & \times & \times & \times & \times \\ w_0 & x_0 & & & & & & \\ \vdots & & \ddots & & & & & \\ w_{n-4} & & & x_{n-4} & & & & \\ \times & & & & \times & \times & & \\ 0 & & & & \times & \times & \times & \\ 0 & & & & & \times & \times & \times \\ 0 & & & & & & \times & \times \end{bmatrix}. \tag{4.6}$$

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:

$$\mathbf{A}_{\frac{n(n+1)}{2}} = \begin{bmatrix} 0 & \times & \times & \times & \cdots & \times & \times \\ \times & \times & \times & & & & \\ & \times & \times & \times & & & \\ & & \times & \times & \ddots & & \\ & & & \ddots & \ddots & \times & \\ & & & & \times & \times & \times \\ & & & & & \times & \times \end{bmatrix}. \tag{4.7}$$

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 nine elements of the matrix when annihilating an element of the first column, and eight 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 four vectors: $\mathbf{w}$, $\mathbf{f}$, the diagonal elements $\mathbf{d}$, and the subdiagonal elements $\mathbf{t}$.

LEMMA 4.1. *The reduction algorithm described in this section results in essentially the same matrix as Algorithm* 1 *proposed in section* 3. *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 resulting from the Givens reduction described in this section.*

Proof. Theorem 5.7.24 of [26, 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. ∎

5. Complex nodes. The reduction processes proposed in sections 3 and 4 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 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 nonunitary transformations 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 section 3 was equivalent to the symmetric Lanczos process. Thus, for complex interpolation nodes the reduction transforms to the complex symmetric Lanczos process; see, for example, [9].

To convert the reduction algorithm presented in section 4 to work for complex interpolation nodes, we need to apply complex orthogonal Givens-like matrices [15] in place of the unitary Givens rotations to retain the complex symmetric structure of the trailing submatrix when annihilating elements of the matrix.

6. Zero leading coefficients. Throughout this paper, we have not specified that the leading coefficients 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).

If the first m leading coefficients 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 $K_j = \{k_1, \ldots, k_j\}$ is a set of unique integers, then we may restate the barycentric interpolation formula as

$$p(z) = \prod_{\substack{i=0 \\ i \notin K_j}}^{n} (z - x_i) \sum_{\substack{\ell=0 \\ \ell \notin K_j}}^{n} \left(\prod_{k \in K_j} (x_\ell - x_k) \frac{w_\ell f_\ell}{(z - x_\ell)} \right) \tag{6.1}$$

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

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

THEOREM 6.1. *If the leading coefficients* $\left[z^{n-j}\right](p(z)) = 0$ *for all* j, $0 \le j \le m - 1$, *then* $[z^{n-m}](p(z)) = \mathbf{f}^T\mathbf{D}^m\mathbf{w}$. *That is, if the first* m *leading coefficients 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 (1.2) can be written as

$$p(z) = \sum_{\ell=0}^{n} \left(\prod_{\substack{k=0 \\ k \neq \ell}}^{n} (z - x_k) \right) w_\ell f_\ell \,, \tag{6.2}$$

whose leading coefficient is

$$[z^n](p(z)) = \sum_{\ell=0}^{n} w_\ell f_\ell = \mathbf{f}^T\mathbf{w}\,. \tag{6.3}$$

The next leading coefficient is given by

$$[z^{n-1}](p(z)) = -\sum_{\ell=0}^{n} w_\ell f_\ell \sum_{\substack{k=0\\k\neq\ell}}^{n} x_k \tag{6.4}$$

$$= -\sum_{\ell=0}^{n} w_\ell f_\ell \left(-x_\ell + \sum_{k=0}^{n} x_k\right) \tag{6.5}$$

$$= \sum_{\ell=0}^{n} w_\ell f_\ell x_\ell - \left(\sum_{j=0}^{n} w_j f_j\right) \sum_{k=0}^{n} x_k \tag{6.6}$$

$$= \mathbf{f}^T\mathbf{D}\mathbf{w} - \mathbf{f}^T\mathbf{w} \sum_{k=0}^{n} x_k\,. \tag{6.7}$$

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 $1 \le m \le M-1$. Thus, if $[z^{n-j}](p(z)) = 0$ for all j, $0 \le j \le M-1$, then $[z^{n-M}](p(z)) = \mathbf{f}^T\mathbf{D}^M\mathbf{w}$. Now suppose that additionally $[z^{n-M}](p(z)) = \mathbf{f}^T\mathbf{D}^M\mathbf{w} = 0$. The $(M+2)$nd leading coefficient of $p(z)$ can be obtained from (6.1):

$$\left[z^{n-(M+1)}\right](p(z)) = \sum_{\ell=0}^{n} q_{m+1}(x_\ell) w_\ell f_\ell \tag{6.8}$$

$$= \mathbf{f}^T q_{m+1}(\mathbf{D})\mathbf{w}\,, \tag{6.9}$$

where $q_j(z)$ is the monic polynomial

$$q_j(z) = \prod_{k\in K_j} (z - x_k)\,. \tag{6.10}$$

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

$$q_{M+1}(\mathbf{D}) = \mathbf{D}^{M+1} + b_M\mathbf{D}^M + \cdots + b_0\mathbf{I}\,, \tag{6.11}$$

where all of the coefficients b_j are expressions in terms of x_k for $k \in K_{M+1}$. Substituting this expansion into (6.9) and expanding the resulting expression, we obtain

$$\left[z^{n-(M+1)}\right](p(z)) = \mathbf{f}^T\mathbf{D}^{M+1}\mathbf{w} + b_M\mathbf{f}^T\mathbf{D}^M\mathbf{w} + \cdots + b_0\mathbf{f}^T\mathbf{w}\,. \tag{6.12}$$

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 (6.12) reduces to

$$\left[z^{n-(M+1)}\right](p(z)) = \mathbf{f}^T\mathbf{D}^{M+1}\mathbf{w}\,. \tag{6.13}$$

Thus, we have proved that if $\left[z^{n-j}\right](p(z)) = 0$ 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 6.1 can be applied to the reduction algorithm proposed in section 3, so that we may determine some more information about the vector $\mathbf{c}_1$ produced from the reduction algorithm.

COROLLARY 6.2. *For the reduction process described in section* 3, *if* $\left[z^{n-j}\right](p(z)) = 0$ *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 $\left[z^{n-j}\right](p(z)) = 0$ for all j, $0 \le j \le m-1$, then Theorem 6.1 implies that

$$\mathbf{f}^T\mathbf{D}^j\mathbf{w} = 0 \tag{6.14}$$

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

$$\begin{bmatrix} c_0 & \cdots & c_{m-1} \end{bmatrix} = -\mathbf{f}^T \begin{bmatrix} \mathbf{q}_0 & \cdots & \mathbf{q}_{m-1} \end{bmatrix} - \|\mathbf{w}\|\mathbf{e}_1^T\,. \tag{6.15}$$

Theorem 3.2 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 (6.14) imply that the vector $\mathbf{f}$ is contained in the orthogonal complement $\mathcal{K}_m(\mathbf{D}, \mathbf{w})^\perp$. Thus, (6.15) reduces to

$$\begin{bmatrix} c_0 & c_1 & \cdots & c_{m-1} \end{bmatrix} = -\|\mathbf{w}\|\mathbf{e}_1^T\,, \tag{6.16}$$

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

7. Deflation of infinite eigenvalues. The matrices in the pair $(\mathbf{A}, \mathbf{B})$ have dimension $(n+2)$ by $(n+2)$, 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, infinite eigenvalues can be deflated from a matrix pair by transforming the pair to the form

$$\left(\widehat{\mathbf{A}}, \widehat{\mathbf{B}}\right) = \left(\begin{bmatrix} \mathbf{R}_\infty & \times \\ & \mathbf{H}_f \end{bmatrix}, \begin{bmatrix} \mathbf{J}_\infty & \times \\ & \mathbf{T}_f \end{bmatrix}\right), \tag{7.1}$$

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 (unreduced), 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 sections 3 and 4, 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 (7.1) 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 sections 3 or 4, we can reduce the matrix pair $(\mathbf{A}, \mathbf{B})$ to the pair

$$(\mathbf{H}, \mathbf{B}) = (\mathbf{T} + \mathbf{e}_1 \mathbf{c}^T, \mathbf{B}), \tag{7.2}$$

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

$$\begin{bmatrix} 0 & t_0 + c_0 & \mathbf{c}_2^T \\ t_0 & d_0 & t_1 \mathbf{e}_1 \\ \mathbf{0} & t_1 \mathbf{e}_1 & \mathbf{T}_2 \end{bmatrix}, \tag{7.3}$$

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

$$(\mathbf{G}_1^* \mathbf{H}, \mathbf{G}_1^* \mathbf{B}) = \left(\begin{bmatrix} t_0 & d_0 & t_1 \mathbf{e}_1 \\ 0 & t_0 + c_0 & \mathbf{c}_2^T \\ & t_1 \mathbf{e}_1 & \mathbf{T}_2 \end{bmatrix}, \begin{bmatrix} 0 & 1 & \\ & 0 & \\ & & \mathbf{I} \end{bmatrix} \right). \tag{7.4}$$

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

$$(\mathbf{H}_1, \mathbf{B}_1) = \left(\begin{bmatrix} t_0 + c_0 & \mathbf{c}_2^T \\ t_1 \mathbf{e}_1 & \mathbf{T}_2 \end{bmatrix}, \begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix} \right). \tag{7.5}$$

Remark 4. If the first m leading coefficients of $p(z)$ are zero, Corollary 6.2 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

$$\mathbf{H}_m = \begin{bmatrix} c_m & c_{m+1} & c_{m+2} & \cdots & c_n \\ t_{m+1} & d_{m+1} & t_{m+2} & & \\ & t_{m+2} & d_{m+2} & \ddots & \\ & & \ddots & \ddots & t_n \\ & & & t_n & d_n \end{bmatrix}, \quad \mathbf{B}_m = \begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix}. \tag{7.6}$$

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

$$\mathbf{G}_2^* = \begin{bmatrix} \overline{h} & \overline{g} & \\ -g & h & \\ & & \mathbf{I} \end{bmatrix}, \tag{7.7}$$

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

(7.8)
$$\mathbf{G}_2^*\mathbf{H}_1 = \begin{bmatrix} \overline{h}(t_0+c_0)+\overline{g}t_1 & \overline{h}c_1+\overline{g}d_1 & \overline{h}c_2+\overline{g}t_2 & \overline{h}c_3 & \cdots & \overline{h}c_n \\ 0 & -gc_1+hd_1 & -gc_2+ht_2 & -gc_3 & \cdots & -gc_n \\ & t_2 & d_2 & t_3 & & \\ & & t_3 & d_3 & \ddots & \\ & & & \ddots & \ddots & t_n \\ & & & & t_n & d_n \end{bmatrix},$$

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

$$\mathbf{G}_2^*\mathbf{B}_1 = \begin{bmatrix} 0 & \overline{g} & \\ & h & \\ & & \mathbf{I} \end{bmatrix}. \tag{7.9}$$

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

$$\mathbf{H}_2 = \begin{bmatrix} -gc_1+hd_1 & -gc_2+ht_2 & -gc_3 & \cdots & -gc_n \\ t_2 & d_2 & t_3 & & \\ & t_3 & d_3 & \ddots & \\ & & \ddots & \ddots & t_n \\ & & & t_n & d_n \end{bmatrix} \tag{7.10}$$

and

$$\mathbf{B}_2 = \begin{bmatrix} h & \\ & \mathbf{I} \end{bmatrix}. \tag{7.11}$$

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

$$\mathbf{H}_s = \mathbf{T}_2 + \beta \mathbf{e}_1 \mathbf{c}_2^T. \tag{7.12}$$

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 [4, 23, 24], and we believe that these methods should be very competitive once the matrix pair is reduced.

8. Connection to the Chebyshev colleague matrix. For polynomials expressed by their values at Chebyshev points of the second kind, 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 [1, 21]. The roots of the polynomial expressed as a finite Chebyshev series

$$p(z) = \sum_{k=0}^{n} a_k T_k(z) \tag{8.1}$$

can be found by computing the eigenvalues of the colleague matrix, discovered independently by Specht [20] and by Good [11]. 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:

$$\mathbf{C} = \begin{bmatrix} 0 & \frac{1}{2} & & & \\ \frac{1}{2} & \ddots & \ddots & & \\ & \ddots & 0 & \frac{1}{2} & \\ & & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ & & & \frac{1}{\sqrt{2}} & 0 \end{bmatrix} - \frac{1}{2a_n}\mathbf{e}_1 \begin{bmatrix} a_{n-1} & \cdots & a_1 & \sqrt{2}a_0 \end{bmatrix}. \tag{8.2}$$

In sections 3, 4, and 7, we were able to construct two nonsingular matrices $\mathbf{U}$ and $\mathbf{V}$ such that

$$(\mathbf{UAV}, \mathbf{UBV}) = \left(\begin{bmatrix} \mathbf{R}_\infty & \times \\ & \mathbf{H}_s \end{bmatrix}, \begin{bmatrix} \mathbf{J}_\infty & \times \\ & \mathbf{I} \end{bmatrix} \right), \tag{8.3}$$

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

$$\left(\widehat{\mathbf{U}}\mathbf{A}\widehat{\mathbf{V}}, \widehat{\mathbf{U}}\mathbf{B}\widehat{\mathbf{V}}\right) = \left(\begin{bmatrix} \mathbf{R}_\infty & \times \\ & \mathbf{C} \end{bmatrix}, \begin{bmatrix} \mathbf{J}_\infty & \times \\ & \mathbf{I} \end{bmatrix} \right). \tag{8.4}$$

For Chebyshev points of the second kind, the barycentric weights w_ℓ are defined by [18, 28]

$$w_\ell = \frac{2^{n-1}}{n}(-1)^\ell \delta_\ell\,, \quad \delta_\ell = \begin{cases} 1/2 & \text{if } \ell = 0 \text{ or } \ell = n, \\ 1 & \text{otherwise.} \end{cases} \tag{8.5}$$

Owing to their magnitude for large n, there is the risk of overflow in floating point computations, so usually [3, 12, 28] the weights are multiplied by $n/2^{n-1}$ to give

$$\hat{w}_\ell = (-1)^\ell \delta_\ell\,. \tag{8.6}$$

This rescaling is possible because, for the second form of the barycentric interpolation formula (1.3), 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 (8.3), 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

$$\mathbf{A} = \begin{bmatrix} 0 & -\mathbf{f}^T \\ \widehat{\mathbf{w}} & \mathbf{D} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix}, \tag{8.7}$$

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

$$\mathbf{S} = \begin{bmatrix} 1 & \\ & \mathbf{S}_1 \end{bmatrix} \tag{8.8}$$

and form

$$\hat{\mathbf{A}} = \mathbf{S}^{-1}\mathbf{A}\mathbf{S}\,. \tag{8.9}$$

Note also that $\mathbf{S}^{-1}\mathbf{B}\mathbf{S} = \mathbf{B}$. Applying either of the reduction algorithms described in sections 3 and 4 to $\hat{\mathbf{A}}$ produces a symmetric tridiagonal plus rank-one matrix with tridiagonal part

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & & & & & & \\ 1 & 0 & \frac{1}{\sqrt{2}} & & & & & \\ & \frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & & & & \\ & & \frac{1}{2} & \ddots & \ddots & & & \\ & & & \ddots & 0 & \frac{1}{2} & & \\ & & & & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} & \\ & & & & & \frac{1}{\sqrt{2}} & 0 & \end{bmatrix}. \tag{8.10}$$

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

9. 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 [16, 17], 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 [25] and in [14]. 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 sections 3 and 4 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.

10. Numerical experiments. In this section, we will investigate the accuracy and stability of the two reduction algorithms proposed in sections 3 and 4, as well as that of the deflation procedure proposed in section 7. Furthermore, we will investigate the application of Theorem 6.1 and Corollary 6.2 numerically.

We also compare the algorithms presented here to the algorithm described in [7]. That algorithm reduces a quasiseparable matrix to Hessenberg form in $O(n^2)$ operations.

10.1. 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 $d_i = 0$, $0 \le i \le n$, and

$$t_i = \begin{cases} \|\mathbf{w}\|_2 & \text{if } i = 0, \\ 1/\sqrt{2} & \text{if } i = n, \\ 1/2 & \text{otherwise.} \end{cases} \tag{10.1}$$

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 two reduction processes described in sections 3 and 4.

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

$$x_j = \cos\frac{(2j+1)\pi}{2n+2}\,, \quad 0 \le j \le n\,. \tag{10.2}$$

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

$$f_j = (-1)^j \sin\frac{(2j+1)\pi}{2n+2}\,, \quad 0 \le j \le n\,, \tag{10.3}$$

and the barycentric weights are

$$w_j = \frac{2^n}{(n+1)}(-1)^j \sin\frac{(2j+1)\pi}{2n+2}\,, \quad 0 \le j \le n\,. \tag{10.4}$$

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

$$(\mathbf{A}, \mathbf{B}) = \left(\begin{bmatrix} 0 & -\mathbf{f}^T \\ -\mathbf{f} & \mathbf{D} \end{bmatrix}, \begin{bmatrix} 0 & \\ & \mathbf{I} \end{bmatrix} \right) \tag{10.5}$$

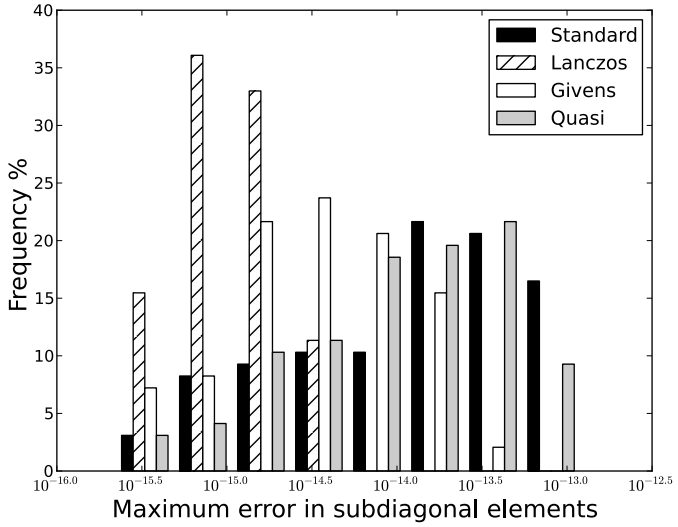


FIG. 10.1. *Distribution of maximum error in the subdiagonal entries of* $\mathbf{T}$.

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

$$\det(z\mathbf{B} - \mathbf{A}) = -\frac{(n+1)}{2^n} T_n(z)\,. \tag{10.6}$$

We measured the accuracy of reducing the matrix pair to Hessenberg form by computing the maximum error in the computed subdiagonal entries, as shown in Figure 10.1. We also measured the maximum forward error in the computed eigenvalues, as shown in Figure 10.2. Four reduction algorithms are compared: the standard Hessenberg-triangular reduction, the Lanczos type reduction of section 3, the Givens type reduction of section 4, and the quasiseparable matrix algorithm proposed in [7]. 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 out 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, the loss in orthogonality appears to be lost at the same rate.

The Givens type reduction performs approximately halfway between the 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.

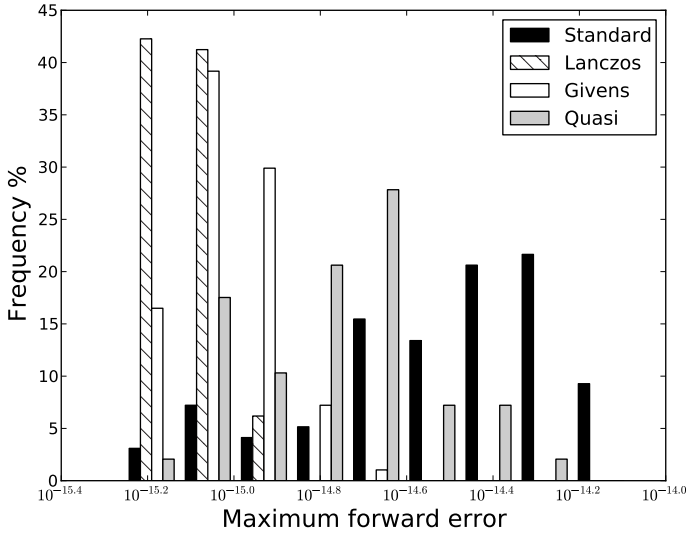


FIG. 10.2. *Maximum forward error in the computed roots from each of the four reduction processes.*

The quasiseparable matrix algorithm of [7] 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, that the algorithm can be applied to a much more general class of matrices than the one considered here.

10.2. Scaled Wilkinson polynomial. Here we investigate the accuracy of computing the roots of a poorly conditioned polynomial, investigated by Wilkinson in [29]. We modify Wilkinson's polynomial to have roots equispaced in the interval $[0, 1]$: the polynomial we investigate is defined by

$$p(z) = \prod_{\ell=1}^{20} \left(z - \frac{\ell}{21} \right). \tag{10.7}$$

We sampled this polynomial at a range of interpolation nodes: equispaced nodes, Chebyshev nodes, and Legendre nodes. Choosing 21 nodes of each distribution on the interval $[1/(2n), 1 - 1/(2n)]$, we applied diagonal scaling 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 Lanczos type, Givens type, and quasiseparable matrix reductions, and then deflated the spurious infinite eigenvalues to obtain the matrices $\mathbf{H}_L$, $\mathbf{H}_G$, and $\mathbf{H}_Q$, respectively. Table 10.1 shows the maximum error between the true roots of $p(z)$ and the computed eigenvalues of $(\mathbf{A}, \mathbf{B})$, $\mathbf{H}_L$, $\mathbf{H}_G$, 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 original matrix pair.

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 10.2, 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 10.1
Maximum error in computed eigenvalues for Wilkinson's polynomial, sampled at different node distributions.

Point distribution	$(\mathbf{A}, \mathbf{B})$	$\mathbf{H}_L$	$\mathbf{H}_G$	$\mathbf{H}_Q$
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 10.2
Measure of loss of orthogonality of vectors $\mathbf{q}_0, \ldots, \mathbf{q}_n$ produced from the Lanczos type reduction process of section 3.

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 reducing the pair $(\mathbf{A}, \mathbf{B})$ 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.

10.3. Polynomials with zero leading coefficients. We will now investigate the numerical application of Theorem 6.1 to detect when the leading coefficients of the interpolant are zero. First we give a very simple example: we will interpolate the polynomial

$$f(z) = z^2 + 4z + 1 \tag{10.8}$$

at seven Chebyshev points of the second kind. Because the first four leading coefficients of the interpolant are zero, by Theorem 6.1 we may compute the first five leading coefficients using $\mathbf{f}^T\mathbf{D}^k\mathbf{w}$ for $0 \le k \le 4$, as shown in Table 10.3. The first four leading coefficients are all less than 32 times machine epsilon ($\varepsilon_M \approx 2.2 \times 10^{-16}$), and the first nonzero leading coefficient is less than 16 times machine epsilon different to the exact value of 1. This example 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.

TABLE 10.3
Computed leading coefficients of the degree six interpolant to (10.8).

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

However, in applications such as the MATLAB package CHEBFUN [22], 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 6.1 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

$$f(z) = 10^{-12}T_9(z) + 10^{-10}T_8(z) + 10^{-8}T_7(z) + 10^{-6}T_6(z) + 10^{-4}T_5(z) + 10^{-2}T_4(z) + T_3(z) + 3T_2(z) - 2T_1(z) - T_0(z)\,, \tag{10.9}$$

where $T_k(z)$ is the kth 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 $0 \le k \le 2$ as shown in Table 10.4. The leading coefficients which should be zero are not very accurate, although in this case the difference between the exact leading coefficient 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 6.1 may not give numerically useful results, Corollary 6.2 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 $\mathbf{c}_1$ are shown in Table 10.5. 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 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

TABLE 10.4
Computed leading coefficients of the degree 11 *interpolant to* (10.9).

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

TABLE 10.5
Elements of $\mathbf{c}_1$ *from Givens reduction algorithm.*

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

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 sections 3 or 4 to determine the first nonzero leading coefficient.

10.4. 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 section 3 occurs for barycentric rational interpolants described by Floater and Hormann [8]. In the simplest case, which was previously discovered by Berrut [2], for an arbitrary set of nodes one can prescribe the barycentric weights w_j to be

$$w_j = (-1)^j, \quad 0 \le j \le n\,. \tag{10.10}$$

As discussed in section 1, the second form of the barycentric formula (1.3) interpolates the values f_j, as long as the weights w_j are not zero. The choice of weights (10.10) 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 $f_j = 1$.

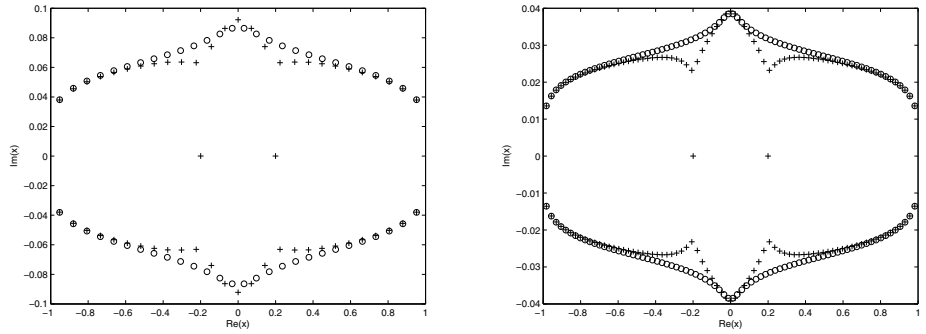
As an example, let us interpolate the function

$$f(z) = \frac{1}{(1 + 25z^2)} - \frac{1}{2} \tag{10.11}$$

at equispaced points on the interval $[-1, 1]$, using the weights given in (10.10). 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 section 4, and deflate the two spurious infinite eigenvalues. The roots and poles of the rational interpolant are shown in Figure 10.3. 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 10.6 shows the interpolation error and the error between the two real 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 section 3. Figure 10.4 illustrates the degradation in the orthogonality 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.

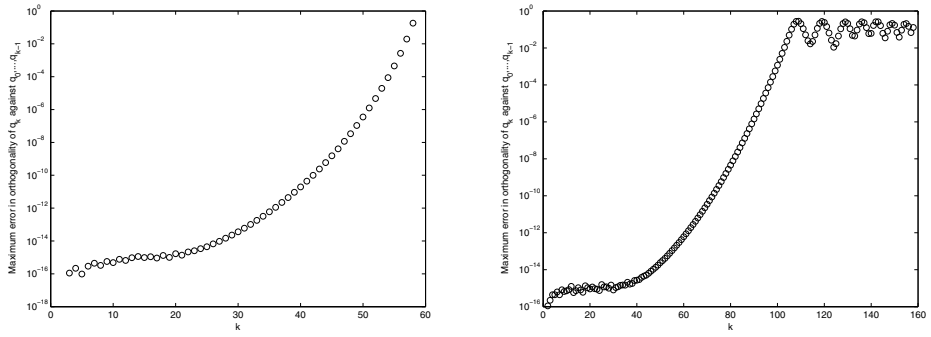


(a) $n = 56$ (b) $n = 156$

FIG. 10.3. *Roots (+) and poles (∘) of the rational interpolant.*

TABLE 10.6
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) $n = 157$

FIG. 10.4. *Error in orthogonality of vectors* $\mathbf{q}_0, \ldots, \mathbf{q}_n$*:* $\max_{0\le i\le k-1} |\mathbf{q}_i^* \mathbf{q}_k|$.

11. Concluding remarks. In this paper, 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.

Acknowledgments. The author thanks Luca Gemignani for providing an implementation of the quasiseparable matrix algorithm described in [7]. The author is also grateful to Rob Corless and David Jeffrey for their constructive and useful comments on draft versions of the present paper.

REFERENCES

[1] Z. Battles and L. N. Trefethen, *An extension of MATLAB to continuous functions and operators*, SIAM J. Sci. Comput., 25 (2004), pp. 1743–1770.
[2] J.-P. Berrut, *Rational functions for guaranteed and experimentally well-conditioned global interpolation*, Comput. Math. Appl., 15 (1988), pp. 1–16.
[3] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Rev., 46 (2004), pp. 501–517.
[4] D. A. Bini, L. Gemignani, and V. Y. Pan, *Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations*, Numer. Math., 100 (2005), pp. 373–408.
[5] R. M. Corless, *Generalized companion matrices in the Lagrange basis*, in Proceedings EACA, L. Gonzalez-Vega and T. Recio, eds., 2004, pp. 317–322.
[6] D. Day and L. Romero, *Roots of polynomials expressed in terms of orthogonal polynomials*, SIAM J. Numer. Anal., 43 (2005), pp. 1969–1987.
[7] Y. Eidelman, I. Gohberg, and L. Gemignani, *On the fast reduction of a quasiseparable matrix to Hessenberg and tridiagonal forms*, Linear Algebra Appl., 420 (2007), pp. 86–101.
[8] M. S. Floater and K. Hormann, *Barycentric rational interpolation with no poles and high rates of approximation*, Numer. Math., 107 (2007), pp. 315–331.
[9] 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.
[10] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[11] I. J. Good, *The Colleague Matrix, a Chebyshev analogue of the companion matrix*, Quart. J. Math. Oxford Ser. (2), 12 (1961), pp. 61–68.
[12] N. J. Higham, *The numerical stability of barycentric Lagrange interpolation*, IMA J. Numer. Anal., 24 (2004), pp. 547–556.
[13] P. W. Lawrence and R. M. Corless, *Stability of rootfinding for barycentric Lagrange interpolants*, Numer. Algorithms, submitted.
[14] D. Lemonnier and P. Van Dooren, *Balancing regular matrix pencils*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 253–263.
[15] D. S. Mackey, N. Mackey, and F. Tisseur, *Structured tools for structured matrices*, Electron. J. Linear Algebra, 10 (2003), pp. 106–145.
[16] E. E. Osborne, *On pre-conditioning of matrices*, J. ACM, 7 (1960), pp. 338–345.
[17] B. N. Parlett and C. H. Reinsch, *Handbook series linear algebra: Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numer. Math., 13 (1969), pp. 293–304.
[18] M. Riesz, *Eine trigonometrische Interpolationsformel und einige Ungleichungen für Polynome*, Jahresbericht der Deutschen Mathematiker-Vereinigung, 23 (1914), pp. 354–368.
[19] H. Rutishauser, *Vorlesungen über Numerische Mathematik*, Vol. 1, Birkhäuser, Basel, Stuttgart, 1976; English translation, *Lectures on Numerical Mathematics*, W. Gautschi, ed., Birkhäuser, Boston, 1990.
[20] W. Specht, *Die Lage der Nullstellen eines Polynoms.* III, Math. Nachr., 16 (1957), pp. 369–389.
[21] L. N. Trefethen, *Spectral Methods in MATLAB*, Software Environ. Tools 10, SIAM, Philadelphia, 2000.
[22] L. N. Trefethen et al., *Chebfun Version* 4.2, The Chebfun Development Team, 2011, http://www.maths.ox.ac.uk/chebfun/.
[23] M. Van Barel, R. Vandebril, P. Van Dooren, and K. Frederix, *Implicit double shift QR-algorithm for companion matrices*, Numer. Math., 116 (2010), pp. 177–212.
[24] R. Vandebril, M. Van Barel, and N. Mastronardi, *A multiple shift QR-step for structured rank matrices*, J. Comput. Appl. Math., 233 (2010), pp. 1326–1344.
[25] R. C. Ward, *Balancing the generalized eigenvalue problem*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 141–152.

[26] D. S. Watkins, *Fundamentals of Matrix Computations*, 2nd ed., John Wiley and Sons, New York, 2002.
[27] D. S. Watkins, *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*, SIAM, Philadelphia, 2007.
[28] M. Webb, L. N. Trefethen, and P. Gonnet, *Stability of barycentric interpolation formulas for extrapolation*, SIAM J. Sci. Comput., 34 (2012), pp. A3009–A3015.
[29] J. H. Wilkinson, *The perfidious polynomial*, in Studies in Numerical Analysis, MAA Stud. Math. 24, Math. Assoc. America, Washington, DC, 1984, pp. 1–28.