

# Numerical Behavior of the DQR Method for Rank-Structured Matrices

Luca Gemignani<sup>1</sup>

*Dipartimento di Matematica, Università di Pisa, Largo Bruno Pontecorvo 5,  
56127 Pisa, Italy.*

Frank Uhlig

*Department of Mathematics and Statistics, Auburn University,  
Auburn, AL 36849 - 5310, USA*

---

## Abstract

In this paper we consider fast numerical algorithms for solving certain modified matrix eigenvalue problems associated with algebraic equations. The matrices under consideration have the form  $A = T + \mathbf{u}\mathbf{v}^T$ , where  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{n \times n}$  and  $T = (t_{i,j}) \in \mathbb{R}^{n \times n}$  is a tridiagonal matrix such that  $t_{j+1,j} = \pm t_{j,j+1}$ ,  $1 \leq j \leq n - 1$ . We show that the DQR approach proposed in [Uhlig F., Numer. Math. 76 (1997), no. 4, 515–553] can compute the eigenvalues of such matrices efficiently. Our algorithm employs a fast Hessenberg reduction scheme for tridiagonal plus rank-one matrices combined with an efficient eigenvalue method for the resulting Hessenberg matrices. Exploiting the rank structure of the input matrix enables both steps to be carried out using quadratic time and linear storage. Numerical implementation of these techniques is presented and discussed for a number of test problems.

*AMS classification:* 65F15

*Key words:* Tridiagonal matrices, algebraic equation, rank structure, eigenvalue problem, complexity.

---

---

*Email addresses:* gemignan@dm.unipi.it (Luca Gemignani),  
uhligfd@auburn.edu (Frank Uhlig).

<sup>1</sup> Supported by MIUR under project 2004015437.

## 1 Introduction

To compute eigenvalues of general real tridiagonal matrices is a challenging problem. If a general tridiagonal matrix is irreducible, then by diagonal similarity it can be transformed into a balanced tridiagonal matrix  $T = (t_{i,j})$  with  $t_{j+1,j} = \pm t_{j,j+1}$ ,  $1 \leq j \leq n - 1$ . This property generalizes to the *D-property* or, equivalently, the *sign-symmetry property* of matrices, namely a matrix  $A \in \mathbb{C}^{n \times n}$  has the D-property or is sign-symmetric if there exists a diagonal matrix  $D = \text{diag}[1, \pm 1]$  such that  $D \cdot A$  is a symmetric matrix. Eigenvalue of sign symmetric tridiagonal matrices that are not symmetric can be computed by the unsymmetric Lanczos process (Lanczos, 1951). The real tridiagonal eigenvalue problem is also related to polynomial root-finding. Given a monic polynomial  $p(x) \in \mathbb{C}[x]$  of degree  $n$  a (block) tridiagonal matrix  $T$  can be constructed having  $p(x)$  as its characteristic polynomial. An efficient root-finding algorithm based on the computation of the eigenvalues of the associated matrix  $T$  was proposed in (Uhlig, 1999). The algorithm exploits the D-property of  $T$  to carry out the eigenvalue computation using D-orthogonal transformations in quadratic time and linear storage. Eigenvalue problems for matrices of the form  $A = T + \mathbf{u}\mathbf{v}^T$ ,  $\mathbf{u}, \mathbf{v} \in \mathbb{C}^{n \times n}$  are encountered in zero-finding methods for polynomials expressed in terms of bases satisfying a three-term recurrence relation, as well as in approximation problems solved by means of the unsymmetric Lanczos iteration.

Similar eigenproblems also arise in the field of Computer Aided Geometric Design, where alternative polynomial bases other than the power basis are often used to approximate curves and surfaces. The use of the Lagrange basis is suggested quite often for addressing problems in the so-called framework of “Polynomial Algebra By Values” (Amiraslani et al., 2004), mostly because Lagrange polynomials can directly be specified in terms of the values attained at the sample points. It is also invoked in the iterative refinement techniques for polynomial roots at the core of Fortune’s algorithm (Fortune, 2001, 2002). Recent papers (Berrut and Trefethen, 2004), (Higham, 2004) have shown that working directly in the Lagrange basis is both numerically stable and efficient. On the contrary, conversion between different bases can be dramatically ill-conditioned already for input polynomials of small degree (Hermann, 1996). The polynomial root-finding problem for a polynomial  $p(x)$  expressed in the Lagrange basis can be reduced to compute the eigenvalues of a diagonal plus rank one matrix. If  $p(x) \in \mathbb{R}[x]$  and the interpolation nodes are partitioned into two disjoint subsets  $\mathcal{X}_{\mathbb{C}} \subset \mathbb{C} \setminus \mathbb{R}$  and  $\mathcal{X}_{\mathbb{R}} \subset \mathbb{R}$ , then a real form of the interpolating polynomial can be obtained in such a way that the polynomial root-finding problem reduces to solving the eigenproblem for a matrix  $A \in \mathbb{R}^{n \times n}$  which is a rank-one correction of a real block diagonal matrix with diagonal blocks that are  $1 \times 1$  or  $2 \times 2$  matrices  $D$  such that  $|D_{1,2}| = |D_{2,1}|$ .

In this paper we apply the DQR method, devised in (Uhlig, 1997) for computing the eigenvalues of sign-symmetric tridiagonal matrices, to the more general class of matrices of the form  $A = T + \mathbf{u}\mathbf{v}^T$ , where  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$  are arbitrary and  $T \in \mathbb{R}^{n \times n}$  is a sign-symmetric tridiagonal matrix. In order to preserve the quadratic complexity of the algorithm in (Uhlig, 1997) we rely upon some recently proposed adaptations of customary QR-like eigenvalue methods for small rank perturbations of Hermitian tridiagonal matrices (Eidelman et al., 2007, 2008). Specifically, we develop a two-step algorithm which first reduces the matrix  $A$  by D-similarity to an upper Hessenberg form  $B$  and then uses a suitable modification of the tridiagonal DQR eigenvalue algorithm in (Uhlig, 1997) for computing the eigenvalues of  $B$ . It is shown that each step can be performed in quadratic time using linear memory space. The speedup with respect to customary methods based on orthogonal transformations is achieved by exploiting the rank structures of the matrices generated by the reduction process. The price for this advance is to replace stable algorithms with potentially unstable computations. The stable QR algorithm requires  $O(n^3)$  flops and  $O(n^2)$  memory storage. The DQR-based approach of this paper is a compromise between stability and computational efficiency. We present empirical results showing the accuracy and the performance of reduction to Hessenberg form where the size of elementary D-transformations is properly monitored. Concerning the iterative phase a double-shift DQR iteration is incorporated in the implicit DQR algorithm to improve its accuracy. In view of the conditionally stability result for the DQR iteration proved in (Uhlig, 1997), the combination of single-shift and the double-shift iterations guarantees the accuracy of the results in almost all cases. Our conclusion is that, even though no stable fast eigensolver is known for the considered class of structured matrices, the novel algorithm significantly broadens the subset of eigenproblems that can be solved successfully and efficiently.

The paper is organized as follows. In Section 2 we review preliminaries and basic properties of the matrices involved. The D-similarity reduction of these matrices to Hessenberg form and the DQR-like eigenvalue solver for the resulting Hessenberg matrices are described in Section 3. In Section 4 we report and discuss the results of numerical experiments. Finally, conclusion and possible further developments are drawn in Section 5.

## 2 Preliminaries

Customary eigenvalue methods for dense matrices make use of unitary transformations to reduce the input matrix in some convenient form. A generalization of the concept of unitary matrix in vector spaces equipped with an indefinite scalar product is provided by the following (Gohberg et al., 1983).

**Definition 2.1.** For a given diagonal *signature* matrix  $D = \text{diag}[\pm 1] \in \mathbb{R}^{n \times n}$  a matrix  $Q \in \mathbb{C}^{n \times n}$  is called D-orthogonal if  $Q^T \cdot D \cdot Q = D$ .

The reduction of a vector  $\mathbf{x} \in \mathbb{C}^n$  to a scalar multiple of the first column of  $I_n$  by using a D-orthogonal matrix  $Q$  can be carried out by annihilating one entry of  $\mathbf{x}$  at a time in a process akin to the use of orthogonal Givens rotations. Thus we can restrict ourselves to the case  $n = 2$ .

If  $D = \pm I_2$  and  $\mathbf{x}^T \mathbf{x} \neq \mathbf{0}$ ,  $\mathbf{x} = [x_1, x_2]^T$ , then the Givens-like transformation

$$\mathcal{G} = \text{Givens}(x_1, x_2) = \begin{bmatrix} \frac{x_1}{\sqrt{x_1^2 + x_2^2}} & \frac{x_2}{\sqrt{x_1^2 + x_2^2}} \\ -\frac{x_2}{\sqrt{x_1^2 + x_2^2}} & \frac{x_1}{\sqrt{x_1^2 + x_2^2}} \end{bmatrix}$$

is such that

$$\mathcal{G} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ 0 \end{bmatrix}, \quad \mathcal{G}^T \cdot I_2 \cdot \mathcal{G} = I_2.$$

When  $D = \pm \text{diag}[1, -1]$  and  $\mathbf{x}^T \cdot D \cdot \mathbf{x} \neq \mathbf{0}$ , then the hyperbolic Givens matrix  $\mathcal{G}$

$$\mathcal{G} = \text{Hyp}(x_1, x_2) = \begin{bmatrix} \frac{x_1}{\sqrt{x_1^2 - x_2^2}} & -\frac{x_2}{\sqrt{x_1^2 - x_2^2}} \\ -\frac{x_2}{\sqrt{x_1^2 - x_2^2}} & \frac{x_1}{\sqrt{x_1^2 - x_2^2}} \end{bmatrix}$$

is D-orthogonal and maps  $\mathbf{x}$  as desired

$$\mathcal{G} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ 0 \end{bmatrix}, \quad \mathcal{G}^T \cdot D \cdot \mathcal{G} = D. \quad (2.1)$$

The next procedure, called **reduce** is used to annihilate the entries of an input vector  $\mathbf{x} \in \mathbb{C}^n$  in positions  $2, 3, \dots, n$  via a mixture of orthogonal Givens and hyperbolic Givens rotations, depending on the sign pattern of  $D$ .

### Procedure reduce

**Input:**  $\mathbf{x} = [x_1, \dots, x_n]^T$ ,  $D = \text{diag}[\pm 1]$ ;

**for**  $k = n : -1 : 2$

**if**  $D(k, k) * D(k-1, k-1) = 1$

**if**  $(x_{k-1}^2 + x_k^2 \neq 0)$

$\mathcal{G} = \text{Givens}(x_{k-1}, x_k)$ ;  $[y, 0]^T = \mathcal{G}\mathbf{x}(k-1 : k)$ ;  $x_{k-1} \leftarrow y$ ;

**else**

**disp**(error); **return**;

```

    end;
else
    if  $(x_{k-1}^2 - x_k^2 \neq 0)$ 
         $\mathcal{G} = \text{Hyp}(x_{k-1}, x_k); [y, 0]^T = \mathcal{G}\mathbf{x}(k-1:k); x_{k-1} \leftarrow y;$ 
    else
        disp(error); return;
    end;
end;
end
end

```

Whenever the procedure terminates without an error message it returns an upper Hessenberg matrix  $Q \in \mathbb{C}^{n \times n}$  satisfying

$$Q\mathbf{x} = [y, 0, \dots, 0]^T, \quad Q^T \cdot D \cdot Q = D,$$

where  $D$  is the input signature matrix. The key observation is that the *D-property* is invariant under the similarity transformation  $B \leftarrow Q \cdot A \cdot Q^{-1}$  applied to the matrix  $A$  such that  $DA = A^T D$ . Specifically, we find that

$$DB = DQAQ^{-1} = Q^{-T}DAQ^{-1} = Q^{-T}A^TDQ^{-1} = Q^{-T}A^TQ^TD = B^TD.$$

Thus the transformed or updated matrix  $B$  fulfills the *D-property* for the same  $D$  as  $A$ . This allows us to generalize fast adaptations of customary eigenvalue methods for dense matrices to the class of matrices satisfying the *D-property* for a signature matrix  $D$ . In the next section we describe generalizations of the fast Hessenberg reduction algorithm in (Eidelman et al., 2007) and the fast implicit QR algorithm in (Eidelman et al., 2008). The modified routines can be combined into a composite algorithm for computing the eigenvalues of matrices with the *D-property* quickly.

### 3 Fast Eigenvalue Computation via Generalized Orthogonal Transformations

In this section we develop fast algorithms that use generalized orthogonal similarity transformations for approximating the eigenvalues of an input matrix  $A \in \mathbb{R}^{n \times n}$  of the form

$$A = T + \mathbf{u}\mathbf{v}^T, \quad \mathbf{u}, \mathbf{v} \in \mathbb{R}^n,$$

where  $T \in \mathbb{R}^{n \times n}$  is a tridiagonal matrix with the D-property for  $D = \text{diag}[d_1, \dots, d_n]$ . The matrix  $A$  is then reduced iteratively to simpler block triangular form that reveals the approximate eigenvalues.

Set  $A^{(0)} = A$  and observe that  $DT = T^T D$  implies

$$\begin{aligned}
DA^{(0)} - A^{(0)T}D &= DT + D\mathbf{u}\mathbf{v}^T - T^T D - \mathbf{v}\mathbf{u}^T D = \\
&= D\mathbf{u}\mathbf{v}^T - \mathbf{v}\mathbf{u}^T D = D\mathbf{u}^{(0)}\mathbf{v}^{(0)T} - \mathbf{v}^{(0)}\mathbf{u}^{(0)T} D.
\end{aligned}$$

Hence the entries of  $A = A^{(0)}$  that are located in the strictly upper triangular part can immediately be reconstructed from the entries in the strictly lower triangular part, the vectors  $\mathbf{u} = \mathbf{u}^{(0)}$  and  $\mathbf{v} = \mathbf{v}^{(0)}$ , and the diagonal entries of  $D$ . Furthermore, notice that the strictly lower triangular part of  $A^{(0)}$  is

$$\text{tril}(A^{(0)}, -2) = \text{tril}(\mathbf{u}\mathbf{v}^T, -2) = \text{tril}(\mathbf{u}^{(0)}\mathbf{v}^{(0)T}, -2), \quad (3.2)$$

where we adopt Matlab notation to denote the lower triangular portion of  $B$  as  $S = \text{tril}(B, k)$  with  $s_{i,j} = b_{i,j}$  for  $j - i \leq k$  and  $s_{i,j} = 0$  elsewhere.

For computational purposes it is useful to express the (3.2) by using a more suited representation for the strictly lower triangular part of the rank-one matrix  $\mathbf{u}\mathbf{v}^T$ . Thus, for suitable  $p_i, q_i, t_i \in \mathbb{R}$  we write

$$\text{tril}(A, -2) = L(\{p_i\}, \{q_i\}, \{t_i\}) = \begin{bmatrix} 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & & & & \vdots \\ p_3 q_1 & \ddots & \ddots & & & \vdots \\ p_4 t_2 q_1 & p_4 q_2 & \ddots & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 \\ p_n t_{n,1}^\times q_1 & p_n t_{n,2}^\times q_2 & \dots & p_n q_{n-2} & 0 & 0 \end{bmatrix},$$

where  $t_{i,j}^\times = t_{i-2} \cdots t_{j+1}$  if  $i - 2 \geq j + 1$  and  $t_{i,j}^\times = 1$  elsewhere.

Summing up, the entries of  $A = A^{(0)} = (a_{i,j}^{(0)})$  can be specified as follows:

$$a_{i,j}^{(0)} = \begin{cases} p_i^{(0)} t_{i,j}^{(0)\times} q_j^{(0)} & \text{for } 1 \leq j \leq n-2, 3 \leq i \leq n; \\ \beta_j^{(0)} & \text{for } 1 \leq j = i-1 \leq n-1; \\ \alpha_j^{(0)} & \text{for } 1 \leq j = i \leq n-1; \\ \frac{d_j \beta_i^{(0)} - d_j u_j^{(0)} v_i^{(0)} + d_i u_i^{(0)} v_j^{(0)}}{d_i} & \text{for } 1 \leq i = j-1 \leq n-1; \\ \frac{d_j p_j^{(0)} t_{j,i}^{(0)\times} q_i^{(0)} - d_j u_j^{(0)} v_i^{(0)} + d_i u_i^{(0)} v_j^{(0)}}{d_i} & \text{for } 1 \leq i \leq n-2, 3 \leq j \leq n. \end{cases} \quad (3.3)$$

The elements  $p_i^{(0)}, q_i^{(0)}, t_i^{(0)}, \beta_i^{(0)}, \alpha_i^{(0)}, u_i^{(0)}, v_i^{(0)} \in \mathbb{R}$  are referred to as the *generating elements* of  $A^{(0)}$ . Fast algorithms for approximating the eigenvalues of  $A^{(0)}$  perform the reduction of  $A^{(0)}$  to simpler form through manipulation of the generating elements rather than the matrix entries. This decreases the size

of the input data from  $O(n^2)$  to  $O(n)$ . The next two subsections describe fast adaptations of customary methods for the Hessenberg reduction and for the implicit QR method applied to Hessenberg matrices.

### 3.1 Hessenberg Reduction

A fast algorithm for the Hessenberg reduction of real matrices of the form (3.3) with  $d_i = 1$ ,  $1 \leq i \leq n$ , was proposed in (Eidelman et al., 2007). It is based on the earlier work in (Eidelman and Gohberg, 2002). The algorithm uses orthogonal matrices for converting  $A = A^{(0)}$  to an upper Hessenberg matrix  $H^{(0)}$  iteratively. Replacing the orthogonal transformations with their generalized orthogonal analogues extends this algorithm to work with an arbitrary signature matrix  $D$ .

The hyperbolic reduction of  $A = A^{(0)}$  to  $H^{(0)} = A^{(n-2)}$  is performed in  $n - 1$  steps  $A^{(j-1)} \rightarrow A^{(j)}$ ,  $1 \leq j \leq n - 2$ , where  $A^{(j)} = Q_j \cdot A^{(j-1)} \cdot Q_j^{-1}$  and  $Q_j^T \cdot D \cdot Q_j = D$ . Each step is realized by a sequence of generalized D-orthogonal transformations. For the sake of illustration, consider the first step  $A = A^{(0)} \rightarrow A^{(1)}$ . The matrix  $A^{(1)}$  is determined so that  $A^{(1)}(3:n, 1) = \mathbf{0}$ . The zeroing process proceeds as follows:

$$A = A^{(0)} = A_0 \rightarrow A_1 \rightarrow \dots \rightarrow A_{n-3} \rightarrow A_{n-2} = A^{(1)},$$

where

$$A_j = (a_{i,k}^{(j)}) = G_{n-j} \cdot A_{j-1} \cdot G_{n-j}^{-1}, \quad G_{n-j}^T \cdot D \cdot G_{n-j} = D, \quad 1 \leq j \leq n - 2,$$

and  $G_{n-j}$  has the form

$$G_{n-j} = I_{n-j-1} \oplus \mathcal{G}_{n-j} \oplus I_{j-1}, \quad 1 \leq j \leq n - 2.$$

Here the  $2 \times 2$  matrices  $\mathcal{G}_{n-j}$ ,  $1 \leq j \leq n - 2$ , are defined as in **reduce** of the previous section. They zero the entries of the input vector  $[x_2, \dots, x_n]^T = A^{(0)}(2:n, 1)$ . For later use we write

$$\mathcal{G}_k^{-1} = \begin{bmatrix} a_k & \psi_k b_k \\ b_k & a_k \end{bmatrix} \quad \text{with } \psi_k \in \{\pm 1\}, \quad 2 \leq k \leq n - 1.$$

The following result describes the structure of the matrix  $A_{n-2} = A^{(1)}$ .

**Theorem 3.1.** For the matrix  $A^{(1)} = Q_1 \cdot A^{(0)} \cdot Q_1^{-1}$  with  $Q_1^T \cdot D \cdot Q_1 = D$ , we have

$$DA^{(1)} - A^{(1)T}D = D\mathbf{u}^{(1)}\mathbf{v}^{(1)T} - \mathbf{v}^{(1)}\mathbf{u}^{(1)T}D, \quad \mathbf{u}^{(1)} = Q_1\mathbf{u}^{(0)}, \quad \mathbf{v}^{(1)} = Q_1^{-T}\mathbf{v}^{(0)}$$

and

$$A^{(1)} = \left[ \begin{array}{c|ccc} \alpha_1 & a_{1,2}^{(1)} & \dots & a_{1,n}^{(1)} \\ \hline \beta_1 & & & \\ 0 & & & \\ \vdots & & & \\ 0 & & & \end{array} \right] A_1^{(1)},$$

where the strictly lower triangular part of  $A_1^{(1)} \in \mathbb{C}^{(n-1) \times (n-1)}$  is described by

$$\text{tril}(A_1^{(1)}, -2) = L(\{a_{i+2,i}^{(n-i)}\}_{i=2}^{n-2}, \{a_i\}_{i=1}^{n-3}, \{b_i\}_{i=1}^{n-3}), \quad (a_1 = 1).$$

**PROOF.** The first relation follows from

$$\begin{aligned} DA^{(1)} - A^{(1)T}D &= DQ_1A^{(0)}Q_1^{-1} - Q_1^{-T}A^{(0)T}Q_1^TD = \\ &= Q_1^{-T}DA^{(0)}Q_1^{-1} - Q_1^{-T}A^{(0)T}DQ_1^{-1} = Q_1^{-T}(D\mathbf{u}^{(0)}\mathbf{v}^{(0)T} - \mathbf{v}^{(0)}\mathbf{u}^{(0)T}D)Q_1^{-1} = \\ &= D\mathbf{u}^{(1)}\mathbf{v}^{(1)T} - \mathbf{v}^{(1)}\mathbf{u}^{(1)T}D, \quad \mathbf{u}^{(1)} = Q_1\mathbf{u}^{(0)}, \quad \mathbf{v}^{(1)} = Q_1^{-T}\mathbf{v}^{(0)}. \end{aligned}$$

The structure of  $A_1^{(1)}$  is obtained by observing that

$$A^{(1)} = G_3 \cdots G_{n-1} \cdot A^{(0)} \cdot G_{n-1}^{-1} \cdots G_3^{-1} = E^{(0)} \cdot G_{n-1}^{-1} \cdots G_2^{-1},$$

where  $E^{(0)}$  is lower banded with bandwidth 2.

This theorem allows us to replace the computation of the generating elements of  $A_1^{(1)}$  with determining the sequence  $\mathcal{G}_{n-j}$  for  $1 \leq j \leq n-2$  together with the entries in the first three subdiagonals of  $A_j$ ,  $1 \leq j \leq n-2$ . The sequence of generalized orthogonal transformations can be generated at the cost of  $O(n)$  operations. Moreover, the calculation of  $\{a_{i+2,i}^{(n-i)}\}$ ,  $\{a_{i+1,i}^{(n-i)}\}$  and  $\{a_{i,i}^{(n-i)}\}$  can be performed in linear time by simply modifying the pentadiagonal part of the

matrices  $A_j$ ,  $1 \leq j \leq n - 2$ . More specifically, we observe that

$$A_{n-k}(1: k+2, 1: k+2) = \begin{bmatrix} * & * & * \\ R_{n-k} & P_{n-k} & * \\ \circ & Z_{n-k} & * \end{bmatrix} = \begin{bmatrix} * & * & * \\ p_{k-1}^{(0)} t_{k-1,1}^{(0)} \times q_1^{(0)} \cdots p_{k-1}^{(0)} q_{k-3}^{(0)} \beta_{k-2}^{(0)} & a_{k-1,k-1}^{(n-k)} & * & * \\ \hat{p}_k^{(0)} t_{k,1}^{(0)} \times q_1^{(0)} \cdots \cdots \hat{p}_k^{(0)} q_{k-2}^{(0)} & a_{k,k-1}^{(n-k)} & a_{k,k}^{(n-k)} & * \\ \circ & a_{k+1,k-1}^{(n-k)} & a_{k+1,k}^{(n-k)} & * \\ & 0 & a_{k+2,k}^{(n-k)} & * \end{bmatrix}.$$

Therefore, the transformation  $A_{n-k} \rightarrow A_{n-k+1}$  essentially amounts to computing the matrices

$$\mathcal{G}_{k-1} \cdot P_{n-k} \cdot \mathcal{G}_{k-1}^{-1}, \quad Z_{n-k} \cdot \mathcal{G}_{k-1}^{-1},$$

and

$$\mathcal{G}_{k-1} \cdot R_{n-k} = \begin{bmatrix} \hat{p}_{k-1}^{(0)} t_{k-1,1}^{(0)} \times q_1^{(0)} \cdots \hat{p}_{k-1}^{(0)} q_{k-3}^{(0)} & a_{k-1,k-2}^{(n-k+1)} \\ 0 & \cdots & 0 & a_{k,k-2}^{(n-k+1)} \end{bmatrix}.$$

Due to the structure of  $R_{n-k}$ , the last matrix multiplication can be performed in constant time. Moreover, all remaining entries of  $A_{n-k+1}$  can be determined by use of the D-property for the matrix as shown in (3.3). Thus, the cumulative transformation  $A^{(0)} \rightarrow A^{(1)}$  can be carried out in  $O(n)$  operations and  $O(n)$  storage.

Under the assumption that all the generalized orthogonal transformations involved are well-defined, at the very end of this process the matrix  $A = A^{(0)}$  has been converted to upper Hessenberg form  $H^{(0)}$  by similarity using generalized orthogonal transformations at an overall cost of  $O(n^2)$  operations with  $O(n)$  storage. The matrix  $H^{(0)}$  satisfies

$$DH^{(0)} - H^{(0)T}D = D\mathbf{u}^{(n-2)}\mathbf{v}^{(n-2)T} - \mathbf{v}^{(n-2)}\mathbf{u}^{(n-2)T}D, \quad (3.4)$$

for the given signature matrix  $D$ . The next subsection describes fast modifications of the implicit QR method for approximating the eigenvalues of  $H = H^{(0)}$ .

### 3.2 Matrix Eigenvalue Computations via Structured DQR Iteration

The shifted QR eigenvalue algorithm of (Francis, 1961, 1962), applied to the input matrix  $H = H^{(0)}$  generates a sequence of unitarily similar matrices  $\{H^{(k)}\}_{k \geq 0}$  that converges to the (block) Schur normal form of  $H$ . The QR iteration with shift proceeds as follows:

$$\begin{aligned} H^{(0)} &= H \\ p_k(H^{(k)}) &= Q^{(k)} R^{(k)}, \text{ (QR factorization)} \\ H^{(k+1)} &:= Q^{(k)H} H^{(k)} Q^{(k)}, \end{aligned} \tag{3.5}$$

where  $p_k(z)$  is a monic polynomial of degree one (*single-shift step*) or two (*double-shift step*) suitably chosen to accelerate convergence. The explicit computation of  $p_k(H^{(k)})$  is expensive when  $\deg(p_k) = 2$ . To circumvent this and to stay within real arithmetic for real matrices,, the *implicit QR technique*, invented in (Francis, 1961, 1962) manages to perform the transformation  $H^{(k)} \rightarrow H^{(k+1)}$  without explicitly forming the matrix  $p_k(H^{(k)})$  when  $\deg(p_k) = 2$ .

Unfortunately, the complexity of the QR eigenvalue algorithm (3.5) applied to  $H = H^{(0)}$  satisfying (3.4) is generally  $O(n^3)$  flops and  $O(n^2)$  storage. Fast adaptations of the implicit single-shift and double-shift QR algorithms for real matrices satisfying (3.4) with  $D = I_n$  were proposed in (Eidelman et al., 2008). These fast methods do not apply in the complex case nor for general signature matrices. The desired extension is possible, however, if we replace the orthogonal or unitary matrices of the QR algorithm by D-orthogonal matrices. The multishift DQR iteration then has the form

$$\begin{aligned} H^{(0)} &= H \\ p_k(H^{(k)}) &= Q^{(k)} R^{(k)}, \text{ (DQR factorization)} \\ H^{(k+1)} &:= (Q^{(k)})^{-1} H^{(k)} Q^{(k)}, \end{aligned} \tag{3.6}$$

where the  $Q^{(k)}$  now are D-orthogonal matrices. The single-shift DQR iteration was first considered in (Uhlig, 1997) for applications to sign-symmetric tridiagonal matrices. Extensions to the more general class of matrices satisfying (3.4) can be obtained by modifying the routines in (Eidelman et al., 2008).

For notational simplicity we omit the superscript and subscript  $k$  whenever it is possible. If  $p(z) = z - \alpha$  is a linear polynomial, then the DQR iteration (3.6) reduces to the classical shifted DQR iteration with shift  $\alpha \in \mathbb{C}$ . The input matrix  $H$  fulfilling (3.4) is specified by the subdiagonal entries  $\beta_k$ ,  $1 \leq k \leq n - 1$ , the diagonal entries  $\vartheta_k$ ,  $1 \leq k \leq n$ , and the elements of the vectors

$\mathbf{u}$  and  $\mathbf{v}$  for which (3.4) holds. A structured implicit variant, applied to the input matrix  $H$  that is specified by its generating elements is described below. The outputs are the generating elements of the new iterate  $H^{(1)}$ .

**Input:**  $\vartheta_k, 1 \leq k \leq n; \beta_k, 1 \leq k \leq n, (\beta_n := 0); \mathbf{u}_k, \mathbf{v}_k, 1 \leq k \leq n; \alpha$ .

**Output:**  $\vartheta_k^{(1)}, 1 \leq k \leq n; \beta_k^{(1)}, 1 \leq k \leq n-1; \mathbf{u}_k^{(1)}, \mathbf{v}_k^{(1)}, 1 \leq k \leq n$ .

$\tilde{\vartheta}_1 := \vartheta_1; \tilde{\mathbf{u}}_1 := \mathbf{u}_1; \tilde{\mathbf{v}}_1 := \mathbf{v}_1; \delta_0 := \vartheta_1 - \alpha; \varepsilon_0 = \beta_1; \tilde{\beta}_1 = \beta_1;$

**for**  $k = 1, \dots, n-1$

$\gamma_k = (d_{k+1}\beta_k - d_{k+1}\tilde{\mathbf{v}}_k\mathbf{u}_{k+1} + d_k\tilde{\mathbf{u}}_k\mathbf{v}_{k+1})/d_k;$

$[\mathcal{G}_k, \beta_{k-1}^{(1)}] = \mathbf{reduce}([\delta_{k-1}; \varepsilon_{k-1}]);$

$\begin{bmatrix} \mathbf{u}_k^{(1)} \\ \tilde{\mathbf{u}}_{k+1} \end{bmatrix} = \mathcal{G}_k \begin{bmatrix} \tilde{\mathbf{u}}_k \\ \mathbf{u}_{k+1} \end{bmatrix}; \begin{bmatrix} \mathbf{v}_k^{(1)} & \tilde{\mathbf{v}}_{k+1} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{v}}_k & \mathbf{v}_{k+1} \end{bmatrix} \mathcal{G}_k^{-1};$

$\begin{bmatrix} \vartheta_k^{(1)} & * \\ \delta_k & \tilde{\vartheta}_{k+1} \\ \varepsilon_k & \tilde{\beta}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathcal{G}_k & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{\vartheta}_k & \gamma_k \\ \tilde{\beta}_k & \vartheta_{k+1} \\ 0 & \beta_{k+1} \end{bmatrix} \mathcal{G}_k^{-1};$

**end**

$\vartheta_n^{(1)} = \tilde{\vartheta}_n; \mathbf{u}_n^{(1)} = \tilde{\mathbf{u}}_n; \mathbf{v}_n^{(1)} = \tilde{\mathbf{v}}_n; \beta_{n-1}^{(1)} = \delta_{n-1}.$

The code employs the function **reduce** of the previous section. Property (3.4) that is fulfilled by the input matrix  $H$  is inherited by the intermediate matrices generated in the inner loop. These can therefore be reconstructed easily from their generating elements. In particular, this is true for the superdiagonal entries  $\gamma_k$ . The overall computational cost of one structured single shift DQR iteration is  $O(n)$  flops.

When  $p(z) = (z - \alpha_1)(z - \alpha_2)$  for  $\alpha_i \in \mathbb{C}$  is monic quadratic, then the process (3.6) is an (implicit) double shift DQR iteration. A structured adaptation for the input  $H$  that is expressed in terms of its generating elements is described below. Similar to the single shift iteration, all intermediate matrices in the inner loop of the code below can be specified by  $O(n)$  parameters. In particular, the superdiagonal entries  $\gamma_k$ ,  $\psi_k$ , and  $\hat{\gamma}_k$  are obtained from their generating elements. The overall computational cost of the structured implicit double shift DQR iteration is approximately twice the cost of the single shift version. In the next section we report and discuss the result of numerical experiments with these procedures.

**Input:**  $\vartheta_k, 1 \leq k \leq n; \beta_k, 1 \leq k \leq n, (\beta_N := 0); \mathbf{u}_k, \mathbf{v}_k, 1 \leq k \leq n; \alpha_1, \alpha_2$ .

**Output:**  $\vartheta_k^{(1)}, 1 \leq k \leq n; \beta_k^{(1)}, 1 \leq k \leq n-1; \mathbf{u}_k^{(1)}, \mathbf{v}_k^{(1)}, 1 \leq k \leq n$ .

$\tilde{\vartheta}_1 := \vartheta_1; \hat{\vartheta}_2 := \vartheta_2; \tilde{\mathbf{u}}_1 := \mathbf{u}_1; \tilde{\mathbf{v}}_1 := \mathbf{v}_1; \hat{\mathbf{u}}_2 := \mathbf{u}_2; \hat{\mathbf{v}}_2 := \mathbf{v}_2;$

$\tilde{\beta}_1 := \beta_1; \hat{\beta}_2 := \beta_2; \theta_1 := 0;$

$$\begin{aligned}
\delta_0 &= \vartheta_1^2 + \gamma_1 \tilde{\beta}_1 - (\alpha_1 + \alpha_2) \vartheta_1 + \alpha_1 \alpha_2; \\
\varepsilon_0 &= \tilde{\beta}_1 (\tilde{\vartheta}_1 + \hat{\vartheta}_2 - (\alpha_1 + \alpha_2)); \\
\rho_0 &= \tilde{\beta}_1 \hat{\beta}_2; \\
\text{for } k &= 1, \dots, n-2 \\
\gamma_k &= (d_{k+1} \tilde{\beta}_k - d_{k+1} \tilde{\mathbf{v}}_k \hat{\mathbf{u}}_{k+1} + d_k \tilde{\mathbf{u}}_k \hat{\mathbf{v}}_{k+1}) / d_k; \\
\hat{\gamma}_{k+1} &= (d_{k+2} \hat{\beta}_{k+1} - d_{k+2} \hat{\mathbf{v}}_{k+1} \mathbf{u}_{k+2} + d_{k+1} \hat{\mathbf{u}}_{k+1} \mathbf{v}_{k+2}) / d_{k+1}; \\
\psi_k &= (d_{k+2} \theta_k - d_{k+2} \tilde{\mathbf{v}}_k \mathbf{u}_{k+2} + d_k \tilde{\mathbf{u}}_k \mathbf{v}_{k+2}) / d_k; \\
[\mathcal{G}_k, \beta_{k-1}^{(1)}] &= \text{reduce}([\delta_{k-1}; \varepsilon_{k-1}; \rho_{k-1}]); \\
\begin{bmatrix} \mathbf{u}_k^{(1)} \\ \tilde{\mathbf{u}}_{k+1} \\ \hat{\mathbf{u}}_{k+2} \end{bmatrix} &= \mathcal{G}_k \begin{bmatrix} \tilde{\mathbf{u}}_k \\ \hat{\mathbf{u}}_{k+1} \\ \mathbf{u}_{k+2} \end{bmatrix}; \quad \begin{bmatrix} \mathbf{v}_k^{(1)} & \tilde{\mathbf{v}}_{k+1} & \hat{\mathbf{v}}_{k+2} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{v}}_k & \hat{\mathbf{v}}_{k+1} & \mathbf{v}_{k+2} \end{bmatrix} \mathcal{G}_k^{-1}; \\
\begin{bmatrix} \vartheta_k^{(1)} & * & * \\ \delta_k & \tilde{\vartheta}_{k+1} & * \\ \varepsilon_k & \tilde{\beta}_{k+1} & \hat{\vartheta}_{k+2} \\ \rho_k & \theta_{k+1} & \hat{\beta}_{k+2} \end{bmatrix} &= \begin{bmatrix} \mathcal{G}_k & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{\vartheta}_k & \gamma_k & \psi_k \\ \tilde{\beta}_k & \hat{\vartheta}_{k+1} & \hat{\gamma}_{k+1} \\ \theta_k & \hat{\beta}_{k+1} & \vartheta_{k+2} \\ 0 & 0 & \beta_{k+2} \end{bmatrix} \mathcal{G}_k^{-1}; \\
\text{end} \\
\gamma_{n-1} &= (d_n \tilde{\beta}_{n-1} - d_n \tilde{\mathbf{v}}_{n-1} \hat{\mathbf{u}}_n + d_{n-1} \tilde{\mathbf{u}}_{n-1} \hat{\mathbf{v}}_n) / d_{n-1}; \\
[\mathcal{G}_{n-1}, \beta_{n-2}^{(1)}] &= \text{reduce}([\delta_{n-2}; \varepsilon_{n-2}]); \\
\begin{bmatrix} \mathbf{u}_{n-1}^{(1)} \\ \tilde{\mathbf{u}}_n \end{bmatrix} &= \mathcal{G}_{n-1} \begin{bmatrix} \tilde{\mathbf{u}}_{n-1} \\ \hat{\mathbf{u}}_n \end{bmatrix}; \quad \begin{bmatrix} \mathbf{v}_{n-1}^{(1)} & \tilde{\mathbf{v}}_n \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{v}}_{n-1} & \hat{\mathbf{v}}_n \end{bmatrix} \mathcal{G}_{n-1}^{-1}; \\
\begin{bmatrix} \vartheta_{n-1}^{(1)} & * \\ \delta_{n-1} & \tilde{\vartheta}_n \\ \varepsilon_{n-1} & \tilde{\beta}_n \end{bmatrix} &= \begin{bmatrix} \mathcal{G}_{n-1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{\vartheta}_{n-1} & \gamma_{n-1} \\ \tilde{\beta}_{n-1} & \hat{\vartheta}_n \\ \theta_{n-1} & \hat{\beta}_n \end{bmatrix} \mathcal{G}_{n-1}^{-1}; \\
\vartheta_n^{(1)} = \tilde{\vartheta}_n; \mathbf{u}_n^{(1)} = \tilde{\mathbf{u}}_n; \mathbf{v}_n^{(1)} = \tilde{\mathbf{v}}_n; \beta_{n-1}^{(1)} = \delta_{n-1}.
\end{aligned}$$

Multi-shift versions of our codes for  $\deg(p_k) > 2$  in (3.6) can be derived by extending the multi-shift orthogonal QR algorithm of (Braman et al. I, 2002; Braman et al. II, 2002) to work with DQR factorizations for structured matrices such as ours.

## 4 Numerical Experiments

We have tested the accuracy of the proposed algorithm experimentally. The fast Hessenberg reduction and the implicit DQR iterations using D-orthogonal

transformations have been implemented in Matlab<sup>2</sup> and then applied to find the approximate eigenvalues of real sign symmetric tridiagonal matrices that were modified by rank-one perturbations. The software is available upon request to the authors.

Our test suite consists of both random and specific input matrices. Random matrices  $A = T + \mathbf{u}\mathbf{v}^T \in \mathbb{R}^{n \times n}$  are generated in Matlab by setting  $\mathbf{u} = \text{rand}(n, 1)$ ,  $\mathbf{v} = \text{rand}(n, 1)$ . And  $T = (t_{i,j})$  is a sign symmetric tridiagonal matrix satisfying  $t_{i,i} = \alpha_i$ ,  $|t_{i,i+1}| = |t_{i+1,i}| = |\beta_i|$ , where  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_n] = \text{rand}(1, n)$ ,  $\boldsymbol{\beta} = [\beta_1, \dots, \beta_{n-1}] = \text{rand}(1, n - 1)$  and  $D \cdot T = T^T \cdot D$  where  $D = \text{diag}[\pm 1]$  is determined by the procedure

```

for  $k = 1:n$ 
  if  $\text{rand} < .5$ 
     $D(k, k) = -1;$ 
  else
     $D(k, k) = 1;$ 
  end
end

```

A first, detailed error analysis of Hessenberg reduction methods using non-orthogonal transformations appears in Wilkinson's book (Wilkinson, 1965). He states in (Wilkinson, 1965, ch. 6, sect. 46 )

that large errors in eigenvalues computations of reduced Hessenberg matrices occur when multipliers of large magnitude are used during the reduction process. More precisely, if the multipliers are as large as  $2^t$ , then  $2t$  digits can be lost in the accuracy of computed results. In our Hessenberg reduction procedure we monitor the size of D-orthogonal transformations as follows. We define the *control parameter* for the  $2 \times 2$  D-orthogonal matrix  $\mathcal{G}$  as  $\mu(\mathcal{G}) = \max(\text{abs}(\mathcal{G}))$ . If  $\mu(\mathcal{G})$  is greater than a specified value  $\mu$ , then a *breakdown* is said to have occurred and the algorithm aborts by returning *failure*. Table 1 reports the number of breakdowns as a function of the specified value  $\mu = 2^r$  over 100 tests with random matrices of size  $n = 1024$ .

$r$	7	9	11	13	16
breakdowns	98	70	5	1	0

**Table 1.** Number of breakdowns as function of the control parameter  $r$

By Wilkinson's analysis, the probability of having a 3 or 4 digits loss of ac-

<sup>2</sup> Matlab is a registered trademark of The Mathworks, Inc..

curacy in the computed eigenvalues of the transformed Hessenberg matrix is quite low, even when we use a control parameter as large as  $\mu = 2^{16}$  ( $= 65536$ ) to avoid breakdown.

In Figure 1, 2, 3 and 4 we show the numerical performances of our fast DQR eigenvalue algorithm for input matrices  $A$  in Hessenberg form satisfying (3.4). The value of the control parameter in the DQR algorithm is set to  $2^7 = 128$ . The algorithm proceeds by performing a single-shift DQR iteration using the Wilkinson shift defined as the eigenvalue  $\tilde{\lambda}$  of the trailing  $2 \times 2$  sub-matrix that is closest to the last diagonal entry of the current matrix. The shift can move the iteration into the complex plane, but this is no drawback since D-orthogonal transformations generally do the same. If a breakdown occurs in the single-shift DQR iteration, then a double-shift DQR iteration with shift polynomial  $p(z) = (z - \tilde{\lambda})^2$  is carried out. The rationale for this strategy is the convergence analysis of the single-shift DQR iteration developed in (Uhlig, 1997). Specifically, it is proved that the effect of a large D-orthogonal transformation at a certain step can be “cooled” at the subsequent iteration in such a way that the matrix obtained after two steps is computed robustly by a double-shift iteration. In the exceptional cases where the strategy does not work, then a sequence of at most  $m = 8$  single-shift DQR iterations using random shifts is repeated until the breakdown is bypassed. After  $m$  unsuccessful attempts with exceptional shifts the program reports an error message and the eigenvalue computation is completed by invoking the customary QR method.

In each experiment we measure the distance between the spectrum of the input matrix  $A$  as computed by the Matlab function `eig` and the set of approximations returned by our fast eigensolver. Let  $\lambda(A)$  and  $\tilde{\lambda}(A)$  denote the set of eigenvalues computed by the Matlab function `eig` applied to  $A$  and the set of eigenvalue approximations returned by our algorithm from the generating elements of  $A$ , respectively. One might define a measure of the distance between the sets  $\lambda(A)$  and  $\tilde{\lambda}(A)$  as

$$\text{dist}(\lambda(A), \tilde{\lambda}(A)) = \max\left\{ \max_{\tilde{\lambda} \in \tilde{\lambda}(A)} \|\tilde{\lambda} - \lambda(A)\|, \max_{\lambda \in \lambda(A)} \|\lambda - \tilde{\lambda}(A)\| \right\},$$

where  $\|\lambda - \tilde{\lambda}(A)\| = \min_{\tilde{\lambda} \in \tilde{\lambda}(A)} |\lambda - \tilde{\lambda}|$ . An inconvenience of this definition is that one eigenvalue in  $\lambda(A)$  can match up with different approximations in  $\tilde{\lambda}(A)$  and we do not get an appropriate global measure of the closeness of these two complex number sets. To circumvent this problem we apply a rook pivoting strategy for comparing the two sequences  $\{\lambda_k\} = \lambda(A)$  and  $\{\mu_k\} = \tilde{\lambda}(A)$ . First they are sorted according to:

```
while (length( $\lambda$ ) > 0)
     $lf = \text{length}(\lambda)$ ; [ $y, k$ ] = min(abs( $\lambda_1$ *ones(1,  $lf$ )-  $\mu$ ));
```

```

[z, l] = min(abs(mu_k*ones(1, lf) - lambda));
if z < y
    f = lambda_1; lambda_1 = lambda_l; lambda_l = f;
end
lambda_1 = []; mu_k = []; end

```

Then we compute the discrepancy of the two sorted sequences in the weighted 1-norm, normalized to give the unit n-cube a diagonal of length 1. The result is referred to as the *computed error* of the two sequences. According to the analysis performed in (Tisseur, 1996) for the orthogonal QR eigenvalue algorithm, the *expected error* of computed eigenvalues is bounded above by  $\epsilon \cdot 2^{14} \cdot \|A\|_2 \cdot \max(\text{condeig}(A))$ , where  $\epsilon = 2^{-52}$  is the machine precision,  $\|\cdot\|_2$  denotes the Euclidean norm, and `condeig` is an internal Matlab function for calculating eigenvalue condition numbers.

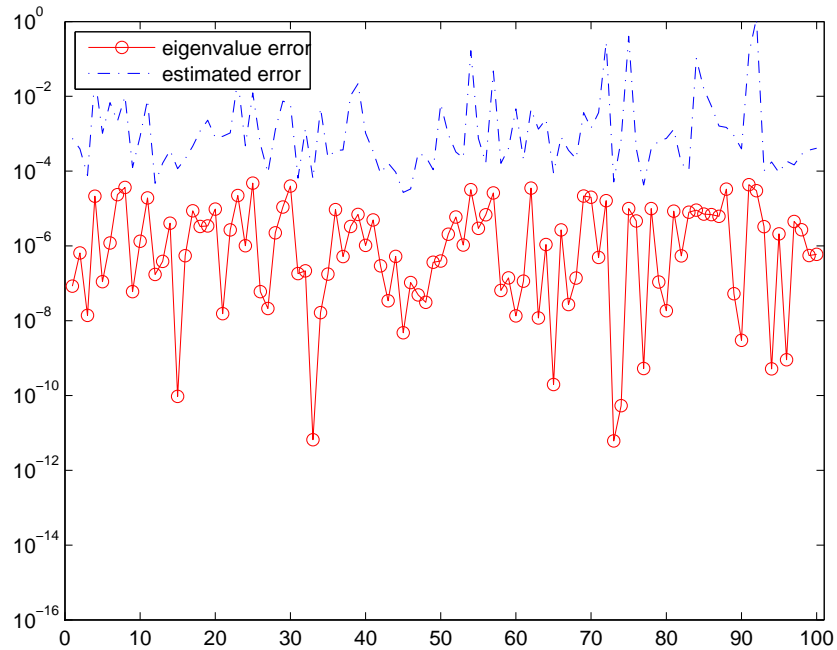


Fig. 1. Errors for moderately ill-conditioned matrices for 100 tests

Figure 1 shows the error plot generated in trial runs with one hundred random test matrices of size  $n = 1024$ . These were obtained from the Hessenberg reduction of random matrices  $A$  generated as described earlier. The reduction

phase has a slight effect on the norm and conditioning of the input matrices. The results indicate a good match of the two eigenvalue sets, well in accordance with the expected bounds. The average number of iterations per eigenvalues is 3. The single-shift/double-shift strategy is successful in 78 percent of our samples. All the remaining cases are solved by performing a small (less than 5) number of exceptional shifts. The algorithm never reports failure.

In Figure 2 the input matrices of the DQR eigenvalue routine are generated by the procedure described earlier with  $\mathbf{v}^T = [0, \dots, 0, 1]$ . The average number of DQR iterations per eigenvalues is again 3. The matrices are fairly well-conditioned and the results indicate an even better match of the two eigenvalue sets.

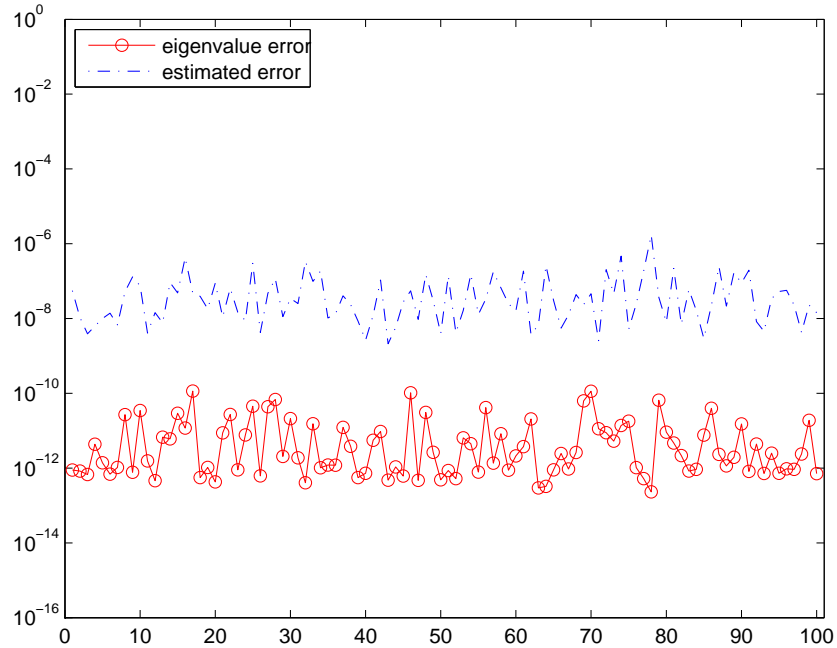


Fig. 2. Errors for well-conditioned matrices for 100 tests

Figure 3 illustrates the behavior of our algorithm applied for eigenvalue computation of sign symmetric tridiagonal matrices. The error plot is generated in trial runs with one hundred test matrices of size  $n = 1024$  of the form  $T + (\text{rand} + \sqrt{-1} \cdot \text{rand})I_n$ , where  $T$  is the tridiagonal sign symmetric matrix generated by  $\boldsymbol{\alpha} = \text{ones}(1, n)$ ,  $\boldsymbol{\beta} = \text{ones}(1, n - 1)$  and  $D \cdot T = T^T \cdot D$  where  $D = \text{diag}[1, 1, 1, -1, 1, -1, \dots]$ . For these matrices the double-shift technique

is generally successful by jumping over the difficult steps without requiring exceptional shifts. The average number of iterations per eigenvalues is 2.

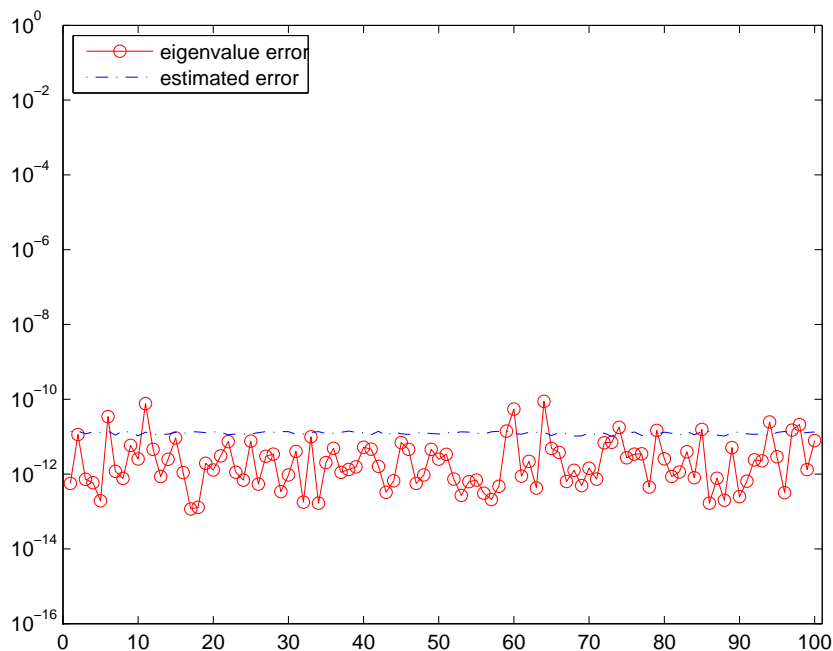


Fig. 3. Errors for sign symmetric tridiagonal matrices for 100 tests

Finally, we apply the DQR-based eigenvalue solver for the refinement of polynomial zeros as performed by Fortune's algorithm (Fortune, 2001, 2002). We consider the characteristic polynomial  $p(z)$  of the skew-symmetric tridiagonal Toeplitz matrix of size  $n = 32$  having zero diagonal entries and unit subdiagonal entries. We compute the vector  $\mathbf{p}$  of the coefficients of  $p(z)$  in the usual power form using high precision (vpa) arithmetic and we approximate its zeros by the Matlab function `roots` applied to `double(p)`. Then the Lagrange form of  $p(z)$  is computed by interpolating  $p(z)$  on the approximate zeros using vpa arithmetic. In this way we obtain a new diagonal plus rank-one matrix having  $p(z)$  as its characteristic polynomial. The eigenvalues of this matrix are computed by means of our DQR-based eigenvalue solver and the returned approximations are the refined zeros. In Figure 4 we show how the two sets of approximations match the eigenvalues of the initial matrix computed using vpa arithmetic.

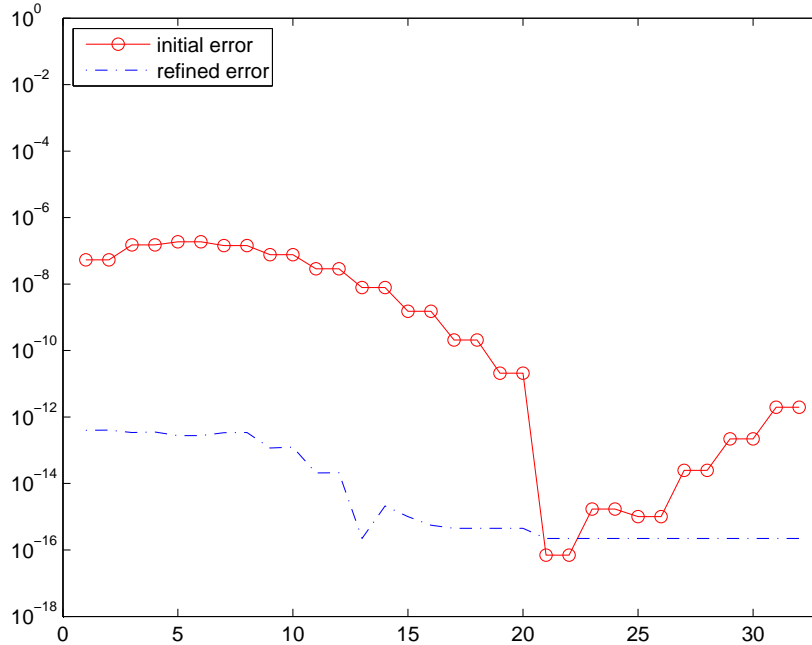


Fig. 4. Errors before and after the refinement of the zeros of a polynomial of degree 32

## 5 Conclusion

In this paper we have presented a new eigenvalue algorithm that uses generalized orthogonal transformations for sign symmetric tridiagonal matrices modified by small rank perturbations. The algorithm is appealing because of its low memory requirements and low computational cost. Exploiting the structure of the associated eigenvalue problems enables us to obtain the eigenvalues in quadratic time  $O(n^2)$  and in  $O(n)$  memory. The results of numerical experiments indicate that the accuracy of the output depends on the magnitudes of the transformations involved. The single-shift/double-shift strategy seems to be effective in controlling the norm of the transformations during the iterative phase. The design of adaptive procedures for tuning the machine precision in the finite reduction to Hessenberg form would allow to achieve the same level of accuracy as is customary in orthogonal  $O(n^3)$  eigensolvers.

## References

- Amiraslani, A., Corless, R. M., Gonzalez-Vega, L., Shakoory, A., 2004. Polynomial algebra by values. Tech. Rep. 04-01, Ontario Research Centre for Computer Algebra.
- Berrut, J. P., Trefethen, L. N., 2004. Barycentric Lagrange interpolation. *SIAM Rev.* 46 (3), 501–517 (electronic).
- Braman, Karen; Byers, Ralph; Mathias, Roy The multishift  $QR$  algorithm. I. Maintaining well-focused shifts and level 3 performance. *SIAM J. Matrix Anal. Appl.* 23 (2002), no. 4, 929–947.
- Braman, Karen; Byers, Ralph; Mathias, Roy, 2002. The multishift  $QR$  algorithm. II. Aggressive early deflation. *SIAM J. Matrix Anal. Appl.* 23 (2002), no. 4, 948–973.
- Eidelman, Y., Gemignani, L., Gohberg, I., 2007. On the fast reduction of a quasiseparable matrix to Hessenberg and tridiagonal forms. *Linear Algebra Appl.* 420.
- Eidelman, Y., Gemignani, L., Gohberg, I., 2008. Efficient eigenvalue computation for quasiseparable Hermitian matrices under low rank perturbations. *Numer. Algorithms* 47 (3), 253–273.
- Eidelman, Y., Gohberg, I., 2002. A modification of the Dewilde-van der Veen method for inversion of finite structured matrices. *Linear Algebra Appl.* 343-344, 419–450.
- Fortune, S., 2001. Polynomial root finding using iterated eigenvalue computation. In: *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*. ACM, New York, pp. 121–128 (electronic).
- Fortune, S., 2002. An iterated eigenvalue algorithm for approximating roots of univariate polynomials. *J. Symbolic Comput.* 33 (5), 627–646, computer algebra (London, ON, 2001).
- John G. F. Francis, 1961, The QR transformation, a unitary analogue to the LR transformation – part 1. *Comput. J.*, vol. 4 (1961/62), 265 - 271.
- John G. F. Francis, 1962. The QR transformation – part 2. *Comput. J.*, vol. 4 (1961/62), 332 - 345.
- Gohberg, I., Lancaster, P., Rodman, L., 1983. *Matrices and indefinite scalar products*. Vol. 8 of *Operator Theory: Advances and Applications*. Birkhäuser Verlag, Basel.
- Hermann, T., 1996. On the stability of polynomial transformations between Taylor, Bernstein and Hermite forms. *Numer. Algorithms* 13 (3-4), 307–320 (1997).
- Higham, N. J., 2004. The numerical stability of barycentric Lagrange interpolation. *IMA J. Numer. Anal.* 24 (4), 547–556.
- Lanczos, C., 1951. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. In: *Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery, 1949*. Harvard University Press, Cambridge, Mass., pp. 164–206.
- Tisseur, F., 1996. Backward stability of the QR algorithm. Tech. Rep. 239, UMR 5585 Lyon Saint-Etienne.
- Uhlig, F., 1997. The DQR algorithm, basic theory, convergence, and conditional stability. *Numer. Math.* 76 (4), 515–553.
- Uhlig, F., 1999. General polynomial roots and their multiplicities in  $O(n)$  memory and  $O(n^2)$  time. *Linear and Multilinear Algebra* 46 (4), 327–359.

Wilkinson, J. H., 1965. The algebraic eigenvalue problem. Monographs on Numerical Analysis. The Clarendon Press Oxford University Press, New York, Oxford Science Publications.