

An orthogonal similarity reduction of a matrix to semiseparable form.

R. Vandebril
M. Van Barel
N. Mastronardi

Report TW 355, February 2003



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

An orthogonal similarity reduction of a matrix to semiseparable form.

R. Vandebril

M. Van Barel

N. Mastronardi

Report TW 355, February 2003

Department of Computer Science, K.U.Leuven

Abstract

It is well known how any symmetric matrix can be reduced by an orthogonal similarity transformation into tridiagonal form. Once the tridiagonal matrix has been computed, several algorithms can be used to compute either the whole spectrum or part of it. In this paper, we propose an algorithm to reduce any symmetric matrix into a similar semiseparable one of semiseparability rank 1, by orthogonal similarity transformations. A remarkable feature of the algorithm is that, after few steps of it, the largest eigenvalues, in absolute value, are already computed with high precision.

Once the semiseparable matrix has been computed, to compute the whole spectrum either the same algorithm can be iterated or algorithms for computing the eigendecomposition of diagonal plus semiseparable matrices, available in the literature, can be used.

These features of the proposed algorithm are confirmed by some numerical experiments.

Keywords : Similarity transformation, semiseparable matrix, QR algorit
AMS(MOS) Classification : Primary : 15A18, Secondary : 15A21, 65F15.

An orthogonal similarity reduction of a matrix to semiseparable form. *

R. VANDEBRIL¹, M. VAN BAREL¹ and N. MASTRONARDI² †

¹*Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven (Heverlee), Belgium. email: {Raf.Vandebril, Marc.VanBarel}@cs.kuleuven.ac.be*

²*Istituto per le Applicazioni del Calcolo “M. Picone”, sez. Bari, Consiglio Nazionale delle Ricerche, Via G. Amendola, 122/I, I-70126 Bari, Italy. email: irmanm21@area.ba.cnr.it*

Abstract.

It is well known how any symmetric matrix can be reduced by an orthogonal similarity transformation into tridiagonal form. Once the tridiagonal matrix has been computed, several algorithms can be used to compute either the whole spectrum or part of it. In this paper, we propose an algorithm to reduce any symmetric matrix into a similar semiseparable one of semiseparability rank 1, by orthogonal similarity transformations. A remarkable feature of the algorithm is that, after few steps of it, the largest eigenvalues, in absolute value, are already computed with high precision.

Once the semiseparable matrix has been computed, to compute the whole spectrum either the same algorithm can be iterated or algorithms for computing the eigendecomposition of diagonal plus semiseparable matrices, available in the literature, can be used.

These features of the proposed algorithm are confirmed by some numerical experiments.

AMS subject classification: 15A18, 15A21, 65F15

Key words: Similarity transformation, semiseparable matrix, QR algorithm.

1 Introduction.

The standard procedure to compute the eigendecomposition of a dense symmetric matrix first reduces it into a similar symmetric tridiagonal one by means of orthogonal transformations. This step is accomplished in $\frac{4}{3}n^3 + O(n^2)$ flops¹ and it is the most expensive one. Once the similar symmetric tridiagonal matrix has been computed, many fast and stable algorithms can be considered to compute its spectral decomposition (see, for instance, [8, 11, 14] and the references therein).

*Received ???, Revised ???, Communicated by ???.

†The research of the second author was supported by the Research Council K.U.Leuven, project OT/00/16 (SLAP: Structured Linear Algebra Package), by the Fund for Scientific Research–Flanders (Belgium), projects G.0078.01 (SMA: Structured Matrices and their Applications), G.0176.02 (Asymptotic analysis of the convergence behavior of iterative methods in numerical linear algebra), and G.0184.02 (CORFU: Constructive study of Orthogonal Rational Functions), by the K.U.Leuven (Bijzonder Onderzoeksfonds), and by the Belgian Programme on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister’s Office for Science, Technology and Culture, project IUAP V-22 (Dynamical Systems and Control: Computation, Identification & Modelling). The scientific responsibility rests with the authors.

¹In this paper the definition given in [8] is adopted: “A flop is a floating point operation.” The square root is considered a flop, too.

Recently, many fast and stable algorithms that compute the eigendecomposition of symmetric diagonal plus semiseparable matrices have been developed [2, 3, 6, 13].

In this paper we consider an algorithm that transforms symmetric matrices into similar symmetric semiseparable ones by means of orthogonal transformations with $\frac{4}{3}n^3 + O(n^2)$ flops. Hence, combining the latter algorithm with one of those for computing the eigendecomposition of semiseparable matrices we have an alternative way to compute the eigenvalue decomposition of symmetric matrices, at the same computational complexity of the standard approach mentioned before.

Moreover, the proposed algorithm, while running to the completion, gives information on the spectrum of the similar initial matrix. In fact, we will show that the proposed algorithm can be considered as a kind of subspace-like iteration method, where the size of the subspace increases by one dimension at each step of the algorithm. Hence, in many cases, after few steps of the algorithm, the largest eigenvalues and the corresponding eigenvectors are already computed with high precision. Therefore it turns out that the proposed algorithm is a good candidate for computing an approximation of the subspace associated to the largest eigenvalues of large dense matrices. Such problems arise in many different areas, such as, principal component analysis, data mining, magnetic resonance spectroscopy, microarray data analysis, gene expression data, computing the eigensolutions of the Schrödinger equation on a grid ... (see, for instance, [9, 15, 16, 19, 20, 1] and the references therein).

In this paper only symmetric matrices are handled. However, there is no loss of generality because it will be shown that similar techniques can be applied to real unsymmetric matrices, as well.

Furthermore an algorithm that transforms tridiagonal matrices into similar semiseparable ones is described in this paper, too.

The paper is organized as follows. In § 2 the definition and basic concepts of semiseparable matrices are introduced. The algorithm for reducing symmetric matrices into similar symmetric semiseparable ones is described in § 3. The unsymmetric case is considered in § 4 and the properties of the proposed algorithm are discussed in § 5. In § 6 different implementations of the proposed algorithm are described, together with the description of an algorithm for reducing symmetric tridiagonal matrices into similar symmetric semiseparable ones. The numerical experiments are shown in § 7 followed by the conclusions and future work.

2 Semiseparable matrices.

A semiseparable matrix is constructed from the upper and lower triangular part of two low-rank matrices. We will define a general semiseparable matrix of semiseparability rank r .

DEFINITION 2.1. *S is called a semiseparable matrix of semiseparability rank r if there exist two matrices R_1 and R_2 both of rank r , such that*

$$S = \text{triu}(R_1) + \text{tril}(R_2).$$

$\text{triu}(R_1)$ and $\text{tril}(R_2)$ denote respectively the upper triangular part of the matrix R_1 and the strictly lower triangular part of the matrix R_2 .

A special class of semiseparable matrices is the class of semiseparability rank 1. Because the two matrices R_1 and R_2 are both of rank 1 there exist four vectors \mathbf{u}, \mathbf{v} and \mathbf{s}, \mathbf{t} of the same length n , such that $R_1 = \mathbf{t} \mathbf{s}^T$ and $R_2 = \mathbf{u} \mathbf{v}^T$. This means that the upper and the strictly lower triangular part of S are the upper and the strictly lower triangular

part of two rank 1 matrices, respectively. Hence, the latter semiseparable matrix S looks like:

$$\begin{pmatrix} t_1 s_1 & t_1 s_2 & t_1 s_3 & \dots & t_1 s_n \\ v_1 u_2 & t_2 s_2 & t_2 s_3 & \dots & t_2 s_n \\ v_1 u_3 & v_2 u_3 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & & \\ v_1 u_n & v_2 u_n & \dots & & t_n s_n \end{pmatrix}.$$

In this paper we focus on the class of symmetric matrices, i.e. the matrices A satisfying $A = A^T$. A symmetric semiseparable matrix therefore looks as follows:

$$\begin{pmatrix} v_1 u_1 & v_1 u_2 & v_1 u_3 & \dots & v_1 u_n \\ v_1 u_2 & v_2 s_2 & v_2 u_3 & \dots & v_2 u_n \\ v_1 u_3 & v_2 u_3 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & & \\ v_1 u_n & v_2 u_n & \dots & & v_n u_n \end{pmatrix}.$$

In the next section we will show the main result of this paper, namely the algorithm to reduce a symmetric matrix with orthogonal similarity transformations to a symmetric semiseparable one.

3 Orthogonal similarity transformations to reduce a symmetric matrix to a semiseparable one.

An algorithm to transform a symmetric matrix into a semiseparable one is presented in this section. The constructive proof of the next theorem, provides the algorithm.

THEOREM 3.1. *Let A be a symmetric matrix. Then there exists an orthogonal matrix U such that*

$$U^T A U = S,$$

where S is a semiseparable matrix.

PROOF. The proof is a constructive one and it is made by induction on the rows of the involved matrix. We will construct a similar symmetric semiseparable matrix from a symmetric one. Let $A_0^{(0)} \equiv A$. Let $G_i^{(k)}$ be the Givens transformation, such that the product $A_{i-1}^{(k)} G_i^{(k)}$ has the entry $(i, n-k)$ annihilated and modifies the i th and the $(i+1)$ th columns of $A_{i-1}^{(k)}$.

Step 1 We will start by constructing a similarity transformation which will make the last two rows (columns) linearly dependent in the lower (upper) triangular part. To this end, we multiply $A_0^{(0)}$ to the left by $G_1^{(0)T}$ and to the right by $G_1^{(0)}$ to annihilate

the elements in position $(1, n)$ and $(n, 1)$ in $A_0^{(0)}$, respectively,

$$\begin{aligned} & \begin{matrix} \downarrow & \downarrow & & & & \\ \rightarrow & \left(\begin{array}{cccccc} \times & \times & \cdots & \times & \otimes \\ \times & \ddots & \vdots & \vdots & \times \\ \cdots & \cdots & \ddots & \vdots & \vdots \\ \times & \cdots & \cdots & \ddots & \times \\ \otimes & \times & \cdots & \times & \times \end{array} \right) & \xrightarrow{G_1^{(0)T} A_0^{(0)} G_1^{(0)}} & \left(\begin{array}{cccccc} \times & \times & \cdots & \times & \mathbf{0} \\ \times & \ddots & \vdots & \vdots & \times \\ \cdots & \cdots & \ddots & \vdots & \vdots \\ \times & \cdots & \cdots & \ddots & \times \\ \mathbf{0} & \times & \cdots & \times & \times \end{array} \right) \end{matrix} \\ & \qquad \qquad \qquad \updownarrow \\ & A_0^{(0)} \xrightarrow{G_1^{(0)T} A_0^{(0)} G_1^{(0)}} A_1^{(0)}. \end{aligned}$$

Then $A_1^{(0)} \equiv G_1^{(0)T} A_0^{(0)} G_1^{(0)}$.

Continuing this process of annihilating all the elements in the last row (column), except for the element in position $(n, n-1)$ ($(n-1, n)$), we have

$$\left(\begin{array}{cccccc} \times & \times & \cdots & \times & \mathbf{0} \\ \times & \ddots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \ddots & \times & \mathbf{0} \\ \times & \cdots & \times & \times & \times \\ \mathbf{0} & \cdots & \mathbf{0} & \times & \times \end{array} \right)$$

Multiplying to the left $A_{n-2}^{(0)}$ by $G_{n-1}^{(0)T}$, we have the following situation,

$$\begin{aligned} & \begin{matrix} \rightarrow & \left(\begin{array}{cccccc} \times & \times & \cdots & \times & \mathbf{0} \\ \times & \ddots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \ddots & \vdots & \mathbf{0} \\ \times & \cdots & \cdots & \times & \otimes \\ \mathbf{0} & \cdots & \mathbf{0} & \times & \times \end{array} \right) & \xrightarrow{G_{n-1}^{(0)T} A_{n-2}^{(0)}} & \left(\begin{array}{cccccc} \times & \times & \cdots & \times & \mathbf{0} \\ \times & \ddots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \ddots & \times & \mathbf{0} \\ \boxtimes & \cdots & \boxtimes & \times & \mathbf{0} \\ \boxtimes & \cdots & \boxtimes & \times & \times \end{array} \right) \end{matrix} \\ & \qquad \qquad \qquad \updownarrow \\ & A_{n-2}^{(0)} \xrightarrow{G_{n-1}^{(0)T} A_{n-2}^{(0)}} \tilde{A}_{n-2}^{(0)}. \end{aligned}$$

i.e., the last two rows are proportional with the exception of the entries in the last two columns, (to emphasize the linear dependency among the rows (columns) we denote by \boxtimes the element of the matrix belonging to these rows (columns)).

Let $\tilde{A}_{n-2}^{(0)} \equiv G_{n-1}^{(0)T} A_{n-2}^{(0)}$. Moreover, multiplying $\tilde{A}_{n-2}^{(0)}$ to the right by $G_{n-1}^{(0)}$, the last two columns become linearly dependent above the main diagonal, and, for

symmetry, the last two rows become linearly dependent below the main diagonal,

$$\begin{pmatrix} \times & \times & \cdots & \times & \mathbf{0} \\ \times & \ddots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \ddots & \vdots & \mathbf{0} \\ \boxtimes & \cdots & \boxtimes & \times & \mathbf{0} \\ \boxtimes & \cdots & \boxtimes & \otimes & \times \end{pmatrix} \xrightarrow{\tilde{A}_{n-2}^{(0)} G_{n-1}^{(0)}} \begin{pmatrix} \times & \times & \cdots & \boxtimes & \boxtimes \\ \times & \ddots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \ddots & \boxtimes & \boxtimes \\ \boxtimes & \cdots & \boxtimes & \boxtimes & \boxtimes \\ \boxtimes & \cdots & \boxtimes & \boxtimes & \boxtimes \end{pmatrix}$$

$$\Downarrow$$

$$\tilde{A}_{n-2}^{(0)} \xrightarrow{\tilde{A}_{n-2}^{(0)} G_{n-1}^{(0)}} A_{n-1}^{(0)}.$$

Then define $A_0^{(1)}$ in the following way $A_0^{(1)} \equiv A_{n-2}^{(0)}$.

Step 2 Let $m = n - r$, $1 \leq r < n$. Assume by induction that $A_0^{(m)}$, has the lower (upper) triangular part already semiseparable from row n up to row r (column r to n). We will now prove that we can make the lower (upper) triangular part semiseparable up to row $r - 1$ (column $r - 1$). Let us denote the elements of the rows which are dependent in the lower triangular part with \boxtimes . So matrix $A_0^{(m)}$ looks like:

$$A_0^{(m)} = \begin{pmatrix} \times & \cdots & \times & \boxtimes & \cdots & \boxtimes \\ \vdots & \ddots & \cdots & \vdots & \vdots & \vdots \\ \times & \cdots & \times & \boxtimes & \cdots & \boxtimes \\ \boxtimes & \cdots & \boxtimes & \boxtimes & \cdots & \boxtimes \\ \vdots & \cdots & \cdots & \cdots & \ddots & \vdots \\ \boxtimes & \cdots & \boxtimes & \boxtimes & \cdots & \boxtimes \end{pmatrix} \begin{matrix} \leftarrow 1 \\ \vdots \\ \leftarrow r-1 \\ \leftarrow r \\ \vdots \\ \leftarrow n \end{matrix}$$

When we annihilate the elements in position $(1, r)$ ($(r, 1)$) by using Givens transformations, all the elements in the same row (column) from r to n become zero, because the rows (columns) underneath are dependent. So annihilating all the elements up to element $(r - 2, r)$ ($(r, r - 2)$) gives us the following matrix,

$$A_{r-2}^{(m)} \equiv G_{r-2}^{(m)T} \cdots G_1^{(m)T} A_0^{(m)} G_1^{(m)} \cdots G_{r-2}^{(m)} = \begin{pmatrix} \times & \cdots & \times & \times & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \times & \cdots & \ddots & \times & \mathbf{0} & \vdots & \mathbf{0} \\ \times & \cdots & \times & \times & \boxtimes & \cdots & \boxtimes \\ \mathbf{0} & \cdots & \mathbf{0} & \boxtimes & \boxtimes & \cdots & \boxtimes \\ \vdots & \cdots & \vdots & \cdots & \cdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \boxtimes & \boxtimes & \cdots & \boxtimes \end{pmatrix} \begin{matrix} \leftarrow 1 \\ \vdots \\ \vdots \\ \leftarrow r-1 \\ \leftarrow r \\ \vdots \\ \leftarrow n \end{matrix}$$

Since the rows $r - 1$ and r are proportional for the indexes of columns greater than $r - 1$, applying $G_{r-1}^{(m)T}$ in order to annihilate the element $(r - 1, r)$, all the entries in row $r - 1$ with column index greater than $r - 1$ are annihilated, too. Furthermore,

plexity of the algorithm are given also in § 6. However, it is not hard to see that the complexity of the algorithm is $O(n^3)$. Once the semiseparable matrix has been computed, many algorithms can be considered to compute its eigenvalues (see, for instance, [2, 3, 6, 13]).

It is remarkable to notice that, the proposed algorithm, while running to the completion, gives information on the spectrum of the initial matrix. In particular, depending on the distribution of the eigenvalues, often, after few steps of the algorithm, the largest eigenvalues are already computed. This important feature of the algorithm is analyzed in § 5.

4 The reduction of an arbitrary matrix.

As already said in the introduction, it is well known how any matrix can be transformed into an upper Hessenberg one by an orthogonal similarity transformation. To find the eigenvalues of the original matrix, then the QR algorithm is applied to this upper Hessenberg matrix. By using an algorithm similar to that one described in the previous section, it is possible to transform every matrix into the sum of a semiseparable and a strictly upper triangular part. This is proved in the next theorem.

THEOREM 4.1. *Let A be an $n \times n$ matrix. Then there exist a symmetric semiseparable matrix S of rank 1, a strictly upper triangular matrix R and an orthogonal matrix U such that the following equation is satisfied:*

$$A = U(S + R)U^T,$$

i.e., A is orthogonally similar to the sum of a symmetric semiseparable and a strictly upper triangular matrix R .

PROOF. Following the same reduction as in the proof of Theorem 3.1, we can easily see that A can be written in the following way, by orthogonal similarity transformations,

$$U^T A U = \begin{pmatrix} v_1 u_1 & r_{1,2} & r_{1,3} & \dots & r_{1,n} \\ v_1 u_2 & v_2 u_2 & r_{2,3} & \dots & r_{2,n} \\ v_1 u_3 & v_2 u_3 & \ddots & & \vdots \\ \vdots & \vdots & \ddots & & \vdots \\ v_1 u_{n-1} & v_2 u_{n-1} & & v_{n-1} u_{n-1} & r_{n-1,n} \\ v_1 u_n & v_2 u_n & \dots & v_{n-1} u_n & v_n u_n \end{pmatrix},$$

where the lower triangular part comes from a semiseparable matrix. It can now easily be seen, that A can be written as:

$$U^T A U = \begin{pmatrix} v_1 u_1 & v_1 u_2 & \dots & v_1 u_n \\ v_1 u_2 & v_2 u_2 & & v_2 u_n \\ \vdots & & \ddots & \\ v_1 u_n & v_2 u_n & \dots & v_n u_n \end{pmatrix} + \begin{pmatrix} 0 & r_{1,2} - v_1 u_2 & r_{1,3} - v_1 u_3 & \dots & r_{1,n} - v_1 u_n \\ 0 & 0 & r_{2,3} - v_2 u_3 & \dots & r_{2,n} - v_2 u_n \\ \vdots & & & \ddots & \vdots \\ 0 & \dots & 0 & & r_{n-1,n} - v_{n-1} u_n \\ 0 & \dots & & & 0 \end{pmatrix}$$

which we can write as

$$A = U(S + R)U^T$$

□

Note that the algorithm to reduce an arbitrary matrix into an orthogonally similar matrix whose lower triangular part is semiseparable requires also $O(n^3)$ operations.

5 Convergence properties of the reduction.

At each step of the algorithm introduced in Theorem 3.1, one more row (column) is added by means of orthogonal transformations to the set of the rows (columns) of the matrix already proportional to each other. In this section, using arguments similar to those considered in [18, 17], we show that this algorithm can be interpreted as a kind of nested subspace iteration method [8], where the size of the vector subspace is increased by one and a change of coordinate system is made at each step of the algorithm. As a consequence, the blocks along the main diagonal in the part of the matrix already semiseparable, give information on the largest eigenvalues of the matrix. Given a matrix A and an initial subspace $\mathcal{S}^{(0)}$, the subspace iteration method [8] can be described as follows

$$\mathcal{S}^{(i)} = A\mathcal{S}^{(i-1)}, \quad i = 1, 2, 3, \dots$$

Under weak assumptions on A and $\mathcal{S}^{(0)}$, the $\mathcal{S}^{(i)}$ converge to an invariant subspace. We will see that the reduction algorithm from symmetric to semiseparable matrix can be seen as such a kind of subspace iteration, where the size of the subspace grows by one dimension at each step of the algorithm. Let $A_0^{(0)} = A$. Suppose we have only performed the first orthogonal similarity transformations such that the rows (columns) n and $n - 1$ are already proportional:

$$(5.1) \quad A_0^{(1)} = Q_1^T A_0^{(0)} Q_1,$$

where $A_0^{(1)}$ has the semiseparable structure in the rows (columns) n and $n - 1$ and $Q_1 = [q_1^{(1)}, \dots, q_n^{(1)}]$. From (5.1), we can write

$$(5.2) \quad A_0^{(0)} = Q_1 \begin{pmatrix} \times & \dots & \times & 0 \\ \vdots & & \vdots & \vdots \\ \times & \dots & \times & 0 \\ \times & \dots & \times & \times \end{pmatrix} \equiv Q_1 L_1.$$

Let e_1, \dots, e_n be the standard basis vectors of \mathbb{R}^n . From (5.2), because of the structure of L_1 , it can clearly be seen that:

$$A_0^{(0)} \langle e_n \rangle = \langle q_n^{(1)} \rangle,$$

where $\langle x, y, z, \dots \rangle$ denotes the subspace spanned by the vectors x, y, z, \dots . This means that the last column of $A_0^{(0)}$ and $q_n^{(1)}$ span the same one-dimensional space. In fact one subspace iteration step is performed on the column e_n . The first step of the algorithm is completed when the following orthogonal transformation is performed,

$$A_0^{(1)} = Q_1^T A_0^{(0)} Q_1.$$

The latter transformation can be interpreted as a change of coordinate system: $A_0^{(0)}$ and $A_0^{(1)}$ represent the same linear transformation with respect to different coordinate systems. Let $y \in \mathbb{R}^n$. Then y is represented in the new system by $Q_1 y$. This means that for the vector $q_n^{(1)}$ we get $Q_1 q_n^{(1)} = e_n$. Summarizing, this means that one step of subspace iteration on the subspace $\langle e_n \rangle$ is performed, resulting in a new subspace $\langle q_n^{(1)} \rangle$, and then, by means of a coordinate transformation, it is transformed back into the subspace $\langle e_n \rangle$. So, instead of working with a fixed matrix and changing subspaces, we work with fixed subspaces and changing matrices. Therefore, denoting by $z^{(i)}$ the eigenvector corresponding to the largest eigenvalue of $A_0^{(i)}$, $i = 0, 1, 2, \dots$, we can say that the sequence $\{z^{(i)}\}$ converges to e_n , and, consequently, the lower-right element of $A_0^{(i)}$ converges to the largest eigenvalue in absolute value.

The second step can be interpreted in a completely analogous way. Suppose we have already the semiseparable structure in the last two rows (columns). Then we perform the following similarity transformation on $A_0^{(1)}$,

$$(5.3) \quad A_0^{(2)} = Q_2^T A_0^{(1)} Q_2,$$

in order to make the rows (columns) n up to $n-2$ dependent. Using (5.3), $A_0^{(1)}$ can be written as follows,

$$A_0^{(1)} = Q_2 \begin{pmatrix} \times & \cdots & \times & 0 & 0 \\ \vdots & & & \vdots & \vdots \\ \times & \cdots & \times & 0 & 0 \\ \times & \cdots & \times & \times & 0 \\ \times & \cdots & \times & \times & \times \end{pmatrix} = Q_2 L_2.$$

Considering the subspace $\langle e_{n-1}, e_n \rangle$ and using the same notation as above, we have,

$$A_0^{(1)} \langle e_{n-1}, e_n \rangle = \langle q_{n-1}^{(2)}, q_n^{(2)} \rangle.$$

This means that the second step of the algorithm is a step of subspace iteration performed on a slightly grown subspace. For every new dependency that is created in the symmetric matrix $A_0^{(i)}$, the dimension of the subspace is increased by one.

This means that from step i , $i = 1, \dots, n$, (so there is dependency between the rows n up to $n-i$), all the consecutive steps perform subspace iterations on the subspace of dimension $n-i$. From [18], we know that these consecutive iterations on subspaces tend to create block upper triangular matrices. Hence, for a symmetric matrix these are block diagonal. Furthermore, the process works for all the nested subspaces at the same time, and so the matrices $A_0^{(i)}$ generated by the proposed algorithm, become more and more block diagonal, and the blocks contain the eigenvalues of the same absolute value. This explains why the lower-right block already gives a good estimate of the largest eigenvalues, since they are connected to a subspace on which the subspace iterations are performed mostly.

This insight also opens a lot of new perspectives. In a lot of problems, only the few largest eigenvalues (in absolute values) need to be computed [9, 15, 16, 19, 20, 1]. In such cases, the proposed algorithm gives the required information after only few

steps, without running the algorithm to completion. Moreover, because the sequence of similar matrices generated at each step of the algorithm converges to a block diagonal matrix, the original problem can be divided into smaller independent subproblems. Furthermore, if the algorithm is “iterated”, the sequence of semiseparable matrices generated converges to a block diagonal matrix.

We finish this section with a theorem from [18] concerning the speed of convergence of subspace iterations. (Theorem 5.4 [18])

DEFINITION 5.1. *Denote with \mathcal{S} and \mathcal{T} two subspaces, then the distance $d(\mathcal{S}, \mathcal{T})$ between these two subspaces is defined in the following way:*

$$d(\mathcal{S}, \mathcal{T}) = \sup_{s \in \mathcal{S}, \|s\|_2=1} \inf_{t \in \mathcal{T}} \|s - t\|_2.$$

Using this definition, we can state the following convergence theorem:

THEOREM 5.1. *Let $A \in \mathbb{C}^{n \times n}$, and let p be a polynomial of degree $\leq n$. Let $\lambda_1, \dots, \lambda_n$ denote the eigenvalues of A , ordered so that $|p(\lambda_1)| \geq |p(\lambda_2)| \geq \dots \geq |p(\lambda_n)|$. Suppose k is a positive integer less than n for which $|p(\lambda_k)| > |p(\lambda_{k+1})|$. Let (p_i) be a sequence of polynomials of degree $\leq n$ such that $p_i \rightarrow p$ as $i \rightarrow \infty$ and $p_i(\lambda_j) \neq 0$ for $j = 1, \dots, k$ and all i . Let $\rho = |p(\lambda_k)|/|p(\lambda_{k+1})|$. Let \mathcal{T} and \mathcal{U} be the invariant subspaces of A associated with $\lambda_1, \dots, \lambda_k$ and $\lambda_{k+1}, \dots, \lambda_n$ respectively. Consider the nonstationary subspace iteration*

$$\mathcal{S}_i = p_i(A)\mathcal{S}_{i-1}$$

where \mathcal{S}_0 is a k -dimensional subspace of \mathbb{C}^n satisfying $\mathcal{S} \cap \mathcal{U} = \{0\}$. Then for every $\hat{\rho}$ satisfying $\rho < \hat{\rho} < 1$ there exists a constant \hat{C} such that

$$d(\mathcal{S}_i, \mathcal{T}) \leq \hat{C}\hat{\rho}^i, \quad i = 1, 2, 3, \dots$$

In our case Theorem 5.1 can be applied with the polynomials $p_i(z)$ and $p(z)$ chosen in the following sense: $p_i(z) = p(z) = z$.

6 The algorithms and operation count.

This section is divided in different parts, because the algorithm can be implemented in some different ways. Every implementation requires a different number of operations.

6.1 Implementation with Givens transformations.

The algorithm can be implemented completely with Givens transformations, as described in Theorem 3.1. When we perform everything with Givens transformations we have the following algorithm:

ALGORITHM 6.1. *Suppose we have the following function:*

```
A=SymGiv(A, k, l, m)
```

which transforms A into $G^T * A * G$ where G^T is the Givens transformation such that $G^T * A$ eliminates the element in position (k, m) using the element (l, m) . Using this function our algorithm looks as follows: (in matlab style notation)

```
for j=n:-1:2
  for i=2:(j-2)
    A=SymGiv(A, i-1, i, j);
```

```

end
for i=j:1:n
    A=SymGiv(A,i-1,i,i);
end
end

```

The computational complexity can be easily computed. When exploiting the symmetry and the fact that the lower part always becomes semiseparable (which can easily be stored in two vectors, rather than in a full matrix), we derive the following count:

$$\sum_{j=n:-1:2} \left(\sum_{i=2:j-1} (6 * (j-1) + \alpha) + \sum_{i=j:n} \beta \right)$$

where α and β are two constants. This gives us the following complexity:

$$2n^3 + O(n^2)$$

6.2 Implementation with Householder and Givens transformations.

The second way to implement the algorithm is to use instead of Givens, Householder transformations for annihilating the first elements. This means that the first innerloop in Algorithm 6.1 can be replaced by performing one Householder transformation. The second inner loop is not changed.

ALGORITHM 6.2. *Suppose we have the following function:*

```
A=SymHous(A,k,m)
```

*which transforms A into $H^T * A * H$ where H is the Householder transformation such that $H^T * A$ eliminates the elements in column m from position 1 up to the element k. Using this function and the function SymGiv from Algorithm 6.1, our algorithm looks like: (in matlab style notation)*

```

for j=n:-1:2
    A=SymHous(A,j-2,j);
    for i=j:1:n
        A=SymGiv(A,i-1,i,i);
    end
end

```

Because the inner loop only gives an extra $O(n^2)$ operations, the overall cost of this algorithm is the same as the complexity to tridiagonalize a symmetric matrix by means of Householder transformations. Therefore the complexity equals:

$$(6.1) \quad \frac{4}{3}n^3 + O(n^2).$$

6.3 Reduction to a tridiagonal matrix.

The third approach to reduce a symmetric matrix into semiseparable form is the following. First we reduce the matrix into a similar tridiagonal one by standard algorithms (This can be achieved with Householder or Givens transformations (see, i.e., [8])). The latter tridiagonal matrix can then be reduced into the semiseparable form. The reduction from tridiagonal into semiseparable form will be performed by the following algorithm:

ALGORITHM 6.3. *We use the same notation for the symmetric Givens transformation as in Algorithm 6.1.*

```

for j=n:-1:2
  for i=j:1:n
    A=SymGiv(A, i-1, i, i);
  end
end
end

```

It is clear that the first inner loop of Algorithm 6.1 is not needed in this case, because of the tridiagonal structure of the matrix. We will now compute the extra cost needed to transform this tridiagonal to semiseparable form. The calculations we make are optimized in all possible ways, exploiting the structure of the matrix, which means, that multiplications and summations with zeros are not counted. Also semiseparable parts are not treated as full matrices but as products of vectors. The calculation of the Givens gives 7 flops, as described in [8]. The multiplication on the left hand and right hand side takes 10 operations, when everything is optimized and the semiseparable part is stored in two vectors. Therefore, the computational complexity is

$$\sum_{j=n:-1:2} \left(\sum_{i=j:1:n} 10 \right) = 5n^2 + O(n).$$

6.4 Reduction of a nonsymmetric matrix.

As already mentioned, this algorithm applied to nonsymmetric matrices, reduces the matrix to the sum of a semiseparable and an upper triangular matrix. To implement this the same algorithms as mentioned above can be used. We now take a look at the complexity of this reduction. Let us assume that we first reduce the matrix to an upper Hessenberg one, and then apply the algorithm to make the lower triangular part semiseparable (This is the same algorithm as applied to tridiagonal matrices). Then we first have the reduction to Hessenberg form, which costs $8n^3 + O(n^2)$ flops. The reduction from Hessenberg to lower semiseparable form costs $5n^2 + \frac{7}{2}n^2 + O(n)$. $5(n^2)$ flops are needed to reduce the lower triangular part into a semiseparable form and $\frac{7}{2}n^2$ flops have to be performed on the upper triangular part.

6.5 Comparison with other algorithms.

The algorithm based on Householder and Givens transformations, and the algorithm which first tridiagonalizes the matrix, both have the same complexity order. They perform both the same number of Givens and Householder transformations. Their complexity is a little bit less than the algorithm completely based on Givens transformations. When we now compare the algorithm which tridiagonalizes a symmetric matrix, with the algorithm which constructs the similar semiseparable matrix, then we find that the latter algorithm only performs $5n^2$ more than the former algorithm.

This means that both algorithms are of the same overall complexity, namely $O(n^3)$. Still one small remark has to be made, when one for example wants to use this algorithm to find information about the eigenvalues, as explained in Section 5, one can still reduce the complexity. Because the algorithm tends to form a block diagonal matrix, it is possible to cut off, during the algorithm, blocks. This step will reduce the complexity.

7 Numerical experiments.

As we already indicated, to solve a symmetric eigenvalue problem, we can use the traditional approach, i.e., first reduce the matrix to tridiagonal form and then apply the QR algorithm on this tridiagonal matrix. Or we can use an alternative approach, i.e., first reduce the matrix to semiseparable form and then apply the QR algorithm on this semiseparable matrix. To show the viability of the alternative approach, we performed two experiments.

7.1 Convergence to the largest eigenvalues

In this first experiment, we compare the behaviour of the proposed algorithm with that one of the Lanczos' algorithm² [8] applied to the matrix

$$A = Q \operatorname{diag}(\mathbf{d}) Q^T,$$

wher Q is a random orthogonal matrix generated by `matlab` and \mathbf{d} is the following vector,

$$d_i = \begin{cases} d_1 = -30, d_2 = -20, d_3 = -10, \\ \operatorname{randn}(1) & \text{if } 4 \leq i \leq 77 \\ d_{78} = 10, d_{79} = 20, d_{80} = 30. \end{cases}$$

The sequence of matrices generated by the Lanczos' algorithm will be denoted by T_i , $i = 1, 2, \dots$. Before applying the proposed method, the matrix A is permuted such that the elements in the main diagonal are increasingly ordered, i.e., $|a_{i,i}| \leq |a_{i+1,i+1}|$, $i = 1, \dots, n-1$.

In the first two columns of table 7.1, the eigenvalues of the matrices T_i and ($A^{(i)}(80 - i + 1 : 80, 80 - i + 1 : 80)$) are listed, for $i = 6, 10, 14$, respectively. Moreover, in the last column of table 7.1, the eigenvalues of the semiseparable submatrices ($A^{(i)}(75 : 80, 75 : 80)$) are reported, for $i = 6, 10, 14$, respectively. We can see that the proposed algorithm gives better results of the Lanczos' algorithm. Moreover, for i large enough, a good approximation of the 6 largest eigenvalues in absolute value of the matrix can be gained by computing the eigenvalues of the semiseparable submatrix $A^{(i)}(75 : 80, 75 : 80)$. In fact, as said in § 5, the sequence of matrices $A^{(i)}$ converges to a block diagonal matrix. In table 7.2 we show the 8×8 semiseparable submatrix $A^{(14)}(73 : 80, 73 : 80)$, generated after 14 steps of the proposed algorithm. It is remarkable to see that the elements not belonging to the diagonal blocks are already very small in magnitude.

7.2 Speed comparison.

In the first experiment, the speed of the two methods is compared. To this end, some random tridiagonal and semiseparable matrices are generated that are orthogonally similar to each other. Then, we apply the QR algorithm with shift [8] to the tridiagonal and the semiseparable matrix, respectively, and we compared the number of iterations needed to satisfy a certain stopping criterion for each of the eigenvalues. More precisely, an eigenvalue is considered computed, if the corresponding subdiagonal element is smaller in magnitude than 10^{-12} .

It is well known that the QR algorithm applied to a tridiagonal matrix requires $O(n)$ operations for each iteration step. In a further paper, we will show that this is also true

²A random vector is considered as the initial vector of the Lanczos' algorithm.

Table 7.1: Left: Eigenvalues of the tridiagonal matrices of order i , generated by i steps than the Lanczos' algorithm, for $i = 6, 10, 14$, respectively. Centre : Eigenvalues of the semiseparable submatrices $A^{(i)}((80 - i + 1 : 80, 80 - i + 1 : 80))$ generated by i steps of the proposed method, for $i = 6, 10, 14$, respectively. Right: Eigenvalues of the semiseparable submatrices $A^{(i)}(75 : 80, 75 : 80)$ generated by i steps of the proposed method, for $i = 6, 10, 14$, respectively.

$\text{eig}(T_6)$	$\text{eig}(A^{(6)}(75 : 80, 75 : 80))$	$\text{eig}(A^{(6)}(75 : 80, 75 : 80))$
$-2.985936735616989e + 01$	$-2.997071076484503e + 01$	$-2.997071076484503e + 01$
$-1.582680968620563e + 01$	$-1.998290973482237e + 01$	$-1.998290973482237e + 01$
$-5.576417612704368e + 00$	$-8.715379354248411e + 00$	$-8.715379354248411e + 00$
$3.162612757991903e - 01$	$6.573204227086380e + 00$	$6.573204227086380e + 00$
$1.050775347998979e + 01$	$1.997761137252159e + 01$	$1.997761137252159e + 01$
$2.999232864847275e + 01$	$2.993200008843372e + 01$	$2.993200008843372e + 01$
$\text{eig}(T_{10})$	$\text{eig}(A^{(10)}(71 : 80, 71 : 80))$	$\text{eig}(A^{(10)}(75 : 80, 75 : 80))$
$-2.99999999994999e + 01$	$-2.9999999999980e + 01$	$-2.9999999999769e + 01$
$-1.999999949230353e + 01$	$-1.99999999999829e + 01$	$-1.99999999996583e + 01$
$-9.999785201795993e + 00$	$-9.99999972439188e + 00$	$-9.99999454207639e + 00$
$-1.745258286111385e + 00$	$-1.644882672528439e + 00$	$9.99998271316842e + 00$
$-7.008173720102790e - 01$	$-9.973669620026927e - 01$	$1.99999999994123e + 01$
$5.350227010997197e - 01$	$8.363911575479067e - 01$	$2.9999999999376e + 01$
$1.550299981928200e + 00$	$1.990493215812659e + 00$	
$9.999989627741455e + 00$	$9.99999898945770e + 00$	
$1.99999998660500e + 01$	$1.9999999999688e + 01$	
$2.9999999995332e + 01$	$2.999999999962e + 01$	
$\text{eig}(T_{14})$	$\text{eig}(A^{(14)}(67 : 80, 67 : 80))$	$\text{eig}(A^{(14)}(75 : 80, 75 : 80))$
$-3.00000000000000e + 01$	$-2.9999999999993e + 01$	$-2.9999999999993e + 01$
$-1.9999999999998e + 01$	$-1.9999999999996e + 01$	$-1.9999999999997e + 01$
$-9.9999999999966e + 00$	$-9.9999999999993e + 00$	$-9.99999999998787e + 00$
$-2.167237147182936e + 00$	$-1.953158086013472e + 00$	$9.99999999994975e + 00$
$-1.518438468233005e + 00$	$-1.657844192350129e + 00$	$1.9999999999998e + 01$
$-9.419626065923228e - 01$	$-1.310048116883243e + 00$	$2.9999999999993e + 01$
$-2.413393354762497e - 01$	$-5.670493325847645e - 01$	
$4.345160276760946e - 01$	$6.081895139847837e - 01$	
$9.251950481138359e - 01$	$1.034515504944102e + 00$	
$1.452322355996672e + 00$	$1.569293290753957e + 00$	
$2.168650824692831e + 00$	$2.181739795568665e + 00$	
$9.9999999999943e + 00$	$9.9999999999995e + 00$	
$2.00000000000000e + 01$	$1.9999999999995e + 01$	
$2.9999999999999e + 01$	$2.9999999999991e + 01$	

Table 7.2: The semiseparable submatrix $A^{(14)}(73 : 80, 73 : 80)$ generated by 14 steps of the proposed method.

$-1.1e+00$	$-1.5e+00$	$1.2e-07$	$2.0e-07$	$6.8e-12$	$2.4e-11$	$2.5e-13$	$-4.8e-13$
$-1.5e+00$	$1.4e+00$	$-1.2e-07$	$-2.0e-07$	$-6.7e-12$	$-2.4e-11$	$-2.5e-13$	$4.7e-13$
$1.2e-07$	$-1.2e-07$	$5.2e+00$	$8.5e+00$	$2.8e-04$	$1.0e-03$	$1.0e-05$	$-2.0e-05$
$2.0e-07$	$-2.0e-07$	$8.5e+00$	$-5.2e+00$	$-1.7e-04$	$-6.3e-04$	$-6.5e-06$	$1.2e-05$
$6.8e-12$	$-6.7e-12$	$2.8e-04$	$-1.7e-04$	$5.3e+00$	$1.9e+01$	$1.9e-01$	$-3.7e-01$
$2.4e-11$	$-2.4e-11$	$1.0e-03$	$-6.3e-04$	$1.9e+01$	$-5.3e+00$	$-5.5e-02$	$1.0e-01$
$2.5e-13$	$-2.5e-13$	$1.0e-05$	$-6.5e-06$	$1.9e-01$	$-5.5e-02$	$1.4e+01$	$-2.6e+01$
$-4.8e-13$	$4.7e-13$	$-2.0e-05$	$1.2e-05$	$-3.7e-01$	$1.0e-01$	$-2.6e+01$	$-1.4e+01$

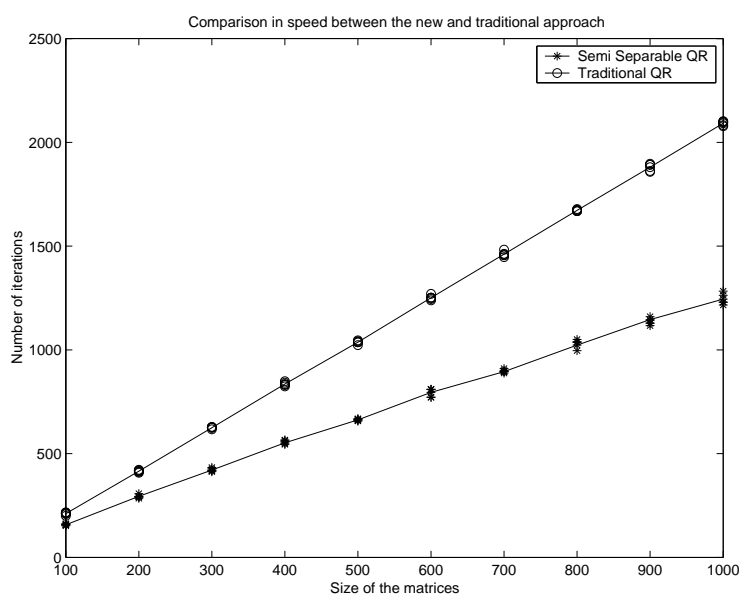


Figure 7.1: A speed comparison between the two QR algorithms

for the QR algorithm applied to a semiseparable matrix. However, each iteration step costs almost twice as much as an iteration step applied to the tridiagonal. On the other hand, Figure 7.1 shows that the alternative approach requires less iterations compared to the tridiagonal approach.

7.3 Accuracy comparison.

In a second experiment, we compare the accuracy of the two approaches. To this end, given a set of eigenvalues $\{\lambda_i\}$, a symmetric matrix A is constructed by performing an orthogonal similarity transformation with a random orthogonal matrix. The eigenvalues

of this matrix A are approximated using the QR algorithm implemented in `matlab`, for the classical approach, applied to the tridiagonal and, for the alternative approach, applied to the semiseparable matrix, both similar to matrix A . So, the sets $\{\lambda_{T,i}\}$ and $\{\lambda_{S,i}\}$ are obtained respectively. To measure the accuracy of the computed eigenvalues, we use the maximum of the relative errors between corresponding eigenvalues $\lambda_{T,i}$ and λ_i , and between $\lambda_{S,i}$ and λ_i respectively.

The different sets of eigenvalues are the following:

1. The eigenvalues are chosen equally spaced in the interval $(0, 1]$.
2. The eigenvalues are $1, 2, 3, \dots, n$ with n the dimension of the matrix.
3. The eigenvalues are the same as in the first case but now a small eigenvalue of the order 10^{-6} is added to the spectrum.
4. The same eigenvalues as in the previous case but now instead of a small one a large one is added of the order about 10^8 .
5. The eigenvalues are constructed in such a way that in the middle of the spectrum there are two eigenvalues not so different from each other.

The main objective of these experiments is to check that the accuracy of the two approaches is comparable for larger matrix dimensions and in case there are small, large or close eigenvalues.

In the following figures we show the results for the different choices of the spectrum.

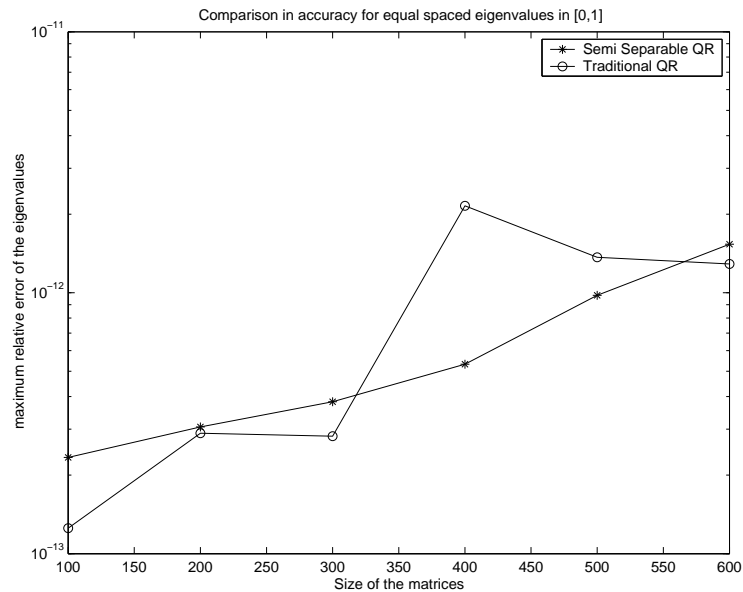


Figure 7.2: Equal spaced eigenvalues in $(0, 1]$

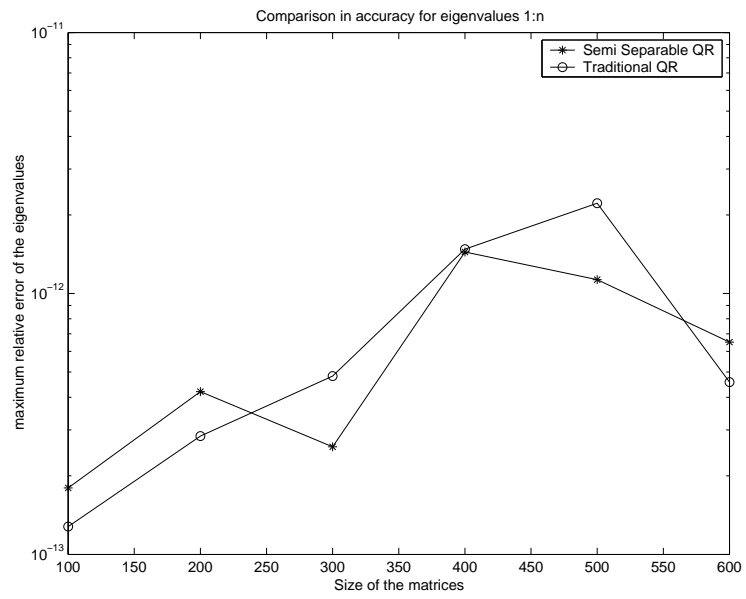


Figure 7.3: Eigenvalues 1 : n

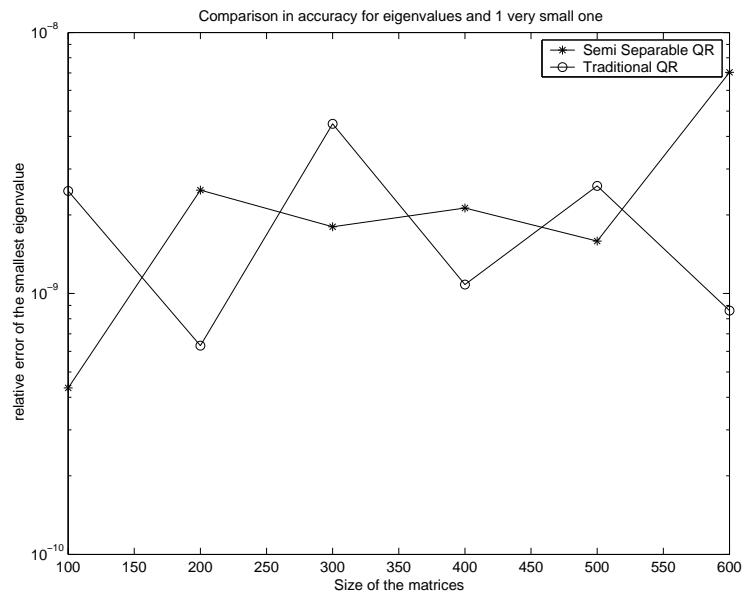


Figure 7.4: One small eigenvalue

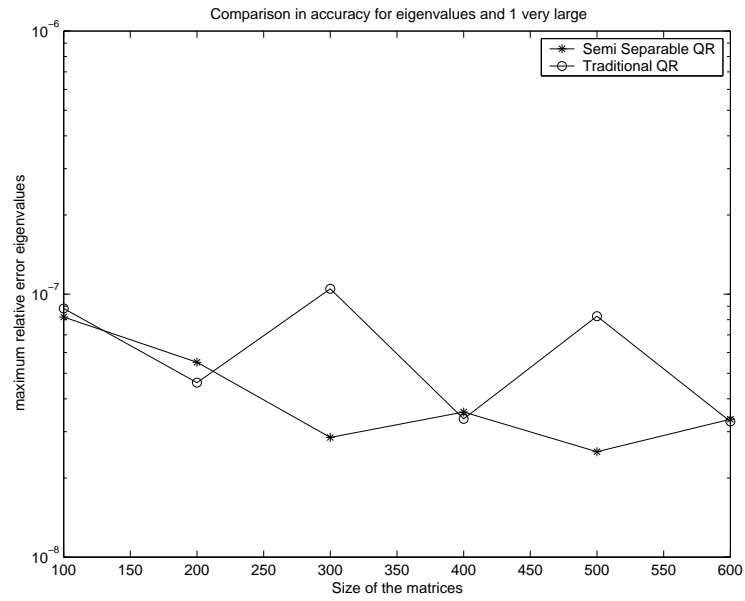


Figure 7.5: One large eigenvalue

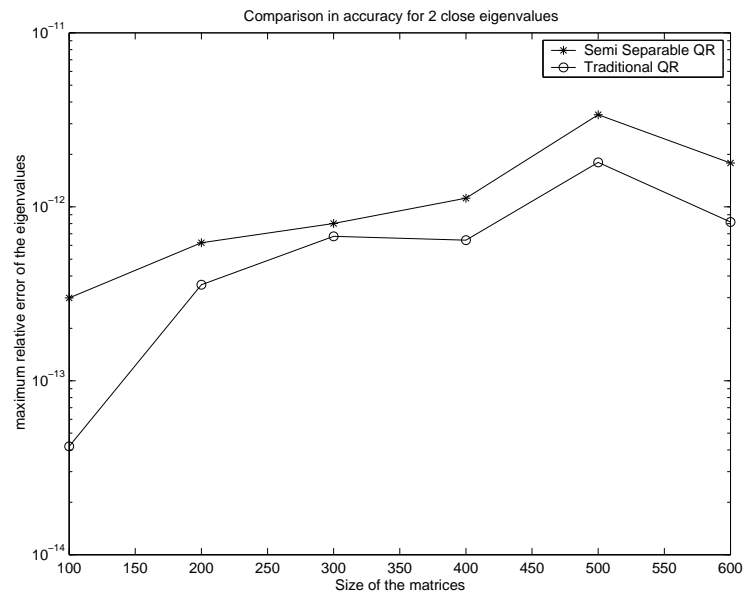


Figure 7.6: Two very close eigenvalues

8 Conclusion and future work.

In this paper an algorithm to transform a symmetric matrix into a semiseparable one by means of orthogonal transformations has been proposed. We have proved that the algorithm can be considered as a kind of nested subspace iteration, where the subspace is increased by one dimension at each step of the algorithm. As a consequence the largest eigenvalues, in absolute value, are already computed with high precision after few steps of the algorithm. Hence this algorithm can be used for solving problems where only the knowledge of the largest eigenvalues, in absolute value, and the corresponding eigenvectors, is needed [9, 15, 16, 19, 20, 1].

Once the semiseparable matrix has been computed, its eigenvalues can be computed either reapplying the proposed algorithm to the semiseparable matrix in an iterative fashion (in this case the sequence of the generated semiseparable matrices converges to a block diagonal matrix, where the eigenvalues of each block have the same absolute value), or applying fast and stable algorithms for computing the eigendecomposition of diagonal plus semiseparable matrices available in the literature [2, 3, 6, 13].

Future work will be concerned with an efficient implementation of the algorithm and a detailed study on an accurate criterion of splitting the semiseparable matrix generated by the algorithm into a block diagonal matrix, i.e., a criterion that allows to divide when a nondiagonal block of the matrix can be considered “negligible”. Also alternative representations of the semiseparable parts of the matrices will be investigated.

Furthermore, a similar technique will be considered to compute the singular value decomposition of a matrix.

REFERENCES

1. M. Braun, S.A. Sofianos, D.G. Papageorgiou and I.E. Lagaris, An efficient Chebychev–Lanczos method for obtaining eigensolutions of the Schrödinger equation on a grid, *J. Comput. Phys.*, 126 (1996), pp. 315–327.
2. S. Chandrasekaran and M. Gu. Fast and stable eigendecomposition of symmetric banded plus semi-separable matrices, *Linear Algebra and Its Applications*, 313:107–114, 2000.
3. S. Chandrasekaran and M. Gu. A divide and conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semi-separable matrices, <http://citeseer.nj.nec.com/241825.html>. submitted for publication, 2001.
4. Y. Eidelman and I. Gohberg. Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices. *Computers & Mathematics with Applications*, 33(4):69–79, August 1996.
5. D. Fasino and L. Gemignani. Direct and inverse eigenvalue problems, for diagonal-plus-semiseparable matrices, <http://www.dimi.uniud.it/fasino/papers.html2001>, submitted 2002.
6. D. Fasino, N. Mastronardi, and M. Van Barel. Fast and stable algorithms for reducing diagonal plus semiseparable matrices to tridiagonal and bidiagonal form. *Technical report: 3/2002 IAC Sez. Bari*, 2002, to appear in *Contemporary Mathematics*, AMS.
7. L. Gemignani and D. Fasino. Fast and stable solution of banded-plus-semiseparable linear systems. *Calcolo*, to appear.
8. G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins university Press, 1996, Third edition.

9. I.T. Joliffe *Principal Component Analysis*. Springer-Verlag, 1986.
10. N. Mastronardi, S. Chandrasekaran, and S. Van Huffel. Fast and stable algorithms for reducing diagonal plus semi-separable matrices to tridiagonal and bidiagonal form. *BIT* 41:1, 149–157, 2001.
11. T. Lloyd N. and B. David. *Numerical Linear Algebra*. Siam, 1997.
12. E. Van Camp, N. Mastronardi, and M. Van Barel. A fast QR solver for diagonal plus semiseparable linear systems. *Tech. Rep. IAC/CNR, Bari, 17/2002*, 2002.
13. E. Van Camp, N. Mastronardi, and M. Van Barel. Divide and Conquer–type algorithms for computing the eigendecomposition of diagonal plus semiseparable matrices. *Tech. Rep. IAC/CNR, Bari, 2/2003*, 2002.
14. G. W. Stewart *Matrix Algorithms, Vol II Eigensystems*. Siam, 1999.
15. M.E. Wall, A. Rechtsteiner, L.M. Rocha. Singular Value Decomposition and Principal Component Analysis. To appear in *A Practical Approach to Microarray Data Analysis*. D.P. Berrar, W. Dubitzky, M. Granzow, eds, kluwer.
16. M.E. Wall, P.A. Dyck and T.S. Brettin. SVDMAN - singular value decomposition analysis of microarray data. *Bioinformatics*, 17:6, 566–568, 2001.
17. D. S. Watkins. QR like algorithms: an overview of convergence theory and practice. *The Mathematics of Numerical Analysis, Lectures in Applied Math.*, 32:879–893, 1996.
18. D. S. Watkins and L. Elsner. Convergence of algorithms of decomposition type for the eigenvalue problem. *Linear Algebra and Its Applications*, 143:19–47, 1991.
19. M.K.S. Yeung, and W.L. Ruzzo, Principal Component Analysis for clustering gene expression data. *Bioinformatics*, 17:9, 763–774, 2001.
20. M.K.S. Yeung, J. Tegnér, and J.J. Collins, Reverse engineering gene networks using singular value decomposition and robust regression. *Proceedings National Academy of Sciences*, 99:6163–6168, 2002.