Using semiseparable matrices to compute the SVD of a general matrix product/quotient

Marc Van Barel [a,*], Yvette Vanberghen [a], Paul Van Dooren [b]

[a] Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven, Heverlee, Belgium
[b] Department of Mathematical Engineering, Catholic University of Louvain, Bâtiment Euler, Avenue Georges Lemaitre 4, B-1348 Louvain-la-Neuve, Belgium

ARTICLE INFO

ABSTRACT

In this work we reduce the computation of the singular values of a general product/quotient of matrices to the computation of the singular values of an upper triangular semiseparable matrix. Compared to the reduction into a bidiagonal matrix the reduction into semiseparable form exhibits a nested subspace iteration. Hence, when there are large gaps between the singular values, these gaps manifest themselves already during the reduction algorithm in contrast to the bidiagonal case.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

The Singular Value Decomposition (SVD) of matrices is a widely used tool. The Golub–Kahan algorithm [1], used to find the SVD of a matrix, first reduces the matrix to bidiagonal form using orthogonal transformations. Then the *QR* method is performed to reduce the bidiagonal matrix to a diagonal one. Vandebril et al. presented a similar method in [2] which uses an upper triangular semiseparable matrix as an intermediate matrix. Here an upper triangular matrix *S* is called semiseparable if all submatrices taken out of the upper triangular part have rank at most 1. An advantage of the approach using semiseparable matrices is that after a few steps of the reduction, large singular values and gaps in the spectrum are often revealed.

Consider a matrix *P* of the following form:

$$P = A_1^{s_1} A_2^{s_2} \cdots A_k^{s_k} \tag{1}$$

with $s_i \in \{1, -1\}$. There exist several algorithms for computing the SVD of *P* without explicitly computing the product. There are some algorithms which compute the SVD of a product of two or three matrices, such as those of Heath, Laub, Paige and Ward in [3] and Ewerbring, Luk, Bojanczyk and Van Dooren in [4]. These algorithms use a Jacobi-type method to compute the SVD because that easily extends to products.

Golub, Sølna and Van Dooren proposed an implicit *QR*-like method for the SVD of a general matrix product/quotient [5] with a complexity that was less than that of the Jacobi-type methods. As in the classical Golub–Kahan procedure for computing the singular value decomposition, this method uses a bidiagonal form of the product/quotient as an intermediate result. In general, when the number of matrices in the product increases, the gaps between the successive singular values increase as well. Therefore an approach with an upper triangular semiseparable matrix could be advantageous. This approach is the subject of this paper.

Section 2 describes the reduction of a product/quotient of matrices to semiseparable form. The accuracy and convergence properties of these algorithms are discussed in Section 3 by means of numerical experiments. Finally some conclusions are presented in Section 4.

---

* Corresponding author.
  E-mail addresses: marc.vanbarel@cs.kuleuven.be (M. Van Barel), yvette.vanberghen@cs.kuleuven.be (Y. Vanberghen), paul.vandooren@uclouvain.be (P. Van Dooren).

2. Reduction

In this section the reduction using orthogonal matrices of a sequence of matrices to an upper triangular semiseparable matrix is explained. But first we will introduce some notation. If a certain capital letter is used to denote a matrix (e.g. A) then the corresponding lower case letter with subscript ij refers to the (ij) entry of that matrix (e.g. a_{ij}). Submatrices are denoted in a Matlab-like fashion, for example $M(i{:}j, k{:}l)$ is the submatrix consisting of rows i to j and columns k to l of the matrix M. The group of Householder transformations that operate on a submatrix $M(i{:}j, i{:}j)$ is denoted by $\mathcal{H}(i, j)$, and the group of Givens transformations that operate on rows/columns $(i, i+1)$ is denoted by $\mathcal{G}(i, i+1)$. We use several matrix figures to explain the different steps of our algorithms. In these figures non-zero elements of a matrix are denoted by $\times$, elements that satisfy the semiseparable structure by $\boxtimes$ and elements that are be annihilated in the next transformation by $\otimes$. Although some elements are not computed explicitly, they are included in the figures when they clarify the process but are omitted from the figures otherwise. They are also marked in bold in order to distinguish them from the explicitly computed elements.

2.1. Reduction of a product

Consider the matrix P as defined in (1) with all $s_i = 1$. Without loss of generality we can assume all matrices A_i to be square $n \times n$ matrices since, for a given sequence of matrices with compatible dimensions, one can always derive another sequence of square matrices whose product has the same singular values as the product of the original sequence. The *QR*-like reduction used for this transformation is described in [6]. The first step of the reduction consists of creating a rank 1 block in $P(1{:}2, 2{:}n)$. To this end we annihilate all but one element in the first row of the product P, such that it is of the following form: $(\times 0 \cdots 0)$. In order to avoid computation of the whole product, Householder transformations are applied to the A_i separately as follows. The first Householder transformation $H_1^{(0)} \in \mathcal{H}(1, n)$ is applied to the right of A_1 and annihilates all but one element in its first row. In order to preserve the product, $H_1^{(0)T}$ is applied to the left of A_2. Next a second Householder transformation $H_2^{(0)} \in \mathcal{H}(1, n)$ is chosen to reduce the first row of A_2 to the same form. This transformation is performed on the columns of A_2 and on the rows of A_3. The first row of the remaining matrices A_i, $i = 3 \ldots k - 1$, are reduced in a similar manner until we have

$$P = A_1^{(0)} H_1^{(0)} H_1^{(0)T} A_2^{(0)} H_2^{(0)} H_2^{(0)T} A_3^{(0)} \cdots A_{k-1}^{(0)} H_{k-1}^{(0)} H_{k-1}^{(0)T} A_k^{(0)}$$
$$= A_1^{(1)} A_2^{(1)} A_3^{(1)} \cdots A_{k-1}^{(1)} A_k^{(1)}$$

where all $A_i^{(1)}$ with $i = 1 \ldots k - 1$ have the following form:

$$\begin{pmatrix} \mathbf{\times} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{\times} & \times & \cdots & \times \\ \vdots & \vdots & & \vdots \\ \mathbf{\times} & \times & \cdots & \times \end{pmatrix}.$$

It is clear that the Householder transformation $H_k^{(0)}$, designed to reduce the first row of $A_k^{(0)}$ to the desired form, also creates the necessary zeros in the first row of P.

At this point the first column of P is computed as the product of the first column of $A_k^{(1)}$ with the matrices to the left of it. All elements of this column are annihilated by the Householder transformation $H_P^{(0)} \in \mathcal{H}(2, n)$ except for p_{11} and p_{21}. $H_P^{(0)}$ is also applied to the left of the matrix $A_1^{(1)}$. The Givens transformation $G_1^{(0)} \in \mathcal{G}(1, 2)$ is responsible for the annihilation of p_{21}. In applying this Givens transformation to the first two rows of P, the first rank 1 block is virtually created, as pictured below.

$$G_1^{(0)} \begin{pmatrix} \times & 0 & \cdots & 0 \\ \otimes & \times & \cdots & \times \\ \mathbf{0} & \times & \cdots & \times \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \times & \cdots & \times \end{pmatrix} = \begin{pmatrix} \boxtimes & \boxtimes & \cdots & \boxtimes \\ \mathbf{0} & \boxtimes & \cdots & \boxtimes \\ \mathbf{0} & \times & \cdots & \times \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \times & \cdots & \times \end{pmatrix}.$$

Notice that since only the first column of P has been computed, $G_1^{(0)}$ is stored to be used later on. As a final part of this first step, the first row and column of each of the A_i are discarded. The next steps of the algorithm are similar to this first step; the only difference is that the structure already present in P has to be preserved. Also since some rows/columns of the A_i have been deleted, we need the stored Givens transformations to compute the next column of P. The process is illustrated below. For reasons of clarity a product of two (5×5) matrices is shown. Consider the following stage of the algorithm where two rank 1 blocks have already been created in P.

$$\begin{pmatrix} & & & & \\ & & & & \\ & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{pmatrix} \begin{pmatrix} & & & & \\ & & & & \\ & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{pmatrix} \Rightarrow \begin{pmatrix} \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \mathbf{0} & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ \mathbf{0} & \mathbf{0} & \boxtimes & \boxtimes & \boxtimes \\ \mathbf{0} & \mathbf{0} & \times & \times & \times \\ \mathbf{0} & \mathbf{0} & \times & \times & \times \end{pmatrix}.$$

Householder transformations $H_1^{(2)}$ and $H_2^{(2)} \in \mathcal{H}(3, 5)$ are computed in order to annihilate the last two elements in the third row of A_1 and A_2. They are applied in the following manner:

$$P = A_1 H_1^{(2)} H_1^{(2)T} A_2 H_2^{(2)}.$$

The resulting matrices are pictured below. Notice that because of the rank structure already present in P, the transformation $H_2^{(2)}$ annihilates $P(1:3, 4:5)$.

$$
\begin{pmatrix}
 & & & & \\
 & & & & \\
 & \times & 0 & 0 \\
 & \times & \times & \times \\
 & \times & \times & \times
\end{pmatrix}
\begin{pmatrix}
 & & & & \\
 & & & & \\
 & \times & 0 & 0 \\
 & \times & \times & \times \\
 & \times & \times & \times
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
\boxtimes & \boxtimes & \boxtimes & 0 & 0 \\
0 & \boxtimes & \boxtimes & 0 & 0 \\
0 & 0 & \boxtimes & 0 & 0 \\
0 & 0 & \times & \times & \times \\
0 & 0 & \times & \times & \times
\end{pmatrix}.
$$

The unstructured part of the third column of P is calculated next as the product of $A_1(3:5, 3:5)$ and $A_2(3:5, 3)$. Using the stored Givens transformations $G_3^{(2)} \in \mathcal{G}(1, 2)$ and $G_1^{(2)} \in \mathcal{G}(2, 3)$ of the previous step, the remaining part of the third column of P can be retrieved, and this property will be needed later on.

Next a Householder transformation $H_1^{(3)} \in \mathcal{G}(4, 5)$ is applied to $P(4:5, 1:3)$ and $A_1(4:5, 3:5)$ which annihilates $P(5, 3)$. Note that since only one element needs to be annihilated, one could perform a Givens transformation instead. A Givens transformation $G_1^{(3)} \in \mathcal{G}(3, 4)$ is responsible for the annihilation of $P(4, 3)$ and the creation of a rank 1 block. Just like in previous steps, this transformation is stored for later use.

$$
G_1^{(3)}
\begin{pmatrix}
\boxtimes & \boxtimes & \boxtimes & 0 & 0 \\
0 & \boxtimes & \boxtimes & 0 & 0 \\
0 & 0 & \boxtimes & 0 & 0 \\
0 & 0 & \otimes & \times & \times \\
0 & 0 & 0 & \times & \times
\end{pmatrix}
=
\begin{pmatrix}
\boxtimes & \boxtimes & \boxtimes & 0 & 0 \\
0 & \boxtimes & \boxtimes & 0 & 0 \\
0 & 0 & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & 0 & \boxtimes & \boxtimes \\
0 & 0 & 0 & \times & \times
\end{pmatrix}.
$$

Notice that when applying this transformation to P the semiseparable structure is lost, but only in the block $P(1:3, 3:5)$, which has rank 2 instead of 1. The semiseparable structure can be restored by chasing this rank 2 block out of the matrix, as shown below.

$$
\begin{pmatrix}
\boxtimes & \boxtimes & \otimes & 0 & 0 \\
0 & \boxtimes & \otimes & 0 & 0 \\
0 & 0 & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & 0 & \boxtimes & \boxtimes \\
0 & 0 & 0 & \times & \times
\end{pmatrix}
G_2^{(3)}
=
\begin{pmatrix}
\boxtimes & \boxtimes & 0 & 0 & 0 \\
0 & \boxtimes & 0 & 0 & 0 \\
0 & \times & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & 0 & \boxtimes & \boxtimes \\
0 & 0 & 0 & \times & \times
\end{pmatrix}.
$$

Because of the structure present in P, $G_2^{(3)} \in \mathcal{G}(2, 3)$ annihilates all elements in the strictly upper triangular part of the third column of P. When applying this transformation a disturbance is introduced in the lower triangular part. This disturbance is annihilated by a Givens transformation $G_3^{(3)} \in \mathcal{G}(2, 3)$, and now $P(1:2, 2:5)$ is the block of rank 2 instead of 1. Note that $G_3^{(3)}$ is again stored for later use.

$$
G_3^{(3)}
\begin{pmatrix}
\boxtimes & \boxtimes & 0 & 0 & 0 \\
0 & \boxtimes & 0 & 0 & 0 \\
0 & \otimes & \times & \boxtimes & \boxtimes \\
0 & 0 & 0 & \boxtimes & \boxtimes \\
0 & 0 & 0 & \times & \times
\end{pmatrix}
=
\begin{pmatrix}
\boxtimes & \boxtimes & 0 & 0 & 0 \\
0 & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & 0 & \boxtimes & \boxtimes \\
0 & 0 & 0 & \times & \times
\end{pmatrix}.
$$

Two extra Givens transformations $G_4^{(3)}$ and $G_5^{(3)} \in \mathcal{G}(1, 2)$ are needed to fully restore the semiseparable structure of P as illustrated below.

$$
\begin{pmatrix}
\boxtimes & \otimes & 0 & 0 & 0 \\
0 & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & 0 & \boxtimes & \boxtimes \\
0 & 0 & 0 & \times & \times
\end{pmatrix}
G_4^{(3)}
=
\begin{pmatrix}
\boxtimes & 0 & 0 & 0 & 0 \\
\times & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & 0 & \boxtimes & \boxtimes \\
0 & 0 & 0 & \times & \times
\end{pmatrix},
$$

$$
G_5^{(3)}
\begin{pmatrix}
\boxtimes & 0 & 0 & 0 & 0 \\
\otimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & 0 & \boxtimes & \boxtimes \\
0 & 0 & 0 & \times & \times
\end{pmatrix}
=
\begin{pmatrix}
\boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\
0 & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & \boxtimes & \boxtimes & \boxtimes \\
0 & 0 & 0 & \boxtimes & \boxtimes \\
0 & 0 & 0 & \times & \times
\end{pmatrix}.
$$

At the end of this step an extra row and column are deleted from the A_i matrices. The other rank 1 blocks can be created in a similar manner.
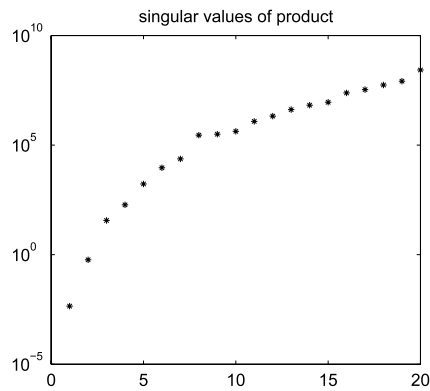
Fig. 1. This figure illustrates the singular values of the product of 16 random matrices.

2.2. Reduction of a quotient

In case in (1) some of the s_i are equal to -1, additional operations have to be performed at the beginning of the reduction. More precisely, the corresponding matrices A_i are triangularized to facilitate the computation of the columns of P. Consider the following example:

$$P = A_1 A_2^{-1} A_3$$

in which the matrix A_2 has to be triangularized. This can be done by a QR factorization. Let $Q_2^{(0)}$ be the sequence of Householder transformations that triangularizes A_2. Then P can be written as

$$P = A_1 (Q_2^{(0)} A_2)^{-1} (Q_2^{(0)} A_3).$$

After this preliminary phase the above procedure can again be followed, with the only difference that Givens transformations are applied to each upper triangular A_i instead of Householder transformations in order to preserve the upper triangular form of the matrix, since each disturbance created by a Givens transformation can easily be annihilated by another Givens transformation applied to the other side of A_i. For a more detailed description and an illustrating example we refer to [5].

3. Numerical experiments

In this section we will show through numerical examples that the method proposed above is accurate and has a special convergence behaviour. All experiments described in this section were executed in Matlab[1] on a Linux workstation.

3.1. Accuracy

In order to illustrate the accuracy of the method, the product of 16 (20×20) normally distributed random matrices was computed explicitly along with its singular values using extended precision in Matlab. These singular values were considered to be the exact singular values of the product and are pictured in Fig. 1. We reduced the product to upper triangular semiseparable form using the algorithm described above. Then the singular values of the semiseparable matrix can be computed by the algorithm described in [2]. The singular values were also computed using the bidiagonal approach of Golub, Sølna and Van Dooren. Both sets of singular values were compared to the exact ones. This is illustrated in Fig. 2 on the left. The relative errors resulting from the singular values computed using our approach are denoted by '∗' and those computed by the bidiagonal approach are denoted by '∘'. One can see that both methods give comparable results and that all the singular values are computed to high relative accuracy, despite the fact that they differ by several orders of magnitude. The figure on the right also contains relative errors of both methods, but this time the problem was a quotient of matrices. The relative errors are again comparable in size.

3.2. Convergence

The added convergence behaviour, which was mentioned in the introduction, will be illustrated here. In the first experiments of this section we use a matrix product P that can be constructed as follows. The singular value decomposition of a randomly generated $n \times n$ matrix is computed, resulting in two unitary matrices U and V. Let S be a diagonal matrix, with diagonal elements $(1.2)^i$ where i ranges from $1 \ldots n$. Define two matrices A and B as

$$A = USV^*$$
$$B = VSU^*.$$

---

[1] Matlab is a registered trademark of the Mathworks Inc.
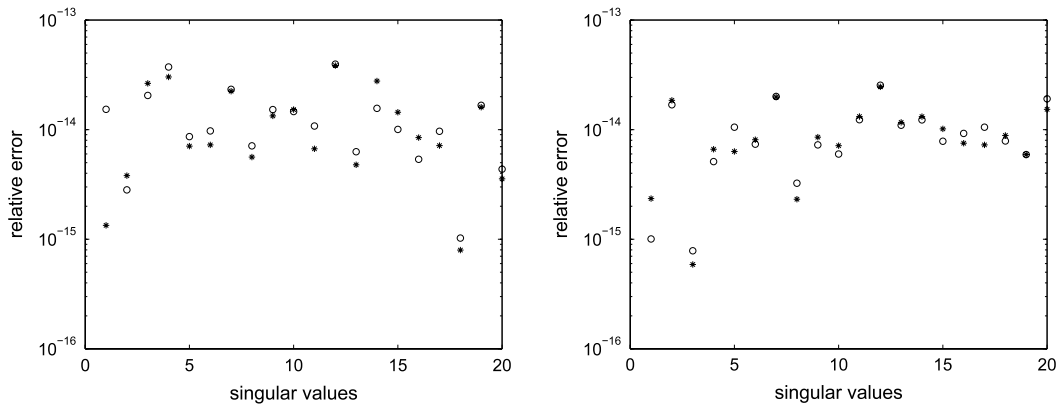
Fig. 2. The figure on the left contains relative errors made in the singular values of a product by the bidiagonal approach '○' and by the semiseparable approach '∗'. Relative errors made by both algorithms in the singular values of a quotient of matrices are pictured on the right.
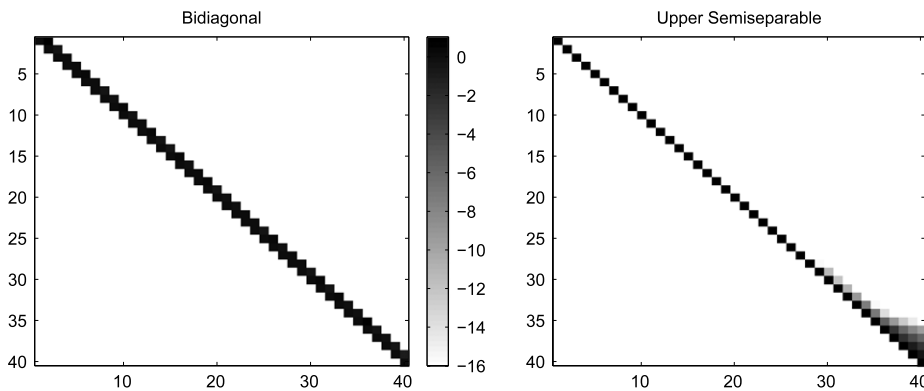

Fig. 3. The elements of the reduced product in bidiagonal form are illustrated in the figure on the left. The semiseparable form of the product is illustrated on the right.

Then the product $P = (AB)^k$ has $(1.2)^{2ki}$ as singular values. This means that there are large gaps between the singular values of the product.

For the first experiment we took $k = 4$ and $n = 40$. The product was reduced to upper triangular semiseparable form using the method described above. The algorithm described in [5] was used to reduce the same product to bidiagonal form as well. When visualizing the two reduced matrices, we divided each row by their diagonal element because the elements of the reduced matrices have a large range. These scaled matrices are pictured in Fig. 3. As one can see, at the end of the reduction a lot of singular values have already been found in the semiseparable case. When P is reduced to a bidiagonal matrix this is not the case. However, the reduction to semiseparable form is more expensive than the reduction to a bidiagonal matrix. So one could easily extract the singular values from the bidiagonal matrix without performing more operations in total. The advantage of the semiseparable approach lies in the fact that the convergence happens during the reduction and if one is only interested in a few dominant singular values, the reduction can be terminated prematurely.

In order to illustrate this, we considered the case where $n = 50$ and $k = 5, 10, 20$. The same procedure as described above was followed but in this case the reduction to semiseparable form was stopped as soon as the first dominant singular value was found. In order to be able to time the execution we repeated the procedure fifty times. Then the experiment was repeated to find the first 2, 3 . . . 20 dominant singular values. For each value of k the bidiagonal approach was timed as well. The results of this experiment are pictured in Fig. 4. The dotted lines represent the timings for $k = 5$, the dashed ones for $k = 10$ and the remaining lines are for $k = 20$. Looking at the curves we see that the difference in performance is more noticeable when k is larger. This was to be expected since, as the number of matrices in the product increases, so do the gaps between the singular values. The slopes of the lines indicate that the new approach is not recommended if one needs all the singular values.

As a final example consider a product P similar to the one described at the beginning of this section with $n = 20$ and $k = 8$, the only differences being that now the diagonal elements of S equal $(1, 2, 3 . . . 19, 25)$. The gap between 19 and 25 will cause a significant gap between the largest and second-largest singular value of P. In Fig. 5 in each step of the reduction, the first diagonal element of the matrix P is compared to the largest singular value. One can see that after four steps this singular value has already appeared on the diagonal. If one is only interested in this dominant singular value the reduction can already be stopped at this point. The same has been done for the other singular values. All but the last four needed ten
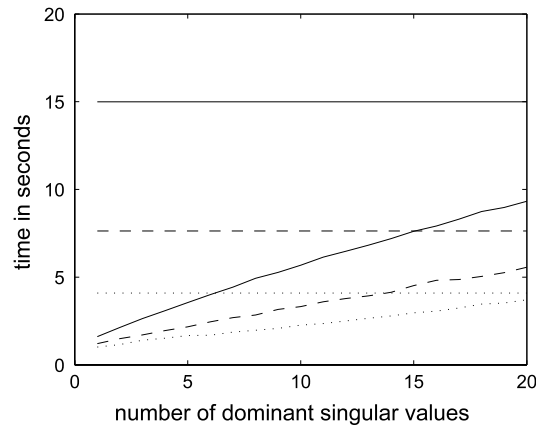
Fig. 4. This figure illustrates the time required by both the bidiagonal and the semiseparable approach to find the first 1 to 20 dominant singular values. The horizontal lines represent the timings for the bidiagonal approach.
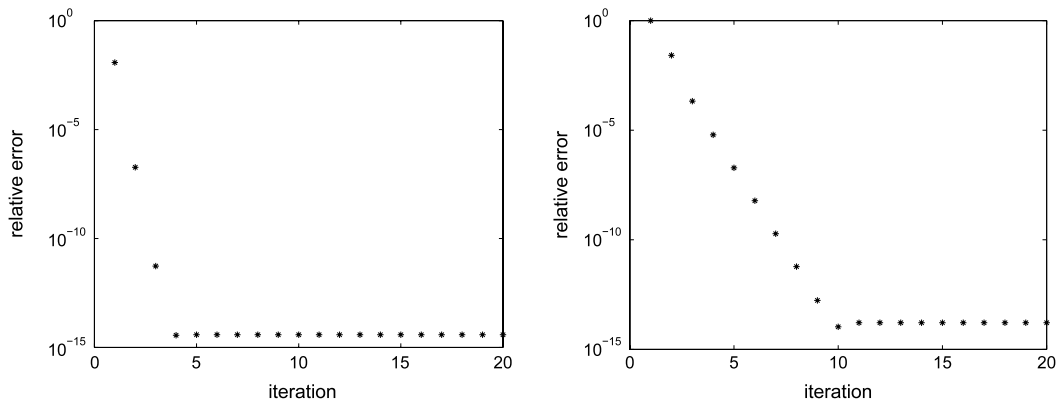

Fig. 5. The accuracy of the estimate of the dominant singular value is pictured on the left. On the right the relative error of the estimate of the second-largest eigenvalue is illustrated.

to twelve iterations to be found on the diagonal. This is pictured in Fig. 5 for the second-largest singular value. When the singular values are clustered with large gaps between the clusters, one can therefore expect the clusters to be separated during the reduction process. One can then use a divide and conquer approach to compute all individual singular values.

4. Conclusion

In this work we proposed a new method for computing the singular values of a product/quotient of a sequence of matrices. The algorithm described in this work reduces the product/quotient of a given sequence to an upper triangular semiseparable matrix. Then the singular values can be computed using the algorithm described in [2]. Important in this reduction is the observed nested subspace iteration, which leads to extra convergence behaviour.

References

[1] G.H. Golub, W. Kahan, Calculating the singular values and pseudo-inverse of a matrix, SIAM Journal on Numerical Analysis 2 (1965) 205–224.
[2] R. Vandebril, M. Van Barel, N. Mastronardi, A *QR*-method for computing the singular values via semiseparable matrices, Numerische Mathematik 99 (2004) 163–195.
[3] M.T. Heath, A.J. Laub, C.C. Paige, R.C. Ward, Computing the singular value decomposition of a product of two matrices, SIAM Journal on Scientific and Statistical Computation 7 (4) (1986) 1147–1159.
[4] A.W. Bojanczyk, M. Ewerbring, F.T. Luk, P. Van Dooren, An accurate product SVD algorithm, Signal Processing 25 (2) (1991) 189–201.
[5] G.H. Golub, K. Sølna, P. Van Dooren, Computing the SVD of a general matrix product/quotient, SIAM Journal on Matrix Analysis and its Applications 22 (1) (2000) 1–19.
[6] B.L.R. De Moor, P. Van Dooren, Generalizations of the singular value and QR decompositions, SIAM Journal on Matrix Analysis and its Applications 13 (4) (1992) 993–1014.