

A fast algorithm for the recursive calculation of dominant singular subspaces

Nicola Mastronardi Marc Van Barel
Raf Vandebril

Report TW 468, September 2006



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

A fast algorithm for the recursive calculation of dominant singular subspaces

*Nicola Mastronardi Marc Van Barel
Raf Vandebril*

Report TW 468, September 2006

Department of Computer Science, K.U.Leuven

Abstract

In many engineering applications it is required to compute the dominant subspace of a matrix A of dimension $m \times n$, with $m \gg n$. Often the matrix A is produced incrementally, so all the columns are not available simultaneously. This problem arises, e.g., in image processing, where each column of the matrix A represents an image of a given sequence leading to a singular value decomposition based compression [1]. Furthermore, the so called *proper orthogonal decomposition* approximation uses the left dominant subspace of a matrix A where a column consists of a time instance of the solution of an evolution equation, e.g., the flow field from a fluid dynamics simulation. Since these flow fields tend to be very large, only a small number can be stored efficiently during the simulation, and therefore an incremental approach is useful [7].

In this paper an algorithm for computing an approximation of the left dominant subspace of size k of $A \in \mathbb{R}^{m \times n}$, with $k \ll m, n$, is proposed requiring at each iteration $O(mk + k^2)$ floating point operations. Moreover, the proposed algorithm exhibits a lot of parallelism that can be exploited for a suitable implementation on a parallel computer.

Keywords : unitary matrix, rank structured matrix, Givens transformation, pull-through operation, shift correction term.

AMS(MOS) Classification : Primary : 15A18, Secondary : 65Y20.

A fast algorithm for the recursive calculation of dominant singular subspaces

N. Mastronardi ^{a,1}, M. Van Barel ^{b,*,2}, R. Vandebril ^{b,2}

^a*Istituto per le Applicazioni del Calcolo, CNR, via Amendola 122/D, 70126, Bari, Italy*

^b*Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium*

Abstract

In many engineering applications it is required to compute the dominant subspace of a matrix A of dimension $m \times n$, with $m \gg n$. Often the matrix A is produced incrementally, so all the columns are not available simultaneously. This problem arises, e.g., in image processing, where each column of the matrix A represents an image of a given sequence leading to a singular value decomposition based compression [1]. Furthermore, the so called *proper orthogonal decomposition* approximation uses the left dominant subspace of a matrix A where a column consists of a time instance of the solution of an evolution equation, e.g., the flow field from a fluid dynamics simulation. Since these flow fields tend to be very large, only a small number can be stored efficiently during the simulation, and therefore an incremental approach is useful [7].

In this paper an algorithm for computing an approximation of the left dominant subspace of size k of $A \in \mathbb{R}^{m \times n}$, with $k \ll m, n$, is proposed requiring at each iteration $O(mk + k^2)$ floating point operations. Moreover, the proposed algorithm exhibits a lot of parallelism that can be exploited for a suitable implementation on a parallel computer.

Key words: Householder matrix, Givens rotation, URV factorization, updating, singular value decomposition

* *Email address corresponding author:* Marc.VanBarel@cs.kuleuven.be

¹ This work was partially supported by MIUR, grant number 2004015437 and by the National Research Council of Italy under the Short Term Mobility Program.

² The research was partially supported by the Research Council K.U.Leuven, project OT/05/40 (Large rank structured matrix computations), Center of Excellence: Optimization in Engineering, by the Fund for Scientific Research–Flanders (Belgium), G.0455.0 (RHPH: Riemann-Hilbert problems, random matrices and

1 Introduction

In many engineering applications it is required to compute the dominant subspace of a matrix A of dimension $m \times n$, with $m \gg n$. Often the matrix A is produced incrementally, so all the columns are not available simultaneously. This problem arises, e.g., in image processing, where each column of the matrix A represents an image of a given sequence leading to a singular value decomposition based compression [1]. Furthermore, the so called *proper orthogonal decomposition* approximation uses the left dominant subspace of a matrix A where a column consists of a time instance of the solution of an evolution equation, e.g., the flow field from a fluid dynamics simulation. Since these flow fields tend to be very large, only a small number can be stored efficiently during the simulation, and therefore an incremental approach is useful [7].

In [2] a recursive procedure has been designed for computing an approximation of the left dominant singular subspace of a matrix, whose columns are produced incrementally, with computational complexity $O(mk + k^3)$ for each step of the recursion.

In this paper an alternative algorithm is proposed with computational complexity $O(mk + k^2)$ for each step of the recursion. Moreover, the proposed algorithm exhibits a lot of parallelism that can be exploited for a suitable implementation on a parallel computer.

The paper is organized as follows. In section 2 the algorithm proposed in [2] is summarized. The new algorithm is described in section 3 followed by the numerical experiments in section 4 and the conclusions.

2 Recursive Calculation of Dominant Singular Subspaces

In this section we shortly describe the algorithm proposed in [2] to recursively compute dominant singular subspaces, whose columns are known incrementally, i.e., at each step one more column is added to the matrix. The algorithm is particularly suited for matrices $A \in \mathbb{R}^{m \times n}$, $m \gg n$.

Padé-Hermite approximation), G.0423.05 (RAM: Rational modelling: optimal conditioning and stable algorithms), 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 third author has a grant as "Postdoctoraal Onderzoeker" from the Fund for Scientific Research-Flanders (Belgium). The scientific responsibility rests with the authors.

Suppose a URV decomposition of the matrix $\tilde{A} \in \mathbb{R}^{m \times k}$, $k < n \ll m$, is computed, $\tilde{A} = URV$, with $U \in \mathbb{R}^{m \times k}$, $V \in \mathbb{R}^{k \times k}$ orthogonal matrices.

At each recursion, the following steps are performed:

Step A.1. A new column b is added to \tilde{A} ,

$$\hat{A} = [\tilde{A}, b].$$

Step A.2. Update the URV factorization of the extended matrix \hat{A} , e.g., via Gram-Schmidt (or modified Gram-Schmidt) orthogonalization,

$$\begin{aligned} r &= U^T b \\ \hat{b} &= b - Ur \\ \rho &= \|\hat{b}\|_2 \\ \hat{u} &= \hat{b}/\rho. \end{aligned}$$

Then

$$[\tilde{A}|b] = [UR|b] \left[\begin{array}{c|c} V & \\ \hline & 1 \end{array} \right] = [U|\hat{u}] \left[\begin{array}{c|c} R & r \\ \hline & \rho \end{array} \right] \left[\begin{array}{c|c} V & \\ \hline & 1 \end{array} \right] = [U|\hat{u}]\hat{R} \left[\begin{array}{c|c} V & \\ \hline & 1 \end{array} \right].$$

Step A.3. Compute $\hat{\sigma}_{k+1}$, the smallest singular value of \hat{R} and \hat{u}_{k+1} , the corresponding left singular vector. Let G_u be an orthogonal matrix such that

$$G_u^T \hat{u}_{k+1} = e_{k+1} = [0, \dots, 0, 1]^T.$$

and compute

$$G_u^T \hat{R} = \hat{S},$$

with $\hat{S} \in \mathbb{R}^{(k+1) \times (k+1)}$ no more upper triangular.

Step A.4. Compute the RQ factorization of \hat{S} ,

$$\hat{S} = \tilde{R}\tilde{H},$$

where \tilde{R} is an upper triangular and \tilde{H} an orthogonal matrix of order $k+1$. It turns out that

$$\tilde{R} = \left[\begin{array}{c|c} R_k & \\ \hline & \hat{\sigma}_{k+1} \end{array} \right].$$

Step A.5. Update the orthogonal matrix U

$$U \leftarrow UG_u, \quad V \leftarrow \tilde{H}^T V,$$

and set

$$\tilde{A} = U(:, 1 : k)R_kV(1 : k, :).$$

Remark 1 *We observe that it is not necessary to update the matrix V , as stated in [2], if only the left dominant singular subspaces are needed. In fact, in the orthogonalization process in step A.2, only the matrix U is involved.*

In Step A.3, the computation of $\hat{\sigma}_{k+1}$, the smallest singular value of \hat{R} and \hat{u}_{k+1} , the corresponding left singular vector, requires $O(k^3)$ floating point operations [2]. Furthermore, the matrix G_u in Step A.3 could be either a product of k Givens rotations or a Householder matrix. If the product of k Givens rotations is chosen, Step A.4 can be accomplished in $O(k^2)$ operations in a way similar to the one described in [5], and the matrix \tilde{H} is given by the product of k different Givens rotations. In [5] it is shown that Step A.4 can be accomplished with $O(k^2)$ floating point operations even in the case the reduction is computed by using a Householder transformation.

However, the costly part of the algorithm is the update of the matrix U , and hence it is preferable to choose G_u to be a Householder transformation. In fact, the cost of the latter product UG_u is $4mk$ if G_u is a Householder transformation, whereas the cost is $6mk$ if the orthogonal matrix G_u is given by the product of k Givens rotations.

In the next section we propose a different algorithm for tracking the dominant singular subspace.

3 Bidiagonal updating estimation of the dominant singular subspace

The important information in the algorithm described in the latter section is given by the computed orthogonal signal subspace U . In this section we propose a new algorithm for computing an orthogonal basis approximating the original dominant subspace.

The role of the upper triangular matrix R in the latter algorithm is played by an upper bidiagonal matrix B in the new algorithm.

The initialization can be either accomplished by a reduction of the matrix \tilde{A} into an upper bidiagonal one by Householder transformations [4],

$$\tilde{A} = \tilde{U}\tilde{B}\tilde{V}, \tag{1}$$

with $\tilde{U} \in \mathbb{R}^{n \times k}$ orthogonal and

$$\tilde{B} = \begin{bmatrix} \gamma_1 & \beta_1 & & & & \\ & \gamma_2 & \beta_2 & & & \\ & & \ddots & \ddots & & \\ & & & \gamma_{k-1} & \beta_{k-1} & \\ & & & & \gamma_k & \end{bmatrix},$$

or simply computing the singular value decomposition of the matrix \tilde{A} . In this paper we will use the reduction of \tilde{A} into a bidiagonal matrix as initialization. The problem now is to compute the new signal subspace of the augmented matrix $[\tilde{A} \mid b]$, where $b \in \mathbb{R}^m$ is the new vector.

The proposed recursive procedure is made by the following steps.

Step B.1. Decompose the new processed vector b into its projection onto \tilde{U} and on its projection onto the orthogonal complement \tilde{U}^\perp ,

$$b = \tilde{U}\tilde{\alpha} + \tilde{\alpha}_{k+1}\tilde{u}_{k+1},$$

with

$$\tilde{\alpha} = [\tilde{\alpha}_1, \tilde{\alpha}_2, \dots, \tilde{\alpha}_k]^T = \tilde{U}^T b, \quad \tilde{\alpha}_{k+1} = \|(I_m - \tilde{U}\tilde{U}^T)b\|_2,$$

$$\tilde{u}_{k+1} = \frac{(I_m - \tilde{U}\tilde{U}^T)b}{\tilde{\alpha}_{k+1}}, \quad \tilde{U}^T \tilde{u}_{k+1} = 0,$$

where I_m is the identity matrix of order m . The problem is transformed into the following one,

$$\begin{aligned} [\tilde{A}|b] &= [\tilde{U}\tilde{B}\tilde{V} \mid b] \\ &= [\tilde{U} \mid \tilde{u}_{k+1}] \left[\begin{array}{c|c} \tilde{B} & \tilde{\alpha} \\ \hline & \tilde{\alpha}_{k+1} \end{array} \right] \left[\begin{array}{c|c} \tilde{V} & \\ \hline & 1 \end{array} \right] \\ &= [\tilde{U} \mid \tilde{u}_{k+1}] B_c \left[\begin{array}{c|c} \tilde{V} & \\ \hline & 1 \end{array} \right], \end{aligned}$$

with B_c an upper bidiagonal matrix with a column appended. This step is similar to the one of the algorithm described in section 2, requiring the same number of floating point operations, i.e., $2km$ floating point operations.

Step B.2. Reduce the matrix B_c into an upper tridiagonal one T by means of orthogonal transformations,

$$T = \tilde{Q}_1^T B_c \hat{Q}_1^T. \quad (2)$$

This reduction can be accomplished by an algorithm similar to those used for reducing a bordered matrix into upper bidiagonal form [6,8]. The reduction is accomplished by using Givens rotations. The transformation of a bidiagonal matrix of order 9, with a column appended, is depicted in Fig. 1. We observe that some Givens rotations can be per-

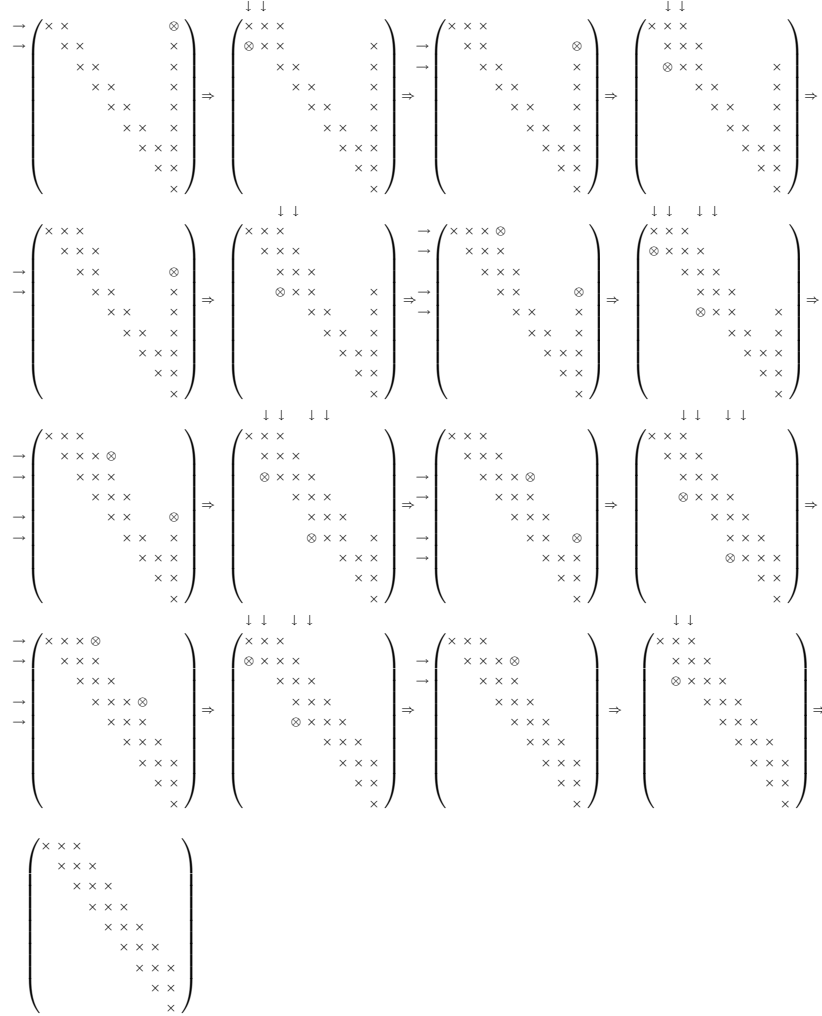


Fig. 1. Description of the reduction of a bidiagonal matrix with a column appended to an upper tridiagonal one by using Givens rotations. The symbol \otimes denotes the entries to be annihilated by applying the Givens rotation. The arrows show the rows (columns) to be modified by the Givens rotations.

formed independently of each other. This can be exploited for an efficient implementation of the algorithm on a parallel computer. To annihilate the entries of the appended column and to chase the bulges during the

construction of the upper tridiagonal matrix, $k/2(k/2 - 1)$ Givens rotations are used. Hence, the computational complexity of this step³ is $5k^2$. We observe that the orthogonal matrices \tilde{Q}_1 and \hat{Q}_1 in (2), given by the product of the involved Givens rotations, are not explicitly computed. The Givens coefficients are stored, requiring $O(k^2)$ memory and used in Step B.4.

Step B.3. Reduce the upper tridiagonal matrix T into an upper bidiagonal one B_1 by orthogonal transformations,

$$B_1 = \tilde{Q}_2^T T \hat{Q}_2^T. \quad (3)$$

The reduction of an upper tridiagonal matrix into an upper bidiagonal one can be accomplished by using standard bulge–chasing techniques [4]. Also this step is accomplished by using $k/2(k/2 - 1)$ Givens rotations with $5k^2$ computational complexity. Again, also in this case the orthogonal matrices \tilde{Q}_2 and \hat{Q}_2 in (3) are not explicitly computed. The Givens coefficients are stored, requiring $O(k^2)$ memory and used in Step B.4. This reduction is depicted in Fig. 2 for a matrix of order 9. Also in this step, we observe that some Givens rotations can be performed independently of each other. This can be exploited for an efficient implementation of the algorithm on a parallel computer.

Step B.4. Update the dominant subspace.

From (2) and (3),

$$B_c = \tilde{Q}_1^T T \hat{Q}_1 = \tilde{Q}_1 \tilde{Q}_2 B_1 \hat{Q}_2 \hat{Q}_1.$$

Then

$$[\tilde{A} \mid b] = [\tilde{U} \mid \tilde{u}_{k+1}] \tilde{Q}_1 \tilde{Q}_2 B_1 \hat{Q}_2 \hat{Q}_1 \left[\begin{array}{c|c} \tilde{V} & \\ \hline & 1 \end{array} \right].$$

Let $B_1 = \hat{U} \hat{\Sigma} \hat{V}^T$ be the singular value decomposition of B_1 , with $\hat{\Sigma} = \text{diag}(\hat{\sigma}_1, \hat{\sigma}_2, \dots, \hat{\sigma}_{k+1})$, $\hat{\sigma}_1 \geq \hat{\sigma}_2 \geq \dots \geq \hat{\sigma}_{k+1} \geq 0$. The smallest singular value $\hat{\sigma}_{k+1}$ and the corresponding left singular vector can be efficiently computed with $O(k)$ floating point operations [5,9,10]. In the proposed paper we have used the algorithm described in [5] to compute the left singular vector corresponding to $\hat{\sigma}_{k+1}$. The latter algorithm is based on applying very few steps, let us say j steps, (from a theoretical point of view only one step is needed) of the QR method [4] with shift $\hat{\sigma}_{k+1}$ to the matrix \hat{B} . It turns out

$$B_1 = \tilde{Q}_3 \left[\begin{array}{c|c} B_2 & \\ \hline & \hat{\sigma}_{k+1} \end{array} \right] \hat{Q}_3, \quad (4)$$

³ The computation of the Givens coefficients requires 6 floating point operations, as described in [4].

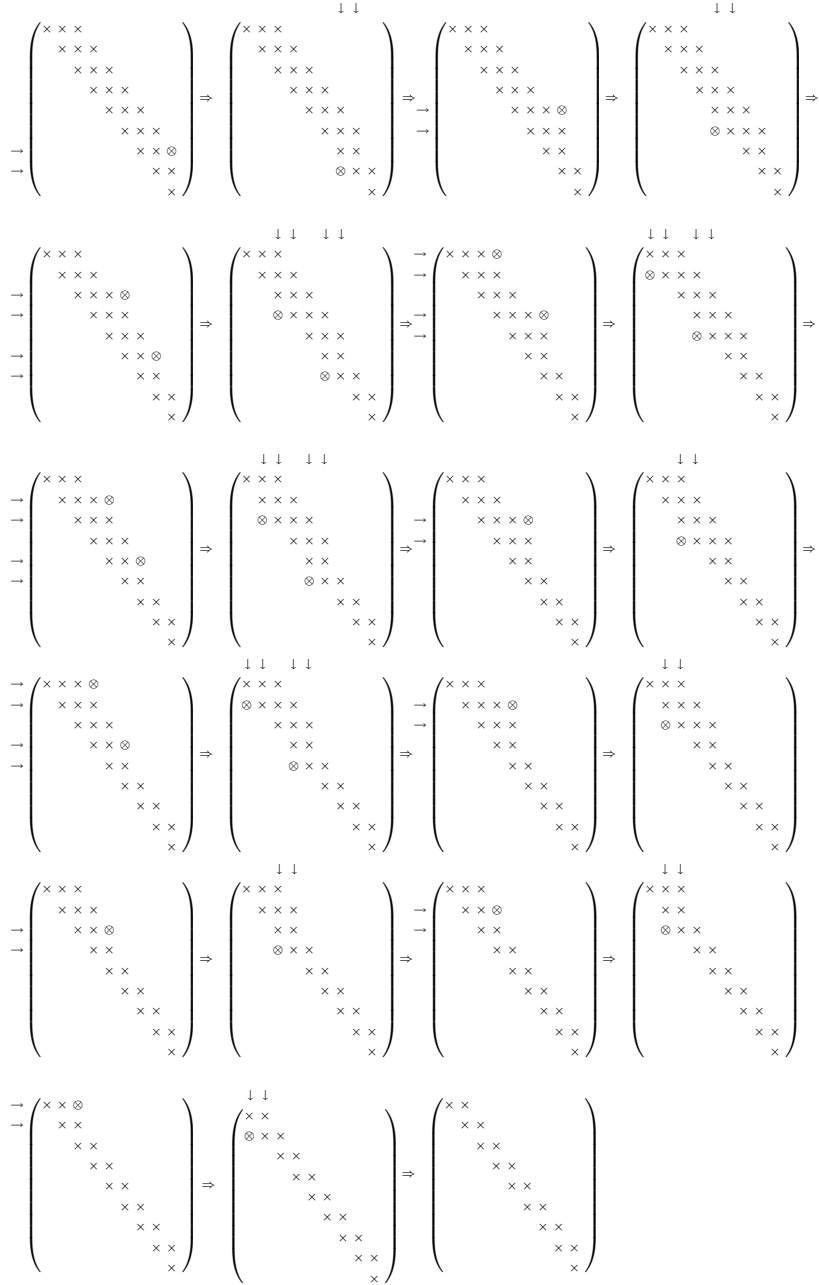


Fig. 2. Description of the reduction of a tridiagonal matrix to an upper bidiagonal one by using Givens rotations. The symbol \otimes denotes the entries to be annihilated by applying the Givens rotation. The arrows show the rows (columns) to be modified by the Givens rotations.

where B_2 is a bidiagonal matrix of order k and \tilde{Q}_3 and \hat{Q}_3 are orthogonal matrices of order $k + 1$. Let us define

$$B_3 \equiv \left[\begin{array}{c|c} B_2 & \\ \hline & \hat{\sigma}_{k+1} \end{array} \right]. \quad (5)$$

We observe that $\tilde{q}_{k+1}^{(3)}$, the last column of the matrix \tilde{Q}_3 , is the singular vector of B_1 corresponding to $\hat{\sigma}_{k+1}$, i.e.,

$$\tilde{q}_{k+1}^{(3)} = \tilde{Q}_3 e_{k+1}.$$

Moreover,

$$z_{k+1} \equiv \tilde{Q}_1 \tilde{Q}_2 \tilde{Q}_3 e_{k+1} = \tilde{Q}_1 \tilde{Q}_2 \tilde{q}_{k+1}^{(3)}$$

is the left singular vector of B_c corresponding to $\tilde{\sigma}_{k+1}$. We observe that the matrix \tilde{Q}_3 is given by the product of j upper Hessenberg orthogonal matrices, each one generated by one step of the QR -method, i.e., each one is the product of k Givens rotations. Therefore, having stored the involved Givens coefficients, the singular vector $q_{k+1}^{(3)}$ can be computed with $O(jk)$ floating point operations. Furthermore, the product $\tilde{Q}_1 \tilde{Q}_2 \tilde{q}_{k+1}^{(3)}$ can be accomplished with $3/2k^2$ floating point operations.

Let \tilde{H} be the Householder matrix such that

$$\tilde{H} w_{k+1} = \mp e_{k+1}, \quad w_{k+1} = \tilde{Q}_1 \tilde{Q}_2 \tilde{Q}_3 e_{k+1}. \quad (6)$$

Therefore

$$\tilde{H} \tilde{Q}_1 \tilde{Q}_2 \tilde{Q}_3 = \left[\begin{array}{c|c} \tilde{W} & \\ \hline & \mp 1 \end{array} \right],$$

with $\tilde{W} \in \mathbb{R}^{k \times k}$ orthogonal.

Hence, taking (4), (5) and (6) into account, it turns out

$$\begin{aligned} [\tilde{A} \mid b] &= [\tilde{U} \mid \tilde{u}_{r+1}] \tilde{H}^T \tilde{H} \tilde{Q}_1 \tilde{Q}_2 \tilde{Q}_3 \left[\begin{array}{c|c} B_2 & \\ \hline & \hat{\sigma}_{k+1} \end{array} \right] \hat{Q}_3 \hat{Q}_2 \hat{Q}_1 \left[\begin{array}{c|c} \tilde{V} & \\ \hline & 1 \end{array} \right] \\ &= [\tilde{U} \mid \tilde{u}_{r+1}] \tilde{H}^T \left[\begin{array}{c|c} \tilde{W} & \\ \hline & \mp 1 \end{array} \right] \left[\begin{array}{c|c} B_2 & \\ \hline & \hat{\sigma}_{k+1} \end{array} \right] \hat{Q}_3 \hat{Q}_2 \hat{Q}_1 \left[\begin{array}{c|c} \tilde{V} & \\ \hline & 1 \end{array} \right]. \end{aligned}$$

Let $\hat{U} \equiv [\tilde{U} \mid \tilde{u}_{k+1}] H^T$. Due to the special block bidiagonal structure of B_3 , the signal subspace associated to the largest k singular values of \tilde{B} is given by the first k columns of \hat{U} . Known the matrix \tilde{H} , the latter orthogonal matrix is computed with $4nk$ floating point operations.

In the end of this step, we set

$$\begin{aligned} \tilde{U} &\leftarrow \hat{U}(:, 1:k), \\ \tilde{B} &\leftarrow B_2. \end{aligned}$$

Remark 2 Also for this algorithm we observe that it is not necessary to update the matrix V , if only the left dominant singular subspaces are needed. In fact, in the orthogonalization process in step B.1, only the matrix U is involved.

At first glance, the proposed algorithm and the algorithm in [2] seem to be equivalent, i.e., seem to compute the same left dominant singular subspace. This is true if only one step is considered. However, if more iterations are considered, i.e., new vectors are processed, we come up with two different dominant singular subspaces because, in the proposed algorithm, the updating of the dominant subspace is made neglecting the contribution of the orthogonal matrices \tilde{Q}_1, \tilde{Q}_2 and \tilde{Q}_3 .

The proposed algorithm has been implemented in `Matlab`⁴. The code can be obtained from the authors upon request.

4 Numerical Experiments

In this section we compare the accuracy of the algorithm with respect to the algorithm proposed in [2].

Example 4.1 *Let $V \in \mathbb{R}^{m \times k}$ and $W \in \mathbb{R}^{k \times n}$ be full rank random matrices, generated by the `Matlab` function `randn` for different values of m, n , and k . Let $X = VW + \tau Z$, with $Z \in \mathbb{R}^{m \times n}$ be random matrices generated by the `Matlab` function `randn` and τ a parameter, assuming different values, that gives an indication of the level of noise, i.e., of the distance from the subspace spanned by the columns of V and that one spanned by the columns of X . If $\tau = 0$, V and X span the same subspace. Let V_1 and V_2 be the orthogonal subspaces tracked by the algorithm proposed in [2] (Alg1) and by the proposed algorithm (Alg 2). As a measure of the accuracy of the subspaces tracked, we have computed*

$$\|Q^H(I - V_i V_i^H)\|_2, \quad i = 1, 2,$$

with Q the Q -factor of the QR -factorization of V . The results are reported respectively in tables 1 and 2 for different values of m, n , and k .

It can be noticed that the accuracy of the subspaces tracked by the two methods is comparable and proportional to the amount of the noise introduced (the term τZ).

5 Conclusions

Many engineering applications require to compute the left dominant subspace of a matrix A of dimension $m \times n$, with $m \gg n$, whose columns are produced incrementally and only few columns can be processed at the same time. An

⁴ `Matlab` is a trademark of The MathWorks Inc.

$n = 200, m = 400, k = 10$		
τ	$\ Q^H(I - V_1V_1^H)\ _2$	$\ Q^H(I - V_2V_2^H)\ _2$
1.0e-002	8.9165e-03	2.7613e-002
1.0e-004	8.9164e-05	2.3277e-004
1.0e-006	8.9164e-07	1.8880e-006
1.0e-008	8.9164e-09	2.2056e-008
1.0e-010	8.9164e-11	2.2549e-010
1.0e-012	8.9681e-13	2.1514e-012
1.0e-014	2.7230e-13	2.8943e-013
$n = 200, m = 400, k = 20$		
τ	$\ Q^H(I - V_1V_1^H)\ _2$	$\ Q^H(I - V_2V_2^H)\ _2$
1.0e-02	1.5337e-02	8.3913e-02
1.0e-04	9.1570e-05	6.1078e-04
1.0e-06	9.1570e-07	6.3607e-06
1.0e-08	9.1570e-09	6.3626e-08
1.0e-10	9.1567e-11	7.1385e-10
1.0e-12	9.2926e-13	6.6060e-12
1.0e-14	3.9700e-13	4.8841e-13

Table 1

Levels of orthogonality between the subspace V and the tracked subspaces $V_i, i = 1, 2$.

algorithm for recursively computing an approximation of the left dominant subspace has been proposed in this paper. Although it requires a lower complexity for iteration of other algorithms available in the literature, the accuracy of the approximation of the left singular subspace computed by the proposed algorithm is comparable. Moreover, the proposed algorithm exhibits a lot of parallelism that can be exploited in a suitable implementation on a parallel computer.

References

- [1] S. Chandrasekaran, B.S. Manjunath, Y.F. Wang, J. Winkeler and H. Zhang, *An eigenspace update algorithm for image analysis*, Graphical Models and Image Processing **59** (5) (1997) 321–332.

$n = 200, m = 400, k = 50$		
τ	$\ Q^H(I - V_1V_1^H)\ _2$	$\ Q^H(I - V_2V_2^H)\ _2$
1.0e-02	1.1129e-02	4.5946e-01
1.0e-04	1.1128e-04	4.7361e-03
1.0e-06	1.1128e-06	6.0974e-05
1.0e-08	1.1128e-08	5.3917e-07
1.0e-10	1.1129e-10	5.2289e-09
1.0e-12	1.1342e-12	5.0099e-11
1.0e-14	4.9964e-13	7.6363e-13
$n = 800, m = 800, k = 20$		
τ	$\ Q^H(I - V_1V_1^H)\ _2$	$\ Q^H(I - V_2V_2^H)\ _2$
1.0e-02	1.1996e-02	1.1329e-01
1.0e-04	1.1996e-04	2.2984e-03
1.0e-06	1.1996e-06	8.3127e-06
1.0e-08	1.1996e-08	2.0457e-07
1.0e-10	1.1997e-10	1.8178e-09
1.0e-12	1.9262e-12	8.0744e-12
1.0e-14	1.9455e-12	2.3060e-12

Table 2

Levels of orthogonality between the subspace V and the tracked subspaces $V_i, i = 1, 2$.

- [2] Y. Chahlaoui, K. Gallivan, and P. Van Dooren, *Recursive calculation of dominant singular subspaces*, SIAM J. Matrix Anal. Appl. **25** (2) (2003) 445–463.
- [3] N. Mastronardi, M. Van Barel and R. Vandebril, *A note on the recursive calculation of dominant singular subspaces*, Numer. Algorithms, **38** (4) (2005) 237–242.
- [4] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Second ed., The Johns Hopkins University Press, Baltimore, MD, 1989.
- [5] N. Mastronardi, M. Van Barel, E. Van Camp, and R. Vandebril, *On computing the eigenvectors of a class of structured matrices*, J. Comput. Appl. Math., **189** (1–2) (2006) 580–591.
- [6] H. Park and S. Van Huffel, *Two-way bidiagonalization scheme for downdating the singular value decomposition*, Lin. Alg. Appl. **222** (1995) 23–39.
- [7] P. Van Dooren, *Gramian based model reduction of large-scale dynamical*

systems, in Numerical Analysis 1999, Chapman Hall/CRC Press, London (2000) 231–247.

- [8] S. Van Huffel and H. Park, *Parallel Tri- and Bi-Diagonalization of Bordered Bidiagonal Matrices*, Parallel Computing **20** (8) (1994) 1107–1128.
- [9] S. Van Huffel, J. Vandewalle and A. Haegemans, *An efficient and reliable algorithm for computing the singular subspace of a matrix, associated with its smallest singular values*, J. Comput. Appl. Math., **19** (1987) 313–330.
- [10] S. Van Huffel, *Iterative algorithms for computing the singular subspace of a matrix associated with its smallest singular values*, Lin. Alg. Appl., **154–156** (1991) 675–709.