SPIE conference
SAN DIEGO, August '89

120

1152-08

89-38C

Updating Singular Value Decompositions. A parallel implementation.

Marc Moonen
ESAT Katholieke Universiteit Leuven
K.Mercierlaan 94, 3030 Heverlee, Belgium

Paul Van Dooren
Philips Research Laboratory
Av. Van Becelaere 2, Box 8, 1170 Brussels, Belgium

Joos Vandewalle
ESAT Katholieke Universiteit Leuven
K.Mercierlaan 94, 3030 Heverlee, Belgium

Abstract

In this paper, we give an overview of a few recently obtained results regarding algorithms and systolic arrays for updating singular value decompositions. The Ordinary SVD as well as the Product SVD and the Quotient SVD will be discussed.

The updating algorithms consist in an interlacing of QR-updatings and a Jacobi-type SVD-algorithm applied to the triangular factor(s). At any time step an approximate decomposition is computed from a previous approximation, with a limited number of operations ($O(n^2)$). When combined with exponential weighting, these algorithms are seen to be highly applicable to tracking problems. Furthermore, they can elegantly be mapped onto systolic arrays, making use of slight modifications of well known systolic implementations for the matrix-vector product, the QR-updating and the SVD.

1 Introduction

Let A be an $m \times n$ matrix, with $(m \geq n)$. The *Ordinary Singular Value Decomposition* (OSVD) is a factorization of A into a product of three matrices

$$A_{m\times n} = U_{m\times n}\Sigma_{n\times n}V^t_{n\times n}$$

where $U^tU = I$, $V^tV = I$, and Σ is a diagonal matrix, with the singular values on the main diagonal.

The updating problem considered here, consists in computing the OSVD of the modified matrix

$$\begin{bmatrix} \lambda.A \\ a^t \end{bmatrix} = U^\star_{(m+1)\times n}\Sigma^\star_{n\times n}V^{\star t}_{n\times n}$$

making use of the original OSVD of A. Here a^t is a new row vector and λ is an exponential weighting factor, de-emphasizing "older" data. In many signal processing applications, it is necessary to continuously update the decomposition as additional rows are appended, which correspond to new data vectors. Such problems frequently occur in beamforming, direction finding, and spectral analysis [18]. In most cases only V and Σ are of interest.

Common techniques for updating the OSVD of a matrix after appending new rows, mostly reduce to solving the corresponding modified symmetric eigenvalue problem [2,3,7]. A drawback is the necessary knowledge of the *exact* original eigenstructure, in order to compute the updated eigenstructure. On the one hand, this considerably increases the computational load (e.g. in an on-line environment, where at each time step an *exact* updating is thus to be performed), while on the other hand, round-off errors are likely to accumulate unboundedly.

In this paper, we present an OSVD updating procedure that interlaces a Jacobi-type OSVD algorithm [1,10] with a QR-updating each time a new row is to be worked in. After each QR-update, the OSVD algorithm reduces the R-factor to an approximately diagonal matrix, by applying a sequence of plane rotations. It may be appropriate to try and perform only a fixed number of rotations (e.g. $n-1$, see below) on the triangular factor after each QR-update. The crux is then to show how only a few OSVD-steps after each QR-update can restore an acceptable approximation at every stage.

Such an updating algorithm can be mapped onto a *systolic array*, by appropriately combining systolic implementations for the matrix-vector product, the QR-updating and the SVD. Furthermore, it turns out that the procedure can be extended to generalized decompositions, viz. the *Quotient Singular Value Decomposition (QSVD)* (also called *Generalized Singular Value Decomposition*), and the *Product Singular Value Decomposition (PSVD)*, when new rows are appended to either one or both of the matrices involved. These updating problems had not really been looked at before, as far as we know. For the QSVD-updating problem, use is made of a QSVD-algorithm due to Paige. For the PSVD-updating on the other hand, we make use of an algorithm that essentially reduces to an algorithm due to Heath, Laub, Paige and Ward. Systolic arrays will be described as well.

The OSVD updating procedure is briefly presented in section 2. In section 3, a systolic implementation is presented, while in sections 4 and 5 it is shown how the updating procedure can be extended to QSVD-updating and PSVD-updating respectively.

2 Updating the Ordinary Singular Value Decomposition

For the time being, let us assume that the matrix A is $n \times n$ upper triangular. The OSVD of A can be computed by making use of a Jacobi strategy [10,12], that generates a sequence of triangular matrices R_k via the recurrence

$$\begin{aligned} R_1 &= A \\ R_{k+1} &= \Theta_k^t . R_k . \Phi_k \end{aligned}$$

Here Θ_k and Φ_k are *plane rotations*. To every k, there correspond a pivot index i and rotation angles θ_k and ϕ_k, such that Θ_k and Φ_k differ from the identity matrices only in the following entries

$$\begin{bmatrix} \Theta_k^{i,i} & \Theta_k^{i,i+1} \\ \Theta_k^{i+1,i} & \Theta_k^{i+1,i+1} \end{bmatrix} = \begin{bmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{bmatrix}$$

$$\begin{bmatrix} \Phi_k^{i,i} & \Phi_k^{i,i+1} \\ \Phi_k^{i+1,i} & \Phi_k^{i+1,i+1} \end{bmatrix} = \begin{bmatrix} \cos\phi_k & \sin\phi_k \\ -\sin\phi_k & \cos\phi_k \end{bmatrix}$$

The rotation angles θ_k and ϕ_k are chosen such that the $(i, i+1)$ element ("pivot") in R_k transforms to zero,

$$\begin{bmatrix} r_{k+1}^{i,i} & 0 \\ 0 & r_{k+1}^{i+1,i+1} \end{bmatrix} := \begin{bmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{bmatrix} \begin{bmatrix} r_k^{i,i} & r_k^{i,i+1} \\ 0 & r_k^{i+1,i+1} \end{bmatrix} \begin{bmatrix} \cos\phi_k & \sin\phi_k \\ -\sin\phi_k & \cos\phi_k \end{bmatrix}$$

As each iteration reduces the norm of the off-diagonal elements, R is gradually reduced to a diagonal matrix. Suppose that after r rotations a diagonal matrix (up to the relative precision ϵ) is obtained, then

$$\Theta_r^t.\Theta_{r-1}^t \ldots \Theta_1^t.(R_1).\Phi_1 \ldots \Phi_{r-1}.\Phi_r = R_r \overset{\epsilon}{\simeq} \Sigma$$

and

$$V \overset{\epsilon}{\simeq} \Phi_1 \ldots\ldots \Phi_{r-1}.\Phi_r$$

Each iteration thus reduces to a 2×2 OSVD on the main diagonal, setting the pivot element $r^{i,i+1}$ equal to zero. It is well known that by making use of *outer rotations*, one can obtain an efficient implementation [19], where the triangular structure is preserved at every stage. It suffices to let the pivot index successively take up all possible values

$$i = 1, 2, \ldots, n-1$$

in a cyclic manner, and perform 2×2 SVD's (with outer rotations) on the main diagonal accordingly [12].

If the rows of A become sequentially available, as is the case in an on-line environment, the diagonalization procedure can be interlaced with QR-updates [8]. After each QR-updating, the iterative diagonalization process of the R-factor is resumed. A sequential description is as follows.

Suppose that, at a certain time step k, the $A^{(k)}$-matrix is reduced to $R^{(k)}$ (upper triangular, almost diagonal). The corresponding V-matrix is $V^{(k)}$. (The $U^{(k)}$-matrix is not stored).

$$A^{(k)} = U^{(k)}.R^{(k)}.V^{(k)^t}$$

The modified matrix then equals

$$A^{(k+1)} = \begin{bmatrix} \lambda.A^{(k)} \\ a^{(k)^t} \end{bmatrix} = \begin{bmatrix} U^{(k)} & \\ & 1 \end{bmatrix} \begin{bmatrix} \lambda.R^{(k)} \\ a^{(k)^t}.V^{(k)} \end{bmatrix} V^{(k)^t}$$

Clearly, the R-factor is to be updated with the transformed input vector $a_\star^{(k)^t} = a^{(k)^t}.V^{(k)}$, resulting in

$$A^{(k+1)} = \left(\begin{bmatrix} U^{(k)} & \\ & 1 \end{bmatrix} . Q^{(k)}_{(n+1)\times n} \right) \begin{bmatrix} R^{(k)}_\star \\ 0 \end{bmatrix} V^{(k)^t}$$

The QR-updating clearly does not alter the $V^{(k)}$-matrix. Finally this updated $R^{(k)}_\star$ is further diagonalized through a series of subsequent rotations.

$$\begin{aligned} R^{(k+1)} &= \Theta^t_{k_r} . \Theta^t_{k_{r-1}} \ldots \Theta^t_{k_1} . (R^{(k)}_\star) . \Phi_{k_1} \ldots \Phi_{k_{r-1}} . \Phi_{k_r} \\ V^{(k+1)} &= V^{(k)} . \Phi_{k_1} . \ldots . \Phi_{k_{r-1}} . \Phi_{k_r} \end{aligned}$$

(where k_i refers to the i^{th} rotation in time step k.)

Thus, the QR-updatings and the diagonalization procedure are interlaced, yielding an approximate V-matrix at every stage. The *number of rotations* k_r should be chosen as a function of the off-diagonal part ("off-norm") in $R^{(k)}_\star$. Usually only very few rotations are necessary to restore a reasonable approximation at any time step. In the sequel, we assume that $(n-1)$ rotations are performed following each QR-update (where the pivot index successively takes up all values $i = 1, \ldots, n-1$). This choice seems particularly well suited for an efficient interlacing of QR-updates and OSVD-steps. Moreover, for practical applications, a reasonably low off-norm is observed when only $n-1$ rotations are performed after each QR-update. If however the number of rotations after each QR-update is e.g. halved or doubled, for instance the systolic array can easily be modified accordingly.

The updating procedure can now be summarized as follows (with $n-1$ rotations after each update) :

Initialization

$$V := I_{n\times n}$$
$$R := O_{n\times n}$$

Loop

for $k = 1, \ldots, \infty$
input new measurement vector $a^{(k)}$
QR*-updating*

$$\begin{aligned} a^{(k)}_\star &= a^{(k)^t} . V \\ R_{n\times n} &:= Q^t . \begin{bmatrix} \lambda . R \\ a^{(k)^t}_\star \end{bmatrix} \end{aligned}$$

further diagonalization
for $i = 1, \ldots, n-1$

$$\begin{aligned} R &:= \Theta^t_{k_i} . R . \Phi_{k_i} \\ V &:= V . \Phi_{k_i} \end{aligned}$$

end
end

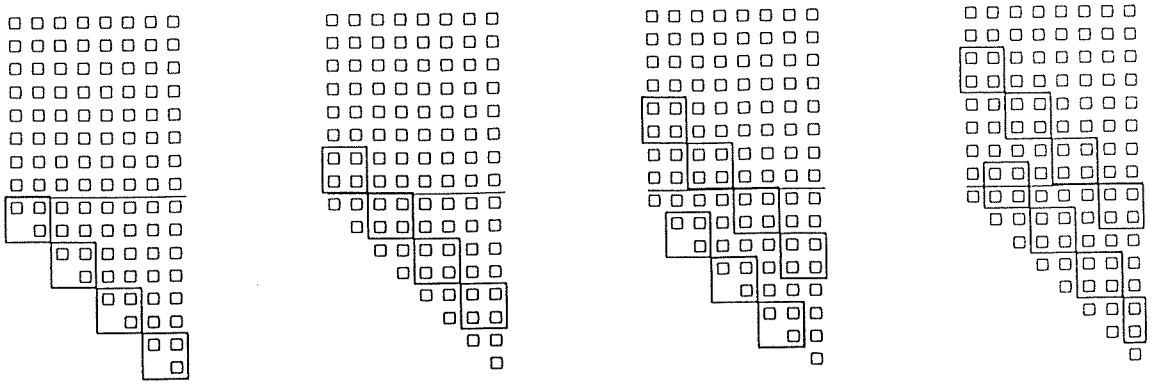


Figure 1: OSVD array

The numerical stability of this recursive algorithm is determined by local rounding errors on the one hand, and the propagation of global errors on the other hand. As the updating procedure requires only numerically well behaved orthogonal transformations and multiplications in which the multipliers are bounded by 1 in magnitude, local errors are apparently bounded. In [13] it is shown how the weighting factor $\lambda < 1$ prevents an unlimited growth of the errors, on condition that the stored V-matrix can be kept "close" to orthogonal. Furthermore it is shown that normalizing the columns of V from time to time, suffices to keep V close to orthogonal. See [13] for furhter details.

Finally, let us only mention that by combining modified rotations and a two-sided factorization of R on the one hand, and approximate OSVD schemes on the other hand, a *square root free updating algorithm* can be obtained, similar to well known square root free QR-updating algorithms. See [14] for further details.

3 A systolic OSVD updating array

The OSVD-updating algorithm can elegantly be mapped onto a systolic array. In particular if after each QR-update, $n-1$ Jacobi rotations are performed, an efficient parallel implementation is conceivable with $O(n^2)$ parallelism for $O(n^2)$ complexity. (If the number of rotations after each QR-update is e.g. halved or doubled, the array can easily be modified accordingly.)

As we intend to combine OSVD-diagonalization and QR-updating on the same array, let us first briefly review the triangular OSVD-array, as described in [12] (Figure 1). Processor elements on the main diagonal perform 2×2 OSVD's (combined with column and row permutations). Row transformation parameters are passed on to the right, while column transformation parameters are passed on upwards. Off-diagonal processors only apply and propagate these transformations. Column transformations are also propagated through the upper square part, containing the right singular V-matrix (V's first row in the top row, etc.).

Off-diagonal elements with odd and even row numbers are being annihilated in an alternative fashion *(odd-even ordering)*. It can easily be verified that an odd-even ordering corresponds to a cyclic by rows or columns ordering, except for a different start-up phase. The 2×2 OSVD's that are performed in parallel on the main diagonal can then be thought

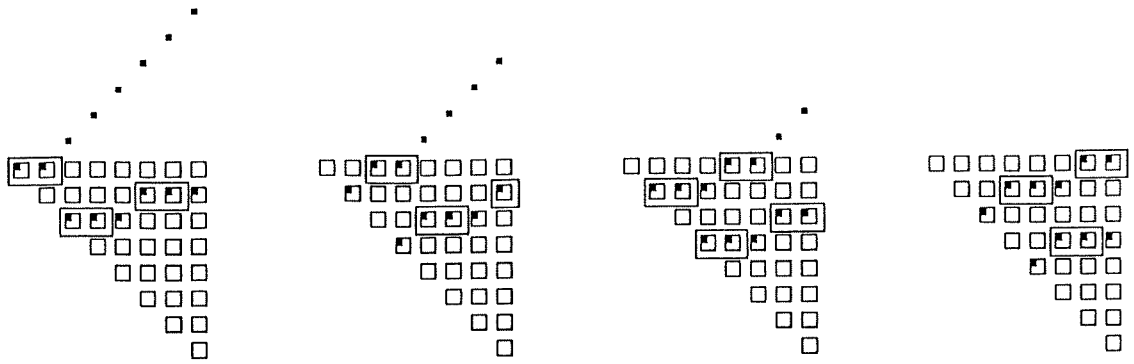


Figure 2: Modified QR-updating array

of as corresponding to different pipelined sequences of $n-1$ rotations, where in each sequence the pivot index successively takes up the values $i = 1, \ldots, n-1$. As pointed out in section 2, the QR-updatings should be inserted in between these sequences.

A QR-updating is performed by applying a sequence of orthogonal transformations *(Givens rotations)* [8]. The well known Gentleman-Kung QR-updating array [6] can be matched to the OSVD-array, as shown in Figure 2. Processor elements on the main diagonal compute and propagate the rotation parameters, while off-diagonal processors apply and propagate rotation parameters and intermediate results. While all operations remain unaltered as compared to the original array, the pipelining is somewhat different, so that the data vectors run through the array in a slightly different manner. The distribution of active processors now turns out to be the same as in the OSVD-array. Framed elements in Figure 2 then constitute the upper parts of the 2×2 blocks in Figure 1.

Without going into details, we give in Figure 3 a snapshot of the overall OSVD updating array (see [14] for a detailed description). New data vectors are continuously fed in, to the left-hand side of the array. The matrix-vector products $a^t.V$ are performed in the square part, in between two OSVD-rotation fronts. Intermediate result are propagated upwards, while the a-components are propagated to the right. The resulting vectors $a^\star = a^t.V$ become available at the top end, and are bounced back into the array, as indicated (■ becomes ▪). While propagating downwards, the $a_\star$-vector crosses upgoing rotation fronts. Clearly, the corresponding transformations should be applied to the $a_\star$-vector as well (briefly : $a^t.(V.\theta) = (a^t.V).\theta = a^t_\star\theta$). Each frame in the square array then corresponds to a (column) transformation, that is applied to a 2×2 block of the V-matrix, *and* to the two available components of the $a_\star$-vector. These components are propagated one step downwards next. The downward propagation of an $a_\star$-vector is thus carried out in exactly the same manner as the propagation in the modified Gentleman-Kung array (Figure 2).

Finally, the triangular array performs the OSVD-diagonalization, with interlaced QR-updatings. In each frame, column and row transformations corresponding to the OSVD diagonalization are performed first (see also Figure 1), where now column transformations should be applied to the $a_\star$-components as well. In a second step, only row transformations are performed corresponding to the modified QR-updating, affecting the $a_\star$-components and the upper part of the 2×2 -blocks (see also Figure 2).

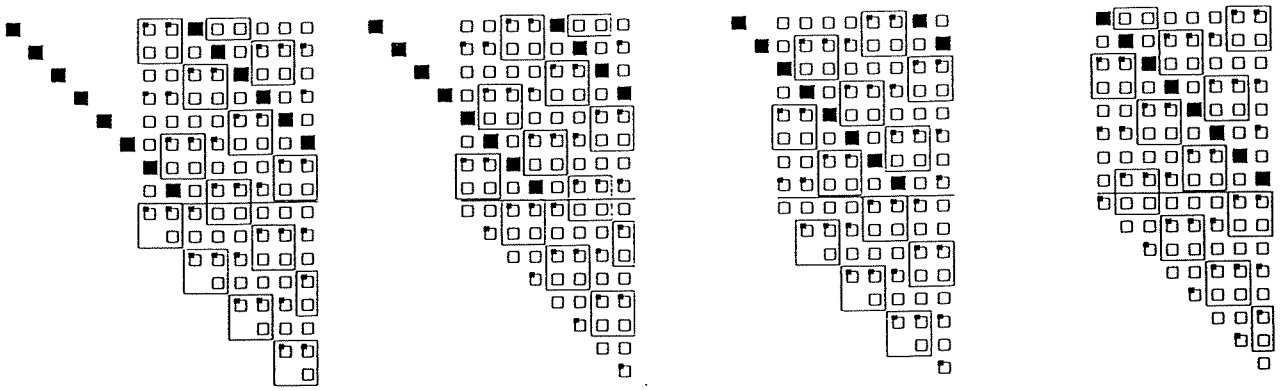


Figure 3: OSVD-updating array

4 Updating the Quotient Singular Value Decomposition

Let A and B be $m \times n$ and $p \times n$ matrices respectively, where $p, m \geq n$. The *quotient singular value decomposition* (QSVD) is a simultaneous reduction of A and B, by two matrices $(U_A)_{m\times n}$ and $(U_B)_{p\times n}$ (with orthonormal columns) and a common orthogonal matrix $Q_{n\times n}$, to triangular matrices, the corresponding rows of which are parallel [16]

$$U_A^t.A.Q = \Sigma_A.R$$

$$U_B^t.B.Q = \Sigma_B.R$$

We consider the following updating problem, that consists in computing the QSVD of the modified matrices, after applying exponential weighting and appending new rows to A and/or B.

$$\begin{bmatrix} \lambda.A \\ a^t \end{bmatrix} = (U_A^\star)_{(m+1)\times n}\Sigma_A^\star R^\star Q^{\star^t}$$

$$\begin{bmatrix} \lambda.B \\ b^t \end{bmatrix} = (U_B^\star)_{(p+1)\times n}\Sigma_B^\star R^\star Q^{\star^t}$$

Here a^t and b^t are new rows, while $\lambda \leq 1$ is an exponential weighting factor. In on-line applications, a continuous updating after each sampling is to be performed, for instance in adaptive total least squares computations, where one of the matrices is a data matrix, while the other one corresponds to the error covariance matrix. The U-matrices (with growing matrix dimensions when new rows are appended to the corresponding data matrix!) are never computed explicitly, as only Q and $\Sigma_A.R, \Sigma_B.R$ are of interest (in most cases).

Once again let us first assume that A and B are upper triangular. In [17], Paige developed a Jacobi-type QSVD algorithm, that has been elaborated into a fully parallel algorithm by Luk [11]. The algorithm generates a sequence of triangular matrices via the recurrences

$$\begin{aligned} (R_A)_1 &= A \\ (R_B)_1 &= B \end{aligned}$$

$$
\begin{aligned}
(R_A)_{k+1} &= \Theta_k^t.(R_A)_k.Q_k \\
(R_B)_{k+1} &= \Phi_k^t.(R_B)_k.Q_k
\end{aligned}
$$

where, similar to the OSVD case, 2×2 QSVD's are computed on the main diagonal, and the pivot index successively takes up all the values $i = 1, 2, \ldots, n-1$ in a cyclic manner. See [11,17] for further details.

A QSVD updating algorithm can then straightforwardly be devised, by interlacing the QSVD procedure with QR-updates each time new rows are to be worked in. For instance with a fixed number of rotations after each update ($n-1$ say) the updating procedure is summarized as follows:

Initialization

$$
\begin{aligned}
Q &:= I_{n\times n} \\
R_A &:= O_{n\times n} \\
R_B &:= O_{n\times n}
\end{aligned}
$$

Loop

for $k = 1, \ldots, \infty$

input new measurement vectors $a^{(k)}, b^{(k)}$

QR-updatings

$$a_\star^{(k)} = a^{(k)^t}.Q$$

$$b_\star^{(k)} = b^{(k)^t}.Q$$

$$R_A := (Q_A^t)_{n\times(n+1)} \begin{bmatrix} \lambda.R_A \\ a_\star^{(k)^t} \end{bmatrix}$$

$$R_B := (Q_B^t)_{n\times(n+1)} \begin{bmatrix} \lambda.R_B \\ b_\star^{(k)^t} \end{bmatrix}$$

QSVD steps

for $i = 1, \ldots, n-1$

$$
\begin{aligned}
R_A &:= \Theta_{k_i}^t.R_A.Q_{k_i} \\
R_B &:= \Phi_{k_i}^t.R_B.Q_{k_i} \\
Q &:= Q.Q_{k_i}
\end{aligned}
$$

end

end

Similar to the OSVD-updating array, one can straightforwardly design a QSVD-updating array, with a triangular part containing the triangular factors $R_A^{(k)}$ and $R_B^{(k)}$, and a square part containing the matrix $Q^{(k)}$. Figure 4 gives an overview. See [15] for further details.

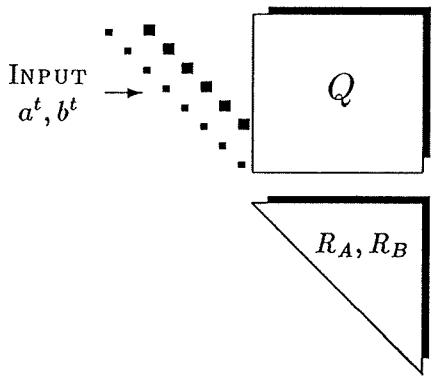


Figure 4: QSVD updating array

5 Updating the Product Singular Value Decomposition

The *Product Singular Value Decomposition* (PSVD) is a generalized singular value decomposition, at the same level of generality as the QSVD [5,4]. If we let A and B again be $m \times n$ and $p \times n$ matrices respectively, where $p, m \geq n$, then the PSVD of the matrix pair (A, B) is a simultaneous reduction of A and B, by two matrices $(U_A)_{m \times n}$ and $(U_B)_{p \times n}$ (with orthonormal columns) and a common orthogonal matrix $Q_{n \times n}$, to an upper and a lower triangular matrix as follows:

$$U_A^t.A.Q = \Sigma_A.R$$
$$U_B^t.B.Q = \Sigma_B.R^{-t}$$

The PSVD obviously reveals the OSVD of the product $A.B^t$, hence the name.

Analogous to the QSVD updating, we can consider the following updating problem

$$\begin{bmatrix} \lambda.A \\ a^t \end{bmatrix} = (U_A^\star)_{(m+1)\times n} \Sigma_A^\star R^\star Q^{\star t}$$

$$\begin{bmatrix} \lambda.B \\ b^t \end{bmatrix} = (U_B^\star)_{(p+1)\times n} \Sigma_B^\star R^{\star -t} Q^{\star t}$$

where a^t and b^t are new rows, and $\lambda \leq 1$ is an exponential weighting factor.

In order to compute the PSVD of the matrix pair (A, B), we can make use of a Jacoby-type algorithm, developed in [9]. If we assume that A and B are upper, resp. lower triangular matrices, a sequence of triangular matrices can be generated via the recurrences

$$\begin{aligned} (R_A)_1 &= A \\ (L_B)_1 &= B \\ (R_A)_{k+1} &= \Theta_k^t.(R_A)_k.Q_k \\ (L_B)_{k+1} &= \Phi_k^t.(L_B)_k.Q_k \end{aligned}$$

by performing 2×2 PSVD's on the main diagonal, where the pivot index successively takes up all the values $i = 1, 2, \ldots, n-1$ in a cyclic manner. See [9] for further details.

Again, a PSVD updating algorithm can straightforwardly be devised as follows (with $n-1$ rotations after each update). Note that QL-updating is used instead of QR updating for the lower triangular L_B-factor.

Initialization

$$\begin{aligned} Q &:= I_{n\times n} \\ R_A &:= O_{n\times n} \\ L_B &:= O_{n\times n} \end{aligned}$$

Loop

for $k = 1, \ldots, \infty$

input new measurement vectors $a^{(k)}, b^{(k)}$

QR and/or QL-updatings

$$a_\star^{(k)} = a^{(k)^t}.Q$$

$$b_\star^{(k)} = b^{(k)^t}.Q$$

$$R_A := (Q_A^t)_{n\times(n+1)} \begin{bmatrix} \lambda.R_A \\ a_\star^{(k)^t} \end{bmatrix}$$

$$L_B := (Q_B^t)_{n\times(n+1)} \begin{bmatrix} \lambda.L_B \\ b_\star^{(k)^t} \end{bmatrix}$$

PSVD steps

for $i = 1, \ldots, n-1$

$$\begin{aligned} R_A &:= \Theta_{k_i}^t.R_A.Q_{k_i} \\ L_B &:= \Phi_{k_i}^t.L_B.Q_{k_i} \\ Q &:= Q.Q_{k_i} \end{aligned}$$

end

end

A PSVD-updating array performing the above algorithm, is somewhat more involved, compared to the corresponding QSVD-array, due to the fact that L_B is lower triangular instead of upper triangular. An easy solution consists in providing two triangular arrays, containing R_A, resp. L_B, and two square arrays, both containing the orthogonal Q-matrix, as depicted in Figure 5. The topmost array performs matrix-vector products $a_\star^{(k)^t} = a^{(k)^t}.Q^{(k)}$, while the lowermost performs the products $b_\star^{(k)^t} = b^{(k)^t}.Q^{(k)}$. See [14] for details.

With respect to numerical reliability, it should be noted that duplicating the Q-matrix into a second square array, will not introduce any additional source of error build-up. Although the pipelining is somewhat different (which seems to rule out the possibility of devising a PSVD-updating array with one single square part), both square arrays perform exactly the same operations, introducing exactly the same round-off errors. Therefore the stored matrices in these arrays remain in perfect mutual agreement.

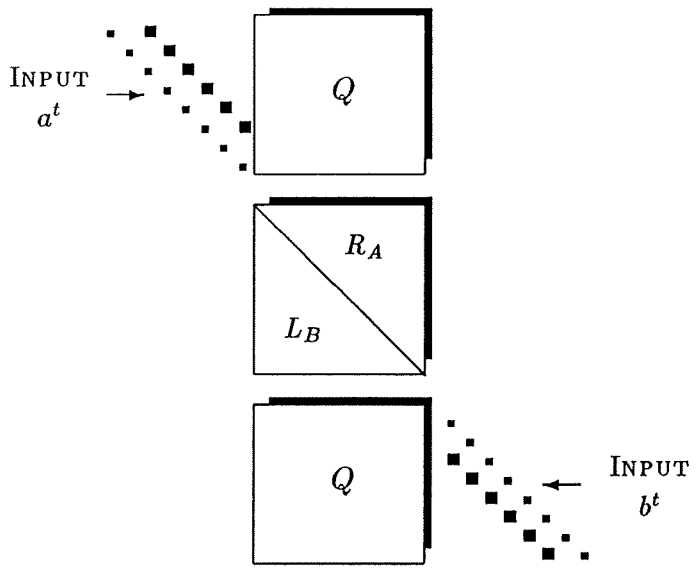


Figure 5: PSVD updating array

6 Conclusion

SVD updating procedures can be constructed as combinations of QR-updating and Jacobi-type SVD algorithms applied to the triangular factor(s). OSVD as well as QSVD and PSVD updating problems were discussed. These algorithms can be mapped onto systolic arrays. The updating is then performed at a speed comparable to the speed of merely QR-updating. For further details, we refer to [13,14] and [15].

Acknowledgement: M. Moonen is supported by the Belgian National Fund for Scientific Research (N.F.W.O.).

References

[1] BRENT R.P., LUK F.T., 1985. The solution of singular value and symmetric eigenvalue problems on multiprocessor arrays. SIAM J.Sc.Stat.Comp., Vol.6, No 1., pp 69-84.

[2] BUNCH J.R.,NIELSEN C.P. 1978. Updating the singular value decomposition Num.Math.31, pp 111-129.

[3] BUNCH J.R.,NIELSEN C.P.,SORENSEN D.C. 1978. Rank one modification of the symmetric eigenproblem. Num.Math.31, pp 31-48.

[4] DE MOOR B., 1989. *On the structure and geometry of the product singular value decomposition.* Numerical Analysis project, manuscript NA-89-05. Dept. Comp. Science, Stanford University, May 1989.

[5] FERNANDO K.V., HAMMARLING S.J., 1987. *A product induced singular value decomposition for two matrices and balanced realisation.* NAG Technical Report TR8/87.

MOONEN 12

[6] GENTLEMAN W.M, KUNG H.T., 1981. *Matrix triangularization by systolic arrays.* Real-Time Signal Processing IV, Proc. SPIE, Vol. 298.

[7] GOLUB G.H. 1973. Some modified eigenvalue problems. SIAM Review, vol. 15, no. 2, pp 318-334.

[8] GILL P.E., GOLUB G.H., MURRAY W., SAUNDERS M.A., 1974. *Methods for modifying matrix factorizations.* Mathematics of Computation, Vol. 28, 126, pp 505-535.

[9] HEATH M.T., LAUB A.J., PAIGE C.C, WARD R.C., 1986. *Computing the singular value decomposition of a product of two matrices.* SIAM J. Sc. Stat. Comp., Vol.7, No. 4, pp 1147-1159.

[10] KOGBETLIANTZ E., 1955. Solution of linear equations by diagonalization of coefficient matrices. Quart. Appl. Math. 13, 123-132.

[11] LUK F.T. 1985. *A parallel method for computing the GSVD.* Int. J. Parallel & distr. comp., no. 2, pp 250-260.

[12] LUK F.T., 1986. A triangular processor array for computing singular values. Lin. Alg. Appl., 77, pp 259-273.

[13] MOONEN M., VANDEWALLE J. 1988. (G)SVD updating for tracking slowly time-varying systems. ESAT K.U.Leuven, Internal report.

[14] MOONEN M., VAN DOOREN P. , VANDEWALLE J. *Parallel updating algorithms, part one : the ordinary singular value decomposition.* Submitted for publication.

[15] MOONEN M., VAN DOOREN P. , VANDEWALLE J. *Parallel updating algorithms, part two : generalized singular value decompositions.* Submitted for publication.

[16] PAIGE C.C., SAUNDERS M.A., 1981. *Towards a generalized singular value decomposition.* SIAM J. Num. Anal., Vol. 18, No. 3, pp 398-405.

[17] PAIGE C.C. 1986. *Computing the generalized singular value decomposition.* SIAM J. Sc. Stat. Comp., 7, pp 1126-1146.

[18] SPEISER J.M., 1986. *Signal processing computational needs.* Proc. SPIE, Vol. 696, pp 2-6.

[19] STEWART G.W., 1985. A Jacobi-like algorithm for computing the Schur decomposition of a nonhermitian matrix. SIAM J. Sc. Stat. Comp., Vol. 6, No. 4, pp 853-863.

[20] VAN LOAN C.F., 1976. *Generalizing the singular value decomposition.* SIAM J. Num. Anal., Vol. 13, No. 1, pp 76-83.