Rhodes Island

90-33

Combined Jacobi-type Algorithms in Signal Processing

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

A brief overview is given of recently developed parallel algorithms in the area of signal processing and systems theory and control. Firstly, approximate $O\{n^2\}$ algorithms are described for *SVD updating* problems, which are then shown to be stable and fairly accurate. Use is made of Jacobi-type SVD algorithms mainly, plus additional matrix operations such as triangular updates, matrix-vector multiplications, etc. This *combined Jacobi approach* results in regularly structured algorithms, which are efficiently implemented on parallel architectures such as systolic arrays. Similar developments have led to efficient algorithms for *Kalman filtering*, for solving *Riccati and Lyapunov equations*, for *Recursive Least Squares*, etc.

1 Introduction

Linear algebra and matrix computation play a central role in modern systems theory and digital signal processing. Representative areas in which matrix computation is important include beamforming, direction finding, spectrum analysis, image processing, systems identification, robust control, etc. As problems have grown in difficulty and complexity, numerical issues such as error propagation and problem sensitivity have become more important. The impact of *numerical linear algebra* or *numerical analysis* is apparent through the increased use of orthogonal techniques such as QR factorizations and singular value decompositions (SVD's). The SVD has become more or less *the* analytical tool for statistics, control engineering, etc., where numerous theoretical concepts have been reformulated in the language of the SVD [GoVl].

A main point of focus is *fast adaptive SVD* or *SVD updating* for real time applications. Previously described techniques for SVD row updating mostly reduce to solving the corresponding rank-one modified symmetric eigenvalue problem [BNS, BN, Go]. In each time step an *exact* decomposition is computed from a previous decomposition. The operation count is $O\{n^3\}$ where n is the problem size. For many practical applications this is excessive. Moreover, these procedures are apparently unstable.

As an alternative to these methods, we derived an SVD updating procedure as a combination of a Jacobi-type SVD procedure [Lu2] with QR updating [GK]. Apparently only a few SVD steps after each QR update suffice in order to restore an acceptable approximation for the SVD. This updating procedure is shown to be *cheap* –$O\{n^2\}$, similar to merely QR updating– *fairly accurate* and *particularly suited for parallel implementation.* Furthermore this procedure is easily extended to various generalized SVD updating problems –QSVD updating for instance– problems which had not been looked at before, as far as we know.

Our approach to SVD updating mainly consists in combining the Jacobi-type algorithm for the SVD with a few additional elementary matrix operations. In general, a broad range of algorithms can be pieced together with a fairly small sequence of key matrix algorithms –Jacobi-type algorithms mainly– especially orthogonal reductions (SVD's and QR decompositions), matrix inversions, matrix multiplications or matrix-vector multiplications, triangular backsolves, etc. Elementary (Jacobi-type) algorithms are then elegantly assembled into combined (Jacobi-type) algorithms. Such *combined algorithms* have been derived for *Kalman filtering*, for solving *Riccati and Lyapunov equations*, for *Recursive Least Squares*, etc.

Finally, for applications with substantial computational requirements such as speech and radar processing, *parallel algorithms and architectures* are indispensable, especially when a real time implementation is aimed at. Systolic or wavefront arrays then provide a means of exploiting large amounts of parallelism and hence achieving significant speed-ups. While practical parallel algorithms for most of the key matrix operations are by now available, implementing combined algorithms appears to be a major problem. We developed *systolic arrays* for the above mentioned combined algorithms.

The main points of focus are thus (i) fast algorithms for

the SVD updating and (ii) for similar combined problems, and (iii) parallel (systolic) implementations of these. In *section 2*, the SVD updating algorithm is reviewed, together with its performance and error analysis. In *section 3*, a parallel implementation for this SVD updating is outlined. In *Section 4* and *5*, we give an overview of SVD related generalizations and various other related algorithms. For detailed descriptions, reference is made to earlier reports.

2 SVD updating

In many signal processing applications, it is necessary to continuously *update matrix decompositions* as new observations are continuously added to the problem. Efficient updating techniques have long been known for the QR decomposition [GGMS] and the systolic implementation of this is well known by now [GK]. The similar and equally important, but more difficult problem of updating the SVD when new rows are appended, has only recently been addressed, see e.g. [Sch, CG].

An efficient updating technique for the SVD is derived by supplementing the QR updating with an SVD procedure applied to the triangular factor. Computing an *exact* decomposition after each QR update, would then take $O\{n^3\}$ operations. For real-time applications this is mostly excessive. The computational load is alleviated if only an *approximate* decomposition is computed after each QR update instead. The trade-off is between accuracy and efficiency. As for *subspace tracking problems*, it is shown how an $O\{n^2\}$ *approximate SVD updating algorithm* keeps the tracking error down to approximately the time variation in $O\{n\}$ time steps. For applications with slowly time-varying systems this is acceptable. The SVD updating is seen to exhibit all of the desirable properties of merely QR updating. Apart from being cheap, fairly accurate and particularly amenable to parallel implementation –see *Section 3*– the SVD updating is shown to be stable when supplemented with a certain re-orthogonalization procedure, which elegantly combines with the updating scheme.

We give an overview, referring to [MVV1, MVV2] for the details

Algorithmic description

Given that we have the SVD of a matrix $A_{m\times n} = U_{m\times n} \cdot \Sigma_{n\times n} \cdot V^t_{n\times n}$, $(m \geq n)$, we may need to calculate the SVD of a matrix $\underline{A}$ that is obtained after appending a new row to A.

$$\underline{A} = \begin{bmatrix} A \\ a^t \end{bmatrix} = \underline{U}_{(m+1)\times n} \cdot \underline{\Sigma}_{n\times n} \cdot \underline{V}^t_{n\times n}$$

In on-line applications, a new updating often needs to be performed after each sampling. The data matrix at time step k is then defined in a recursive manner $(k \geq n)$

$$A^{\{k\}} = \begin{bmatrix} \lambda^{\{k\}} \cdot A^{\{k-1\}} \\ a^{\{k\}t} \end{bmatrix}$$

Factor $\lambda^{\{k\}}$ is a *weighting factor*, and $a^{\{k\}}$ is the *input vector* at time instance k. For the sake of brevity, we consider only the case where $\lambda^{\{k\}}$ is a constant λ, although everything can easily be recast for the case where it is time-varying. Finally, in most cases the $U^{\{k\}}$-matrix –with growing matrix dimensions– need not be computed explicitly, and only $V^{\{k\}}$ and $\Sigma^{\{k\}}$ are explicitly updated.

In the algorithmic description below, the R-matrix is a triangular factor which is meant to be close to Σ. In each time step, a new row is worked in (matrix-vector multiplication and QR updating). The R-matrix is then slightly adjusted with an SVD procedure. Index i is the *pivot index*, and the matrices $\Theta_i^{\{k\}}$ and $\Phi_i^{\{k\}}$ represent rotations in the $\{i, i+1\}$-plane. Each rotation reduces the off-norm in R, such that R is kept close to a (block)-diagonal matrix, in spite of the QR updates. Both the QR updating and the rotations following the update take $O\{n^2\}$ operations, which in particular results in an elegant parallel implementation, see *Section 3*.

Initialization

$$V^{\{0\}} \Leftarrow I_{n\times n}$$
$$R^{\{0\}} \Leftarrow O_{n\times n}$$

Loop

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

input new measurement vector $a^{\{k\}}$

matrix-vector multiplication and exp. weighting

$$a_\star^{\{k\}t} \Leftarrow a^{\{k\}t} \cdot V^{\{k-1\}}$$
$$R_\lambda^{\{k\}} \Leftarrow \lambda \cdot R^{\{k-1\}}$$

QR updating

$$\begin{bmatrix} R_\star^{\{k\}} \\ 0 \end{bmatrix} \Leftarrow Q^{\{k\}t} \cdot \begin{bmatrix} R_\lambda^{\{k\}} \\ a_\star^{\{k\}t} \end{bmatrix}$$

$$R^{\{k\}} \Leftarrow R_\star^{\{k\}}$$
$$V^{\{k\}} \Leftarrow V^{\{k-1\}}$$

SVD steps

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

$$R^{\{k\}} \Leftarrow \Theta_i^{\{k\}t} \cdot R^{\{k\}} \cdot \Phi_i^{\{k\}}$$
$$V^{\{k\}} \Leftarrow V^{\{k\}} \cdot \Phi_i^{\{k\}}$$

end

end

Performance analysis

The performance of the SVD updating scheme has been analyzed in detail. First of all, a *data model* is put forward, which applies to popular signal processing applications, such as harmonic retrieval (e.g. MUSIC) direction finding (e.g. ESPRIT) and system identification.

For such applications, an SVD step is used to separate a signal subspace from a noise subspace, see below, corresponding to certain submatrices of $V^{(\cdot)}$. Further information –e.g. the angles of arrival for ESPRIT– can then be computed from the knowledge of these submatrices only. As for an *approximate SVD* –with an *approximately* diagonal triangular factor $R^{(\cdot)}$– the *subspace separation error (tracking error)* is shown to be related to the norm of some off-diagonal block of cross terms in the triangular factor. The effect of the SVD procedure on this norm is investigated and finally, all this is applied to the updating problem.

The *data model* we consider only assumes that at each time step k the data matrix $A^{(k)}$ has a fixed number d of *large* singular values

$$\sigma_i^{(k)} \qquad i = 1, \ldots, d$$

and a remaining number of *small* singular values

$$\sigma_i^{(k)} \qquad i = d+1, \ldots, n$$

The ratio

$$\mathrm{SN}^{(k)} = \frac{\sigma_d^{(k)}}{\sigma_{d+1}^{(k)}}$$

can be interpreted as a *signal-to-noise ratio*, and is assumed to be large, e.g. at least 10 or 100. The corresponding submatrices $V_{\mathrm{s}}^{(k)^\circ}$ and $V_{\mathrm{n}}^{(k)^\circ}$ in $V^{(k)^\circ}$ [1] define the *signal subspace* $\Re\{V_{\mathrm{s}}^{(k)^\circ}\}$ [2], and the *noise subspace* $\Re\{V_{\mathrm{n}}^{(k)^\circ}\}$ [3], orthogonal to each other.

As we consider time-varying systems, we need to define a measure of *time variation* one way or another. In view of the applications at hand, it is indicated to make use of the canonical angles $\theta_i^{\{k-1\}\to\{k\}}$ between $\Re\{V_{\mathrm{s}}^{\{k-1\}^\circ}\}$ and $\Re\{V_{\mathrm{s}}^{\{k\}^\circ}\}$. We then define the *time variation* from time step $k-1$ to time step k –for a prespecified choice for d– as the *distance* between the corresponding signal subspaces, which in turn can be defined in terms of the canonical angles as follows

$$\begin{aligned} \mathrm{TV}_d^{\{k-1\}\to\{k\}} &\stackrel{\mathrm{def}}{=} \mathrm{dist}\{\Re\{V_{\mathrm{s}}^{\{k-1\}^\circ}\}, \Re\{V_{\mathrm{s}}^{\{k\}^\circ}\}\} \\ &\stackrel{\mathrm{def}}{=} \sqrt{\sum_{i=1}^{d} \tan^2 \theta_i^{\{k-1\}\to\{k\}}} \end{aligned}$$

The adaptive SVD algorithm at each time step stores a triangular factor $R^{(k)}$ of the data matrix $A^{(k)}$, as well as an approximation $V^{(k)}$ for the matrix of right singular vectors. Suppose that at a certain time step k, $R^{(k)}$ and $V^{(k)}$ can be split up as follows

$$\begin{aligned} R^{(k)} &= \begin{bmatrix} R_{\mathrm{s}} & R_{\mathrm{sn}} \\ 0 & R_{\mathrm{n}} \end{bmatrix} \\ V^{(k)} &= \begin{bmatrix} V_{\mathrm{s}} & V_{\mathrm{n}} \end{bmatrix} \end{aligned}$$

where R_{n} and R_{sn} are "small". It is clear that V_{s} provides an *approximate basis* for the signal subspace. We can then define the *tracking error* at time step k as follows

$$\mathrm{TE}_d^{\{k\}} \stackrel{\mathrm{def}}{=} \mathrm{dist}\{\Re\{V_{\mathrm{s}}^{\{k\}}\}, \Re\{V_{\mathrm{s}}^{\{k\}^\circ}\}\}$$

The aim is now to derive a useful estimate for the tracking error TE_d in terms of the time variation TV_d. The performance analysis results in an estimate for the tracking error as follows

$$\mathrm{TE}_d^{\{k+n\}} \simeq \mathrm{TV}_d^{\{k\}\to\{k+n\}}$$

The tracking error is thus approximately bounded by the time variation in n time steps. In other words

$$\mathrm{TE}_d^{\{k\}} = O\{n\}$$

An approximate SVD scheme with $O\{n^2\}$ instead of $O\{n^3\}$ operations per update, thus yields an $O\{n\}$ tracking error. For applications with slowly time-varying systems, this error is sufficiently small.

Error analysis

The updating algorithm in principle keeps on running as long as new data vectors are fed in. Clearly, round-off error build-up is to be avoided at any cost. For QR updating, performing orthogonal transformations in combination with exponential weighting –de-emphasizing older data as well as accumulated errors– is found to be sufficiently stable in practice [McW]. For the SVD updating problem however, one has in addition to store an orthogonal matrix V and iteratively update it by orthogonal column transformations, along with the triangular factor.

With infinite precision arithmetic, at each time step k, the updating algorithm would provide a triangular factor $R^{(k)}$ and an orthogonal matrix $V^{(k)}$. Due to round-off errors, the matrices stored in the computer, $\mathcal{R}^{(k)}$ and $\mathcal{V}^{(k)}$ say, differ from the exact ones. The computed transformations for the QR updating and the SVD steps –apart from the round-off– then differ as well. As we are only interested in the singular values, together with the right singular vectors, the only meaningful requirement would then be that $R^{(k)}V^{(k)'}$ and $\mathcal{R}^{(k)}\mathcal{V}^{(k)'}$ have "close" singular values and right singular vectors. Therefore, the relevant *error matrix* $(n \times n)$ is defined as follows

$$\begin{aligned} \Delta^{(k)} &\stackrel{\mathrm{def}}{=} (R^{(k)}V^{(k)'})' \cdot (R^{(k)}V^{(k)'}) \\ &\qquad -(\mathcal{R}^{(k)}\mathcal{V}^{(k)'})' \cdot (\mathcal{R}^{(k)}\mathcal{V}^{(k)'}) \\ &= A^{(k)'} \cdot A^{(k)} - (\mathcal{R}^{(k)}\mathcal{V}^{(k)'})' \cdot (\mathcal{R}^{(k)}\mathcal{V}^{(k)'}) \end{aligned}$$

An *error analysis* has been performed, resulting in a *first order difference equation* for the propagation of $\Delta^{(\cdot)}$

$$\Delta^{(k)} = \lambda^2 \cdot \Delta^{(k-1)} + \delta E^{(k)}$$

where λ is the exponential weighting factor ($\lambda < 1$). *As long as $\mathcal{V}^{(\cdot)}$ is sufficiently close to an orthogonal matrix, $\delta E^{(k)}$ contains only bounded local errors, independent of*

[1] superscripts $^\circ$ refer to exact decompositions
[2] row space of $A^{(k)}$
[3] kernel of $A^{(k)}$

$\Delta^{(k-1)}$. The algorithm is thus *stable on condition that the stored V-matrix can be kept close to orthogonal.* It is shown how an additional *re-orthogonalization procedure* keeps the deviation from orthogonality sufficiently small in order to guarantee stability of the overall updating scheme.

3 A systolic array for SVD updating

The approximate SVD updating scheme of the previous section is seen to consist of a Jacobi-type SVD procedure, interlaced with QR updatings and matrix-vector multiplications, and possibly supplemented with a certain re-orthogonalization scheme. This updating algorithm can be mapped onto a *systolic array* with $O\{n^2\}$ processors. It is seen that *all of the above operations can perfectly be pipelined on one single processor array, if one allows for a limited number of supplementary computations.* In our array, different computational fronts run in opposite directions. Whenever two such fronts meet, intermediate results have to be updated, hence the additional computations. However, in this way we are able to fully exploit the available parallelism. The obtained throughput is $O\{n^0\}$ –much like it is the case for merely QR updating. The array is complicated at first sight, but extremely efficient, perfectly modular, repetitive, etc. The idea of designing this type of arrays –with crossing computational fronts– is apparently new.

The SVD updating array is similar to the triangular SVD array in [Lu2] where the SVD diagonalization and a preliminary QR factorization are performed on the same array. As for the SVD updating algorithm, the diagonalization process and the QR updating are interlaced, so that the array has to be modified accordingly. Also, from the algorithmic description it follows that the V-matrix should be stored as well. Hence we have to provide for an additional square array, which furthermore performs the matrix-vector products $a^t \cdot V$. In [MVV2] it is shown how the matrix-vector product, the QR updating and the SVD can perfectly be pipelined at the cost of a small computational overhead. Finally, it is shown how e.g. total least squares solutions can be generated at each time step, with only very few additional computations.

Figure 1 gives an overview of the array. New data vectors are continuously fed in, to the left-hand side of the array. The matrix-vector product is computed in the square part, and the resulting vector is passed on to the triangular array that performs the QR updating and the SVD diagonalization. Solution vectors are flushed in the opposite direction, and become available at the right-hand side of the array. All these operations can be carried out simultaneously. The correctness of this array has been verified by software simulation. See [MVV2] for details.

A main computational bottle-neck for such an array appears to be the computation of the rotation parameters in the boundary processors, with explicit computation of a number of square roots. It is shown how a *square root free (SQF) implementation* can be obtained by combining modified Givens rotations with approximate SVD schemes. This is an extension of the well known SQF QR updating [Ge]. See [MVV2] for details.

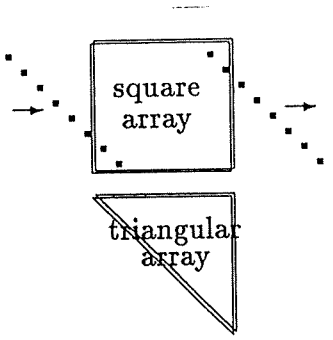


Figure 1

Finally, Jacobi-type matrix algorithms are mostly mapped on *orthogonally* connected processor arrays. An alternative partitioning has been developed, resulting in a grid of *hexagonally* connected processors. This partitioning is shown to be over four times more efficient, compared to the original configuration. See [MV6] for details.

4 Generalized SVD and rank-one updating

The SVD updating procedure is straightforwardly extended to *generalized SVD's*, viz. the quotient SVD (QSVD) and the product SVD (PSVD), and furthermore to various *rank-1 updating problems.*

The QSVD is a simultaneous diagonalization of any two matrices A and B with the same number of columns [PS, DG]. QSVD applications include generalized least squares problems associated with the general Gauss-Markov linear model, generalized total least squares problems, generalized state space identification, etc. The PSVD is a generalized singular value decomposition, at the same level of generality as the QSVD [FH, Dem, DG]. The PSVD appears in a number of problems in systems theory and signal processing, e.g. via the orthogonal Procrustes problems in harmonic retrieval and direction finding, in computing balancing transformations, in computing the Kalman decomposition, etc.

We considered P- and QSVD updating problems where new rows are appended to either one or both of the matrices involved. As far as we know, these problems had not been looked at so far. For the QSVD updating problem, use is made of a Jacobi-type QSVD algorithm due to Paige [Pa]. For the PSVD updating on the other hand, we make use of a similar algorithm that essentially reduces to one in [HLPW], except that the context is somewhat different here. Systolic arrays can again be obtained as combined implementations of Jacobi-type SVD algorithms with matrix-vector multipli-

cations and QR updates. As for the QSVD it is shown how additional backsubstitutions are required for computing solution vectors. The backsubstitutions are intertwined with the other operations at the cost of a few additional computations, such that the throughput remains $O\{n^0\}$.

Finally, rank-1 updating algorithms can be constructed by combining Jacobi-type algorithms with the rank-1 updating itself –instead of QR updating. We considered the symmetric eigenvalue problem, the (generalized) SVD, and (generalized) Schur decomposition. Systolic arrays are derived in a similar fashion.

For details, we refer to [MVV3].

5 Related problems

While the SVD updating problem is important in its own right, the suggested updating procedure seems to be only one representative of a whole class of important (updating) algorithms for signal processing and systems theory. The key tool is the *Jacobi approach*. For almost all matrix decompositions, Jacobi-type algorithms are available, or existing algorithms can be reformulated and worked into Jacobi-type algorithms based on locally computed plane transformations. Apart from the SVD algorithm [Lu2], efficient Jacobi-type algorithms have been derived for the symmetric eigenvalue problem [BL], for the Schur [St] and generalized Schur decomposition [ChVd1], for the generalized SVD [Lu1, Pa] and the product SVD [HLPW], for the QR decomposition [Lu3], LDU and Cholesky decomposition, for inverting triangular matrices, etc. The primary appeal of the Jacobi approach is its simplicity, which translates into an efficient and elegant implementation on parallel processor arrays.

Many practical algorithms in e.g. control and filtering essentially consist of a *combination* of elementary matrix operations, most of which can be recast as Jacobi-type operations. As an example the SVD updating algorithm essentially consists of a Jacobi-type SVD algorithm combined with a QR updating –which can be re-interpreted as a Jacobi-type algorithm as well– plus additional matrix-vector multiplications. More generally, we can assemble *elementary* Jacobi-type algorithms into *combined* Jacobi-type algorithms. The resulting algorithms are then regularly structured and have potential for efficient parallel implementation. Apart from the SVD updating algorithm, we derived Jacobi-type algorithms for Kalman filtering, for solving Riccati and Lyapunov equations and for recursive least squares computations.

1. Recently developed *recursive least squares (RLS) schemes with inverse factorization* [HShMcW, PP] are known to be particularly suited for parallel implementation, but unfortunately unstable. We show how the procedure can be stabilized by including an error correction scheme based on Jacobi-type matrix inversion. The RLS scheme –QR updating mainly– and the correction scheme can be combined on a systolic array, much like it is the case for the SVD updating procedure. For details, we refer to [MV4].

2. Secondly, a triangular processor array for *square root Kalman filtering* is developed starting from well known updating formulas. The QR updating is combined with a Jacobi-type algorithm for QR factorization. The resulting array is slightly less efficient than the array in [Ku] –which seems to be the most efficient array ever described in the literature– but on the other hand possesses a much more regular structure than the latter and is therefore thought to offer a useful alternative in view of hardware implementation. For details, we refer to [MV5].

3. Finally, a systolic algorithm is devised solving *Riccati and Lyapunov equations*. The recursive matrix sign function algorithm is known to provide a suitable iterative method for solving these equations. In a recent paper a factored version of this algorithm is given, which is particularly amenable to parallel implementation [ChVd2]. The basic operations involved are triangular matrix inversions and triangular updates. This algorithm is worked into a Jacobi-type algorithm, by combining Jacobi-type algorithms for inverting and updating triangular matrices. The resulting array is conceptually simpler and furthermore roughly three times more efficient than the array in [ChVd2]. For details, we refer to [MV7].

6 Conclusion

Apart from a collection of new algorithms and corresponding systolic arrays –which are important in their own right– we believe our main contribution is in the development of a framework for devising combined algorithms –based on the Jacobi approach– linked up with the development of systolic arrays for implementing these combined algorithms. As such, the benefit of developing algorithms and architectures together is especially highlighted.

Acknowledgement : This research work was carried out at the ESAT laboratorium of the Katholieke Universiteit Leuven, in the framework of the Community Research programme ESPRIT Basic Research Action Nr. 3280, and was partially sponsored by the European Commission. Marc Moonen is a research assistant with the N.F.W.O. (Belgian National Fund for Scientific Research).

References

[BL] R.P. Brent and F.T. Luk. The solution of singular value and symmetric eigenvalue problems on multiprocessor arrays. *SIAM J. Sci. Stat. Comp.*, **6**(1985), No. 1, 69-84.

[BNS] J.R. Bunch, C.P. Nielsen and D.C. Sorensen. Rank one modification of the symmetric eigenproblem. *Numer. Math.*, **31**(1978), pp 31-48.

[BN] J.R. Bunch and C.P. Nielsen. Updating the singular value decomposition. *Numer. Math.*, **31**(1978), pp 111-

129.

[ChVd1] J.-P. Charlier and P. Van Dooren. A Jacobi-like algorithm for computing the generalized Schur form of a regular pencil. *J. Comp. Appl. Math.*, **27**(1989), 17-36.

[ChVd2] J.-P. Charlier and P. Van Dooren. A systolic algorithm for Riccati and Lyapunov equations. *MCSS*, **2**(1989), 109-136.

[CG] P. Comon and G.H. Golub. Tracking a few extreme singular values and vectors in signal processing. *Numerical Analysis Project, Computer Science Dept. Stanford University*, NA-89-01(1989).

[Dem] B. De Moor. On the structure and geometry of the product singular value decomposition. *Numerical Analysis project, Dept. Comp. Science, Stanford University*, NA-89-05(1989).

[DG] B. De Moor and G.H. Golub. Generalized Singular Value Decompositions: A proposal for a standardized nomenclature. *Internal Report, Dept. Comp. Sc., Stanford University*, (1989).

[FH] K.V. Fernando and S.J. Hammarling. A product induced singular value decomposition for two matrices and balanced realization. *NAG Technical Report*, **TR-8/87**(1987).

[GI] F. Gaston and G. Irwin. Systolic approach to square root information Kalman filtering. *Int. J. Control*, **50**(1989), No. 1, 225-248.

[Ge] W.M. Gentleman. Least squares computations by Givens transformations without square roots. *J. Inst. Maths Applics.*, **12**(1973), pp 329-336.

[GK] W.M Gentleman and H.T. Kung. Matrix triangularization by systolic arrays. *Real-Time Signal Processing IV, Proc. SPIE*, Vol. **298**(1981), 19-26.

[GGMS] P.E. Gill, G.H. Golub, W. Murray, M.A. Saunders. Methods for modifying matrix factorizations. *Math. Comp.*, Vol. **28**(1974), No. 126, pp 505-535.

[Go] G.H. Golub. Some modified eigenvalue problems. *SIAM Review*, Vol. **15**(1973), No. 2, pp 318-334.

[GoVl] G.H. Golub and C.F. Van Loan. *Matrix computations.* North Oxford academic Publishing Co., Johns Hopkins Press (1983).

[HLPW] M.T. Heath, A.J. Laub, C.C. Paige and R.C. Ward. Computing the singular value decomposition of a product of two matrices. *SIAM J. Sci. Stat. Comp.*, **7**(1986), No. 4, 1147-1159.

[HShMcW] J. Hudson, T.J. Shepherd, J.G. McWhirter. Parallel weight extraction by a systolic least squares algorithm. *Advanced Algorithms and Architectures for Signal Processing IV, Proc. SPIE*, Vol. **1152**(1989) (in press).

[Ku] S.Y. Kung. *VLSI array processors.* Englewood Cliffs, NJ., Prentice Hall (1988).

[Lu1] F.T. Luk. A parallel method for computing the GSVD. *Int. J. Parallel & distr. comp.*, **2**(1985), 250-260.

[Lu2] F.T. Luk. A triangular processor array for computing singular values. *Lin. Alg. Appl.*, **77**(1986), 259-273.

[Lu3] F.T. Luk. A rotation method for computing the QR-decomposition. *SIAM J. Sci.Stat. Comp.*, **7**(1986), No 2, 452-459.

[McW] J.G. McWhirter. Recursive Least Squares Minimization using a systolic array. *Real-Time Signal Processing VI, Proc. SPIE*, Vol. **431**(1983), 105-112.

[MVV1] M. Moonen, P. Van Dooren, J. Vandewalle. An SVD updating algorithm for subspace tracking. *Submitted for publication* (also ESAT-SISTA report 89-13a)

[MVV2] M. Moonen, P. Van Dooren, J. Vandewalle. A systolic array for SVD updating. *Submitted for publication* (also ESAT-SISTA report 89-13b).

[MVV3] M. Moonen, P. Van Dooren, J. Vandewalle. Parallel updating algorithms, part two : generalized singular value decompositions. ESAT-SISTA report 89-23.

[MV4] M. Moonen, J. Vandewalle. Recursive least squares with stabilized inverse factorization. *To appear in Signal Processing* (also ESAT-SISTA report 89-26).

[MV5] M. Moonen, J. Vandewalle. A systolic array for square root Kalman filtering. *Submitted for publication* (also ESAT-SISTA report 89-27).

[MV6] M. Moonen, J. Vandewalle. A hexagonally connected processor array for Jacobi-type matrix algorithms. *Electronics Letters, March 1990, pp 400-401.*

[MV7] M. Moonen, J. Vandewalle. A systolic algorithm for Riccati and Lyapunov equations. *Submitted for publication* (1990).

[PS] C.C. Paige and M. Saunders. Towards a generalized singular value decomposition. *SIAM J. Numer. Anal.*, Vol. **18**(1981), No. 3, pp 398-405.

[Pa] C.C. Paige. Computing the generalized singular value decomposition. *SIAM J. Sci. Stat. Comp.*, Vol. **7**(1986), pp 1126-1146.

[PP] C.T. Pan, R.J. Plemmons. Least squares modifications with inverse factorization: parallel implications. *J. Computational and Applied Mathematics*, **27**(1989), No. 1-2, 109-127.

[Sch] R. Schreiber. Implementation of adaptive array algorithms. *IEEE Trans.* **ASSP-34**(1986), pp 1038-1045.

[St] G.W. Stewart, A Jacobi-like algorithm for computing the Schur decomposition of a nonhermitian matrix. *SIAM J. Sci. Stat. Comp.*, **6**(1985), No. 4, 853-863.