# PARALLEL AND STABLE SPHERICAL SUBSPACE TRACKING

*Filiep Vanpoucke and Marc Moonen*

Katholieke Universiteit Leuven
Dept. of Electrical Engineering, ESAT
K. Mercierlaan 94, 3001 Leuven, Belgium

## ABSTRACT

We introduce a factored spherical SVD updating algorithm which can be used for subspace tracking. It is a non-iterative algorithm for approximate SVD updating. The orthogonal matrix tracking the signal subspace is parameterized as a sequence of Givens rotations. This factorization has two important advantages. On the algorithmic level it cures the error accumulation problem inherent in the algorithm. The subspace matrix is now confined to the manifold of orthogonal matrices at all time. On the architectural level the factored algorithm is more amenable to parallel - even systolic - implementation. Moreover, the SFG contains only rotation nodes. Therefore, an ideal processor for a real-time parallel ASIC architecture is a CORDIC processor.

## 1. SPHERICAL SVD UPDATING

Estimating a $D$-dimensional dominant eigenspace of an $M \times M$ correlation matrix is a key component in high resolution direction finding algorithms such as MUSIC, ESPRIT and WSF. In this array processing application $M$ is the size of the sensor array, and $D$ is the number of narrow-band emitters at the center frequency of interest. The accuracy of the signal parameter estimates ultimately depends on the accuracy of the signal eigenspace estimate. In on-line applications one has to track the dominant subspace of the growing data matrix

$$X_{[k]} = \left[\begin{array}{c} \alpha X_{[k-1]} \\ \hline x_{[k]}^H \end{array}\right],$$

where $x_{[k]} \in \mathbb{C}^M$ is the new data vector at sampling time $t_k$ and $\alpha < 1$ is a scalar exponential weighting factor. A cheap approximate updating algorithm is the ROSA (Rank-one update, signal averaging) algorithm [1, 2]. At each time the exact singular value decomposition (SVD) of the data matrix is approximated by

$$X_{[k]} \approx \tilde{X}_{[k]} = \left[\begin{array}{c|c} U_{[k]}^s & U_{[k]}^n \end{array}\right] \cdot D_{[k]} \cdot \left[\begin{array}{c|c} V_{[k]}^s & V_{[k]}^n \end{array}\right]^H,$$

where $U_{[k]} \in \mathbb{C}^{k \times M}$ and $V_{[k]} \in \mathbb{C}^{M \times M}$ are partitioned unitary matrices and $D_{[k]} \in \mathbb{R}^{M \times M}$ is a block-identity matrix

$$D_{[k]} = \left[\begin{array}{c|c} \sigma_{[k]}^s I_D & O \\ \hline O & \sigma_{[k]}^n I_{M-D} \end{array}\right]$$

containing averaged signal and noise singular values

$$\sigma_{[k]}^s = \frac{1}{D} \sum_{i=1}^{D} \sigma_{i,[k]}$$

$$\sigma_{[k]}^n = \frac{1}{M-D} \sum_{i=D+1}^{M} \sigma_{i,[k]}.$$

A subspace for which the corresponding singular values are averaged is called a spherical subspace. Any orthogonal basis can be chosen as a basis of singular vectors. The ROSA algorithm exploits this freedom to formulate a non-iterative subspace update. As shown in Algorithm 1, the signal subspace matrix $V_{[k-1]}^s$ is updated by a sequence of $D$ Givens rotations $\Phi_{[k]}^{i|i+1}$. Such a Givens rotation matrix $Q_{[k]}^{i|j}$ is an embedded complex $2 \times 2$ rotation matrix, differing from the identity matrix at four entries $Q_{[k]}^{i|j}(i,i) = Q_{[k]}^{i|j}(j,j) = c$ and $Q_{[k]}^{i|j}(i,j) = -Q_{[k]}^{i|j}{}^*(j,i) = s$ where $c^2 + s^* s = 1$ and $c \in \mathbb{R}, s \in \mathbb{C}$. The rotation angles are computed by gradually zeroing the projection of the new incoming data vector onto the signal and noise subspace. For more details on the computation of the rotation matrices we refer to [2]. A reduction in computation and in storage to $\mathcal{O}(MD)$ is achieved by only tracking the smaller of the two matrices $V_{[k]}^s$ or $V_{[k]}^n$ (Here we assume $D < M - D$.).

This work was supported in part by the ESPRIT BRA 6632 project of the EU. Filiep Vanpoucke is a research assistant with the N.F.W.O. Marc Moonen is a research associate with the N.F.W.O. (Belgian National Fund for Scientific Research). e-mail: {filiep.vanpoucke,marc.moonen}@esat.kuleuven.ac.be

0-7803-2431-5/95 $4.00 

**Algorithm 1**

$$V^s_{[0]} \leftarrow \left[\frac{I_D}{O_{M-D\times D}}\right]$$
$$\sigma^s_{[0]}, \sigma^n_{[0]} \leftarrow 0$$

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

1. projected signal vector

$$\tilde{x}^{s^H}_{[k]} \leftarrow x^H_{[k]} \cdot V^s_{[k-1]}$$

2. projected noise vector

$$x^n_{[k]} \leftarrow x_{[k]} - V^s_{[k-1]} \cdot \tilde{x}^s_{[k]}$$
$$\gamma^n_{[k]} \leftarrow \|x^n_{[k]}\|$$
$$v^n_{[k]} \leftarrow x^n_{[k]}/\gamma^n_{[k]}$$

3. column rotations

$$\left[\begin{array}{c|c} 0\cdots 0 & \gamma^s_{[k]} \end{array}\right] \leftarrow \tilde{x}^{s^H}_{[k]} \cdot \prod_{i=1}^{D-1} \Phi^{i|i+1}_{[k]}$$

$$\left[\begin{array}{cc} \tilde{\sigma}^s_{[k]} & 0 \\ 0 & \tilde{\sigma}^n_{[k]} \\ 0 & 0 \end{array}\right] \leftarrow \Theta^H_{[k]} \cdot \left[\begin{array}{cc} \alpha\sigma^s_{[k-1]} & 0 \\ 0 & \alpha\sigma^n_{[k-1]} \\ \gamma^s_{[k]} & \gamma^n_{[k]} \end{array}\right] \cdot \Phi^{D|D+1}_{[k]}$$

$$\left[\begin{array}{c|c} V^s_{[k]} & \star \end{array}\right] \leftarrow \left[\begin{array}{c|c} V^s_{[k-1]} & v^n_{[k]} \end{array}\right] \cdot \prod_{i=1}^{D} \Phi^{i|i+1}_{[k]}$$

4. re-averaging the singular values

$$\sigma^s_{[k]} \leftarrow \sqrt{\frac{\tilde{\sigma}^{s^2}_{[k]} + (D-1)\alpha^2\sigma^{s^2}_{[k-1]}}{D}}$$
$$\sigma^n_{[k]} \leftarrow \sqrt{\frac{\tilde{\sigma}^{n^2}_{[k]} + (M-D-1)\alpha^2\sigma^{n^2}_{[k-1]}}{M-D}}$$

endfor

## 2. FACTORED ORTHOGONAL MATRICES

The spherical SVD updating algorithm has two major disadvantages. First subspace tracking is typically part of the front-end processing of a real-time system. If data rates are high, then real-time execution may require a computing power, which can not be delivered by a single processor. Parallel computing is then the natural solution. To obtain a good speed-up, optimal mappings from algorithm to parallel architecture must be investigated. Unfortunately, the spherical SVD updating algorithm has a complicated data flow, mainly due to the second matrix-vector product $V^s_{[k-1]} \cdot \tilde{x}^s_{[k]}$ and the normalization of $\tilde{x}^n_{[k]}$. This makes it hard to pipeline the algorithm efficiently.

Secondly, the orthogonal matrix $V^s_{[k]} \in \mathbb{C}^{M\times D}$, containing the subspace estimate, is continuously updated by sequences of $2 \times 2$ rotations. Rounding errors in these multiplications cause the matrix $V^s_{[k]}$ to drift away from orthogonality, eventually leading to numerical instability. Current stabilization methods are based on periodic re-orthogonalization, keeping the deviation from orthogonality bounded. They are costly and further destroy the regularity of the data flow, thereby complicating parallel implementation.

In [3] we studied the same problems for a structurally related algorithm, *i.e.*, the Jacobi SVD updating algorithm [4]. Both problems were addressed by a minimal parameterization of the full square matrix of short singular vectors $V_{[k]} \in \mathbb{C}^{M\times M}$. It is given by the following factorization lemma.

> *Let $V$ be a unitary matrix ($V^H \cdot V = I_M$). Then $V$ can be factored uniquely into a product of $M \cdot (M-1)/2$ complex Givens rotations $Q^{i|j}$,* i.e.,
>
> $$V = \prod_{i=1}^{M-1} \prod_{j=i+1}^{M} Q^{i|j}.$$

The proof is constructive and is based on the Givens method for QR decomposition [5].

Here we propose to use the same factorization for the rectangular matrix $V^s_{[k-1]} \in \mathbb{C}^{M\times D}$. Since the Givens method can be applied column-wise to $V$, it suffices to truncate the process after column $D$ to obtain a factorization for its first $D$ columns.

$$V^s_{[k]} = \prod_{i=1}^{D} \prod_{j=i+1}^{M} Q^{i|j}_{[k]}.$$

This factorization ensures the orthogonality of the matrix $V^s_{[k]}$ in each iteration. Therefore, numerical error accumulation is avoided. It remains to be shown how to formulate all operations involving $V^s_{[k]}$ in terms of the rotation angles.

In the spherical SVD updating algorithm, the first operation is the matrix-vector multiplication $\tilde{x}^{s^H}_{[k]} = x^H_{[k]} \cdot V^s_{[k-1]}$. In factored form the inner product operations are replaced by Givens rotations

$$\tilde{x}^{s^H}_{[k]} = x^H_{[k]} \cdot \prod_{i=1}^{D} \prod_{j=i+1}^{M} Q^{i|j}_{[k-1]}. \qquad (1)$$

The signal flow graph (SFG) of the factored spherical SVD updating algorithm is shown in Figure 1. The black dots in the SFG represent delay operators. The full functionality of all nodes is given in Figure 2.

Eq. (1) is implemented by nodes A1 and B1. They perform a Givens rotation on their input pair and propagate the output pair to the right and downwards.

The second operation involving $V^s_{[k]}$ is the updating of $V^s_{[k-1]}$

$$\left[\begin{array}{c|c} V^s_{[k]} & \star \end{array}\right] \leftarrow \left[\begin{array}{c|c} V^s_{[k-1]} & v^n_{[k]} \end{array}\right] \cdot \prod_{i=1}^{D} \Phi^{i|i+1}_{[k]}.$$

The Givens rotations $\Phi^{i|i+1}_{[k]}$ are computed in the bottom nodes (D and E) and then propagated upwards to be worked into the factorization of $V_{[k-1]}$. As shown in [3], the updating can be done directly in terms of the rotation angles by two types of transformations, depending on the indices of the interacting rotations $Q^{k|l}$ and $\Phi^{i|i+1}$.

1. If $(k,l)$ and $(i,i+1)$ are equal, then the rotation angles of $Q^{i|i+1}$ and $\Phi^{i|i+1}$ must be added together (nodes A1).

$$Q^{i|i+1}_* = Q^{i|i+1} \cdot \Phi^{i|i+1}.$$

2. If $(k,l)$ and $(i,i+1)$ share one common index, then re-ordering of three Givens rotations is necessary (nodes B1).

$$\Phi^{i|i+1}_* \cdot Q^{i|l}_* \cdot Q^{i+1|l}_* = Q^{i|l} \cdot Q^{i+1|l} \cdot \Phi^{i|i+1}.$$

The product of the right-hand matrices is computed and re-factored in a different order, bringing the $(i,i+1)$-transformation in front. The cost of this operation is 10 Givens rotations.

The last operation involving $V^s_{[k]}$ is computing the parameterization of $x^n_{[k]} = V^n_{[k-1]} \cdot V^{n\,H}_{[k-1]} \cdot x_{[k]}$. In Algorithm 1 this is done by

$$x^n_{[k]} = (I - V^s_{[k-1]} \cdot V^{s\,H}_{[k-1]}) \cdot x_{[k]}.$$

The data flow of this double matrix-vector product contains a long bidirectional vertical dependency path. Here a more elegant computation is possible. The $D+1$st column (nodes C) of the trapezoidal grid should compute the parameterization of the vector $x^n_{[k]}$ after normalization. If this holds, the last $M-D-1$ components of $V^{n\,H}_{[k-1]} \cdot x_{[k]}$ are zero. This can only be true if the outputs at the right hand side of the rotation nodes in the $D+1$st column are zero. Therefore, it suffices to operate these nodes of type C in angle accumulation mode instead of rotation mode, similar to the nodes of type D and E. An additional advantage is the fact that normalization is implicit. The output of the $D+1$st column is automatically the required norm $||x^n_{[k]}||$.

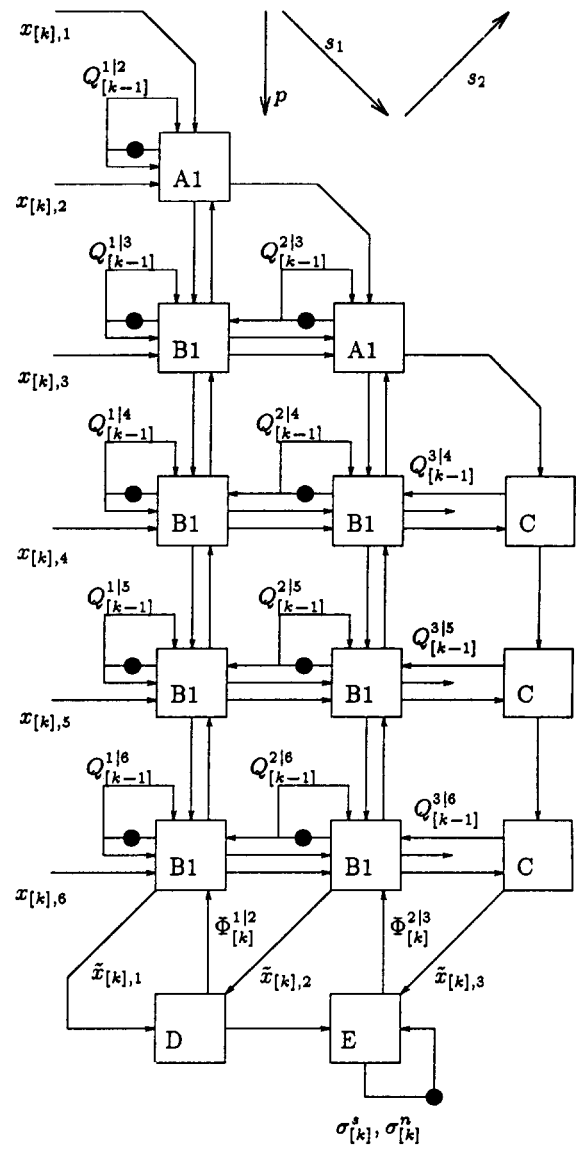


Figure 1: Signal flow graph of the factored spherical subspace algorithm for $M = 6, D = 2$

## 3. A LINEAR ARRAY

A natural way to assign computation to processors, is map all nodes of one column to a single processor. The corresponding placement vector $p$ is shown in Figure 1. The linear array then consists of $D+1$ processors.

Because of the bidirectional vertical data flow, the scheduling (*i.e.*, allocation of the computation over time) can not be done with a single scheduling vector.

For each time iteration the algorithm consists of three phases. In the first two phases the orthogonal matrix-vector product $\tilde{x}_{[k]}$ and the rotations $\Phi^{i|i+1}_{[k]}$ are

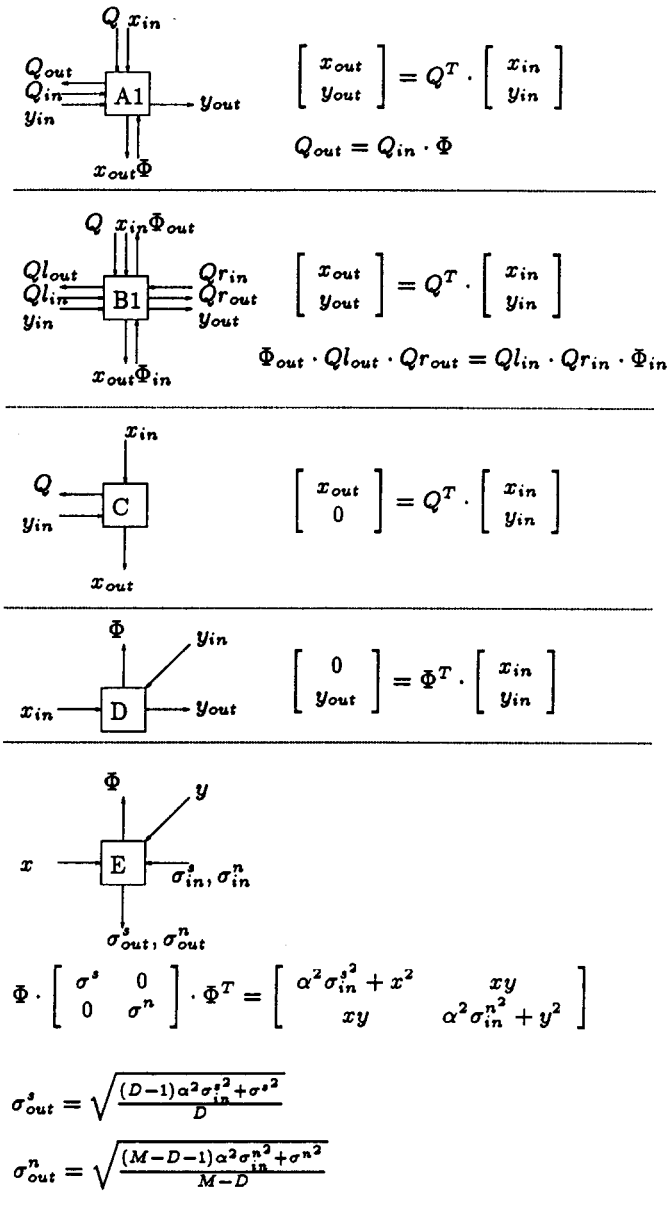


Figure 2: Functionalities of the nodes

generated. Here the data flow evolves from top to bottom and from left to right. In order to respect the ordering imposed by the dependencies, the scheduling vector $s_1$ of Figure 1 is selected. In the third phase the factorization of $V^s_{[k]}$ is updated. This updating process has dependencies pointing upwards and to the right. Therefore, the second scheduling vector $s_2$ points from the bottom-left corner to the diagonal. The linear array with this piecewise linear systolic schedule is efficient since it attains a processor utilization of almost 100 %. Its pipelining period is $2M-1$ cycles.

## 4. CONCLUSION

We have presented a factored approach to spherical subspace tracking. The orthogonal matrix spanning the estimated subspace is stored as a sequence of rotations. A first advantage is that the matrix is always kept orthogonal. There is no need for reorthogonalization and the algorithm is numerically stable. A second advantage is the ease of parallel implementation. The data flow is regular and homogeneous. Moreover, the algorithm consists solely of $2 \times 2$ rotation operations. Therefore, the ideal candidate for an ASIC implementation is a CORDIC processor.

Finally, we have presented a linear array with a systolic schedule. However, a planar array can also be derived. Unfortunately, in order to break the long vertical dependency path of length $\mathcal{O}(2M)$, alterations to the algorithm itself are necessary. This method of 'algorithmic transformations' has been successful in efficiently mapping other algorithms onto systolic arrays as well, *e.g.*, the recursive inverse least squares algorithm [6].

## 5. REFERENCES

[1] R. DeGroat, "Noniterative subspace tracking," *IEEE Trans. on SP*, vol. 40, pp. 571–577, Mar. 1992.

[2] R. DeGroat and R. Roberts, "A family of rank-one subspace updating methods," in *SVD and Signal Processing: Algorithms, Applications and Architectures* (E. Deprettere, ed.), pp. 277–300, North-Holland, 1988.

[3] F. Vanpoucke, M. Moonen, and E. Deprettere, "A numerically stable Jacobi array for parallel SVD updating," in *Advanced Signal Processing Algorithms, Architectures and Implementations V, Proc. of SPIE (To appear)* (F. Luk, ed.), (San Diego, USA), p. 10, July 1994.

[4] M. Moonen, P. Van Dooren, and J. Vandewalle, "A systolic array for SVD updating," *SIAM J. Matrix Anal. Appl.*, vol. 14, pp. 353–371, Apr. 1993.

[5] G. Golub and C. V. Loan, *Matrix Computations*. John Hopkins, 2nd ed., 1989.

[6] M. Moonen, I. Proudler, J. McWhirter, and G. Hekstra, "On the formal derivation of a systolic array for recursive least squares estimation," in *Proc. ICASSP*, vol. 2, pp. 477–480, Apr. 1994.