Simultaneous subspace tracking and rank estimation

Aleksandar Kavčić

Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213

Bin Yang

Department of Electrical Engineering
Ruhr University Bochum, 44780 Bochum, Germany

ABSTRACT

Jacobi-type singular value decomposition (SVD) is an exceptionally suitable method for recursive SVD updating. It has a low computational complexity per update, $O(n^2)$, where n is the problem size. Due to its parallel structure, it is very attractive for real-time applications on systolic arrays. In frequency and angle of arrival tracking problems, SVD can be used to track the signal subspace. Typically, only a few, r, dominant eigencomponents need to be tracked, where $r<n$. In this paper we show how to modify the Jacobi-type SVD to track only the r-dimensional signal subspace by forcing the $(n-r)$-dimensional noise subspace to be spherical. Thereby, the computational complexity is brought down from $O(n^2)$ to $O(nr)$. Furthermore, we show how to exploit the structure of the Jacobi-type SVD to estimate the signal subspace dimension, r, in addition to tracking the subspace itself. Most available computationally efficient subspace tracking algorithms rely on off-line estimation of the signal subspace dimension. This acts as a bottleneck in real-time parallel implementations. The Jacobi-type spherical subspace tracking algorithm presented in this paper is thus a subspace tracking method of computational complexity $O(nr)$, capable of tracking both the signal subspace and its dimension simultaneously. Thereby, the parallelism of the algorithm is not destroyed, as demonstrated in a systolic array implementation. Simulation results are presented to show the applicability of the algorithm in adaptive frequency tracking problems.

Keywords: eigenvalue decomposition, singular value decomposition, QR-decomposition, Jacobi rotation, systolic array, Jacobi-type singular value decomposition, subspace tracking, rank estimation

1 INTRODUCTION

Subspace tracking plays an important role in high resolution temporal and spatial domain spectral analysis algorithms such as MUSIC,[1] and minimum-norm.[2] Typically, in direction of arrival (DOA) or frequency retrieval applications, snapshots of a sensor array or time series data are collected to form a sample correlation matrix, whose eigenvalue decomposition (ED) reveals the subspace structure of the received signal. The obtained eigenvalues are estimates of the power in the orthogonal modes, and can be used to estimate the number of dominant components in the signal. The eigenvectors corresponding to the dominant eigenvalues span the signal subspace and are directly involved in estimating DOAs or frequencies. For better numerical stability, instead of ED of a sample correlation matrix, singular value decomposition (SVD) of a data matrix can be performed, where the singular values are square roots of the corresponding eigenvalues, and the right hand side singular vectors equal the eigenvectors.

Performing batch ED or SVD after every new snapshot is a very time consuming task since both ED and SVD are $O(n^3)$-complexity operations, where n is the dimension of the correlation matrix. Adaptive subspace tracking algorithms represent therefore an attractive alternative. These algorithms span a variety of solution techniques. For example, modifications of some classical ED/SVD methods have found their use in adaptive subspace tracking.[3-6] Bunch's rank-one updating algorithm[7] and its subspace averaging simplifications[8,9] represent another approach. Adaptive subspace tracking can also be formulated as a constrained or unconstrained optimization problem as in the gradient based algorithms[10] and other optimization techniques.[11,12] Yet another approach is demonstrated by Stewart's URV decomposition.[13]

Besides their subspace tracking capabilities, adaptive subspace tracking algorithms can have other preferable characteristics. Low computational complexity is certainly an advantage. Typically, efficient algorithms would track only r dominant eigencomponents ($r < n$), and the fastest such algorithms[5,8,9,14] require only $O(nr)$ operations per update. The second preferred characteristic is the parallel structure of an algorithm so that it can be performed on a systolic array. Most available algorithms can be performed on an array of $O(n)$ processors with a throughput of $O(n^{-1})$. The most efficient array of $O(n^2)$ processor cells and $O(n^0)$ throughput was constructed for the Jacobi-type adaptive SVD by Moonen et. al.[4,15] The third desirable characteristic for the algorithm is to be rank-revealing. That is, the algorithm should be able to adaptively determine the subspace dimension r, i.e. the number of dominant eigencomponents. Most algorithms rely on external sources to supply the signal subspace dimension r to them. Examples of self-relying dimension tracking algorithms are Stewart's URV decomposition,[13] Yang and Gersemsky's[14] modification of the PAST algorithm,[12] and the sphericalized SVD updating proposed by Dowling et. al.[9] While most algorithms have at least one preferable characteristic, it is a real challenge to combine low computational complexity, parallel structure and rank-revealing character into one algorithm. To our knowledge thus far, only the sphericalized SVD updating[9] combines all three characteristics in one algorithm.

In this paper we present a different subspace tracking algorithm that is parallel, rank revealing and requires only $O(nr)$ operations per update. The algorithm is the noise averaged SVD (NASVD)[5] and it is derived from the Jacobi-type SVD updating algorithm proposed by Moonen et. al.[4] Computational savings are achieved by averaging the noise singular values, i.e. by making the noise subspace spherical. Thereby the structure of the algorithm is exploited to track the rank r by a simple threshold comparison technique that doesn't destroy the parallelism of the subspace tracking method. The paper is organized as follows. In section 2 we briefly review the Jacobi-type NASVD algorithm. We develop a parallel dimension tracking strategy in section 3. A systolic array is presented in section 4, while the applicability of the algorithm in adaptive frequency tracking problems is demonstrated in section 5.

We use the following notation throughout the paper. Matrices and vectors are represented by boldface and underlined characters, respectively. Superscripts ^{H}, ^{T} and * are used to denote Hermitian transposition, matrix transposition and complex conjugation. $\|\cdot\|$ represents the 2-norm of matrices and vectors. The (i, j)-th element of a matrix $\mathbf{A}$ is denoted by $\mathrm{A}\{i, j\}$.

2 JACOBI-TYPE NASVD

The Jacobi-type NASVD algorithm[5] is a noise subspace averaged version of the full Jacobi-type SVD updating.[4] Rather than deriving the algorithm, we briefly outline it for future reference when we present the dimension tracking technique. We suppose that $\underline{x}_k$ is an $n \times 1$ dimensional data vector (snapshot of a linear sensor array or a time series data vector) obtained at the time instant k. Supposing that the data matrix at the time instant $k{-}1$ is decomposed as

$$\mathbf{A}_{k-1} = \mathbf{U}_{k-1} \begin{bmatrix} \mathbf{R}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathrm{R}_{k-1}\{r+1, r+1\} \cdot \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{V}^{\mathrm{S}}_{k-1} & \mathbf{V}^{\mathrm{N}}_{k-1} \end{bmatrix}^{\mathrm{H}}, \tag{1}$$

we wish to find a similar decomposition of the updated data matrix

$$\mathbf{A}_k = \begin{bmatrix} \sqrt{\beta}\mathbf{A}_{k-1} \\ \underline{x}_k^{\mathrm{H}} \end{bmatrix}. \tag{2}$$

Thereby $\mathbf{R}_{k-1}$ is a $(r{+}1){\times}(r{+}1)$ upper triangular, almost diagonal matrix. This means that the norm of its off-diagonal elements is much smaller than that of the diagonal entries. Its first r diagonal entries are good approximations of the r signal singular values. The last diagonal entry $\mathrm{R}_{k-1}\{r{+}1, r{+}1\}$ is the approximate averaged noise singular value. Notice that the same value is found in the $n{-}r{-}1$ diagonal entries of the matrix $\mathrm{R}_{k-1}\{r{+}1, r{+}1\} \cdot \mathbf{I}$ in (1). $\mathbf{V}^{\mathrm{S}}_{k-1}$ is the $n{\times}r$ dimensional matrix of orthonormal vectors that represent good approximations of the right signal singular vectors of $\mathbf{A}_{k-1}$. Similarly, $\mathbf{V}^{\mathrm{N}}_{k-1}$ is the approximate $n{\times}(n{-}r)$ dimensional matrix of the right noise singular vectors. The vectors of $\mathbf{V}^{\mathrm{N}}_{k-1}$ are orthonormal as well and they are also orthogonal to the signal singular vectors of $\mathbf{V}^{\mathrm{S}}_{k-1}$. We say that the columns of $\mathbf{V}^{\mathrm{S}}_{k-1}$ span the signal subspace, while the columns of $\mathbf{V}^{\mathrm{N}}_{k-1}$ span the noise subspace. Notice that the noise subspace is spherical since the noise singular values corresponding to the vectors of $\mathbf{V}^{\mathrm{N}}_{k-1}$ all equal $\mathrm{R}_{k-1}\{r{+}1, r{+}1\}$. $\mathbf{U}_{k-1}$ is the approximate matrix of left singular vectors that we are not interested in tracking. Clearly, equation (1) is an approximate SVD of $\mathbf{A}_{k-1}$. The constant β ($0{<}\beta{<}\ 1$) is the forgetting factor and it provides downweighting of past data samples to allow tracking nonstationarities in the signal.

In subspace tracking applications we are interested in tracking only the $n{\times}r$ dimensional matrix of right signal singular vectors $\mathbf{V}^{\mathrm{S}}_k$ because its columns span the signal subspace. However, having in mind the DOA or frequency retrieval applications, tracking the $(r{+}1){\times}(r{+}1)$ upper triangular matrix $\mathbf{R}_k$ is also beneficial. Squares of the diagonal entries of $\mathbf{R}_k$ are good estimates of the power in the signal components. Also, $\mathrm{R}^2_{k-1}\{r{+}1, r{+}1\}$ is an estimate of the noise power in the system. Furthermore, the diagonal entries of $\mathbf{R}_k$ can be used to adaptively estimate the dimension r of the signal subspace as we will show in the next section.

The Jacobi-type NASVD algorithm updates $\mathbf{R}_k$ and $\mathbf{V}^{\mathrm{S}}_k$ in the following manner. We first compute the vector of inner products $\underline{z}^{\mathrm{S}}_k{=}\left(\mathbf{V}^{\mathrm{S}}_{k-1}\right)^{\mathrm{H}} \underline{x}_k$ and form the vector

$$\underline{z}_k = \begin{bmatrix} \underline{z}^{\mathrm{S}}_k \\ z^{\mathrm{N}}_k \end{bmatrix} = \begin{bmatrix} \underline{z}^{\mathrm{S}}_k \\ \sqrt{\|\underline{x}_k\|^2 - \|\underline{z}^{\mathrm{S}}_k\|^2} \end{bmatrix}. \tag{3}$$

Using QR-decomposition

$$\begin{bmatrix} \sqrt{\beta}\mathbf{R}_{k-1} \\ \underline{z}^{\mathrm{H}}_k \end{bmatrix} = \mathbf{Q}_k \begin{bmatrix} \mathbf{R}'_k \\ \underline{0}^{\mathrm{T}} \end{bmatrix}, \tag{4}$$

we obtain the upper triangular matrix $\mathbf{R}'_k$. We further diagonalize $\mathbf{R}'_k$ with a series of r Jacobi rotations to get

$$\mathbf{R}_k = \mathbf{\Theta}_{(k,r)} \cdots \mathbf{\Theta}_{(k,1)} \mathbf{R}'_k \mathbf{\Phi}_{(k,1)} \cdots \mathbf{\Phi}_{(k,r)}. \tag{5}$$

The complex Jacobi rotation pair[16] $\mathbf{\Theta}_{(k,i)}$ and $\mathbf{\Phi}_{(k,i)}$ is applied on the i-th and $(i{+}1)$-st row and column of $\mathbf{R}'_k$ to introduce a zero in the $(i, i{+}1)$-st entry of $\mathbf{R}'_k$, just as in the original full Jacobi-type SVD updating algorithm.[4] However, while in the original Jacobi-type SVD strictly outer rotations are used for better convergence properties,[4,17,18] in the NASVD algorithm only the first $r{-}1$ rotations are outer rotations. To see why the last rotation is an inner one, notice that due to their "mixing" properties,[17–19] outer rotations cyclicly permute the singular values along the diagonal entries of $\mathbf{R}_k$. Since we want the last diagonal entry of $\mathbf{R}_k$ to be always occupied by the averaged noise singular value, we choose the r-th Jacobi rotation to be an inner one. The matrix $\mathbf{R}_k$ has a smaller off-diagonal norm than $\mathbf{R}'_k$ due to the applied Jacobi rotations. Thus $\mathbf{R}_k$ is an upper triangular, almost diagonal matrix. Of course, we can further reduce the off-diagonal norm of $\mathbf{R}_k$ by applying another sweep of Jacobi rotations to $\mathbf{R}_k$ in the same manner as described above. We will, however, use only one sweep per update in our NASVD subspace tracking algorithm.

After the Jacobi rotations have been applied, the sphericity of the noise subspace is destroyed. That is, the newly obtained noise singular value $\mathrm{R}_k\{r{+}1, r{+}1\}$ no longer equals $\sqrt{\beta}\mathrm{R}_{k-1}\{r{+}1, r{+}1\}$, which is the value of the remaining $n{-}r{-}1$ noise singular values. To force sphericity on the noise subspace, we need to reaverage the squares of the noise singular values, i.e. noise eigenvalues of a corresponding sample correlation matrix. Thus we redefine the averaged noise singular value

$$\mathrm{R}_k\{r+1, r+1\} := \sqrt{\frac{\mathrm{R}_k^2\{r+1, r+1\} + (n-r-1)\beta\mathrm{R}_{k-1}^2\{r+1, r+1\}}{n-r}}. \tag{6}$$

To update the matrix of right signal singular vectors $\mathbf{V}_k^{\mathrm{S}}$, we first form the normalized projection of $\underline{x}_k$ on the noise subspace

$$\underline{v}_{k-1}^{\mathrm{N}} = \frac{\underline{x}_k - \mathbf{V}_{k-1}^{\mathrm{S}}\underline{z}_k^{\mathrm{S}}}{z_k^{\mathrm{N}}}. \tag{7}$$

We refer to $\underline{v}_{k-1}^{\mathrm{N}}$ as the averaged noise singular vector. It is exactly this vector that interacts with the signal subspace in the updating process.[8] As a consequence, we update the matrix $\mathbf{V}_k^{\mathrm{S}}$ as

$$\left[\begin{array}{cc} \mathbf{V}_k^{\mathrm{S}} & \underline{\tilde{v}}_k^{\mathrm{N}} \end{array}\right] = \left[\begin{array}{cc} \mathbf{V}_{k-1}^{\mathrm{S}} & \underline{v}_{k-1}^{\mathrm{N}} \end{array}\right] \mathbf{\Phi}_{(k,1)} \cdots \mathbf{\Phi}_{(k,r)}. \tag{8}$$

Notice that for subspace tracking purposes alone, we don't need to calculate the vector $\underline{\tilde{v}}_k^{\mathrm{N}}$. However, as we will show in the next section, the vector $\underline{\tilde{v}}_k^{\mathrm{N}}$ plays an important role in our subspace dimension updating scheme.

With the updated matrices $\mathbf{R}_k$ and $\mathbf{V}_k^{\mathrm{S}}$, we execute a new iteration step of the NASVD updating process as soon as the new data vector $\underline{x}_{k+1}$ becomes available. As a final remark in this section we note that the algorithm converges for any diagonal starting matrix $\mathbf{R}_0$ whose diagonal entries are non-negative, and for any matrix $\mathbf{V}_0^{\mathrm{S}}$ with orthonormal columns.

3 THRESHOLD COMPARISON DIMENSION TRACKING

So far we assumed that the signal subspace dimension r is known a priori. In real subspace tracking applications this may not be the case, and we have to estimate r from the collected data. Information criteria like AIC and MDL are popular tools for dimension tracking.[20] These methods have also been adapted to work with sphericalized subspace tracking algorithms.[14] Following the method proposed by Yang and Gersemsky,[14] we could use the squares of the singular values obtained by the NASVD algorithm as estimates of the correlation matrix eigenvalues to form the AIC criterion and thus estimate the signal subspace dimension r. Although this approach leads to good dimension tracking results, it is not a well suited method for parallel execution on systolic arrays. This is because both AIC and MDL criteria are based on minimization of a global function whose arguments are squares of the estimated singular values of $\mathbf{A}_k$. These arguments are obtained from the diagonal entries of $\mathbf{R}_k$, which are located in non-adjacent positions in the matrix and are therefore difficult to operate on in a systolic fashion. Furthermore, to find the minima of the AIC or the MDL criteria, we would have to go through several iterations per subspace update. This is a task that serves as a bottleneck in parallel applications since the subspace tracking algorithm must wait for the rank r to be estimated before it can resume. To solve these problems, we suggest a simple threshold comparison criterion.

In order to successfully track the signal subspace dimension r, we introduce some changes in the NASVD algorithm. To see why the changes are necessary, suppose that the rank r is suddenly increased by one. Accordingly, there will be an extra signal singular value to be tracked. But since this extra singular value (the last diagonal entry of $\mathbf{R}_k$) is treated as a noise singular value and therefore reaveraged in every update (6), it cannot follow the sudden change in rank. To solve this problem, we propose to track an extra noise singular value and an extra noise singular vector. This extra noise singular value will not enter the reaveraging step (6) and will therefore be free to increase abruptly in case of rank increase. To allow for this change in the algorithm, the matrix $\mathbf{R}_k$ will have to be of size $(r+2)\times(r+2)$ and the matrix $\mathbf{V}_k^{\mathrm{S}}$ will have to be of size $n\times(r+1)$, where only those singular vectors corresponding to the first r singular values are the actual signal subspace basis vectors.

Suppose that we are given a threshold T, above which we know that all signal singular values must lie. Then we formulate the threshold comparison criterion as:

```
IF the lowest signal singular value is less than $T$
    decrease rank $r$ by one;
ELSEIF the non-averaged noise singular value exceeds $T$
    increase rank $r$ by one;
ELSE
    rank $r$ remains unchanged;
END
```

The threshold value T can be either user supplied or it can be adaptively estimated based on some prior knowledge about the tracked signals. The latter case is more useful if the noise level in the system is a time varying function. For example, we found out by experiment that for tracking signals with signal to noise ratios (SNRs) greater or equal to zero, the threshold value should be 50% to 100% larger than the averaged noise singular value.

In order to make the threshold comparison technique work in parallel with NASVD, we need to add a final touch to the algorithm by carefully choosing the type of Jacobi rotations we use. Our goal is to use as many outer Jacobi rotations as possible to enhance the convergence properties of the algorithm, yet we need to take care that the non-averaged and the averaged noise singular values stay locked in the $(r+1)$-st and $(r+2)$-nd diagonal entries of $\mathbf{R}_k$. We manage this by applying the Jacobi rotations in the following manner:

- Choose the first $r-1$ Jacobi rotations to be outer rotations.
- Choose the r-th Jacobi rotation to be an inner one, so that the signal singular values are kept at the first r diagonal entries of $\mathbf{R}_k$.
- Choose the $(r+1)$-st Jacobi rotation so that after the rotation is applied, $\mathrm{R}_k\{r+1, r+1\}$ is never smaller than $\mathrm{R}_k\{r+2, r+2\}$. Since $\mathrm{R}_k\{r+2, r+2\}$ will enter the reaveraging step, this prevents the larger of the two noise singular values to be reaveraged, thus making the algorithm sensitive to rank increase.

Notice that due to the "mixing" properties[17–19] of the $r-1$ outer Jacobi rotations, the r signal singular values rotate within the first r diagonal locations of $\mathbf{R}_k$. This movement is one position to the left in every update, while the singular value from the first diagonal position moves down to the r-th. Using this property, it is not necessary to search for the lowest signal singular value after every update in order to compare it to threshold T. We simply compare only the element $\mathrm{R}_k\{r, r\}$ to the threshold because we know that at least once in every r updates the lowest signal singular value will occupy that position. This reveals that we perform rank deflation only once in r updates, but at least we are sure that the algorithm will certainly detect rank deflation although it might be with a delay of up to r cycles. On the other hand, locating the non-averaged noise singular value for threshold comparison purposes is trivial since, by construction of the Jacobi rotations, it is always at the entry $\mathrm{R}_k\{r+1, r+1\}$. We summarize the extended Jacobi type NASVD algorithm together with the threshold comparison rank estimation:

Initialization

1. Input the initial threshold value $\mathrm{T}_0 \geq 0$, and a user supplied thresholding factor ϕ $(1.5 \leq \phi \leq 2)$.

```
    2. $r = 0$ , $\mathbf{R}_0 = \mathrm{T}_0 \cdot \mathbf{I}_{r+2}$ , $\mathbf{V}_0^{\mathrm{S}} = \begin{bmatrix} 1 \\ \underline{0} \end{bmatrix}$.
**Loop** for $k = 1, \ldots, \infty$
    1. Input $\underline{x}_k$.
    2. Calculate projection of $\underline{x}_k$ on $\mathbf{V}_{k-1}^{\mathrm{S}}$: $\underline{z}_k^{\mathrm{S}} = \left(\mathbf{V}_{k-1}^{\mathrm{S}}\right)^{\mathrm{H}} \underline{x}_k$.
    3. Calculate noise vector: $\underline{v}_{k-1}^{\mathrm{N}} = \dfrac{\underline{x}_k - \mathbf{V}_{k-1}^{\mathrm{S}} \underline{z}_k^{\mathrm{S}}}{\sqrt{\|\underline{x}_k\|^2 - \|\underline{z}_k^{\mathrm{S}}\|^2}}$.
    4. Form vector $\underline{z}_k^{\mathrm{H}} = \underline{x}_k^{\mathrm{H}} \left[ \mathbf{V}_{k-1}^{\mathrm{S}} \quad \underline{v}_{k-1}^{\mathrm{N}} \right] = \left[ \left(\underline{z}_k^{\mathrm{S}}\right)^{\mathrm{H}} \quad \sqrt{\|\underline{x}_k\|^2 - \|\underline{z}_k^{\mathrm{S}}\|^2} \right]$.
    5. QR-decomposition: $\begin{bmatrix} \sqrt{\beta}\mathbf{R}_{k-1} \\ \underline{z}_k^{\mathrm{H}} \end{bmatrix} = \mathbf{Q}_k \begin{bmatrix} \mathbf{R}_k' \\ \underline{0}^{\mathrm{H}} \end{bmatrix}$.
    6. Jacobi rotations: $\mathbf{R}_k = \mathbf{\Theta}_{(k,r+1)} \cdots \mathbf{\Theta}_{(k,1)} \mathbf{R}_k' \mathbf{\Phi}_{(k,1)} \cdots \mathbf{\Phi}_{(k,r+1)}$.
       The first $r-1$ Jacobi rotations are outer rotations. The $r$-th Jacobi rotation is an inner one, while the
       $(r+1)$-st rotation is chosen such that $\mathrm{R}_k\{r+1, r+1\} \geq \mathrm{R}_k\{r+2, r+2\}$.
    7. Signal subspace update:
       $\left[ \mathbf{V}_k^{\mathrm{S}} \quad \underline{\tilde{v}}_k^{\mathrm{N}} \right] = \left[ \mathbf{V}_{k-1}^{\mathrm{S}} \quad \underline{v}_{k-1}^{\mathrm{N}} \right] \mathbf{\Phi}_{(k,1)} \cdots \mathbf{\Phi}_{(k,r+1)}$.
    8. Reaverage to maintain noise subspace sphericity:
       $\mathrm{R}_k\{r+2, r+2\} := \sqrt{\dfrac{\mathrm{R}_k^2\{r+2, r+2\} + (n-r-2)\beta \mathrm{R}_{k-1}^2\{r+2, r+2\}}{n-r-1}}$.
    9. Dimension updating by threshold comparison
       **IF** $\mathrm{R}_k\{r, r\} < \mathrm{T}_{k-1}$ perform rank deflation:
          • Set $\mathrm{R}_k\{r+1, r+1\} := \sqrt{\dfrac{\mathrm{R}_k^2\{r+1, r+1\} + (n-r-1)\mathrm{R}_k^2\{r+2, r+2\}}{n-r}}$.
          • Delete the rightmost column of $\mathbf{V}_k^{\mathrm{S}}$ and $\mathbf{R}_k$.
          • Set $r := r - 1$.
       **ELSEIF** $\mathrm{R}_k\{r+1, r+1\} > \mathrm{T}_{k-1}$ perform rank inflation:
          • Append an extra column to the right of $\mathbf{V}_k^{\mathrm{S}}$ and $\mathbf{R}_k$ by setting:
               $\mathbf{V}_k^{\mathrm{S}} := \left[ \mathbf{V}_k^{\mathrm{S}} \quad \underline{\tilde{v}}_k^{\mathrm{N}} \right]$ and $\mathbf{R}_k := \begin{bmatrix} \mathbf{R}_k & \underline{0} \\ \underline{0}^{\mathrm{H}} & \mathrm{R}_k\{r+2, r+2\} \end{bmatrix}$.
          • Set $r := r + 1$.
       **ELSE** rank stays the same.
   10. Update the threshold value: $\mathrm{T}_k := \phi \mathrm{R}_k\{r+2, r+2\}$.
**end**
```

The determined rank will eventually converge to the correct one for any initial threshold value $T_0 \geq 0$. However, if the initial guess T_0 is good, the correct rank will be determined only in a few updates. Otherwise it might take a much longer time depending on how bad the initial guess is.

4 SYSTOLIC ARRAY

Since the Jacobi-type NASVD algorithm is similar to the TQR-based SVD, we exploit the systolic architecture proposed by Dowling et.al.[6] to implement the NASVD algorithm. We simply replace the TQR-SVD triangle array[6] with a triangular array that performs QR-decomposition and Jacobi rotations on the matrix $\mathbf{R}_k$. For this

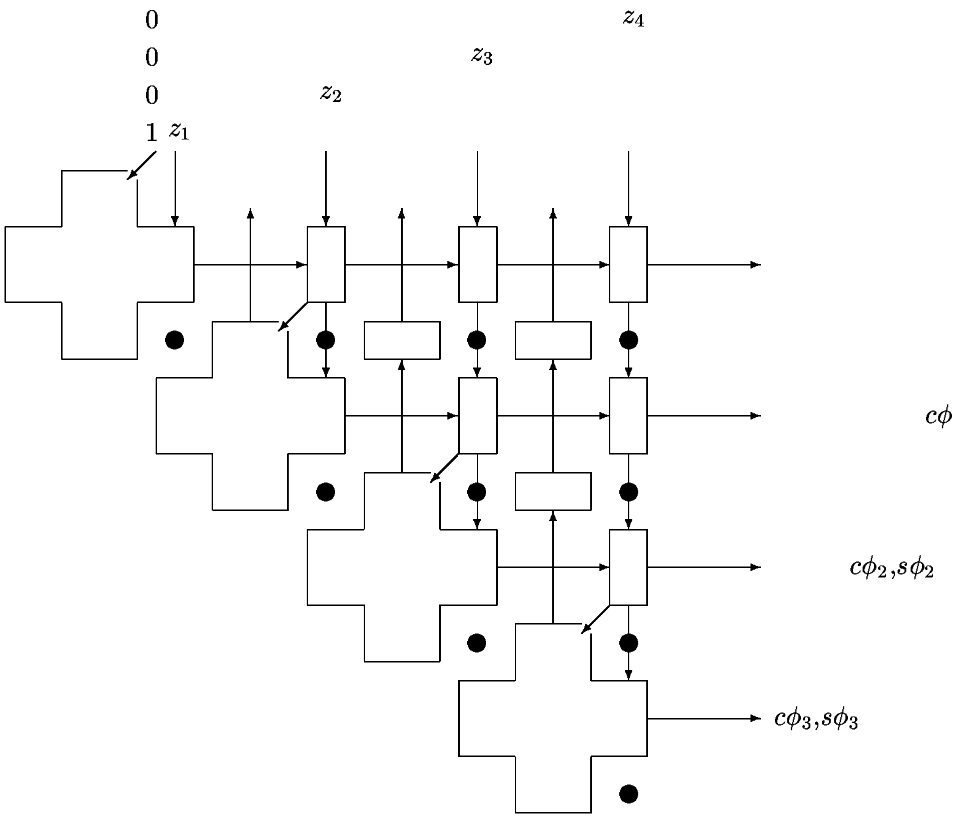
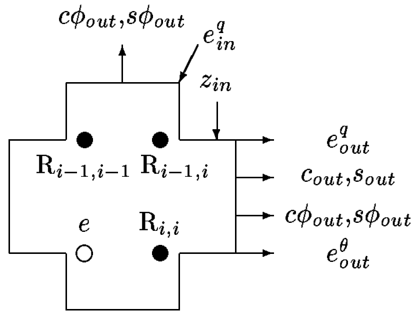


Figure 1: Triangular systolic array for performing QR-decomposition and Jacobi rotations on the matrix $\mathbf{R}_k$.

matter we modify the full Jacobi-type SVD array developed by Moonen et.al.[4,15] to fit the architecture proposed by Dowling et.al. Our intention is to demonstrate that the Jacobi-type NASVD updating algorithm can be implemented in systolic architecture. Therefore we will not discuss details such as timing and synchronization.

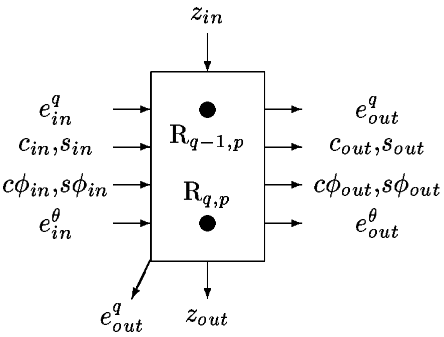
To keep the discussion simple, we assume at first that the rank r is known and is not allowed to change. Figure 1 shows what the triangular array looks like for r=2. The places denoted by the bullet (•) represent memory locations where the elements of the matrix $\mathbf{R}_k$ are stored. Each element of $\mathbf{R}_k$ is accessed by neighboring cells (solid lined boxes). Input to the array is the time-skewed vector $\underline{z}_k^{\mathrm{H}} = \left[\ (\underline{z}_k^{\mathrm{S}})^{\mathrm{H}}\ \ z_k^{\mathrm{N}}\ \right] = [\ z_1\ \ z_2\ \ z_3\ \ z_4\ ]$. The processor elements of the array (cells) update the matrix $\mathbf{R}_k$. The array produces as its output on its right hand side the parameters of the right hand side Jacobi rotations $\mathbf{\Phi}_{(k,1)}, \ldots, \mathbf{\Phi}_{(k,3)}$ in time-skewed fashion. The cell descriptions are given in Figures 2 to 4. Figure 5 shows the snapshots of the array. In each step, cells that perform operations on elements of $\mathbf{R}_k$ are outlined. Thereby, active cells performing QR-decomposition are represented by dashed boundaries, while active cells performing Jacobi rotations appear outlined with solid lines.

The so constructed Jacobi-type triangular array is substituted for the TQR-SVD triangular array in the architecture of Dowling et.al.[6] to form a complete subspace updating systolic array in Figure 6. For the beginning ignore the dashed lines in Figure 6. The vector $\underline{x}_k$ enters the linear array of cells (denoted by z_1, z_2, z_3 and z_4). These cells compute the inner product vector $\underline{z}_k^{\mathrm{H}} = \left[\ (\underline{z}_k^{\mathrm{S}})^{\mathrm{H}}\ \ z_k^{\mathrm{N}}\ \right] = [\ z_1\ \ z_2\ \ z_3\ \ z_4\ ]$. The values z_1, z_2, z_3 and z_4 are passed to the triangular array (labeled $\mathbf{R}_k$ in Figure 6) to calculate the parameters ($c\phi_i$, $s\phi_i$) of the 3 Jacobi rotations. In the meantime the singular vectors $[\ \mathbf{V}_k^{\mathrm{S}}\ \ \underline{v}_k^{\mathrm{N}}\ ] = [\ \underline{v}_1\ \ \underline{v}_2\ \ \underline{v}_3\ \ \underline{v}_k^{\mathrm{N}}\ ]$ come out of the shift registers (denoted by short stacked boxes) and enter the rotation cells (r_1, r_2, r_3). The rotation cells perform Jacobi rotations on the vectors $\underline{v}_1$, $\underline{v}_2$, $\underline{v}_3$ and $\underline{v}_k^{\mathrm{N}}$ to form the updated vectors $[\ \mathbf{V}_{k+1}^{\mathrm{S}}\ \ \tilde{\underline{v}}_k^{\mathrm{N}}\ ] = [\ \tilde{\underline{v}}_1\ \ \tilde{\underline{v}}_2\ \ \tilde{\underline{v}}_3\ \ \tilde{\underline{v}}_k^{\mathrm{N}}\ ]$,



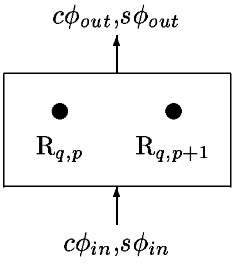
```
IF $e^q_{in} = 1$
    • $R_{i,i} \leftarrow \sqrt{\beta} R_{i,i}$
    • Determine QR-decomposition parameters $cq$ and $sq$
    • $\begin{bmatrix} R_{i,i} \\ 0 \end{bmatrix} \leftarrow \begin{bmatrix} cq & -sq^* \\ sq & cq^* \end{bmatrix} \begin{bmatrix} R_{i,i} \\ z_{in} \end{bmatrix}$
    • $e \leftarrow 1$
    • Propagate parameters
        $e^q_{out} \leftarrow 1$, $c_{out} \leftarrow cq$, $s_{out} \leftarrow sq$, $e^\theta_{out} \leftarrow 0$
(continued in next column)
(continued from previous column)
ELSEIF $e = 1$
    • Determine parameters $c\theta$, $s\theta$, $c\phi$, $s\phi$ of the appropriate Jacobi rotation
    • if $i = r + 2$ then save $R_{i,i}$ for reaveraging
        $e \leftarrow R_{i,i}$
    • $\begin{bmatrix} R_{i-1,1-1} & 0 \\ 0 & R_{i,i} \end{bmatrix} \leftarrow \begin{bmatrix} c\theta & -s\theta^* \\ s\theta & c\theta^* \end{bmatrix} \times$
        $\times \begin{bmatrix} R_{i-1,1-1} & R_{i-1,i} \\ 0 & R_{i,i} \end{bmatrix} \begin{bmatrix} c\phi & -s\phi^* \\ s\phi & c\phi^* \end{bmatrix}$
    • if $i = r + 2$ then reaverage
        $R_{i,i} \leftarrow \sqrt{\frac{R^2_{i,i} + (n - r - 2) \cdot e^2}{n - r - 1}}$
    • $e \leftarrow 0$
    • Propagate parameters
        $e^q_{out} \leftarrow 0$, $c_{out} \leftarrow c\theta$, $s_{out} \leftarrow s\theta$,
        $c\phi_{out} \leftarrow c\phi$, $s\phi_{out} \leftarrow s\phi$, $e^\theta_{out} \leftarrow 1$
END
```

Figure 2: Diagonal cell description



```
IF $e^q_{in} = 1$
    • $R_{q,p} \leftarrow \sqrt{\beta} R_{q,p}$
    • $\begin{bmatrix} R_{q,p} \\ z_{out} \end{bmatrix} \leftarrow \begin{bmatrix} c_{in} & -s^*_{in} \\ s_{in} & c^*_{in} \end{bmatrix} \begin{bmatrix} R_{q,p} \\ z_{in} \end{bmatrix}$
ELSEIF $e^\theta_{in} = 1$
    • $\begin{bmatrix} R_{q-1,p} \\ R_{q,p} \end{bmatrix} \leftarrow \begin{bmatrix} c_{in} & -s^*_{in} \\ s_{in} & c^*_{in} \end{bmatrix} \begin{bmatrix} R_{q-1,p} \\ R_{q,p} \end{bmatrix}$
END
Propagate parameters
    • $e^q_{out} \leftarrow e^q_{in}$, $c_{out} \leftarrow c_{in}$, $s_{out} \leftarrow s_{in}$,
      $c\phi_{out} \leftarrow c\phi_{in}$, $s\phi_{out} \leftarrow s\phi_{in}$, $e^\theta_{out} \leftarrow e^\theta_{in}$
```

Figure 3: Rightward propagation cell



```
$\begin{bmatrix} R_{q,p} & R_{q,p+1} \end{bmatrix} \leftarrow \begin{bmatrix} R_{q,p} & R_{q,p+1} \end{bmatrix} \begin{bmatrix} c\phi_{in} & -s\phi^*_{in} \\ s\phi_{in} & c\phi^*_{in} \end{bmatrix}$
Propagate parameters
    • $c\phi_{out} \leftarrow c\phi_{in}$, $s\phi_{out} \leftarrow s\phi_{in}$
```

Figure 4: Upward propagation cell

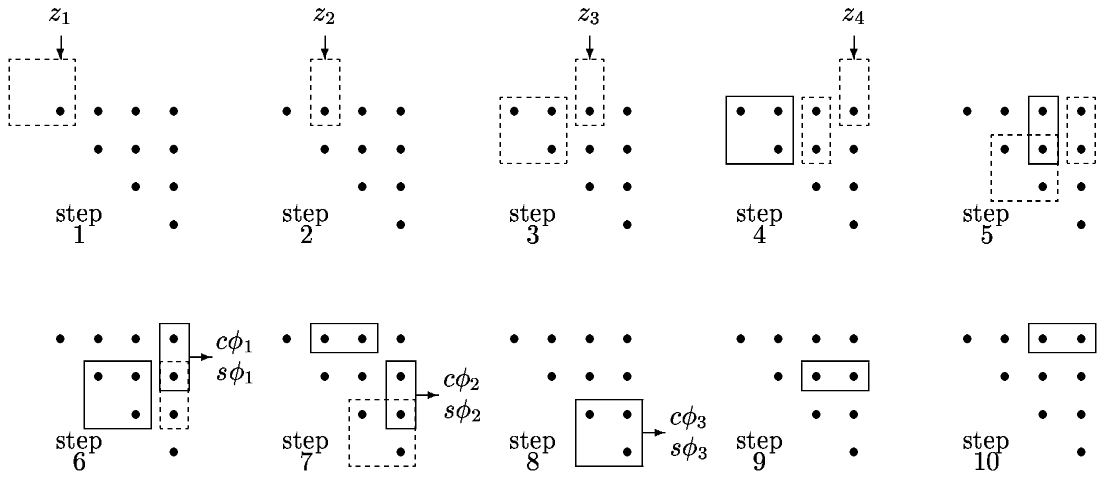


Figure 5: Snapshots of the triangular Jacobi-type NASVD array

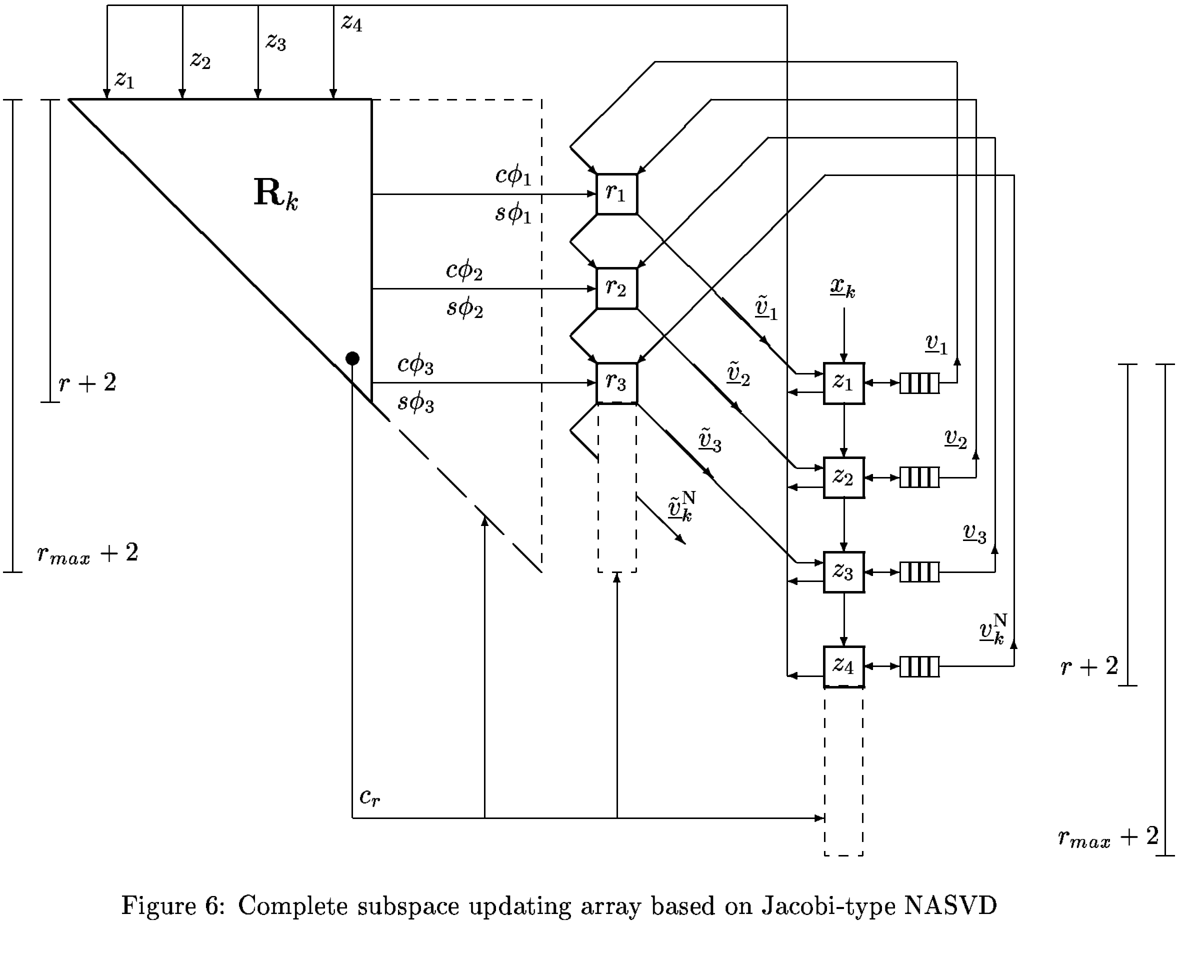


Figure 6: Complete subspace updating array based on Jacobi-type NASVD

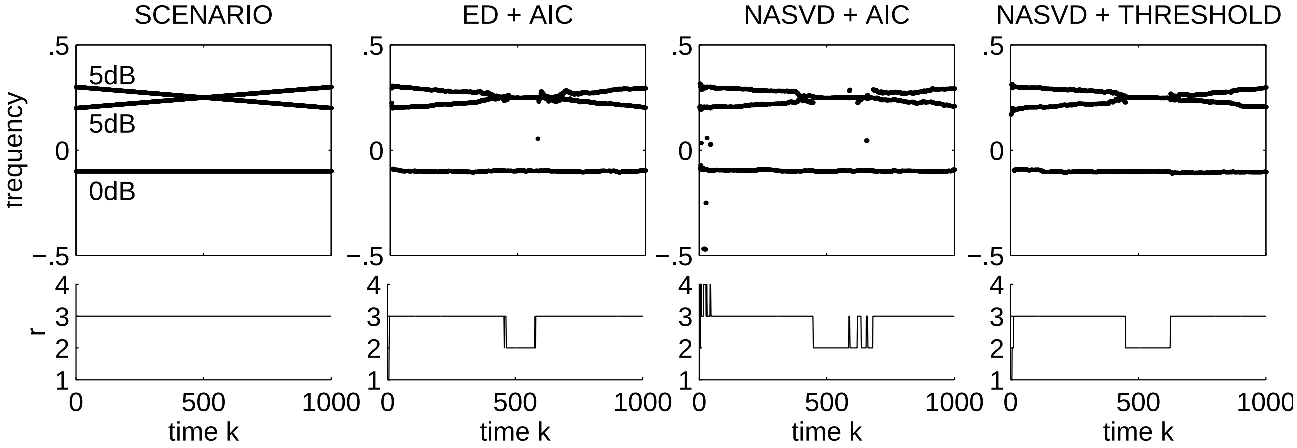


Figure 7: Tracking linearly varying and crossing frequencies.

which are then passed to the linear array for another update. Notice that Jacobi-type NASVD offers an advantage over the TQR-based SVD.[6] Namely, only one column of $r{+}1{=}3$ rotation cells are needed instead of $(r{+}1)(r{+}2)/2$ rotation cells as in the case of TQR-SVD. This offers time savings of r clock cycles per update.

We now briefly discuss the behavior of the array in Figure 6 as the rank r changes. Suppose that we have an a priori knowledge that we are going to track at most r_{max} sources. However, if we are tracking only r sources ($r{\le}r_{max}$), not all cells of the array need to be active. So only the cells of the first $r{+}2$ rows (denoted by thick lines in Figure 6) are active, while the rest (denoted by dashed lines) are idle. If the rank decreases or increases, the change has to be detected in the last two diagonal processor cells of the triangular array since these two cells have access to the elements $\mathrm{R_k}\{r,r\}$ and $\mathrm{R_k}\{r{+}1,r{+}1\}$ that are compared to the threshold. Clearly, we have to alter the cell description of Figure 2 to allow for rank change detection. We can do this on the cell (local) level by introducing appropriate logic flags, or on an array level with appropriate global control signals. Without going into detail of the actual implementation, we just acknowledge that the last active diagonal cell of the array denoted by $\mathbf{R}_k$ in Figure 6 creates a global control signal (denoted by c_r in the figure) that carries information about a possible rank change. According to this control signal the active portion of the systolic array increases or decreases in the same manner as the rank itself.

5 SIMULATION RESULTS

In this section we compare the frequency tracking capabilities of the Jacobi-type NASVD updating to the exact batch eigenvalue decomposition (ED) of a sample correlation matrix. After each update we apply the root-MUSIC algorithm to extract the tracked frequencies from the updated subspace. We also compare the dimension tracking capabilities of the threshold comparison technique to the AIC criterion.[20,14] We plotted the results in Figures 7 and 8.

Figure 7 reflects the following experiment. We are tracking $r{=}3$ uncorrelated complex sinusoids with varying frequencies in additive white noise. The tracked frequencies $f{=}\omega/2\pi$ are normalized to lie between -0.5 and 0.5. In all experiments, the problem size is $n{=}9$, and the forgetting factor β is set equal to 0.97. The frequency of the first sinusoid is constant at -0.1 throughout the tracking window of 1000 updates. Its power is at the same level as the noise power. The second and third sinusoids have linearly time varying frequencies. They start at

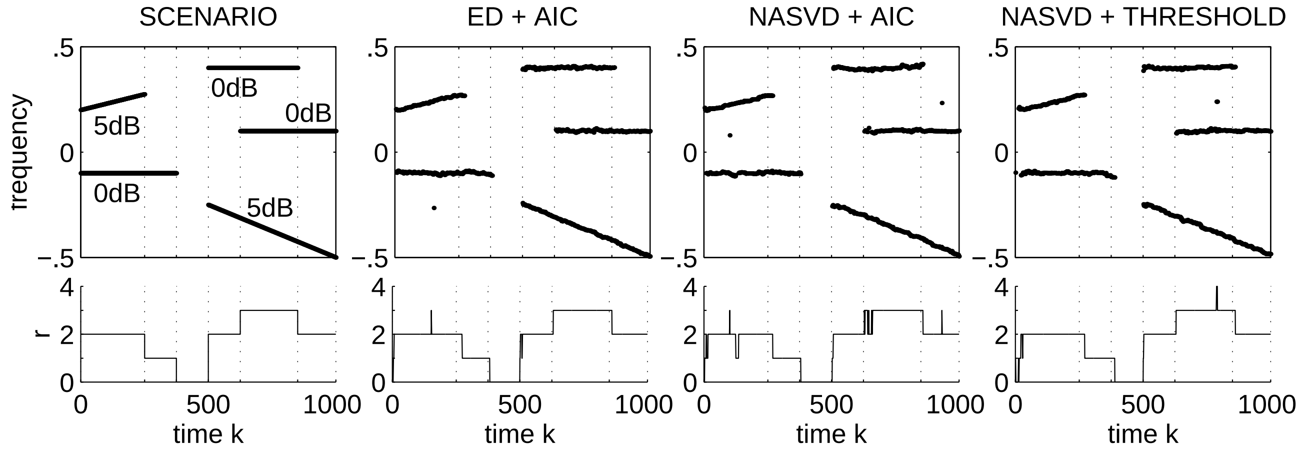


Figure 8: Tracking emerging and vanishing frequencies and the signal subspace dimension r.

frequencies 0.2 and 0.3, cross at 0.25, and finish at 0.3 and 0.2, respectively. The power of each one of the two signals is 5dB above the noise level. For the threshold dimension tracking we used the thresholding factor ϕ=1.7.

In Figure 8, we show a different example to demonstrate that the Jacobi-type NASVD algorithm is capable of tracking suddenly emerging and vanishing frequency components. We choose the forgetting factor to be β=0.9 to allow for faster detection of sudden frequency hops. We have also increased the thresholding factor to ϕ=2. This is necessary since due to a smaller forgetting factor there is less "memory" in the system and the estimated non-averaged singular values fluctuate more rapidly about their expected values. Therefore, to decrease the number of errors made by the threshold comparison subspace dimension tracking, we need to increase the thresholding value. Notice that in Figure 8 the simple threshold comparison criterion delivers comparable results to the much more complex AIC information criterion. Also, notice that the Jacobi-type NASVD algorithm tracks the frequencies as well as the batch ED.

6 CONCLUSION

We presented an efficient subspace tracking algorithm based on Jacobi-type singular value decomposition. Its basic steps are QR-decomposition and Jacobi rotations. The $O(nr)$ computational complexity of the algorithm is achieved by averaging noise singular values. In addition to tracking the subspace basis vectors, the algorithm also tracks the signal singular values, whose squares are estimates of signal component powers biased by the noise power. Together with an averaged noise singular value, the signal singular values are used to track the dimension of the signal subspace. The structure of the presented NASVD algorithm is well suited for easy determination of the tracked subspace's dimension. We demonstrated this by implementing an extremely simple dimension tracking criterion based on threshold comparison. A systolic array was constructed showing that the dimension tracking process doesn't destroy the parallelism of the subspace tracking method. We tested the presented NASVD subspace tracking algorithm in frequency tracking applications. Our results reveal that the Jacobi-type NASVD tracks the signal subspace almost as well as the much more complex batch eigenvalue decomposition. We also compared our threshold comparison dimension tracking criterion to the information theoretic criteria (specifically AIC) and demonstrated that the complex AIC criterion offers no practical advantages over our simpler method. All this supports the claim that the Jacobi-type NASVD, combined with the threshold comparison dimension tracking, is an excellent algorithm for real-time adaptive subspace tracking applications.

7 REFERENCES

[1] R. O. Schmidt, "Multiple emiter location and signal parameter estimation," in *Proc. RADC Spectrum Estimation Workshop*, pp. 243–258, October 1979.

[2] R. Kumaresan and D. W. Tufts, "Estimating the angles of arrival of multiple plane waves," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 19, pp. 134–139, 1983.

[3] P. Comon and G. H. Golub, "Tracking a few extreme singular values and vectors," *Proc. IEEE*, pp. 1327–1343, August 1990.

[4] M. Moonen, P. Van Dooren, and J. Vandewalle, "Updating singular value decomposition. A parallel implementation," in *Proc. SPIE Advanced Algorithms and Architectures for Signal Processing*, vol. 1152, (San Diego), pp. 80–91, August 1989.

[5] A. Kavčić and B. Yang, "A new efficient subspace tracking algorithm based on singular value decomposition," in *Proceedings IEEE ICASSP'94*, (Adelaide), pp. IV485–IV488, April 1994.

[6] E. M. Dowling, L. P. Amman, and R. D. DeGroat, "A TQR-iteration based adaptive SVD for real time angle and frequency tracking," *IEEE Trans. Signal Processing*, vol. 42, pp. 914–926, April 1994.

[7] R. J. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numer. Math.*, vol. 31, pp. 31–48, 1979.

[8] R. D. DeGroat, "Noniterative subspace tracking," *IEEE Trans. Signal Processing*, vol. 4, pp. 571–577, March 1992.

[9] E. M. Dowling and R. D. DeGroat, "Sphericalized SVD updating for subspace tracking," in *Proc. 3rd International Workshop on SVD and Signal Processing*, (Lueven), August 1994.

[10] J. Yang and M. Kaveh, "Adaptive eigensubspace algorithms for direction of frequency estimation tracking," *IEEE Trans. ASSP*, vol. 36, pp. 241–251, 1988.

[11] B. Yang, "Gradient based subspace tracking algorithms and systolic implementation," *Int. J. of High Speed Electronics and Systems*, vol. 4, pp. 203–218, 1993.

[12] B. Yang, "Projection approximation subspace tracking," *IEEE Trans. Signal Processing*, vol. 43, pp. 95–107, January 1995.

[13] G. W. Stewart, "An updating algorithm for subspace tracking," *IEEE Trans. Signal Processing*, vol. 40, pp. 1535–1541, June 1992.

[14] B. Yang and F. Gersemsky, "An adaptive algorithm of linear computational complexity for both rank and subspace tracking," in *Proc. IEEE ICASSP'94*, (Adelaide), pp. IV33–IV36, April 1994.

[15] M. Moonen, *Jacobi-type Updating Algorithms for Signal Processing, Systems Identification and Control.* PhD thesis, Department of Electrical Engineering, Katholike Universiteit Lueven, Heverlee, Belgium, 1990.

[16] K. V. Fernando and S. J. Hammarling, "Systolic array computation of the SVD of complex matrices," in *Proc. SPIE Advanced Algorithms and Architectures for Signal Processing*, vol. 696, pp. 54–61, 1986.

[17] G. W. Stewart, "A Jacobi-like algorithm for computing the Schur decomposition of a non-Hermitian matrix," Tech. Rep. 1321, Computer Science Department, University of Maryland, 1983.

[18] F. T. Luk, "A triangular processor array for computing the singular value decomposition," Tech. Rep. 84-625, Department of Computer Science, Cornell University, 1984.

[19] 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.*, vol. 27, pp. 17–36, 1989.

[20] M. Wax and T. Kailath, "Detection of signals by information theoretic criteria," *IEEE Trans. ASSP*, vol. 33, pp. 387–392, 1985.