SIAM J. MATRIX ANAL. APPL.
Vol. 13, No. 4, pp. 1015–1038, October 1992

A SINGULAR VALUE DECOMPOSITION UPDATING ALGORITHM FOR SUBSPACE TRACKING*

MARC MOONEN†§, PAUL VAN DOOREN‡, AND JOOS VANDEWALLE†

Abstract. In this paper, the well-known QR updating scheme is extended to a similar but more versatile and generally applicable scheme for updating the singular value decomposition (SVD). This is done by supplementing the QR updating with a Jacobi-type SVD procedure, where apparently only a few SVD steps after each QR update suffice in order to restore an acceptable approximation for the SVD. This then results in a reduced computational cost, comparable to the cost for merely QR updating.

The usefulness of such an approximate updating scheme when applied to subspace tracking is examined. It is shown how an $\mathcal{O}(n^2)$ SVD updating algorithm can restore an acceptable approximation at every stage, with a fairly small tracking error of approximately the time variation in $\mathcal{O}(n)$ time steps.

Finally, an error analysis is performed, proving that the algorithm is stable, when supplemented with a Jacobi-type reorthogonalization procedure, which can easily be incorporated into the updating scheme.

Key words. singular value decomposition, recursive algorithms

AMS(MOS) subject classifications. 65F15, 65F25

C.R. classification. G.1.3

1. Introduction.

In many signal processing applications, it is necessary to continuously update matrix decompositions as new measurement vectors are appended as additional rows. Such problems frequently occur in beam forming, direction finding, spectral analysis, etc. [21]. Efficient updating techniques have long been known for the QR decomposition [8], while the more difficult problem of updating the (ordinary) singular value decomposition (SVD) has only recently been addressed [1], [2], [4], [9], [20].

Previously described techniques for row updating of the SVD mostly reduce to computing the rank-one modification of the corresponding symmetric eigenvalue problem [1], [2], [9]. A major drawback is the necessary knowledge of the exact eigenstructure of the original matrix in order to compute the updated eigenstructure. For real-time applications, where in each time step an exact updating is thus to be performed, this results in an unacceptably heavy computational load, viz., $\mathcal{O}(n^3)$ per update. Moreover, round-off errors due to the use of finite precision arithmetic are likely to accumulate unboundedly.

In this paper, we derive a fast SVD updating technique as a combination of QR updating on the one hand and a Jacobi-type SVD procedure on the other hand. The updating scheme provides only an approximate decomposition after each update. It has a low computational complexity—$\mathcal{O}(n^2)$ per update—and it is particularly suited to parallel implementation [18].

* Received by the editors January 24, 1990; accepted for publication (in revised form) December 26, 1990. This work was sponsored in part by the BRA3280 project of the European Community.

† Department of Electrical Engineering (ESAT), Katholieke Universiteit Leuven, K. Mercierlaan 94, 3001 Heverlee, Belgium (moonen@esat.kuleuven.ac.be and vandewalle@esat.kuleuven.ac.be).

‡ Department of Electrical and Computer Engineering, University of Illinois, 1101 West Springfield Avenue, Urbana, Illinois 61801 (vdooren@uicsl.csl.uiuc.edu).

§ This author is a senior research assistant with the N.F.W.O. (Belgian National Fund for Scientific Research).

When combined with exponential weighting, such an algorithm is seen to be highly applicable to subspace tracking problems. SVD methods are known to be extremely reliable in this respect, but are considered "too expensive" when it comes to real-time applications (cf. the $\mathcal{O}(n^3)$ complexity). Cheaper alternatives usually suffer from poor numerical properties (see [4] for a survey). We will show how an $\mathcal{O}(n^2)$ approximate SVD updating algorithm can restore an acceptable approximation at every stage, with a fairly small tracking error approximately equal to the time variation in $\mathcal{O}(n)$ time steps.

Finally, an error analysis is performed, proving that the algorithm is stable, when supplemented with a certain Jacobi-type reorthogonalization procedure, which can easily be incorporated into the updating scheme.

The SVD updating procedure is developed in §2. In §3 a performance analysis for subspace tracking (in infinite precision arithmetic) is sketched. Finally, error build-up issues (finite precision arithmetic) are addressed in §4.

2. An SVD updating algorithm. The SVD of a real matrix $A_{m\times n}$ $(m \geq n)$ is a factorization of A into a product of three matrices [10]

$$\underbrace{A}_{m\times n} = \underbrace{U}_{m\times n} \cdot \underbrace{\Sigma}_{n\times n} \cdot \underbrace{V^T}_{n\times n}$$

where

$$U^T \cdot U = I_{n\times n},$$
$$V^T \cdot V = I_{n\times n},$$

and Σ is a diagonal matrix, with the singular values along the diagonal

$$\Sigma = \operatorname{diag}\{\sigma_1, \sigma_2, \cdots, \sigma_n\}.$$

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0.$$

Updating the SVD after appending a new row consists of computing the SVD of the modified matrix

$$A_+ = \begin{bmatrix} A \\ a^T \end{bmatrix} = \underbrace{U_+}_{(m+1)\times n} \cdot \underbrace{\Sigma_+}_{n\times n} \cdot \underbrace{V_+^T}_{n\times n}$$

by making use of the original SVD of A. In on-line applications, a new updating often has to be performed after each sampling, and the data matrix at time step k is defined in a recursive manner $(k \geq n)$

$$A_{[k]} = \begin{bmatrix} \lambda_{[k]} \cdot A_{[k-1]} \\ a_{[k]}^T \end{bmatrix} = \underbrace{U_{[k]}^{\circ}}_{k\times n} \cdot \underbrace{\Sigma_{[k]}^{\circ}}_{n\times n} \cdot \underbrace{V_{[k]}^{\circ T}}_{n\times n}.$$

Factor $\lambda_{[k]}$ is a weighting factor, and $a_{[k]}$ is the measurement vector at time instant 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. Since we will consider approximate decompositions below, we use an additional superscript $^\circ$ here to denote exact decompositions. Finally, in most applications the $U_{[k]}^{\circ}$-matrix, with growing matrix dimensions, need not be computed, and only $V_{[k]}^{\circ}$ and $\Sigma_{[k]}^{\circ}$ are explicitly updated.

2.1. SVD updating, first version. An SVD updating algorithm (for infinite precision arithmetic) is readily constructed by combining QR updating with a Jacobi-type SVD diagonalization procedure (Kogbetliantz's algorithm, modified for triangular matrices [14], [15]).

Suppose that at a certain time step $k-1$, the $A_{[k-1]}$-matrix is reduced to $R_{[k-1]}$—upper triangular and almost diagonal—with corresponding matrices $U_{[k-1]}$ and $V_{[k-1]}$:

$$A_{[k-1]} = U_{[k-1]} \cdot R_{[k-1]} \cdot V^T_{[k-1]}.$$

After appending a new row $a^T_{[k]}$, we have a decomposition of the type

$$\begin{aligned} A_{[k]} &= \begin{bmatrix} \lambda \cdot A_{[k-1]} \\ a^T_{[k]} \end{bmatrix} \\ &= \begin{bmatrix} U_{[k-1]} & \\ & 1 \end{bmatrix} \cdot \begin{bmatrix} \lambda \cdot R_{[k-1]} \\ a^T_{[k]} \cdot V_{[k-1]} \end{bmatrix} \cdot V^T_{[k-1]}. \end{aligned}$$

The updating can then be carried out in the following three steps.

1. *Matrix-vector multiplication and exponential weighting.* The triangular factor is multiplied by the weighting factor, and the input vector $a_{[k]}$ is transformed to $\tilde{a}_{[k]}$ by making use of the current $V_{[k-1]}$-matrix:

$$\begin{aligned} \tilde{R}_{[k-1]} &= \lambda \cdot R_{[k-1]}, \\ \tilde{a}^T_{[k]} &= a^T_{[k]} \cdot V_{[k-1]}. \end{aligned}$$

2. QR *updating* with $\tilde{a}_{[k]}$ in order to restore the triangular structure:

$$\begin{aligned} A_{[k]} &= \begin{bmatrix} U_{[k-1]} & \\ & 1 \end{bmatrix} \cdot \begin{bmatrix} \tilde{R}_{[k-1]} \\ \tilde{a}^T_{[k]} \end{bmatrix} \cdot V^T_{[k-1]} \\ &= \begin{bmatrix} U_{[k-1]} & \\ & 1 \end{bmatrix} \cdot Q_{[k]} \cdot \begin{bmatrix} \hat{R}_{[k]} \\ 0 \end{bmatrix} \cdot V^T_{[k-1]} \\ &= \underbrace{\begin{bmatrix} U_{[k-1]} & \\ & 1 \end{bmatrix} \cdot Q_{[k]} \cdot \begin{bmatrix} I_{n\times n} \\ 0 \end{bmatrix}}_{\hat{U}_{[k]}} \cdot \hat{R}_{[k]} \cdot V^T_{[k-1]}. \end{aligned}$$

The QR updating is done by applying a sequence of Givens rotations (see, e.g., [8] for details). Note that the QR updating does not alter the V-matrix. The U-matrix does change, but it does not have to be stored anyway, as we are only interested in R and V.

3. SVD *steps* in order to obtain a diagonal matrix. This diagonalization procedure consists in applying a sequence of plane rotations as follows (see [14] and [15] for details):

```
$R_{[k]} \Leftarrow \hat{R}_{[k]}$
$V_{[k]} \Leftarrow V_{[k-1]}$

for $j = 1, \cdots, r$
   for $i = 1, \cdots, n-1$
```

```
            $R_{[k]} \Leftarrow \Theta^T_{[i,j,k]} \cdot R_{[k]} \cdot \Phi_{[i,j,k]}$
            $V_{[k]} \Leftarrow V_{[k]} \cdot \Phi_{[i,j,k]}$
        end
    end
```

The parameter i is called the *pivot index*. The matrices $\Theta_{[i,j,k]}$ and $\Phi_{[i,j,k]}$ represent *rotations in the* $(i, i+1)$*-plane*:

$$\Theta_{[i,j,k]} = \begin{bmatrix} I_{i-1} & & & \\ & \cos\theta_{[i,j,k]} & \sin\theta_{[i,j,k]} & \\ & -\sin\theta_{[i,j,k]} & \cos\theta_{[i,j,k]} & \\ & & & I_{n-i-1} \end{bmatrix},$$

$$\Phi_{[i,j,k]} = \begin{bmatrix} I_{i-1} & & & \\ & \cos\phi_{[i,j,k]} & \sin\phi_{[i,j,k]} & \\ & -\sin\phi_{[i,j,k]} & \cos\phi_{[i,j,k]} & \\ & & & I_{n-i-1} \end{bmatrix},$$

where I_l is an $l \times l$ identity matrix. The rotation angles $\theta_{[i,j,k]}$ and $\phi_{[i,j,k]}$ should be chosen so as to annihilate the $(i, i+1)$ element in $R_{[k]}$, while preserving $R_{[k]}$ in upper triangular form. Each iteration thus essentially reduces to applying a 2×2 SVD on the main diagonal. The SVD procedure then consists in performing r sequences of $n-1$ such plane rotations, where the pivot index repeatedly takes up all the values $i = 1, \cdots, n-1$. It is well known that if use is made of *outer rotations* (see [23] and [16]), exactly n such sequences constitute a double sweep for a cyclic ordering (pipelined forward and backward sweep). Each rotation reduces the off-norm in $R_{[k]}$, and $R_{[k]}$ eventually converges to a diagonal matrix [6], so that finally we will have

$$R_{[k]} = \Sigma^\circ_{[k]},$$
$$V_{[k]} = V^\circ_{[k]}.$$

With the above procedure, the diagonal structure of R can be restored after each update. With $r = \mathcal{O}(n)$ on the average, the operation count is $\mathcal{O}(n^3)$ per update. In practice, however, it generally suffices to keep $R_{[k]}$ "close" to a (block) diagonal matrix, instead of *completely* reducing it to a diagonal matrix in each time step. In some sense (see §3) $V_{[k]}$ is then "close" to $V^\circ_{[k]}$ as well, which, for subspace tracking applications, for instance (ESPRIT, systems identification, recursive total least squares), is the only thing that matters. The number of rotations $r(n-1)$ in a certain time step can then be fixed, turning an iterative algorithm into a seemingly noniterative one. Of course, the crux is then to show how only a few SVD steps after each QR update can restore an acceptable approximation at every stage.

From now on, we make a fairly arbitrary choice and set r equal to 1, so that after each QR update, the pivot index takes up all values $i = 1, \cdots, n-1$ only once. In this case, both the QR updating and the rotations following the update take $\mathcal{O}(n^2)$ operations, which, e.g., results in an elegant parallel implementation [18]. It should be stressed, however, that all of our further results can straightforwardly be recast for other choices for r. In §3, the problem is addressed of "how closely" the obtained estimates then approximate the exact decomposition—in particular for the subspace tracking problem—and for this particular choice for r.

For the time being, the updating procedure is thus summarized as follows (with $n-1$ rotations after each update).

```
Initialization

                    $V_{[0]} \Leftarrow I_{n\times n}$,
                    $R_{[0]} \Leftarrow O_{n\times n}$

Loop
    for $k = 1, \cdots, \infty$
        1. input new measurement vector $a_{[k]}$

                    $\tilde{a}_{[k]}^T \Leftarrow a_{[k]}^T \cdot V_{[k-1]}$,
                    $\tilde{R}_{[k-1]} \Leftarrow \lambda \cdot R_{[k-1]}$

        2. QR updating

                    $\begin{bmatrix} \hat{R}_{[k]} \\ 0 \end{bmatrix} \Leftarrow Q_{[k]}^T \cdot \begin{bmatrix} \tilde{R}_{[k-1]} \\ \tilde{a}_{[k]}^T \end{bmatrix}$.

                    $R_{[k]} \Leftarrow \hat{R}_{[k]}$,
                    $V_{[k]} \Leftarrow V_{[k-1]}$

        3. SVD steps
           for $i = 1, \cdots, n-1$

                    $R_{[k]} \Leftarrow \Theta_{[i,k]}^T \cdot R_{[k]} \cdot \Phi_{[i,k]}$,
                    $V_{[k]} \Leftarrow V_{[k]} \cdot \Phi_{[i,k]}$

           end
    end
```

2.2. Version 2, including reorthogonalizations. In view of round-off accumulation (finite precision arithmetic), the above updating algorithm has one shortcoming. The stored matrix V is iteratively updated by orthogonal column transformations according to

$$V_{[k]} \Leftarrow V_{[k]} \cdot \Phi_{[i,k]}.$$

While $V_{[0]}$ is orthogonal through the initialization, $V_{[k]}$ ($k \gg 1$) is probably not, due to round-off. The deviation from orthogonality apparently grows linearly with k, as can be verified experimentally. Keeping $V_{[k]}$ close to orthogonal is crucial, however, for the overall error propagation stability (see §4). Including some kind of reorthogonalization procedure is therefore indispensable. An efficient procedure that elegantly combines with the updating scheme (e.g., on a systolic array [18]) can be constructed as follows.

Suppose two vectors x_p and x_q are *almost orthonormal*, in the sense that

$$\|x_p\|_2 = 1 + \mathcal{O}(\epsilon),$$
$$\|x_q\|_2 = 1 + \mathcal{O}(\epsilon),$$
$$x_p^T \cdot x_q = \mathcal{O}(\epsilon),$$

TABLE 1

Number of sweeps	$\|X^T \cdot X - I\|_F$
	1.7335e-01
1	2.0767e-03
2	1.5657e-06
3	0.6442e-13
4	

where ϵ is a small number. We can easily verify that a (symmetrized) Gram–Schmidt-like transformation

$$\begin{bmatrix} x_p^\star & x_q^\star \end{bmatrix} = \begin{bmatrix} x_p & x_q \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\|x_p\|} & -\frac{x_p^T \cdot x_q}{2} \\ -\frac{x_p^T \cdot x_q}{2} & \frac{1}{\|x_q\|} \end{bmatrix}$$

yields two new vectors, which satisfy

$$\|x_p^\star\|_2 = 1 + \mathcal{O}(\epsilon^2),$$
$$\|x_q^\star\|_2 = 1 + \mathcal{O}(\epsilon^2),$$
$$x_p^{\star T} \cdot x_q^\star = \mathcal{O}(\epsilon^2).$$

Note that an exact Gram–Schmidt orthogonalization is computationally somewhat more involved, while on the other hand it yields only marginally better results.

For an $n \times n$ close-to-orthogonal matrix

$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix},$$

we can straightforwardly apply the 2×2 transformations in a cyclic manner. One (forward) sweep consists in computing transformations as follows.

```
Loop
        for $p = 1, \cdots, n-1$
          for $q = p+1, \cdots, n$

            $\begin{bmatrix} x_p & x_q \end{bmatrix} \Leftarrow \begin{bmatrix} x_p & x_q \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\|x_p\|} & -\frac{x_p^T \cdot x_q}{2} \\ -\frac{x_p^T \cdot x_q}{2} & \frac{1}{\|x_q\|} \end{bmatrix}$

          end
        end
```

The above algorithm is seen to be a one-sided Jacobi-type procedure with nonunitary transformations, or two-sided if we consider the effect on $X^T \cdot X$. Furthermore, if $\|X^T \cdot X - I\|_F = \mathcal{O}(\epsilon)$ for some small ϵ, the 2×2 transformations are ϵ close to $I_{2\times 2}$. By making use of this, we can easily prove that the procedure converges quadratically, in other words, that $\|X^T \cdot X - I\|_F = \mathcal{O}(\epsilon^2)$ after one sweep. It suffices to copy Wilkinson's proof [24] with appropriate substitutions. In Table 1 we show the effect of the procedure on a *random* 10×10 X-matrix (i.e., *not even close* to an orthogonal one).

Let us now return to the SVD updating algorithm. If we choose

$$X = V$$

and interlace the above reorthogonalization with the updating procedure, the former no longer converges quadratically, due to the updating transformations $X \Leftarrow X \cdot \Phi_{[i,k]}$ that clearly change $X^T \cdot X$ as well and thus interfere with the reorthogonalization. However, if we choose

$$X = V^T,$$

in other words, if we apply the reorthogonalization scheme onto the rows of V, the updating transformations do not change $X^T \cdot X$, at least not for (local) infinite precision arithmetic $(X^T \cdot \Phi_{[i,k]} \cdot \Phi^T_{[i,k]} \cdot X = X^T \cdot X)$, so that *apparently* both processes do not interfere. In finite precision, these processes of course do interfere. Both the updating transformations and the reorthogonalization steps introduce new round-off errors, which have to be annihilated by the reorthogonalization itself. On the other hand, the reorthogonalization (slightly) changes the $V_{[k]}$ matrix in an abitrary manner with respect to the data. These issues will be analyzed in detail in §4.

The updating and the reorthogonalization can now be interlaced. For instance, we could alternately perform one updating rotation, one reorthogonalization step, etc. The body of the inner "for"-loop in the updating algorithm then becomes

```
        ⋮
3. SVD steps and reorthogonalization
   for $i = 1, \cdots, n-1$
        $R_{[k]} \Leftarrow \Theta^T_{[i,k]} \cdot R_{[k]} \cdot \Phi_{[i,k]}$
        $V_{[k]} \Leftarrow T_{[i,k]} \cdot V_{[k]} \cdot \Phi_{[i,k]}$
   end
        ⋮
```

where $T_{[i,k]}$ is a reorthogonalization in the $(p_{[i,k]}, q_{[i,k]})$-plane. Here the number of pairwise reorthogonalizations per update equals the number of updating transformations. Again, this is an arbitrary choice, which in some cases might be overly conservative. Further results on the obtained accuracy can however easily be recast for other choices. Furthermore, e.g., in a systolic array implementation [18] with a separate V-array and R-array, the above additional reorthogonalizations do not introduce any computational overhead, but rather a load balancing between the two arrays, so that there is no point in reducing the number of reorthogonalizations.

Finally, in view of efficient (parallel) implementation (see also [18]), the cyclic reorthogonalization can of course be reordered, by making use of additional permutations (outer transformations) and a pipelining of the forward and backward sweep (so that $p_{[i,k]} = i$ and $q_{[i,k]} = i + 1$). For the time being, however, we do not pursue this, as it would considerably complicate our notation in the subsequent analysis.

3. Subspace tracking.

In this section, we analyze the performance of the SVD updating scheme, when applied to subspace tracking. We assume that all computations are performed with infinite precision, so that the reorthogonalization is superfluous. Round-off errors introduce second-order effects, which for the time being are left out for the sake of simplicity.

First of all, a data model is put forward, which applies to popular signal processing applications, such as direction finding (ESPRIT, MUSIC) and system identification [17]. For such applications, an SVD step is used to separate a signal subspace from a noise subspace (see below), corresponding to certain submatrices of $V^{\circ}_{[k]}$. 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 (block) diagonal triangular factor $R_{[k]}$, the corresponding subspace separation 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 these are applied to the updating problem.

3.1. Data model. The data model we consider only assumes that at each time step k the data matrix $A_{[k]}$ as defined in §2 has a fixed number d of large singular values

$$\sigma_{i_{[k]}}, \quad i = 1, \cdots, d$$

and a remaining number of *small* singular values

$$\sigma_{i_{[k]}}, \quad i = d+1, \cdots, n.$$

The ratio

$$SN_{[k]} = \frac{\sigma_{d_{[k]}}}{\sigma_{d+1_{[k]}}}$$

can be interpreted as a *signal-to-noise ratio*, and is assumed to be large, e.g., at least 10 or 100, and lower bounded by a constant SN

$$SN_{[k]} \geq SN.$$

The corresponding submatrices $Vs^{\circ}_{[k]}$ and $Vn^{\circ}_{[k]}$ in $V^{\circ}_{[k]}$ define the *signal subspace* $\mathcal{R}(Vs^{\circ}_{[k]})$ and the *noise subspace* $\mathcal{R}(Vn^{\circ}_{[k]})$, orthogonal to $\mathcal{R}(Vs^{\circ}_{[k]})$.

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 θ_i between $\mathcal{R}(Vs^{\circ}_{[k-1]})$ and $\mathcal{R}(Vs^{\circ}_{[k]})$, the cosines of which are the singular values of a corresponding matrix product [10]

$$\cos\theta_i = \sigma_i\{Vs^{\circ^T}_{[k-1]} \cdot Vs^{\circ}_{[k]}\}.$$

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 above canonical angles as follows:

$$TV_{[k-1]\to[k]} \overset{\text{def}}{=} \text{dist}\{\mathcal{R}(Vs^{\circ}_{[k-1]}), \mathcal{R}(Vs^{\circ}_{[k]})\}$$
$$\overset{\text{def}}{=} \sqrt{\sum_{i=1}^{d} \tan^2\theta_i}.$$

The reason for this will become clear in what follows.

3.2. Tracking error. The adaptive SVD algorithm of §2 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:

$$R_{[k]} = \begin{bmatrix} Rs_{[k]} & Rsn_{[k]} \\ 0 & Rn_{[k]} \end{bmatrix},$$
$$V_{[k]} = \begin{bmatrix} Vs_{[k]} & Vn_{[k]} \end{bmatrix},$$

where $\|Rn_{[k]}\|$ and $\|Rsn_{[k]}\|$ are "small." For the time being, we thus assume that at time instant k, the large diagonal elements in $R_{[k]}$ occupy adjacent positions, as this considerably simplifies our further analysis. We return to this in Remark 2, below.

It is now clear that $Vs_{[k]}$ provides an approximate basis for the signal subspace. We can then define the *tracking error* at time step k as follows:

$$TE_{[k]} \overset{\text{def}}{=} \text{dist}\{\mathcal{R}(Vs_{[k]}), \mathcal{R}(Vs^{\circ}_{[k]})\}.$$

Our aim is to derive a useful estimate for the tracking error TE in terms of the time variation TV. In order to obtain this, we can make use of a well-known property, relating the tracking error to the distance of $R_{[k]}$ from a (block) diagonal matrix, as follows. If ε denotes the Frobenius norm of the matrix with cross terms

$$\varepsilon = \|Rsn_{[k]}\|_F$$

and δ is the gap between the singular values of $Rs_{[k]}$ and $Rn_{[k]}$.

$$\delta = \sigma_{\min}\{Rs_{[k]}\} - \sigma_{\max}\{Rn_{[k]}\},$$

and furthermore if

$$2\varepsilon < \delta,$$

then it is has been shown [3], [22] that

$$TE_{[k]} < 2\frac{\varepsilon}{\delta}.$$

In other words, the tracking error is proportional to the norm of the block of cross terms in $R_{[k]}$.

3.3. Kogbetliantz's algorithm. Suppose we would now perform a few sweeps of Kogbetliantz's SVD algorithm (without any QR updates!) and then again check the norm of the cross terms. Classical convergence results for Kogbetliantz's algorithm turn out to be useless in this respect. Linear convergence bounds are extremely conservative [5], [6], [11], [13], while ultimate quadratic convergence results do not apply to the initial convergence, where the off-diagonal elements in $Rs_{[k]}$ can be very large [3], [19], [24]. Jacobi-type algorithms are considered extremely fast, but for the general case no estimates are available whatsoever for the speed of the initial convergence. For our specific subspace separation problem, however, it is possible to derive a useful estimate for the reduction of the cross terms in each sweep. In Appendix A, the following rule of thumb is obtained.

If the cross terms are small compared to the gap and the large diagonal elements are grouped, each double sweep in Kogbetliantz's SVD algorithm reduces the cross terms by a factor $1/SN^2$.

TABLE 2

Number of sweeps	ε
	2.3145e−02
2	0.7311e−06
4	0.6931e−10
6	0.7499e−14

Our derivation relies on a few simplifying assumptions, in order to avoid tedious mathematics and overly conservative results. Our results, however, are practical, and easy to verify by experiments.

Example 1. For a triangular factor $R =$

$$\begin{array}{ccccccccccc}
3.7047 & 0.4920 & 0.4312 & 1.1988 & 0.8095 & & -0.0051 & -0.0032 & 0.0009 & -0.0019 & -0.0014 \\
 & 3.0436 & 1.0955 & 1.1852 & 2.5027 & & 0.0099 & -0.0018 & 0.0038 & -0.0093 & -0.0061 \\
 & & 2.8701 & 1.3763 & 0.6623 & & 0.0050 & 0.0026 & -0.0024 & 0.0013 & -0.0020 \\
 & & & 1.4314 & 1.1373 & & 0.0080 & 0.0027 & 0.0058 & 0.0055 & 0.0008 \\
 & & & & 2.5905 & & 0.0017 & -0.0006 & -0.0021 & 0.0070 & 0.0042 \\
 & & & & & & & & & & \\
 & & & & & & 0.0133 & 0.0028 & 0.0031 & 0.0003 & 0.0072 \\
 & & & & & & & 0.0130 & 0.0011 & 0.0027 & 0.0004 \\
 & & & & & & & & 0.0057 & 0.0031 & 0.0035 \\
 & & & & & & & & & 0.0089 & 0.0047 \\
 & & & & & & & & & & 0.0056
\end{array}$$

with singular values

$$\sigma = 5.4467, 3.5381, 2.7569, 1.9772, 1.1424,$$
$$0.0173, 0.0126, 0.0104, 0.0051, 0.0043.$$

SN is approximately equal to 100. If the SVD procedure were carried out as such, the cross terms would be reduced much faster than could be predicted with the above rule of thumb. As the convergence soon turns into (much faster) quadratic convergence, the $1/SN^2$ reduction of the cross terms will not be visible. Therefore, it is more instructive to see what happens if in each sweep *only* the cross terms are being annihilated, while all other rotations are skipped (corresponding to Part (b) in Appendix A). Now the $1/SN^2$ reduction is much more clearly displayed (Table 2). Note, however, that in our updating algorithm, all the rotations *are* performed, such that the cross-term reduction is indeed much faster. The point is that no (sharper) bound is available for this faster convergence. Also, as far as our algorithmic description is concerned, the sizes of the subblocks thus need not be identified whatsoever (see also Remark 2).

From the above rule of thumb, we can straightforwardly infer an estimate for the reduction of the subspace separation error. As this latter is bounded by the norm of the cross terms $\|Rsn_{[k]}\|_F$, we can conclude that each double sweep in Kogbetliantz's SVD algorithm reduces the subspace separation error $\text{dist}\{\mathcal{R}(Vs_{[k]}), \mathcal{R}(Vs^{\circ}_{[k]})\}$ by a factor $(1/SN^2)$.

Example 2. Similar to the experiment in Example 1, we checked the reduction of the subspace separation error per double sweep (for three successive double sweeps), for different triangular matrices. The matrix dimension n ranges from 10 through 50, while the singular value spectra were chosen to be

$$\sigma = \frac{n}{2}, \frac{n}{2} - 1, \cdots, 2, 1,$$
$$\frac{1}{SN}, \frac{\frac{n}{2} - 1}{\frac{n}{2} \cdot SN}, \cdots, \frac{1}{\frac{n}{2} \cdot SN}$$

for $SN = 10$ and $SN = 100$. For all cases, the reduction factor is seen to be approximately equal to $1/SN^2$. Apparently, the matrix dimension has little influence on this.

3.4. Subspace tracking. Let us now return to the SVD updating algorithm, applied to subspace tracking. In the adaptive algorithm, a pipelined double sweep ($n \times n-1$ rotations) is interlaced with n QR updates, one after each series of $n-1$ rotations. If all these QR updates were performed *after* the double sweep, we would end up with the following inequality:

$$\begin{aligned}&\operatorname{dist}\{\mathcal{R}(Vs_{[k+n]}), \mathcal{R}(Vs^{\circ}_{[k+n]})\}\\ &\leq \left(\frac{1}{SN^2}\right)\cdot \operatorname{dist}\{\mathcal{R}(Vs_{[k]}), \mathcal{R}(Vs^{\circ}_{[k]})\} + \operatorname{dist}\{\mathcal{R}(Vs^{\circ}_{[k]}), \mathcal{R}(Vs^{\circ}_{[k+n]})\}.\end{aligned}$$

The "$\leq$" sign is due to the fact that the different terms in the right-hand side can partially eliminate each other. The first term corresponds to the reduction of the subspace separation error in a double sweep (n time steps). The second term corresponds to the time variation from time instant k to time instant $k+n$.

If the QR updates are interlaced in the double sweep, the subspace separation error is expected to be even smaller, as time variations can then immediately be taken into account and corrected correspondingly. Although we can easily think of set-ups where the above statement does not even hold, in general it is confirmed by simulations (see below for an example). The above inequality can therefore be assumed to provide a reasonable estimate for the SVD updating scheme as well.

Furthermore, as we assumed that SN is fairly large (e.g., 100), it follows that

$$\underbrace{\operatorname{dist}\{\mathcal{R}(Vs_{[k+n]}), \mathcal{R}(Vs^{\circ}_{[k+n]})\}}_{TE_{[k+n]}} \leq \underbrace{\operatorname{dist}\{\mathcal{R}(Vs^{\circ}_{[k]}), \mathcal{R}(Vs^{\circ}_{[k+n]})\}}_{TV_{[k]\to[k+n]}}.$$

In other words, we can conclude that the tracking error is bounded by the time variation in n time steps.

A few remarks on the above derivations and results are as follows.

Remark 1. The above results were derived for the case where only one sequence of $n-1$ SVD rotations is performed after each QR update ($r = 1$ in §2). For other choices for r, we would obviously end up with

$$TE_{[k+\frac{n}{r}]} \leq TV_{[k]\to[k+\frac{n}{r}]},$$

as one double sweep is then performed in $\frac{n}{r}$ time steps. In other words, the tracking error is inversely proportional to r.

Remark 2. Throughout the computations (in Appendix A), we have assumed that the small diagonal elements in the triangular factor occupy (circularly) adjacent positions (cf. the configurations in Appendix A after each sequence of rotations). Note that a similar assumption had to be made for the proof of the quadratic convergence for matrices with pathologically close or repeated singular values [3], [24]. In an adaptive scheme, obtaining such a set-up is sometimes merely a matter of careful initialization (once the small elements are grouped, they do not change their "affiliation," at least not for slowly time-varying systems). Furthermore, it is observed (by performing simulations) that even when the large and small diagonal elements are *not* grouped, the tracking error is still bounded by the time variation in n time steps. In other

TABLE 3

	$n = 10$	$n = 20$	$n = 50$
$SN = 10$	$2.8100e-02$ $8.2947e-05$ $3.4249e-07$ $1.4534e-09$	$2.0356e-02$ $8.6065e-05$ $5.6355e-07$ $3.7391e-09$	$1.8847e-02$ $5.8207e-05$ $4.2585e-07$ $3.5933e-09$
$SN = 100$	$2.0213e-02$ $4.7201e-07$ $2.3465e-11$ $1.1801e-15$	$2.01087-02$ $5.8845e-07$ $3.5926e-11$ $2.6241e-15$	$1.9357e-02$ $8.9264e-07$ $7.4129e-11$ $6.9677e-15$

words, this latter bound seems to be conservative enough, so that it applies to the (disadvantageous) "nongrouped" case as well.

Remark 3. The above derivation particularly applies to cases where the signal-to-noise ratio is fairly large (e.g., 100 or more). In practice, however, it is observed that the obtained rules of thumb deliver fairly reasonable estimates for even smaller values of SN (e.g., 10; see also Table 3).

Remark 4. For large values of the matrix dimension, it can be expected that the performance slightly declines, much like it has been observed that the number of necessary sweeps in a classical SVD procedure is slightly larger for large matrices (e.g., $n > 100$). In these cases, we can easily double or triple the number of rotations after each QR update accordingly. Strictly speaking, the computational complexity of the updating algorithm could then become $\mathcal{O}(n^{2+\alpha})$, where α is (much) smaller than 1.

The following example from systems theory illustrates the above results.

Example 3 (adaptive system identification). Suppose we are given a simple first-order time-varying system, with state space equations

$$\begin{aligned} x_{[k+1]} &= .8 \cdot \cos\left(\frac{2\pi}{2000}k\right) \cdot x_{[k]} + u_{[k]}, \\ y_{[k]} &= x_{[k]}, \end{aligned}$$

where $u_{[k]}$, $y_{[k]}$, and $x_{[k]}$ are the input, output, and state at time instant k. For every arbitrary input sequence, the output can be computed accordingly, by making use of the state space equations. Conversely, the state space model at a time instant k can (approximately) be computed from the input-output data, by making use of various *system identification* techniques. In [17], it has been shown how a state space model can be computed from the following exponentially weighted block Hankel matrix:

$$A_{[k]} = W_{[k]} \cdot \begin{bmatrix} z_{[1]} & z_{[2]} & \cdots & z_{[i]} \\ z_{[2]} & z_{[3]} & \cdots & z_{[i+1]} \\ z_{[3]} & z_{[4]} & \cdots & z_{[i+2]} \\ \vdots & \vdots & & \vdots \\ z_{[k-i]} & z_{[k-i+1]} & \cdots & z_{[k-1]} \\ z_{[k-i+1]} & z_{[k-i+2]} & \cdots & z_{[k]} \end{bmatrix},$$

$$z_{[k]} = \begin{bmatrix} u_{[k]} & y_{[k]} \end{bmatrix},$$

$$W_{[k]} = \operatorname{diag}\{\lambda^{k-i}, \lambda^{k-i-1}, \cdots, \lambda^1, \lambda^0\}.$$

For a good choice of λ, the "signal" rank (i.e., the number of singular values larger than the noise level) of this weighted block Hankel matrix equals $i+1$ (i is the number

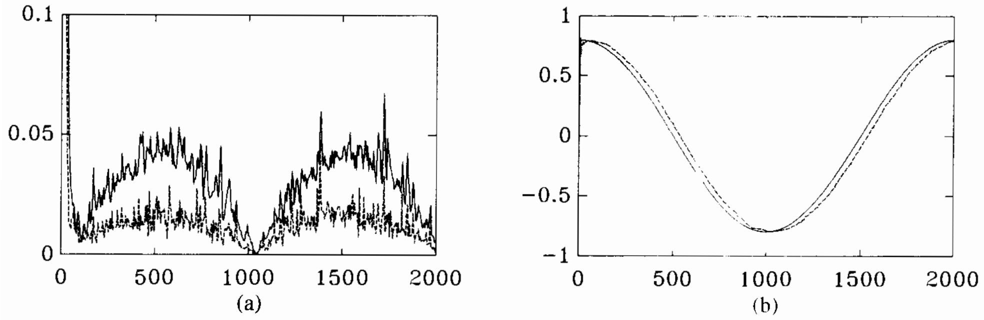

FIG. 1. (a) *Tracking error (lower curve) and time variation (upper curve) versus time* ($\lambda = 1 - 2^{-5}$). (b) *Original pole (solid line) and identified poles (dashed and dotted lines) versus time* ($\lambda = 1 - 2^{-5}$).

of columns with input data, 1 is the system order). The state space model at time instant k can then essentially be computed from the $(i+1)$-dimensional space $\mathcal{R}(Vs^{\circ}_{[k]})$ (see [17] for further details). So, first, the aim is to track the signal subspace of $A_{[k]}$, where in each time step a new row is appended. An adaptive system identification can be performed, by making use of either an exact SVD scheme (in other words, with a complete computation of the SVD at each time instance), or an adaptive SVD scheme with, e.g., $n-1$ rotations after each QR update. The parameter i was set equal to 5, so that the matrix size equals 10, with a six-dimensional signal subspace.

Figure 1(b) shows the original system pole $0.8 \cdot \cos((2\pi/2000)k)$, solid line, together with the identified pole both for the exact scheme (dashed line) and the adaptive scheme (dotted line almost coinciding with the dashed line). The exponential weighting factor λ was set equal to $1 - 2^{-5}$. Figure 1(a) shows the time variation in $n = 10$ time steps $TV_{[k]\to[k+10]}$ (solid line), together with the tracking error at each time instant $TE_{[k]}$ (dashed line). Clearly, the latter generally remains smaller than the former, confirming the above rule of thumb. Finally, note that the time variation of the data nicely reflects the time variation of the underlying system, viz., the system pole.

In Fig. 2, the same quantities have been plotted for a different choice for the weighting factor $\lambda = 1 - 2^{-8}$. A different choice for λ is seen to hardly influence the time variation and the approximation error. As for the identified pole, the exponential weighting is seen to introduce a kind of time delay, which increases when λ approaches 1. The adaptive scheme, however, still delivers quite the same system pole as the exact scheme.

4. Error analysis.

In the previous section, we analyzed the performance of the updating algorithm in infinite precision arithmetic, resulting in an upper bound for the distance between $Vs_{[k]}$ and $Vs^{\circ}_{[k]}$. In finite precision arithmetic, there are a few sources of additional error. Apart from round-off, of course, there is also the reorthogonalization scheme. Both processes change the V-matrix in an arbitrary manner with respect to the original data. As we are only interested in the right singular vectors, together with the singular values, we can define a relevant error matrix in this respect as follows:

$$\Delta_{[k]} = A^T_{[k]} \cdot A_{[k]} - (R_{[k]} \cdot V^T_{[k]})^T \cdot (R_{[k]} \cdot V^T_{[k]}).$$

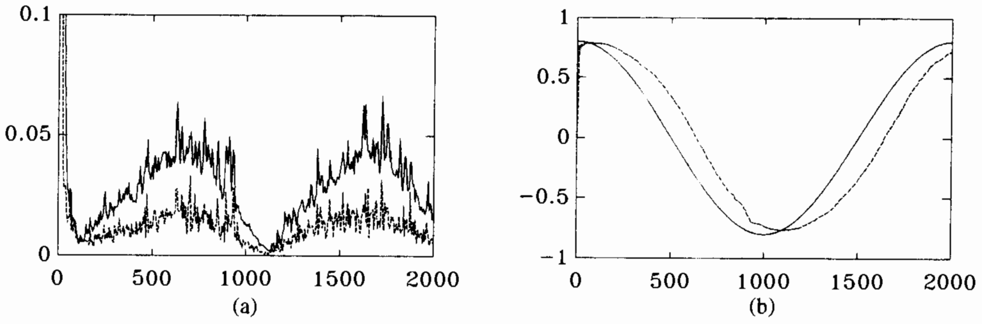

FIG. 2. (a) *Tracking error (lower curve) and time variation (upper curve) versus time* ($\lambda = 1 - 2^{-8}$). (b) *Original pole (solid line) and identified poles (dashed and dotted lines) versus time* ($\lambda = 1 - 2^{-8}$).

This error matrix tells how far the information stored in $R_{[k]} \cdot V_{[k]}^T$ has drifted off from the original data. If $\Delta_{[k]}$ is small, the singular values of $R_{[k]}$ will be close to those of $A_{[k]}$. Furthermore, for large values of SN, the signal subspace of $R_{[k]} \cdot V_{[k]}^T$ will then be close to the signal subspace of $A_{[k]}$.

In the sequel, upper bounds for the propagation of $\Delta_{[k]}$ are derived, resulting in a first-order difference equation

$$\Delta_{[k]} = \lambda^2 \cdot \Delta_{[k-1]} + \delta E_{[k]}$$

where λ is the exponential weighting factor ($\lambda < 1$). As long as $V_{[k]}$ is close to an orthogonal matrix, $\delta E_{[k]}$ contains only bounded local errors, independent of $\Delta_{[k-1]}$. In other words, the norm

$$\|\Delta I_{[k]}\|_F \overset{\text{def}}{=} \|V_{[k]} \cdot V_{[k]}^T - I\|_F$$

(where I is the identity matrix) should be kept small, in order to guarantee stability of the error propagation. Therefore, we first derive an estimate for the above norm by investigating the reorthogonalization scheme of §2. By making use of this, we then derive the above error propagation formula, showing that the overall updating procedure is stable.

4.1. An estimate for $\|V_{[k]} \cdot V_{[k]}^T - I\|_F$. In finite precision arithmetic both the updating and the reorthogonalization steps introduce new errors in the stored $V_{[k]}$-matrix. On the other hand, the reorthogonalization itself annihilates accumulated errors up to machine precision. If the number of reorthogonalizations in the updating scheme is chosen to be equal to the number of SVD rotations, one double sweep in the reorthogonalization procedure is performed after each n time steps. If we let $\Phi_{[\text{ac}]}$ and $T_{[\text{ac}]}$ denote the accumulated right and left transformations applied to V in time steps k through $k+n-1$, we have

$$\begin{aligned} V_{[k+n-1]} &= \mathbf{fl}(T_{[\text{ac}]} \cdot V_{[k-1]} \cdot \Phi_{[\text{ac}]}) \\ &= T_{[\text{ac}]} \cdot V_{[k-1]} \cdot \Phi_{[\text{ac}]} + \delta V_{[\text{ac}]}, \end{aligned}$$

where $\delta V_{[\text{ac}]}$ is an input of local errors in these time steps, and $\mathbf{fl}(\cdot)$ refers to the computer result after a sequence of transformations (in the right order). For a first-order analysis, the reorthogonalizations can be considered as orthogonal transformations, so that Gentleman's analysis [7] applies. The double sweep (with SVD steps and reorthogonalizations) then consists of $4n-4$ different "stages." Each such stage consists of the simultaneous application of disjoint transformations. This then results in an approximate upper bound for the local errors on V in one double sweep [7]:

$$\begin{aligned}\|\delta V_{[\mathrm{ac}]}\|_F &\leq (4n-4)\cdot k_V \cdot \epsilon \cdot (1+k_V\epsilon)^{4n-5}\cdot \|V\|_F\\ &\simeq 4n \cdot k_V \cdot \epsilon \cdot \|V\|_F,\end{aligned}$$

where ϵ is the largest number such that $\mathbf{fl}(1+\epsilon) = 1$ (relative machine precision) and k_V is a constant depending on the specific implementation of both the Givens rotations and the reorthogonalizations.

From the formula for $V_{[k+n-1]}$, we can derive a formal description for the error build-up process as follows:

$$\begin{aligned}V_{[k+n-1]}\cdot V_{[k+n-1]}^T = {}& T_{[\mathrm{ac}]}\cdot V_{[k-1]}\cdot \Phi_{[\mathrm{ac}]}\cdot \Phi_{[\mathrm{ac}]}^T \cdot V_{[k-1]}^T \cdot T_{[\mathrm{ac}]}^T\\ &+T_{[\mathrm{ac}]}\cdot V_{[k-1]}\cdot \Phi_{[\mathrm{ac}]}\cdot \delta V_{[\mathrm{ac}]}^T\\ &+\delta V_{[\mathrm{ac}]}\cdot \Phi_{[\mathrm{ac}]}^T\cdot V_{[k-1]}^T\cdot T_{[\mathrm{ac}]}^T\\ &+\mathcal{O}(\epsilon^2),\end{aligned}$$

$$\begin{aligned}\underbrace{\|V_{[k+n-1]}\cdot V_{[k+n-1]}^T - I\|_F}_{\|\Delta I_{[k+n-1]}\|_F} \leq {}& \underbrace{\|T_{[\mathrm{ac}]}\cdot V_{[k-1]}\cdot V_{[k-1]}^T\cdot T_{[\mathrm{ac}]}^T - I\|_F}_{\eta_{[\mathrm{ac}]}\cdot\|\Delta I_{[k-1]}\|_F}\\ &+2\cdot \|\delta V_{[\mathrm{ac}]}^T\|_F + \mathcal{O}(\epsilon^2).\end{aligned}$$

where $\eta_{[\mathrm{ac}]}$ is a factor describing the effect of the reorthogonalization steps.

As long as $\|\Delta I_{[k-1]}\|_F$ is small ($< \epsilon^{1/4}$), it is reduced to machine precision by the double sweep in the reorthogonalization scheme. The input of local errors, however, interferes with this reorthogonalization, and this introduces additional errors of the same order of magnitude. In conclusion, we end up with

$$\begin{aligned}\|\Delta I_{[k+n-1]}\|_F &\leq \underbrace{\eta_{[\mathrm{ac}]}\cdot \|\Delta I_{[k-1]}\|_F}_{\mathcal{O}(n\epsilon)} + 2\cdot\|\delta V_{[\mathrm{ac}]}\|_F + \mathcal{O}(\epsilon^2)\\ &\simeq \mathcal{O}(n\epsilon) + 8n\cdot k_V\cdot\epsilon\cdot\|V\|_F\\ &\simeq \mathcal{O}(n\epsilon) + 8n\sqrt{n}\cdot k_V\cdot\epsilon\\ &\simeq k_I\cdot n\sqrt{n}\cdot\epsilon,\end{aligned}$$

which is then a bound for $\|\Delta I_{[k]}\|_F$ for all values of k. The reorthogonalization procedure thus keeps the stored V-matrix close to an orthogonal one.

4.2. Error propagation formulas. First of all, for the sake of conciseness, let us assume that there exists an upper bound $\|R\|_F$ such that for all k,

$$\begin{aligned}\|R_{[k]}\|_F &\leq \sqrt{n}\|R_{[k]}\|_2 \leq \|R\|_F,\\ \|\tilde{R}_{[k]}\|_F &\leq \sqrt{n}\|\tilde{R}_{[k]}\|_2 \leq \|R\|_F,\\ \sqrt{n}\|a_{[k]}\|_2 &\leq \|R\|_F.\end{aligned}$$

The error matrix at time step k, viz. $\Delta_{[k]}$, can then be computed from the error matrix $\Delta_{[k-1]}$ at time step $k-1$ as follows.

In a *first step*, appending a new row together with an exponential weighting (weighting factor λ) can be described as

$$\begin{bmatrix} \lambda \cdot R_{[k-1]} \cdot V_{[k-1]}^T \\ a_{[k]}^T \end{bmatrix} = \begin{bmatrix} \lambda \cdot R_{[k-1]} \\ a_{[k]}^T \cdot V_{[k-1]} \end{bmatrix} \cdot V_{[k-1]}^T + \underbrace{\begin{bmatrix} 0 \\ -a_{[k]}^T \cdot \Delta I_{[k-1]} \end{bmatrix}}_{\delta E_{[k]}^1},$$

where $\delta E_{[k]}^1$ is an *"algorithmic"* error due to $V_{[k-1]}$ not being orthogonal. Here we can use the upper bound for ΔI of the previous section:

$$\|\delta E_{[k]}^1\|_2 \le \|\Delta I_{[k-1]}\|_2 \cdot \|a_{[k]}\|_2 \le k_I n\sqrt{n}\epsilon \cdot \|a_{[k]}\|_2 \le k_I n\epsilon \cdot \|R\|_F.$$

In a *second step*, additional round-off errors are introduced by the explicit computation of

$$\begin{aligned} \tilde{R}_{[k-1]} &= \mathbf{fl}\{\lambda \cdot R_{[k-1]}\} = \lambda \cdot R_{[k-1]} + \delta \tilde{R}_{[k-1]}, \\ \tilde{a}_{[k]}^T &= \mathbf{fl}\{a_{[k]}^T \cdot V_{[k-1]}\} = a_{[k]}^T \cdot V_{[k-1]} + \delta \tilde{a}_{[k]}^T. \end{aligned}$$

Substituting this in the above equation, we obtain

$$\begin{aligned} &\begin{bmatrix} \lambda \cdot R_{[k-1]} \cdot V_{[k-1]}^T \\ a_{[k]}^T \end{bmatrix} \\ &= \begin{bmatrix} \tilde{R}_{[k-1]} \\ \tilde{a}_{[k]}^T \end{bmatrix} \cdot V_{[k-1]}^T + \delta E_{[k]}^1 + \underbrace{\begin{bmatrix} -\delta \tilde{R}_{[k-1]} \\ 0 \end{bmatrix} \cdot V_{[k-1]}^T}_{\delta E_{[k]}^2} + \underbrace{\begin{bmatrix} 0 \\ -\delta \tilde{a}_{[k]}^T \end{bmatrix} \cdot V_{[k-1]}^T}_{\delta E_{[k]}^3} \end{aligned}$$

with first-order upper bounds (see [25])

$$\begin{aligned} &\|\delta E_{[k]}^2\|_2 \le \|\delta E_{[k]}^2\|_F \le \epsilon \cdot \|\tilde{R}_{[k-1]}\|_F \le \epsilon \cdot \|R\|_F, \\ &\|\delta E_{[k]}^3\|_2 \le n\epsilon \cdot \|a_{[k]}\|_2 \cdot \|V_{[k-1]}\|_F \le n\sqrt{n}\epsilon \cdot \|a_{[k]}\|_2 + \mathcal{O}(\epsilon^2) \simeq n\epsilon\|R\|_F. \end{aligned}$$

In a *third step*, orthogonal row transformations are performed for the QR update, introducing a round-off error $\delta \hat{R}_{[k]}$. The rotation angles are chosen such that the last row of the computer result transforms to zero:

$$\begin{bmatrix} \hat{R}_{[k]} \\ 0 \end{bmatrix} = \mathbf{fl}\left\{ Q_{[k]}^T \cdot \begin{bmatrix} \tilde{R}_{[k-1]} \\ \tilde{a}_{[k]}^T \end{bmatrix} \right\} = Q_{[k]}^T \cdot \begin{bmatrix} \tilde{R}_{[k]} \\ \tilde{a}_{[k]}^T \end{bmatrix} + \delta \hat{R}_{[k]}.$$

By making use of this, we obtain

$$\begin{aligned} \begin{bmatrix} \lambda \cdot R_{[k-1]} \cdot V_{[k-1]}^T \\ a_{[k]}^T \end{bmatrix} &= Q_{[k]} \cdot \left(Q_{[k]}^T \cdot \begin{bmatrix} \tilde{R}_{[k-1]} \\ \tilde{a}_{[k]}^T \end{bmatrix} \right) \cdot V_{[k-1]}^T + \delta E_{[k]}^1 + \delta E_{[k]}^2 + \delta E_{[k]}^3 \\ &= Q_{[k]} \cdot \begin{bmatrix} \hat{R}_{[k]} \\ 0 \end{bmatrix} \cdot V_{[k-1]}^T + \delta E_{[k]}^1 + \delta E_{[k]}^2 + \delta E_{[k]}^3 \\ &\quad \underbrace{-Q_{[k]} \cdot \delta \hat{R}_{[k]} \cdot V_{[k-1]}^T}_{\delta E_{[k]}^4}. \end{aligned}$$

Following the error analysis in [7],

$$\|\delta E^4_{[k]}\|_2 \leq \|\delta E^4_{[k]}\|_F \leq k_G\epsilon \cdot n(1+k_G\epsilon)^{n-1} \cdot \|\hat{R}_{[k]}\|_F \simeq k_G\epsilon \cdot n \cdot \|R\|_F,$$

where k_G is a constant (depending on the specific implementation of the Givens rotations), and n is the number of different stages for one single QR update.

Finally, in a *fourth step* (SVD steps + reorthogonalizations), a sequence of left and right transformations is applied. If we summarize these into left transformations $\Theta_{[k]}$ and $T_{[k]}$, and a right transformation $\Phi_{[k]}$, we can proceed as follows (introducing round-off errors $\delta R_{[k]}$ and $\delta V_{[k]}$):

$$\begin{aligned} R_{[k]} &= \mathbf{fl}\{\Theta^T_{[k]} \cdot \hat{R}_{[k]} \cdot \Phi_{[k]}\} = \Theta^T_{[k]} \cdot \hat{R}_{[k]} \cdot \Phi_{[k]} + \delta R_{[k]}, \\ V_{[k]} &= \mathbf{fl}\{T_{[k]} \cdot V_{[k-1]} \cdot \Phi_{[k]}\} = T_{[k]} \cdot V_{[k-1]} \cdot \Phi_{[k]} + \delta V_{[k]} \\ &= V_{[k-1]} \cdot \Phi_{[k]} + (T_{[k]} - I) \cdot V_{[k-1]} \cdot \Phi_{[k]} + \delta V_{[k]} \end{aligned}$$

and

$$\begin{aligned} \begin{bmatrix} \lambda \cdot R_{[k-1]} \cdot V^T_{[k-1]} \\ a^T_{[k]} \end{bmatrix} &= Q_{[k]} \cdot \begin{bmatrix} \Theta_{[k]} \cdot (\Theta^T_{[k]} \cdot \hat{R}_{[k]} \cdot \Phi_{[k]}) \\ 0 \end{bmatrix} \cdot (\Phi^T_{[k]} \cdot V^T_{[k-1]}) \\ &\quad + \delta E^1_{[k]} + \delta E^2_{[k]} + \delta E^3_{[k]} + \delta E^4_{[k]} \\ &= Q_{[k]} \cdot \left(\begin{bmatrix} \Theta_{[k]} \cdot R_{[k]} \\ 0 \end{bmatrix} + \begin{bmatrix} -\Theta_{[k]} \cdot \delta R_{[k]} \\ 0 \end{bmatrix} \right) \\ &\quad \cdot (V^T_{[k]} - \delta V^T_{[k]} - \Phi^T_{[k]} V^T_{[k-1]} (T_{[k]} - I)^T) \\ &\quad + \delta E^1_{[k]} + \delta E^2_{[k]} + \delta E^3_{[k]} + \delta E^4_{[k]} \\ &\simeq Q_{[k]} \cdot \begin{bmatrix} \Theta_{[k]} \cdot R_{[k]} \\ 0 \end{bmatrix} \cdot V^T_{[k]} \\ &\quad + \delta E^1_{[k]} + \delta E^2_{[k]} + \delta E^3_{[k]} + \delta E^4_{[k]} \\ &\quad \underbrace{+ Q_{[k]} \cdot \begin{bmatrix} -\Theta_{[k]} \cdot \delta R_{[k]} \\ 0 \end{bmatrix} \cdot V^T_{[k]}}_{\delta E^5_{[k]}} \\ &\quad \underbrace{+ Q_{[k]} \cdot \begin{bmatrix} -\Theta_{[k]} \cdot R_{[k]} \\ 0 \end{bmatrix} \cdot \delta V^T_{[k]}}_{\delta E^6_{[k]}} \\ &\quad \underbrace{+ Q_{[k]} \cdot \begin{bmatrix} -\Theta_{[k]} \cdot R_{[k]} \\ 0 \end{bmatrix} \cdot \Phi^T_{[k]} \cdot V^T_{[k-1]} \cdot (T_{[k]} - I)^T}_{\delta E^7_{[k]}} \\ &\quad + \mathcal{O}(\epsilon^2). \end{aligned}$$

Again applying the error analysis in [7] results in

$$\begin{aligned} \|\delta E^5_{[k]}\|_2 &\leq \|\delta E^5_{[k]}\|_F \\ &\simeq k_G\epsilon \cdot (2n-2)(1+k_G\epsilon)^{2n-3} \cdot \|\hat{R}_{[k]}\|_F \end{aligned}$$

$$\simeq k_G\epsilon \cdot (2n-2)\|R\|_F,$$

$$\begin{aligned}\|\delta E^6_{[k]}\|_2 &\leq \|\delta E^6_{[k]}\|_F\\ &\simeq k_G\epsilon \cdot (2n-2)(1+k_G\epsilon)^{2n-3} \cdot \|V_{[k]}\|_F \cdot \|\hat{R}_{[k]}\|_2\\ &\simeq k_G\epsilon \cdot (2n-2)\sqrt{n} \cdot \|\hat{R}_{[k]}\|_2\\ &\simeq k_G\epsilon \cdot (2n-2)\|R\|_F.\end{aligned}$$

The number of stages in the upper bounds for $\delta E^5_{[k]}$ and $\delta E^6_{[k]}$ equals $2n-2$, as $n-1$ rotations are applied both to the left and to the right.

As for $\delta E^7_{[k]}$, we can estimate an upper bound as follows:

$$\|\delta E^7_{[k]}\|_2 \leq \|T_{[k]} - I\|_F \cdot \|R_{[k]}\|_2.$$

Note that $\delta E^7_{[k]}$ represents an additional error which is introduced by the reorthogonalization scheme. The reorthogonalization indeed changes the V-matrix in an arbitrary manner with respect to the original data, and therefore contributes to $\Delta_{[k]}$. We can easily check that $\|T_{[k]} - I\|_F$ must have an upper bound similar to $\|V_{[k]}V^T_{[k]} - I\|_F$, so that finally

$$\begin{aligned}\|\delta E^7_{[k]}\|_2 &\leq k_T \cdot n\sqrt{n}\epsilon \cdot \|R_{[k]}\|_2\\ &\leq k_T \cdot n\epsilon\|R\|_F.\end{aligned}$$

Adding all the above upper bounds, we obtain

$$\begin{bmatrix} \lambda \cdot R_{[k-1]} \cdot V^T_{[k-1]} \\ a^T_{[k]} \end{bmatrix} = Q_{[k]} \cdot \begin{bmatrix} \Theta_{[k]} \cdot R_{[k]} \\ 0 \end{bmatrix} \cdot V^T_{[k]} + \underbrace{\delta E^1_{[k]} + \delta E^2_{[k]} + \delta E^3_{[k]} + \delta E^4_{[k]} + \delta E^5_{[k]} + \delta E^6_{[k]} + \delta E^7_{[k]}}_{\delta E^{1\to 7}_{[k]}}$$

with

$$\|\delta E^{1\to 7}_{[k]}\|_2 \leq (k_1 \cdot n + k_2)\epsilon \cdot \|R\|_F.$$

Multiplying the left- and right-hand sides with their transpose now results in

$$\lambda^2 \cdot (R_{[k-1]} \cdot V^T_{[k-1]})^T \cdot (R_{[k-1]} \cdot V^T_{[k-1]}) + a_{[k]} \cdot a^T_{[k]} = (R_{[k]} \cdot V^T_{[k]})^T \cdot (R_{[k]} \cdot V^T_{[k]}) + \delta E_{[k]},$$

where

$$\begin{aligned}\delta E_{[k]} = {} & (\delta E^{1\to 7}_{[k]})^T \cdot \left(Q_{[k]} \cdot \begin{bmatrix} \Theta_{[k]} \cdot R_{[k]} \\ 0 \end{bmatrix} \cdot V^T_{[k]}\right)\\ & + \left(Q_{[k]} \cdot \begin{bmatrix} \Theta_{[k]} \cdot R_{[k]} \\ 0 \end{bmatrix} \cdot V^T_{[k]}\right)^T \cdot (\delta E^{1\to 7}_{[k]}) + \mathcal{O}(\epsilon^2)\end{aligned}$$

with an upper bound

$$\begin{aligned}\|\delta E_{[k]}\|_F &\simeq 2\|R_{[k]}\|_F \cdot \|\delta E^{1\to 7}_{[k]}\|_2\\ &\simeq 2(k_1 \cdot n + k_2)\epsilon \cdot \|R\|^2_F.\end{aligned}$$

Substituting the definition of $\Delta_{[\cdot]}$ results in

$$\lambda^2 \cdot A_{[k-1]}^T \cdot A_{[k-1]} - \lambda^2 \cdot \Delta_{[k-1]} + a_{[k]} \cdot a_{[k]}^T = (R_{[k]} \cdot V_{[k]}^T)^T \cdot (R_{[k]} \cdot V_{[k]}^T) + \delta E_{[k]},$$
$$A_{[k]}^T \cdot A_{[k]} - \lambda^2 \cdot \Delta_{[k-1]} = (R_{[k]} \cdot V_{[k]}^T)^T \cdot (R_{[k]} \cdot V_{[k]}^T) + \delta E_{[k]},$$

and finally

$$\Delta_{[k]} = \lambda^2 \cdot \Delta_{[k-1]} + \delta E_{[k]},$$

or if we use norms,

$$\begin{aligned}\|\Delta_{[k]}\|_F &\le \lambda^2 \cdot \|\Delta_{[k-1]}\|_F + \|\delta E_{[k]}\|_F \\ &\le \lambda^2 \cdot \|\Delta_{[k-1]}\|_F + 2(k_1 \cdot n + k_2)\epsilon \cdot \|R\|_F^2.\end{aligned}$$

The first term, $\lambda^2 \cdot \Delta_{[k-1]}$, represents the error propagation, which is stable as $\lambda < 1$. The second term is an upper bound for local errors. If we assume that the weighting factor is constant, we finally obtain (for all values of k)

$$\|\Delta_{[k]}\|_F \le \frac{2(k_1 \cdot n + k_2)}{1 - \lambda^2} \cdot \|R\|_F^2 \cdot \epsilon.$$

or alternatively (a kind of relative error formulation),

$$\frac{\|(R_{[k]} \cdot V_{[k]}^T)^T \cdot (R_{[k]} \cdot V_{[k]}^T) - (R_{[k]} \cdot V_{[k]}^T)^T \cdot (R_{[k]} \cdot V_{[k]}^T)\|_F}{\|R\|_F^2} \le \frac{2(k_1 \cdot n + k_2)}{1 - \lambda^2} \cdot \epsilon.$$

In conclusion, the overall SVD updating scheme is found to be stable, if an exponential weighting is applied, with weighting factor $\lambda < 1$, and if a reorthogonalization procedure is included, which keeps the stored V-matrix close to orthogonal. The obtained upper bound for the error $\|\Delta_{[k]}\|_F$ is clearly overly conservative, as it consists of an accumulation of several worst-case upper bounds.

5. Conclusion. An SVD updating procedure was constructed as a combination of QR updating and a Jacobi-type SVD algorithm applied to a triangular matrix. As for subspace tracking problems, it was shown how only very few SVD steps after each QR updating can restore an acceptable approximation. Furthermore, the updating is shown to be stable when supplemented with a Jacobi-type reorthogonalization scheme. A systolic array for this updating algorithm is developed in [18].

Appendix A. Let us assume that the initial configuration is as follows (we consider a small 6×6 example, from which the results for the general case obviously follow):

$$R_{[k]} = \begin{bmatrix} Rs_{[k]} & Rsn_{[k]} \\ 0 & Rn_{[k]} \end{bmatrix}$$
$$= \begin{bmatrix} r_{11}^s & r_{12}^s & r_{13}^s & \boxed{\varepsilon_{14}} & \boxed{\varepsilon_{15}} & \boxed{\varepsilon_{16}} \\ & r_{22}^s & r_{23}^s & \boxed{\varepsilon_{24}} & \boxed{\varepsilon_{25}} & \boxed{\varepsilon_{26}} \\ & & r_{33}^s & \boxed{\varepsilon_{34}} & \boxed{\varepsilon_{35}} & \boxed{\varepsilon_{36}} \\ & & & r_{44}^n & r_{45}^n & r_{46}^n \\ & & & & r_{55}^n & r_{56}^n \\ & & & & & r_{66}^n \end{bmatrix}.$$

Time indices are omitted for brevity. We assume that

$$\varepsilon \ll \delta,$$

where

$$\varepsilon = \|Rsn_{[k]}\|_F$$

is the Frobenius norm of the matrix with cross terms, and

$$\delta = \sigma_{\min}\{Rs_{[k]}\} - \sigma_{\max}\{Rn_{[k]}\}$$

is the gap between the singular values of $Rs_{[k]}$ and $Rn_{[k]}$. As these are then known to be (ε^2/δ)-close to the singular values of R [3], we end up with

$$\delta \simeq \sigma_{\min}\{Rs_{[k]}\} \quad \text{for } SN \gg 1.$$

The aim is to investigate the effect on ε of one cyclic-by-rows sweep in Kogbetliantz's SVD algorithm (modified for triangular matrices) This essentially consists of a number of 2×2 SVDs on the main diagonal, where the pivot index takes up the values (see [15] and [23] for details)

$$\begin{aligned} i = {} & 1,2,3,4,5 \\ & 1,2,3,4 \\ & 1,2,3 \\ & 1,2 \\ & 1. \end{aligned}$$

As a reminder, each 2×2 SVD can be described as

$$\begin{bmatrix} r_{i,i} & 0 \\ 0 & r_{i+1,i+1} \end{bmatrix} \Leftarrow \begin{bmatrix} \sin\theta & \cos\theta \\ \cos\theta & -\sin\theta \end{bmatrix} \begin{bmatrix} r_{i,i} & r_{i,i+1} \\ 0 & r_{i+1,i+1} \end{bmatrix} \begin{bmatrix} \sin\phi & \cos\phi \\ \cos\phi & -\sin\phi \end{bmatrix},$$

where

$$\tan 2\theta = \frac{2r_{i+1,i+1} \cdot r_{i,i+1}}{r_{i,i}^2 - r_{i+1,i+1}^2 + r_{i,i+1}^2},$$

$$\tan\phi = \frac{r_{i+1,i+1} \cdot \tan\theta + r_{i,i+1}}{r_{i,i}}$$

(for the sake of clarity, we prefer to use inner rotations + permutations, instead of outer rotations).

First of all, we can slightly reorder the 2×2 transformations as follows:

$$\text{Part (a)}\left\{ \begin{array}{lcl} i & = & 1,2 \\ & & 1, \end{array}\right.$$

$$\text{Part (b)}\left\{ \begin{array}{lcl} i & = & .,.,3,4,5 \\ & & .,2,3,4 \\ & & 1,2,3, \end{array}\right.$$

$$\text{Part (c)}\left\{ \begin{array}{lcl} i & = & 1,2 \\ & & 1. \end{array}\right.$$

Part (a) corresponds to rotations within the (approximate) signal subspace, reducing the off-norm in $Rs_{[k]}$. Both ε and δ remain unchanged, so that for the time being, these transformations are irrelevant.

Part (b) corresponds to annihilations of cross terms, and needs further investigation. Referring to the initial configuration (which is basically not changed in Part (a)), let us first remark that the diagonal entries in $Rs_{[k]}$ satisfy

$$r^s_{ii} \geq \sigma_{\min}\{Rs_{[k]}\}$$

(as $Rs_{[k]}$ is triangular), while on the other hand the diagonal entries in $Rn_{[k]}$ obviously satisfy

$$r^n_{ii} \leq \sigma_{\max}\{Rn_{[k]}\}.$$

We easily verify that the above upper and lower bounds remain valid throughout the computations in Part (b) (r^s_{ii} elements always increase; r^n_{ii} elements decrease).

The *first series of transformations*, where the pivot index takes up the values

$$i = .,.,3,4,5$$

turns the initial configuration into

$$\begin{bmatrix} r^s_{11} & r^s_{12} & \boxed{\varepsilon_{13}} & \boxed{\varepsilon_{14}} & \boxed{\varepsilon_{15}} & r^s_{16} \\ & r^s_{22} & \boxed{\varepsilon_{23}} & \boxed{\varepsilon_{24}} & \boxed{\varepsilon_{25}} & r^s_{26} \\ & & r^n_{33} & r^n_{34} & r^n_{35} & \boxed{\varepsilon^\star_{36}} \\ & & & r^n_{44} & r^n_{45} & \boxed{\varepsilon^\star_{46}} \\ & & & & r^n_{55} & \boxed{\varepsilon^\star_{56}} \\ & & & & & r^s_{66} \end{bmatrix}$$

(iteration indices are left out for the sake of clarity; subscripts ij refer to row and column numberings in the full R-matrix).

When $i = 3$, the pivot element ε_{34} is ε-small, from which we can estimate the rotation angles as follows:

$$\sin\theta \simeq \tan\theta \simeq \frac{\sigma_{\max}\{Rn_{[k]}\}}{\sigma^2_{\min}\{Rs_{[k]}\}} \cdot \mathcal{O}(\varepsilon),$$

$$\sin\phi \simeq \tan\phi \simeq \frac{1}{\sigma_{\min}\{Rs_{[k]}\}} \cdot \mathcal{O}(\varepsilon).$$

From the fact that θ (row transformation) is particularly small, it follows that the pivot for $i = 4$ is ε-small as well:

$$\begin{aligned}\varepsilon_{35} \cdot \cos\theta - r^n_{45} \cdot \sin\theta &\simeq \mathcal{O}(\varepsilon) + \sigma_{\max}\{Rn_{[k]}\} \cdot \frac{\sigma_{\max}\{Rn_{[k]}\}}{\sigma^2_{\min}\{Rs_{[k]}\}} \cdot \mathcal{O}(\varepsilon) \\ &\simeq \mathcal{O}(\varepsilon).\end{aligned}$$

Finally, from a similar reasoning it follows that the pivot for $i = 5$ is also ε-small. The estimates for $\tan\theta$ and $\tan\phi$ therefore hold for $i = 3$ as well as for $i = 4, 5$.

From the estimates for the rotation angles, we can estimate the magnitude of r^n_{ij}, $\varepsilon^\star_{ij}$, ε_{ij} after the first series of transformations.

Elements r^n_{ij} remain $\sigma_{\max}\{Rn_{[k]}\}$-small, which follows from

$$\begin{aligned} r^n_{ij}\cdot\cos\theta+\varepsilon^n_{kj}\cdot\sin\theta &\simeq \mathcal{O}(\sigma_{\max}\{Rn_{[k]}\})+\frac{\sigma_{\max}\{Rn_{[k]}\}}{\sigma^2_{\min}\{Rs_{[k]}\}}\cdot\mathcal{O}(\varepsilon^2)\\ &\simeq \mathcal{O}(\sigma_{\max}\{Rn_{[k]}\}). \end{aligned}$$

Cross terms $\varepsilon^\star_{ij}$ are (ε/SN)-small, which follows from

$$\begin{aligned} \varepsilon^\star_{ij} &\simeq \sum_k r^n_{ik}\cdot\sin\phi\\ &\simeq \frac{\sigma_{\max}\{Rn_{[k]}\}}{\sigma_{\min}\{Rs_{[k]}\}}\cdot\mathcal{O}(\varepsilon)\\ &\simeq \mathcal{O}\left(\frac{\varepsilon}{SN}\right). \end{aligned}$$

Cross terms ε_{ij} remain ε-small, which follows from

$$\begin{aligned} \varepsilon_{ij} &\simeq \varepsilon_{ij}+\sum_k r^s_{ik}\cdot\sin\phi\\ &\simeq \mathcal{O}(\varepsilon)+\frac{\mathcal{O}(\|Rs_{[k]}\|_{\text{off}})}{\sigma_{\min}\{Rs_{[k]}\}}\cdot\mathcal{O}(\varepsilon)\\ &\simeq \mathcal{O}(\varepsilon), \end{aligned}$$

where we made an opportunist (but mostly fair) assumption, namely, that $r^s_{ik}=\mathcal{O}(\|Rs_{[k]}\|_{\text{off}})\le\sigma_{\min}\{Rs_{[k]}\}$. This corresponds to a certain degree of convergence within $Rs_{[k]}$, brought about by Part (a).

We can now repeat this reasoning for the *second and the third series of transformations* in Part (b)

$$\begin{aligned} i = {}&.,2,3,4\\ &1,2,3, \end{aligned}$$

in turn transforming the triangular factor into

$$\begin{bmatrix} r^s_{11} & \boxed{\varepsilon_{12}} & \boxed{\varepsilon_{13}} & \boxed{\varepsilon_{14}} & r^s_{15} & r^s_{16}\\ & r^n_{22} & r^n_{23} & r^n_{24} & \boxed{\varepsilon^\star_{25}} & \boxed{\varepsilon^\star_{26}}\\ & & r^n_{33} & r^n_{34} & \boxed{\varepsilon^\star_{35}} & \boxed{\varepsilon^\star_{36}}\\ & & & r^n_{44} & \boxed{\varepsilon^\star_{45}} & \boxed{\varepsilon^\star_{46}}\\ & & & & r^s_{55} & r^s_{56}\\ & & & & & r^s_{66} \end{bmatrix}$$

and

$$\begin{bmatrix} r^n_{11} & r^n_{12} & r^n_{13} & \boxed{\varepsilon^\star_{14}} & \boxed{\varepsilon^\star_{15}} & \boxed{\varepsilon^\star_{16}}\\ & r^n_{22} & r^n_{23} & \boxed{\varepsilon^\star_{24}} & \boxed{\varepsilon^\star_{25}} & \boxed{\varepsilon^\star_{26}}\\ & & r^n_{33} & \boxed{\varepsilon^\star_{34}} & \boxed{\varepsilon^\star_{35}} & \boxed{\varepsilon^\star_{36}}\\ & & & r^s_{44} & r^s_{45} & r^s_{46}\\ & & & & r^s_{55} & r^s_{56}\\ & & & & & r^s_{66} \end{bmatrix},$$

where all cross terms ε^*_{ij} are seen to remain (ε/SN)-small.

Finally, subsequent transformations in Part (c) do not alter the norm of the submatrix with cross terms.

As a main conclusion, we can state that the matrix with cross terms is (ε/SN)-small after the forward sweep. The backward sweep, returning the triangular matrix to the original configuration, again reduces this norm by a factor $1/SN$ (where this time the column transformations are particularly small). A double sweep thus corresponds to a reduction by a factor $(1/SN)^2$.

REFERENCES

[1] J. R. BUNCH AND C. P. NIELSEN, *Updating the singular value decomposition*, Numer. Math., 31 (1978), pp. 111–129.

[2] J. R. BUNCH, C. P. NIELSEN, AND D. C. SORENSEN, *Rank one modification of the symmetric eigenproblem*, Numer. Math., 31 (1978), pp. 31–48.

[3] J. P. CHARLIER AND P. VAN DOOREN, *On Kogbetliantz's* SVD *algorithm in the presence of clusters*, Linear Algebra Appl., 95 (1987), pp. 135–160.

[4] P. COMON AND G. H. GOLUB, *Tracking a few extreme singular values and vectors in signal processing*, Proc. IEEE, 78 (1990), pp. 1327–1343.

[5] K. V. FERNANDO, *Linear convergence of the cyclic Jacobi and Kogbetliantz methods*, Numer. Math., 56 (1989), pp. 73–91.

[6] G. E. FORSYTHE AND P. HENRICI, *The cyclic Jacobi method for computing the principal values of a complex matrix*, Trans. Amer. Math. Soc., 94 (1960), pp. 1–23.

[7] W. M. GENTLEMAN, *Error analysis of* QR *decompositions by Givens transformations*, Linear Algebra Appl., 10 (1975), pp. 189–197.

[8] P. E. GILL, G. H. GOLUB, W. MURRAY, AND M. A. SAUNDERS, *Methods for modifying matrix factorizations*, Math. Comp., 28 (1974), pp. 505–535.

[9] G. H. GOLUB, *Some modified eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–334.

[10] G. H. GOLUB AND C. F. VAN LOAN, *Matrix computations*, Second Edition, The Johns Hopkins University Press, Baltimore, MD, 1989.

[11] V. HARI AND K. VESELIĆ, *On Jacobi methods for singular value decompositions*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 741–754.

[12] 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. Statist. Comput., 7 (1986), pp. 1147–1159.

[13] P. HENRICI AND K. ZIMMERMANN, *An estimate for the norms of certain cyclic Jacobi operators*, Linear Algebra Appl., 1 (1968), pp. 489–501.

[14] E. KOGBETLIANTZ, *Solution of linear equations by diagonalization of coefficient matrices*, Quart. Appl. Math., 13 (1955), pp. 123–132.

[15] F. T. LUK, *A triangular processor array for computing singular values*, Linear Algebra Appl., 77 (1986), pp. 259–273.

[16] F. T. LUK AND H. PARK, *On the equivalence and convergence of parallel Jacobi* SVD *algorithms*, Proc. SPIE, Vol. 826, Advanced Algorithms and Architectures for Signal Processing II, F. T. Luk, ed., 1987, pp. 152–159.

[17] M. MOONEN, B. DE MOOR, L. VANDENBERGHE, AND J. VANDEWALLE, *On- and off-line identification of linear state space models*, Internat. J. Control, 49 (1989), pp. 219–232.

[18] M. MOONEN, P. VAN DOOREN, AND J. VANDEWALLE, *A systolic array for* SVD *updating*, ESAT-SISTA Report 1989-13b, Department of Electrical Engineering, Katholieke Universiteit Leuven, Belgium, SIAM J. Matrix Anal. Appl., 14 (1993), to appear.

[19] C. C. PAIGE AND P. VAN DOOREN, *On the convergence of Kogbetliantz's algorithm for computing the singular value decomposition*, Linear Algebra Appl., 77 (1986), pp. 301–313.

[20] R. SCHREIBER, *Implementation of adaptive array algorithms*, IEEE Trans. Acoust. Speech Signal Process., 34 (1986), pp. 1038–1045.

[21] J. M. SPEISER, *Signal processing computational needs*, Proc. SPIE, Vol 696, Advanced Algorithms and Architectures for Signal Processing, J. M. Speiser, ed., 1986, pp. 2–6.

[22] G. W. STEWART, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Rev., 15 (1973), pp. 727–764.

[23] G. W. STEWART, *A Jacobi-like algorithm for computing the Schur decomposition of a nonhermitian matrix*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 853–863.
[24] J. H. WILKINSON, *Note on the quadratic convergence of the cyclic Jacobi process*, Numer. Math., 4 (1962), pp. 296–300.
[25] ———*The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.