Schur method for low-rank matrix approximation

Alle-Jan van der Veen

Stanford University, Dept. Comp. Sci./SCCM
Stanford, Palo Alto, CA 94305-2140
email: allejan@sccm.stanford.edu

The usual way to compute a low-rank approximant of a matrix H is to take its truncated SVD. However, the SVD is computationally expensive. This paper describes a much simpler generalized Schur-type algorithm to compute similar low-rank approximants. For a given matrix H which has d singular values larger than ε, we find all rank d approximants $\hat{H}$ such that $H-\hat{H}$ has 2-norm less than ε. The set of approximants includes the truncated SVD approximation. The advantages of the Schur algorithm are that it has a much lower computational complexity (similar to a QR factorization), and directly produces estimates of the column space of the approximants. This column space can be updated and downdated in an on-line scheme, amenable to implementation on a parallel array of processors.

Keywords: matrix approximation, rank revealing factorizations, hyperbolic rotations.

1. INTRODUCTION AND MOTIVATION

We consider the following problem. For a given matrix H and tolerance level ε, describe all matrices $\hat{H}$ such that

$$\begin{array}{ll}(a) & \|H-\hat{H}\| \le \varepsilon, \\ (b) & \mathrm{rank}(\hat{H}) = d, \end{array} \tag{1}$$

where d is equal to the number of singular values of H that are larger than ε. ($\|\cdot\|$ denotes the matrix 2-norm.) Such a matrix $\hat{H}$ is a low-rank approximation of H in 2-norm.

The above approximation problem is relevant in signal processing, where many analysis algorithms have a stage in which a data matrix is constructed that is supposed to be of low rank (or rank deficient), but because of noise, this property is lost. A simple, generic, example is given by the overdetermined system of equations

$$Ax = b \qquad \Leftrightarrow \qquad [A \quad b]\begin{bmatrix} x \\ -1 \end{bmatrix} = 0 \qquad ([A \;\; b] : n \times m,\ n > m).$$

For a solution to exist, the matrix $[A \;\; b]$ has to be rank deficient by at least 1. If it is, then the solution is determined by the kernel. Although a one-dimensional kernel is obviously sufficient, in many applications, A is itself supposed to be of low rank, so that we want to go further and ascertain that $[A \;\; b]$ actually has low rank. The solution vector is any of the vectors in the multi-dimensional kernel of $[A \;\; b]$, of which for example the one with minimal norm $\|x\|$ is chosen. With noise added to $[A \;\; b]$, the problem becomes a total least squares problem (one-dimensional kernel) or a generalization thereof: approximate $H = [A \;\; b]^H$ by some $\hat{H} = [\hat{A} \;\; \hat{b}]^H$ such that $\hat{H}$ is rank deficient (or low rank), and find a description of its column space and the complement of the column space (the kernel of $[\hat{A} \;\; \hat{b}]$). This is one of the basic, noise-reducing, steps in subspace based system identification, harmonic retrieval, or high-resolution direction-of-arrival problems [1].

The usual solution of this approximation problem is to compute an SVD of H, determine the number of singular values that are significantly smaller than the others, or that are smaller than a threshold ε determined by the SNR, and set those singular values to zero. The resulting matrix is a low-rank approximant $\hat{H}$, known as the truncated SVD solution (TSVD). It is optimal in the Frobenius-norm, in the sense that it minimizes $\|H-\hat{H}\|_F$ under the condition that $\hat{H}$ has a certain rank. However, there are a few remarks to be made:

1. Computing a TSVD is complex. We have to compute an SVD, even if in the end we only use the kernel or column space of the approximant. Although continuing efforts have rendered the computation of an SVD to be only a factor of 2–3 more expensive than a QR, aspects of updating and regularity of the computations also have to be taken into account.

SPIE 1994, "Advanced Signal Processing Algorithms, Architectures, and Implementations V", vol. 2296

2. It is not necessarily correct. The truncated SVD solution leads to a residual error (the estimated noise matrix) that is singular, too. This is in general not a correct noise model. E.g., if the noise is i.i.d. and zero mean white, it has a covariance matrix $\sigma^2 I$, and asymptotically, the singular values of the noise matrix are given by $\sigma n^{1/2}$. Hence, a more appropriate approximation is obtained by reducing the principal singular values by $\sigma n^{1/2}$. Because the singular vectors do not change, this does not affect the LS application that we have in mind, but it shows that the TSVD approximant *per se* is not necessarily the right choice.

For a finite amount of data so that asymptotic properties do not hold, all we can really say is that we want to find an approximant $\hat{H}$ that has as low rank as possible and is such that the residual error (the norm of $H - \hat{H}$, *i.e.*, the noise matrix) is smaller than $\varepsilon = \sigma n^{1/2}$. In view of this, the proposed approximation problem (1) makes sense. The key point in this problem formulation is that it does not ask for an approximant $\hat{H}$ of rank d that *minimizes* $\| H - \hat{H} \|$, but rather one in which the approximation error is limited by a specified upper bound. Such approximants can be computed with significantly less effort: the generalized Schur method described in this paper does not require knowledge of the singular values, but for a given ε produces bases for the signal subspace and null space using only $O(1/2\, m^2 n)$ operations, the same as a QR factorization of H would require.

Indeed, the Schur method can be thought of as an RQ factorization of a matrix $[\varepsilon I \;\; H]$, but using a J-unitary matrix Θ rather than a unitary matrix Q in the factorization. This factorization provides an implicit decomposition of $HH^H - \varepsilon^2 I$ into a positive semidefinite and a negative semidefinite matrix. The positive matrix corresponds to singular values of H that are larger than ε, and its rank is equal to their number, d. The negative term has rank $m-d$ and corresponds to the singular values that are smaller than ε. After computation of the hyperbolic QR factorization, the column space of the approximants is known: a basis of it is a specific subset of the columns of the R matrix in the factorization. The computation of an approximant itself requires also the inversion of a submatrix of the J-unitary factor. In addition, there is a closed-form formula which describes the set of all possible 2-norm approximants of rank d, in terms of free parameter S_L. Several choices of S_L lead to interesting results. The approximant obtained for $S_L = 0$ is the easiest to compute. For one value of S_L, the TSVD approximant is obtained, but computing this value is prohibitive. Other choices lead to approximants that have certain 'unbiased-ness' properties, or approximants for which the residual error is a full-rank matrix.

In the past few years, a number of other methods have been developed to alleviate the computational burden of the SVD, yet retaining important information such as rank and principal subspaces. Some of these techniques are the URV decomposition [2], and the rank revealing QR decomposition (RRQR) [3, 4, 5]. Recently, there has been an increased interest in updating techniques for the SVD and URV decomposition, which converge to the exact SVD or URV under certain stationarity conditions [6, 7]. It should be noted that all these decompositions require $O(\alpha m^2 n)$ operations, for an $m \times n$ matrix, where α is a multiplication constant which is high (≈ 10) for an exact SVD and much lower for the URV, RRQR and updating techniques. The main difference in the proposed Schur-type technique and the URV and RRQR methods lies in the simplicity and uniformity of the operations. The URV decomposition and rank revealing QR methods are iterative and require estimates of the conditioning of certain submatrices at every step of the iteration. This estimation is a global operation which is not amenable to parallel implementation, and the precise number of operations is dependent on the entries of the data matrix. SVD and URV updating algorithms as in [7] are parallel but iterative schemes which converge to the SVD or URV. Their projected use is in (adaptive) signal processing application. However, in these applications, knowledge of the singular values is only used to determine the noise level, and only the principal singular vectors (spanning the signal subspace) are retained. If, in these applications, the noise level is already approximately known, then the Schur algorithm is a viable candidate which is parallel but non-iterative.

It should be noted that Schur methods *an sich* are well known. Originally, Schur [8] devised this algorithm to test whether a polynomial is bounded within the complex unit disc. Schur algorithms occur in certain constrained interpolation problems, rational approximation by positive real functions, factorization and inversion of positive definite Toeplitz matrices [9], and have been generalized in a number of senses. A generalization that comes close to the description here is by Dewilde and Deprettere [10], for Schur-parametrizations of positive definite matrices, and by Diepold and Pauli [11], for indefinite matrix cases. In [10], the Schur parametrization was used for Cholesky factorizations and for approximating the inverse of positive definite matrices by banded matrices, in Frobenius norm. However, the present application to *low rank* matrix approximation has been unknown so far. It is a special case of a time-varying Hankel-norm model reduction theory developed by Dewilde and Van der Veen [12]. In the linear algebra community, the related J-unitary transformations are well known and widely used, but mainly for

downdating Cholesky factors of definite matrices (*e.g.,* [13, 14]). Indefinite factorizations, as in this paper, are rarely studied and even avoided, because the loss of positivity leads to a breakdown of the Cholesky factorization. Some exceptions are [15, 16].

2. J-UNITARY MATRICES

Some theory of J-unitary matrices is required, which we summarize at this point. A signature matrix $\tilde{J}$ is a diagonal matrix with diagonal entries equal to +1 or −1. A matrix $\tilde{\Theta}$ is J-unitary if it satisfies

$$(\tilde{\Theta})^H \tilde{J}_1 \tilde{\Theta} = \tilde{J}_2 , \qquad \tilde{\Theta} \tilde{J}_2 (\tilde{\Theta})^H = \tilde{J}_1 , \tag{2}$$

for signature matrices $\tilde{J}_1, \tilde{J}_2$. Usually, the entries of a signature matrix are sorted into a positive and a negative block, and we partition Θ accordingly as

$$\Theta = \begin{bmatrix} \Theta_{11} & \Theta_{12} \\ \Theta_{21} & \Theta_{22} \end{bmatrix} , \qquad J = \begin{bmatrix} I & \\ & -I \end{bmatrix} \tag{3}$$

(for identity matrices of appropriate sizes). We will denote an unsorted signature matrix by a tilde. If the signature matrices are sorted, then conservation of inertia gives $J_1 = J_2$ ($= J$, say). The J-unitarity of Θ implies a.o. that it is invertible: $\Theta^{-1} = J\Theta J$, and

$$\begin{aligned} \Theta_{11}^H \Theta_{11} &= I + \Theta_{21}^H \Theta_{21} & \qquad \Theta_{22}^H \Theta_{22} &= I + \Theta_{12}^H \Theta_{12} \\ \Theta_{11} \Theta_{11}^H &= I + \Theta_{12} \Theta_{12}^H & \qquad \Theta_{22} \Theta_{22}^H &= I + \Theta_{21} \Theta_{21}^H . \end{aligned}$$

Hence, Θ_{11} and Θ_{22} are invertible, and

$$\| \Theta_{11}^{-1} \| \le 1 , \qquad \| \Theta_{11}^{-1} \Theta_{12} \| < 1 , \qquad \| \Theta_{22}^{-1} \| \le 1 , \qquad \| \Theta_{12} \Theta_{22}^{-1} \| < 1 . \tag{4}$$

Suppose that X and Y are matrices, related by a J-unitary matrix $\tilde{\Theta}$ as $X\tilde{\Theta} = Y$. Then X and Y satisfy the "energy equation"

$$X \tilde{J}_1 X^H = Y \tilde{J}_2 Y^H .$$

Motivated by this equation, we say that $\tilde{J}_1$ associates a signature to the columns of X, and likewise, that $\tilde{J}_2$ is the signature of the columns of Y. Because X can be viewed as an input matrix, which is mapped by $\tilde{\Theta}$ to a resulting output matrix Y, we will sometimes call $\tilde{J}_1$ the "input signature" of $\tilde{\Theta}$, to distinguish it from $\tilde{J}_2$.

3. APPROXIMATION THEORY

Let $H : m \times n$ be a given matrix and ε be a given tolerance level, and suppose that H has d singular values larger than ε and none equal to ε. We will show that there exists a J-unitary matrix $\tilde{\Theta}$ (which under additional conditions can be computed by a generalized Schur algorithm) such that

$$\begin{matrix} + & - & & +/- & +/- \end{matrix}$$
$$\begin{bmatrix} \varepsilon I_m & H \end{bmatrix} \tilde{\Theta} = \begin{bmatrix} X & 0_{m \times n} \end{bmatrix} . \tag{5}$$

X is an $m \times m$ matrix. $\tilde{\Theta}$ is J-unitary with respect to $(J_1, \tilde{J}_2)$, where the signature matrix J_1 is specified beforehand and is sorted:

$$J_1 = \begin{bmatrix} I_m & \\ & -I_n \end{bmatrix} .$$

J_1 associates to the columns of εI_m in $[\varepsilon I_m \;\; H]$ a positive signature, and to columns of H a negative signature. The signature matrix $\tilde{J}_2$ is produced by the algorithm and is unsorted. The signature associated to the i-th column of $[X \;\; 0]$ is equal to $(\tilde{J}_2)_{ii}$. Let Π be a permutation matrix that sorts $\tilde{J}_2$, *i.e.,* such that $\Pi^H \tilde{J}_2 \Pi = J_2$. By preservation of inertia, $J_2 = J_1 =: J$, and putting $\Theta = \tilde{\Theta}\Pi$ yields $[\varepsilon I_m \;\; H]\Theta = [(A \;\; 0) \;\; (B \;\; 0)]$, where A contains the columns of X that have positive signature, and B contains the columns of X that have negative signature. $[A \;\; 0]$ is an $m \times m$ matrix, $[B \;\; 0]$ is $m \times n$. At this point, we partition Θ into 2×2 blocks as in (3), and define $\hat{H} = [B \;\; 0]\Theta_{22}^{-1}$. We will show that $\hat{H}$ is a rank-d 2-norm approximant of H.

Theorem 1. *Let $H : m \times n$ have d singular values larger than ε and none equal to ε. Then there is a J-unitary matrix Θ such that*

$$[\varepsilon I_m \quad H]\Theta = [A' \quad B'], \qquad \begin{matrix} & m-d & d \\ A' = m & \left[\begin{matrix} A & 0 \end{matrix} \right. \end{matrix}, \qquad \begin{matrix} & d & n-d \\ B' = m & \left[\begin{matrix} B & 0 \end{matrix} \right. \end{matrix}. \tag{6}$$

$\hat{H} = [B \;\; 0]\Theta_{22}^{-1}$ is a rank d approximant such that $\| H - \hat{H} \| \le \varepsilon$.

PROOF Consider $\varepsilon^2 I - HH^H$. It is non-singular by assumption, and hence there is a J-Cholesky factorization such that

$$\varepsilon^2 I - HH^H = XJ'X^H,$$

where X is an $m \times m$ factor, and has full rank m. Put $X = [A \;\; B]$, partitioned according to J', so that $XJ'X^H = AA^H - BB^H$. Since $[\varepsilon I \;\; H]$ has full range, there must be an $n \times m$ matrix, T say, mapping it to X, i.e. $[\varepsilon I \;\; H]T = X$. Since X is also of full rank, it follows that $TJ'T^H = J$. T can be extended to a square invertible J-unitary matrix Θ such that (6) holds.

Let $H = U\Sigma V^H$ be an SVD of H. Then $(\varepsilon^2 I - \Sigma^2)$ has the same signature as $AA^H - BB^H$: d negative entries, and $m-d$ positive entries. Hence, A has $m-d$ columns and is of full rank, while B has d columns and is of full rank. By equation (6), $[B \;\; 0] = \varepsilon I\Theta_{12} + H\Theta_{22}$, so that $H - \hat{H} = -\varepsilon\Theta_{12}\Theta_{22}^{-1}$, and $\Theta_{12}\Theta_{22}^{-1}$ is contractive (equation (4)). Hence $\| H - \hat{H} \| \le \varepsilon$. ∎

Remark that $[A \;\; B]$ as generated in (6) is not unique: for any J-unitary matrix Θ_1, $[A_1 \;\; B_1] = [A \;\; B]\Theta_1$ can also be produced as a result of the Schur method. A systematic way to describe all possible 2-norm approximants of rank d is given by a parametrized chain fraction description in the following theorem. The parametrization is in terms of an $m \times n$ matrix S_L, which has the following 2×2 block partitioning:

$$S_L = \begin{matrix} & d & n-d \\ m-d & (S_L)_{11} & (S_L)_{12} \\ d & (S_L)_{21} & (S_L)_{22} \end{matrix}. \tag{7}$$

Theorem 2. **([17])** *Let $H : m \times n$ be a given matrix, with d singular values larger than ε and none equal to ε. Define Θ, A', B' as in equation (6). Suppose that a matrix $\hat{H}$ satisfies*

$$\begin{matrix} (a) & \| H - \hat{H} \| \le \varepsilon, \\ (b) & \mathrm{rank}(\hat{H}) \le d. \end{matrix}$$

Then $\mathrm{rank}(\hat{H}) = d$*, and $\hat{H} = H - S$ where*

$$S = \varepsilon(\Theta_{11}S_L - \Theta_{12})(\Theta_{22} - \Theta_{21}S_L)^{-1}, \tag{8}$$

for some S_L with $\| S_L \| \le 1$ and $(S_L)_{12} = 0$. $\hat{H}$ satisfies

$$\hat{H} = (B' - A'S_L)(\Theta_{22} - \Theta_{21}S_L)^{-1}. \tag{9}$$

The condition $\| S_L \| \le 1$ ensures that $\| S \| \le \varepsilon$, whereas taking $(S_L)_{12} = 0$ produces rank-d approximants. In particular, equation (9) shows that the column span of $\hat{H}$ is generated by the columns of $(B - A(S_L)_{11})$, which is of full rank d.

The approximant used before is obtained for $S_L = 0$. Other choices of S_L might be considered, in particular choices that minimize the error $\| S \|$. As the Frobenius-norm approximant, $\hat{H} = U_1\Sigma_1 V_1^H$, satisfies the conditions on $\hat{H}$, it is a suitable rank-d approximant, which actually minimizes the approximation error: $\| H - \hat{H} \| = \sigma_{d+1} < \varepsilon$. Hence, there must be some value of S_L (contractive, block lower) which minimizes the expression for S, although computing this S_L is as expensive as computing the SVD itself. Another useful choice for S_L, suitable if $d \ge m/2$, is $S_L = [I_m \;\; 0]$. For this choice, S is an isometry: the residual error matrix has full rank and its norm is precisely equal to ε.

One trivial case in which the optimal S_L can be computed is the case where all singular values of H are larger than ε. Indeed, suppose that H does not have singular values less than ε. Then H has rank $d = m$. The approximant $\hat{H} = B'\Theta_{22}^{-1}$, obtained for $S_L = 0$, is such that $\| H - \hat{H} \| \le \varepsilon$, and $\hat{H}$ is also of rank d. In fact, $\| H - \hat{H} \| = \| \varepsilon\Theta_{12}\Theta_{22}^{-1} \|$, which is, in general, larger than 0. However, there exists an approximant of rank d with zero error: H itself. Hence the 'central approximant', obtained for $S_L = 0$, is not the optimal (norm-minimizing) solution. As $\hat{H} = H$ is a valid approximant, there exists an S_L (contractive, block-lower)

such that $\hat{H} = H - S$ has $S = 0$. The expression for S leads to $S_L = \Theta_{11}^{-1}\Theta_{12}$. Indeed, S_L of this form is contractive (equation (4)). Verifying that $(S_L)_{12} = 0$ takes more effort, and is omitted at this point.

In general, H also has singular values less than ε, and we cannot take $S_L = \Theta_{11}^{-1}\Theta_{12}$ (although it is still contractive) because $(S_L)_{12}$ is not zero. Obviously, there is no approximant of zero error. A conjecture at this point is that by making small modifications to this S_L, to have it both contractive and block lower, we obtain approximants which have smaller errors than those obtained by just taking $S_L = 0$. We propose to take

$$S_L = \Theta_{11}^{-1}\Theta_{12} \begin{bmatrix} I_d & 0 \\ 0 & 0_{n-d} \end{bmatrix} \tag{10}$$

This choice of S_L is both contractive and block lower. Numerical experiments on a subspace estimation application indicate that this choice leads to estimates which have less bias than those obtained by taking $S_L = 0$, and are actually quite close to the principal column space of H (section 6). For subspace estimation, it is also interesting to note that the column space of $\hat{H}$, *i.e.*, the column space of $(B' - A'S_L)$, is not changed by setting $(S_L)_{12} = 0$ and $(S_L)_{22} = 0$, as in fact, the column space is $\mathcal{R}(B - A(S_L)_{11})$.

4. COMPUTATION OF Θ

We will now consider the actual construction of a J-unitary matrix Θ such that (6) holds. The general approach is similar to the usual Schur algorithms for unstructured matrices. In principle, the computations consist of elementary (Givens) rotations which recursively create zero entries at selected positions, much as in Givens rotations techniques for QR factorizations. The main differences with QR factorization, and also with the usual definite Schur algorithms (for which $\varepsilon^2 I - HH^H > 0$) are that, here, the basic operations are J-unitary elementary rotations of up to six different types, and that we have to keep track of signatures to determine which type to use. It turns out that the recursive construction of Θ in this way is not always possible, unless extra regularity conditions on the singular values of certain submatrices of H are posed. This is a well-known complication from which all indefinite Schur methods suffer and that can be treated only by global matrix operations (as in [11]).

Elementary rotations

An elementary rotation matrix is a 2×2 J-unitary matrix $\tilde{\theta}$ which is such that, for given scalars a, b, we have that $[a \quad b]\,\tilde{\theta} = [x \quad 0]$, where x is some resulting scalar. Here, we consider $\tilde{\theta}$ to be J-unitary with respect to unsorted signature matrices $\tilde{j}_1$ and $\tilde{j}_2$, where $\tilde{j}_1$ is a given signature matrix, with diagonal entries that are ±1, and $\tilde{j}_2$ is a resulting signature matrix to be computed along with $\tilde{\theta}$. The matrices $\tilde{\theta}$ and $\tilde{j}_2$ are computed from a, b, $\tilde{j}_1$ in the following way. From the J-unitarity of $\tilde{\theta}$, we have that

$$[a \quad b]\,\tilde{j}_1\,[a \quad b]^H = x(\tilde{j}_2)_{11}x^* \qquad \Rightarrow \qquad (\tilde{j}_2)_{11} = \text{sign}\left([a \quad b]\,\tilde{j}_1\,[a \quad b]^H\right).$$

We have to assume at this point that the expression in brackets is not zero, so that $(\tilde{j}_2)_{11}$ is either +1 or −1. The second diagonal entry of $\tilde{j}_2$ then follows from the inertia rule: since $\tilde{\theta}$ is invertible, the total number of positive entries of $\tilde{j}_2$ is equal to the total number of positive entries of $\tilde{j}_1$, and similarly for the negative entries.

It is straightforward to prove that the matrices $\tilde{\theta}$ in the following list are elementary J-unitary rotations with respect to the

specified signature matrices (taking $s^*s + c^*c = 1$ throughout):

$$
\begin{array}{lll}
1. & \tilde{j}_1 = \begin{bmatrix} 1 & \\ & -1 \end{bmatrix}, & \tilde{j}_2 = \begin{bmatrix} 1 & \\ & -1 \end{bmatrix}, \quad \tilde{\theta} = \begin{bmatrix} 1 & -s \\ -s^* & 1 \end{bmatrix} \frac{1}{c} \\
2. & \tilde{j}_1 = \begin{bmatrix} 1 & \\ & -1 \end{bmatrix}, & \tilde{j}_2 = \begin{bmatrix} -1 & \\ & 1 \end{bmatrix}, \quad \tilde{\theta} = \begin{bmatrix} -s^* & 1 \\ 1 & -s \end{bmatrix} \frac{1}{c} \\
3. & \tilde{j}_1 = \begin{bmatrix} -1 & \\ & 1 \end{bmatrix}, & \tilde{j}_2 = \begin{bmatrix} 1 & \\ & -1 \end{bmatrix}, \quad \tilde{\theta} = \begin{bmatrix} -s^* & 1 \\ 1 & -s \end{bmatrix} \frac{1}{c} \\
4. & \tilde{j}_1 = \begin{bmatrix} -1 & \\ & 1 \end{bmatrix}, & \tilde{j}_2 = \begin{bmatrix} -1 & \\ & 1 \end{bmatrix}, \quad \tilde{\theta} = \begin{bmatrix} 1 & -s \\ -s^* & 1 \end{bmatrix} \frac{1}{c} \\
5. & \tilde{j}_1 = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}, & \tilde{j}_2 = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}, \quad \tilde{\theta} = \begin{bmatrix} c^* & -s \\ s^* & c \end{bmatrix} \\
6. & \tilde{j}_1 = \begin{bmatrix} -1 & \\ & -1 \end{bmatrix}, & \tilde{j}_2 = \begin{bmatrix} -1 & \\ & -1 \end{bmatrix}, \quad \tilde{\theta} = \begin{bmatrix} c^* & -s \\ s^* & c \end{bmatrix}
\end{array}
\tag{11}
$$

These six cases are also sufficient to consider, as every possible signature pair $(\tilde{j}_1, \tilde{j}_2)$ is covered. With $\tilde{j}_1$ and $\tilde{j}_2$ known, we select the appropriate type of rotation matrix, and the rotation parameters s and c follow subsequently from the equation $[a \;\; b]\,\tilde{\theta} = [x \;\; 0]$ as

$$
\begin{array}{lll}
\text{case 1, 4:} & s = b/a\,, & c = (1 - s^*s)^{1/2} \\
\text{case 2, 3:} & s = a/b\,, & c = (1 - s^*s)^{1/2} \\
\text{case 5, 6:} & s = b(a^*a + b^*b)^{-1/2}\,, & c = (1 - s^*s)^{1/2}\,.
\end{array}
$$

Indefinite Schur algorithm

Using the elementary rotations, we will compute Θ such that $[\varepsilon I \;\; H]\Theta = [A' \;\; B']$ in two steps: $\Theta = \tilde{\Theta}\Pi$, where $\tilde{\Theta}$ is a J-unitary matrix with respect to an *unsorted* signature matrix, and Π is a permutation matrix which sorts the signature matrix of $\tilde{\Theta}$. For a given elementary rotation $\tilde{\theta}$, let $\tilde{\Theta}_{(i,k)}$ be the embedding of this rotation into an $(m+n)\times(m+n)$ J-unitary matrix: $\tilde{\Theta}_{(i,k)}$ is equal to the identity matrix, save for four entries, which are together equal to $\tilde{\theta}$:

$$
\begin{bmatrix} \tilde{\Theta}_{(i,k)}(i,i) & \tilde{\Theta}_{(i,k)}(i,m+k) \\ \tilde{\Theta}_{(i,k)}(m+k,i) & \tilde{\Theta}_{(i,k)}(m+k,m+k) \end{bmatrix} = \tilde{\theta}.
$$

$\tilde{\Theta}$ consists of a series of such embedded rotations, such that

$$
\begin{array}{cc} {\scriptstyle +} & {\scriptstyle -} \\ [\,\varepsilon I_m & H\,] \end{array} \tilde{\Theta} = \begin{array}{cc} {\scriptstyle +/-} & {\scriptstyle +/-} \\ [\,X & 0_{m\times n}\,] \end{array}.
$$

As $\tilde{\Theta}$ is applied at the right hand side of $[\varepsilon I \;\; H]$, the rotations act on columns. The entries of H are zeroed, one column at a time, starting with the m-th entry of the first column, continuing with the $m-1$-th entry, etc., till we reach the first entry, after which we zero the m-th entry of the second column:

$$
\tilde{\Theta} = \tilde{\Theta}_{(m,1)}\tilde{\Theta}_{(m-1,1)} \cdots \tilde{\Theta}_{(1,1)} \cdot \tilde{\Theta}_{(m,2)} \cdots \tilde{\Theta}_{(1,2)} \cdot \;\cdots\; \cdot \tilde{\Theta}_{(m,n)} \cdots \tilde{\Theta}_{(1,n)}\,,
$$

where $\tilde{\Theta}_{(i,k)}$ is such that it produces a zero at entry $(i, m+k)$, viz.

$$
\begin{array}{c} {\scriptstyle +\;+\;+\;\;-\;-\;-} \\ \left[\begin{array}{ccc|ccc} \varepsilon & & & \times & \times & \times \\ & \varepsilon & & \times & \times & \times \\ & & \varepsilon & \times & \times & \times \end{array}\right] \end{array}
\overset{\tilde{\Theta}_{(m,1)}}{\rightarrow}
\begin{array}{c} {\scriptstyle +\;+\;-\;\;+\;-\;-} \\ \left[\begin{array}{ccc|ccc} \varepsilon & & \times & \times & \times & \times \\ & \varepsilon & \times & \times & \times & \times \\ & & \times & 0 & \times & \times \end{array}\right] \end{array}
\overset{\tilde{\Theta}_{(m-1,1)}}{\rightarrow}
$$

$$
\begin{array}{c} {\scriptstyle +\;+\;-\;\;+\;-\;-} \\ \left[\begin{array}{ccc|ccc} \varepsilon & \times & \times & \times & \times & \times \\ & \times & \times & 0 & \times & \times \\ & & \times & 0 & \times & \times \end{array}\right] \end{array}
\rightarrow \cdots \overset{\tilde{\Theta}_{(1,n)}}{\rightarrow}
\begin{array}{c} {\scriptstyle -\;+\;-\;\;+\;+\;-} \\ \left[\begin{array}{ccc|ccc} \times & \times & \times & 0 & 0 & 0 \\ & \times & \times & 0 & 0 & 0 \\ & & \times & 0 & 0 & 0 \end{array}\right] \end{array}.
$$

Input: $H : m \times n,\ \varepsilon \geq 0$
Output: $X, \tilde{\Theta}, \tilde{J}_2$ s.t. $[\varepsilon I \;\; H]\tilde{\Theta} = [X \;\; 0]$

$[A \;\; B] := [\varepsilon I_m \;\; H]$

$\tilde{J} := \begin{bmatrix} I_m & \\ & -I_n \end{bmatrix}$, $\tilde{\Theta} := \begin{bmatrix} I_m & \\ & I_n \end{bmatrix}$

for $k = 1$ to n,
  for $i = m$ downto 1,
    $[a \;\; b] := [A(i,i) \;\; B(i,k)]$
    $\tilde{j}_1 := \begin{bmatrix} \tilde{J}(i,i) & 0 \\ 0 & \tilde{J}(m+k, m+k) \end{bmatrix}$
    Compute $\tilde{\theta}, \tilde{j}_2$ from $a, b, \tilde{j}_1$ s.t. $[a \;\; b]\tilde{\theta} = [* \;\; 0]$ (eq. (11))
    Embed $\tilde{\theta}$ into $\tilde{\Theta}_{(i,k)}$
    $[A \;\; B] := [A \;\; B]\tilde{\Theta}_{(i,k)}$
    $\tilde{\Theta} := \tilde{\Theta}\tilde{\Theta}_{(i,k)}$
    $\tilde{J}(i,i) := (\tilde{j}_2)_{1,1}$
    $\tilde{J}(m+k, m+k) := (\tilde{j}_2)_{2,2}$
  end
end

$[X \;\; 0] := [A \;\; B]$
$\tilde{J}_2 := \tilde{J}$

Algorithm 1. Schur recursion to compute the factorization $[\varepsilon I \;\; H]\tilde{\Theta} = [X \;\; 0]$ from H.

(Except for the first matrix, the signatures of the columns in the above matrices are examples, as they are data dependent.) This scheme ensures that $[\varepsilon I \;\; H]\tilde{\Theta} = [X \;\; 0]$, where X is a resulting upper triangular invertible matrix; it contains the columns of A and B in some permuted order.

To compute this ordering, and to compute the rotations $\tilde{\theta}$, we have to keep track of the signature of each column. Suppose that, at the (i,k)-th step, we have to compute $\tilde{\Theta}_{(i,k)}$ such that $[A_1 \;\; B_1]\tilde{\Theta}_{(i,k)} = [A_2 \;\; B_2]$, where $B_2(i,k) = 0$. Associated to every column of $[A_1 \;\; B_1]$ is a signature (+1 or −1). Initially, $[A_1 \;\; B_1] = [\varepsilon I \;\; H]$ and $J = \left[\begin{smallmatrix} I & 0 \\ 0 & -I \end{smallmatrix}\right]$, so that every column of A_1 has a positive signature, and every column of B_1 has a negative signature. At the (i,k)-th step, we act on the i-th column of A_1 and on the k-th column of B_1. In particular, we zero $b = B_1(i,k)$ against $a = A_1(i,i)$, using an elementary rotation $\tilde{\theta}$. The signature $\tilde{j}_1$ to use is equal to the signature of the corresponding columns of A_1 and B_1. At this point, the elementary rotation θ exists if $[a \;\; b]\tilde{j}_1[a \;\; b]^H$ is not equal to zero (if it is, then the recursive scheme breaks down—see the next subsection). After computation of the elementary rotation and applying this rotation to the corresponding two columns, we obtain $[A_2 \;\; B_2]$, and $\tilde{j}_2$ is the new signature of these columns. Having performed $m \times n$ such steps, we have $[A_2 \;\; B_2] = [X \;\; 0]$, and its signature is equal to $\tilde{J}_2$.

The algorithm to compute $\tilde{\Theta}$ is summarized as algorithm 1. The result of the algorithm is $[X \;\; 0] = [\varepsilon I \;\; H]\tilde{\Theta}$, where X is upper triangular and $\tilde{\Theta}$ is J-unitary with respect to $\tilde{J}_1 = J$ and $\tilde{J}_2$. The final step (not listed in the algorithm) is to sort the entries of $\tilde{J}_2$, *i.e.*, to find a permutation Π such that

$$J_2 = \Pi^H \tilde{J}_2 \Pi$$

is a sorted signature matrix. Then $J_2 = J$, and $\Theta = \tilde{\Theta}\Pi$ is J-unitary with respect to J. In addition, putting $[A' \;\; B'] = [X \;\; 0]\Pi$ gives the required result, $[\varepsilon I \;\; H]\Theta = [A' \;\; B']$. The nonzero columns of A' are the columns of X with a positive signature, the nonzero columns of B' correspond to the columns of X with a negative signature.

Breakdown

In the previous subsection, we had to assume that the data matrix H was such that at no point in the algorithm $[a \;\; b]\ \tilde{j}_1[a \;\; b]^H$ is equal to zero. If the expression is zero, then there is no J-unitary rotation $\tilde{\theta}$ such that $[a \;\; b]\tilde{\theta} = [* \;\; 0]$. Note that the condition in theorem 1 that none of the singular values of H are equal to 1 does not preclude this case, but merely assures that there *exists*

a $\tilde{\Theta}$ which will zero H. One simple example is obtained by taking $H = [1 \;\; 1]^T$. It is straightforward to show that there is no J-unitary $\tilde{\Theta}$ such that

$$\left[\begin{array}{cc|c} 1 & & 1 \\ & 1 & 1 \end{array}\right] \tilde{\Theta} = \left[\begin{array}{cc|c} * & * & 0 \\ 0 & * & 0 \end{array}\right]$$

as the J-norms of the last row will not be equal. Hence Θ cannot be obtained by the recursive algorithm. However, a more general $\tilde{\Theta}$ does exist, such that

$$\begin{array}{cccccccc} + & + & & - & & & + & - & & + \end{array}$$
$$\left[\begin{array}{cc|c} 1 & & 1 \\ & 1 & 1 \end{array}\right] \tilde{\Theta} = \frac{1}{\sqrt{2}} \left[\begin{array}{cc|c} 1 & 1 & 0 \\ -1 & 1 & 0 \end{array}\right]$$

viz.

$$\tilde{\Theta} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 & \sqrt{2} \\ -1 & -1 & \sqrt{2} \\ 0 & 2 & -\sqrt{2} \end{bmatrix}, \qquad \tilde{J}_1 = \begin{bmatrix} 1 & & \\ & 1 & \\ & & -1 \end{bmatrix}, \qquad \tilde{J}_2 = \begin{bmatrix} 1 & & \\ & -1 & \\ & & 1 \end{bmatrix}.$$

The difference is that, in this factorization the resulting matrix X is no longer upper triangular. Hence, it is not clear how $\tilde{\Theta}$ can be computed recursively in a similar scheme as we had before. Necessary and sufficient conditions on the singular values of H and a collection of submatrices of H so that the Schur algorithm does not break down are given by the following theorem.

Theorem 3. **([17])** *Let $H : m \times n$ be a given matrix. Denote by $H_{[i,k]}$ the submatrix, consisting of the i-th row to the m-th row and the first k columns of H. Then the Schur algorithm (alg. 1) does not break down if and only if none of the singular values of $H_{[i,k]}$ is equal to ε, for all i and k.*

Similar conditions on the matrices to prevent breakdown have been obtained in other Schur methods as well. It should be noted that the conditions for preventing breakdown are quite elaborate, as only one condition (none of the singular values of H are equal to ε) suffices for the *existence* of Θ. The treatment of the more general case is possible, see *e.g.,* [11], but requires global matrix operations which are not attractive from the parallel implementation point of view. It is possible to consider hyperbolic Householder transformations, as in [15, 16], which lifts some of the regularity conditions while retaining much of the computational structure.

Signal flow diagrams

A signal flow diagram of the algorithm is depicted in figure 1. In this figure, the columns of H are entered at the left hand side (with negative signatures), the columns of εI_m are entered at the top (with positive signatures), and the square blocks denote processors that perform the elementary rotations $[a \;\; b]\tilde{\theta} = [x_1 \;\; x_2]$. The signatures at the inputs of the processors are $\tilde{j}_1$, and at the outputs are $\tilde{j}_2$. A processor in the i-th column of the array skips the first $i-1$ data pairs (which just contains zeros), then computes the appropriate $\tilde{\theta}$ and $\tilde{j}_2$ from the input data pair and the input signature, and subsequently applies $\tilde{\theta}$ to the remaining $m-i$ data pairs. The output of the processor array is $[X \;\; 0]$, where X emerges at the bottom of the array, and the zeros are produced at the right hand side. The corresponding output signature matrix $\tilde{J}_2$ is given by the signs that are produced at the bottom and right side of the array. If Θ is also required, then one has to put $[I_m \;\; 0]$ and $[0 \;\; I_n]$ also into the array, and to apply the already computed elementary operations to these columns as well.

The signal flow diagram can be converted to a (possibly more familiar) processor array form, depicted in figure 2. In this figure, the triangular processor array has dimensions $m \times m$, and is initialized by εI_m. The columns of H are entered one by one at the right hand side of the array, and are made zero step by step. The appropriate elementary rotation $\tilde{\theta}$ is computed by the processors on the diagonal of the triangular part, and is communicated systolically to the processors above the diagonal. After n steps, the triangular array contains the resulting matrix X. If Θ_{11} and Θ_{12} are also needed, then the triangular part may be extended by a square part in which the rotations are accumulated. The square part is also initialized by I_m, and at the right hand side, zeros are entered. Note that actually the unsorted $\tilde{\Theta}_{11}$ and $\tilde{\Theta}_{12}$ are computed.

5. UPDATING AND DOWNDATING

In signal processing applications, the columns of the matrix H are usually not available all at the same time, but become known one by one as time progresses. It is in such cases often desirable to process the available data to obtain the approximant, and to

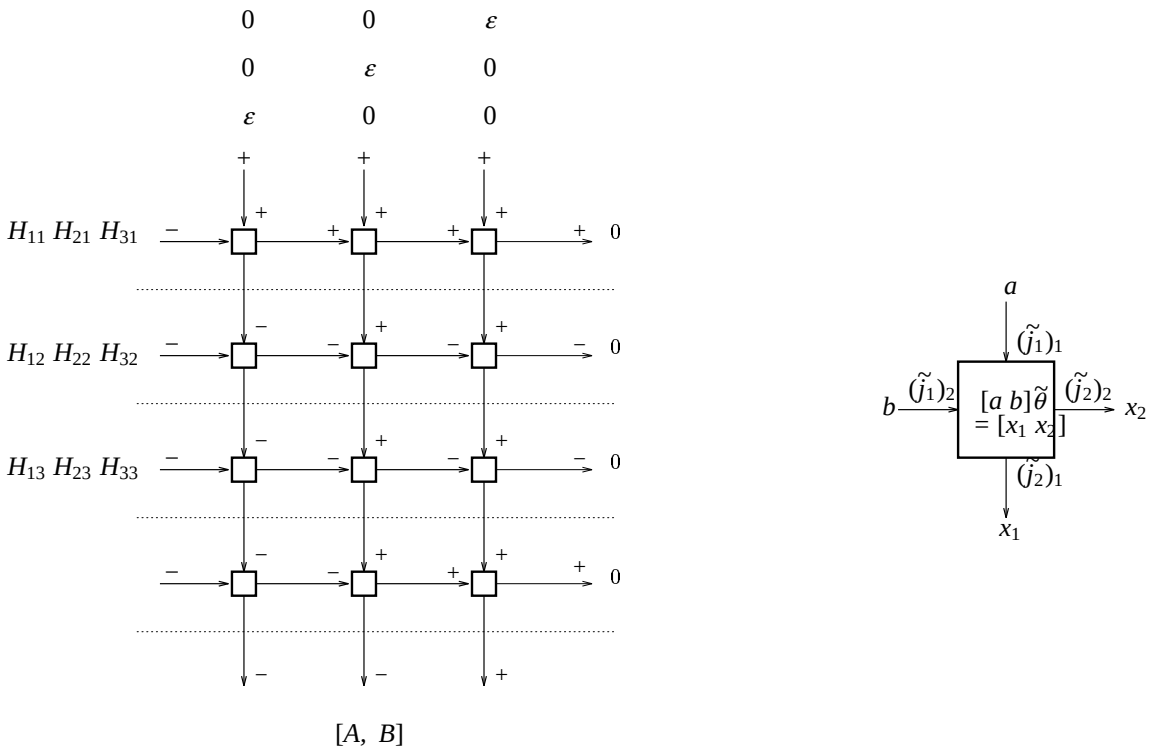

Figure 1. Signal flow diagram of the Schur algorithm. Processors compute $\tilde{\theta}$ from incoming data pairs $[a \;\; b]$ and $\tilde{j}_1$, which is stored internally and applied to subsequent data pairs.

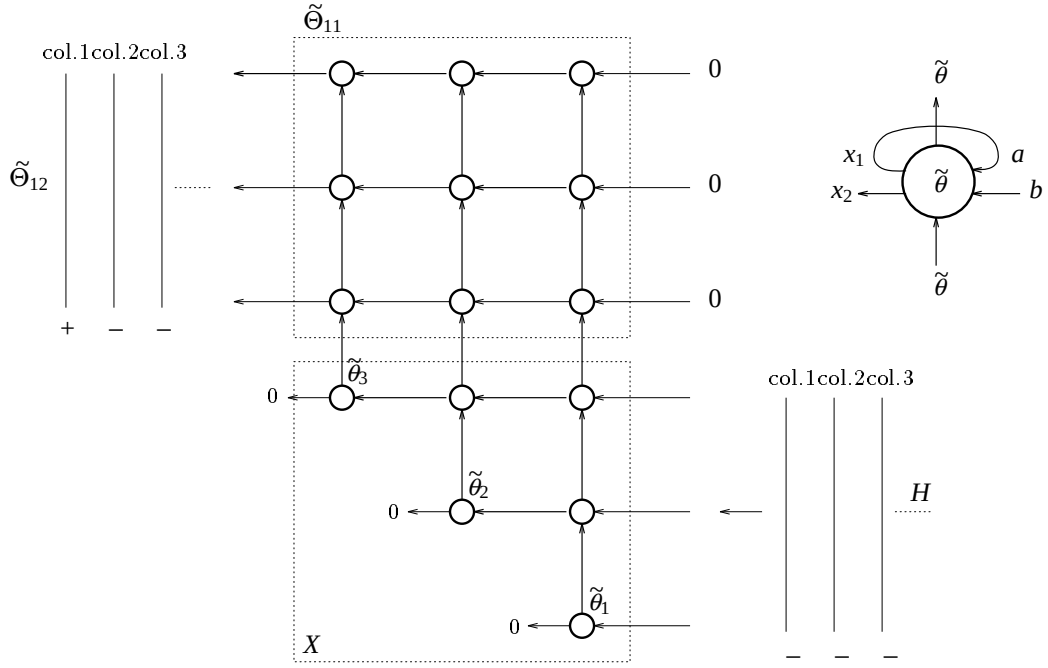

Figure 2. Triangular processor array for computing X, $\tilde{\Theta}_{11}$ and $\tilde{\Theta}_{12}$. Processors store x_1, and communicate $\tilde{\theta}$ upwards.

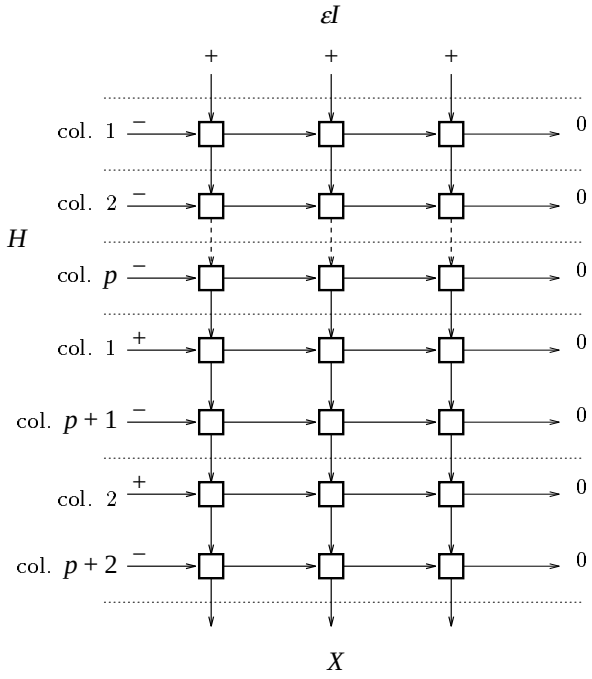

Figure 3. Updating and downdating the Schur algorithm.

update this approximant as new information comes in. A partial solution is to update only the factorization step. Doing so yields the column space $\mathcal{R}(B)$ of the approximant, as the columns of X with negative signature, which in many applications provides sufficient information. Downdating (discarding old columns) is also possible, and is shown to be equivalent to updating with a positive signature. Hence, adaptive approximations, with sliding windows on the data matrix, are straightforward to implement. Updating and downdating of the approximant itself is much harder because the size of Θ is growing, and we omit a discussion of this.

Updating

The generic updating problem that we will consider at this point is, given a matrix H and a vector h, to find the required factorization of the enlarged matrix $H_1 = [H \;\; h]$ from that of H. The factorization obtained by the Schur algorithm is straightforward to update, as in this algorithm the columns of H are processed one by one, starting with the first column. Hence, if $[\varepsilon I \;\; H]\tilde{\Theta} = [X \;\; 0]$ then the computation of $\tilde{\Theta}_1$ such that $[\varepsilon I \;\; H_1]\tilde{\Theta}_1 = [X_1 \;\; 0]$ can be split into two steps:

$$[\varepsilon I \;\; H_1]\tilde{\Theta}_1 = [X \;\; 0 \;\; h]\tilde{\Theta}' = [X_1 \;\; 0], \qquad \tilde{\Theta}_1 = \begin{bmatrix} \tilde{\Theta} & \\ & 1 \end{bmatrix} \tilde{\Theta}'.$$

Hence, a new column h can always be introduced after the preceding set of columns have been processed. Each new column gives m extra conditions on the singular values in order to prevent a breakdown of the Schur algorithm. In the algorithm (alg. 1), introducing a new column h means that the loop over i has to be performed for $k = n + 1$, and in the signal flow diagram (figure 1), an extra row containing m processors has to be added at the bottom of the array. Similarly, in the triangular processor array (figure 2), the new column h is used as input of the processor array at the right hand side.

Downdating

Let H be a given matrix, which decomposes into two blocks: $H = [H_1 \;\; H_2]$. The downdating problem is to find a factorization for $[\varepsilon I \;\; H_2]$, using the factorization for $[\varepsilon I \;\; H]$ that has already been computed. So suppose that $[\varepsilon I \;\; H]\tilde{\Theta} = [X \;\; 0]$ and it is required to compute a factorization $[\varepsilon I \;\; H_2]\tilde{\Theta}_2 = [X_2 \;\; 0]$ using the first factorization. We will use the following uniqueness fact, which is similar to the uniqueness of the QR factorization (viz. [18]).

Lemma 4. **([17])** *Let A_1 and A_2 be two matrices, with associated signature matrices $\widetilde{J}_1$ and $\widetilde{J}_2$, respectively, and suppose that $A_1\widetilde{J}_1A_1^H = A_2\widetilde{J}_2A_2^H$. If $\widetilde{\Theta}_1$, $\widetilde{\Theta}_2$ are J-unitary matrices (with respect to these input signatures) such that*

$$\begin{aligned} A_1\widetilde{\Theta}_1 &= [X_1 \quad 0], \qquad X_1 \text{ upper, } \operatorname{diag}(X_1) > 0 \\ A_2\widetilde{\Theta}_2 &= [X_2 \quad 0], \qquad X_2 \text{ upper, } \operatorname{diag}(X_2) > 0 \end{aligned}$$

then $X_1 = X_2$.

Corollary 5. *Let $H = [H_1 \;\; H_2]$, and let $\widetilde{\Theta}$ be J-unitary with input signature $\widetilde{J}$, such that*

$$\begin{bmatrix} \overset{+}{\varepsilon I} & \overset{-}{H} & \overset{+}{H_1} \end{bmatrix} \widetilde{\Theta} = [X \quad 0 \quad 0], \qquad \widetilde{J} = \begin{bmatrix} I & & \\ & -I & \\ & & I \end{bmatrix}. \tag{12}$$

where X is a square upper triangular matrix with $\operatorname{diag}(X) > 0$. Then there exists a J-unitary matrix $\widetilde{\Theta}_2$ such that

$$\begin{bmatrix} \overset{+}{\varepsilon I} & \overset{-}{H_2} \end{bmatrix} \widetilde{\Theta}_2 = [X \quad 0]. \tag{13}$$

The implication of this lemma is that the downdating problem can be converted into an updating problem. In particular, in order to compute the factorization (13), we can compute the factorization (12), which is obtained by updating a factorization for $[\varepsilon I \;\; H]$ with those columns of H that are to be removed (*i.e.*, H_1), now giving these columns a positive signature.

In adaptive signal processing applications, H is a data matrix whose columns become available one at a time, and it is desirable to compute a factorization of only the last p columns $h_{N-p+1}, \cdots, h_N$ of H that have been collected at time point N. The factorization is computed from a similar factorization at the previous time point $N-1$, by updating with the new column h_N and downdating with the column h_{N-p} which is to be discarded at this point. Figure 3 shows the signal flow diagram that corresponds to this sliding window factorization scheme.

6. APPLICATIONS

Approximate TLS

As a first application of the Schur approximation scheme, we recall the approximate (low rank) total least squares problem that we described in the introduction. The problem is, given A, b, ε where $A : n \times m$, $(n > m)$, find $\hat{A}$, $\hat{b}$ such that

$$\begin{cases} \| [A \;\; b] - [\hat{A} \;\; \hat{b}] \| \le \varepsilon, \\ \operatorname{rank}[\hat{A} \;\; \hat{b}] \text{ is minimal,} \\ \hat{A}x = \hat{b}, \text{ with } \| x \| \text{ minimal.} \end{cases}$$

A solution x of these equations will be such that $\| Ax - b \| \le \varepsilon \sqrt{\| x \| + 1}$. The solution is obtained by computing a minimum-rank approximant for $H = [A \;\; b]^H$, as in theorem 1 or 2. Taking $S_L = 0$ gives

$$[\hat{A} \quad \hat{b}] = \Theta_{22}^{-H} \begin{bmatrix} B^H \\ 0 \end{bmatrix}.$$

Since $[\hat{A} \quad \hat{b}]$ is of rank d, its kernel is non-empty. In fact,

$$[\hat{A} \quad \hat{b}] \begin{bmatrix} x \\ -1 \end{bmatrix} = 0 \quad \Leftrightarrow \quad B^H \begin{bmatrix} x \\ -1 \end{bmatrix} = 0,$$

so that only B, not Θ, has to be computed. Writing $B^H = [B_1 \;\; b_2]$, the minimal-norm solution x to $\hat{A}x = \hat{b}$ is obtained as $x = B_1^{\dagger} b_2$.

Table 1. Estimated DOAs for the ESPRIT algorithm.

DOA	(*a*) $\phi_{1,2} = 10^\circ, 70^\circ$			(*b*) $\phi_{1,2} = 20^\circ, 30^\circ$			(*c*) $\phi_{1,2} = 20^\circ, 23^\circ$		
	SVD	Schur 1	Schur 2	SVD	Schur 1	Schur 2	SVD	Schur 1	Schur 2
mean	9.9948	9.9947	10.0138	19.9447	19.9417	21.0607	19.6720	19.4254	9.1648
	70.0160	70.0160	69.9601	30.0209	30.0228	28.7274	23.3412	23.5344	21.5303
std	0.0122	0.0122	0.0124	0.2230	0.2320	1.5937	2.6122	4.0261	209.8293
	0.1243	0.1242	0.1283	0.2384	0.2528	2.3564	2.3578	4.4285	2.2607
$\overline{\alpha}$	0.0249	0.0182	0.0212	0.0257	0.0326	0.1317	0.1508	0.1412	0.5696

Direction finding

In order to assess the applicability of the Schur-based subspace estimation method, we consider the direction finding problem. Suppose that we have an array of m equispaced omnidirectional sensors, which receives d sinusoidal signals from directions ϕ_k, $k = 1, \cdots, d$. A total number of n samples is taken, which gives an $m \times n$ data matrix X modeled as $X = \mathcal{A}S + N$. Here, $\mathcal{A} = [\mathbf{a}(\phi_1), \cdots, \mathbf{a}(\phi_d)] : m \times d$ is the array response matrix, and $S : d \times n$ contains the n samples of the d source signals. N contains samples of white additive i.i.d. noise sources with variance $\sigma^2 I$, independent of the signals. Given X and the function $\mathbf{a}(\phi)$, the ϕ_k are to be estimated.

The ESPRIT algorithm for estimating the DOAs [19] works in two steps. The first step is to estimate the signal subspace, which is usually taken to be the d principal left singular vectors of X. This leads to the classical SVD-ESPRIT direction finding scheme. We will compare this with the Schur-based subspace estimates, and investigate the choices $\mathcal{R}(B - A(S_L)_{11})$ with $S_L = \Theta_{11}^{-1}\Theta_{12}\left[\begin{smallmatrix} I & 0 \\ 0 & 0 \end{smallmatrix}\right]$ ("Schur-1"), and $\mathcal{R}(B)$ ("Schur-2"). As is well-known, once the signal subspaces are estimated, the DOAs are obtained via a certain eigenvalue decomposition based on these subspaces.

In the computer simulation experiments, a linear array consisting of $m = 4$ sensors is used. Two sources are impinging on the array. The signal to noise ratio is chosen to be 20dB in all cases. One hundred test runs using $n = 30$ samples are executed. Table 1 lists the statistical results, for three different sets of angles of incidence, and averaged over the test runs. In the last row of the table, $\overline{\alpha}$ is the mean angle between the exact and the estimated subspace [18]. As is seen from the table, the difference between the three subspace estimates is negligible if the signals are spatially well separated. If the signals are coming from closer directions, the variance of the Schur estimate with $S_L = 0$ starts to increase, but the choice $S_L = \Theta_{11}^{-1}\Theta_{12}\left[\begin{smallmatrix} I & 0 \\ 0 & 0 \end{smallmatrix}\right]$ still performs the same as the SVD-based estimate. Finally, part (*c*) of the simulations shows that if the signals are so close that the variance clouds are overlapping even for the SVD-based estimates, then the choice $S_L = 0$ breaks down, but the variance of Schur method 1 is still within reasonable bounds.

ACKNOWLEDGEMENTS

This research was supported in part by the commission of the EC under the ESPRIT BRA program 6632 (NANA2), and by ARPA contract no. F49620-91-C-0086, monitored by the AFOSR. A large part of this research was performed while the author was with Delft University of Technology, Dept. EE. The simulations in section 6 were carried out by Jürgen Götze, TU Munich.

REFERENCES

[1] A.J. van der Veen, E.F. Deprettere, and A.L. Swindlehurst, "Subspace based signal analysis using singular value decomposition," *Proceedings of the IEEE*, vol. 81, pp. 1277–1308, Sept. 1993.

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

[3] L.V. Foster, "Rank and null space calculations using matrix decomposition without column interchanges," *Lin. Alg. Appl.*, vol. 74, pp. 47–71, 1986.

[4] T.F. Chan, "Rank revealing QR factorizations," *Lin. Alg. Appl.*, vol. 88/89, pp. 67–82, 1987.

[5] C.H. Bischof and G.M. Shroff, "On updating signal subspaces," *IEEE Trans. Signal Processing*, vol. 40, pp. 96–105, Jan. 1992.

[6] M. Moonen, P. Van Dooren, and J. Vandewalle, "An SVD updating algorithm for subspace tracking," *SIAM J. Matrix Anal. Appl.*, vol. 13, no. 4, 1992.

[7] M. Moonen, P. Van Dooren, and F. Vanpoucke, "On the QR algorithm and updating the SVD and URV decomposition in parallel," *Lin. Alg. Appl.*, vol. 188,189, pp. 549–568, 1993.

[8] I. Schur, "Uber Potenzreihen, die im Innern des Einheitskreises beschränkt sind, I," *J. Reine Angew. Math.*, vol. 147, pp. 205–232, 1917. Eng. Transl. *Operator Theory: Adv. Appl.*, vol. 18, pp. 31-59, Birkhäuser Verlag, 1986.

[9] J. Chun, T. Kailath, and H. Lev-Ari, "Fast parallel algorithms for QR and triangular factorizations," *SIAM J. Sci. Stat. Comp.*, vol. 8, no. 6, pp. 899–913, 1987.

[10] P. Dewilde and E. Deprettere, "The generalized Schur algorithm: Approximation and hierarchy," in *Operator Theory: Advances and Applications*, vol. 29, pp. 97–116, Birkhäuser Verlag, 1988.

[11] K. Diepold and R. Pauli, "Schur parametrization of symmetric indefinite matrices based on a network theoretic model," *Archiv für Elektronik und Übertragungstechnik AEÜ*, vol. 44, pp. 92–96, 1990.

[12] A.J. van der Veen and P.M. Dewilde, "On low-complexity approximation of matrices," *Linear Algebra and its Applications*, vol. 203/204/205, May/June/July 1994.

[13] S.T. Alexander, C.-T. Pan, and R.J. Plemmons, "Analysis of a recursive least squares hyperbolic rotation algorithm for signal processing," *Lin. Alg. Appl.*, vol. 98, pp. 3–40, 1988.

[14] C.H. Bischof, C.-T. Pan, and P.T.P. Tang, "A Cholesky up- and downdating algorithm for systolic and SIMD architectures," *SIAM J. Sci. Stat. Comput.*, vol. 14, pp. 670–676, May 1993.

[15] G. Cybenko and M. Berry, "Hyperbolic Householder algorithms for factoring structured matrices," *SIAM J. Matrix Anal. Appl.*, vol. 11, pp. 499–520, Oct. 1990.

[16] A. Bojanczyk and A.O. Steinhardt, "The hyperbolic transformations in signal processing and control," *preprint*, Feb. 1994.

[17] A.J. van der Veen, "A Schur method for low-rank matrix approximation," *submitted for publication*, Dec. 1993.

[18] G. Golub and C.F. Van Loan, *Matrix Computations*. The Johns Hopkins University Press, 1984.

[19] R. Roy, A. Paulraj, and T. Kailath, "ESPRIT—a subspace rotation approach to estimation of parameters of cisoids in noise," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 34, no. 5, pp. 1340–1342, 1986.