APPROXIMATE INVERSION OF A LARGE SEMISEPARABLE POSITIVE MATRIX

ALLE-JAN VAN DER VEEN*

Abstract. The inversion of a large $(n \times n)$ positive matrix is considered. We assume that the matrix has a semi-separable structure, which implies that all submatrices away from the main diagonal have rank less than q (the matrix itself may be full). In practice, a specified matrix will not exactly have low-rank submatrices. Given a threshold and a positive matrix T, the submatrices of T are rank truncated to this threshold (balanced model reduction) and the inverse of a Cholesky factor of T is computed using time-varying state-space techniques. The proposed algorithm requires $O(n^2q)$ operations, where q is the average rank of the submatrices as detected by the algorithm.

1. Introduction.

We consider the problem of the inversion of a large $(n \times n)$ positive matrix T. Direct techniques require $O(n^3)$ operations (multiplications and additions). We assume that the matrix has a semi-separable structure, which implies that all submatrices away from the main diagonal have rank less than q. An example of such a matrix is a banded matrix with band width q, but also the inverse of a band matrix is semiseparable of order q. Using this structure, we show that $O(n^2q)$ operations are required.

Semiseparable matrices were perhaps first considered in [13]. In [7], matrices with a related structure were studied via an explicit link to time-varying system theory; this produced QR factorization algorithms (called "inner-outer factorization" and "external factorization"), matrix inversion techniques (see also [14]), and low-rank approximation techniques (see also [6]). More recently, inversion algorithms based on related algebraic formalisms were proposed [1–5, 8–12].

In general, matrices with a semiseparable structure allow for inversion with a complexity *linear* in the matrix dimension n. However, this holds for the case where the matrix is already compactly represented, e.g., by a time-varying state space description. In this paper, we consider the more common case where an "unstructured" matrix is specified via its entries, typically as a full matrix. The compact representation has to be derived along with the inversion steps, therefore the complexity cannot be smaller than $O(n^2)$ unless the matrix is sparse.

Moreover, we consider the typical case where the matrix does not exactly have low-rank submatrices: they will be rank truncated by the algorithm. To this end, a threshold γ is specified. More specifically, we take the following approach:

1. Cholesky factorization: $T = F^*F$, where F is upper triangular,
2. Truncate the rank of the submatrices of F according to a threshold γ,
3. Invert the resulting $\tilde{F}$.

The algorithm is derived using a time-varying state-space approach [7].

2. Basic algorithm derivation.

In this section, we summarize some of the results in [7], which will lead to an inversion algorithm of complexity $O(n^3)$. This algorithm forms the basis for subsequent improvements.

2.1. Matrix multiplication.

As in [7], we describe the multiplication of a (row) vector $\mathbf{u} = [u_1, \cdots, u_n]$ by an upper triangular matrix T, i.e., $\mathbf{y} = \mathbf{u}T$, by means of a

*Delft University of Technology, Department of Electrical Engineering, Mekelweg 4, 2628 CD Delft, The Netherlands, email: allejan@cas.et.tudelft.nl. This paper was presented at MTNS'04 (Brussels, 6 July 2004).

time-varying state space description (or recursion)

$$(1) \qquad \begin{cases} x_{k+1} &= x_k A_k + u_k B_k \\ y_k &= x_k C_k + u_k D_k\,, \qquad k = 1, \cdots, n \end{cases}$$

where $\mathbf{y} = [y_1, \cdots, y_n]$. This corresponds to multiplication by an upper-triangular matrix

$$(2) \qquad T = \begin{bmatrix} D_1 & B_1C_2 & B_1A_2C_3 & B_1A_2A_1C_4 & \cdots \\ & D_2 & B_2C_3 & B_2A_3C_4 & \\ & & D_3 & B_3C_4 & \\ & \mathbf{0} & & D_4 & \cdots \\ & & & & \ddots \end{bmatrix}.$$

If the size of the state vector x_k is q_k and the average of q_k is q, then matrix-vector multiplication using the state recursion has a complexity of $O(nq^2)$, instead of $O(n^2)$ for a direct implementation. The complexity can be further reduced to $O(nq)$ by using certain implementations of the recursion (e.g., factored state recursions).

For a given upper-triangular matrix T, define its Hankel submatrices H_k as the collection of submatrices of T away from the main diagonal:

$$(3) \qquad H_k = \begin{bmatrix} T_{k-1,k} & T_{k-1,k+1} & T_{k-1,k+2} & \cdots \\ T_{k-2,k} & T_{k-2,k+1} & & \\ T_{k-3,k} & & \ddots & \\ \vdots & & & \end{bmatrix}, \qquad k = 1, \cdots, n\,.$$

Substituting (2), it is seen that the state recursion corresponds to a low rank factorization of the H_k:

$$(4) \quad H_k = \begin{bmatrix} B_{k-1} \\ B_{k-2}A_{k-1} \\ B_{k-3}A_{k-2}A_{k-1} \\ \vdots \end{bmatrix} [C_k \quad A_kC_{k+1} \quad A_kA_{k+1}C_{k+2} \quad \cdots] =: \mathcal{C}_k\mathcal{O}_k\,,$$

Numerically, the factors $\mathcal{C}_k$ and $\mathcal{O}_k$ represent q_k-dimensional bases of the column span and row span of H_k, respectively. It can be shown that, for any upper-triangular matrix T, minimal realizations with $q_k = \text{rank}(H_k)$ states always exist. Efficient computations are possible if $\text{rank}(H_k)$ is small for all k.

In system theory, $\mathcal{C}_k$ is known as the controllability operator, and $\Lambda_{c,k} := \mathcal{C}_k^*\mathcal{C}_k$ as the controllability gramian at time k. It satisfies the recursion

$$(5) \qquad \Lambda_{c,k+1} = A_k^*\Lambda_{c,k}A_k + B_k^*B_k\,.$$

Similarly $\Lambda_{o,k} = \mathcal{O}_k\mathcal{O}_k^*$ is the observability gramian at time k, with recursion

$$(6) \qquad \Lambda_{o,k+1} = A_k\Lambda_{o,k}A_k^* + C_kC_k^*\,.$$

A realization is said to be in 'output normal form' if $\Lambda_{o,k} = I$ for all k. A realization is minimal if both $\Lambda_{c,k}$ and $\Lambda_{o,k}$ are full rank for all k (in that case, the factorization $H_k = \mathcal{C}_k\mathcal{O}_k$ has full-rank factors for all k).

2.2. Cholesky factorization.

THEOREM 1 (Spectral Factorization Theorem, p.369 of [7]). *Given $T > 0$, where the upper triangular part of T has state-space realization $\{A_k, B_k, C_k, D_k\}$, let $T = F^*F$ where F is upper triangular. Then a realization $\{A_{F,k}, B_{F,k}, C_{F,k}, D_{F,k}\}$ of F is given by*

$$\left\{ \begin{array}{lcl} A_{F,k} & = & A_k \\ C_{F,k} & = & C_k \\ D_{F,k} & = & (D_k - C_k^* \Lambda_k C_k)^{-1/2} \\ B_{F,k} & = & D_{F,k}^{-1}(B_k - C_k^* \Lambda_k A_k) \end{array} \right. \tag{7}$$

where Λ_k is given by the iteration

$$\Lambda_{k+1} = A_k^* \Lambda_k A_k + (B_k^* - A_k^* \Lambda_k C_k)(D_k - C_k^* \Lambda_k C_k)^{-1}(B_k - C_k^* \Lambda_k A_k) . \tag{8}$$

The expression for Λ_k is recognized as a discrete-time Riccati equation. The resulting F is 'outer', meaning that it is upper triangular and that it has a 'stable' upper triangular inverse (stable in a system-theoretical sense [7], which practically means that the entries of F^{-1} do not exponentially grow away from the main diagonal).

Λ_k is interpreted as the controllability gramian of F, because (cf. (5))

$$\Lambda_{k+1} = A_k^* \Lambda_k A_k + B_{F,k}^* B_{F,k} . \tag{9}$$

2.3. Inverse of an outer matrix.

THEOREM 2. *The inverse of F with realization $\{A_k, B_k, C_k, D_k\}$ is $S = F^{-1}$ with realization*

$$\begin{bmatrix} A_{S,k} & C_{S,k} \\ B_{S,k} & D_{S,k} \end{bmatrix} = \begin{bmatrix} A_k - C_k D_k^{-1} B_k & -C_k D_k^{-1} \\ D_k^{-1} B_k & D_k^{-1} \end{bmatrix} .$$

Proof. This is because

$$\left\{ \begin{array}{lcl} x_{k+1} & = & x_k A_k + u_k B_k \\ y_k & = & x_k C_k + u_k D_k \end{array} \right. \quad \Leftrightarrow \quad \left\{ \begin{array}{lcll} x_{k+1} & = & x_k(A_k - C_k D_k^{-1} B_k) & + y_k D_k^{-1} B_k \\ u_k & = & -x_k C_k D_k^{-1} & + y_k D_k^{-1} \end{array} \right.$$

□

2.4. Trivial realization. Consider an upper triangular matrix T. A realization of T is given by

$$\begin{bmatrix} A_k & C_k \\ B_k & D_k \end{bmatrix} = \left[\begin{array}{cccc|c} 1 & & & & 0 \\ & \ddots & & & \vdots \\ & & & 1 & 0 \\ 0 & & \cdots & 0 & 1 \\ \hline T_{k,n} & & \cdots & T_{k,k+1} & T_{kk} \end{array} \right] .$$

This trivial realization is in output normal form ($A_k A_k^* + C_k C_k^* = I$), but not minimal: $\Lambda_{o,k} = I$ but $\Lambda_{c,k}$ is singular. It is used as a starting point to be able to apply the preceding theorems, which were all formulated in terms of state space realizations.

2.5. Algorithm 0 (complexity $O(n^3)$). Consider a matrix $T > 0$ of size $n \times n$. By combining the preceding sections, we can find a trivial realization of T, factor $T = F^*F$ in state space, and compute a state space representation of $S = F^{-1}$. This gives the following algorithm:

```
1.  $\Lambda_1 = 0_{n\times n}$
2.  for $k = 1$ to $n$ do
3.      $[A_k \quad C_k] = I_{n-k+1}$
4.      $B_k = [T_{k,n} \cdots T_{k,k+1}]$
5.      $D_k = T_{kk}$
6.      $D_{F,k} = (D_k - C_k^* \Lambda_k C_k)^{-1/2}$
7.      $B_{F,k} = D_{F,k}^{-1}(B_k - C_k^* \Lambda_k A_k)$
8.      $\Lambda_{k+1} = A_k^* \Lambda_k A_k + B_{F,k}^* B_{F,k}$
9.      $\begin{bmatrix} A_{F,k} & C_{F,k} \\ B_{F,k} & D_{F,k} \end{bmatrix} = \begin{bmatrix} A_k & C_k \\ B_{F,k} & D_{F,k} \end{bmatrix}$
10.     $\begin{bmatrix} A_{S,k} & C_{S,k} \\ B_{S,k} & D_{S,k} \end{bmatrix} = \begin{bmatrix} A_k - C_k D_{F,k}^{-1} B_{F,k} & -C_k D_{F,k}^{-1} \\ D_{F,k}^{-1} B_{F,k} & D_{F,k}^{-1} \end{bmatrix}$
11. end
```

The complexity of this algorithm is determined by the size of B_k $(n-k \times 1)$ and Λ_k $(n-k \times n-k)$. Multiplication of Λ_k by A_k or C_k in steps 6 and 7 amounts to the selection of certain submatrices, and to not count for the complexity. The total complexity (due mostly to the rank-1 updates in step 8 and 10) is $O(n^3)$. Since each $A_{S,k}$ has size $n-k \times n-k$, storage is large, $O(n^3)$. It is actually more efficient to store only $B_{F,k}, D_{F,k}$, which are simply the entries of the Cholesky factor and which requires $O(n^2)$ storage entries. Application of the inverse factor to a vector can then be computed from step 10 without computing the matrices explicitly (this corresponds to backsubstitution in triangular matrix inversion).

In subsequent sections, we will make refinements to this algorithm to reduce its complexity. A first observation is that Λ_k is usually rank deficient, because it starts as an all-zero matrix (step 1) and step 8 gives only rank-1 updates. Therefore, we would like to work with a minimal factor M_k of $\Lambda_k = M_k^* M_k$, where the rows of M_k are linearly independent. Suppose that M_k has size $q_k \times (n-k)$, then we will first derive an algorithm that has complexity $O(n^2 q^2)$, where q is the 'average' size of q_k over k; and in section 4 this is reduced further to $O(n^2 q)$. We also will implement a rank approximation of Λ_k, i.e., truncate all eigenvalues smaller than a threshold. This will reduce the size q_k of the approximated $\widetilde{M}_k$.

Along with the rank reduction, we will use the observation that $[A_k \quad C_k] = I$ is trivial, but unnecessarily large because the realization we start with does not have a minimal number of states. We will derive a realization where $A_{F,k}$ has size $q_k \times q_{k+1}$.

3. Approximate inversion.

3.1. State transformation and minimal realization. Let R_k be a sequence of invertible matrices, and compute

$$\begin{bmatrix} A'_k & C'_k \\ B'_k & D'_k \end{bmatrix} = \begin{bmatrix} R_k^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} A_k & C_k \\ B_k & D_k \end{bmatrix} \begin{bmatrix} R_{k+1} & \\ & I \end{bmatrix}.$$

Then $\{A'_k, B'_k, C'_k, D'_k\}$ has the same input-output relationship as $\{A_k, B_k, C_k, D_k\}$ (it is an equivalent realization of the same upper triangular matrix but with different internal states), and

$$\Lambda'_{c,k} = R_k^* \Lambda_{c,k} R_k \,, \qquad \Lambda'_{o,k} = R_k^{-*} \Lambda_{o,k} R_k^{-1} \,.$$

In our application we start with a trivial realization for which $\Lambda_{o,k} = I$, but $\Lambda_{c,k}$ is often rank deficient: the realization does not have a minimal number of states. The following theorem will be instrumental to reduce the number of states:

THEOREM 3. *Suppose an upper triangular matrix T has a realization $\{A_k, B_k, C_k, D_k\}$ with controllability gramian*

$$\Lambda_{c,k} = \begin{bmatrix} \tilde{\Lambda}_k & \\ & 0 \end{bmatrix} .$$

Then the realization matrix (partitioned conformably) has the following structure:

$$\begin{bmatrix} A_k & C_k \\ B_k & D_k \end{bmatrix} = \left[\begin{array}{cc|c} A_{11,k} & 0 & C_{1,k} \\ A_{21,k} & A_{22,k} & C_{2,k} \\ \hline B_{1,k} & 0 & D_k \end{array}\right] .$$

Furthermore, $\{A_{11,k}, B_{1,k}, C_{1,k}, D_k\}$ is a realization of T, with controllability gramian $\tilde{\Lambda}_k$.

To apply this theorem in more general cases, consider a realization with a singular controllability gramian $\Lambda_{c,k}$ of rank $\tilde{q}_k$, and compute its eigenvalue decomposition

$$\Lambda_{c,k} =: V_k \begin{bmatrix} \tilde{\Lambda}_k & \\ & 0 \end{bmatrix} V_k^* = \tilde{V}_k \tilde{\Lambda}_k \tilde{V}_k^* \,, \tag{10}$$

where $\tilde{\Lambda}_k$ is a diagonal matrix containing the nonzero eigenvalues of $\Lambda_{c,k}$, V_k is unitary, and $\tilde{V}_k$ contains the columns of V_k corresponding to the entries in $\tilde{\Lambda}_k$. Applying V_k as a state transformation gives

$$\begin{bmatrix} A'_k & C'_k \\ B'_k & D'_k \end{bmatrix} = \begin{bmatrix} V_k^* & \\ & I \end{bmatrix} \begin{bmatrix} A_k & C_k \\ B_k & D_k \end{bmatrix} \begin{bmatrix} V_{k+1} & \\ & I \end{bmatrix} , \qquad \Lambda'_{c,k} = V_k^* \Lambda_{c,k} V_k = \begin{bmatrix} \tilde{\Lambda}_k & \\ & 0 \end{bmatrix} .$$

Therefore, an equivalent but reduced size realization ($\tilde{q}_k$ states) is

$$\begin{bmatrix} \tilde{A}_k & \tilde{C}_k \\ \tilde{B}_k & \tilde{D}_k \end{bmatrix} = \begin{bmatrix} \tilde{V}_k^* & \\ & I \end{bmatrix} \begin{bmatrix} A_k & C_k \\ B_k & D_k \end{bmatrix} \begin{bmatrix} \tilde{V}_{k+1} & \\ & I \end{bmatrix} . \tag{11}$$

which has as controllability gramian $\tilde{\Lambda}_k$.

3.2. Square-root implementation of Λ. Equation (9) to update Λ_k can be implemented in square root form: let $\Lambda_k = M_k^* M_k$ where M_k is a full-rank factor (minimal number of rows). Define the SVD

$$\begin{bmatrix} M_k A_k \\ B_{F,k} \end{bmatrix} =: U_{k+1} \Sigma_{k+1} V_{k+1}^* = \tilde{U}_{k+1} \tilde{\Sigma}_{k+1} \tilde{V}_{k+1}^* \tag{12}$$

where U_{k+1} and V_{k+1} are square and unitary, and Σ_{k+1} is diagonal and of the same size as the matrix at the left hand side. Similarly $\tilde{U}_{k+1} \tilde{\Sigma}_{k+1} \tilde{V}_{k+1}^*$ is an SVD in 'economy size', where $\tilde{\Sigma}_{k+1}$ is square diagonal and contains the nonzero singular values in Σ_{k+1}, and $\tilde{U}_{k+1}$ and $\tilde{V}_{k+1}$ collect the corresponding columns of U_{k+1} and V_{k+1}. In terms of this factorization, we can define the recursion

$$M_{k+1} = \tilde{U}_{k+1}^* \begin{bmatrix} M_k A_k \\ B_{F,k} \end{bmatrix} = \tilde{\Sigma}_{k+1} \tilde{V}_{k+1}^* \,. \tag{13}$$

It is such that $M_{k+1}^* M_{k+1} = \Lambda_{k+1}$ as required by (9). Moreover, $\Lambda_k = \tilde{V}_k \tilde{\Sigma}_k^2 \tilde{V}_k^*$, which has a correspondence with (10). Therefore, we can use $\tilde{V}_k$ as a state transformation as in (11) to reduce the number of states of the resulting realization of F. The controllability gramian of this realization is $\tilde{\Sigma}_k^2$.

3.3. Balanced model reduction. Assume the realization of F is in output normal form ($\Lambda_{o,k} = I$, valid e.g., for the trivial realization), and the input gramian is $\Lambda_k = M_k^* M_k$. Let H_k be the k-th Hankel matrix of F. Since

$$H_k = \mathcal{C}_k \mathcal{O}_k \,, \qquad \Lambda_{c,k} = \mathcal{C}_k^* \mathcal{C}_k \,, \qquad \Lambda_{o,k} = \mathcal{O}_k \mathcal{O}_k^*$$

it follows immediately that the nonzero eigenvalues of $H_k H_k^*$ are equal to the nonzero eigenvalues of $\Lambda_{c,k}\Lambda_{o,k}$. Since in the trivial realization for F we have $\Lambda_{o,k} = I$, it follows for this realization that the nonzero singular values of H_k are equal to the singular values of M_k.

Therefore, if we compute a singular value decomposition of M_k and truncate the singular values at a threshold γ (setting those that are smaller than γ to zero), then in fact this is equal to reducing the rank of the Hankel matrices of F, and $\|\tilde{H}_k - H_k\| = \|\tilde{M}_k - M_k\|$, in 2-norm or Frobenius norm. This technique is known as balanced model reduction.[1]

Suppose q_k singular values are larger than γ, and let $\tilde{M}_k$ be the rank-q_k approximation, with corresponding right singular vectors $\tilde{V}_k$. Then the corresponding realization of the approximant of F is given by (11). This realization has q_k states and is minimal.

To reduce the complexity of the computations, a further approximation is needed, namely in the propagation of M_k to M_{k+1}, equation (13). Here we propose to use the approximation $\tilde{M}_k$ as well. The effect of this approximation is *not* predicted by the theory of balanced model reduction. Therefore the approximation error $\|\tilde{H}_k - H_k\|$ will be somewhat larger than γ.

For future use, we will also apply to the realization in (11) a subsequent state transformation by $\tilde{\Sigma}_k^{-1}$, which will give

$$\begin{bmatrix} \tilde{A}_k & \tilde{C}_k \\ \tilde{B}_k & \tilde{D}_k \end{bmatrix} = \begin{bmatrix} \tilde{\Sigma}_k \tilde{V}_k^* & \\ & I \end{bmatrix} \begin{bmatrix} A_k & C_k \\ B_{F,k} & D_{F,k} \end{bmatrix} \begin{bmatrix} \tilde{V}_{k+1} \tilde{\Sigma}_{k+1}^{-1} & \\ & I \end{bmatrix} \tag{14}$$

The controllability gramian of this realization is $I_{q\times q}$. Note that $\tilde{\Sigma}_k \tilde{V}_k^* = \tilde{M}_k$, and since $[A_k \;\; C_k] = I$, we have $\tilde{M}_k [A_k \;\; C_k] = [M' \;\; \mathbf{m}]$, a partitioning of $\tilde{M}_k$ where $\mathbf{m}$ is the last column of $\tilde{M}_k$ and M' contains the preceding columns. Further note that (using equation (11))

$$\begin{bmatrix} \tilde{M}_k A_k \\ B_{F,k} \end{bmatrix} \tilde{V}_{k+1} \tilde{\Sigma}_{k+1}^{-1} = \tilde{U}_{k+1} \tag{15}$$

We have obtained the following algorithm.

[1] In the theory of balanced model reduction, one would first apply a state transformation to make $\Lambda'_{c,k} = \Lambda'_{o,k} = \Sigma$ (diagonal), then truncate Σ and select the corresponding submatrices of the realization. Since in our case $\Lambda_{o,k}$ is already diagonal, this procedure is actually equivalent to truncating the rank of M_k.

3.4. Algorithm 1 (order n^2q^2). Given a positive matrix T (size $n \times n$), the following algorithm computes a realization for an upper triangular approximate factor F and its inverse $S = F^{-1}$. The threshold γ determines the number of states in the realization of F and S as q_k, approximately corresponding to the number of singular values of the Hankel matrices of T larger than γ.

```
 1.  $q_1 = 0, \quad M_1 = \cdot \qquad (0 \times n)$
 2.  for $k = 1$ to $n$ do
 3.      $B = [T_{k,n} \cdots T_{k,k+1}]$                                              $1 \times n-k$
 4.      $D = T_{kk}$
 5.      $M_k =: [M' \quad \mathbf{m}]$                                                 $q_k \times n-k+1$
 6.      $D_F = (D - \mathbf{m}^*\mathbf{m})^{1/2}$
 7.      $B_F = D_F^{-*}(B - \mathbf{m}^*M')$
 8.      $X := \begin{bmatrix} M' \\ B_F \end{bmatrix} =: U\Sigma V^*$ (economy-size SVD)       $q_k+1 \times n-k$
 9.      $q_{k+1} = \#(\sigma_i > \gamma), \quad \tilde{U} = U(:, 1:q_{k+1})$                       $q_k+1 \times q_{k+1}$
10.      $M_{k+1} = \tilde{U}^*X = \tilde{\Sigma}\tilde{V}^*$                                     $q_{k+1} \times n-k$
11.      $\begin{bmatrix} A_{F,k} & C_{F,k} \\ B_{F,k} & D_{F,k} \end{bmatrix} = \left[\begin{array}{c|c} \tilde{U} & \begin{matrix} \mathbf{m} \\ D_F \end{matrix} \end{array}\right]$
12.      $\begin{bmatrix} A_{S,k} & C_{S,k} \\ B_{S,k} & D_{S,k} \end{bmatrix} = \begin{bmatrix} A_{F,k} - C_{F,k}D_{F,k}^{-1}B_{F,k} & -C_{F,k}D_{F,k}^{-1} \\ D_{F,k}^{-1}B_{F,k} & D_{F,k}^{-1} \end{bmatrix}$
13.  end
```

The computational complexity is determined by the SVD of X (size $q_k \times n-k$) in step 8, which is of order $O(nq^2)$, where q is the average number of states q_k. The total complexity of the algorithm is $O(n^2q^2)$.

Interpreting the steps in the algorithm, we see that M_k is a basis of the (dominant) row span of the k-th Hankel matrix of F. In going to the $k+1$-st Hankel, the tail ($\mathbf{m}$) is chopped, and a new row is added. The SVD in step 8 incorporates the new row and a minimal basis is kept. The new basis is $M_{k+1} = \tilde{\Sigma}\tilde{V}^*$, whereas $\tilde{U}$ relates the old basis to the new, and becomes part of the realization of the factor.

4. Reduced complexity.

4.1. Derivation. In the complexity of the algorithm (n^2q^2), the factor n^2 cannot be avoided: all entries of the matrix have to be visited at least once. However, the factor q^2 can be reduced to q, by using the fact that the q_k rows of M_k are orthogonal. The necessary modifications to the algorithm are discussed in this section.

Thus introduce a QR factorization: $M_k = R_kQ_k$, where $R_k : q_k \times q_k$, and $Q_k : q_k \times n-k+1$, with orthonormal rows. Initially, $R_k = \tilde{\Sigma}_k$ (diagonal) and $Q_k = \tilde{V}_k^*$ (orthonormal), but during the computations this may change. We update R_k and Q_k separately, and minimize operations on Q_k.

The following subsections list the required changes to the algorithm.

Step 5: $M_k =: [M' \quad \mathbf{m}]$. Let $M_k = R_kQ_k$ with Q_k having orthonormal rows. We aim to have

$$M_k =: [M' \quad \mathbf{m}] = R_k[Q' \quad \alpha\mathbf{e}_1]$$

where $\mathbf{e}_1$ is a unit vector in the direction of the x-axis. Note that Q' has orthonormal rows (up to a scaling of the first row), so that we can continue in subsequent steps with the factorization $M' = R_kQ'$.

Starting from $M_k = R_k Q_k$, we compute a sequence of $q_k - 1$ Givens rotations G to map $\mathbf{m}$ to $\alpha\mathbf{e}_1$. The Givens rotations are applied both to R_k and to Q_k. The result is

$$R_k := R_k G^* \,; \quad Q_k := G Q_k =: [Q' \quad \alpha\mathbf{e}_1]$$

If $\alpha = 1$, then the first row of Q' is equal to zero and should be dropped. Also, if Q' is tall its dimension must be reduced by dropping the first row.

Step 8: $X := \begin{bmatrix} M' \\ B_F \end{bmatrix}$**.** We aim to have

$$X = \begin{bmatrix} M' \\ B_F \end{bmatrix} = \begin{bmatrix} R_k Q' \\ B_F \end{bmatrix} =: RQ$$

Therefore the new row B_F must be made orthogonal to the rows of Q'. This is obtained by a Gram-Schmidt orthonormalization of B_F versus the basis Q',

$$\begin{array}{lcl} \mathbf{b} & := & B_F Q'^* \\ \beta\mathbf{q}_1 & := & B_F - \mathbf{b}Q', \qquad \mathbf{q}_1 \text{ normalized} \\ R & := & \begin{bmatrix} R & 0 \\ \mathbf{b} & \beta \end{bmatrix}, \quad Q := \begin{bmatrix} Q' \\ \mathbf{q}_1 \end{bmatrix} \end{array}$$

If $\beta = 0$ then B_F was already contained in the subspace spanned by the rows of Q'. In that case, $\mathbf{q}_1$ and the corresponding column of the new R must be dropped.

Step 8: $X =: U\Sigma V^*$ **.** Since Q is orthonormal, this step is easily carried out on R.

$$R =: U\Sigma V^* \qquad (\text{size } q+1 \times q+1)$$

Step 10: $M_{k+1} = \tilde{\Sigma}\tilde{V}^*$**.** After step 8, step 10 is now

$$M_{k+1} = \tilde{\Sigma}\tilde{V}Q = RQ \qquad (\tilde{\Sigma} : q \times q, \ \tilde{V} : q \times q+1)$$

Since the rows of Q have length of order n, and $\tilde{V}$ is full, a direct multiplication $Q := \tilde{V}Q$ is too expensive. But (i) $\tilde{V}$ is a submatrix of a unitary matrix, and (ii) we only have to compute a basis for the subspace spanned by $\tilde{V}Q$, so we can apply some multiplications on R instead. In fact all row operations applied to $\tilde{V}$ (and matched by column operations on R) can be used to reduce the number of computations.

We start by setting $R = \tilde{\Sigma}$ and then update R and $\tilde{V}$ to reduce the complexity of $\tilde{V}$. Compute a sequence of Givens rotations G acting on the rows of $\tilde{V}$ such that entries $\tilde{V}_{j,i}$ below the main diagonal of $\tilde{V}$ are zeroed against the pivot entries $\tilde{V}_{i,i}$ on the main diagonal:

$$R := RG^*, \qquad \tilde{V} := G\tilde{V} = \left[\begin{array}{ccc|c} * & \cdots & * & * \\ 0 & \ddots & \vdots & \vdots \\ 0 & 0 & * & * \end{array}\right]$$

For this, $\frac{1}{2}q(q-1)$ rotations are needed. The complexity of $\tilde{V}$ cannot be reduced by row operations any further. Subsequently, column operations are applied to $\tilde{V}$ and matched by similar operations on Q, to cancel all entries beyond the q-th column of

$\tilde{V}$. Using the diagonal entries of $\tilde{V}$ as pivots and starting at the (q,q)-th entry, we obtain after q rotations

$$\tilde{V} := \tilde{V}G = \left[\begin{array}{ccc|c} * & \cdots & * & 0 \\ 0 & \ddots & \vdots & \vdots \\ 0 & 0 & * & 0 \end{array}\right], \qquad Q := G^*Q$$

We can ensure that the diagonal entries are real and positive. Since the rows of $\tilde{V}$ are orthonormal, it follows that at this point in fact

$$\tilde{V} = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

We finish by reducing the size of Q appropriately:

$$Q := \tilde{V}Q = Q(1:q,:)$$

We have thus obtained $M_{k+1} = RQ$, with a complexity of order q rotations on the rows of Q. The resulting algorithm appears in the following subsection.

4.2. Algorithm 2 (order n^2q).

```
1.   $q = 0, \quad R = \cdot \quad (0 \times 0), \quad Q = \cdot \quad (0 \times n)$
2.   for $k = 1$ to $n$ do
3.      $B = [T_{k,n} \cdots T_{k,k+1}], \quad D = T_{kk}$
4.      $Q =: [Q \quad \mathbf{q}]$                                   $q \times (n-k+1)$
        $\mathbf{m} = R\mathbf{q}$                                     $q \times 1$
5.      $D_F = (D - \mathbf{m}^*\mathbf{m})^{1/2}, \quad B_F = D_F^{-*}[B - (\mathbf{m}^*R)Q]$
6.      for $i = q-1, \cdots, 1$ do
           $G_i = \text{givens}([\mathbf{q}_i, \mathbf{q}_{i+1}]^T)$
           $\mathbf{q}([i, i+1]) = G_i\mathbf{q}([i, i+1])$
           $Q([i, i+1], :) = G_iQ([i, i+1], :)$
           $R(:, [i, i+1]) = R(:, [i, i+1])G_i^*$
        end
        $\alpha = \|Q(1, :)\|$
        if $\alpha > 0$ and $Q$ square or wide
           $Q(1, :) = (1/\alpha)Q(1, :)$                               $q \times (n-k)$
           $R(:, 1) = \alpha R(:, 1)$                                  $q \times q$
        else
           $Q = Q(2 : q, :)$                                          $(q-1) \times (n-k)$
           $R = R(:, 2 : q)$                                          $q \times (q-1)$
        end
7.      $\mathbf{b} = B_FQ^*$                                          $1 \times q$
        $\mathbf{q} = B_F - \mathbf{b}Q$                                $1 \times (n-k)$
        $\alpha = \|\mathbf{q}\|$
        if $\alpha > 0$ and $Q$ square or wide
           $R := \begin{bmatrix} R & 0 \\ \mathbf{b} & \alpha \end{bmatrix}, \quad Q := \begin{bmatrix} Q \\ (1/\alpha)\mathbf{q} \end{bmatrix}$      $R : (q+1) \times (q+1)$ or $(q+1) \times q$
                                                              $Q : (q+1) \times (n-k)$ or $q \times (n-k)$
        else
           $R := \begin{bmatrix} R \\ \mathbf{b} \end{bmatrix}$                                  $R : (q+1) \times q$ or $(q+1) \times (q-1)$
                                                              $Q : q \times (n-k)$ or $(q-1) \times (n-k)$
        end
8.      $R =: U\Sigma V$ (economy-size SVD)                          $(q+1) \times m$, where $m \in \{q+1, q, q-1\}$
9.      $q = \#(\sigma_i > \gamma)$
        $\tilde{U} = U(:, 1 : q), \ \tilde{\Sigma} = \Sigma(1 : q, 1 : q), \ \tilde{V} = V(1 : q, :)$      $\tilde{V} : q \times m$
10.     $R = \tilde{\Sigma}$
        for $i = 1, \cdots, q-1$ do
           for $j = i+1, \cdots, q$ do
              $G_{ij} = \text{givens}([\tilde{V}_{i,i}, \tilde{V}_{j,i}]^T)$
              $\tilde{V}([i, j], :) = G_{ij}\tilde{V}([i, j], :)$
              $R(:, [i, j]) = R(:, [i, j])G_{ij}^*$
           end
        end
        for $i = q, \cdots, 1$ do
           for $j = q+1, \cdots, m$ do
              $G_{ij} = \text{givens}([\tilde{V}_{i,i}, \tilde{V}_{i,j}]^T)$
              $\tilde{V}(:, [i, j]) = \tilde{V}(:, [i, j])G^*$
              $Q([i, j], :) = GQ([i, j], :)$
           end
        end
        $Q = Q(1 : q, :)$
11.     $\begin{bmatrix} A_{F,k} & C_{F,k} \\ B_{F,k} & D_{F,k} \end{bmatrix} = \left[\begin{array}{c|c} \tilde{U} & \begin{matrix} \mathbf{m} \\ D_F \end{matrix} \end{array}\right]$      $q+1 \times q+1$
12.     $\begin{bmatrix} A_{S,k} & C_{S,k} \\ B_{S,k} & D_{S,k} \end{bmatrix} = \begin{bmatrix} A_{F,k} - C_{F,k}D_{F,k}^{-1}B_{F,k} & -C_{F,k}D_{F,k}^{-1} \\ D_{F,k}^{-1}B_{F,k} & D_{F,k}^{-1} \end{bmatrix}$
13.  end
```

5. Conclusions. We have derived an algorithm for the approximate inversion of a general strictly positive matrix. The approximation involves a low-rank truncation of the off-diagonal submatrices H_k. This approximation is well controlled (threshold γ) but we do not have an expression for the final approximation error.

The computational complexity is order n^2q as opposed to n^3.

The numerical stability of the algorithm needs to be studied. There is one weak point, the computation of $D_F = (D - \mathbf{m}^*\mathbf{m})^{1/2}$ and its subsequent inversion to obtain B_F. Clearly, $D - \mathbf{m}^*\mathbf{m}$ should should remain strictly positive (condition corresponds to positivity of T). In theory this is guaranteed but if the matrix is close to singular numerical problems may occur.

REFERENCES

[1] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, and A.-J. van der Veen, *Fast stable solver for sequentially semi-separable linear systems of equations*, in Lecture Notes in Computer Science, vol. 2552, Springer Verlag, Heidelberg, 2002.

[2] S. Chandrasekaran and M. Gu, *A divide-and-conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semiseparable matrices*, accepted for Numerische Mathematik, (1999).

[3] ———, *Fast and stable eigendecomposition of symmetric banded plus semi-separable matrices*, Linear Algebra and its Applications, 313 (2000), pp. 107–114.

[4] ———, *A fast and stable solver for recursively semi-separable systems of equations*, Contemporary Mathematics series, AMS publications, 2001.

[5] ———, *Fast and stable algorithms for banded plus semiseparable systems of linear equations*, SIAM Journal on Matrix Analysis and Applications, 25 (2003), pp. 373–384.

[6] P. Dewilde and A.J. van der Veen, *On the Hankel-norm approximation of upper-triangular operators and matrices*, Integral Equations and Operator Theory, 17 (1993), pp. 1–45.

[7] P.M. Dewilde and A.J. van der Veen, *Time-varying systems and computations*, Kluwer academic publishers, Dordrecht, June 1998.

[8] P. Dewilde and A.J. van der Veen, *Inner-outer factorization and the inversion of locally finite systems of equations*, Linear Algebra Appl., 313 (2000), pp. 53–100.

[9] Y. Eidelman and I. Gohberg, *Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices*, Computers and Mathematics with Applications, 33 (1997), pp. 69–79.

[10] ———, *A look-ahead block Schur algorithm for diagonal plus semiseparable matrices*, Computers and Mathematics with Applications, 35 (1998), pp. 25–34.

[11] ———, *On a new class of structured matrices*, Integral Equations and Operator Theory, 34 (1998), pp. 293–324.

[12] ———, *A modification of the dewilde van der veen method for inversion of finite structured matrices*, Linear Algebra and its Applications, 343-344 (2002), pp. 419–450.

[13] T. Kailath I. Gohberg and I. Koltracht, *Linear complexity algorithms for semiseparable matrices*, Integral Equations and Operator Theory, 8 (1985), pp. 780–804.

[14] A.J. van der Veen, *Time-varying lossless systems and the inversion of large structured matrices*, Archiv f. Elektronik u. Übertragungstechnik, 49 (1995), pp. 372–382.