UTV Expansion Pack

Special-Purpose Rank Revealing Algorithms

Version 1.0 for MATLAB 6.5

Ricardo D. Fierro

Department of Mathematics
California State University San Marcos
San Marcos, CA 92096

Per Christian Hansen

Department of Mathematical Modelling
Building 321, Technical University of Denmark
DK-2800 Lyngby, Denmark

April 2004

Contents

Abstract

This collection of Matlab software supplements and complements the package UTV Tools from 1999, and includes implementations of special-purpose rank-revealing algorithms developed since the publication of the original package. We provide algorithms for computing and modifying symmetric rank-revealing VSV decompositions, we expand the algorithms for the ULLV decomposition of a matrix pair to handle interference-type problems with a rank-deficient covariance matrix, and we provide a robust and reliable Lanczos algorithm which – despite its simplicity – is able to capture all the dominant singular values of a sparse or structured matrix. These new algorithms have applications in signal processing, optimization and LSI information retrieval. The corresponding manuscript is:

- R. D. Fierro and P. C. Hansen, *UTV Expansion Pack: Special-Purpose Rank-Revealing Algorithms*, submitted to Numerical Algorithms.

Acknowledgements

This work was supported by travel grants from Julie Damms Studiefond and Jubilæumsfonden ved DTU. The routine lqlp was written by David Huckaby, and we acknowledge the permission to include it here. We also thank Nicholas J. Higham for permission to include routines cholp and ldlt_symm from his Matrix Computation Toolbox.

1. Introduction

The Matlab package UTV Tools [5] from 1999 provides a collection of algorithms for computing and modifying (i.e., up- and downdating) rank-revealing decompositions of general matrices. These decompositions have many applications in signal processing, where they are used as fast and reliable alternatives [9], [20] to the versatile but computationally expensive and hard-to-update singular value decomposition (SVD).

Since the publication of UTV Tools more work has been done in the area of rank-revealing decompositions and algorithms. This work is motivated by the interest in using specialized rank-revealing algorithms, designed to take advantage of the underlying structure of the problem in consideration. The present package provides Matlab implementations of some of these newly developed algorithms, with emphasis on algorithms that expand the application areas of the original package (hence the name of the new package). Similar to the first package, the routines in this package can be considered as templates for more specialized implementations, perhaps in other computer languages, that can exploit the computer hardware.

Symmetric rank-revealing VSV decompositions for semidefinite and indefinite matrices were developed in [13] and [16] to provide algorithms and decompositions that take into account the symmetry of the matrix. Compared to the general UTV decompositions, the VSV decompositions lead to savings in computer time as well as advantages in the approximation properties of reduced-rank matrix approximations derived from the symmetric decompositions. The rank and subspace information provided by the VSV decompositions have applications, e.g., in deflation methods for solving block-structured symmetric indefinite systems [7] arising in optimization algorithms and PDE solvers.

The rank-revealing ULLV decomposition was originally developed for revealing the rank of a matrix quotient, defined as the product of one matrix and the pseudoinverse of another matrix, and with applications in noise reduction problems with broadband noise where the noise covariance matrix has full rank. When the noise covariance matrix is rank deficient (which is the case for interference or narrow-band noise) then the correct matrix quotient involves a weighted pseudoinverse [10], and the corresponding ULLV decomposition must reflect this. The most convenient way to deal with the full-rank and the rank deficient cases is to provide two different ULLV algorithms for the two variations of the decomposition.

While rank-revealing decompositions are convenient tools for dense matrices, they may be less suited for large sparse or structured matrices. For this reason we also provide a Lanczos algorithm for computing the dominant singular triplets of a matrix. Our algorithm demonstrates that if such an algorithm is based on complete reorthogonalization and explicit restarts, then the code need not be very complicated. The core of our implementation requires less than 100 lines of Matlab code, has a simple structure, and is thus suited for implementation on dedicated hardware platforms (in contrast to many other sophisticated – and much more general – implementations in mathematical software libraries).

In addition to the above algorithms, and for completeness, we provide implementations of a few simple and "heuristic" algorithms which will often reveal the numerical rank, but without the safety (and slight overhead) of a genuine rank-revealing algorithm.

Finally we provide a few scripts that demonstrate the use of our functions in connection with rank-deficient KKT systems in optimization, noise and interference reduction in signal processing, and signal extraction in NMR signals.

In the following sections we summarize the algorithms, giving new theory where it is needed. We conclude with a few numerical examples and an overview of the new package. Throughout the paper, the norm $\|\cdot\|$ denotes the 2-norm, while I and E denote the identity matrix and the exchange matrix (consisting of the columns of the identity matrix in reverse order). Moreover, L and R always denote lower and upper triangular matrices, V is always an orthogonal matrix, and Ω is always a signature matrix. We also make use of Matlab's colon notation to indicate submatrices.

2. The Symmetric VSV Decomposition

The rank-revealing VSV decomposition of a symmetric matrix was first discussed by Luk and Qiao [16] (for Toeplitz matrices). A careful study of various algorithms based on initial triangular factorization can be found in [13], while a study of the accuracy of approximations based on the VSV decomposition is given in [4]

2.1. Definitions

Assume that the symmetric matrix $A \in \mathbb{R}^{n\times n}$ has numerical rank k, i.e., there is a well-defined gap between the kth singular value σ_k and the next. Then the rank-revealing VSV decomposition of A takes the form

$$A = V\,S\,V^T, \qquad S = \begin{pmatrix} S_{11} & S_{12} \\ S_{12}^T & S_{22} \end{pmatrix}, \qquad V = (\,V_1\,,\,V_2\,), \tag{2.1}$$

where the symmetric matrix S is partitioned such that it reveals the numerical rank of A, i.e., the singular values of the $k \times k$ leading submatrix S_{11} approximate the first k singular values of A, while the norms $\|S_{12}\|$ and $\|S_{22}\|$ of the off-diagonal and trailing blocks are both of the order σ_{k+1}, cf. [13], [16].

The matrix V is orthogonal, and it is partitioned such that the column spaces of the two blocks V_1 and V_2 are approximations to the subspaces spanned by the first k and the last $n-k$ right singular vectors of A, respectively. In the signal processing literature, these two subspaces are referred to as the signal and noise subspaces. See [4] concerning the accuracy of these approximations.

For practical purposes, we choose to compute and represent the matrix S in the factored form

$$S = T^T \Omega\, T, \tag{2.2}$$

in which T is an upper or lower triangular matrix, and Ω is a signature matrix, i.e., a diagonal matrix with ± 1 on the diagonal, such that the inertia of A is preserved in the inertia of Ω. If A is positive definite then Ω is the identity matrix.

For semidefinite matrices, it was found in [13] that the optimal form of T is lower triangular, because this choice leads to more accurate approximations of the signal and noise subspaces. Our package therefore includes software for computing the VSV decomposition $A = V\,L^T L\,V^T$ of a symmetric semidefinite matrix. We provide two functions hvsvsd and lvsvsd, optimized for the high-rank case ($k \approx n$) and low-rank case ($k \ll n$), respectively.

For indefinite matrices, the singular vector estimation (which is part of the VSV algorithm) is simpler when T is upper triangular, while a lower triangular T provides a decomposition that is consistent with the semidefinite case. We provide high-rank VSV algorithms for both forms: the functions hvsvid_L and hvsvid_R compute the lower triangular form $A = V\,L^T \Omega\, L\,V^T$ and the upper triangular form $A = V\,R^T \Omega\, L\,R^T$, respectively. In the

LOW-RANK VSV ALGORITHM lvsvid:

1. Compute $A = V\,L^T\Omega\,L\,V^T$ and let $k \leftarrow 1$.
2. Condition estimation: let $\widetilde{\sigma}_k$ estimate $\|L(k{:}n,k{:}n)^T\Omega(k{:}n,k{:}n)L(k{:}n,k{:}n)\|$ and let w_k estimate the corresponding right singular vector.
3. If $\widetilde{\sigma}_k < \tau$ then $k \leftarrow k-1$, **exit**.
4. Revealment: determine orthogonal P_k such that $P_k^T w_k = (1,0,\ldots,0)^T$;
5. update $L(k{:}n,k{:}n) \leftarrow L(k{:}n,k{:}n)P_k$ and $V(:,k{:}n) \leftarrow V(:,k{:}n)P_k$;
6. update $L(k{:}n,k{:}n) \leftarrow Q_k^T L(k{:}n,k{:}n)$, $\Omega(k{:}n,k{:}n) \leftarrow Q_k^T\Omega(k{:}n,k{:}n)Q_k$, where the hypernormal Q_k ensures that the updated L is triangular.
7. Deflation: let $k \leftarrow k+1$.
8. Go to step 2.

HIGH-RANK VSV ALGORITHMS hvsvid_T WITH T = L or R:

1. Compute $A = V\,T^T\Omega\,T\,V^T$ and let $k \leftarrow 1$.
2. Condition estimation: let $\widetilde{\sigma}_k$ estimate $\sigma_{\min}\left(T(1{:}k,1{:}k)^T\Omega(1{:}k,1{:}k)T(1{:}k,1{:}k)\right)$ and let w_k estimate the corresponding right singular vector.
3. If $\widetilde{\sigma}_k > \tau$ then **exit**.
4. Revealment: determine orthogonal P_k such that $P_k^T w_k = (0,\ldots,0,1)^T$;
5. update $T(1{:}k,1{:}k) \leftarrow T(1{:}k,1{:}k)P_k$ and $V(:,1{:}k) \leftarrow V(:,1{:}k)P_k$;
6. update $T(1{:}k,1{:}k) \leftarrow Q_k^T T(1{:}k,1{:}k)$ and $\Omega(1{:}k,1{:}k) \leftarrow Q_k^T\Omega(1{:}k,1{:}k)Q_k$, where the hypernormal Q_k ensures that the updated T is triangular.
7. Deflation: let $k \leftarrow k-1$.
8. Go to step 2.

Figure 2.1: The VSV algorithms for symmetric indefinite matrices.

low-rank case the dilemma vanishes, and the function lvsvid computes the lower triangular form.

An alternative, but more expensive, approach to computing a high-rank indefinite VSV decomposition with a lower triangular T is to first compute the upper triangular form $A = V\,R^T\Omega\,R\,V^T$ and then compute the QR factorization $R^T = Q\,L^T$ which yields the lower triangular form $A = (VQ)\,L^T\Omega\,L\,(VQ)^T$. This approach is easy to implement using Matlab's qr function, but it is more expensive and therefore we do not provide an implementation.

2.2. Algorithms

The generic algorithm for computing the VSV decomposition of a symmetric semidefinite matrix is quite simple, because the singular values of T are the square roots of the singular values of A when $\Omega = I$. First we compute the symmetrically pivoted Cholesky factorization $\Pi\,A\,\Pi^T = \overline{C}^T\overline{C}$ (we use rook pivoting as implemented in [14]), followed by the computation of the rank-revealing ULV decomposition $E\,\overline{C}\,E = \widehat{U}\,L\,\widehat{V}^T$ (using high-rank and low-rank functions from UTV Tools). As a result, we obtain the desired decomposition $A = (\Pi\widehat{V})\,L^T L\,(\Pi\widehat{V})^T$.

The generic algorithm for indefinite matrices starts with a symmetrically pivoted LDL^T factorization $\Pi\,A\,\Pi^T = \overline{L}\,\overline{D}\,\overline{L}^T$, using the rook pivoting implemented in [14]. Next, the middle block diagonal matrix $\overline{D}$ is replaced by the signature matrix, $\overline{L}\,\overline{D}\,\overline{L}^T = \widehat{W}\,\widehat{L}\,\widehat{\Omega}\,\widehat{L}^T\widehat{W}^T$, where $\widehat{W}$ is an orthogonal block diagonal matrix with 1×1 and 2×2 blocks on the diagonal; see §4.2 in [13].

Finally we reveal the rank of the product $\widehat{L}\,\widehat{\Omega}\,\widehat{L}^T$, by "peeling off" the small or large singular values one at a time. Our algorithms take basis in the following two reformulations and partitionings with $R = \widehat{L}^T$, $L = E\,\widehat{L}^T E$ and $\Omega = E\,\widehat{\Omega}\,E$:

$$\begin{aligned} R^T\widehat{\Omega}\,R &= \begin{pmatrix} R_{11}^T & 0 \\ R_{12}^T & R_{22}^T \end{pmatrix} \begin{pmatrix} \widehat{\Omega}_1 & 0 \\ 0 & \widehat{\Omega}_2 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \\ &= \begin{pmatrix} R_{11}^T\widehat{\Omega}_1 R_{11} & R_{11}^T\widehat{\Omega}_1 R_{12} \\ R_{12}^T\widehat{\Omega}_1 R_{11} & R_{12}^T\widehat{\Omega}_1 R_{12} + R_{22}^T\widehat{\Omega}_2 R_{22} \end{pmatrix} \end{aligned} \tag{2.3}$$

$$\begin{aligned} L^T\Omega\,L &= \begin{pmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{pmatrix} \begin{pmatrix} \Omega_1 & 0 \\ 0 & \Omega_2 \end{pmatrix} \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \\ &= \begin{pmatrix} L_{11}^T\Omega_1 L_{11} + L_{21}^T\Omega_2 L_{21} & L_{21}^T\Omega_2 L_{22} \\ L_{22}^T\Omega_2 L_{21} & L_{22}^T\Omega_2 L_{22} \end{pmatrix}. \end{aligned} \tag{2.4}$$

The indefinite VSV algorithms are summarized in Fig. 2.1. Following the ideas from [3] in the low-rank case, we can now determined orthogonal transformations such that they, when applied symmetrically to $L^T\Omega\,L$, ensure that the singular values of the (1,1)-block in (2.4) approximate the largest singular values of A. The construction of these transformations involves the computation of the largest singular value and corresponding right singular vector of the submatrix $L_{22}^T\Omega_2 L_{22}$, and the user can choose between power iterations and the Lanczos method. At the same time, hypernormal rotations are used to maintain the triangular form of L.

In the high-rank case we follow the ideas from [16] and construct orthogonal transformations which ensure that the smallest singular values of A are revealed in the (2,2)-block in (2.3) and (2.4). This involves the computation of the smallest singular value and corresponding right singular vector of the (1,1)-block. In the upper triangular case (2.3) this is done by means of inverse iterations applied to the submatrix $R_{11}^T\widehat{\Omega}_1 R_{12}$.

In the lower triangular case (2.4), however, it is impractical to apply inverse iterations to the submatrix $L_{11}^T\Omega_1 L_{11} + L_{21}^T\Omega_2 L_{21}$ because we do not have a useful factorization of this matrix. The inverse iterations are much easier to use when we can ignore the second term, but unfortunately this is not always the case: according to Thm. 4.3 in [13] $\|L_{21}^T\Omega_2 L_{22}\|$ and $\|L_{22}^T\Omega_2 L_{22}\|$ are guaranteed to be small, but this does not imply that $\|L_{21}^T\Omega_2 L_{21}\|$ is small. Our solution is to apply a single step of block QR refinement to L, as described below; this ensures that the norm $\|L_{21}\|$ of the off-diagonal block in L is always small.

2.3. Hypernormal Rotations and Their Break-Down

Hypernormal transformations are introduced in [2], and their use in our VSV algorithms is discussed in [13]. The "building blocks" of hypernormal transformations are Givens and hyperbolic rotations, the latter performing the transformation (for $|\alpha| > |\beta|$):

$$\begin{pmatrix} c & -s \\ -s & c \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} (\alpha^2 - \beta^2)^{1/2} \\ 0 \end{pmatrix}$$

where $c^2 - s^2 = 1$. The hyperbolic transformation is not defined when $|\alpha| = |\beta|$, and it is has large elements $|\alpha|$ and $|\beta|$ when $|\alpha|$ is close to $|\beta|$.

In our algorithms, the hyperbolic transformations are used to annihilate fill in the triangular matrix during the revealment steps (see Fig. 2.1). Consider the following situation, where a right Givens rotation has introduced a nonzero element "$*$" in position $(i+1,i)$:

$$
\begin{array}{rc cccccc}
 & & \times & \times & \times & \times & \times & \times \\
 & & & \times & \times & \times & \times & \times \\
i & \rightarrow & & & \times & \times & \times & \times \\
i+1 & \rightarrow & & & * & \times & \times & \times \\
 & & & & & & \times & \times \\
 & & & & & & & \times
\end{array}
$$

If the fill satisfies $|r_{i+1,i}| \approx |r_{ii}|$ then we introduce large elements in the updated R which cancel in the product $R^T \Omega\, R$; an undesirable situation in numerical computations. Our remedy is to detect this situation and resort to a "fix." When $||r_{i+1,i}| - |r_{ii}|| < 10^{-5} \|R(i\colon i+1, i\colon i+1)\|$, we perform a cyclic permutation of columns i through $i+2$, leading to the form

$$
\begin{array}{rc cccccc}
 & & \times & \times & \times & \times & \times & \times \\
 & & & \times & \times & \times & \times & \times \\
i & \rightarrow & & & \times & \times & \times & \times \\
i+1 & \rightarrow & & & \times & \times & * & \times \\
 & & & & & \times & & \times \\
 & & & & & & & \times
\end{array}
$$

after which we use hypernormal transformations to annihilate the two elements below the diagonal in columns i and $i+1$. We then return to the condition estimation step and restart the revealment process. If we only permuted columns i and $i+1$ then the difficulty would arise again in the restarted revealment step.

2.4. Block QR Refinement

In analogy with block QR refinement of UTV decompositions, we can apply a similar algorithm to the VSV decompositions in order to reduce the norm of the off-diagonal blocks. We discuss the algorithm for the upper triangular version only; the algorithm for the lower triangular version is practically the same.

Given R partitioned as in (2.3) we first apply a sequence of right orthogonal transformations to annihilate the submatrix R_{12}, thus filling out the elements in the (2,1)-block. These elements, in turn, are annihilated by means of left hypernormal transformations which create new elements in the (1,2)-block.

We now justify this approach when applied to $S = R^T \Omega R$. Let $R = L_B Q_B$ where L_B is lower block triangular and Q_B is orthogonal; then $S = (R^T \Omega L_B) Q_B$ and a block QR step consists of formally forming the product $S_B \equiv Q_B (R^T \Omega L_B) = L_B^T \Omega L_B$. Next, let $L_B = H_B R_B$ where R_B is upper triangular and H_B is hypernormal with $H_B^T \Omega H_B = \Omega_B$ (in which Ω_B is a new signature matrix). Inserting this we obtain $S_B = R_B^T \Omega_B R_B$, showing that the new factors R_B and Ω_B indeed correspond to performing a block QR step on S.

The block QR refinement is implemented in the function `vsv_qrit` which determines whether it is applied to a semidefinite or an indefinite matrix and, in the latter case, whether it is applied to the L or R version.

2.5. Rank-One Modifications

We also provide algorithms for rank-one modifications of the form

$$A' = A + \omega\, v\, v^T, \qquad (2.5)$$

where $\omega = \pm 1$ and v a vector. Equation (2.5) can be recast as

$$A' = V \begin{pmatrix} T \\ v^T V \end{pmatrix}^T \begin{pmatrix} \Omega & 0 \\ 0 & \omega \end{pmatrix} \begin{pmatrix} T \\ v^T V \end{pmatrix} V^T.$$

When A is semidefinite and $\omega = 1$, the updating is equivalent to a rank-one update of the ULV decomposition where the numerical rank cannot decrease. The updating is implemented in function `vsvsd_up` and uses functions from UTV Tools.

When $\omega = -1$ or when A is indefinite then the numerical rank of A' can stay the same, or it can increase or decrease by one. Then the updated factors are computed by applying left hypernormal rotations to annihilate the row $v^T V$. This modification algorithm is implemented in the two functions `vsvid_L_mod` and `vsvid_R_mod`.

For efficiency reasons one should avoid to apply the rank-revealing post processing to the full S matrix. We partition $v^T V = d^T = (d_1^T, d_2^T)$ according to (2.3) and apply right Givens rotations G such that

$$d_2^T G^T = e_2^T = (\|d_2\|, 0, \ldots, 0)^T.$$

At the same time we apply left hypernormal rotations H to maintain the triangular form of the (2,2)-block. Introducing $R'_{12} = R_{12} G$, $R'_{22} = H^T R_{22} G$ and $\Omega'_2 = H^T \Omega_2 H$, we now have

$$\begin{aligned} V^T A' V &= \begin{pmatrix} I & 0 \\ 0 & G \end{pmatrix} \begin{pmatrix} R_{11} & R'_{12} \\ 0 & R'_{22} \\ d_1^T & e_2^T \end{pmatrix}^T \begin{pmatrix} \Omega_1 & 0 & 0 \\ 0 & \Omega'_2 & 0 \\ 0 & 0 & \omega \end{pmatrix} \begin{pmatrix} R_{11} & R'_{12} \\ 0 & R'_{22} \\ d_1^T & e_2^T \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & G \end{pmatrix}^T \\ &= \begin{pmatrix} R_{11}^T \Omega_1 R_{11} & R_{11}^T \Omega_1 R_{12} G + \omega d_1 e_2^T \\ G^T R_{12}^T \Omega_1 R_{11} + \omega e_2 d_1^T & G^T (R_{12}^T \Omega_1 R_{12} + R_{22}^T \Omega_2 R_{22}) G + \omega e_2 e_2^T \end{pmatrix}. \end{aligned}$$

Since $S = R^T \Omega\, R$ reveals the rank of A, we know that both norms $\|R_{11}^T \Omega_1 R_{12} G\| = \|R_{11}^T \Omega_1 R_{12}\|$ and $\|G^T (R_{12}^T \Omega_1 R_{12} + R_{22}^T \Omega_2 R_{22}) G\| = \|R_{12}^T \Omega_1 R_{12} + R_{22}^T \Omega_2 R_{22}\|$ are small. Hence, due to the structure of $d_1 e_2^T$ and $e_2 e_2^T$, it is not possible to have any elements of large magnitude in the last $n-k-1$ rows or columns of the above matrix. Therefore, once d^T has been annihilated, it suffices to reveal the rank of the leading $(k+1) \times (k+1)$ submatrix of the updated S factor.

3. The Gap-Revealing QLP Factorization

Stewart [21] introduced the so-called QLP factorization

$$A = Q\,L\,P^T \tag{3.1}$$

in which Q and P are orthogonal, and L is lower triangular. The factorization is gap revealing in the sense that the absolute values of the diagonal elements of L often track the singular values of A; but there is no guarantee that this is always the case. Hence, the factorization is not rank-revealing in the strict sense used in this package.

To compute the QLP decomposition, we compute a pivoted QR factorization $A\,\Pi_P = Q\,R$ followed by a second pivoted QR factorization $R^T\Pi_Q = P\,L^T$, and thus $A = (Q\Pi_Q)\,L\,(\Pi_P\,P)^T$. For high-rank matrices, this is easy to implement with Matlab's QR factorization, and it is implemented in function hqlp.

For low-rank matrices, Huckaby and Chan [15] implemented an algorithm using interleaved left and right Householder transformations. The algorithm essentially produces one row of L at a time, starting from the top, and stops as soon as a gap is revealed. A this stage, the heuristic is that if we compute the full QLP factorization then the norms of the (2,1)- and (2,2)-blocks of L will be small. Hence we can neglect these blocks and return the low-rank approximation

$$A_k = Q(:\,,1{:}k)\,L(1{:}k,1{:}k)\,P(:\,,1{:}k)^T, \tag{3.2}$$

where the gap appears between singular values σ_k and σ_{k+1}. This algorithm is implemented in function lqlp, and we emphasize that is computes the rank-k matrix *approximation* in (3.2), not a full factorization.

4. The ULLIV Decomposition

The ULLV decomposition of a matrix pair (A, B) with $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times n}$ was originally defined for the case $m \geq n \geq \mathrm{rank}(A)$ and $p \geq n = \mathrm{rank}(B)$ in [17]. Algorithms for computing and modifying this decomposition are included in UTV Tools.

In certain applications, such as interference reduction [10], [12], the matrix B does not have full column rank. This led Luk and Qiao [18] to define an alternative decomposition, which we shall refer to as the ULLIV decomposition. Assume again that $m \geq n \geq \mathrm{rank}(A)$ while B has full row rank, i.e., $\mathrm{rank}(B) = p < n$. Then the ULLIV decomposition takes the form

$$A = U_A\, L_A \begin{pmatrix} L & 0 \\ 0 & I \end{pmatrix} V^T, \qquad B = U_B\,(\,L\,,\,0\,)V^T \tag{4.1}$$

in which I is an identity matrix of order $n-p$, $U_A \in \mathbb{R}^{m\times n}$ has orthonormal columns, and $U_B \in \mathbb{R}^{p\times p}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal. Moreover, $L_A \in \mathbb{R}^{n\times n}$ and $L \in \mathbb{R}^{p\times p}$ are both lower triangular.

As shown in [10], when $\mathrm{rank}(B) < n$ it is the matrix quotient $A\,B_A^\dagger$ that is required, and whose numerical rank should be revealed. Here $B_A^\dagger$ is the A-weighted pseudoinverse of B. Given the ULLIV decomposition (4.1) it is proved in [10] that

$$A\,B_A^\dagger = U_A\, L_A(1{:}\,p, 1{:}\,p)\, U_B^T,$$

showing that the leading $p\times p$ block of L_A must be rank revealing. To compute such a ULLIV decomposition, we start with the QR factorization $B = (\,L\,,\,0\,)\,V^T$ followed by the QR factorization $A\,V \begin{pmatrix} L^{-1} & 0 \\ 0 & I \end{pmatrix} = U_A\,L_A$. Setting $U_B = I$ we thus have an initial decomposition, which is then made rank-revealing by applying the similar steps from the ULLV algorithm to L and U_B as well as to the first p columns of L_A, U_A and V. This algorithm is implemented in function ulliv.

An efficient algorithm for updating the ULLIV decomposition when a row a^T is appended to A is described in [18]. The algorithm takes its basis in the formulation

$$\begin{pmatrix} A \\ a^T \end{pmatrix} = \widetilde{U}_A \begin{pmatrix} L_A \\ d^T \end{pmatrix} \begin{pmatrix} L & 0 \\ 0 & I \end{pmatrix} V^T, \qquad B = U_B\,(\,L\,,\,0\,)\,V^T$$

with

$$\widetilde{U}_A = \begin{pmatrix} U_A & 0 \\ 0 & 1 \end{pmatrix}, \quad d^T = a^T V \begin{pmatrix} L^{-1} & 0 \\ 0 & I \end{pmatrix}.$$

In the first stage, the partial row $d(p+1{:}\,n)^T$ is annihilated by means of left Givens rotations which are absorbed in $\widetilde{U}_A$; these rotations maintain the small and rank-revealing elements in $L_A(1{:}p, 1{:}p)$. In the second stage, the remaining elements of the row d^T are annihilated by interleaved right and left Givens rotations – this stage is identical to the ULLV updating algorithm from UTV tools, and modifies U_B and L as well as the first p columns of $\widetilde{U}_A$, L_A

and V. Small elements are maintained in rows $k+2$ to p of L_A, where k is the numerical rank. In the third stage, which is also identical to that of ULLV updating, the numerical rank of the updated $L_A(1\colon p, 1\colon p)$ is revealed. The complete algorithm is implemented in function ulliv_up_A.

When a row b^T is appended to B then we must distinguish whether the rank increases or stays the same, because of our assumption that B has full row rank. An updating algorithm for the former case is described in [18]; we found it convenient to augment the algorithm with an additional stage, which simplifies the rank-revealing step. This algorithm takes its basis in the formulation

$$A = U_A \, (\, L_A \, , \, 0 \,) \begin{pmatrix} L & 0 \\ 0 & I \\ f_1^T & f_2^T \end{pmatrix} V^T, \qquad \begin{pmatrix} B \\ b^T \end{pmatrix} = \widetilde{U}_B \, \widetilde{I} \begin{pmatrix} L & 0 \\ 0 & I \\ f_1^T & f_2^T \end{pmatrix} V^T$$

with

$$(\, f_1^T \, , \, f_2^T \,) = b^T V, \qquad \widetilde{U}_B = \begin{pmatrix} U_B & 0 \\ 0 & 1 \end{pmatrix}, \qquad \widetilde{I} = \begin{pmatrix} I_p & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

In the first stage, we use right Givens rotations to reduce the partial row f_2^T to the form $(\phi, 0, \ldots, 0)$ with $\phi = \|f_2\|$. These rotations are also applied to the columns of V, and due to the presence of the matrix I the same rotations are propagated from the right to the last $n-p$ columns of L_A. The resulting fill is annihilated again by means of left Givens rotations which are absorbed in U_A.

We now interchange rows $p+1$ and $n+1$ of the third factor, as well as the same columns of the second factors. This results in factors with the structure

$$A = U_A \left(\begin{array}{ccccccccccc}
\times &&&&&&&&&& \\
\times & \times &&&&&&&&& \\
\times & \times & \times &&&&&&&& \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon &&&&&&& \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon &&&&&& \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon &&&&& \\
\times & \times & \times & \times & \times & \times &&&&& \times \\
\times & \times & \times & \times & \times & \times && \times &&& \times \\
\times & \times & \times & \times & \times & \times && \times & \times && \times \\
\times & \times & \times & \times & \times & \times && \times & \times & \times & \times
\end{array} \right) \left(\begin{array}{cccccccccc}
\times &&&&&&&&& \\
\times & \times &&&&&&&& \\
\times & \times & \times &&&&&&& \\
\times & \times & \times & \times &&&&&& \\
\times & \times & \times & \times & \times &&&&& \\
\times & \times & \times & \times & \times & \times &&&& \\
\times & \times & \times & \times & \times & \times & \times &&& \\
&&&&&&& 1 && \\
&&&&&&&& 1 & \\
&&&&&&&&& 1 \\
&&&&&& 1 &&&
\end{array} \right) V^T$$

$$\begin{pmatrix} B \\ b^T \end{pmatrix} = \widetilde{U}_B \begin{pmatrix} I_{p+1} & 0 \\ 0 & 0 \end{pmatrix} \left(\begin{array}{cccccccccc}
\times &&&&&&&&& \\
\times & \times &&&&&&&& \\
\times & \times & \times &&&&&&& \\
\times & \times & \times & \times &&&&&& \\
\times & \times & \times & \times & \times &&&&& \\
\times & \times & \times & \times & \times & \times &&&& \\
\times & \times & \times & \times & \times & \times & \times &&& \\
&&&&&&& 1 && \\
&&&&&&&& 1 & \\
&&&&&&&&& 1 \\
&&&&&& 1 &&&
\end{array} \right) V^T.$$

Note the zero column and the spike in the second factor of A. The ε symbols represent rows $k{+}1, \ldots, p$ with small elements that reveal the numerical rank k. In order to maintain as many small elements as possible in L_A we augment the Luk-Qiao algorithm by first performing a cyclic downshift of rows $k+1$ through $p+1$, and then we annihilate the resulting horizontal spike in row $k+1$ by means of right Givens rotations:

$$
\begin{pmatrix}
\times &&&&&&&&&& \\
\times & \times &&&&&&&&& \\
\times & \times & \times &&&&&&&& \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon &&&&&&& \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon &&&&&& \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon &&&&& \\
\times & \times & \times & \times & \times & \times &&&&& \times \\
\times & \times & \times & \times & \times & \times && \times &&& \times \\
\times & \times & \times & \times & \times & \times && \times & \times && \times \\
\times & \times & \times & \times & \times & \times && \times & \times & \times & \times
\end{pmatrix}
\rightarrow
\begin{pmatrix}
\times &&&&&&&&&& \\
\times & \times &&&&&&&&& \\
\times & \times & \times &&&&&&&& \\
\times & \times & \times & \times & \times & \times &&&&& \times \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon &&&&&&& \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon &&&&&& \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon &&&&& \\
\times & \times & \times & \times & \times & \times && \times &&& \times \\
\times & \times & \times & \times & \times & \times && \times & \times && \times \\
\times & \times & \times & \times & \times & \times && \times & \times & \times & \times
\end{pmatrix}
$$

$$
\rightarrow
\begin{pmatrix}
\times &&&&&&&&&& \\
\times & \times &&&&&&&&& \\
\times & \times & \times &&&&&&&& \\
\times & \times & \times & \times &&&&&&& \times \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon &&&&&& \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon &&&&& \\
\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon &&&&& \\
\times & \times & \times & \times & \times & \times && \times &&& \times \\
\times & \times & \times & \times & \times & \times && \times & \times && \times \\
\times & \times & \times & \times & \times & \times && \times & \times & \times & \times
\end{pmatrix}
$$

When these rotations are propagated into L from the left, they create fill which is annihilated again by means of right Givens rotations. The triangular structure of L_A and L is thus maintained, and L_A has small elements in rows $k+2$ through $p+1$.

The remaining steps are identical to the algorithm from [18]. The single "1" in the bottom row of the third factor is chased to the left by interleaved swaps of neighbor columns and Givens rotations applied to the rows to annihilate the fill. These transformations are propagated to the left and right, and they create fill in the second factor of A which, in turn, is annihilated with left Givens rotations. A final Givens rotation applied to rows 1 and $n+1$ annihilates the "1" in position $(n+1, 1)$, and a simple scaling restores the identity matrix I_{p+1}.

Having thus obtained a zero bottom row in the third factor we neglect the rightmost column of the first two factors. The result is a new factorization in which L and I have dimensions $(p+1) \times (p+1)$ and $(n-p-1) \times (n-p-1)$, respectively. Finally we reveal the rank of the updated $L_A(1\colon p+1, 1\colon p+1)$ by at most two rank-revealing steps. The complete algorithm is implemented in function ulliv_up_B.

If the rank of B is known to stay the same after the updating, then the algorithm should accommodate this fact. Ideally, the element ϕ in the reduced f_2^T should be zero, or very small. When this is the case, it is simple to annihilate f_1^T by means of strategic Givens rotations. Otherwise it is required to apply a ULV rank-revealing step to the $(p+1) \times (p+1)$ matrix

$$
\begin{pmatrix} L & 0 \\ f_1^T & \phi \end{pmatrix}.
$$

This step is guaranteed to produce small elements in the bottom row which can therefore safely be neglected. When applied to the full third factor, a single fill is created in position $(p+1, p)$, and we can use strategic Givens rotations to chase this element to the left. In both cases the resulting U_B has dimensions $(p+1) \times p$, and thus neither the updated B nor the updated U_B conform to our requirements of the ULLIV decomposition. We do not provide implementations.

5. A Lanczos Algorithm with Reorthogonalization and Restarts

We now describe our Lanczos-based routine lsvdrr for computing the largest p singular values σ_i and associated right singular vectors v_i of a matrix A. Our algorithm uses full reorthogonalization and explicit restarts, and it is based on the work in [6] with modifications that make it more reliable and possibly faster.

If we apply k steps of the Lanczos algorithm [8, §§9.1–2] to the matrix $A^T A$ (by first multiplying with A and then by A^T), cf. Fig. 5.1, then in exact arithmetic we produce an $n \times k$ matrix V_k with orthonormal columns, and a $k \times k$ symmetric semidefinite tridiagonal matrix T_k, such that $A\,V_k = V_k\,T_k$. Then it is well known that some of the large eigenvalues of T_k will approximate some of the large eigenvalues of $A^T A$. Since these eigenvalues are the squares of the singular values of A, we thus have a basic procedure for iteratively computing approximations to the large singular values of a matrix.

More precisely, let $T_k = S_k\,\Theta_k\,S_k^T$ denote the eigenvalue decomposition of T_k, and let $\theta_i^{(k)}$ denote these eigenvalues. Moreover, let $y_i^{(k)}$ denote the columns of the matrix $Y_k = V_k\,S_k$. Then $(\theta_i^{(k)}, y_i^{(k)})$ are called the Ritz pairs associated with the kth step of the Lanczos process, and some of the Ritz pairs will approximate some of the eigenpairs of $A^T A$.

Since this Lanczos algorithm is based on the implicit formation of the matrix $A^T A$, there is no guarantee that it can provide accurate estimates of the small singular values of A in finite-precision computations. This does not cause a problem here, however, because our algorithm is intended solely for the computation of the largest singular values.

A more severe difficulty with finite-precision computations in the Lanczos algorithm is that the Lanczos vectors (the columns of V_k) lose orthogonality as the Ritz values converge. This, in turn, leads to various difficulties with repeated and spurious eigenvalues of T_k that do not represent approximations to eigenvalues of $A^T A$. A number of sophisticated remedies have been proposed for overcoming these difficulties, many of them involving partial or selective reorthogonalizations, combined with methods for monitoring the accuracy of the Ritz values, cf. [8, §9.2]. With the inclusion of these techniques, the Lanczos algorithm can be used as a general-purpose method for computing, in principle, any portion of the eigenvalue spectrum of A.

Our goal here, on the other hand, is to provide a simple Lanczos algorithm solely for computing the largest p singular values of a matrix, suited as a basis for dedicated hardware implementations. For this reason, we use complete reorthogonalization among the Lanczos vectors (which takes place after step 5 in the getrtzp algorithm in Fig. 5.1). As long as p is not large, the additional computational work involved in this approach is acceptable, the actual code is very simple, and the storage requirements for the Lanczos vectors and the converged Ritz vectors are known a priori.

Our stopping criterion for the Lanczos process is based on an estimate of the error in $(\theta_i^{(k)})^{1/2}$, when considered an approximation to σ_j; different indices i and j are needed

Basic Lanczos Algorithm getrtzp:

1. Initialization: $\beta_0 \leftarrow 0$; $v_0 \leftarrow 0$; $v_1 \leftarrow$ initial vector.
2. For $k = 1, \ldots, \ell$
3. $\quad w \leftarrow A^T(A\, v_k) - \beta_{k-1}\, v_{k-1}$;
4. $\quad T_{k,k} \leftarrow \alpha_k \leftarrow v_k^T w$;
5. $\quad w \leftarrow \alpha_k\, v_k$;
6. $\quad T_{k,k+1} = T_{k+1,k} \leftarrow \beta_k \leftarrow \|w\|$;
7. $\quad v_{k+1} \leftarrow w/\beta_k$;
8. $T_k = S_k\, \Theta_k\, S_k^T$ (eigenvalue decomposition).
9. Use error estimates $e_i^{(k)}$ (5.1) to identify n_{c} converged Ritz pairs.

Lanczos SVD Alg. w/ Reorthogonalization and Restarts lsvdrr:

1. Initialization: $v_{\text{init}} \leftarrow A^T e$; $n_{\text{crp}} \leftarrow 0$; $\mathcal{RP} \leftarrow \emptyset$ (no Ritz pairs).
2. While $n_{\text{crp}} < k$
3. $\quad$ use getrtzp to compute n_{c} Ritz pairs;
4. $\quad \mathcal{RP} \leftarrow \mathcal{RP} \cup \{$set of new Ritz pairs$\}$;
5. $\quad n_{\text{crp}} \leftarrow n_{\text{crp}} + n_{\mathrm{c}}$;
6. $\quad v_{\text{init}} \leftarrow v_{k+1}$ from getrtzp.
7. For $i = 1, \ldots, \rho_0$
8. $\quad$ use getrtzp to compute n_{c} Ritz pairs;
9. $\quad$ if necessary, swap new Ritz pair(s) with pair(s) in $\mathcal{RP}$.

Figure 5.1: Top: the basic algorithm getrtzp for computing Ritz pairs of A^TA. Bottom: the complete Lanczos algorithm lsvdrr, in which $e = (1, \ldots, 1)^T$, $\mathcal{RP}$ is the set of converged Ritz pairs, and n_{crp} is the total number of converged Ritz pairs.

because there is no guarantee that the Ritz values converge in the "natural order." From Theorem 9.1.2 in [8], we know that the error $\sigma_j^2 - \theta_i^{(k)}$ in the ith Ritz value is bounded above as

$$|\sigma_j^2 - \theta_i^{(k)}| \le |\beta_k\, s_{ki}| \ , \qquad i = 1, \ldots, k,$$

where β_k is the bottom off-diagonal element of T_k, and s_{ki} is the ith element of the bottom row of S_k. If we write $(\theta_i^{(k)})^{1/2} = \tilde{\sigma}_j^{(k)} = \sigma_j + \delta_j^{(k)}$, then

$$|\sigma_j^2 - \theta_i^{(k)}| = |\sigma_j + \tilde{\sigma}_j^{(k)}|\, |\sigma_j - \tilde{\sigma}_j^{(k)}| = |2\,\tilde{\sigma}_j^{(k)} - \delta_j^{(k)}|\, |\delta_j^{(k)}| \approx 2\,\tilde{\sigma}_j^{(k)} |\delta_j^{(k)}|,$$

showing that the quantity

$$e_i^{(k)} = |\beta_k\, s_{ki}|\, (\theta_i^{(k)})^{-1/2} \ , \qquad i = 1, \ldots, k \tag{5.1}$$

provides an estimate of the error in $\tilde{\sigma}_i^{(k)} = (\theta_i^{(k)})^{1/2}$ considered as an approximation to a singular value σ_j of A. Our criterion for accepting a Ritz value as converged is therefore

$$e_i^{(k)} < \tau\, \sigma_{\max}, \tag{5.2}$$

where τ is a user-specified tolerance, and $\sigma_{\max}$ is an estimate of the largest singular value σ_1.

From Thm. 8.1.2 in [8] we also know that if $\tilde{v}_i^{(k)}$ is the approximate eigenvector associated with $\theta_i^{(k)}$ then

$$\|A^T A\, \tilde{v}_i^{(k)} - \theta_i^{(k)} \tilde{v}_i^{(k)}\| = |\beta_k\, x_{ki}| = e_i^{(k)}\, \tilde{\sigma}_i^{(k)},$$

showing that small error estimates $e_i^{(k)}$ guarantee small residuals. Furthermore, according to Thm. 11.7.2 in [19], small residuals imply a small subspace angle between the subspaces spanned by the exact and approximate eigenvectors.

Unfortunately, the number of Lanczos iterations needed to capture p singular values, within the accuracy estimates provided by (5.1), may exceed p by a large factor. The cure to this difficulty is to *restart* the Lanczos process with an initial vector that is orthogonal to the set of converged Ritz vectors. This is easily archived in our algorithm, where the Ritz vectors are explicitly saved.

Let n_{crp} denote the total number of converged Ritz pairs, and let ℓ_0 be a fixed number greater than p, chosen by the user. Each time the Lanczos process is (re)started, we perform ℓ iterations. In our algorithm, one can either choose $\ell = \ell_0$ or $\ell = \ell_0 - n_{\mathrm{crp}}$. The latter choice, which is default, ensures that a total of ℓ_0 Lanczos vectors are used.

When we have reached a stage where $n_{\mathrm{crp}} = p$ Ritz values $\theta_i^{(k)}$ have converged according to (5.2), there is no guarantee that we have computed approximations to the desired p largest singular values. Our heuristic remedy for this difficulty is to restart the Lanczos process additional ρ_0 times, where ρ_0 is a small number (the default is $\rho_0 = 2$). Experiments in [6] show that these additional restarts indeed improve the reliability of the algorithm, at little extra cost.

Upon completion our algorithm `lsvdrr` returns approximations $\tilde{\sigma}_i$ and $\tilde{v}_i$ to the largest p singular values and corresponding right singular vectors. Approximations to the left singular vectors can then be computed as $\tilde{u}_i = A\tilde{v}_i/\tilde{\sigma}_i$, and we emphasize that these vectors are not orthonormal. If an SVD routine is available, one can instead compute a diagonal matrix $\widehat{\Sigma}$ and two matrices $\widehat{U}$ and $\widehat{V}$ with orthonormal columns satisfying $A\,\widehat{V} = \widehat{U}\,\widehat{\Sigma}$, by means of the following procedure:

1. $A\,(\tilde{v}_1, \ldots, \tilde{v}_p) = \widehat{U}\,\widehat{\Sigma}\,\widehat{V}^T$ (SVD computation),
2. $\widehat{V} \leftarrow (\tilde{v}_1, \ldots, \tilde{v}_p)\,\widehat{V}$.

More details about this approach can be found in [6], which also includes numerical results concerning the accuracy and efficiency of the lsvdrr algorithm.

6. Numerical Examples

We conclude with three demonstrations of the use of our package. The first example is available in the script vsviddemo, and shows how the three routines hvsvid_L, hvsvid_R and lvsvid can be used to compute symmetric indefinite VSV decompositions (2.1) of rank deficient KKT matrices of the form

$$A = \begin{pmatrix} M & N^T \\ N & 0 \end{pmatrix}.$$

In our test problems, M is an $m \times m$ symmetric semidefinite and rank-deficient matrix, and $N = \Theta\, M$ with Θ a $q \times m$ random matrix. The resulting matrix A has dimension $n = m + q$, and it is rank deficient and symmetric indefinite.

We generate 500 test matrices, and for each matrix and each VSV decomposition we compute the numerical rank r, the backward error $\|A - V\,T^T\,\Omega\,T\,V^T\|$, and the subspace angle between the numerical null space (spanned by the last $n - r$ singular vectors of A) and the approximate null space spanned by the last $n - r$ columns of V. A typical example of the results from such a test is shown in Fig. 6.1 for $m = 12$ and $q = 2$. Occasionally, the errors are in the range 10^{-13}–10^{-9}, while most of the errors are less than 10^{-13}.

The second test problem is available in the function ullivdemo and illustrates the use of the ULLIV decomposition is noise and interference reduction. We generate a clean signal $s \in \mathbb{R}^N$ of length $N = 350$ consisting of a sum of 9 sinusoids with unit amplitude; see the DFT spectrum in the top of Fig. 6.2. The noisy signal is generated by adding white noise and an interfering signal to the clean signal; the white noise is generated by the Matlab command 0.5*randn(N,1), and the interfering signal is a sum of 16 sinusoids with amplitude 0.2. The DFT spectrum of the noisy signal is shown in the middle of Fig.6.2.

The filtered signal is then computed by means of the subspace method described in [12]. This involves the computation of the ULLIV decomposition (4.1) of two Hankel matrices A and B, the first being 311×40 and consisting of the noisy signal, and the second being 32×40 and representing the signal subspace of the interfering signal. From the ULLIV decomposition and the numerical rank p we then construct the matrix

$$X = U_A\,\Psi\,L_A \begin{pmatrix} L & 0 \\ 0 & I \end{pmatrix} V^T, \qquad \Psi = \mathrm{diag}(I_p, 0).$$

Finally we construct the filtered signal by averaging along the antidiagonals of X. The DFT spectrum of the filtered signal is shown in the bottom of Fig.6.2, and we see that we have indeed reduced the interference while maintaining most of the clean signal.

In the third test problem, which is available in the script lsvdrrdemo, we compute the k largest singular values and right singular vectors of a complex Toeplitz matrix of size 513×512 constructed from an NMR signal available as Data Set 002 at the BioSource database [1] of MRS signals. See, e.g., [22] for an application of such computations. The computations are performed for $k = 5$, 10, 15 and 20, and for each k we compare the errors in the singular values and vectors, as well as the computing times, with those of the Matlab function svds.

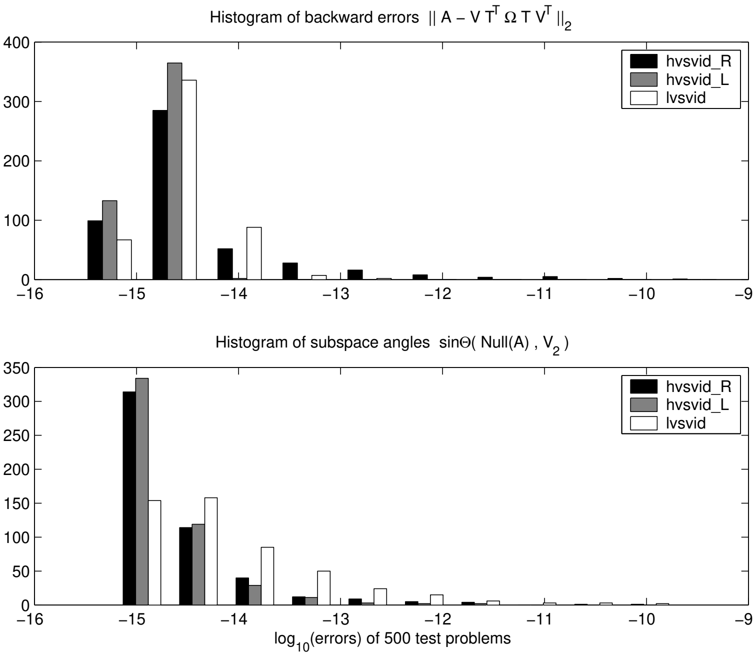

Figure 6.1: Backward errors and subspace angles for 500 KKT test problems of dimension 14×14.

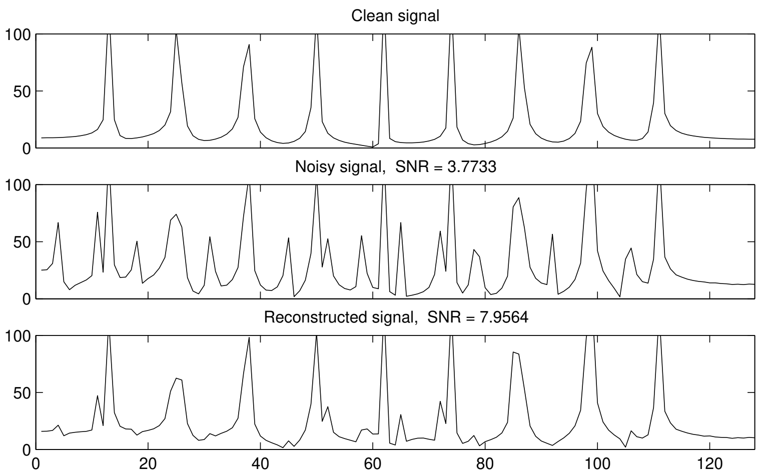

Figure 6.2: DFT spectra of the clean, noisy and filtered signals in ullivdemo test problem.

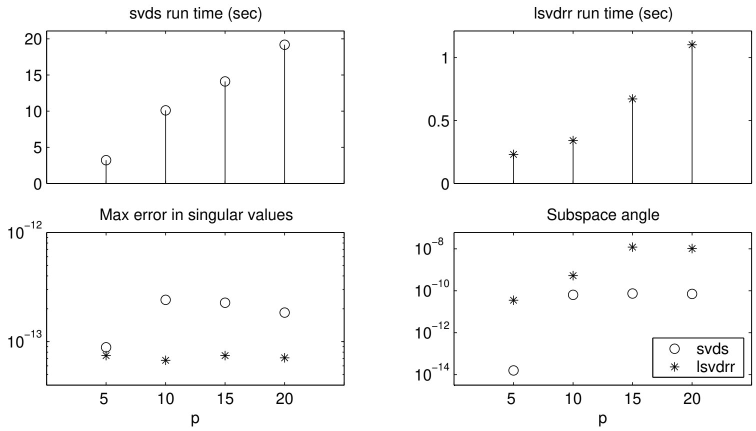

Figure 6.3: Computing times and errors for the largest k singular values and right singular vectors, computed by means of our Lanczos algorithm lsvdrr as well as Matlab's Lanczos algorithm implemented in the svds function. The bottom left figure reports $\max|\sigma_i - \tilde{\sigma}_i|$, $i = 1, \ldots, p$, while the bottom right figure reports the subspace angle between the subspaces spanned by the dominant p right singular vectors and the approximations $\tilde{v}_i, \ldots, \tilde{v}_p$.

We used the tolerance $\tau = 10^{-10}$ in the convergence criterion (5.2), and the matrix-vector multiplications with the Toeplitz matrix are done via Matlab's fft function. Figure 6.3 shows that for this test problem (using a 1.4 GHz Pentium), lsvdrr computes accurate singular values and right singular vectors faster than the general-purpose Lanczos routine svds. For comparison, Matlab's dense SVD routine computed all the singular values and right singular vectors in 8.5 seconds.

Bibliography

[1] BioSource database, data set 002 is available at:
http://www.esat.kuleuven.ac.be/sista/members/biomed/data002.htm.

[2] A. Bojanczyk, S. Qiao and A. O. Steinhardt, *Unifying unitary and hyperbolic transformations*, Lin. Alg. Appl, 316 (2000), pp. 183–197.

[3] R. D. Fierro and P. C. Hansen, *Low-rank revealing UTV decompositions*, Numerical Algorithms, 15 (1997), pp. 37–55.

[4] R. D. Fierro and P. C. Hansen, *Truncated VSV solutions to symmetric rank-deficient problems*, BIT, 42 (2002), pp. 531–540.

[5] R. D. Fierro, P. C. Hansen and P. S. K. Hansen, *UTV Tools: Matlab templates for rank-revealing UTV decompositions*, Numer. Algo., 20 (1999), pp. 165–194.

[6] R. D. Fierro and E. P. Jiang, *Lanczos and the Riemannian SVD in information retrieval applications,* Numer. Lin. Alg. Appl., to appear.

[7] G. H. Golub and C. Greif, *On solving block-structured indefinite linear systems*, SIAM J. Sci. Comput., 24 (2003), pp. 2076–2092.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2. Ed., Johns Hopkins University Press, Baltimore, 1996.

[9] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, 1998.

[10] P. C. Hansen, *Rank-deficient prewhitening with quotient SVD and ULV decompositions*, BIT, 38 (1998), pp. 34–43.

[11] P. C. Hansen, *Regularization Tools Version 3.0 for Matlab 5.2*, Numer. Algorithms, 20 (1999), pp. 195–196. Software and manual is available from Netlib in directory numeralgo/na4.

[12] P. C. Hansen and S. H. Jensen, *Prewhitening for narrow-band noise in subspace methods for noise reduction*, Technical Report IMM-TR-2004-5, IMM, 2004 (8 pages); submitted to IEEE Trans. Signal Proc.

[13] P. C. Hansen and P. Y. Yalamov, *Computing symmetric rank-revealing decompositions via triangular factorization*, SIAM J. Matrix Anal. Appl., 28 (2001), pp. 443–458.

[14] N. J. Higham, *The Matrix Computation Toolbox for MATLAB (Version 1.0)*, Numerical Analysis Report No. 410, Department of Mathematics, University of Manchester, 2002. `http://www.ma.man.ac.uk/~higham/mctoolbox`.

[15] D. A. Huckaby and T. F. Chan, *Stewart's pivoted QLP decomposition for low-rank matrices*, Tech. Report CAM-02-54, Dept. of Mathematics, UCLA, 2002.

[16] F. T. Luk and S, Qiao, *A symmetric rank-revealing Toeplitz matrix decomposition*, J. VLSI Signal Proc., 14 (1996), pp. 19–28.

[17] F. T. Luk and S. Qiao, *A new matrix decomposition for signal processing*, Automatica, 30 (1994), pp. 39–43.

[18] F. T. Luk and S. Qiao, *An adaptive algoithm for interference cancelling in array processing;* in F. T. Luk (Ed.), *Advanced Signal Processing Algorithms, Architectures, and Implementations VI*, SPIE Proceedings, Vol. 2846 (1996), pp. 151–161.

[19] B. N.. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.

[20] G. W. Stewart, *An updating algorithm for subspace tracking*, IEEE Trans. Signal Proc., 40 (1992), pp. 1535–1541.

[21] G. W. Stewart, *The QLP approximation to the singular value decomposition*, SIAM J. Sci. Comp., 20 (1999), pp. 1336–1348.

[22] L. Vanhamme, R. D. Fierro, S. Van Huffel and R. de Beer, *Fast removal of residual water in proton spectra*, J. Magnetic Resonance, 132 (1998), pp. 197–203.

7. Overview of Routines

Demo Functions and Solvers	
lsvdrrdemo	Demonstrates the use of the restarted Lanczos algorithm implemented in lsvdrr and applied to NMR data from [1]
tvsv	Solves a symmetric num. rank-deficient system of equations
ullivdemo	Demonstrates the use of the high-rank ULLIV algorithm ulliv, applied to noise and interference reduction
vsviddemo	Demonstrates the use of the indefinite VSV algorithms hvsvid_L, hvsvid_R and lvsvid applied to symmetric indefinite KKT systems

Rank-Revealing Symmetric VSV Algorithms	
hvsvid_L	High-rank algorithm for indefinite matrix, L version
hvsvid_R	High-rank algorithm for indefinite matrix, R version
hvsvsd	High-rank algorithm for semidefinite matrix
lvsvid	Low-rank algorithm for indefinite matrix
lvsvsd	Low-rank algorithm for semidefinite matrix
vsv_qrit	Block QR refinement of VSV decomposition

VSV Up- and Downdating	
vsvid_L_mod	Rank-one modification of indefinite matrix, L version
vsvid_R_mod	Rank-one modification of indefinite matrix, R version
vsvsd_up	Rank-one update of semidefinite matrix

QLP Algorithms	
hqlp	High-rank gap-revealing QLP factorization
lqlp	QLP matrix approximation of a low-rank matrix

ULLIV Algorithms for a Matrix Pair (A, B)	
ulliv	Rank-revealing ULLIV decomposition, B has full row rank
ulliv_up_A	Rank-one update of the A matrix
ulliv_up_B	Rank-one update of the B matrix (rank increases)

Lanczos Algorithm with Reorthogonalization and Restarts	
getrtzp	Computation of Ritz pairs
lsvdrr	Driver routine for Lanczos algorithm
tprod	Toeplitz matrix-vector multiplication using FFT
tprodinit	Initialization routine for tprod

Misc. Tools	
agl5	Ashcraft-Grimes-Lewis 5×5 test matrix for LDL^T factorization
app_hyp	Apply a stabilized hyperbolic rotation
app_qrot	Apply a quadratic rotation
csi-10-10	Mat-file with NMR data for lsvdrrdemo
gen_hyp	Generate a stabilized hyperbolic rotation
gen_qrot	Generate a quadratic rotation
hvsvid_cdef	Column deflation of upper triang. matrix in indef. VSV decomp.
hvsvid_rdef	Row deflation of lower triang. matrix in indef. VSV decomp.
lvsvid_cdef	Column deflation of lower triang. matrix in indef. VSV decomp.
TOTinviter	Inverse iterations applied to $T^T \Omega T$
TOTlanczos	Lanczos method applied to $T^T \Omega T$
TOTpowiter	Power iterations applied to $T^T \Omega T$
vsvid_ip	Interim processor for indefinite VSV decomposition

Two routines from the Matrix Computation Toolbox [14]	
cholp	Pivoted Cholesky factorization
ldlt_symm	LDL^T factorization with symmetric or rook pivoting

agl5

Purpose

Ashcraft-Grimes-Lewis 5-times-5 test problem for LDL^T factorization

Synopsis

[A,L,D] = agl5

Description

Generates a 5-times-5 test problem for the LDL^T factorization, such that L has a large entry when partial pivoting (Bunch-Kaufman) is used.

References

[1] C. Ashcraft, R.G. Grimes and J.G. Lewis, "Accurate symmetric indefinite linear equation solvers," SIAM J. Matrix Anal. Appl., 20 (1999), pp. 513-561.

app_hyp

Purpose

Apply a stabilized hyperbolic rotation (left/right).

Synopsis

[u1,u2] = app_hyp(v1,v2,c,s,sgn)

Description

Apply a stabilized hyperbolic rotation, defined by the parameters c and s, from the left to the row vectors v1 and v2 such that

```
[u1] = [ ch -sh ] [v1]
[u2]   [-sh  ch ] [v2]
```

or from the right to the column vectors v1 and v2 such that

```
[u1 u2] = [v1 v2] [ ch -sh ]
                  [-sh  ch ]
```

where ch = 1/s and sh = c/s.

See Also

gen_hyp – Determine a 2-by-2 stabilized hyperbolic rotation.

References

[1] S.T. Alexander, C.-T. Pan & R.J. Plemmons, "Analysis of recursive least squares hyperbolic rotation algorithms for signal processing," Lin. Alg. Appl. 98 (1998), 3-40.

app_qrot

Purpose

Apply a quadratic rotation (left/right).

Synopsis

[u1,u2,dd1,dd2] = app_qrot(v1,v2,c,s,d1,d2,sgn)

Description

Apply a quadratic rotation H, defined by the parameters c and s, from the left to the row vectors v1 and v2 such that

```
[u1] = H [v1]
[u2]     [v2]
```

or from the right to the column vectors v1 and v2 such that

```
[u1 u2] = [u1 u2] H'
```

Also update the signature matrix:

```
[dd1  0 ] = sgn*[d1  0 ]
[ 0  dd2]       [ 0  d2]
```

where sgn is determined by the rotation.

See Also

gen_qrot – Determine a 2-by-2 quadratic rotation.

gen_hyp

Purpose

Determine a 2-by-2 stabilized hyperbolic rotation matrix.

Synopsis

[c,s,r,sgn] = gen_hyp(a,b)

Description

Compute a stabilized hyperbolic rotation to annihilate b using a, i.e., compute parameters c, s, and r such that

```
[ ch -sh ] [a] = [r]   with   [ ch -sh ] S [ ch -sh ] = sgn*S
[-sh  ch ] [b]   [0]          [-sh  ch ]   [-sh  ch ]
```

where ch = 1/s and sh = c/s, and where the signature matrix S is either

```
S = [-1  0 ]   or   S = [ 1  0 ] .
    [ 0  1 ]            [ 0 -1 ]
```

See Also

app_hyp – Apply a stabilized hyperbolic rotation.

References

[1] S.T. Alexander, C.-T. Pan & R.J. Plemmons, "Analysis of recursive least squares hyperbolic rotation algorithms for signal processing," Lin. Alg. Appl. 98 (1998), 3-40.

gen_qrot

Purpose

Determine a 2-by-2 quadratic rotation matrix.

Synopsis

[c,s,r,sgn] = gen_qrot(a,b,d1,d2)

Description

Compute a real quadratic (Givens or hyperbolic) rotation H to annihilate b using a, i.e., compute c, s, and r such that

```
H [a] = [r]   with   H' S H = sgn*S
  [b]   [0]
```

where the signature matrix S is

```
S = [ d1  0 ] ,   d1,d2 = +1,-1 .
    [ 0  d2 ]
```

See Also

app_qrot – Apply a quadratic rotation.

getrtzp

Purpose

Compute (additional) Ritz pairs for a cross-product matrix

Synopsis

```
[rtzvals,rtzvecs,errests,nconv,num_iter,vnext,smax] = ...
getrtzp(A,k,ncrp,Vk,max_iter,vinit,tol,smax)
```

Description

Applies Lanczos iterations to the matrix A'∗A; the Lanczos vectors are explicitly re-orthogonalized internally, as well as with respect to an existing set of converged vectors. A singular value estimate is considered as converged when an error estimate is below smax∗tol.

Input Parameters

A	matrix;
k	number of desired singular values;
ncrp	number of converged Ritz pairs so far;
Vk	previously converged Ritz vectors;
max_iter	max. no. of Lanczos iterations in this call to getrtzp;
vinit	start vector;
tol	relative residual tolerance;
smax	current estimate of largest singular value;

Output Parameters

rtzvals	converged Ritz values og A'∗A;
rtzvecs	corresponding converged Ritz vectors;
errests	corresponding error estimates of singular values;
nconv	number of converged Ritz pairs;
num_iter	number of iterations used in the call to getrtzp;
vnext	vector for next call to getrtzp;
smax	updated value of smax;

References

[1] R.D. Fierro and E.P. Jiang, "Lanczos and the Riemannian SVD in information retrieval applications," Numer. Lin. Alg. Appl., to appear.

hqlp

Purpose

High-rank gap-revealing QLP factorization.

Synopsis

[p,L,P,Q] = hqlp(A)
[p,L,P,Q] = hqlp(A,gap_tol)

Description

Computes a gap-revealing factorization A = Q*L*P', in which Q and P are orthogonal matrices, and L is a lower triangular matrix whose diagonal elements sometimes approximate the singular values of A. Also returns the largest p such that abs(L(p,p)/L(p+1,p+1)) > gap_tol. Designed for high-rank matrices; use lqlp for low-rank matrices.

Input Parameters

A	general matrix;
gap_tol	tolerance for gap detection;
Defaults	gap_tol = min(size(A))/eps;

Output Parameters

p	estimate of numerical rank of A;
L	lower triangular matrix in A = Q*L*P';
P	right orthogonal matrix;
Q	left orthogonal matrix;

Algorithm

A pivoted QR factorization A*Pi_p = Q*R is followed by a pivoted QR factorization R'*Pi_q = P*L'; thus A = (Q*Pi_q)*L*(Pi_p*P)'. The diagonal elements of L sometimes track the singular values of A, but this is not guaranteed; hence the factorization cannot be guaranteed to reveal rank.

See Also

lqlp – pivoted QLP matrix approximation with interleaved factorizations.

References

[1] G.W. Stewart, "The QLP approximation to the singular value decomposition," SIAM J. Sci. Comp., 20 (1999), pp. 1336-1348.

hvsvid_L

Purpose

High-rank revealing decomp. of a sym. indef. matrix, L version.

Synopsis

```
[p,L,Omega,V] = hvsvid_L(A)
[p,L,Omega,V] = hvsvid_L(A,tol_rank)
[p,L,Omega,V] = hvsvid_L(A,tol_rank,max_iter)
[p,L,Omega,V] = hvsvid_L(A,tol_rank,max_iter,fixed_rank)
```

Description

Computes a rank-revealing VSV decompostion A = V∗(L'∗Omega∗L)∗V' of a symmetric indefinite n-by-n matrix. Only the upper triangular part needs to be specified. Optimized for matrices whose rank p close to n. Function hvsvid_R computes the R version.

Input Parameters

A	symmetric indefinite matrix;
tol_rank	rank decision tolerance;
max_iter	max. number of inverse iterations per deflation step, used in the singular vector estimator;
fixed_rank	deflate to the fixed rank given by fixed_rank instead of using the rank decision tolerance;
Defaults	tol_rank = n∗norm(A,1)∗eps; max_iter = 5;

Output Parameters

p	numerical rank of A;
L	lower triangular matrix in A = V∗(L'∗Omega∗L)∗V';
Omega	signature matrix in A = V∗(L'∗Omega∗L)∗V';
V	orthogonal matrix in A = V∗(L'∗Omega∗L)∗V';

Algorithm

The symmetric indefinite matrix A is preprocessed by a pivoted LDL' factorization. An interim stage (where D is made diagonal) is followed by a rank-revealing ULV-like decomposition, using inverse iterations for singular vector estimation.

See Also

hvsvid_R	–	High-rank revealing VSV alg. for sym. indef. matrices, R version
hvsvsd	–	High-rank revealing VSV alg. for symmetric semidefinite matrices
lvsvid	–	Low-rank revealing VSV alg. for symmetric indefinite matrices
lvsvsd	–	Low-rank revealing VSV alg. for symmetric semidefinite matrices

References

[1] P.C. Hansen & P.Y. Yalamov, "Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization", SIAM J. Matrix Anal. Appl., 23 (2001), pp. 443-458.

hvsvid_R

Purpose

High-rank revealing decomp. of a sym. indef. matrix, R version.

Synopsis

```
[p,R,Omega,V] = hvsvid_R(A)
[p,R,Omega,V] = hvsvid_R(A,tol_rank)
[p,R,Omega,V] = hvsvid_R(A,tol_rank,max_iter)
[p,R,Omega,V] = hvsvid_R(A,tol_rank,max_iter,fixed_rank)
```

Description

Computes a rank-revealing VSV decompostion A = V∗(R'∗Omega∗R)∗V' of a symmetric indefinite n-by-n matrix. Only the upper triangular part needs to be specified. Optimized for matrices whose rank p is close to n. Functions hvsvid and lvsvid compute the L-version of the decomposition.

Input Parameters

A	symmetric indefinite matrix;
tol_rank	rank decision tolerance;
max_iter	max. number of inverse iterations per deflation step, used in the singular vector estimator;
fixed_rank	deflate to the fixed rank given by fixed_rank instead of using the rank decision tolerance;
Defaults	tol_rank = n∗norm(A,1)∗eps; max_iter = 5;

Output Parameters

p	numerical rank of A;
R	upper triangular matrix in A = V∗(R'∗Omega∗R)∗V';
Omega	signature matrix in A = V∗(R'∗Omega∗R)∗V';
V	orthogonal matrix in A = V∗(R'∗Omega∗R)∗V';

Algorithm

The symmetric indefinite matrix A is preprocessed by a pivoted LDL' factorization. An interim stage (where D is made diagonal) is followed by a rank-revealing URV-like decomposition, using inverse iterations for singular vector estimation.

See Also

hvsvid_L	–	High-rank revealing VSV alg. for sym. indef. matrices, L version
hvsvsd	–	High-rank revealing VSV alg. for symmetric semidefinite matrices
lvsvid	–	Low-rank revealing VSV alg. for symmetric indefinite matrices
lvsvsd	–	Low-rank revealing VSV alg. for symmetric semidefinite matrices

References

[1] P.C. Hansen & P.Y. Yalamov, "Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization", SIAM J. Matrix Anal. Appl., 23 (2001), pp. 443-458.

hvsvid_cdef

Purpose

Deflate one column of R in the high-rank URV-like VSV algorithm.

Synopsis

[R,Omega,V,fail] = hvsvid_cdef(R,Omega,V,r,vmin)

Description

Given the decomposition V*R'*Omega*R*V', the function deflates the last column of R(1:r,1:r). vmin is an estimate of the right singular vector of R(1:r,1:r)'*Omega(1:r,1:r)*R(1:r,1:r) associated with the smallest singular value sigma_r. On return, the norm of the last column of R(1:r,1:r)'*Omega(1:r,1:r)*R(1:r,1:r) is of the order sigma_r. The matrix V can be left out by inserting an empty matrix [].

Input Parameters

R	upper triangular matrix in A = V*(R'*Omega*R)*V';
Omega	signature matrix in A = V*(R'*Omega*R)*V';
V	orthogonal matrix in A = V*(R'*Omega*R)*V';
r	size of submatrix to be deflated;
vmin	estimate of the smallest right singular vector of the product R(1:r,1:r)'*Omega(1:r,1:r)*R(1:r,1:r);

Output Parameters

R	upper triangular matrix in resulting A = V*(R'*Omega*R)*V';
Omega	signature matrix in resulting A = V*(R'*Omega*R)*V';
V	orthogonal matrix in resulting A = V*(R'*Omega*R)*V';
fail	true if a hypernormal rotation is ill defined;

References

[1] P.C. Hansen & P.Y. Yalamov, "Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization", SIAM J. Matrix Anal. Appl., 23 (2001), pp. 443-458.

hvsvid_rdef

Purpose

Deflate one row of L in the high-rank ULV-like VSV algorithm.

Synopsis

[L,V,Omega] = hvsvid_rdef(L,V,Omega,r,vmin)

Description

Given the decomposition V∗L'∗Omega∗L∗V', the function deflates the last row of L(1:r,1:r). vmin is an estimate of the right singular vector of L(1:r,1:r)'∗Omega(1:r,1:r)∗L(1:r,1:r) associated with the smallest singular value sigma_r. On return, the norm of the last column of L(1:r,1:r)'∗Omega(1:r,1:r)∗L(1:r,1:r) is of the order sigma_r. The matrix V can be left out by inserting an empty matrix [].

Input Parameters

L	lower triangular matrix in A = V∗(L'∗Omega∗L)∗V';
Omega	signature matrix in A = V∗(L'∗Omega∗L)∗V';
V	orthogonal matrix in A = V∗(L'∗Omega∗L)∗V';
r	size of submatrix to be deflated;
vmin	estimate of the smallest right singular vector of the product L(1:r,1:r)'∗Omega(1:r,1:r)∗L(1:r,1:r);

Output Parameters

L	upper triangular matrix in resulting V∗(L'∗Omega∗L)∗V';
V	orthogonal matrix in resulting V∗(L'∗Omega∗L)∗V';
Omega	signature matrix in resulting V∗(L'∗Omega∗L)∗V';

References

[1] P.C. Hansen & P.Y. Yalamov, "Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization", SIAM J. Matrix Anal. Appl., 23 (2001), pp. 443-458.

hvsvsd

Purpose

High-rank revealing decompostion of a symmetric semidefinite matrix.

Synopsis

```
[p,L,V] = hvsvsd(A)
[p,L,V] = hvsvsd(A,tol_rank)
[p,L,V] = hvsvsd(A,tol_rank,fixed_rank)
```

Description

Computes a rank-revealing VSV decomposition A = V*(L'*L)*V' of a symmetric semidefinite n-by-n matrix. Only the upper triangular part needs to be specified. Optimized for matrices whose rank p is close to n.

Input Parameters

A	symmetric semidefinite matrix;
tol_rank	rank decision tolerance;
fixed_rank	deflate to the fixed rank given by fixed_rank instead of using the rank decision tolerance;
Defaults	tol_rank = n*norm(A,1)*eps;

Output Parameters

p	numerical rank of A;
L	lower triangular matrix in A = V*(L'*L)*V';
V	orthogonal matrix in A = V*(L'*L)*V';

Algorithm

The symmetric semidefinite matrix A is preprocessed by a pivoted Cholesky factorization and then postprocessed by a high-rank-revealing ULV decomposition. An indefinite matrix results in an error message during the Cholesky factorization.

See Also

hvsvid_L	–	High-rank revealing VSV alg. for sym. indef. matrices, L version
hvsvid_R	–	High-rank revealing VSV alg. for sym. indef. matrices, R version
lvsvid	–	Low-rank revealing VSV alg. for symmetric indefinite matrices
lvsvsd	–	Low-rank revealing VSV alg. for symmetric semidefinite matrices

References

[1] P.C. Hansen & P.Y. Yalamov, "Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization", SIAM J. Matrix Anal. Appl., 23 (2001), pp. 443–458.

lqlp

Purpose

Pivoted QLP matrix approximation with interleaved factorizations.

Synopsis

```
[p,L,P,Q,gap_ratio] = lqlp(A)
[p,L,P,Q,gap_ratio] = lqlp(A,tol_gap)
[p,L,P,Q,gap_ratio] = lqlp(A,tol_gap,fixed_rank)
```

Description

Computes a rank-p pivoted QLP matrix approximation of an m-by-n matrix A ($m \geq n$) satisfying $A*P = Q*L$ with a lower triangular p-by-p matrix L. The rank (or stopping point) p is either fixed_rank or is dynamically determined by tol_gap. The absolute value of the diagonal elements of L are approximations to the first p singular values of A, while the columns of Q and P approximate the first p left and right singular vectors of A.

Input Parameters

A	m-by-n matrix ($m \geq n$);
tol_gap	truncate the decomposition after compuing a rank-p approximation to A, where p is the smallest integer such that abs(L(p,p)/L(p+1,p+1)) $\geq$ tol_gap;
fixed_rank	ignore tol_gap and truncate the decomposition after computing an approximation to A of rank fixed_rank.
Defaults	tol_gap = n/eps; fixed_rank = n.

Output Parameters

p	smallest integer such that abs(L(p,p)/L(p+1,p+1)) ¿ tol_gap (or fixed_rank);
L	p-by-p lower triangular matrix whose diagonal elements, in absolute value, track the largest p singular values of A;
P, Q	matrices with p orthonormal columns;
gap_ratio	abs(L(p+1,p+1)/L(p,p)), that is, the ratio of the first approximate singular value excluded to the last one included, empty if p = n or fixed_rank = n.

Algorithm

The first p rows and columns of the pivoted QR factorization of A are computed, A*Pi_1 = Q*R. Then the pivoted QR factorization of R' is computed, R'*Pi_2 = P*L', where L' is p-by-p. The rank p is either fixed_rank or is determined dynamically by the following. The computation of R is stopped after k1 rows and columns, where abs(R(k1,k1)/R(k1-1,k1-1)) $\leq$ tol_gap. Then rows and columns of L' are computed to see whether abs(L(j,j)/L(j-1,j-1)) $\leq$ tol_gap for any j $\leq$ k1. If so, the algorithm halts after computing these j rows and columns of L, and the final approximation to the SVD of A is rank j. If not, the next k2 rows and columns of R are computed until tol_gap is achieved, then corresponding rows and columns of L (k1 + j, $1 \leq j \leq$ k2) are computed to see whether tol_gap holds at any j and the computation can be halted. If not, more rows and columns of R are computed, etc. This process is called interleaving.

See Also

hqlp	–	high-rank gap-revealing QLP factorization.

References

[1] G.W. Stewart, "The QLP approximation to the singular value decomposition," SIAM J. Sci. Comp., 20 (1999), pp. 1336-1348.

[2] D.A. Huckaby and T.F. Chan, "Stewart's pivoted QLP decomposition for low-rank matrices," Tech. Report CAM-02-54, Dept. Mathematics, UCLA, 2002.

lsvdrr

Purpose

Lanczos SVD with reorthogonalization and explicit restarts

Synopsis

[sk,Vk,nits,nrst,errests] = lsvdrr(A,k,max_iter,max_restarts,conserve,safety)

Description

Computes approximations to the k dominant singular triplets, using the Lanczos method with reorthogonalization and explicit restarts. The stopping criterion is that the error in each computed singular value must be smaller than tol times the largest singular value.

Input Parameters

A	dense/sparse matrix or structure with Toeplitz matrix;
k	number of desired singular triplets;
tol	tolerance for relative residual of triplets;
max_iter	maximum number of Lanczos iterations per restart;
max_restarts	maximum number of Lanczos restarts;
conserve	if 1 then max_iter Lanczos vectors are used in total, otherwise max_iter vectors are used in each restart;
safety	perform additional safety restarts, after k Ritz values have converged;

Defaults	tol = 1e-4;
	max_iter = min(2∗k,n)
	max_restarts = 100;
	conserve = 1;
	safety = 2;

Output Parameters

sk	vector of singular value estimates;
Vk	matrix of right singular vector approximations;
nits	number of times A and A' combined have been referenced;
nrst	number of restarts;
errests	vector of error estimates for singular values;

Matrix Representation

The input parameter A can be either a matrix (dense or sparse) or a structure that holds information about a Toeplitz matrix: A.col = first column of A, A.row = first row of A.

References

[1] R.D. Fierro and E.P. Jiang, "Lanczos and the Riemannian SVD in information retrieval applications," Numer. Lin. Alg. Appl., to appear.

lvsvid

Purpose

Low-rank revealing decompostion of a symmectric indefinite matrix.

Synopsis

```
[p,L,Omega,V] = lvsvid(A)
[p,L,Omega,V] = lvsvid(A,tol_rank)
[p,L,Omega,V] = lvsvid(A,tol_rank,max_iter)
[p,L,Omega,V] = lvsvid(A,tol_rank,max_iter,est_type)
[p,L,Omega,V] = lvsvid(A,tol_rank,max_iter,est_type,fixed_rank)
```

Description

Computes a rank-revealing VSV decompostion A = V∗(R'∗Omega∗R)∗V' of a symmetric indefinite n-by-n matrix. Only the upper triangular part needs to be specified. Optimized for matrices whose rank p is small compared to n.

Input Parameters

A	symmetric indefinite matrix;
tol_rank	rank decision tolerance;
max_iter	max. number of power/Lanczos iterations per deflation step, used in the singular vector estimator;
est_type	if true, then estimate singular vectors by means of the Lanczos procedure, else use the power method;
fixed_rank	deflate to the fixed rank given by fixed_rank instead of using the rank decision tolerance;

Defaults	tol_rank = n∗norm(A,1)∗eps;
	max_iter = 5;
	est_type = 0 (power method);

Output Parameters

p	numerical rank of A;
L	lower triangular matrix in A = V∗(L'∗Omega∗L)∗V';
Omega	signature matrix in A = V∗(L'∗Omega∗L)∗V';
V	orthogonal matrix in A = V∗(L'∗Omega∗L)∗V';

See Also

hvsvid	–	High-rank revealing VSV alg. for symmetric indefinite matrices
hvsvid_R	–	High-rank revealing VSV alg., sym. indef. matrices, R version
hvsvsd	–	High-rank revealing VSV alg. for symmetric semidefinite matrices
lvsvsd	–	Low-rank revealing VSV alg. for symmetric semidefinite matrices

References

[1] P.C. Hansen & P.Y. Yalamov, "Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization", SIAM J. Matrix Anal. Appl., 23 (2001), pp. 443–458.

lvsvid_cdef

Purpose

Deflate one column of L in the low-rank ULV-like VSV algorithm.

Synopsis

[L,Omega,V] = lvsvid_cdef(L,Omega,V,r,vmax)

Description

Given the VSV decomposition V∗L'∗Omega∗L∗V', the function deflates the first column of L(r:n,r:n). vmax is an estimate of the right singular vector of L(r:n,r:n)'∗Omega(r:n,r:n)∗L(r:n,r:n) associated with the largest singular value sigma_1. On return, the norm of the first column of L(r:n,r:n)'∗Omega(r:n,r:n)∗L(r:n,r:n) is of the order sigma_1. The matrix V can be left out by inserting an empty matrix [].

Input Parameters

L	lower triangular matrix in A = V∗(L'∗Omega∗L)∗V';
Omega	signature matrix in A = V∗(L'∗Omega∗L)∗V';
V	orthogonal matrix in A = V∗(L'∗Omega∗L)∗V';
r	size of submatrix to be deflated;
vmin	estimate of the smallest right singular vector of the product L(1:r,1:r)'∗Omega(1:r,1:r)∗L(1:r,1:r);

Output Parameters

L	lower triangular matrix in resulting A = V∗(L'∗Omega∗L)∗V';
Omega	signature matrix in resulting A = V∗(L'∗Omega∗L)∗V';
V	orthogonal matrix in resulting A = V∗(L'∗Omega∗L)∗V';
V	rthogonal matrix in resulting A = V∗(L'∗Omega∗L)∗V';

References

[1] R.D. Fierro and P.C. Hansen, "Low-Rank Revealing UTV Decompositions", Numerical Algorithms, 15 (1997), pp. 37–55.

lvsvsd

Purpose

Low-rank revealing decompostion of a symmetric semidefinite matrix.

Synopsis

```
[p,L,V] = lvsvsd(A)
[p,L,V] = lvsvsd(A,tol_rank)
[p,L,V] = lvsvsd(A,tol_rank,max_iter)
[p,L,V] = lvsvsd(A,tol_rank,max_iter,est_type)
[p,L,V] = lvsvsd(A,tol_rank,max_iter,est_type,fixed_rank)
```

Description

Computes a rank-revealing VSV decompostion A = V∗(L'∗L)∗V' of a symmetric semidefinite n-by-n matrix. Only the upper triangular part of needs to be specified. Optimized for matrices whose rank p is small compared to n.

Input Parameters

A	symmetric semidefinite matrix;
tol_rank	rank decision tolerance;
max_iter	max. number of inverse iterations per deflation step, used in the singular vector estimator;
est_type	if true, then estimate singular vectors by means of the Lanczos procedure, else use the power method;
fixed_rank	deflate to the fixed rank given by fixed_rank instead of using the rank decision tolerance;

Defaults	tol_rank = n∗norm(A,1)∗eps;
	max_iter = 5;
	est_type = 0 (power method);

Output Parameters

p	numerical rank of A;
L	lower triangular matrix in A = V∗(L'∗L)∗V';
V	orthogonal matrix in A = V∗(L'∗L)∗V';

Algorithm

The symmetric semidefinite matrix A is preprocessed by a pivoted Cholesky factorization and then postprocessed by a low-rank revealing ULV decomposition, using either power or Lanczos iterations to estimate the dominant singular vectors. An indefinite matrix results in an error message during the Cholesky factorization.

See Also

hvsvid	–	High-rank revealing VSV alg. for symmetric indefinite matrices
hvsvid_R	–	High-rank revealing VSV alg., sym. indef. matrices, R version
hvsvsd	–	High-rank revealing VSV alg. for symmetric semidefinite matrices
lvsvid	–	Low-rank revealing VSV alg. for symmetric indefinite matrices

References

[1] P.C. Hansen & P.Y. Yalamov, “Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization”, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 443–458.

TOTinviter

Purpose

Inverse iterations for T'∗Omega∗T.

Synopsis

```
[smin,vmin] = TOTinviter(T,Omega,itmax)
```

Description

Inverse iterations on the product T'∗Omega∗T to compute the smallest singular value and the corresponding singular vector.

Input Parameters

T	triangular matrix;
Omega	diagonal matrix;
itmax	maximum number of iterations;

Output Parameters

smin	estimate of smallest singular value;
vmin	estimate of corresponding singular vector;

TOTlanczos

Purpose

Symmetric Lanczos procedure for T'*Omega*T.

Synopsis

[smax,vmax] = TOTlanczos(T,Omega,itmax)
[smax,vmax] = TOTlanczos(T,Omega,itmax,reorth)

Description

Computes an estimate of the largest singular value and the associated singular vector of the matrix T'*Omega*T using a maximum of itmax Lanczos iterations. If reorth = 1, then MGS reorthogonalization is used.

Input Parameters

T	matrix;
Omega	diagonal matrix;
itmax	maximum number of iterations;
reorth	MGS reorthogonalization if true;
Defaults	reorth = 1 (reorthogonalization).

Output Parameters

smin	estimate of smallest singular value;
vmin	estimate of corresponding singular vector;

See Also

TOTpowiter – Power iterations for T'*Omega*T

References

[1] G.H. Golub and C.F. Van Loan, "Matrix Computations", Johns Hopkins University Press, 3. Ed., 1996; p. 480.

TOTpowiter

Purpose

Power iterations for T'∗Omega∗T.

Synopsis

```
[smin,vmin] = TOTpowiter(T,Omega,itmax)
```

Description

Power iterations on the product T'∗Omega∗T to compute the largest singular value and the corresponding singular vector.

Input Parameters

T	triangular matrix;
Omega	diagonal matrix;
itmax	maximum number of iterations;

Output Parameters

smin	estimate of largest singular value;
vmin	estimate of corresponding singular vector;

See Also

TOTlanczos – Lanczos procedure for T'∗Omega∗T

tprod

Purpose

Toeplitz matrix-vector multiplication via FFT.

Synopsis

y = tprod(lambda,m,n,x,transp)

Description

This routine computes T∗x for tranps = 0 or T'∗x for value = 1, where T is an m-by-n Toeplitz matrix, using the eigenvalues lambda of a related circulant matrix computed ny means of tprodinit.

Input Parameters

lambda	eigenvalue vector need for the FFT;
m, n	dimensions of Toeplitz matrix;
x	vector to be multiplied by T;
transp	if 0 or nonexisting, multiply with T, ortherwise multiply with T';

Output Parameters

y	matrix-vector product

Algorithm

Let lambda be the eigenvalues of a circulant matrix derived from T (see tprodinit); then the product T*x consists of the first m elements of the vector ifft(lambda.*fft([x,z])), where z is a vector of zeros, while the product T'*x consists of the first n elements of the vector ifft(conj(lambda).*fft([x,z])).

See Also

tprodinit – Initialization routine for tprod

References

[1] P.C. Hansen, "Decconvolution and regularization with Toeplitz matrices," Numer. Algo. 29 (2002), pp. 323-378.

tprodinit

Purpose

Initialization routine for tprod (Toeplitz matrix-vector product)

Synopsis

lambda = tprodinit(colT,rowT)

Description

lambda contains the eigenvalues of a related circulant matrix, needed for matrix-vector multiplication with the Toeplitz matrix specified by colT and rowT. The length of lmbda is a powere of 2.

Input Parameters

colT	contains the first column of the Toeplitz matrix;
rowT	contains the first row of the Toeplitz matrix;

Output Parameters

lambda	contains the eigenvalues of an extended circulant matrix;

Algorithm

lambda contains the eigenvalues of the circulant matrix C whose first column is c = [colT;z;rowT(end:-1:2)], where z is a zero columns with dimensions such that length(c) is a power of 2 (for efficiency).

See Also

tprod – Toeplitz matrix-vector product, using tprodinit

References

[1] P.C. Hansen, "Decconvolution and regularization with Toeplitz matrices," Numer. Algo. 29 (2002), pp. 323-378.

tvsv

Purpose

Solves a rank-deficient system using the VSV decomposition.

Synopsis

x_tvsv = tvsv(L,V,p,b)
x_tvsv = tvsv(R,Omega,V,p,b)
x_tvsv = tvsv(L,Omega,V,p,b)

Description

Solves the symmetric and near-rank deficient system of equations A x = b, using the rank-revealing VSV decomposition of A. Three decompositions of A can be used:

```
A = V*L'*L*V'          (semidefinite A, lower triangular L)
A = V*R'*Omega*R*V'    (indefinite A, upper triangular R)
A = V*L'*Omega*L*V'    (indefinite A, lower triangular L)
```

Input Parameters

Semindefinite case:

L	lower triangular matrix;
V	orthognal matrix;
p	numerical rank;
b	right-hand side;

Indefinite case:

L or R	lower or upper triangular matrix;
Omega	signature matrix
V	orthognal matrix;
p	numerical rank;
b	right-hand side;

Output Parameters

x_tvsv	truncated VSV solution;

References

[1] P. C. Hansen and P. Y. Yalamov, "Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization," SIAM J. Matrix Anal. Appl., 23 (2001), pp. 443–458.

ulliv

Purpose

High-rank-revealing ULLV algorithm, B full row rank.

Synopsis

[p,LA,L,V,UA,UB,vec] = ulliv(A,B)
[p,LA,L,V,UA,UB,vec] = ulliv(A,B,tol_rank)
[p,LA,L,V,UA,UB,vec] = ulliv(A,B,tol_rank,tol_ref,max_ref)
[p,LA,L,V,UA,UB,vec] = ulliv(A,B,tol_rank,tol_ref,max_ref,fixed_rank)

Description

Computes a rank-revealing ULLIV decomposition of an mA-by-n matrix A with mA $\geq$ n, and an mB-by-n matrix B with full row rank mB $\leq$ n:

```
A = UA*LA*[ L 0 ]*V'   and    B = UB*[ L 0 ]*V'
          [ 0 I ]
```

Here, LA is n-by-n and L is mB-by-mB and both are lower triangular; UA, UB and V are unitary matrices, where only the first nA column of UA are computed.

The ULLV decomposition is a quotient ULV decomposition of the quotient A∗pinv(B)_A, where pinv(B)_A is the A-weighted pseudoinverse of B:

```
A*pinv(B)_A = UA(:,1:mB)*LA(1:mB,1:mB)*UB' .
```

Hence the lower triangular matrix LA(1:mB,1:mB) reveals the numerical rank p of the matrix quotient.

Note that the algorithm is optimized for numerical rank p close to nB, and that this algorithm should not be used if B is ill conditioned or rank deficient. Use the function ullv if B has full column rank.

Input Parameters

A	mA-by-n matrix (mA $\geq$ n);
B	mB-by-n matrix (mB $\leq$ n);
tol_rank	rank decision tolerance;
tol_ref	upper bound on the 2-norm of the off-diagonal block LA(p+1:n,1:p) relative to the Frobenius-norm of LA;
max_ref	max. number of refinement steps per singular value to achieve the upper bound tol_ref;
fixed_rank	deflate to the fixed rank given by fixed_rank instead of using the rank decision tolerance;

Defaults	tol_rank = sqrt(n)∗norm(A,1)∗eps;
	tol_ref = 1e-04;
	max_ref = 0;

Output Parameters

p	numerical rank of A∗pinv(B)_A;
LA,L,V,UA,UB	the ULLV factors;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(LA(p+1:n,1:p)), vec(2) = estimate of pth singular value, vec(3) = estimate of (p+1)th singular value, vec(4) = a posteriori upper bound of num. nullspace angle, vec(5) = a posteriori upper bound of num. range angle.

Algorithm

First compute the LQ factorization B = [L,0]∗V' and then form the matrix X = A∗V∗inv(diag(L,I)), followed by the QR factorization X = UA∗LA. Thus, A = UA∗LA∗diag(L,I)∗V' and B = L∗V'. Then deflation and refinement (optional) are employed to produce a rank-revealing decomposition. The deflation procedure is based on the generalized LINPACK condition estimator, and the refinement steps on QR-iterations.

See Also

ullv – Rank-revealing ULLV algorithm, B full column rank.

References

[1] F.T. Luk and S. Qiao, "A New Matrix Decomposition for Signal Processing", Automatica, 30 (1994), pp. 39–43.

[2] F.T. Luk and S. Qiao, "An adaptive algoithm for interference cancelling in array processing; in F.T. Luk (Ed.), "Advanced Signal Processing Algorithms, Architectures, and Implementations VI," SPIE Proceedings, Vol. 2846 (1996), pp. 151-161.

ulliv_up_a

Purpose

Update the A-part of the rank-revealing ULLIV decomposition.

Synopsis

[p,LA,L,V,UA,UB,vec] = ulliv_up_A(p,LA,L,V,UA,UB,a)
[p,LA,L,V,UA,UB,vec] = ulliv_up_A(p,LA,L,V,UA,UB,a,tol_rank)
[p,LA,L,V,UA,UB,vec] = ulliv_up_A(p,LA,L,V,UA,UB,a,tol_rank,tol_ref,max_ref)
[p,LA,L,V,UA,UB,vec] = ulliv_up_A(p,LA,L,V,UA,UB,a,tol_rank,tol_ref,max_ref,fixed_rank)

Description

Given a rank-revealing ULLIV decomposition of the mA-by-n matrix with mA $\geq$ n, and the full-rank mB-by-n matrix B = UB∗L∗V' with mB < n, the function computes the updated decomposition

```
[ A ] = UA*LA*[ L 0 ]*V'  and  B = UB*[ L 0 ]*V'
[ a ]         [ 0 I ]
```

where a is the new row added to A. Note that B must have full row rank, that the row dimension of UA will increase by one, and that the matrices UA and UB can be left out by inserting an empty matrix `[]` while V is required.

Input Parameters

p	numerical rank of A∗pinv(B)_A revealed in LA;
LA,L,V,UA,UB	the ULLIV factors of A and B;
a	the new row added to A;
tol_rank	rank decision tolerance;
tol_ref	upper bound on the 2-norm of the off-diagonal block LA(p+1:mB,1:p) relative to the Frobenius-norm of LA;
max_ref	max. number of refinement steps per singular value to achieve the upper bound tol_ref;
fixed_rank	if true, deflate to the fixed rank given by p instead of using the rank decision tolerance;

Defaults	tol_rank = sqrt(n)∗norm(LA,1)∗eps;
	tol_ref = 1e-04;
	max_ref = 0;

Output Parameters

p	numerical rank of updated quotient;
LA,L,V,UA,UB	the updated ULLV factors;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(LA(p+1:mB,1:p)), vec(2) = estimate of pth singular value, vec(3) = estimate of (p+1)th singular value, vec(4) = a posteriori upper bound of num. nullspace angle, vec(5) = a posteriori upper bound of num. range angle.

See Also

ulliv_up_B – Update the B-part of the rank-revealing ULLIV decomp.

References

[1] F.T.Luk and S. Qiao, "A New Matrix Decomposition for Signal Processing", Automatica, 30 (1994), pp. 39–43.

[2] F.T.Luk and S. Qiao, "An adaptive algoithm for interference cancelling in array processing; in F.T. Luk (Ed.), "Advanced Signal Processing Algorithms, Architectures, and Implementations VI," SPIE Proceedings, Vol. 2846 (1996), pp. 151-161.

ulliv_up_B

Purpose

Update the B-part of the rank-revealing ULLIV decomposition.

Synopsis

[p,LA,L,V,UA,UB,vec] = ulliv_up_B(p,LA,L,V,UA,UB,b)
[p,LA,L,V,UA,UB,vec] = ulliv_up_B(p,LA,L,V,UA,UB,b,tol_rank)
[p,LA,L,V,UA,UB,vec] = ulliv_up_B(p,LA,L,V,UA,UB,b,tol_rank,tol_ref,max_ref)
[p,LA,L,V,UA,UB,vec] = ulliv_up_B(p,LA,L,V,UA,UB,b,tol_rank,tol_ref,max_ref,fixed_rank)

Description

Given a rank-revealing ULLIV decomposition of the mA-by-n matrix with mA $\geq$ n, and the full-rank mB-by-n matrix B = UB∗L∗V' with mB < n, the function computes the updated decomposition

```
A = UA*LA*[ L 0 ]*V'  and  [ B ] = UB*[ L 0 ]*V'
          [ 0 I ]          [ b ]
```

where b is the new row added to B. Note that the updated matrix [B;b] must have full row rank, that the row dimension of UB will increase by one, and that the matrices UA and UB can be left out by inserting an empty matrix [] while V is required.

Input Parameters

p	numerical rank of A∗pinv(B)_A revealed in LA;
LA,L,V,UA,UB	the ULLIV factors of A and B;
b	the new row added to B;
tol_rank	rank decision tolerance;
tol_ref	upper bound on the 2-norm of the off-diagonal block LA(p+1:mB,1:p) relative to the Frobenius-norm of LA;
max_ref	max. number of refinement steps per singular value to achieve the upper bound tol_ref;
fixed_rank	if true, deflate to the fixed rank given by p instead of using the rank decision tolerance;
Defaults	tol_rank = sqrt(n)∗norm(LA,1)∗eps; tol_ref = 1e-04; max_ref = 0;

Output Parameters

p	numerical rank of updated quotient;
LA,L,V,UA,UB	the updated ULLV factors;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(LA(p+1:mB,1:p)), vec(2) = estimate of pth singular value, vec(3) = estimate of (p+1)th singular value, vec(4) = a posteriori upper bound of num. nullspace angle, vec(5) = a posteriori upper bound of num. range angle.

See Also

ulliv_up_A – Update the A-part of the rank-revealing ULLIV decomp.

References

[1] F.T.Luk and S. Qiao, "A New Matrix Decomposition for Signal Processing", Automatica, 30 (1994), pp. 39–43.

[2] F.T.Luk and S. Qiao, "An adaptive algoithm for interference cancelling in array processing; in F.T. Luk (Ed.), "Advanced Signal Processing Algorithms, Architectures, and Implementations VI," SPIE Proceedings, Vol. 2846 (1996), pp. 151-161.

vsv_qrit

Purpose

Refinement of VSV decomposition via block QR-iterations.

Synopsis

[L] = vsv_qrit(p,num_ref,L)
[L,V] = vsv_qrit(p,num_ref,L,[],V)
[L,Omega] = vsv_qrit(p,num_ref,L,Omega)
[L,Omega,V] = vsv_qrit(p,num_ref,L,Omega,V)
[R,Omega] = vsv_qrit(p,num_ref,R,Omega)
[R,Omega,V] = vsv_qrit(p,num_ref,R,Omega,V)

Description

Given a VSV decomposition with numerical rank p, of one of the forms

```
A = V*L'*L*V'          (semidefinite A, lower triangular L)
A = V*L'*Omega*L*V'    (indefinite A, lower triangular L)
A = V*R'*Omega*R*V'    (indefinite A, upper triangular R)
```

the function refines the rank-revealing decomposition via num_ref steps of block QR iterations applied to the triangular matrix.

Input Parameters

p	numerical rank of A;
num_ref	number of refinement iterations;
T	triangur matrix (L or R, depending on VSV decomposition);
Omega	signature matrix (indef. case) or empty (semidef. case);
V	orthogonal matrix;

Output Parameters

T	refined triangular matrix
Omega or V	refined Omega (indef. case) or refined V (semidef. case)
V	refined V (indef. case)

Algorithm

Refinement is identical to block QR iteration, in which the off-diagonal block of the triangular matrix is “flipped” to the diagonally opposite position and then back again.

See Also

ulv_ref – Refine one column of L in the ULV decomposition.
urv_ref – Refine one column of R in the URV decomposition.

References

[1] R. Mathias and G.W. Stewart, “A Block QR Algorithm and the Singular Value Decomposition”, Lin. Alg. Appl., 182 (1993), pp. 91–100.

vsvid_L_mod

Purpose

Rank-one modification of VSV decomp. of sym. indef. matrix, L version.

Synopsis

```
[p,L,Omega,V] = vsvid_L_mod(p,L,Omega,V,omega,v)
[p,L,Omega,V] = vsvid_L_mod(p,L,Omega,V,omega,v,tol_rank)
[p,L,Omega,V] = vsvid_L_mod(p,L,Omega,V,omega,v,tol_rank,inv_iter)
[p,L,Omega,V] = vsvid_L_mod(p,L,Omega,V,omega,v,tol_rank,inv_iter,fixed_rank)
```

Description

Given a rank-revealing VSV decomposition of a symmetric indefinite matrix A = V∗(L'∗Omega∗L)∗V', the function computes the updated rank-revealing decomposition of the matrix A + omega∗v∗v', where omega = +1 or -1.

Input Parameters

p	numerical rank of A;
L	lower triangular matrix in A = V∗(L'∗Omega∗L)∗V';
Omega	signature matrix in A = V∗(L'∗Omega∗L)∗V';
V	orthogonal matrix in A = V∗(L'∗Omega∗L)∗V';
omega	update (+1) or downdate (-1);
v	rank-one update column vector;
tol_rank	rank decision tolerance;
inv_iter	number of inverse iterations per deflation step;
fixed_rank	deflate to the fixed rank given by fixed_rank instead of using the rank decision tolerance;
Defaults	tol_rank = n∗norm(L,1)∗eps; inv_iter = 5;

Output Parameters

p	numerical rank of modified matrix;
L	updated lower triangular matrix;
Omega	updated signature matrix;
V	updated orthogonal matrix;

See Also

vsvid_R_mod – Rank-one mod. of VSV decomp. of sym. indef. matrix, R version.
vsvsd_up – Rank-one update of VSV decomposition of semidefinite matrix.

References

[1] P.C. Hansen & P.Y. Yalamov, "Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization", SIAM J. Matrix Anal. Appl., 23 (2001), pp. 443–458.

vsvid_R_mod

Purpose

Rank-one modification of VSV decomp. of indef. matrix, R version.

Synopsis

```
[p,R,Omega,V] = vsvid_R_mod(p,R,Omega,V,omega,v)
[p,R,Omega,V] = vsvid_R_mod(p,R,Omega,V,omega,v,tol_rank)
[p,R,Omega,V] = vsvid_R_mod(p,R,Omega,V,omega,v,tol_rank,inv_iter)
[p,R,Omega,V] = vsvid_R_mod(p,R,Omega,V,omega,v,tol_rank,inv_iter,fixed_rank)
```

Description

Given a rank-revealing VSV decomposition of a symmetric indefinite matrix A = V∗(R'∗Omega∗R)∗V', the function computes the updated rank-revealing decomposition of the matrix A + omega∗v∗v', where omega = +1 or -1.

Input Parameters

p	numerical rank of A;
R	upper triangular matrix in A = V∗(R'∗Omega∗R)∗V';
Omega	signature matrix in A = V∗(R'∗Omega∗R)∗V';
V	orthogonal matrix in A = V∗(R'∗Omega∗R)∗V';
omega	update (+1) or downdate (-1);
v	rank-one update column vector;
tol_rank	rank decision tolerance;
inv_iter	number of inverse iterations per deflation step;
fixed_rank	deflate to the fixed rank given by fixed_rank instead of using the rank decision tolerance;
Defaults	tol_rank = n∗norm(R,1)∗eps; inv_iter = 5;

Output Parameters

p	numerical rank of updated A;
R	updated upper triangular;
Omega	updated signature matrix;
V	updated orthogonal matrix;

See Also

vsvid_mod – Rank-one modification of VSV decomposition of indefinite matrix.
vsvsd_up – Rank-one update of VSV decomposition of semidefinite matrix.

References

[1] P.C. Hansen & P.Y. Yalamov, "Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization", SIAM J. Matrix Anal. Appl., 23 (2001), pp. 443–458.

vsvid_ip

Purpose

Interim process for indefinite VSV algorithm.

Synopsis

`[W,C,Omega] = vsvid_ip(L,D)`

Input Parameters

L,D	Factors in LDL^T factorization;

Output Parameters

W,C,Omega	Matrices in L*D*L' = W*C'*Omega*C*W'.

Algorithm

The factorization L*D*L' is replaced with W*C'*Omega*C*W', where W is orthogonal, C is upper triangular, and Omega is a signature matrix. This is accomplished via small eigenvalue decompositions of the 1-by-1 and 2-by-2 blocks of D.

References

[1] P.C. Hansen, & P.Y. Yalamov, "Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization", SIAM J. Matrix Anal. Appl., 23 (2001), pp. 443–458.

vsvsd_up

Purpose

Rank-one update of VSV decomposition of semidefinite matrix.

Synopsis

[p,L,V] = vsvsd_up(p,L,V,v)
[p,L,V] = vsvsd_up(p,L,V,v,tol_rank)
[p,L,V] = vsvsd_up(p,R,V,v,tol_rank,fixed_rank)

Description

Given a rank-revealing VSV decomposition of a symmetric semidefinite matrix A = V*(L'*L)*V', the function computes the updated rank-revealing decomposition of A + v*v'. Use function vsvid_mod with omega = -1 to downdate the VSV decomposition of a symmetric semidefinite matrix.

Input Parameters

p	numerical rank of A;
L	lower triangular matrix in A = V*(L'*L)*V';
V	orthogonal matrix in A = V*(L'*L)*V';
v	rank-one update column vector;
tol_rank	rank decision tolerance;
fixed_rank	deflate to the fixed rank given by fixed_rank instead of using the rank decision tolerance;
Defaults	tol_rank = n*norm(L,1)*eps;

Output Parameters

p	numerical rank of updated A;
L	updated lower triangular matrix;
V	updated orthogonal matrix;

See Also

vsvid_L_mod – Rank-one mod. of VSV decomp. of sy,. indef. matrix, L version.
vsvid_R_mod – Rank-one mod. of VSV decomp. of sy,. indef. matrix, R version.

References

[1] P.C. Hansen & P.Y. Yalamov, "Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization", SIAM J. Matrix Anal. Appl., 23 (2001), pp. 443–458.