UTV Tools

Matlab Templates for Rank-Revealing UTV Decompositions

Version 1.0 for MATLAB 5.2

Ricardo D. Fierro

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

Per Christian Hansen

and

Peter Søren Kirk Hansen

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

March 1990

This work was supported in part by NATO Collaborative Research Grant No. 951327. The software is available from Netlib at `http://www.netlib.org/numeralgo`. The software was published in Numerical Algorithms and the paper's text is reprinted here by kind permission of the publisher.

Contents

1. Introduction

Algorithms based on orthogonal transformations play an important role in many signal processing applications. There are several reasons for this. Orthogonal transformations are numerically stable, which is particularly important when the matrix dimensions m and n increase and/or the condition number increases, and when the numerical rank of the matrix is an issue. Decompositions based on orthogonal transformations are often easy to update in a reliable fashion, thus reducing the computational burden by a factor m or n. And, finally, these decompositions can yield information about certain subspaces defined on the matrix which play an essential role in noise suppression techniques and other signal processing applications.

One of the main numerical tools in signal processing based on orthogonal transformations is the singular value decomposition (SVD), cf. [5], [26, §2.5], [57, §1.4.3], and its generalizations to matrix pairs, triplets, etc. The SVD detects near-rank deficiency in a matrix very reliably and yields all the necessary subspace information. Because the SVD algorithm is so reliable and numerically stable, it is used in a wide variety of applications, such as frequency estimation via least squares and total least squares [48], [49], [58], principal component analysis [60], noise reduction in speech processing [33], computer-aided geometric design [40], and information retrieval [4]. Additional applications of the SVD can be found in the *International Workshop on SVD and Signal Processing* proceedings [17], [42], [59].

Although the SVD is a valuable analytical and computational tool, it has certain drawbacks. First, for many problems the SVD is unpractical because the algorithm is unable to take advantage of important matrix properties, such as structure or sparsity, to minimize both computational work-load and storage requirements. This is due to the full bidiagonalization phase in the algorithm. This drawback also appears in specialized algorithms such as the partial SVD algorithm designed to compute only the "needed" information. Second, the SVD is difficult to update and downdate [9], [27], and thus it is not always suitable for applications with real-time constraints. Depending on the application, these difficulties with the SVD make alternative decompositions attractive, provided they are nearly as reliable and more efficient to compute and up/downdate.

The rank-revealing QR (RRQR) decomposition [10], [24] is one of the alternatives to the SVD, being faster to compute and yet providing reliable estimates for the rank and the desired subspaces. Indeed, the RRQR decomposition has advantages in sparse matrix computations [47] and subset selection problems [26, §12.2], but unfortunately its representation of the numerical null space is not well suited for up- and downdating [6].

Rank-revealing two-sided orthogonal decompositions, also referred to as UTV decompositions [55], [57, §5.4] or complete orthogonal decompositions [26, §5.4.2], are other promising alternatives to the SVD that provide reliable estimates for the numerical rank and the desired subspaces. There are two main advantages in using rank-revealing UTV decompositions instead of the SVD or RRQR decomposition: UTV decompositions can be computed more efficiently than the SVD, and their subspace information is easier to up- and downdate, cf. [3], [26, §12.5.5], [45], [52], [53]. Some applications of UTV decompositions can be found in [1], [2], [23], [37], [46], [53].

The SVD can be generalized to pairs of matrices in several ways, depending on the application [16], and the same holds for the UTV decompositions. For example, the socalled ULLV decomposition due to Luk and Qiao [38] reveals the numerical rank of the matrix "quotient" $A\,B^\dagger$ (where $B^\dagger$ is the pseudoinverse of the second matrix B), and thus it matches the quotient SVD. The ULLV decomposition can be up- and downdated by means of the same techniques as the UTV decompositions.

The purpose of this work is to provide a package with easy-to-use Matlab templates for computing and working with UTV decompositions. For completeness, we include a few templates for computing the RRQR decomposition. In our implementations we focus on robustness and modularity, rather than ultimate performance. The reason behind this choice is that in most signal processing applications, the algorithms must be tuned to the particular application anyway. Hence, we consider Matlab templates the optimal way to communicate algorithms, developed by numerical analysts, to the signal processing community and other users. Our package requires Matlab Version 5.2, and it is available from Netlib at `http://www.netlib.org/numeralgo`.

Our notation is standard linear algebra notation plus Matlab-style matrix indexing where, e.g., $A(1\colon k, 1\colon k)$ denotes the leading $k \times k$ submatrix of A. The particular notation used here follows closely the one used in [21] where the accuracy of the various quantities, computed by means of UTV and RRQR decompositions, are investigated, and where several numerical examples can be found.

After a brief introduction to rank-revealing decompositions, we summarize some important properties of UTV decompositions in Section 2. Next, in Sections 3 and 4, we describe the algorithms used in this package for computing and up/downdating UTV decompositions. In Section 5 we turn to definitions and algorithms for the ULLV decomposition of a matrix pair. We do by no means attempt to be complete; all algorithmic details can be found elsewhere in the literature, and pointers to the relevant literature is always given. We conclude with manual pages in Section 6 for all 46 functions included in the package.

We wish to thank Adam Bojanczyk for sharing his ULLV algorithms with us during the early stages of this project. Also thanks to Sanzheng Qiao for kindly providing us with his Matlab software.

2. Rank-Revealing Orthogonal Decompositions

Roughly speaking, a rank-revealing decomposition is a decomposition in which information about the numerical rank of the matrix can be easily extracted. Here, "numerical rank" usually means the number of singular values larger than a certain threshold, and it is important to realize that for this concept to make sense, there has to be a well-determined gap in the singular value spectrum at the threshold [25], [29, §3.1]. Hence, rank-revealing decompositions may also be labeled as "gap-revealing decompositions," a phrase coined by Stewart [56]. General treatments of rank-revealing decompositions are presented in [29, Chap. 3] and [57, Chap. 5].

Turning to algorithms for computing rank-revealing orthogonal decompositions, experience shows that it is natural to distinguish between high-rank and low-rank algorithms for the two important cases where the numerical rank is either close to the number of rows or columns of the matrix, or much smaller. So far, no efficient algorithm has been developed for computing a rank-revealing orthogonal decomposition of a matrix whose numerical rank is approximately half the number of rows or columns.

All rank-revealing orthogonal decompositions introduced so far are two-sided in nature, i.e., they are of the general form $A = X\,M\,Y^T$, where the two "outer matrices" X and Y are orthogonal — occasionally they are permutation matrices — and the "middle matrix" M is the matrix that reveals the numerical rank or gap. We conjecture that rank-revealing decompositions must be two-sided; for example, in connection with the RRQR decomposition, column permutations are needed to guarantee that one reliably detects the numerical rank.

2.1. The Singular Value Decomposition

The most well-known example of a rank-revealing two-sided orthogonal decomposition is the *singular value decomposition* (SVD), cf. [26, §2.5]. The SVD of an $m \times n$ matrix A with $m \geq n$ is given by

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T = U_1\,\Sigma\,V^T = \sum_{i=1}^{r} \sigma_i\,u_i\,v_i^T, \tag{2.1}$$

where $U_1 = U(:,1\!:\!n)$ and $r = \mathrm{rank}(A)$. Both U and V are orthogonal, i.e., $U^T U = I_m$ and $U_1^T U_1 = V^T V = I_n$. The diagonal elements σ_i of the $n \times n$ diagonal matrix Σ are called the *singular values* of A with the ordering

$$\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0.$$

The columns of U and V are referred to as the *left* and *right singular vectors*, respectively. The first r columns of U and V are the orthonormal eigenvectors associated with the r nonzero eigenvalues of AA^T and $A^T A$, respectively.

Given an integer $k \le r$, we partition the SVD according to

$$A = (U_k,\ U_0,\ U_\perp) \begin{pmatrix} \Sigma_k & 0 \\ 0 & \Sigma_0 \\ 0 & 0 \end{pmatrix} (V_k,\ V_0)^T, \tag{2.2}$$

where $\Sigma_k = \mathrm{diag}(\sigma_1, \ldots, \sigma_k)$ and $\Sigma_0 = \mathrm{diag}(\sigma_{k+1}, \ldots, \sigma_n)$ are diagonal matrices consisting of the k largest and the $n-k$ smallest singular values, respectively. The matrix A_k defined by $A_k = U_k\,\Sigma_k\,V_k^T$ is a rank-k matrix approximation to A, and is the nearest one in the 2-norm. This matrix is called the truncated SVD matrix, and it has important theoretical and practical value [29, §3.2].

2.2. Numerical Rank and Singular Subspaces

The selection of k obviously depends on both the application and the method used to determine the parameter. One way is to simply specify the first k or last $n-k$ singular triplets $(\sigma_i,\ u_i,\ v_i)$ needed to capture the most relevant information in the data matrix A for the particular application. This approach is used, for example, in information retrieval [4]. A difficult aspect of this approach is that *ad hoc* procedures are often used to choose k.

Another way is to specify a *threshold* τ, and then k is identified as the largest integer such that $\sigma_k > \tau$. The parameter k is then called the *numerical rank* of A with respect to τ. Suppose, for example, that the singular values of A are

$$\sigma_1 = 1.0,\quad \sigma_2 = 0.5,\quad \sigma_3 = 0.1,\quad \sigma_4 = 10^{-5},\quad \text{and}\quad \sigma_5 = 10^{-10}.$$

Then $k = 3$ with respect to $\tau = 10^{-3}$, but $k = 2$ with respect to $\tau = 0.3$. The parameter k plays an important role in signal processing in distinguishing signal from noise, when the signal can be considered as a sum of a pure signal plus additive white noise. The parameter τ reflects the noise level, and k is related to the number of prominent signals. Moreover, the matrices U_k, Σ_k, and V_k carry information about the pure signal (plus some noise), while $(U_0,\ U_\perp)$, Σ_0, and V_0 carry information solely about the noise. Some issues concerning the very important subproblem of selecting τ for numerical rank detection are discussed in [28], [51], [54].

Once k is specified, there are four fundamental *numerical subspaces* defined by the SVD of A. They are

$$\begin{array}{lcll}
\mathcal{R}(A_k) & = & \mathcal{R}(U_k) & \text{the numerical range of } A \\
\mathcal{N}(A_k) & = & \mathcal{R}(V_0) & \text{the numerical null space of } A \\
\mathcal{R}(A_k^T) & = & \mathcal{R}(V_k) & \text{the numerical row space of } A \\
\mathcal{N}(A_k^T) & = & \mathcal{R}((U_0,\ U_\perp)) & \text{the numerical null space of } A^T.
\end{array}$$

Here, $\mathcal{R}(M)$ denotes the range (or column space) of the matrix M and $\mathcal{N}(M)$ denotes the null space of M.

2.3. UTV Decompositions

The SVD is a special two-sided decomposition because the middle matrix Σ is diagonal, and this is what makes the algorithm computationally expensive and also difficult to update. In many circumstances one can sacrifice the diagonal structure of Σ for a more efficient algorithm that computes a decomposition which provides approximately the same rank and subspace information, and which can be updated efficiently. This is the main idea behind the UTV decomposition, which is a product of three matrices: an orthogonal matrix, a middle matrix that is triangular or block-triangular, and another orthogonal matrix.

If the middle matrix is *upper triangular*, then the decomposition is called the URV decomposition, and for $m \geq n$ it takes the form

$$A = U_R \begin{pmatrix} R \\ 0 \end{pmatrix} V_R^T = (U_{Rk},\ U_{R0},\ U_{R\perp}) \begin{pmatrix} R_k & F \\ 0 & G \\ 0 & 0 \end{pmatrix} (V_{Rk},\ V_{R0})^T, \qquad (2.3)$$

where R_k is a $k \times k$ non-singular matrix and G is an $(n-k) \times (n-k)$ matrix. If A has a well-defined gap ($\sigma_{k+1} \ll \sigma_k$), then the URV decomposition is said to be *rank-revealing* if

$$\sigma_{\min}(R_k) = \mathcal{O}(\sigma_k) \qquad \text{and} \qquad \|(\, F^T,\ G^T\,)\|_2 = \mathcal{O}(\sigma_{k+1}). \qquad (2.4)$$

The second form, in which the middle matrix is *lower triangular*, is called the ULV decomposition, and it takes the form

$$A = U_L \begin{pmatrix} L \\ 0 \end{pmatrix} V_L^T = (U_{Lk},\ U_{L0},\ U_{L\perp}) \begin{pmatrix} L_k & 0 \\ H & E \\ 0 & 0 \end{pmatrix} (V_{Lk},\ V_{L0})^T, \qquad (2.5)$$

where L_k is a $k \times k$ non-singular matrix and E is an $(n-k) \times (n-k)$ matrix. If A has a well-defined gap ($\sigma_{k+1} \ll \sigma_k$), then the ULV decomposition is said to be *rank-revealing* if

$$\sigma_{\min}(L_k) = \mathcal{O}(\sigma_k) \qquad \text{and} \qquad \|(\, H,\ E\,)\|_2 = \mathcal{O}(\sigma_{k+1}). \qquad (2.6)$$

Although decompositions of the form (2.3) and (2.5) have been around for some time (they are discussed in the classical book by Lawson and Hanson [35] from 1974), algorithms which guarantee the rank-revealing property (2.4) and (2.6) are more recent. From the standard perturbation theory for singular values, cf. [26, §8.6.1], it follows that the smaller the norm of the off-diagonal block, the better the approximations in (2.4) and (2.6).

We mention that there are situations where σ_k and σ_{k+1} are not well separated but Σ_k, U_k, and V_k of the SVD are still useful. In some of these cases a UTV decomposition may still be an appropriate tool, but more research is needed in order to understand precisely when (see [31] for an example).

UTV decompositions are often used to supply good estimates of basis vectors for the numerical subspaces. For example, the subspace $\mathcal{R}(U_{Rk})$ or $\mathcal{R}(U_{Lk})$ can be considered an approximation to the numerical range $\mathcal{R}(U_k)$, and the subspaces are identical if the off-diagonal block is zero. Hence it is natural to compare the numerical subspaces of A to the corresponding UTV-based subspaces of A. The following theorem gives bounds for the distance between the SVD- and UTV-based subspaces (see [26, §2.6.3] for more information about subspace distances).

Theorem 1. (See Corollaries 2.3 and 2.5 in [20].) *Let A have the UTV decompositions as in (2.3) and (2.5) and the SVD as in (2.1). If $\sigma_{\min}(R_k) > \|G\|_2$, then*

$$\operatorname{dist}(\mathcal{R}(U_k), \mathcal{R}(U_{Rk})) \leq \frac{\|F\|_2 \, \|G\|_2}{\sigma_{\min}(R_k)^2 - \|G\|_2^2} \tag{2.7}$$

and

$$\frac{\|F\|_2}{2\,\|R\|_2} \leq \operatorname{dist}(\mathcal{R}(V_0), \mathcal{R}(V_{R0})) \leq \frac{\sigma_{\min}(R_k)\, \|F\|_2}{\sigma_{\min}(R_k)^2 - \|G\|_2^2}. \tag{2.8}$$

Similarly, if $\sigma_{\min}(L_k) > \|E\|_2$, then

$$\frac{\|H|_2}{2\,\|L\|_2} \leq \operatorname{dist}(\mathcal{R}(U_k), \mathcal{R}(U_{Lk})) \leq \frac{\sigma_{\min}(L_k)\, \|H\|_2}{\sigma_{\min}(L_k)^2 - \|E\|_2^2} \tag{2.9}$$

and

$$\operatorname{dist}(\mathcal{R}(V_0), \mathcal{R}(V_{L0})) \leq \frac{\|H\|_2 \, \|E\|_2}{\sigma_{\min}(L_k)^2 - \|E\|_2^2}. \tag{2.10}$$

The *a posteriori* bounds in (2.7)–(2.10) show that the UTV-based subspaces of A are accurate approximations of the singular subspaces of A provided the off-diagonal block of the middle matrix is sufficiently small in norm. In the next section we discuss strategies to compute UTV decompositions so that the off-diagonal block is sufficiently small.

Another important result that follows from Theorem 1 is that the URV-matrix U_{Rk}, considered as an approximate basis for the numerical range $\mathcal{R}(U_k)$, has a smaller upper bound than the corresponding ULV-matrix U_{Lk}, due to the factor $\|F\|_2$ instead of the factor $\sigma_{\min}(L_k)$. On the other hand, the ULV-matrix V_{L0}, considered as an approximate basis for the numerical null space $\mathcal{R}(V_0)$, has a smaller upper bound than the corresponding URV-matrix V_{R0}. We conclude that the choice of decomposition depends on which quantities one wants to estimate; e.g., if one wants to estimate numerical null spaces then the ULV decomposition is preferred.

At this stage we mention that the matrix A is often a noisy realization of a pure matrix $\overline{A}$ plus additive noise, where $\overline{A}$ can be assumed to be exactly rank deficient. Hence, it is of interest to compare the UTV-based subspaces, computed from A, with the corresponding exact subspaces defined from $\overline{A}$ which, in turn, are identical to the fundamental SVD-based subspaces of $\overline{A}$. The relevant perturbation theory can be found in [19].

The standard way to use the UTV decompositions in solving numerically rank-deficient least squares problems $\min \|Ax - b\|_2$ is to "plug in" either of the UTV decompositions for A and then neglect the two blocks with small norm, i.e., either F and G in the URV decomposition, or E and H in the ULV decomposition. The corresponding UTV-based least squares solutions are then given by

$$x_{Rk} = V_{Rk} R_k^{-1} U_{Rk}^T b \qquad \text{and} \qquad x_{Lk} = V_{Lk} L_k^{-1} U_{Lk}^T b,$$

and the accuracy of these solutions is investigated in [21]. The computation of truncated UTV solutions is implemented in the two Matlab functions `tulv` and `turv` for computing x_{Rk} and x_{Lk}, respectively. Similar UTV-based total least squares solutions are studied in [62].

Although the RRQR decomposition was not introduced this way, it can be considered as a special URV decomposition in which the right orthogonal matrix is a permutation matrix Π. It is customary to write this decomposition in the form

$$A\,\Pi = Q \begin{pmatrix} R_Q \\ 0 \end{pmatrix} = (Q_1,\; Q_2,\; Q_\perp) \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \\ 0 & 0 \end{pmatrix}, \qquad \Pi = (\Pi_1,\; \Pi_2).$$

Computation of the RRQR decomposition is implemented in the two Matlab functions **hrrqr** and **lrrqr**, designed for the high- and low-rank cases, respectively, cf. [10], [12], and [24]. The corresponding truncated RRQR solution is given by $x_{Qk} = \Pi_1(R_{11},\, R_{12})^\dagger Q_1^T b$, and computation of this solution is implemented in the Matlab function **trrqr**. We note that RRQR algorithms often, in addition to the three matrices Q, R, and Π, return a matrix W whose columns span an approximation to either $\mathcal{N}(A_k)$, in the high-rank case, or $\mathcal{R}(A_k^T)$, in the low-rank case, and RRQR-based total least squares solutions can be based on this matrix. More details about RRQR-based solutions can be found in [11], while a study of the accuracy of the RRQR-based subspaces and solutions is presented in [21].

2.4. A Numerical Example

To illustrate some of the quantities defined above, we generate an 8×6 matrix A with singular values

$$2,\ 1,\ 0.5,\ 0.2,\ 0.05,\ 0.001,$$

and the numerical rank of A, with respect to the threshold $\tau = 0.1$, is $k = 4$. Then we use the Matlab function **hurv** (described in the next section) to compute a rank-revealing decomposition of A. The computed triangular factor R has the form

$$R = \begin{pmatrix} 0.47 & 0.67 & -0.54 & 0.92 & 7.7\cdot 10^{-5} & -4.3\cdot 10^{-6} \\ 0 & 1.46 & -0.79 & 0.03 & -3.8\cdot 10^{-5} & 1.2\cdot 10^{-6} \\ 0 & 0 & 0.64 & -0.34 & 5.2\cdot 10^{-5} & -1.8\cdot 10^{-6} \\ 0 & 0 & 0 & 0.46 & -1.1\cdot 10^{-4} & 6.5\cdot 10^{-6} \\ 0 & 0 & 0 & 0 & 4.5\cdot 10^{-3} & 2.3\cdot 10^{-4} \\ 0 & 0 & 0 & 0 & 0 & 1.0\cdot 10^{-3} \end{pmatrix}.$$

The norms of the three nonzero blocks are $\|R_k\|_2 = 2.00$, $\|F\|_2 = 1.46\cdot 10^{-4}$, and $\|G\|_2 = 5.00\cdot 10^{-3}$, clearly revealing the numerical rank of A, and the smallest singular value of R_k is $\sigma_{\min}(R_k) = 0.20$. The accuracy of the estimated subspaces, and the corresponding upper bounds from Theorem 1, are as follows:

$$\mathrm{dist}(\mathcal{R}(U_k), \mathcal{R}(U_{Rk})) = 1.68\cdot 10^{-5}, \qquad \frac{\|F\|_2\,\|G\|_2}{\sigma_{\min}(R_k)^2 - \|G\|_2^2} = 1.85\cdot 10^{-5}$$

$$\mathrm{dist}(\mathcal{R}(V_0), \mathcal{R}(V_{R0})) = 6.85\cdot 10^{-4}, \qquad \frac{\sigma_{\min}(R_k)\,\|F\|_2}{\sigma_{\min}(R_k)^2 - \|G\|_2^2} = 7.35\cdot 10^{-4}$$

and we see that the upper bounds are very close to the actual subspace distances. This example illustrates that the rank-revealing URV decomposition indeed provides good estimates for the SVD-based quantities, and that the numerical range is better approximated than the numerical null space.

We also compute rank-revealing ULV and RRQR decompositions by means of the Matlab functions **hulv** and **hrrqr**, and the corresponding triangular matrices L and R_Q are

$$L = \begin{pmatrix} 0.22 & 0 & 0 & 0 & 0 & 0 \\ -0.11 & 0.90 & 0 & 0 & 0 & 0 \\ -0.02 & -1.18 & 0.93 & 0 & 0 & 0 \\ 0.39 & -0.61 & 0.68 & -1.08 & 0 & 0 \\ 1.4 \cdot 10^{-4} & 4.7 \cdot 10^{-5} & 9.1 \cdot 10^{-5} & 5.6 \cdot 10^{-5} & 5.0 \cdot 10^{-3} & 0 \\ 4.4 \cdot 10^{-6} & 1.5 \cdot 10^{-6} & 2.9 \cdot 10^{-6} & 1.8 \cdot 10^{-6} & 1.3 \cdot 10^{-4} & 1.0 \cdot 10^{-3} \end{pmatrix}$$

and

$$R_Q = \begin{pmatrix} 0.89 & 0.74 & 0.81 & 0.22 & 0.56 & 0.94 \\ 0 & 10.56 & 0.30 & 0.76 & 0.21 & 0.33 \\ 0 & 0 & 0.43 & 0.52 & 0.37 & -0.14 \\ 0 & 0 & 0 & 0.49 & 0.08 & -0.06 \\ 0 & 0 & 0 & 0 & 5.9 \cdot 10^{-3} & 7.3 \cdot 10^{-4} \\ 0 & 0 & 0 & 0 & 0 & 1.6 \cdot 10^{-3} \end{pmatrix},$$

both revealing the numerical rank of A. The ULV subspace distances and their upper bounds from Theorem 1 are

$$\operatorname{dist}(\mathcal{R}(U_k), \mathcal{R}(U_{Lk})) = 8.50 \cdot 10^{-4}, \qquad \frac{\sigma_{\min}(L_k)\,\|H\|_2}{\sigma_{\min}(L_k)^2 - \|E\|_2^2} = 8.99 \cdot 10^{-4}$$

$$\operatorname{dist}(\mathcal{R}(V_0), \mathcal{R}(V_{L0})) = 2.10 \cdot 10^{-5}, \qquad \frac{\|H\|_2\,\|E\|_2}{\sigma_{\min}(L_k)^2 - \|E\|_2^2} = 2.25 \cdot 10^{-5}$$

illustrating that for the rank-revealing ULV decomposition, the null space estimate is indeed more accurate than the estimate of the numerical range. Finally, turning to the RRQR decomposition, the quantities Q_1 and W provide approximate bases for the numerical range and null space with

$$\operatorname{dist}(\mathcal{R}(U_k), \mathcal{R}(Q_1)) = 5.29 \cdot 10^{-3}, \qquad \operatorname{dist}(\mathcal{R}(V_0), \mathcal{R}(W)) = 6.78 \cdot 10^{-4},$$

and we see that both approximations are poorer than those from the UTV decompositions.

3. UTV Algorithms

Turning to algorithms for computing UTV decompositions, experience shows that it is natural to distinguish between high-rank algorithms for the case $k \approx n$, and low-rank algorithms for the case $k \ll n$. So far, no efficient algorithm has been developed for computing a rank-revealing orthogonal decomposition of a matrix whose numerical rank is approximately half the number of rows or columns.

We concentrate on the ULV algorithms; the URV algorithms are very similar, and thus we omit a discussion of these algorithms. We assume that the reader is familiar with standard "building blocks" of numerical linear algebra such as orthogonal transformations and condition estimation. Finally, we mention that all of our algorithms are designed for the case $m \geq n$, and if the left orthogonal matrix is required then we always compute the "skinny" part, i.e., U_1.

3.1. High-Rank Algorithms

Many applications give rise to high-rank matrices A where $k \approx n$. One example is direction-of-arrival estimation in signal processing, where k corresponds to the number of incoming signals which is usually comparable to the number of sensors n. Another example is discretizations of certain deconvolution problems (Fredholm integral equations of the first kind), in which the integral operator has a null space of small dimension.

For such high-rank matrices, Stewart introduced the rank-revealing URV and ULV decompositions and algorithms [52], [53] as alternatives to the SVD. In these algorithms, the rectangular matrix A is preprocessed by a standard orthogonal triangular factorization, in which the "skinny" form is computed if only U_1 is required. This factorization can take advantage of the structure of A, such as Toeplitz structure (although the feature is not implemented in our package). Then condition estimation, plane rotations from the left and right, and deflation steps are used to achieve the rank-revealing form. Here, by condition estimation we mean estimation of the smallest singular value of a matrix and the corresponding left or right singular vector.

Stewart's high-rank algorithms "peel off" the small singular values of A one at a time, starting with the smallest. In each step, the estimated singular vector is used to generate Givens rotations which, when applied to A, produce the desired rank-revealing triangular form. If τ denotes the threshold used in determining the numerical rank, then the generic high-rank ULV algorithm can be summarized as follows for the case $m \geq n$:

Generic High-Rank ULV Algorithm (Stewart).

1. Let $k \leftarrow n$ and compute an initial factorization $A = U_1 L$ with a lower triangular L.
2. Condition estimation: let $\widetilde{\sigma}_k$ estimate $\sigma_{\min}(L(1\colon k, 1\colon k))$ and let w_k estimate the corresponding left singular vector.
3. If $\widetilde{\sigma}_k > \tau$ then **exit**.
4. Revealment: determine an orthogonal P_k such that $P_k w_k = (0, \ldots, 0, 1)^T$;
5. update $L(1\colon k, 1\colon k) \leftarrow P_k^T L(1\colon k, 1\colon k)$;
6. update $L(1\colon k, 1\colon k) \leftarrow L(1\colon k, 1\colon k) Q_k$, where the orthogonal matrix Q_k is chosen such that the updated L is triangular;
7. Refinement (optional): while $\|L(k, 1\colon k-1)\|_2 > \delta \, \|L\|_F$ apply QR-refinement to the bottom row of $L(1\colon k, 1\colon k)$.
8. Deflation: let $k \leftarrow k - 1$.
9. Go to step 2.

The three phases that we — for clarity — call revealment (steps 4–6), refinement (step 7), and deflation (step 8) are usually referred to collectively as "refinement." Before step 8, a small singular value has revealed itself in the form of small elements in absolute value in the bottom row of $L(1\colon k, 1\colon k)$.

Steps 5 and 6 consist of interleaved left and right Givens transformations applied in such a way that intermediate fill-in is restricted to the upper bidiagonal of L. The left and right transformations are accumulated into U_1 and I_n in order to compute the two final orthogonal matrices U_1 and V. This approach is efficient for high-rank matrices with $k \approx n$, because the smallest $n-k$ singular values are guaranteed to emerge first, one per deflation step, and thus the algorithm terminates after $n-k$ deflation steps when only large singular values remain.

The optional QR-refinement in step 7, which can be used to reduce and control the norm of the off-diagonal block, is explained in Section 3.3 (the iteration is, of course, safeguarded by allowing only a small number of refinement steps for each k). If refinement is used, then upon completion of the ULV algorithm we can guarantee that $\|(H\,,\,E)\|_F \leq \sqrt{n-k}\,\delta\,\|L\|_F$.

The condition estimation in step 2 can be implemented in various ways, and there are many algorithms available for triangular matrices, cf. the survey [32]. The algorithm we have chosen was designed by Cline, Conn, and Van Loan [13] to be consistent with the 2-norm, and it is implemented in the Matlab function ccvl. The complete high-rank algorithms are implemented in the two Matlab functions hulv and hurv for computing ULV and URV decompositions, respectively.

There is an intimate and subtle relationship between the accuracy of the condition estimator and the norm of the off-diagonal block H or F, cf. the study in [20]. The main conclusion is that an accurate condition estimator will produce an off-diagonal block with small norm. This fact has inspired us to develop alternative high-rank UTV algorithms, implemented in Matlab functions hulv_a and hurv_a, in which the condition estimation and revealment steps can be repeated — for each value of k — until the norm of the current off-diagonal block is sufficiently small. Thus, we have replaced QR-refinement with refinement of the condition estimation. The condition estimation used in these algorithms is inverse iteration (implemented in the Matlab function inviter), which allows us — for a fixed k — to restart the iterations in a simple way. The use of inverse iterations has also been suggested by Yoon and Barlow in connection with a ULV downdating algorithm [65]. The alternative ULV algorithm takes the following form for $m \geq n$.

Generic High-Rank ULV Algorithm (Alternative Version).

1. Let $k \leftarrow n$ and compute an initial factorization $A = U_1 L$ with a lower triangular L.
2. Let $w_k^o \leftarrow k^{-1/2} \cdot \mathsf{ones}(k, 1)$.
3. Condition estimation: let $(\widetilde{\sigma}_k, w_k) \leftarrow \mathsf{inviter}(L(1\colon k, 1\colon k), w_k^o)$.
4. If $\widetilde{\sigma}_k > \tau$ then **exit**.
5. Revealment — similar to steps 4–6 in Stewart's algorithm.
6. Refinement (optional): if $\|L(k, 1\colon k-1)\|_2 > \delta \, \|L\|_F$
7. let $w_k^o \leftarrow (0, \ldots, 0, 1)^T$,
8. go to step 3.
9. Deflate: let $k \leftarrow k - 1$.
10. Go to step 2.

Here, ones is Matlab's built-in function for generating a vector of all ones, and the revealment step 4 is identical to steps 4–6 in Stewart's algorithm. The notation used in step 3 means that the estimates $\widetilde{\sigma}_k$ and w_k are computed by means of inverse iteration with starting vector w_k^o. We use a fixed starting vector in order to ensure that the decomposition can be reproduced. Ideally, we would like to stop the inverse iterations when the estimate w_k ensures that the norm $\|L(k, 1\colon k-1)\|_2$ — after the revealment step — is sufficiently small. Unfortunately, the only available a priori bound is $\|L(k, 1\colon k-1)\|_2 < \widetilde{\sigma}_k$, which is too crude. Hence, when refinement is used it is necessary to perform alternating condition-estimation and revealment steps.

3.2. Low-Rank Algorithms

Certain applications give rise to low-rank matrices A in which $k \ll n$. For instance, low-rank matrices arise in information retrieval using latent semantic indexing (LSI) [4], where the elements of the $m \times n$ matrix A provide an incomplete connection between n documents which define the database, and m key words pertaining to the database. The parameter k is typically 0.1% of $\min(m, n)$, thus relatively few factors are adequate for the LSI approach.

Low-rank problems are traditionally solved using SVD-based techniques, and more details can be found in [61]. Low-rank UTV algorithms [22] are computationally attractive alternatives to the SVD because they provide enough important information, but more efficiently than the SVD.

Our generic low-rank UTV algorithms are very similar to the alternative version of the high-rank algorithm. They "peel off" the large singular values of A one at a time, starting with the largest, and in each step the estimated singular vector is used to generate Givens rotations which, when applied to A, produce the desired rank-revealing triangular form. The low-rank revealment step differs from the high-rank version in that the permutation matrix P_k is chosen such that $P_k w_k = (1, 0, \ldots, 0)$. If again τ denotes the threshold used in determining the numerical rank, the generic ULV algorithm can be summarized as follows for $m \geq n$:

Generic Low-Rank ULV Algorithm.

1. Let $k \leftarrow 1$ and compute an initial factorization $A = U_1 L$ with a lower triangular L.
2. Let $w_k^o \leftarrow (n-k+1)^{-1/2} \cdot \mathsf{ones}(n-k+1,1)$ and $\ell = 1$.
3. Norm estimation: $(\widetilde{\sigma}_k, w_k) \leftarrow \mathsf{normest}(L(k\colon n, k\colon n), w_k^o)$.
4. If $\widetilde{\sigma}_k > \tau$ then **exit**.
5. Revealment — as explained in the text above.
6. Refinement (optional): if $\ell < \ell_{\max}$
7. let $w_k^o \leftarrow (1,0,\ldots,0)^T$ and $\ell = \ell + 1$,
8. go to step 3.
9. Deflate: let $k \leftarrow k+1$.
10. Go to step 2.

The notation in step 3 means that the estimates $\widetilde{\sigma}_k$ and w_k are computed by means of a 2-norm estimator with starting vector w_k^o. We say that this algorithm is "warm started" because of the initial triangular factorization, and again the left and right orthogonal transformations are accumulated into U_1 and I_n, respectively, to produce the final U_1 and V.

In the low-rank algorithms, the condition estimation of the high-rank algorithms is replaced by estimation of the largest singular value (which is identical to the 2-norm) and a corresponding singular vector. This can be accomplished by means of the classical power method [26, §8.2] or by means of Lanczos bidiagonalization [26, §9.2]. The number of iterations in both methods depends on the gap between the largest and the second largest singular values in the current submatrix, and often the Lanczos method is faster than the power method; see [22] for more details.

The most important difference between the high- and low-rank algorithms is that in the low-rank algorithm, the norm of the current off-diagonal block (i.e., $L(k+1\colon n, 1\colon k)$ in the ULV algorithm) does not become small until k reaches its final value — the details are discussed in [22]. Hence, we cannot use the norm of the current off-diagonal block to control the refinement process, so this process is controlled only by the maximum number $\ell_{\max}$ of refinement steps allowed for each k.

The two warm-started low-rank Matlab functions included in this package are **lulv** and **lurv** for computing low-rank ULV and URV decompositions, respectively, and the underlying power and Lanczos methods are implemented in the Matlab functions **powiter** and **lanczos**. From a probabilistic point of view, random starting vectors for the iterative singular value estimators are superior to fixed vectors [34], but still we have chosen to use a fixed starting vector because this ensures that the computed decompositions can be reproduced.

We remark that several Lanczos-based algorithms have been suggested for computing good estimates of the low-dimensional signal subspaces associated with various problems [14], [18], [63], [64]. None of these algorithms produce a UTV decomposition, only approximations to the desired subspaces — whether this is a drawback depends on the particular application.

When estimating the largest singular value of a matrix, there is no particular need for working with a triangular matrix (which, on the other hand, is essential when estimating the smallest singular value efficiently in the high-rank algorithms). This lead to the definition of an alternative form of the UTV decomposition in which the initial triangularization is omitted, and the final matrix middle matrix is block triangular with a square $(m-k) \times (n-k)$ bottom right submatrix. This version, and the corresponding "cold started" algorithm, is described in [22], and the corresponding Householder-based low-rank algorithms are implemented in the Matlab functions lulv_a and lurv_a.

3.3. Refinement Techniques

Once a UTV decomposition has been computed, one may want to improve the accuracy of the estimated singular subspaces, represented by the columns of the matrices U and V. It follows from Theorem 1 that this can be achieved by reducing the norm of the off-diagonal block, i.e., either F in the URV decomposition, or H in the ULV decomposition.

This can be accomplished by a block QR iteration applied to either R or L, as described in [41]. Consider the URV decomposition. In the first step, right Givens rotations are constructed such that F is annihilated and the (2,1)-block fills out. In the second step, left Givens rotations are applied to the updated matrix in order to annihilate the (2,1)-block again and thus restore the upper triangular form. If these two steps are repeated, then it is proved in [41] that the norm of the off-diagonal block converges linearly to zero, with a factor equal to $\|G\|_2/\sigma_{\min}(R_k)$ for the URV decomposition, and $\|E\|_2/\sigma_{\min}(L_k)$ for the ULV decomposition. These "post processing" refinement operations are implemented in the two Matlab functions ulv_qrit and urv_qrit.

Refinement can also be applied in each step of the UTV algorithms, as shown in Stewart's high-rank algorithm above, by "flipping" the last row of the current $L(1\colon k, 1\colon k)$ — or the last column of $R(1\colon k, 1\colon k)$ — as in the block QR iteration. Again, it follows from the theory in [41] that refinement of a single row or columns of the current triangular matrix will reduce the norm of the off-diagonal block. In the alternative high-rank algorithm, as well as in the low-rank algorithms, the QR-refinement steps are replaced by repeated restarts of the condition or norm estimator followed by revealment. The initial guess in the restart is chosen so that one continues the iterations in order to further improve the norm estimate, thereby resulting in a reduction of the off-diagonal block's norm.

A different flavor of refinement is used in Stewart's PLQ decomposition [56], where a pivoted QR factorization is followed by an orthogonal reduction to lower triangular form, which can be considered as "half a QR iteration". The lower triangular matrix produced in this way is sometimes quite good at revealing gaps in the singular value spectrum, but without the theoretical justification underlying the UTV decompositions.

3.4. Numerical Examples

We will first show the influence of the number of power iterations used to estimate the largest singular value in the low-rank ULV algorithm. We generate a low-rank 8×6 matrix A with singular values 0.3, 0.2, 0.05, 0.03, 0.02, and 0.01, and the numerical rank of A with respect to the threshold $\tau = 0.1$ is $k = 2$. Then we use the Matlab function lulv to compute ULV decompositions of A using a fixed number of 1, 2, 3, and 4 power iterations in each stage of the algorithm, and the results are shown in Table 3.1 where we use the notation

$$d(U_k) = \mathrm{dist}(\mathcal{R}(U_k), \mathcal{R}(U_{Rk})), \qquad d(V_0) = \mathrm{dist}(\mathcal{R}(V_0), \mathcal{R}(V_{R0})).$$

Power iterations	$\|H\|_2$	$d(U_k)$	$d(V_0)$
1	$3.4 \cdot 10^{-2}$	$1.6 \cdot 10^{-1}$	$3.7 \cdot 10^{-2}$
2	$1.8 \cdot 10^{-3}$	$9.2 \cdot 10^{-3}$	$2.3 \cdot 10^{-3}$
3	$1.1 \cdot 10^{-4}$	$5.6 \cdot 10^{-4}$	$1.4 \cdot 10^{-4}$
4	$6.5 \cdot 10^{-6}$	$3.4 \cdot 10^{-5}$	$8.6 \cdot 10^{-6}$

Table 3.1: Results from the low-rank ULV algorithm with the call lulv(A,0.1,power_its).

Refinement steps	$\|H\|_2$	$d(U_k)$	$d(V_0)$
0	$3.4 \cdot 10^{-2}$	$1.6 \cdot 10^{-1}$	$3.7 \cdot 10^{-2}$
1	$2.0 \cdot 10^{-3}$	$1.1 \cdot 10^{-2}$	$2.6 \cdot 10^{-3}$
2	$5.1 \cdot 10^{-5}$	$2.7 \cdot 10^{-4}$	$6.8 \cdot 10^{-5}$
3	$3.0 \cdot 10^{-6}$	$1.6 \cdot 10^{-5}$	$4.0 \cdot 10^{-6}$
4	$1.8 \cdot 10^{-7}$	$9.7 \cdot 10^{-7}$	$2.4 \cdot 10^{-7}$

Table 3.2: Results from the low-rank ULV algorithm with the call lulv(A,0.1,1,ref_steps).

First of all, we see that the norm of the off-diagonal block H decreases as the number of power iterations increases, reflecting the fact that the more accurate the singular vector estimate, the closer the triangular matrix is to block diagonal form, and thus the more accurate the subspace estimates. Moreover, as expected from Theorem 1, we see that the approximate null space bases are always more accurate than the bases for the numerical range.

Another way to achieve accurate subspace estimates is to perform one or more refinement steps in each stage of the algorithm. To illustrate this, we use lulv again with one power iteration followed by refinement in the form of 0, 1, 2, 3, or 4 refinement steps in each stage. The results of this experiment are shown in Table 3.2, and it is no surprise that the results improve as the number of refinement steps increases.

The third way to improve the accuracy of the UTV subspaces is to perform block QR iterations on the final triangular matrix L, and we illustrate this by applying block iterations (using the function ulv_qrit) to the matrix L computed with one power iteration and no refinement in each stage; cf. Table 3.3. Clearly, the block QR iterations reduce the off-diagonal block's norm and improve the subspace estimates. The same conclusions hold for the high-rank algorithms; we do not show the results here.

Block iterations	$\|H\|_2$	$d(U_k)$	$d(V_0)$
0	$3.4 \cdot 10^{-2}$	$1.6 \cdot 10^{-1}$	$3.7 \cdot 10^{-2}$
1	$1.7 \cdot 10^{-3}$	$8.9 \cdot 10^{-3}$	$2.2 \cdot 10^{-3}$
2	$1.0 \cdot 10^{-4}$	$5.4 \cdot 10^{-4}$	$1.3 \cdot 10^{-4}$
3	$6.3 \cdot 10^{-6}$	$3.4 \cdot 10^{-5}$	$8.4 \cdot 10^{-6}$
4	$3.9 \cdot 10^{-7}$	$2.1 \cdot 10^{-6}$	$5.2 \cdot 10^{-7}$

Table 3.3: Results from the ULV refinement algorithm using block QR iterations, with the call ulv_qrit(k,block_its,L,V,U).

4. Up- and Downdating

One of the most important properties of the UTV decompositions is their ability to be updated and downdated efficiently and stably. Here we briefly summarize the algorithms used in the package. It should be noted that none of the algorithms described below apply to the block triangular low-rank UTV decomposition computed by the cold-started algorithms mentioned in Section 3.2.

4.1. Updating

Consider first updating the UTV decomposition with an additional row w^T and with a positive weighting factor $\beta \le 1$ applied to A (which is a standard operation in signal processing), i.e., given the UTV decomposition $A = U_1 T V^T$, where T is either L or R, we want to compute the new UTV decomposition

$$\begin{pmatrix} \beta A \\ w^T \end{pmatrix} = \overline{U}_1 \overline{T}\, \overline{V}^T.$$

The updating is accomplished by promoting w^T to the middle matrix,

$$\begin{pmatrix} \beta A \\ w^T \end{pmatrix} = \begin{pmatrix} U_1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} T \\ w^T V \end{pmatrix} V^T,$$

and then left and right Givens rotations G and H are used to annihilate the elements of $w^T V$. Thus we obtain

$$\begin{pmatrix} \beta A \\ w^T \end{pmatrix} = \begin{pmatrix} U_1 & 0 \\ 0 & 1 \end{pmatrix} G \begin{pmatrix} \overline{T} \\ 0 \end{pmatrix} H^T V^T = \begin{pmatrix} \overline{U}_1 & u_2 \end{pmatrix} \begin{pmatrix} \overline{T} \\ 0 \end{pmatrix} \overline{V}^T = \overline{U}_1 \overline{T}\, \overline{V}^T,$$

i.e., $\overline{V} = V$. Notice that V is always needed in order to accomplish the updating, while U_1 is not required.

The order in which the Givens rotations are applied is important because we wish, as far as possible, to maintain the small elements present in the triangular matrix T. Using the scheme proposed by Stewart [52] together with the fact that if $\beta = 1$ then the numerical rank can either remain at k or increase by one, we merely have to apply one step of condition estimation and at most one deflation step in this case. See [26, §12.5.5] or [52] for more details.

The updating algorithm is implemented in the Matlab functions `ulv_up` and `urv_up`, and after all the Givens rotations have been applied, we normalize the columns of $\overline{V}$ as recommend in [43]. In these implementations, the condition estimation is accomplished by means of `ccvl`, and we allow refinement of the updated triangular matrix.

Moonen et al. [44] presented a related updating algorithm in which the updating step is followed by one — or a few — sweeps of Kogbetliantz's iterative SVD algorithm. The result of this post-processing is that if the initial middle matrix T is close to diagonal (e.g., if the initial decomposition is the SVD), then the norm of the off-diagonal part of T stays relatively small after each updating step. We have not implemented this variant in our package.

4.2. Downdating

The downdating problem is the following: given the UTV decomposition $A = U_1 T V^T$, where T is either L or R, compute the new UTV decomposition of the $(m-1)\times n$ submatrix $A(2{:}\, m, :)$, i.e.,

$$A = \begin{pmatrix} w^T \\ A(2{:}\, m, :) \end{pmatrix}, \qquad A(2{:}\, m, :) = \overline{U}_1 \, \overline{T} \, \overline{V}^T .$$

Downdating is a more complicated problem than updating, and the algorithm depends on whether the matrix U_1 is explicitly available, because its first row is required in the downdating algorithm.

If U_1 is available, then the first step is to augment U_1 with one additional column u_2 that is orthogonal to the columns of U_1, i.e., $U_1^T u_2 = 0$, in such a way that the norm of the first row of the matrix $(U_1,\, u_2)$ is one, and this can almost always be achieved by orthonormalizing the unit vector $e_1 = (1, 0, \ldots, 0)^T$ to U_1 by means of the modified Gram-Schmidt (MGS) process [15]. At this stage, the UTV decomposition of A can be reformulated as

$$A = (U_1,\, u_2) \begin{pmatrix} T \\ 0^T \end{pmatrix} V^T .$$

If $T = R$, then we use the standard algorithm for downdating a QR factorization, cf. [26, §12.5.3]. We apply a sequence of right Givens rotations G that annihilate all but the first element of the first row of (U_1, u_2), starting from the right, and these rotations are also applied from the left to the middle matrix:

$$(U_1,\, u_2)\, G = \begin{pmatrix} \pm 1 & 0^T \\ 0 & \overline{U} \end{pmatrix} \qquad \text{and} \qquad G^T \begin{pmatrix} R \\ 0^T \end{pmatrix} = \begin{pmatrix} y^T \\ \overline{R} \end{pmatrix},$$

where the $(m-1)\times n$ matrix $\overline{U}$ has orthonormal columns. Then it follows immediately that the three matrices $\overline{U}$, $\overline{R}$, and V constitute a URV decomposition of $A(2{:}\, m, :)$.

If $T = L$, then we use the algorithm from [8]. First we annihilate all but the first element of U_1, again starting from the right, and when these rotations G are applied from the left to L then they must be interleaved with right rotations H that restore the triangular form:

$$U_1 G = \widetilde{U} = \begin{pmatrix} v\, e_1^T \\ \widetilde{U}(2{:}\, m, :) \end{pmatrix}, \qquad G^T L\, H = \widetilde{L}, \qquad \widetilde{V} = V\, H,$$

where $|v| \le 1$. We finish by a single Givens rotation $\widetilde{G}$ involving u_2 and the first column of $\widetilde{U}$ in which $v = \widetilde{U}(1,1)$ is annihilated, and when $\widetilde{G}$ is applied from the left to the middle matrix, it introduces a single fill-in in the bottom row:

$$\left(\widetilde{U},\, u_2\right) \widetilde{G} = \begin{pmatrix} 0^T & \pm 1 \\ \overline{U} & 0 \end{pmatrix} \qquad \text{and} \qquad \widetilde{G}^T \begin{pmatrix} \widetilde{L} \\ 0^T \end{pmatrix} = \begin{pmatrix} \overline{L} \\ \alpha\, e_1^T \end{pmatrix}.$$

Then the ULV decomposition of $A(2{:}\,m,:)$ consists of $\overline{U}$, $\overline{L}$, and $\widetilde{V}$.

Finally, we need to make the new UTV decomposition a rank-revealing one. We note that the numerical rank can either remain at k or decrease by one. Hence, due to the ordering of the Givens rotations, most of the small elements in T remain small, and we need only perform a few condition estimation and deflation steps.

Consider now the case where U_1 is not available, which is common in signal processing applications due to constraints in both memory size and computational complexity. The vector $q^T = U_1(1,:)$ can be computed from the relation $w^T = q^T T V^T$ by solving this equation for q, i.e.

$$q = \left(T^T\right)^{-1} V^T w, \tag{4.1}$$

and the first element of u_2 is then given by $u_2(1) = (1 - \|q\|_2^2)^{1/2}$. Once this information is available, it follows from the relations above that the downdating can be accomplished. Unfortunately, this so-called Linpack procedure is numerically inferior when $\|q\|_2$ is close to one, in which case it is more safe to use algorithms based on the corrected semi-normal equation (CSNE) approach [7]. Two versions of this approach have been developed: the first by Bojanczyk and Lebak [8], and the second by Park and Eldén [45] and further improved by Barlow, Yoon, and Zha [3]. It is outside the scope of this work to present the details of these sophisticated downdating algorithms; instead we refer to the papers for details.

The process of orthogonalizing e_1 to U_1 breaks down when e_1 lies in the range of U_1, and one instance where this happens is when the exact rank of the coefficient matrix decreases during the downdating process, as shown in the following theorem.

Theorem 2. *Let* $q^T = U_1(1,:)$. *If* $\mathrm{rank}(A(2{:}\,m,:)) < \mathrm{rank}(A)$, *then*

$$e_1 \in \mathcal{R}(U_1) \qquad \Longleftrightarrow \qquad \|q\|_2 = 1. \tag{4.2}$$

Proof. Since $\mathrm{rank}(T) = \mathrm{rank}(A)$, and since

$$\mathrm{rank}(U_1(2{:}\,m,:)\,T) = \mathrm{rank}(A(2{:}\,m,:)) = \mathrm{rank}(A) - 1,$$

we conclude that $U_1(2{:}\,m,:)$ must be rank deficient. Now, from the CS decomposition [26, §2.6.4] of U_1 it is clear that $\|q\|_2 = 1 \Leftrightarrow \mathrm{rank}(U_1(2{:}\,m,:)\,T) = n-1$. Hence, $\mathrm{rank}(A(2{:}\,m,:)) < \mathrm{rank}(A)$ implies that $\|q\|_2 = 1$. This, in turn, is equivalent to $e_1 \in \mathcal{R}(U_1)$ because $\|U_1^T e_1\|_2 = \|q\|_2$ is the norm of the orthogonal projection of e_1 on the range of U_1.

Our algorithms detect and overcome this difficulty as follows. If U_1 is available, the situation is detected reliably by the "twice is enough" strategy in MGS, and instead we orthonormalize the vector $(1, 2, \ldots, m)^T$ to U_1 which yields a vector u_2 whose first component is of the order of the machine precision. If U_1 is not available, the situation is detected during the CSNE algorithm which returns an exact zero for $u_2(1)$.

At this stage, we want to emphasize that numerically stable UTV downdating algorithms have become very complex, and the computational overhead can become quite large, especially when the exact rank decreases. In certain real-time applications where the complexity of the software is limited, it may be worth to consider whether recomputation of the ULV decomposition — which simplifies the code at the expense of introducing a time delay or a gap in the output data — is to be preferred to the more complex algorithms. The decision to recompute the UTV decomposition should then be linked to the detection of the situation when e_1 on the range of U_1.

The downdating algorithms described above are implemented in the two Matlab functions ulv_dw and urv_dw for downdating the ULV and URV decompositions, respectively. Similar to the updating implementations, we normalize the columns of U_1 and V once all Givens rotations have been applied.

Downdating frequently arises in signal processing in connection with sliding window applications where, in each time step, the top row of A is skipped and a new row is appended to the bottom of A. Algorithmically, this is treated by means of an updating step followed by a downdating step, and this combined action is implemented in the two Matlab functions ulv_win and urv_win.

4.3. A Numerical Example

The subspace tracking capability of the up- and downdating algorithms is demonstrated in the Matlab demo functions wulvdemo and wurvdemo. Here, we illustrate some of the inherent numerical difficulties associated with a ULV downdating step as implemented in ulv_dw. Consider first the case where U_1 is available, and let A be a 6×4 matrix such that svd(A)=(2.08, 1.03, 0.21, $1.24 \cdot 10^{-16}$), svd$(A(2\colon m, :)) = (2,\ 1,\ 1.96 \cdot 10^{-16},\ 8.11 \cdot 10^{-17})$, and svd$(U_1(2\colon m, :)) = (1,\ 1,\ 1,\ 2.41 \cdot 10^{-16})$. First we use MGS with one reorthogonalization step to orthogonalize $z = e_1$ to U_1, by means of the generic algorithm:

Generic MGS Algorithm.

1. For $j = 1\colon n$, $z \leftarrow z - U_1(:,j)\,\big(U_1(:,j)^T z\big)$; end
2. If $\|z\|_2 < 2^{-1/2}$ then
3. for $j = 1\colon n$, $z \leftarrow z - U_1(:,j)\,\big(U_1(:,j)^T z\big)$; end

We obtain $\|z\|_2 = 2.32 \cdot 10^{-16}$, and we conclude that $e_1 \in \mathcal{R}(U_1)$. Instead, we choose $z = (1, 2, \ldots, m)^T$ and apply the above MGS algorithm to this vector. After normalization we end up with a vector u_2 whose first component, as expected, is practically zero, $u_2(1) = -1.34 \cdot 10^{-17}$. Then the downdating process can be completed.

Next, consider the situation where U_1 is not available, and where q must be computed from w via (4.1). With the same data as above, the Linpack procedure yields

$$q = (-0.968,\ -0.200,\ 0.153,\ -4.40 \cdot 10^{-16})$$

with $\|q\|_2 = 1$ exactly, and thus $u_2(1) = 0$. Hence, in this exactly rank deficient case, there is in principle no problem in using the simple Linpack approach. However, in finite precision arithmetic we cannot distinguish an exactly rank-deficient problem from a near-rank-deficient problem. To illustrate this, we modify the matrix slightly, such the small singular values of A and $A(2\colon m, :)$ become somewhat larger than the machine precision ($8.49 \cdot 10^{-7}$ and 10^{-6}, 10^{-9}, respectively); now $\|q\|_2 \neq 1$, but it is so close to one that $u_2(1)$ cannot be computed as $(1 - \|q\|_2^2)^{1/2}$. Thus, we must switch to the CSNE approach, which leads to the result $u_2(1) = 5.36 \cdot 10^{-9}$.

5. Quotient UTV Decompositions

Throughout the years, rank-revealing orthogonal decompositions — and in particular the SVD — have been generalized to matrix pairs, triplets, etc. One of the most well known and most frequently used generalizations is the quotient SVD (QSVD), or generalized SVD, of two matrices A and B with the same number of columns [26, §8.7.3], which yields information about the numerical rank and numerical subspaces of the matrix "quotient" $A\,B^{-1}$ when B is invertible, and $A\,B^{\dagger}$ (where $B^{\dagger}$ is the pseudoinverse [26, §5.5.4] of B) when B has full column rank. The QSVD has numerous applications in signal processing as well as in many other applications, cf. [16], [29], and is available in Matlab 5.2 as function `gsvd`.

5.1. The Rank-Revealing ULLV Decomposition

In signal processing applications, the QSVD is often used in connection with problems that involve additive colored noise, where the matrix A represents the sampled noisy signal, while the matrix B represents the noise. As long a A and B have the same number of columns and B has full column rank, the matrix quotient $A\,B^{\dagger}$ represents a so-called prewhitened signal with white noise, to which the standard filtering and noise-reduction techniques can be applied; see [33] for details.

In these applications it is natural to generalize the UTV decomposition to such pairs of matrices with the same number of columns. In this work, we focus on the important case where B has full column rank, which is very often the case in signal processing applications. Then the quotient ULV composition, also referred to as the ULLV decomposition [38], takes the form

$$A = U_A\, L_A\, L\, V^T, \qquad B = U_B\, L\, V^T, \tag{5.1}$$

where U_A and U_B have orthonormal columns and V is orthogonal, i.e., $U_A^T U_A = U_B^T U_B = V^T V = I_n$, while L_A and L are both lower triangular. Moreover, L has full rank, because we assume that B has full column rank. The corresponding quotient URV decomposition is defined analogously, and we shall not pursue this decomposition here.

Since B has full rank, its pseudoinverse is given by $B^{\dagger} = V\,L^{-1} U_B^T$, and thus the matrix quotient can be written in terms of the ULLV decomposition as

$$A\,B^{\dagger} = U_A\, L_A\, U_B^T.$$

We see that the three matrices U_A, L_A, and U_B form the ULV decomposition of $A\,B^{\dagger}$, and this decomposition can always be made to reveal the numerical rank of $A\,B^{\dagger}$ by means of ULV revealment steps. When this is the case, we say that the ULLV decomposition (5.1) is a rank-revealing quotient ULV decomposition of A and B.

For completeness, we mention that if B does not have column full rank, then we can always assume that preprocessing has been applied to the matrix pair such that B is $p \times n$ with $\mathrm{rank}(B) = p < n$. The corresponding quotient ULV decomposition then takes the form

$$A = U_A\, L_A \begin{pmatrix} L & 0 \\ 0 & I_{n-p} \end{pmatrix} V^T, \qquad B = U_B\, (L,\, 0)\, V^T,$$

where L is now $p \times p$; for more details about this version and its application in interference problems, see [30] and [39]. Other generalized UTV decompositions are discussed in [50] (matrix quotients of the form $A^{-1}B$) and [8] (a decomposition of the form $A = U_A\, L_A\, L\, V^T$, $B = U_B\, L_B\, L\, V^T$).

We note that the approximation bounds in Theorem 1 immediately carry over to the numerical subspaces of the matrix quotient $A\,B^\dagger$, when applied to the columns of U_A and U_B and the corresponding vectors of the QSVD.

5.2. ULLV Algorithms

The algorithms for computing and modifying rank-revealing quotient UTV decompositions are similar to those for the ordinary UTV decompositions, except that care must be taken to maintain the triangular structure of both L_A and L. Here, we assume that the matrix B has full rank. We restrict ourselves to a ULLV algorithm for the high-rank case; corresponding low-rank ULLV algorithms can always derived from the low-rank ULV implementations **lulv** and **lulv_a**.

5.2.1. A Simple ULLV Algorithm

To compute a rank-revealing ULLV decomposition in the high-rank case, we need an initial decomposition with the same structure. As long as B has full rank and is well conditioned, we can use the following approach.

Initial ULLV Algorithm.

1. Compute the QL factorization $B = U_B\, L$.
2. Solve $A = Z\, L$ for Z (i.e., formally, $Z = A\, L^{-1}$).
3. Compute the QL factorization $Z = U_A\, L_A$.

Then the condition estimation and deflation steps of Stewart's high-rank UTV algorithm are applied to the three matrices U_A, L_A, and U_B in order to make the ULLV decomposition reveal the numerical rank of $A\,B^\dagger$. Note that some of the Givens rotations in these steps are also be applied from the left to L, and they must be interleaved with right Givens rotations (which are also applied to V) in order to maintain the triangular form of L. The complete algorithm is implemented in the Matlab function **ullv**, where the details of the condition estimation and deflation steps can be studied. We do not provide low-rank algorithms for the ULLV decomposition.

5.2.2. Updating Algorithms

Algorithms for updating the ULLV decomposition (5.1) when a row is appended to A and/or B are described in [38]. Consider the case where a row w^T is appended to A; then we promote this row to L as follows (where a weighting factor β is included for completeness):

$$\begin{pmatrix} \beta\, A \\ w^T \end{pmatrix} = \begin{pmatrix} U_A & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} \beta\, L_A & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} L \\ w^T V \end{pmatrix} V^T, \qquad B = U_B\,(I_n,\,0) \begin{pmatrix} L \\ w^T V \end{pmatrix} V^T.$$

Now left and right Givens rotations are used to annihilate all but the leftmost element of the row $w^T V$ and, simultaneous, maintain the triangular structure of L. The left rotations are also applied from the right to L_A and interleaved with other rotations applied from the left that maintain the triangular form of L_A, and we arrive at the intermediate form

$$\begin{pmatrix} \beta\, A \\ w^T \end{pmatrix} = \begin{pmatrix} \widetilde{U}_A & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} \widetilde{L}_A & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} \widetilde{L} \\ \xi\, e_1^T \end{pmatrix} \widetilde{V}^T, \qquad B = \widetilde{U}_B\,(I_n,\,0) \begin{pmatrix} \widetilde{L} \\ \xi\, e_1^T \end{pmatrix} \widetilde{V}^T.$$

Next, the element ξ is annihilated by means of a scaled rotation Y from the left, satisfying

$$Y \begin{pmatrix} \widetilde{L} \\ \xi\, e_1^T \end{pmatrix} = \begin{pmatrix} \widetilde{L} \\ 0 \end{pmatrix}.$$

The transformation Y is a Givens rotation scaled by c, and it has the form

$$Y = \begin{pmatrix} c^2 & 0^T & cs \\ 0 & I_{n-1} & 0 \\ -cs & 0^T & c^2 \end{pmatrix}, \qquad Y^{-1} = \begin{pmatrix} 1 & 0^T & -s/c \\ 0 & I_{n-1} & 0 \\ s/c & 0^T & 1 \end{pmatrix}. \tag{5.2}$$

When Y^{-1} is propagated to the left it creates fill,

$$\begin{pmatrix} \widetilde{L}_A & 0 \\ 0^T & 1 \end{pmatrix} Y^{-1} = \begin{pmatrix} \widetilde{L}_A & \widetilde{z} \\ \eta\, e_1^T & 1 \end{pmatrix}, \qquad (I_n,\,0)\,Y^{-1} = (I_n,\,-\eta\, e_1),$$

where $\eta = s/c$, and fortunately this fill does not contribute to the updated A or to B because of the newly created zero row. Notice that the scaled rotation maintains the submatrices $\widetilde{L}$, $\widetilde{L}_A$, and I_n. At this second intermediate stage, we have

$$\begin{pmatrix} \beta A \\ w^T \end{pmatrix} = \begin{pmatrix} \widetilde{U}_A & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} \widetilde{L}_A \\ \eta\, e_1^T \end{pmatrix} \widetilde{L}\, \overline{V}^T, \qquad B = \widetilde{U}_B\, \widetilde{L}\, \widetilde{V}^T,$$

and now η is annihilated by means of a single left Givens rotation which creates fill in the last column of the leftmost factor of A that can be neglected:

$$\begin{pmatrix} \widetilde{U}_A & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} \widetilde{L}_A \\ \eta\, e_1^T \end{pmatrix} = (\overline{U}_A,\, \overline{z}) \begin{pmatrix} \overline{L}_A \\ 0 \end{pmatrix} = \overline{U}_A\, \overline{L}_A.$$

Hence, we arrive at

$$\begin{pmatrix} \beta A \\ w^T \end{pmatrix} = \overline{U}_A\, \overline{L}_A\, \widetilde{L}\, \overline{V}^T, \qquad B = \widetilde{U}_B\, \widetilde{L}\, \overline{V}^T.$$

The updating process concludes, as usual, with condition estimation, revealment, and deflation. We refer to [38] for more details as well as a similar algorithm for updating B. The updating algorithms are implemented in the two Matlab functions **ullv_up_a** and **ullv_up_b**, respectively, where further details can be found.

Whenever B is ill conditioned or rank deficient, the initial ULLV algorithm described above must be avoided, and instead one should apply the updating algorithm **ullv_up_a** a number of times to the initial matrix pair 0 and B with $\beta = 1$, in such a way that the rows of A are introduced one at a time, cf. [38].

5.2.3. Downdating Algorithms

Algorithms for downdating the ULLV decomposition (5.1) when a row is removed from A and/or B are described in [31]. These algorithms, in turn, are adapted from the downdating algorithms presented in the unpublished report [36]. They are not as sophisticated as the algorithms in [3] and [45], but more research is necessary to extend the latter algorithms to the ULLV decomposition.

When A is downdated, then the matrix U_A is first augmented with an additional column u_2 that is orthonormal to the columns of U_A in such a way that the norm of the first row of $(U_A,\, u_2)$ has norm one. Then we formally write the ULLV decomposition as

$$A = \begin{pmatrix} w^T \\ A(2\colon m,:) \end{pmatrix} = (U_A,\, u_2) \begin{pmatrix} L_A & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} L \\ 0^T \end{pmatrix} V^T,$$

$$B = U_B\,(I_n,\, -\eta\, e_1) \begin{pmatrix} L \\ 0^T \end{pmatrix} V^T,$$

where η is a parameter to be determined later. Now we annihilate all but the leftmost element of the top row of U_A by means of a sequence of Givens rotations, starting from the right. Applying the necessary Givens rotations in order to maintain the triangular form of L_A and L, we compute

$$A = \begin{pmatrix} v\, e_1^T & u_2(1) \\ \widetilde{U}_A & u_2(2\colon m) \end{pmatrix} \begin{pmatrix} \widetilde{L}_A & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} \widetilde{L} \\ 0^T \end{pmatrix} \widetilde{V}^T, \qquad B = \widetilde{U}_B\,(I_n,\, -\eta\, e_1) \begin{pmatrix} \widetilde{L} \\ 0^T \end{pmatrix} \widetilde{V}^T.$$

Next, we apply a single Givens rotation to the first and last columns of the leftmost matrix in the expressions for A to annihilate v, and we obtain

$$\begin{pmatrix} v\, e_1^T & u_2(1) \\ \widetilde{U}_A & u_2(2\colon m) \end{pmatrix} \begin{pmatrix} \widetilde{L}_A & 0 \\ 0^T & 1 \end{pmatrix} = \begin{pmatrix} 0^T & \pm 1 \\ \widehat{U}_A & 0 \end{pmatrix} \begin{pmatrix} \widehat{L}_A & -s\, e_1 \\ \eta\, e_1^T & c \end{pmatrix}.$$

This relation defines the quantity η used in the augmented expression for B. Finally, we apply the scaled transformation Y (5.2) from the right to the rightmost matrix in the above expression, with c and s determined from the relations $\eta = s/c$ and $c^2 + s^2 = 1$. This transformation annihilates η, and we obtain

$$\begin{pmatrix} \widehat{L}_A & -s\, e_1 \\ \eta\, e_1^T & c \end{pmatrix} \begin{pmatrix} \widetilde{L} \\ 0^T \end{pmatrix} = \begin{pmatrix} \overline{L}_A & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \overline{L} \\ \xi\, e_1^T \end{pmatrix},$$

$$(I_n,\, -\eta\, e_1) \begin{pmatrix} \widetilde{L} \\ 0^T \end{pmatrix} = (I_n,\, 0) \begin{pmatrix} \widetilde{L} \\ \xi\, e_1^T \end{pmatrix}.$$

We thus arrive at the expressions

$$A(2\colon m,:) = \widehat{U}_A\, \overline{L}_A\, \overline{L}\, \widetilde{V}^T, \qquad B = \widetilde{U}_B\, \overline{L}\, \widetilde{V}^T.$$

Finally, condition estimation and deflation steps are applied. We refer to [31] for more details and for a similar algorithm for downdating B. The downdating algorithms are implemented in the two Matlab functions ullv_dw_a and ullv_dw_b, respectively, where details can also be found.

5.3. Numerical Examples

Our first example illustrates the structure of the two triangular matrices L_A and L. Both A and B are random 8×5 matrices; A has numerical rank $k = 3$ with respect to the threshold $\tau = 0.005$, and B is well conditioned with condition number equal to 6.3. Using the function ullv, we compute

$$L_A = \begin{pmatrix} 2.7 \cdot 10^{-2} & 0 & 0 & 0 & 0 \\ -3.2 \cdot 10^{-1} & 2.8 \cdot 10^{-1} & 0 & 0 & 0 \\ 1.1 \cdot 10^{-2} & 5.3 \cdot 10^{-1} & 7.5 \cdot 10^{-1} & 0 & 0 \\ 1.1 \cdot 10^{-4} & 1.3 \cdot 10^{-4} & -9.2 \cdot 10^{-5} & 2.9 \cdot 10^{-3} & 0 \\ -5.4 \cdot 10^{-6} & -6.3 \cdot 10^{-6} & 4.5 \cdot 10^{-6} & -1.6 \cdot 10^{-4} & 1.2 \cdot 10^{-3} \end{pmatrix}$$

and

$$L = \begin{pmatrix} 0.45 & 0 & 0 & 0 & 0 \\ 0.06 & 0.62 & 0 & 0 & 0 \\ -2.4 & 0.38 & 2.5 & 0 & 0 \\ 0.45 & 0.05 & -0.74 & 1.07 & 0 \\ -0.19 & 0.11 & -0.04 & -0.12 & 1.01 \end{pmatrix}.$$

With respect to the same threshold $\tau = 0.005$, the numerical rank of $A\,B^\dagger$ is clearly revealed through L_A as 3.

Our second example illustrates that the numerical rank of the matrix coefficient $A\,B^\dagger$ can differ from that of A, thus showing the need for a quotient ULV decomposition. The matrix A is 8×5 and its singular values are

$$\sigma_1 = 10, \quad \sigma_2 = 7, \quad \sigma_1 = 4, \quad \sigma_1 = 0.4, \quad \sigma_1 = 0.2.$$

There is obviously a cluster of three large singular values, i.e., the rank is three with respect to the threshold $\tau = 1$. The 8×5 matrix B is again well conditioned with $\|B\|_2 = 23.1$ and condition number equal to 58.9. The five singular values of the matrix quotient $A\,B^\dagger$ are

$$\sigma_1 = 10, \quad \sigma_2 = 5, \quad \sigma_1 = 0.5, \quad \sigma_1 = 0.2, \quad \sigma_1 = 0.1,$$

showing that the gap in the singular value spectrum has changed; the matrix quotient has a cluster of only two large singular values, i.e., the numerical rank is now two with respect to the same threshold $\tau = 1$. These two different numerical ranks are estimated correctly by the ULV decomposition of A and the ULLV decomposition of (A, B). By means of hulv we compute A's lower triangular factor in the ULV decomposition,

$$\begin{pmatrix} 5.14 & 0 & 0 & 0 & 0 \\ 4.98 & 8.09 & 0 & 0 & 0 \\ 0.87 & -1.48 & 6.73 & 0 & 0 \\ -9.5 \cdot 10^{-4} & 5.1 \cdot 10^{-4} & 9.7 \cdot 10^{-4} & 0.40 & 0 \\ -3.7 \cdot 10^{-4} & 2.2 \cdot 10^{-4} & 1.7 \cdot 10^{-4} & 3.2 \cdot 10^{-2} & 0.20 \end{pmatrix},$$

and by means of **ullv** we compute the following L_A-factor in the ULLV decomposition:

$$L_A = \begin{pmatrix} 5.13 & 0 & 0 & 0 & 0 \\ -1.94 & 9.74 & 0 & 0 & 0 \\ -1.2 \cdot 10^{-3} & 4.1 \cdot 10^{-3} & 0.50 & 0 & 0 \\ 1.9 \cdot 10^{-5} & 2.3 \cdot 10^{-4} & 1.0 \cdot 10^{-2} & 0.19 & 0 \\ -8.6 \cdot 10^{-5} & 3.7 \cdot 10^{-5} & 2.6 \cdot 10^{-3} & -4.3 \cdot 10^{-2} & 0.10 \end{pmatrix}.$$

Both triangular factors reveal the correct numerical rank of A and $A\,B^{\dagger}$, respectively.

6. Manual Pages

This section starts with nine tables that give an overview of the software in the package, followed by manual pages for all 46 functions.

Demo Functions	
hulvdemo	Demonstrates the use of the high-rank ULV algorithms hulv and hulv_a.
hurvdemo	Demonstrates the use of the high-rank URV algorithms hurv and hurv_a.
lulvdemo	Demonstrates the use of the low-rank ULV algorithms lulv and lulv_a.
lurvdemo	Demonstrates the use of the low-rank URV algorithms lurv and lurv_a.
rrqrdemo	Demonstrates the use of the RRQR algorithms hrrqr and lrrqr.
ullvdemo	Demonstrates the use of the high-rank ULLV algorithm ullv.
wulvdemo	Demonstrates the use of the ULV up- and downdating algorithms, implemented in ulv_win, applied to a sliding window example.
wurvdemo	Demonstrates the use of the URV up- and downdating algorithms, implemented in urv_win, applied to a sliding window example.

UTV-Based Solvers	
tulv	Solves a numerically rank-deficient least squares problem using the rank-revealing ULV decomposition.
turv	Similar to tulv, except it uses the rank-revealing URV decomposition.

High-Rank UTV Algorithms	
hulv	Stewart's rank-revealing ULV algorithm.
hulv_a	The alternative rank-revealing ULV algorithm.
hurv	Stewart's rank-revealing URV algorithm.
hurv_a	The alternative rank-revealing URV algorithm.

Low-Rank UTV Algorithms	
lulv	Warm-started rank-revealing ULV algorithm.
lurv	Warm-started rank-revealing URV algorithm.
lulv_a	Cold-started rank-revealing ULV algorithm.
lurv_a	Cold-started rank-revealing URV algorithm.

Block QR Refinement	
ulv_qrit	Refinement of L in the ULV decomposition.
urv_qrit	Refinement of R in the URV decomposition.

UTV Up- and Downdating	
ulv_dw	Downdate the rank-revealing ULV decomposition.
ulv_up	Update the rank-revealing ULV decomposition.
ulv_win	Sliding window modification of the rank-revealing ULV decomposition.
urv_dw	Downdate the rank-revealing URV decomposition.
urv_up	Update the rank-revealing URV decomposition.
urv_win	Sliding window modification of the rank-revealing URV decomposition.

ULLV Algorithms	
ullv	Compute a high-rank revealing ULLV decomposition.
ullv_dw_a	Downdate A in the rank-revealing ULLV decomposition.
ullv_dw_b	Downdate B in the rank-revealing ULLV decomposition.
ullv_up_a	Update A in the rank-revealing ULLV decomposition.
ullv_up_b	Update B in the rank-revealing ULLV decomposition.

RRQR Algorithms	
hrrqr	Chan/Foster high-rank RRQR algorithm.
lrrqr	Chan-Hansen low-rank RRQR algorithm.
trrqr	Solves a numerically rank-deficient least squares problem using the RRQR decomposition.

Misc. Tools	
app_giv	Apply a Givens rotation (from the left or right).
app_hous	Apply a Householder reflection (from the left or right).
ccvl	Estimation of the smallest singular value via the Cline-Conn-Van Loan algorithm.
gen_giv	Determine a 2×2 Givens rotation matrix.
gen_hous	Determine a Householder reflection matrix.
inviter	Estimation of the smallest singular value via inverse iteration.
lanczos	Estimation of the largest singular value via Lanczos bidiagonalization.
mgsr	Modified Gram-Schmidt expansion step with reorthogonalization.
powiter	Estimation of the largest singular value via the power method.
ullv_csne	Corrected semi-normal equations expansion step (for ULLV).
ullv_rdef	Deflate one row of L_A in the ULLV decomposition.
ullv_ref	Refine one row of L_A in the ULLV decomposition.
ulv_cdef	Deflate one column of L in the ULV decomposition.
ulv_csne	Corrected semi-normal equations expansion step (for ULV).
ulv_rdef	Deflate one row of L in the ULV decomposition.
ulv_ref	Refine one row of L in the ULV decomposition.
urv_cdef	Deflate one column of R in the URV decomposition.
urv_csne	Corrected semi-normal equations expansion step (for URV).
urv_rdef	Deflate one row of R in the URV decomposition.
urv_ref	Refine one column of R in the URV decomposition.

The Demo Functions

The package includes eight demo functions that illustrate the use and functionality of the main algorithms, by applying them to small test matrices with known singular values. In addition, we demonstrate that our up- and downdating algorithms are capable of tracking the numerical rank of a difficult test problem from [36].

The test matrices used in the ULV, URV, and RRQR demos are "random" matrices with dimensions 50×20, and their singular values are generated by means of the Matlab commands

```
s1 = 2*logspace(1,-3,13);
s2 = 5*logspace(-4,-6,7);
s = [s1,s2];
```

The ULLV demo makes use of a "random" pair of test matrices (A, B) with dimensions 50×20 and 20×20, respectively, whose generalized singular values – i.e., the singular values of $A\,B^{-1}$ – are generated by the above Matlab commands. Hence, the numerical rank in all the test problems is 13 with respect to the threshold $\tau = 10^{-3}$. These matrices are used in the following six demos:

Demo function	Functions illustrated	
hulvdemo	hulv	hulv_a
hurvdemo	hurv	hurv_a
lulvdemo	lulv	lulv_a
lurvdemo	lurv	lurv_a
rrqrdemo	hrrqr	lrrqr
ullvdemo	ullv	

The test matrices used to illustrate the up- and downdating algorithms are from [36]. A "sliding window" technique is used, in which a 5×5 matrix is modified in a series of 19 steps, each step involving updating with one new row and downdating involving the oldest row of the matrix. The matrix is constructed such that the numerical rank in the 20 stages (including the initial stage) are given by the sequence

$$3,\ 3,\ 3,\ 3,\ 4,\ 5,\ 4,\ 3,\ 2,\ 2,\ 2,\ 2,\ 3,\ 4,\ 5,\ 4,\ 3,\ 3,\ 3,\ 3.$$

The available up- and downdating strategies are illustrated as follows:

Demo function	Function illustrated	Techniques illustrated
ullvdemo	ullv_up_a, ullv_dw_a	U available
		U not available, CSNE approach
wulvdemo	ulv_win	U available
		U not available, CSNE approach
		U not available, improved CSNE approach
wurvdemo	urv_win	U available
		U not available, CSNE approach
		U not available, improved CSNE approach

app_giv

Purpose

Apply a Givens rotation (left/right).

Synopsis

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

Description

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

```
[u1] = [ c  s'] [v1]
[u2]   [-s  c ] [v2]
```

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

```
[u1 u2] = [v1 v2] [c  -s]
                  [s'  c]
```

See Also

gen_giv   Determine a 2-by-2 Givens rotation matrix.

References

[1] G.H. Golub and C.F. Van Loan, “Matrix Computations”, Johns Hopkins University Press, 3. Ed., p. 216, (1996).

app_hous

Purpose

Apply a Householder transformation.

Synopsis

A = app_hous(A,beta,v)

Description

Applies the Householder transformation, defined by vector v and scalar beta, to the matrix A, i.e., A = (I - beta∗v∗v^T)∗A.

See Also

gen_hous	Determine a Householder transformation.

References

[1] G.H. Golub and C.F. Van Loan, "Matrix Computations", Johns Hopkins University Press, 3. Ed., p. 211, (1996).

ccvl

Purpose
Singular value/vector estimates via condition estimation.

Synopsis
[smin,vmin] = ccvl(R)

Description
Compute estimates smin and vmin of the smallest singular value and corresponding right singular vector of the upper triangular matrix R.

Algorithm
The function is based on the generalized LINPACK codition number estimator.

See Also

inviter	Singular value/vector estimates via inverse iteration.

References

[1] A.K. Cline, A.R. Conn & C.F. Van Loan, "Generalizing the LINPACK Condition Estimator"; in J.P. Hennart (Ed.), "Numerical Analysis", Lecture Notes in Mathematics, Vol. 909, pp. 73–83, Springer Verlag, (1982).

gen_giv

Purpose

Determine a 2-by-2 Givens rotation matrix.

Synopsis

[c,s,r] = gen_giv(a,b)

Description

Compute a (complex) Givens rotation to annihilate b using a, i.e., compute c, s, and r such that

```
[ c   s'] [a] = [r]    or    [a b] [c   -s] = [r 0]
[-s   c ] [b]   [0]                [s'   c]
```

See Also

app_giv   Apply a Givens rotation (left/right).

References

[1] G.H. Golub and C.F. Van Loan, "Matrix Computations", Johns Hopkins University Press, 3. Ed., p. 215, (1996).

gen_hous

Purpose

Determine a Householder transformation.

Synopsis

[beta,v,r] = gen_hous(x)

Description

Given a vector x, this function computes the scalar beta and the vector v such that (I - beta∗v∗v^T)x is zero in all but the first component r = -sign(x1)∗norm(x). If x = 0 then v = 0 and beta = 1 is returned.

See Also

app_hous	Apply a Householder transformation.

References

[1] G.H. Golub and C.F. Van Loan, "Matrix Computations", Johns Hopkins University Press, 3. Ed., p. 210, (1996).

hrrqr

Purpose

Chan/Foster high-rank-revealing RRQR algorithm.

Synopsis

[p,R,Pi,Q,W,vec] = hrrqr(A)
[p,R,Pi,Q,W,vec] = hrrqr(A,tol_rank)
[p,R,Pi,Q,W,vec] = hrrqr(A,tol_rank,fixed_rank)

Description

Computes a rank-revealing RRQR decomposition of an m-by-n matrix A ($m \geq n$) with numerical rank p close to n. The n-by-n matrix R is upper triangular and will reveal the numerical rank p of A. The norm of the (2,2) block of R is of the order sigma_(p+1).

Input Parameters

A	m-by-n matrix ($m \geq n$);
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 = sqrt(n)*norm(A,1)*eps;

Output Parameters

p	numerical rank of A;
R, Pi, Q	the RRQR factors so that A*Pi = Q*R;
W	an n-by-p matrix whose columns span an approximation to the null space of A;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(R(1:p,p+1:n)), 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

The rectangular matrix A is preprocessed by a QR factorization, A = Q*R. Then deflation steps based on the generalized LINPACK condition estimator are employed to produce a rank-revealing decomposition.

See Also

lrrqr	Chan-Hansen low-rank-revealing RRQR algorithm.

References

[1] T.F. Chan, "Rank Revealing QR Factorizations", Lin. Alg. Appl., 88/89 (1987), pp. 67–82.

[2] L. Foster, "Rank and Null Space Calculations Using Matrix Decomposition Without Column Interchanges", Lin. Alg. Appl., 74 (1986), pp. 47–71.

hulv

Purpose

Stewart's high-rank-revealing ULV algorithm.

Synopsis

[p,L,V,U,vec] = hulv(A)
[p,L,V,U,vec] = hulv(A,tol_rank)
[p,L,V,U,vec] = hulv(A,tol_rank,tol_ref,max_ref)
[p,L,V,U,vec] = hulv(A,tol_rank,tol_ref,max_ref,fixed_rank)

Description

Computes a rank-revealing ULV decomposition of an m-by-n matrix A with m $\geq$ n, where the algorithm is optimized for numerical rank p close to n. In the two-sided orthogonal decomposition, the n-by-n matrix L is lower triangular and will reveal the numerical rank p of A. The norm of the (2,1) and (2,2) blocks of L are of the order sigma_(p+1). U and V are unitary matrices, where only the first n columns of U are computed.

Input Parameters

A	m-by-n matrix (m $\geq$ n);
tol_rank	rank decision tolerance;
tol_ref	upper bound on the 2-norm of the off-diagonal block L(p+1:n,1:p) relative to the Frobenius-norm of L;
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;
L, V, U	the ULV factors such that A = U*L*V^T;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(L(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

The rectangular matrix A is preprocessed by a QL factorization, A = U*L. 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

hulv_a — An alternative high-rank-revealing ULV algorithm.

References

[1] G.W. Stewart, "Updating a Rank-Revealing ULV Decomposition", SIAM J. Matrix Anal. and Appl., 14 (1993), pp. 494–499.

hulv_a

Purpose

An alternative high-rank-revealing ULV algorithm.

Synopsis

```
[p,L,V,U,vec] = hulv_a(A)
[p,L,V,U,vec] = hulv_a(A,tol_rank)
[p,L,V,U,vec] = hulv_a(A,tol_rank,max_iter)
[p,L,V,U,vec] = hulv_a(A,tol_rank,max_iter,tol_ref,max_ref)
[p,L,V,U,vec] = hulv_a(A,tol_rank,max_iter,tol_ref,max_ref,fixed_rank)
```

Description

Computes a rank-revealing ULV decomposition of an m-by-n matrix A with m $\geq$ n, where the algorithm is optimized for numerical rank p close to n. In the two-sided orthogonal decomposition, the n-by-n matrix L is lower triangular and will reveal the numerical rank p of A. Thus, the norm of the (2,1) and (2,2) blocks of L are of the order sigma_(p+1). U and V are unitary matrices, where only the first n columns of U are computed.

Input Parameters

A	m-by-n matrix (m $\geq$ n);
tol_rank	rank decision tolerance;
max_iter	maximum number of steps of inverse iteration in the singular vector estimator;
tol_ref	upper bound on the 2-norm of the off-diagonal block L(p+1:n,1:p) relative to the Frobenius-norm of L;
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;
	max_iter = 5;
	tol_ref = 1e-04;
	max_ref = 0;

Output Parameters

p	numerical rank of A;
L, V, U	the ULV factors such that A = U*L*V^T;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(L(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

The rectangular matrix A is preprocessed by a QL factorization, A = U∗L. Then deflation and refinement (optional) are employed to produce a rank-revealing decomposition. The deflation procedure is based on singular vector estimation via inverse iteration, which can be repeated using refined singular vector estimates.

See Also

hulv	Stewart's high-rank-revealing ULV algorithm.

References

[1] R.D. Fierro, L. Vanhamme and S. Van Huffel, "Total Least Squares Algorithms Based on Rank-Revealing Complete Orthogonal Decompositions". In "Recent Advances in Total Least Squares Techniques and Errors-in-Variables Modeling", pp. 99–116, SIAM, Philadelphia, 1997.

hurv

Purpose

Stewart's high-rank-revealing URV algorithm.

Synopsis

[p,R,V,U,vec] = hurv(A)
[p,R,V,U,vec] = hurv(A,tol_rank)
[p,R,V,U,vec] = hurv(A,tol_rank,tol_ref,max_ref)
[p,R,V,U,vec] = hurv(A,tol_rank,tol_ref,max_ref,fixed_rank)

Description

Computes a rank-revealing URV decomposition of an m-by-n matrix A with m $\geq$ n, where the algorithm is optimized for numerical rank p close to n. In the two-sided orthogonal decomposition, the n-by-n matrix R is upper triangular and will reveal the numerical rank p of A. Thus, the norm of the (1,2) and (2,2) blocks of R are of the order sigma_(p+1). U and V are unitary matrices, where only the first n columns of U are computed.

Input Parameters

A	m-by-n matrix (m $\geq$ n);
tol_rank	rank decision tolerance;
tol_ref	upper bound on the 2-norm of the off-diagonal block R(1:p,p+1:n) relative to the Frobenius-norm of R;
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;
R, V, U	the URV factors such that A = U*R*V^T;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(R(1:p,p+1:n)), 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

The rectangular matrix A is preprocessed by a QR factorization, A = U*R. 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

hurv_a	An alternative high-rank-revealing URV algorithm.

References

[1] G.W. Stewart, “An Updating Algorithm for Subspace Tracking”, IEEE Trans. on SP, 40 (1992), pp. 1535–1541.

hurv_a

Purpose

An alternative high-rank-revealing URV algorithm.

Synopsis

```
[p,R,V,U,vec] = hurv_a(A)
[p,R,V,U,vec] = hurv_a(A,tol_rank)
[p,R,V,U,vec] = hurv_a(A,tol_rank,max_iter)
[p,R,V,U,vec] = hurv_a(A,tol_rank,max_iter,tol_ref,max_ref)
[p,R,V,U,vec] = hurv_a(A,tol_rank,max_iter,tol_ref,max_ref,fixed_rank)
```

Description

Computes a rank-revealing URV decomposition of an m-by-n matrix A with $m \geq n$, where the algorithm is optimized for numerical rank p close to n. In the two-sided orthogonal decomposition, the n-by-n matrix R is upper triangular and will reveal the numerical rank p of A. Thus, the norm of the (1,2) and (2,2) blocks of R are of the order sigma_(p+1). U and V are unitary matrices, where only the first n columns of U are computed.

Input Parameters

A	m-by-n matrix ($m \geq n$);
tol_rank	rank decision tolerance;
max_iter	maximum number of steps of inverse iteration in the singular vector estimator;
tol_ref	upper bound on the 2-norm of the off-diagonal block R(1:p,p+1:n) relative to the Frobenius-norm of R;
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;
	max_iter = 5;
	tol_ref = 1e-04;
	max_ref = 0;

Output Parameters

p	the numerical rank of A;
R, V, U	the URV factors such that $A = U*R*V^T$;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(R(1:p,p+1:n)), 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

The rectangular matrix A is preprocessed by a QR factorization, $A = U*R$. Then deflation and refinement (optional) are employed to produce a rank-revealing decomposition. The deflation procedure is based on singular vector estimation via inverse iteration, which can be repeated using refined singular vector estimates.

See Also

hurv	Stewart's high-rank-revealing URV algorithm.

References

[1] R.D. Fierro, L. Vanhamme and S. Van Huffel, "Total Least Squares Algorithms Based on Rank-Revealing Complete Orthogonal Decompositions". In "Recent Advances in Total Least Squares Techniques and Errors-in-Variables Modeling", pp. 99–116, SIAM, Philadelphia, 1997.

inviter

Purpose

Singular value/vector estimates via inverse iteration.

Synopsis

[smin,vmin] = inviter(R,max_iter,guess_v)

Description

Compute estimates smin and vmin of the smallest singular value and corresponding right singular vector of the upper triangular matrix R via inverse iteration using max_iter iterations. The vector guess_v is the initial guess.

Input Parameters

R	upper triangular matrix;
max_iter	maximum number of steps of inverse iteration;
guess_v	initial guess vector;

See Also

ccvl	Singular value/vector estimates via condition estimation.

References

[1] G.H. Golub and C.F. Van Loan, "Matrix Computations", Johns Hopkins University Press, 3. Ed., p. 362, 1996.

lanczos

Purpose

Singular value/vector estimates using the Lanczos procedure.

Synopsis

[umax,smax,vmax] = lanczos(A,max_iter,guess_u)
[umax,smax,vmax] = lanczos(A,max_iter,guess_u,reorth)

Description

Computes an estimate of the largest singular value and the associated singular vectors of the matrix A using Lanczos bidiagonalization with a maximum of max_iter iterations. The vector guess_u is the starting vector, and if reorth is true, then MGS reorthogonalization is used.

Input Parameters

A	m-by-n matrix;
max_iter	maximum number of iterations;
guess_u	initial guess vector;
reorth	MGS reorthogonalization if true;
Defaults	reorth = 1 (reorthogonalization).

See Also

powiter	Singular value/vector estimates using the power method.

References

[1] G.H. Golub and C.F. Van Loan, “Matrix Computations”, Johns Hopkins University Press, 3. Ed., p. 495, 1996.

lrrqr

Purpose

Chan-Hansen low-rank-revealing RRQR algorithm.

Synopsis

```
[p,R,Pi,Q,W,vec] = lrrqr(A)
[p,R,Pi,Q,W,vec] = lrrqr(A,tol_rank)
[p,R,Pi,Q,W,vec] = lrrqr(A,tol_rank,max_iter)
[p,R,Pi,Q,W,vec] = lrrqr(A,tol_rank,max_iter,fixed_rank)
```

Description

Computes a rank-revealing RRQR decomposition of an m-by-n matrix A ($m \geq n$) with numerical rank p close to 1. The n-by-n matrix R is upper triangular and will reveal the numerical rank p of A. Thus, the norm of the (2,2) block of R is of the order sigma_(p+1).

Input Parameters

A	m-by-n matrix ($m \geq n$);
tol_rank	rank decision tolerance;
max_iter	max. number of steps of 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 = sqrt(n)*norm(A,1)*eps; max_iter = 5;

Output Parameters

p	numerical rank of A;
R, Pi, Q	the RRQR factors so that A*Pi = Q*R;
W	an n-by-p matrix whose columns span an approximation to the null space of A;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(R(1:p,p+1:n)), 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

The rectangular matrix A is preprocessed by a QR factorization, A = Q*R. Then deflation steps based on principal singular vector estimation via the power method are employed to produce a rank-revealing decomposition.

See Also

hrrqr	Chan/Foster high-rank-revealing RRQR algorithm.

References

[1] T.F. Chan and P. C. Hansen, "Low-Rank Revealing QR Factorizations", Num. Lin. Alg. with Applications, 1 (1994), pp. 33–44.

lulv

Purpose

Warm-started low-rank-revealing ULV algorithm.

Synopsis

```
[p,L,V,U,vec] = lulv(A)
[p,L,V,U,vec] = lulv(A,tol_rank)
[p,L,V,U,vec] = lulv(A,tol_rank,max_iter)
[p,L,V,U,vec] = lulv(A,tol_rank,max_iter,num_ref)
[p,L,V,U,vec] = lulv(A,tol_rank,max_iter,num_ref,est_type)
[p,L,V,U,vec] = lulv(A,tol_rank,max_iter,num_ref,est_type,fixed_rank)
```

Description

Computes a rank-revealing ULV decomposition of an m-by-n matrix A with $m \geq n$, where the algorithm is optimized for numerical rank $p \ll n$. In the two-sided orthogonal decomposition, the n-by-n matrix L is lower triangular and will reveal the numerical rank p of A. Thus, the norm of the (2,1) and (2,2) blocks of L are of the order sigma_(p+1). U and V are unitary matrices, where only the first n columns of U are computed.

Input Parameters

A	m-by-n matrix ($m \geq n$);
tol_rank	rank decision tolerance;
max_iter	max. number of steps of the singular vector estimator;
num_ref	number of refinement steps per singular value;
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 = sqrt(n)∗norm(A,1)∗eps;
	max_iter = 5;
	num_ref = 0;
	est_type = 0 (power method);

Output Parameters

p	the numerical rank of A;
L, V, U	the ULV factors such that $A = U*L*V^T$;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(L(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

The rectangular matrix A is preprocessed by a QL factorization, A = U∗L. Then deflation and refinement (optional) are employed to produce a rank-revealing decomposition.

The deflation procedure is based on principal singular vector estimation via the Lanczos or power method, which can be repeated using refined singular vector estimates.

See Also

lulv_a Cold-started low-rank-revealing ULV algorithm.

References

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

lulv_a

Purpose

Cold-started low-rank-revealing ULV algorithm.

Synopsis

[p,L,V,U,vec] = lulv_a(A)
[p,L,V,U,vec] = lulv_a(A,tol_rank)
[p,L,V,U,vec] = lulv_a(A,tol_rank,max_iter)
[p,L,V,U,vec] = lulv_a(A,tol_rank,max_iter,num_ref)
[p,L,V,U,vec] = lulv_a(A,tol_rank,max_iter,num_ref,est_type)
[p,L,V,U,vec] = lulv_a(A,tol_rank,max_iter,num_ref,est_type,fixed_rank)

Description

Computes a rank-revealing ULV decomposition of an m-by-n matrix A with $m \geq n$, where the algorithm is optimized for numerical rank $p \ll n$. In the two-sided orthogonal decomposition, the m-by-n matrix L is lower block-triangular and will reveal the numerical rank p of A. Thus, the norm of the (2,1) and (2,2) blocks of L are of the order sigma_(p+1). U and V are unitary matrices.

Input Parameters

A	m-by-n matrix ($m \geq n$);
tol_rank	rank decision tolerance;
max_iter	max. number of steps of the singular vector estimator;
num_ref	number of refinement steps per singular value;
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 = sqrt(n)*norm(A,1)*eps;
	max_iter = 5;
	num_ref = 0;
	est_type = 0 (power method);

Output Parameters

p	the numerical rank of A;
L, V, U	the ULV factors such that $A = U*L*V^T$;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(L(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

There is no initial decomposition. Deflation and refinement (optional) are employed by means of Householder transformations to produce a rank-revealing decomposition. The deflation procedure is based on principal singular vector estimation via the Lanczos or power method, which can be repeated using refined singular vector estimates.

See Also

lulv	Warm-started low-rank-revealing ULV algorithm.

References

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

lurv

Purpose

Warm-started low-rank-revealing URV algorithm.

Synopsis

```
[p,R,V,U,vec] = lurv(A)
[p,R,V,U,vec] = lurv(A,tol_rank)
[p,R,V,U,vec] = lurv(A,tol_rank,max_iter)
[p,R,V,U,vec] = lurv(A,tol_rank,max_iter,num_ref)
[p,R,V,U,vec] = lurv(A,tol_rank,max_iter,num_ref,est_type)
[p,R,V,U,vec] = lurv(A,tol_rank,max_iter,num_ref,est_type,fixed_rank)
```

Description

Computes a rank-revealing URV decomposition of an m-by-n matrix A with $m \geq n$, where the algorithm is optimized for numerical rank $p \ll n$. In the two-sided orthogonal decomposition, the n-by-n matrix R is upper triangular and will reveal the numerical rank p of A. Thus, the norm of the (1,2) and (2,2) blocks of R are of the order sigma_(p+1). U and V are unitary matrices, where only the first n columns of U are computed.

Input Parameters

A	m-by-n matrix ($m \geq n$);
tol_rank	rank decision tolerance;
max_iter	max. number of steps of the singular vector estimator;
num_ref	number of refinement steps per singular value;
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 = sqrt(n)∗norm(A,1)∗eps;
	max_iter = 5;
	num_ref = 0;
	est_type = 0 (power method);

Output Parameters

p	the numerical rank of A;
R, V, U	the URV factors such that $A = U*R*V^T$;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(R(1:p,p+1:n)), 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

The rectangular matrix A is preprocessed by a QR factorization, A = U∗R. Then deflation and refinement (optional) are employed to produce a rank-revealing decomposition.

The deflation procedure is based on principal singular vector estimation via the Lanczos or power method, which can be repeated using refined singular vector estimates.

See Also

lurv_a	Cold-started low-rank-revealing URV algorithm.

References

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

lurv_a

Purpose

Cold-started low-rank-revealing URV algorithm.

Synopsis

[p,R,V,U,vec] = lurv_a(A)
[p,R,V,U,vec] = lurv_a(A,tol_rank)
[p,R,V,U,vec] = lurv_a(A,tol_rank,max_iter)
[p,R,V,U,vec] = lurv_a(A,tol_rank,max_iter,num_ref)
[p,R,V,U,vec] = lurv_a(A,tol_rank,max_iter,num_ref,est_type)
[p,R,V,U,vec] = lurv_a(A,tol_rank,max_iter,num_ref,est_type,fixed_rank)

Description

Computes a rank-revealing URV decomposition of an m-by-n matrix A with m $\geq$ n, where the algorithm is optimized for numerical rank p $\ll$ n. In the two-sided orthogonal decomposition, the m-by-n matrix R is upper block-triangular and will reveal the numerical rank p of A. Thus, the norm of the (1,2) and (2,2) blocks of R are of the order sigma_(p+1). U and V are unitary matrices.

Input Parameters

A	m-by-n matrix (m $\geq$ n);
tol_rank	rank decision tolerance;
max_iter	max. number of steps of the singular vector estimator;
num_ref	number of refinement steps per singular value;
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 = sqrt(n)$*$norm(A,1)$*$eps;
	max_iter = 5;
	num_ref = 0;
	est_type = 0 (power method);

Output Parameters

p	the numerical rank of A;
R, V, U	the URV factors such that A = U$*$R$*$V^T;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(R(1:p,p+1:n)), 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

There is no initial decomposition. Deflation and refinement (optional) are employed by means of Householder transformations to produce a rank-revealing decomposition. The deflation procedure is based on principal singular vector estimation via the Lanczos or power method, which can be repeated using refined singular vector estimates.

See Also

lurv	Warm-started low-rank-revealing URV algorithm.

References

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

mgsr

Purpose

Modified Gram-Schmidt with re-orthogonalization (expansion step).

Synopsis

q = mgsr(U,kappa)

Description

Modified Gram-Schmidt with re-orthogonalization is used to expand the m-by-n matrix U having orthogonal columns with a new column q, which is orthogonal to the other columns of U. The parameter kappa (greater than one) is used to decide if re-orthogonalization is needed; kappa = 1 ensures re-orthogonalization, however, a typical value for kappa is sqrt(2).

Algorithm

The algorithm relies on the fact that one re-orthogonalization is always enough.

See Also

ullv_csne	Corrected semi-normal equations expansion (ULLV).
ulv_csne	Corrected semi-normal equations expansion (ULV).
urv_csne	Corrected semi-normal equations expansion (URV).

References

[1] J.W. Daniel, W.B. Gragg, L. Kaufman and G.W. Stewart, "Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization", Math. Comp., 30 (1976), pp. 772–795.

powiter

Purpose

Singular value/vector estimates using the power method.

Synopsis

[umax,smax,vmax] = powiter(A,max_iter,guess_u)

Description

Compute approximations smax, umax, and vmax to the principal singular value and the corresponding left and right singular vectors of the m-by-n matrix A using the power method. The initial guess of umax is guess_u, and the maximum number of iterations is determined by max_iter.

See Also

lanczos	Singular value/vector estimates using the Lanczos procedure.

References

[1] G.H. Golub and C.F. Van Loan, “Matrix Computations”, Johns Hopkins University Press, 3. Ed., p. 330, 1996.

trrqr

Purpose

Solves a least squares problem using the RRQR decomposition.

Synopsis

x_trrqr = trrqr(Q,R,Pi,p,b)

Description

Solves the near-rank deficient least squares problem

```
min_x || b-A*x ||_2
```

using the RRQR decomposition. Here, A∗Pi = Q∗R is the RRQR decomposition of A, p is the numerical rank of A, and the TRRQR solution is defined by

```
x_trrqr = Pi(:,1:p)*inv(R(1:p,1:p))*Q(:,1:p)'*b.
```

See Also

tulv	Solves a least squares problem using the ULV decomposition.
turv	Solves a least squares problem using the URV decomposition.

tulv

Purpose

Solves a least squares problem using the ULV decomposition.

Synopsis

x_tulv = tulv(U,L,V,p,b)

Description

Solves the near-rank deficient least squares problem

```
min_x || b-A*x ||_2
```

using the ULV decomposition. Here, A = U∗L∗V^T is the ULV decomposition of A, p is the numerical rank of A, and the TULV solution is defined by

```
x_tulv = V(:,1:p)*inv(L(1:p,1:p))*U(:,1:p)'*b.
```

See Also

turv	Solves a least squares problem using the URV decomposition.
trrqr	Solves a least squares problem using the RRQR decomposition.

turv

Purpose

Solves a least squares problem using the URV decomposition.

Synopsis

x_turv = turv(U,R,V,p,b)

Description

Solves the near-rank deficient least squares problem

```
min_x || b-A*x ||_2
```

using the URV decomposition. Here, A = U∗R∗V^T is the URV decomposition of A, p is the numerical rank of A, and the TURV solution is defined by

```
x_turv = V(:,1:p)*inv(R(1:p,1:p))*U(:,1:p)'*b.
```

See Also

tulv	Solves a least squares problem using the ULV decomposition.
trrqr	Solves a least squares problem using the RRQR decomposition.

ullv

Purpose

High-rank-revealing ULLV algorithm.

Synopsis

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

Description

Computes a rank-revealing ULLV decomposition of an mA-by-n matrix A (mA $\geq$ n) and an mB-by-n full-rank matrix B (mB $\geq$ n):

```
A = UA*LA*L*V'    and    B = UB*L*V'
```

The ULLV decomposition is a quotient ULV decomposition, i.e., the n-by-n matrix LA is lower triangular and will reveal the numerical rank p of A*pinv(B). Thus, the norm of the (2,1) and (2,2) blocks of LA are of the order sigma_(p+1). U and V are unitary matrices, where only the first n columns of UA and UB are computed.

Note that the algorithm is optimized for numerical rank p close to n, and that this algorithm should not be used if B is ill conditioned or rank deficient.

Input Parameters

A	mA-by-n matrix (mA $\geq$ n);
B	mB-by-n matrix (mB $\geq$ 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∗pseudoinverse(B);
LA,L,V,UA,UB	the ULLV factors such that A = UA∗LA∗L∗V^T and B = UB∗L∗V^T;
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 find the QL factorization of B = UB∗L and solve X = A∗inv(L) followed by another QL factorization of X = UA∗LA. Thus, A = UA∗LA∗L and B = UB∗L. 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

hulv	Stewart's high-rank-revealing ULV algorithm.
hulv_a	An alternative high-rank-revealing ULV algorithm.

References

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

ullv_csne

Purpose

Corrected semi-normal equations expansion (ULLV).

Synopsis

[u1,q1,flag_csne] = ullv_csne(A,LA,L,V,kappa)

Description

Compute the first row u1 of the m-by-n matrix UA and the first element q1 of the expanded column q which is orthogonal to the columns of UA, by using the LINPACK approach if it is safe, and if not, by solving the following least squares problem by means of the CSNE method:

```
(A*V)'*(A*V)*z = (LA*L)'*(LA*L)*z = (A*V)'*e1
```

where

```
A = UA*LA*L*V'
```

If the parameter flag_csne is true, the CSNE approach has been used. The parameter kappa (greater than one) is used to control the orthogonalization procedure. A typical value for kappa is sqrt(2).

Algorithm

The algorithm is based on triangular solves. If LA is rank deficient, then the rank information is used in the triangular solves.

See Also

mgsr	Modified Gram-Schmidt expansion.
ulv_csne	Corrected semi-normal equations expansion (ULV).

References

[1] A. Björck, H. Park and L. Eldén, "Accurate Downdating of Least Squares Solutions", SIAM J. Matrix. Anal. Appl., 15 (1994), pp. 549–568.

[2] H. Park and L. Eldén, "Downdating the Rank Revealing URV Decomposition", SIAM J. Matrix Anal. Appl., 16 (1995), pp. 138–155.

ullv_dw_a

Purpose

Downdate the A-part of the rank-revealing ULLV decomposition.

Synopsis

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

Description

Given a rank-revealing ULLV decomposition of the mA-by-n matrix A = UA∗LA∗L∗V^T and mB-by-n matrix B = UB∗L∗V^T (mA > n), the function computes the downdated decomposition

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

where a is the top row being removed from A. If the matrix UA is maintained, the modified Gram-Schmidt algorithm is used in the expansion step of the downdating algorithm. If the matrix UA is left out by inserting an empty matrix `[]`, the method of LINPACK/CSNE (corrected semi-normal equations) is used, and the matrix A is needed. Note that the row dimension of UA will decrease by one, and that the matrix UB can always be left out by inserting an empty matrix `[]`.

Input Parameters

p	numerical rank of A∗pseudoinv(B) revealed in LA;
LA,L,V,UA,UB	the ULLV factors such that A = UA∗LA∗L∗V^T and B = UB∗L∗V^T;
A	mA-by-n matrix (mA > 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	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 the downdated decomposition;
LA,L,V,UA,UB	the ULLV factors such that A = [a; UA*LA*L*V^T] and B = UB*L*V^T;
vec	a 6-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. vec(6) = true if CSNE approach has been used.

See Also

ullv_up_a	Update the A-part of the rank-revealing ULLV decomposition.

References

[1] A.W. Bojanczyk and J.M. Lebak, "Downdating a ULLV Decomposition of Two Matrices"; in J.G. Lewis (Ed.), "Applied Linear Algebra", SIAM, Philadelphia, 1994.

[2] J.M. Lebak and A.W. Bojanczyk, "Modifying a Rank-Revealing ULLV Decomposition", Report CTC94TR186, School of Electrical Engineering, Cornell University, 1994.

[3] M. Moonen, P. Van Dooren and J. Vandewalle, "A Note on Efficient Numerically Stabilized Rank-One Eigenstructure Updating", IEEE Trans. on Signal Processing, 39 (1991), pp. 1911–1913.

ullv_dw_b

Purpose

Downdate the B-part of the rank-revealing ULLV decomposition.

Synopsis

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

Description

Given a rank-revealing ULLV decomposition of the mA-by-n matrix A = UA∗LA∗L∗V^T and mB-by-n matrix B = UB∗L∗V^T (mB > n), the function computes the downdated decomposition

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

where b is the top row being removed from B. If the matrix UB is maintained, the modified Gram-Schmidt algorithm is used in the expansion step of the downdating algorithm. If the matrix UB is left out by inserting an empty matrix `[]`, the method of LINPACK/CSNE (corrected semi-normal equations) is used, and the matrix B is needed. Note that the row dimension of UB will decrease by one, and that the matrix UA can always be left out by inserting an empty matrix `[]`.

Input Parameters

p	numerical rank of A∗pseudoinv(B) revealed in LA;
LA,L,V,UA,UB	the ULLV factors such that A = UA∗LA∗L∗V^T and B = UB∗L∗V^T;
B	mB-by-n matrix (mB > 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	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 the downdated decomposition;
LA,L,V,UA,UB	the ULLV factors such that A = UA*LA*L*V^T and B = [b; UB*L*V^T];
vec	a 6-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. vec(6) = true if CSNE approach has been used.

See Also

ullv_up_b	Update the B-part of the rank-revealing ULLV decomposition.

References

[1] A.W. Bojanczyk and J.M. Lebak, "Downdating a ULLV Decomposition of Two Matrices"; in J.G. Lewis (Ed.), "Applied Linear Algebra", SIAM, Philadelphia, 1994.

[2] J.M. Lebak and A.W. Bojanczyk, "Modifying a Rank-Revealing ULLV Decomposition", Report CTC94TR186, School of Electrical Engineering, Cornell University, 1994.

[3] M. Moonen, P. Van Dooren and J. Vandewalle, "A Note on Efficient Numerically Stabilized Rank-One Eigenstructure Updating", IEEE Trans. on Signal Processing, 39 (1991), pp. 1911–1913.

ullv_rdef

Purpose

Deflate one row of LA in the ULLV decomposition.

Synopsis

[LA,L,V,UA,UB] = ullv_rdef(LA,L,V,UA,UB,r,umin)

Description

Given the ULLV decomposition of the matrix pair $A = UA*LA*L*V^T$ and $B = UB*L*V^T$, the function deflates LA(1:r,1:r). umin is an estimate of the left singular vector of LA(1:r,1:r) associated with the smallest singular value. On return, norm(LA(r,1:r)) is of the order sigma_(r). The matrices UA, UB and V can be left out by inserting an empty matrix [].

See Also

ullv_ref	Refine one row of LA in the ULLV decomposition.

References

[1] G.W. Stewart, "An Updating Algorithm for Subspace Tracking", IEEE Trans. on SP 40 (1992), pp. 1535–1541.

[2] G.W. Stewart, "Updating a Rank-Revealing ULV Decomposition", SIAM J. Matrix Anal. Appl., 14 (1993), pp. 494–499.

ullv_ref

Purpose

Refine one row of LA in the ULLV decomposition.

Synopsis

[LA,L,V,UA,UB] = ullv_ref(LA,L,V,UA,UB,r)

Description

Given the ULLV decomposition of the matrix pair A = UA∗LA∗L∗V^T and B = UB∗L∗V^T, the function refines the last row of LA(1:r,1:r). The matrices UA, UB and V can be left out by inserting an empty matrix `[]`.

Algorithm

Refinement is an iterative algorithm, which reduces the norm of the target row by applying one block QR iteration to LA.

See Also

ullv_rdef	Deflate one row of LA in the ULLV decomposition.

References

[1] S.Qiao, “Computing the ULLV Decomposition”, CRL Report 278, Communications Research Laboratory, McMaster Uni., Hamilton, Canada, pp. 1–13, January, (1994).

[2] G.W. Stewart, “An Updating Algorithm for Subspace Tracking”, IEEE Trans. on SP, 40 (1992), pp. 1535–1541.

[3] G.W. Stewart, “Updating a Rank-Revealing ULV Decomposition”, SIAM J. Matrix Anal. and Appl., 14 (1993), pp. 494–499.

ullv_up_a

Purpose

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

Synopsis

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

Description

Given a rank-revealing ULLV decomposition of the mA-by-n matrix A = UA$*$LA$*$L$*$V^T and mB-by-n matrix B = UB$*$L$*$V^T (mA,mB $\geq$ n), the function computes the updated decomposition

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

where a is the new row added to A, and beta is a forgetting factor in [0;1], which is multiplied to existing rows to damp out the old data. Note 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 [].

Input Parameters

p	numerical rank of A$*$pseudoinv(B) revealed in LA;
LA,L,V,UA,UB	the ULLV factors such that A = UA$*$LA$*$L$*$V^T and B = UB$*$L$*$V^T;
a	the new row added to A;
beta	forgetting factor in [0;1];
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	if true, deflate to the fixed rank given by p instead of using the rank decision tolerance;

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

Output Parameters

p	numerical rank of [beta*A; a]*pseudoinverse(B);
LA,L,V,UA,UB	the ULLV factors such that [beta*A; a] = UA*LA*L*V^T and B = UB*L*V^T;
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.

See Also

ullv_up_b	Update the B-part of the rank-revealing ULLV decomp.
ullv_dw_a	Downdate the A-part of the rank-revealing ULLV decomp.

References

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

[2] M. Moonen, P. Van Dooren and J. Vandewalle, "A Note on Efficient Numerically Stabilized Rank-One Eigenstructure Updating", IEEE Trans. on Signal Processing, 39 (1991), pp. 1911–1913.

ullv_up_b

Purpose

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

Synopsis

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

Description

Given a rank-revealing ULLV decomposition of the mA-by-n matrix A = UA$*$LA$*$L$*$V^T and mB-by-n matrix B = UB$*$L$*$V^T (mA,mB $\geq$ n), the function computes the updated decomposition

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

where b is the new row added to B, and beta is a forgetting factor in [0;1], which is multiplied to existing rows to damp out the old data. Note 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 [].

Input Parameters

p	numerical rank of A$*$pseudoinv(B) revealed in LA;
LA,L,V,UA,UB	the ULLV factors such that A = UA$*$LA$*$L$*$V^T and B = UB$*$L$*$V^T;
b	the new row added to B;
beta	forgetting factor in [0;1];
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	if true, deflate to the fixed rank given by p instead of using the rank decision tolerance;

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

Output Parameters

p	numerical rank of A∗pseudoinverse([beta∗B; b]);
LA,L,V,UA,UB	the ULLV factors such that A = UA∗LA∗L∗V^T and [beta∗B; b] = UB∗L∗V^T;
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.

See Also

ullv_up_a	Update the A-part of the rank-revealing ULLV decomp.
ullv_dw_b	Downdate the B-part of the rank-revealing ULLV decomp.

References

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

[2] M. Moonen, P. Van Dooren and J. Vandewalle, “A Note on Efficient Numerically Stabilized Rank-One Eigenstructure Updating”, IEEE Trans. on Signal Processing, 39 (1991), pp. 1911–1913.

ulv_cdef

Purpose

Deflate one column of L in the ULV decomposition.

Synopsis

`[L,V,U] = ulv_cdef(L,V,U,r,umax)`

Description

Given the ULV decomposition $U*L*V^T$, the function deflates L(r:n,r:n). umax is an estimate of the left singular vector of L(r:n,r:n) associated with the largest singular value. On return, norm(L(r:n,r)) is of the order sigma_r. The matrices U and V can be left out by inserting an empty matrix `[]`.

See Also

ulv_rdef	Deflate one row of L in the ULV decomposition.

References

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

ulv_csne

Purpose

Corrected semi-normal equations expansion (ULV).

Synopsis

[u1,q1,flag_csne] = ulv_csne(A,L,V,kappa)

Description

Compute the first row u1 of the m-by-n matrix U and the first element q1 of the expanded column q which is orthogonal to the columns of U, by using the LINPACK approach if it is safe, and if not, by solving the following least squares problem by means of the CSNE method:

```
(A*V)'*(A*V)*z = L'*L*z = (A*V)'*e1
```

where

```
A = U*L*V'
```

If the parameter flag_csne is true, the CSNE approach has been used. The parameter kappa (greater than one) is used to control the orthogonalization procedure. A typical value for kappa is sqrt(2).

Algorithm

The algorithm is based on triangular solves. If L is rank deficient, then the rank information is used in the triangular solves.

See Also

mgsr	Modified Gram-Schmidt expansion.
ullv_csne	Corrected semi-normal equations expansion (ULLV).
urv_csne	Corrected semi-normal equations expansion (URV).

References

[1] A. Björck, H. Park and L. Eldén, "Accurate Downdating of Least Squares Solutions", SIAM J. Matrix. Anal. Appl., 15 (1994), pp. 549–568.

[2] H. Park and L. Eldén, "Downdating the Rank Revealing URV Decomposition", SIAM J. Matrix Anal. Appl., 16 (1995), pp. 138–155.

ulv_dw

Purpose

Downdating a row in the rank-revealing ULV decomposition.

Synopsis

```
[p,L,V,U,vec] = ulv_dw(p,L,V,U)
[p,L,V,U,vec] = ulv_dw(p,L,V,U,A)
[p,L,V,U,vec] = ulv_dw(p,L,V,U,A,alg_type)
[p,L,V,U,vec] = ulv_dw(p,L,V,U,A,alg_type,tol_rank)
[p,L,V,U,vec] = ulv_dw(p,L,V,U,A,alg_type,tol_rank,tol_ref,max_ref)
[p,L,V,U,vec] = ulv_dw(p,L,V,U,A,alg_type,tol_rank,tol_ref,max_ref,fixed_rank)
```

Description

Given a rank-revealing ULV decomposition of an m-by-n matrix A = U$*$L$*$V^T with m $\geq$ n, the function computes the downdated decomposition

```
A = [   a   ]
    [U*L*V']
```

where a is the top row being removed from A. Two of the downdating algorithms operate on the lower triangular matrix L without using the information of its rank-revealing structure in the downdate step. The two variants differ in the way they obtain information of the first row of U. If the matrix U is maintained, the modified Gram-Schmidt algorithm is used in the expansion step of the downdating algorithm (alg_type=3). Note that the row dimension of the returned U has decreased by one. If the matrix U is left out by inserting an empty matrix [], the method of LINPACK/CSNE (corrected semi-normal equations) is used (alg_type=1), and the matrix A is needed.

The third downdating algorithm operates on the lower triangular matrix L by using the information of its rank-revealing structure to the extent possible (alg_type=2). This variant can be considered as a improved version of LINPACK/CSNE, and its accuracy will be in between the two other algorithms.

Input Parameters

p	numerical rank of A revealed in L;
L, V, U	the ULV factors such that A = U$*$L$*$V^T;
A	m-by-n matrix (m > n);
alg_type	algorithm type (see Description);
tol_rank	rank decision tolerance;
tol_ref	upper bound on the 2-norm of the off-diagonal block L(p+1:n,1:p) relative to the Frobenius-norm of L;
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	alg_type = 3;
	tol_rank = sqrt(n)∗norm(L,1)∗eps;
	tol_ref = 1e-04;
	max_ref = 0;

Output Parameters

p	numerical rank of the downdated decomposition;
L, V, U	the ULV factors such that A = [a; U∗L∗V^T];
vec	a 6-by-1 vector with: vec(1) = upper bound of norm(L(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. vec(6) = true if CSNE approach has been used.

See Also

ulv_up	Updating a row in the rank-revealing ULV decomposition.
ulv_win	Sliding window modification of the rank-revealing ULV decomp.

References

[1] A. Björck, H. Park and L. Eldén, "Accurate Downdating of Least Squares Solutions", SIAM J. Matrix. Anal. Appl., 15 (1994), pp. 549–568.

[2] H. Park and L. Eldén, "Downdating the Rank Revealing URV Decomposition", SIAM J. Matrix Anal. Appl., 16 (1995), pp. 138–155.

[3] A.W. Bojanczyk and J.M. Lebak, "Downdating a ULLV Decomposition of Two Matrices"; in J.G. Lewis (Ed.), "Applied Linear Algebra", SIAM, Philadelphia, 1994.

[4] J. L. Barlow, P. A. Yoon and H. Zha, "An Algorithm and a Stability Theory for Downdating the ULV Decomposition", BIT, 36 (1996), pp. 14–40.

[5] M. Moonen, P. Van Dooren and J. Vandewalle, "A Note on Efficient Numerically Stabilized Rank-One Eigenstructure Updating", IEEE Trans. on Signal Processing, 39 (1991), pp. 1911–1913.

ulv_qrit

Purpose

Refinement of L in the ULV decomposition via QR-iterations.

Synopsis

```
[L,V,U] = ulv_qrit(p,num_ref,L,V,U)
[L,V] = ulv_qrit(p,num_ref,L,V)
[L] = ulv_qrit(p,num_ref,L)
```

Description

Given the ULV decomposition $U{*}L{*}V^T$ with numerical rank p, the function refines the rank-revealing decomposition via num_ref steps of block QR iterations.

Algorithm

Refinement is identical to block QR iteration, in which the off-diagonal block of the lower triangular matrix L is "flipped" to the (1,2)-position and then back again.

See Also

ulv_ref	Refine one row of L in the ULV 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.

ulv_rdef

Purpose

Deflate one row of L in the ULV decomposition.

Synopsis

[L,V,U] = ulv_rdef(L,V,U,r,umin)

Description

Given the ULV decomposition U$*$L$*$V^T, the function deflates L(1:r,1:r). umin is an estimate of the left singular vector of L(1:r,1:r) associated with the smallest singular value. On return, norm(L(r,1:r)) is of the order sigma_r. The matrices U and V can be left out by inserting an empty matrix [].

See Also

ulv_cdef	Deflate one column of L in the ULV decomposition.
ulv_ref	Refine one row of L in the ULV decomposition.

References

[1] G.W. Stewart, "An Updating Algorithm for Subspace Tracking", IEEE Trans. on SP 40 (1992), pp. 1535–1541.

[2] G.W. Stewart, "Updating a Rank-Revealing ULV Decomposition", SIAM J. Matrix Anal. Appl., 14 (1993), pp. 494–499.

ulv_ref

Purpose

Refine one row of L in the ULV decomposition.

Synopsis

[L,V,U] = ulv_ref(L,V,U,r)

Description

Given the ULV decomposition $U*L*V^T$, the function refines the last row of L(1:r,1:r). The matrices U and V can be left out by inserting an empty matrix [].

Algorithm

Refinement is an iterative algorithm, which reduces the norm of the target row by applying one block QR iteration to L.

See Also

ulv_rdef	Deflate one row of L in the ULV decomposition.

References

[1] G.W. Stewart, "An Updating Algorithm for Subspace Tracking", IEEE Trans. on SP, 40 (1992), pp. 1535–1541.

[2] G.W. Stewart, "Updating a Rank-Revealing ULV Decomposition", SIAM J. Matrix Anal. and Appl., 14 (1993), pp. 494–499.

ulv_up

Purpose

Updating a row in the rank-revealing ULV decomposition.

Synopsis

```
[p,L,V,U,vec] = ulv_up(p,L,V,U,a)
[p,L,V,U,vec] = ulv_up(p,L,V,U,a,beta)
[p,L,V,U,vec] = ulv_up(p,L,V,U,a,beta,tol_rank)
[p,L,V,U,vec] = ulv_up(p,L,V,U,a,beta,tol_rank,tol_ref,max_ref)
[p,L,V,U,vec] = ulv_up(p,L,V,U,a,beta,tol_rank,tol_ref,max_ref,fixed_rank)
```

Description

Given a rank-revealing ULV decomposition of an m-by-n matrix A = U$*$L$*$V^T with m $\geq$ n, the function computes the updated decomposition

```
[beta*A] = U*L*V'
[  a   ]
```

where a is the new row added to A, and beta is a forgetting factor in [0;1], which is multiplied to existing rows to damp out the old data. Note that the row dimension of U will increase by one, and that the matrix U can be left out by inserting an empty matrix [].

Input Parameters

p	numerical rank of A revealed in L;
L, V, U	the ULV factors such that A = U$*$L$*$V^T;
a	the new row added to A;
beta	forgetting factor in [0;1];
tol_rank	rank decision tolerance;
tol_ref	upper bound on the 2-norm of the off-diagonal block L(p+1:n,1:p) relative to the Frobenius-norm of L;
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	beta = 1;
	tol_rank = sqrt(n)$*$norm(L,1)$*$eps;
	tol_ref = 1e-04;
	max_ref = 0;

Output Parameters

p	numerical rank of [beta*A; a];
L, V, U	the ULV factors such that [beta*A; a] = U*L*V^T;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(L(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.

See Also

ulv_dw	Downdating a row in the rank-revealing ULV decomposition.
ulv_win	Sliding window modification of the rank-revealing ULV decomp.

References

[1] G.W. Stewart, "An Updating Algorithm for Subspace Tracking", IEEE Trans. on SP, 40 (1992), pp. 1535–1541.

[2] G.W. Stewart, "Updating a Rank-Revealing ULV Decomposition", SIAM J. Matrix Anal. and Appl., 14 (1993), pp. 494–499.

[3] M. Moonen, P. Van Dooren and J. Vandewalle, "A Note on Efficient Numerically Stabilized Rank-One Eigenstructure Updating", IEEE Trans. on Signal Processing, 39 (1991), pp. 1911–1913.

ulv_win

Purpose

Sliding window modification of the rank-revealing ULV decomp.

Synopsis

[p,L,V,U,vec] = ulv_win(p,L,V,U,A,a)
[p,L,V,U,vec] = ulv_win(p,L,V,U,A,a,alg_type)
[p,L,V,U,vec] = ulv_win(p,L,V,U,A,a,alg_type,tol_rank)
[p,L,V,U,vec] = ulv_win(p,L,V,U,A,a,alg_type,tol_rank,tol_ref,max_ref)
[p,L,V,U,vec] = ulv_win(p,L,V,U,A,a,alg_type,tol_rank,tol_ref,max_ref,fixed_rank)

Description

Given a rank-revealing ULV decomposition of an m-by-n matrix A = U*L*V^T (m $\geq$ n), the function computes the updated decomposition corresponding to the combined up- and down-dating action

```
A  ->  [ A ]  ->  [ w ]
       [ a ]      [ A ]
```

where a is a new row added to A, and w is the row that is downdated after the updating process. If U is not available, then insert the empty matrix `[]`.

Input Parameters

p	numerical rank of A revealed in L;
L, V, U	the ULV factors such that A = U*L*V^T;
A	m-by-n matrix (m $\geq$ n);
a	new row added to A;
alg_type	algorithm type (see Description of ulv_dw);
tol_rank	rank decision tolerance;
tol_ref	upper bound on the 2-norm of the off-diagonal block L(p+1:n,1:p) relative to the Frobenius-norm of L;
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	alg_type = 3;
	tol_rank = sqrt(n)*norm(L,1)*eps;
	tol_ref = 1e-04;
	max_ref = 0;

Output Parameters

p	numerical rank of the modified A;
L, V, U	the ULV factors such that the modified A = U*L*V^T;
vec	a 6-by-1 vector with: vec(1) = upper bound of norm(L(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. vec(6) = true if CSNE approach has been used.

See Also

urv_win	Sliding window modification of the rank-revealing URV decomp.

References

[1] G.W. Stewart, "Updating a Rank-Revealing ULV Decomposition", SIAM J. Matrix Anal. Appl., 14 (1993), pp. 494–499.

urv_cdef

Purpose

Deflate one column of R in the URV decomposition.

Synopsis

[R,V,U] = urv_cdef(R,V,U,r,vmin)

Description

Given the URV decomposition $U*R*V^T$, the function deflates R(1:r,1:r). vmin is an estimate of the right singular vector of R(1:r,1:r) associated with the smallest singular value. On return, norm(R(1:r,r)) is of the order sigma_r. The matrices U and V can be left out by inserting an empty matrix [].

See Also

urv_rdef	Deflate one row of R in the URV decomposition.
urv_ref	Refine one column of R in the URV decomposition.

References

[1] G.W. Stewart, "An Updating Algorithm for Subspace Tracking", IEEE Trans. on SP, 40 (1992), pp. 1535–1541.

[2] G.W. Stewart, "Updating a Rank-Revealing ULV Decomposition", SIAM J. Matrix Anal. Appl., 14 (1993), pp. 494–499.

urv_csne

Purpose

Corrected semi-normal equations expansion (URV).

Synopsis

[u1,q1,flag_csne] = urv_csne(A,R,V,kappa)

Description

Compute the first row u1 of the m-by-n matrix U and the first element q1 of the expanded column q which is orthogonal to the columns of U, by using the LINPACK approach if it is safe, and if not, by solving the following least squares problem by means of the CSNE method:

```
(A*V)'*(A*V)*z = R'*R*z = (A*V)'*e1
```

where

```
A = U*R*V'
```

If the parameter flag_csne is true, the CSNE approach has been used. The parameter kappa (greater than one) is used to control the orthogonalization procedure. A typical value for kappa is sqrt(2).

Algorithm

The algorithm is based on triangular solves. If R is rank deficient, then the rank information is used in the triangular solves.

See Also

mgsr	Modified Gram-Schmidt expansion.
ulv_csne	Corrected semi-normal equations expansion (ULV).

References

[1] A. Björck, H. Park and L. Eldén, "Accurate Downdating of Least Squares Solutions", SIAM J. Matrix. Anal. Appl., 15 (1994), pp. 549–568.

[2] H. Park and L. Eldén, "Downdating the Rank Revealing URV Decomposition", SIAM J. Matrix Anal. Appl., 16 (1995), pp. 138–155.

urv_dw

Purpose

Downdating a row in the rank-revealing URV decomposition.

Synopsis

[p,R,V,U,vec] = urv_dw(p,R,V,U)
[p,R,V,U,vec] = urv_dw(p,R,V,U,A)
[p,R,V,U,vec] = urv_dw(p,R,V,U,A,alg_type)
[p,R,V,U,vec] = urv_dw(p,R,V,U,A,alg_type,tol_rank)
[p,R,V,U,vec] = urv_dw(p,R,V,U,A,alg_type,tol_rank,tol_ref,max_ref)
[p,R,V,U,vec] = urv_dw(p,R,V,U,A,alg_type,tol_rank,tol_ref,max_ref,fixed_rank)

Description

Given a rank-revealing URV decomposition of an m-by-n matrix A = U$*$R$*$V^T with m $\geq$ n, the function computes the downdated decomposition

```
A = [   a   ]
    [U*R*V']
```

where a is the top row being removed from A. Two of the downdating algorithms operate on the upper triangular matrix R without using the information of its rank-revealing structure in the downdate step. The two variants differ in the way they obtain information of the first row of U. If the matrix U is maintained, the modified Gram-Schmidt algorithm is used in the expansion step of the downdating algorithm (alg_type=3). Note that the row dimension of the returned U has decreased by one. If the matrix U is left out by inserting an empty matrix [], the method of LINPACK/CSNE (corrected semi-normal equations) is used (alg_type=1), and the matrix A is needed.

The third downdating algorithm operates on the upper triangular matrix R by using the information of its rank-revealing structure to the extent possible (alg_type=2). This variant can be considered as an improved version of LINPACK/CSNE, and its accuracy will be in between the two other algorithms.

Input Parameters

p	numerical rank of A revealed in R;
R, V, U	the URV factors such that A = U$*$R$*$V^T;
A	m-by-n matrix (m > n);
alg_type	algorithm type (see Description);
tol_rank	rank decision tolerance;
tol_ref	upper bound on the 2-norm of the off-diagonal block R(1:p,p+1:n) relative to the Frobenius-norm of R;
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	alg_type = 3;
	tol_rank = sqrt(n)*norm(R,1)*eps;
	tol_ref = 1e-04;
	max_ref = 0;

Output Parameters

p	the numerical rank of the downdated decomposition;
R, V, U	the URV factors such that A = [a; U*R*V^T];
vec	a 6-by-1 vector with: vec(1) = upper bound of norm(R(1:p,p+1:n)), 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. vec(6) = true if CSNE approach has been used.

See Also

urv_up	Updating a row in the rank-revealing URV decomposition.
urv_win	Sliding window modification of the rank-revealing URV decomp.

References

[1] A. Björck, H. Park and L. Eldén, "Accurate Downdating of Least Squares Solutions", SIAM J. Matrix. Anal. Appl., 15 (1994), pp. 549–568.

[2] H. Park and L. Eldén, "Downdating the Rank Revealing URV Decomposition", SIAM J. Matrix Anal. Appl., 16 (1995), pp. 138–155.

[3] A.W. Bojanczyk and J.M. Lebak, "Downdating a ULLV Decomposition of Two Matrices"; in J.G. Lewis (Ed.), "Applied Linear Algebra", SIAM, Philadelphia, 1994.

[4] J. L. Barlow, P. A. Yoon and H. Zha, "An Algorithm and a Stability Theory for Downdating the ULV Decomposition", BIT, 36 (1996), pp. 14–40.

[5] M. Moonen, P. Van Dooren and J. Vandewalle, "A Note on Efficient Numerically Stabilized Rank-One Eigenstructure Updating", IEEE Trans. on Signal Processing, 39 (1991), pp. 1911–1913.

urv_qrit

Purpose

Refinement of R in the URV decomposition via QR-iterations.

Synopsis

```
[R,V,U] = urv_qrit(p,num_ref,R,V,U)
[R,V] = urv_qrit(p,num_ref,R,V)
[R] = urv_qrit(p,num_ref,R)
```

Description

Given the URV decomposition U∗R∗V^T with numerical rank p, the function refines the rank-revealing decomposition via num_ref steps of block QR iterations.

Algorithm

Refinement is identical to block QR iteration, in which the off-diagonal block of the upper triangular matrix R is "flipped" to the (2,1)-position and then back again.

See Also

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.

urv_rdef

Purpose

Deflate one row of R in the URV decomposition.

Synopsis

[R,V,U] = urv_rdef(R,V,U,r,vmax)

Description

Given the URV decomposition U$*$R$*$V^T, the function deflates R(r:n,r:n). vmax is an estimate of the right singular vector of R(r:n,r:n) associated with the largest singular value. On return, norm(R(r,r:n)) is of the order sigma_r. The matrices U and V can be left out by inserting an empty matrix `[]`.

See Also

urv_cdef	Deflate one column of R in the URV decomposition.

References

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

urv_ref

Purpose

Refine one column of R in the URV decomposition.

Synopsis

[R,V,U] = urv_ref(R,V,U,r)

Description

Given the URV decomposition $U*R*V^T$, the function refines the last column of R(1:r,1:r). The matrices U and V can be left out by inserting an empty matrix `[]`.

Algorithm

Refinement is an iterative algorithm, which reduces the norm of the target column by applying one block QR iteration to R.

See Also

urv_cdef	Deflate one column of R in the URV decomposition.

References

[1] G.W. Stewart, "An Updating Algorithm for Subspace Tracking", IEEE Trans. on SP, 40 (1992), pp. 1535–1541.

[2] G.W. Stewart, "Updating a Rank-Revealing ULV Decomposition", SIAM J. Matrix Anal. and Appl., 14 (1993), pp. 494–499.

urv_up

Purpose

Updating a row in the rank-revealing URV decomposition.

Synopsis

```
[p,R,V,U,vec] = urv_up(p,R,V,U,a)
[p,R,V,U,vec] = urv_up(p,R,V,U,a,beta)
[p,R,V,U,vec] = urv_up(p,R,V,U,a,beta,tol_rank)
[p,R,V,U,vec] = urv_up(p,R,V,U,a,beta,tol_rank,tol_ref,max_ref)
[p,R,V,U,vec] = urv_up(p,R,V,U,a,beta,tol_rank,tol_ref,max_ref,fixed_rank)
```

Description

Given a rank-revealing URV decomposition of an m-by-n matrix A = U∗R∗V^T with m $\geq$ n, the function computes the updated decomposition

```
[beta*A] = U*R*V'
[  a   ]
```

where a is the new row added to A, and beta is a forgetting factor in [0;1], which is multiplied to existing rows to damp out the old data. Note that the row dimension of U will increase by one, and that the matrix U can be left out by inserting an empty matrix [].

Input Parameters

p	numerical rank of A revealed in R;
L, V, U	the URV factors such that A = U∗R∗V^T;
a	the new row added to A;
beta	forgetting factor in [0;1];
tol_rank	rank decision tolerance;
tol_ref	upper bound on the 2-norm of the off-diagonal block R(1:p,p+1:n) relative to the Frobenius-norm of R;
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	beta = 1;
	tol_rank = sqrt(n)∗norm(R,1)∗eps;
	tol_ref = 1e-04;
	max_ref = 0;

Output Parameters

p	numerical rank of [beta*A; a];
R, V, U	the URV factors such that [beta*A; a] = U*R*V^T;
vec	a 5-by-1 vector with: vec(1) = upper bound of norm(R(1:p,p+1:n)), 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

urv_dw	Downdating a row in the rank-revealing URV decomposition.
urv_win	Sliding window modification of the rank-revealing URV decomp.

References

[1] G.W. Stewart, "An Updating Algorithm for Subspace Tracking", IEEE Trans. on SP, 40 (1992), pp. 1535–1541.

[2] G.W. Stewart, "Updating a Rank-Revealing ULV Decomposition", SIAM J. Matrix Anal. and Appl., 14 (1993), pp. 494–499.

[3] M. Moonen, P. Van Dooren and J. Vandewalle, "A Note on Efficient Numerically Stabilized Rank-One Eigenstructure Updating", IEEE Trans. on Signal Processing, 39 (1991), pp. 1911–1913.

urv_win

Purpose

Sliding window modification of the rank-revealing URV decomp.

Synopsis

[p,R,V,U,vec] = urv_win(p,R,V,U,A,a)
[p,R,V,U,vec] = urv_win(p,R,V,U,A,a,alg_type)
[p,R,V,U,vec] = urv_win(p,R,V,U,A,a,alg_type,tol_rank)
[p,R,V,U,vec] = urv_win(p,R,V,U,A,a,alg_type,tol_rank,tol_ref,max_ref)
[p,R,V,U,vec] = urv_win(p,R,V,U,A,a,alg_type,tol_rank,tol_ref,max_ref,fixed_rank)

Description

Given a rank-revealing URV decomposition of an m-by-n matrix A = U∗R∗V^T (m ≥ n), the function computes the updated decomposition corresponding to the combined up- and down-dating action

```
A  ->  [ A ]  ->  [ w ]
       [ a ]      [ A ]
```

where a is a new row added to A, and w is the row that is downdated after the updating process. If U is not available, then insert the empty matrix [].

Input Parameters

p	numerical rank of A revealed in R;
R, V, U	the URV factors such that A = U∗R∗V^T;
A	m-by-n matrix (m ≥ n);
a	new row added to A;
alg_type	algorithm type (see Description of urv_dw);
tol_rank	rank decision tolerance;
tol_ref	upper bound on the 2-norm of the off-diagonal block R(1:p,p+1:n) relative to the Frobenius-norm of R;
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	alg_type = 3;
	tol_rank = sqrt(n)∗norm(R,1)∗eps;
	tol_ref = 1e-04;
	max_ref = 0;

Output Parameters

p	numerical rank of the modified A;
R, V, U	the URV factors such that the modified A = U*R*V^T;
vec	a 6-by-1 vector with: vec(1) = upper bound of norm(R(1:p,p+1:n)), 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. vec(6) = true if CSNE approach has been used.

See Also

ulv_win	Sliding window modification of the rank-revealing ULV decomp.

References

[1] G.W. Stewart, "An Updating Algorithm for Subspace Tracking", IEEE Trans. on SP, 40 (1992), pp. 1535–1541.

Bibliography

[1] G. Adams, M.F. Griffin, and G.W. Stewart, *Direction-of-Arrival Estimation Using the Rank-Revealing URV Decomposition*; in Proc. IEEE Internat. Conf. Acoustics, Speech, and Signal Processing, Washington, DC, 1991.

[2] J.L. Barlow and P.A. Yoon, *Solving Recursive TLS Problems Using the Rank-Revealing ULV Decomposition*; in S. Van Huffel (Ed.), *Recent Advances in Total Least Squares Techniques and Errors-In-Variables Modeling*, SIAM, Philadelphia, 1997, pp. 117–126.

[3] J.L. Barlow, P.A. Yoon, and H. Zha, *An Algorithm and a Stability Theory for Downdating the ULV Decomposition*, BIT, 36 (1996), pp. 15–40.

[4] M.W. Berry, S.T. Dumais, and G.W. O'Brien, *Using Linear Algebra for Intelligent Information Retrieval*, SIAM Review, 37 (1995), 573–595.

[5] E. Biglieri and K. Yao, *Some Properties of Singular Value Decomposition and Their Applications to Digital Signal Processing*, Signal Processing, 18 (1989), pp. 277–289.

[6] C.H. Bischof and G.M. Shroff, *On Updating Signal Subspaces*, IEEE Trans. Signal Processing, 40 (1992), pp. 96-105.

[7] Å. Björck, H. Park, and L. Eldén, *Accurate Downdating of Least Squares Solutions*, SIAM J. Matrix. Anal. Appl., 15 (1994), pp. 549–568.

[8] A.W. Bojanczyk and J.M. Lebak, *Downdating a ULLV Decomposition of Two Matrices*; in J.G. Lewis (Ed.), *Applied Linear Algebra*, SIAM, Philadelphia, 1994.

[9] J.R. Bunch and N.P. Nielsen, *Updating the Singular Value Decomposition*, Numer. Math., 31 (1978), pp. 111–129.

[10] T.F. Chan, *Rank Revealing QR Factorizations*, Lin. Alg. Appl., 88/89 (1987), pp. 67-82.

[11] T.F. Chan and P.C. Hansen, *Some Applications of the Rank Revealing QR Factorization*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 727-741.

[12] T.F. Chan & P.C. Hansen, *Low-Rank Revealing QR Factorizations*, Num. Lin. Alg. Appl., 1 (1994), 33-44.

[13] A.K. Cline, A.R. Conn, and C.F. Van Loan, *Generalizing the LINPACK Condition Estimator*; in J.P. Hennart (Ed.), *Numerical Analysis*, Lecture Notes In Mathematics, Vol. 909, Springer, Berlin, 1982.

[14] P. Comon and G.H. Golub, *Tracking a Few Extreme Singular Values and Vectors in Signal Processing*, Proc. IEEE, 78 (1990), pp. 1337–1343.

[15] J.W. Daniel, W.B. Gragg, L. Kaufman, and G.W. Stewart, *Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization*, Math. Comp., 30 (1976), pp. 772–795.

[16] B. De Moor, *Generalizations of the OSVD: Structure, Properties and Applications;* pp. 83–98 in [59].

[17] F. Deprettere, *SVD and Signal Processing, Algorithms, Applications, and Architectures*, North-Holland, Amsterdam, 1988.

[18] L. Eldén and E. Sjöström, *Fast Computation of the Principal Singular Vectors of Toeplitz Matrices Arising in Exponential Data Modelling*, Signal Proc., 50 (1996), pp. 151–164.

[19] R.D. Fierro, *Perturbation Analysis for Two-Sided (or Complete) Orthogonal Decompositions*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 383–400.

[20] R.D. Fierro and J.R. Bunch, *Bounding the Subspaces From Rank Revealing Two-Sided Orthogonal Decompositions*, SIAM Matrix Anal. Appl., 16 (1995), pp. 743–759.

[21] R.D. Fierro and P.C. Hansen, *Accuracy of TSVD Solutions Computed from Rank-Revealing Decompositions* , Numer. Math., 70 (1995), pp. 453–471.

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

[23] R.D. Fierro, L. Vanhamme, and S. Van Huffel, *Total Least Squares Algorithms Based on Rank-Revealing Complete Orthogonal Decompositions*; in S. Van Huffel (Ed.), *Recent Advances in Total Least Squares Techniques and Errors-In-Variables Modeling*, SIAM, Philadelphia, 1997, pp. 99–116.

[24] L. Foster, *Rank and Null Space Calculations Using Matrix Decomposition Without Column Interchanges*, Lin. Alg. Appl., 74 (1986), pp. 47–71.

[25] G.H. Golub, V. Klema, and G.W. Stewart, *Rank Degeneracy and Least Squares Problems*, Technical Report TR-456, Dept. of Computer Science, University of Maryland, Maryland, 1976.

[26] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3. Ed., Johns Hopkins University Press, 1996.

[27] M. Gu and S.C. Eisenstat, *Downdating the Singular Value Decomposition*, SIAM J. Matrix Anal. Appl, 16 (1995), pp. 793–810.

[28] P.C. Hansen, *The 2-Norm of Random Matrices*, J. Comput. Appl. Math., 23 (1988), pp. 117–120.

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

[30] P.C. Hansen, *Rank-Deficient Prewhitening by Quotient SVD and UTV*, BIT, 38 (1998), pp. 34–43.

[31] P.S.K. Hansen, *Signal Subspace Methods for Speech Enhancement*, Ph.D. Thesis, Dept. of Mathematical Modelling, Technical University of Denmark, 1997.

[32] N.J. Higham, *A Survey of Condition Number Estimation for Triangular Matrices*, SIAM Review, 29 (1987), pp. 575–596.

[33] S.H. Jensen, P.C. Hansen, S.D. Hansen, and J.Aa. Sørensen, *Reduction of Broad-Band Noise in Speech by Truncated QSVD*, IEEE Trans. Audio Speech Proc. 3 (1995), pp. 439–448.

[34] J. Kuczynski and H. Wozniakowski, *Estimating the Largest Eigenvalue by the Power and Lanczos Algorithms with a Random Start*, SIAM J. Matrix Anal., 4 (1992), pp. 1094–1122.

[35] C.L. Lawson and R.J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, N.J., 1974. Reprinted by SIAM, Philadelphia.

[36] J.M. Lebak and A.W. Bojanczyk, *Modifying a Rank-Revealing ULLV Decomposition*, Manuscript, School of Electrical Engineering, Cornell University, 1994.

[37] K.J.R. Liu, D.P. O'Leary, G.W. Stewart, and Y.-J. Wu, *URV ESPRIT for Tracking Time-Varying Signals*, IEEE Trans. Signal Proc., 42 (1994), pp. 3441–3448.

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

[39] F.T. Luk and S. Qiao, *An Adaptive Algorithm 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.

[40] W. Ma and J.P. Kruth, *Mathematical Modelling of Free-Form Curves and Surfaces from Discrete Points with NURBS*; in P.J. Laurent, A. Le Mehaute, and L.L Schumaker (Eds), *Curves and Surfaces in Geometric Design*, A.K. Peters, Ltd., Wellesley, Mass., 1994.

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

[42] M. Moonen and B. De Moor, *SVD and Signal Processing, III, Algorithms, Architectures and Applications*, Elsevier, Amsterdam, 1995.

[43] M. Moonen, P. Van Dooren, and J. Vandewalle, *A Note on Efficient Numerically Stabilized Rank-One Eigenstructure Updating,* IEEE Trans. Signal Proc. 39 (1991), 1911–1913.

[44] M. Moonen, P. Van Dooren, and J. Vandewalle, *A Singular Value Decomposition Updating Algorithm for Subspace Tracking*, SIAM J. Matrix Anal. Appl., 13 (1992), 1015–1038.

[45] H. Park and L. Eldén, *Downdating the Rank Revealing URV Decomposition*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 138–155.

[46] H. Park, S. Van Huffel, and L. Eldén, *Fast Algorithms for Exponential Data Modeling*, Proc. 1994 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), April 19–22, Adelaïde, Australia, Vol. 4, pp. 25–28, 1994.

[47] D.J. Pierce and J.G. Lewis, *Sparse Multifrontal Rank Revealing QR Factorization*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 159–180.

[48] M.A. Rahman and K. Yu, *Total Least Squares Approach for Frequency Estimation Using Linear Prediction*, IEEE Trans. ASSP, 35 (1987), pp. 1442-54.

[49] L.L. Scharf, *The SVD and Reduced Rank Signal Processing*, Signal Proc., 25 (1991), pp. 113–133.

[50] M. Stewart and P. Van Dooren, *Updating a Generalized URV Decomposition*, SIAM J. Matrix Anal. Appl., to appear.

[51] G.W. Stewart, *Rank Degeneracy*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 403-413.

[52] G.W. Stewart, *An Updating Algorithm for Subspace Tracking*, IEEE Trans. Signal Processing, 40 (1992), pp. 1535-1541.

[53] G.W. Stewart, *Updating a Rank-Revealing ULV Decomposition*, SIAM J. matrix Anal. Appl., 14 (1993), pp. 494-499.

[54] G.W. Stewart, *Determining Rank in the Presence of Error*; in M.S. Moonen, G.H. Golub, and B.L.R. DeMoor (Eds.), *Linear Algebra for Large Scale and Real-Time Applications*, Kluwer Academic Publishers, 1993, pp. 275–292.

[55] G.W. Stewart, *UTV Decompositions*; in D.F. Griffith and G.A. Watson (Eds), *Numerical Analysis, 1993*, Pitman Research Notes in Mathematical Sciences, New York, 1994.

[56] G.W. Stewart, *A Gap-Revealing Matrix Decomposition*, Report TR-3771, Dept. of Computer Science, University of Maryland, 1997. To appear in SIAM J. Sci. Comput.

[57] G.W. Stewart, *Matrix Algorithms. Volume I: Basic Decompositions*, SIAM, Philadelphia, 1998.

[58] D.W. Tufts and R. Kumaresan, *Estimation of Frequencies of Multiple Sinusoids: Making Linear Prediction Perform Like Maximum Likelihood*, Proc. IEEE, 70 (1982), pp. 975–989.

[59] R. Vaccaro, *SVD and Signal Processing, II, Algorithms, Analysis and Applications*, Elsevier, Amsterdam, 1991.

[60] R.J. Vaccaro, D.W. Tufts, and G.F. Boudreaux-Bartels, *Advances in Principal Component Signal Processing*; pp. 115–146 in [17]

[61] A. van der Veen and E.F. Deprettere, *SVD-Based Low-Rank Approximations of Rational Models*; pp. 431–454 in [59].

[62] S. Van Huffel and H. Zha, *An Efficient Total Least Squares Algorithm Based on a Rank Revealing Two-Sided Orthogonal Decomposition*, Numerical Algorithms, 4 (1993), pp. 101–133.

[63] G. Xu and T. Kailath, *Fast Estimation of Principle Eigenspace Using Lanczos Algorithm*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 974–994.

[64] G. Xu, H. Zha, G.H. Golub, and T. Kailath, *Fast and Robust Algorithms for Updating Signal Subspaces*, IEEE Trans. Circuits and Systems, 41 (1994), pp. 537–549.

[65] P.A. Yoon and J.L. Barlow, *An Efficient Rank Detection Procedure for Modifying the ULV Decomposition*, BIT, 38 (1998), pp. 781–801.