UTV Tools
Matlab Templates for Rank-Revealing UTV Decompositions

Fierro, Ricardo D.; Hansen, Per Christian; Hansen, Peter Søren Kirk

Published in:
Numerical Algorithms

Link to article, DOI:
10.1023/A:1019112103049

Publication date:
1999

Document Version
Publisher’s PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
UTV Tools: Matlab templates for rank-revealing UTV decompositions *

Ricardo D. Fierro a, Per Christian Hansen b and Peter Søren Kirk Hansen b

a Department of Mathematics, California State University at San Marcos, San Marcos, CA 92096, USA
E-mail: fierro@thunder.csusm.edu

b Department of Mathematical Modelling, Building 321, Technical University of Denmark,
DK-2800 Lyngby, Denmark
E-mail: {pch; pskh}@imm.dtu.dk

We describe a Matlab 5.2 package for computing and modifying certain rank-revealing decompositions that have found widespread use in signal processing and other applications. The package focuses on algorithms for URV and ULV decompositions, collectively known as UTV decompositions. We include algorithms for the ULLV decomposition, which generalizes the ULV decomposition to a pair of matrices. For completeness a few algorithms for computation of the RRQR decomposition are also included. The software in this package can be used as is, or can be considered as templates for specialized implementations on signal processors and similar dedicated hardware platforms.

Keywords: rank-revealing decompositions, rank deficiency, numerical rank, up- and down-dating, Matlab

AMS subject classification: 65F25, 65F20

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, section 2.5; 56, section 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

* This work was supported in part by NATO Collaborative Research Grant No. 951327.

© J.C. Baltzer AG, Science Publishers
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 impractical 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, section 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 [54; 56, section 5.4] or complete orthogonal decompositions [26, section 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, section 12.5.5; 45; 51; 52]. Some applications of UTV decompositions can be found in [1,2,23,37,46,52].

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 so-called ULLV decomposition due to Luk and Qiao [38] reveals the numerical rank of the matrix “quotient” \( AB^\dagger \) (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. The package for Matlab 5.2, as well as the accompanying manual, is available from Netlib.

Our notation is standard linear algebra notation plus Matlab-style matrix indexing where, e.g., $A(1:k, 1:k)$ 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 are always given. We conclude with a brief overview of the 46 functions included in the package in section 6.

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 easily be 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, section 3.1]. Hence, rank-revealing decompositions may also be labeled “gap-revealing decompositions”, a phrase coined by Stewart [55]. General treatments of rank-revealing decompositions are presented in [29, chapter 3] and [56, chapter 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 $A = XMY^T$, 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, section 2.5]. The SVD of an $m \times n$ matrix $A$ with $m \geq n$ is given by

\[ A = U \Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T, \]

where $U_1 = U(:, 1:n)$ and $r = \text{rank}(A)$. 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 $\sigma_i$ of the $n \times n$ diagonal matrix $\Sigma$ 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 $A^TA$, respectively.

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

\[ A = (U_k, U_0, U_\perp) \left( \begin{array}{cc} \Sigma_k & 0 \\ 0 & \Sigma_0 \end{array} \right) (V_k, V_0)^T, \]

where $\Sigma_k = \text{diag}(\sigma_1, \ldots, \sigma_k)$ and $\Sigma_0 = \text{diag}(\sigma_{k+1}, \ldots, \sigma_n)$ 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, section 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 $\tau$, 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 $\tau$. 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 $\tau$ reflects the noise level, and $k$ is related to the number of prominent signals. Moreover, the matrices $U_k$, $\Sigma_k$, and $V_k$ carry information about the pure signal (plus some noise), while $(U_0, U_\perp)$, $\Sigma_0$, and $V_0$ carry information solely about the noise. Some issues concerning the very important subproblem of selecting $\tau$ for numerical rank detection are discussed in [28,50,53].

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

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

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 $\Sigma$ 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 $\Sigma$ 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

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

where $R_k$ is a $k \times k$ nonsingular 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

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

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

$$A = U_L \begin{pmatrix} L_k \\ 0 \end{pmatrix} V_L^T = (U_{Lk}, U_{L0}, U_{L\perp}) \begin{pmatrix} L_k \\ H \\ 0 \end{pmatrix} \begin{pmatrix} V_{Lk}, V_{L0} \end{pmatrix}^T,$$
where \( L_k \) is a \( k \times k \) nonsingular 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

\[
\sigma_{\min}(L_k) = O(\sigma_k) \quad \text{and} \quad \|(H, E)\|_2 = O(\sigma_{k+1}).
\] (6)

Although decompositions of the form (3) and (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 (4) and (6) are more recent. From the standard perturbation theory for singular values, cf. [26, section 8.6.1], it follows that the smaller the norm of the off-diagonal block, the better the approximations in (4) and (6).

We mention that there are situations where \( \sigma_k \) and \( \sigma_{k+1} \) are not well separated but \( \Sigma_k, U_k \), 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, section 2.6.3] for more information about subspace distances).

**Theorem 1** (See [20, corollaries 2.3 and 2.5]). Let \( A \) have the UTV decompositions as in (3) and (5) and the SVD as in (1). If \( \sigma_{\min}(R_k) > \|G\|_2 \), then

\[
\text{dist}(\mathcal{R}(U_k), \mathcal{R}(U_{Rk})) \leq \frac{\|F\|_2 \|G\|_2}{\sigma_{\min}(R_k)^2 - \|G\|_2^2}
\] (7)

and

\[
\frac{\|F\|_2}{2 \|R\|_2} \leq \text{dist}(\mathcal{R}(V_0), \mathcal{R}(V_{R0})) \leq \frac{\sigma_{\min}(R_k) \|F\|_2}{\sigma_{\min}(R_k)^2 - \|G\|_2^2}.
\] (8)

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

\[
\frac{\|H\|_2}{2 \|L\|_2} \leq \text{dist}(\mathcal{R}(U_k), \mathcal{R}(U_{Lk})) \leq \frac{\sigma_{\min}(L_k) \|H\|_2}{\sigma_{\min}(L_k)^2 - \|E\|_2^2}
\] (9)

and

\[
\text{dist}(\mathcal{R}(V_0), \mathcal{R}(V_{L0})) \leq \frac{\|H\|_2 \|E\|_2}{\sigma_{\min}(L_k)^2 - \|E\|_2^2}.
\] (10)

The \textit{a posteriori} bounds in (7)-(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 \( R(U_k) \), has a smaller upper bound than the corresponding ULV-matrix \( U_{Lk} \), due to the factor \( \|F\|_2 \) instead of the factor \( \min(L_k) \). On the other hand, the ULV-matrix \( V_{L0} \), considered as an approximate basis for the numerical null space \( R(V_0) \), 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 \( \hat{A} \) plus additive noise, where \( \hat{A} \) 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 \( \hat{A} \) which, in turn, are identical to the fundamental SVD-based subspaces of \( \hat{A} \). 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

\[
x_{Rk} = V_{Rk} R_{k}^{-1} U_{Rk}^T b \quad \text{and} \quad x_{Lk} = V_{Lk} L_{k}^{-1} U_{Lk}^T b,
\]

and the accuracy of these solutions is investigated in [21]. The computation of truncated UTV solutions is implemented in the two Matlab functions \( tuv \) and \( turv \) for computing \( x_{Lk} \) and \( x_{Rk} \), 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 \( \Pi \). It is customary to write this decomposition in the form

\[
A \Pi = Q \begin{pmatrix} R_0 & 0 \\ 0 & 0 \end{pmatrix} = (Q_1, Q_2, Q_\bot) \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}, \quad \Pi = (\Pi_1, \Pi_2).
\]

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. [10,12,24]. The corresponding truncated RRQR solution is given by \( x_{Qk} = \Pi_1(R_{11}, R_{12})^T Q_1^T b \), 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 \( \Pi \), return a matrix \( W \) whose columns span an approximation to either \( N(A_k) \), in the high-rank case, or \( R(A_k^T) \), 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 $8 \times 6$ matrix $A$ with singular values

$$2, 1, 0.5, 0.2, 0.005, 0.001,$$

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 URV decomposition of $A$. The computed triangular factor $R$ has the form

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

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:

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

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

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

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

and

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

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

both revealing the numerical rank of \(A\). The ULV subspace distances and their upper bounds from theorem 1 are

\[
\begin{align*}
\text{dist}(\mathcal{R}(U_k), \mathcal{R}(U_{Lk})) &= 8.50 \cdot 10^{-4}, \\
\frac{\sigma_{\min}(L_k) \|H\|_2}{\sigma_{\min}(L_k)^2 - \|E\|_2^2} &= 8.99 \cdot 10^{-4}, \\
\text{dist}(\mathcal{R}(V_0), \mathcal{R}(V_{L0})) &= 2.10 \cdot 10^{-5}, \\
\frac{\|H\|_2 \|E\|_2}{\sigma_{\min}(L_k)^2 - \|E\|_2^2} &= 2.25 \cdot 10^{-5},
\end{align*}
\]

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

\[
\begin{align*}
\text{dist}(\mathcal{R}(U_k), \mathcal{R}(Q_1)) &= 5.29 \cdot 10^{-3}, \\
\text{dist}(\mathcal{R}(V_0), \mathcal{R}(W)) &= 6.78 \cdot 10^{-4},
\end{align*}
\]

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 [51,52] 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. The generic high-rank ULV algorithm for the case $m \geq n$ can be summarized as follows ($\tau$ denotes the threshold used in determining the numerical rank):

**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 $\tilde{\sigma}_k$ estimate $\sigma_{\min}(L(1:k,1:k))$ and let $w_k$ estimate the corresponding left singular vector.
3. If $\tilde{\sigma}_k > \tau$ then exit.
4. Revealment: determine an orthogonal $P_k$ such that $P_k w_k = (0, \ldots, 0, 1)^T$;
5. update $L(1:k,1:k) \leftarrow P_k^T L(1:k,1:k)$;
6. update $L(1:k,1:k) \leftarrow L(1:k,1:k) Q_k$, where the orthogonal matrix $Q_k$ is chosen such that the updated $L$ is triangular.
7. Refinement (optional): while $\|L(k,1:k-1)\|_2 > \delta \|L\|_F$ apply QR-refinement to the bottom row of $L(1:k,1:k)$.
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 $L(1:k,1:k)$.

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 et al. [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 $F$, see 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 $w_k^0 \leftarrow k^{-1/2} \cdot \text{ones}(k, 1)$.
3. Condition estimation: let $(\tilde{\sigma}_k, w_k) \leftarrow \text{inviter}(L(1: k, 1: k), w_k^0)$.
4. If $\tilde{\sigma}_k > \tau$ then **exit**.
5. Revealm (similar to steps 4–6 in Stewart’s algorithm).
6. Refinement (optional): if $\|L(k, 1: k - 1)\|_2 > \delta\|L\|_F$
7. let $w_k^0 \leftarrow (0, \ldots, 0, 1)^T$,
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 $\hat{\sigma}_k$ and $w_k$ are computed by means of inverse iteration with starting vector $w_0^0$. 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 $\|L(k, 1:k - 1)\|_2$ after the revealment step is sufficiently small. Unfortunately, the only available a priori bound is $\|L(k, 1:k - 1)\|_2 < \hat{\sigma}_k$, 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 orthogonal matrix $P_k$ is chosen such that $P_kw_k = (1, 0, \ldots, 0)$. If again $\tau$ 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 $A = U_1L$ with a lower triangular $L$.
2. Let $w_0^0 \leftarrow (n - k + 1)^{-1/2} \cdot \text{ones}(n - k + 1, 1)$ and $\ell = 1$.
3. Norm estimation: $(\hat{\sigma}_k, w_k) \leftarrow \text{normest}(L(k:n, k:n), w_0^0)$.
4. If $\hat{\sigma}_k < \tau$ then $k \leftarrow k - 1$ **exit**.
5. Revealment – as explained in the text above.
6. Refinement (optional): if $\ell < \ell_{\text{max}}$
7. \hspace{1em} let $w_0^\ell \leftarrow (1, 0, \ldots, 0)^T$ and $\ell = \ell + 1$.
8. \hspace{1em} 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 $\tilde{\sigma}_k$ and $w_k$ are computed by means of a 2-norm estimator with starting vector $w_0$. 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, section 8.2] or by means of Lanczos bidiagonalization [26, section 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., $L(k + 1:n, 1:k)$ 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 leads to the definition of an alternative form of the UTV decomposition in which the initial triangularization is omitted, and the final 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 ma-
trices $U$ 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 $\text{ulvqr}$ and $\text{urvqr}$.

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 $L(1:k,1:k)$ – or the last column of $R(1:k,1:k)$ – 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, thus obtaining a reduction of the off-diagonal block’s norm.

A different flavor of refinement is used in Stewart’s PLQ decomposition [55], 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 $8 \times 6$ matrix $A$ with singular values

$$0.3, 0.2, 0.05, 0.03, 0.02, 0.01,$$

and the numerical rank of $A$ with respect to the threshold $\tau = 0.1$ is $k = 2$. Then we use the Matlab function $\text{lulv}$ 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 1 where we use the notation

$$d(U_k) = \text{dist}(\mathcal{R}(U_k), \mathcal{R}(U_{Rk})), \quad d(V_0) = \text{dist}(\mathcal{R}(V_0), \mathcal{R}(V_{R0})).$$
Table 1
Results from the low-rank ULV algorithm $\text{ulv}(A,0.1,\text{power\_its})$.

<table>
<thead>
<tr>
<th>Power iterations</th>
<th>$|H|_2$</th>
<th>$d(U_k)$</th>
<th>$d(V_0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$3.4 \times 10^{-2}$</td>
<td>$1.6 \times 10^{-1}$</td>
<td>$3.7 \times 10^{-2}$</td>
</tr>
<tr>
<td>2</td>
<td>$1.8 \times 10^{-3}$</td>
<td>$9.2 \times 10^{-3}$</td>
<td>$2.3 \times 10^{-3}$</td>
</tr>
<tr>
<td>3</td>
<td>$1.1 \times 10^{-4}$</td>
<td>$5.6 \times 10^{-4}$</td>
<td>$1.4 \times 10^{-4}$</td>
</tr>
<tr>
<td>4</td>
<td>$6.5 \times 10^{-6}$</td>
<td>$3.4 \times 10^{-5}$</td>
<td>$8.6 \times 10^{-6}$</td>
</tr>
</tbody>
</table>

Table 2
Results from the low-rank ULV algorithm $\text{ulv}(A,0.1,1,\text{ref\_steps})$.

<table>
<thead>
<tr>
<th>Refinement steps</th>
<th>$|H|_2$</th>
<th>$d(U_k)$</th>
<th>$d(V_0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$3.4 \times 10^{-2}$</td>
<td>$1.6 \times 10^{-1}$</td>
<td>$3.7 \times 10^{-2}$</td>
</tr>
<tr>
<td>1</td>
<td>$2.0 \times 10^{-3}$</td>
<td>$1.1 \times 10^{-2}$</td>
<td>$2.6 \times 10^{-3}$</td>
</tr>
<tr>
<td>2</td>
<td>$5.1 \times 10^{-5}$</td>
<td>$2.7 \times 10^{-4}$</td>
<td>$6.8 \times 10^{-5}$</td>
</tr>
<tr>
<td>3</td>
<td>$3.0 \times 10^{-6}$</td>
<td>$1.6 \times 10^{-5}$</td>
<td>$4.0 \times 10^{-6}$</td>
</tr>
<tr>
<td>4</td>
<td>$1.8 \times 10^{-7}$</td>
<td>$9.7 \times 10^{-7}$</td>
<td>$2.4 \times 10^{-7}$</td>
</tr>
</tbody>
</table>

Table 3
Results from the ULV refinement algorithm using block QR iterations $\text{ulv\_qrit}(k,\text{block\_its},L,V,U)$.

<table>
<thead>
<tr>
<th>Block iterations</th>
<th>$|H|_2$</th>
<th>$d(U_k)$</th>
<th>$d(V_0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$3.4 \times 10^{-2}$</td>
<td>$1.6 \times 10^{-1}$</td>
<td>$3.7 \times 10^{-2}$</td>
</tr>
<tr>
<td>1</td>
<td>$1.7 \times 10^{-3}$</td>
<td>$8.9 \times 10^{-3}$</td>
<td>$2.2 \times 10^{-3}$</td>
</tr>
<tr>
<td>2</td>
<td>$1.0 \times 10^{-4}$</td>
<td>$5.4 \times 10^{-4}$</td>
<td>$1.3 \times 10^{-4}$</td>
</tr>
<tr>
<td>3</td>
<td>$6.3 \times 10^{-6}$</td>
<td>$3.4 \times 10^{-5}$</td>
<td>$8.4 \times 10^{-6}$</td>
</tr>
<tr>
<td>4</td>
<td>$3.9 \times 10^{-7}$</td>
<td>$2.1 \times 10^{-6}$</td>
<td>$5.2 \times 10^{-7}$</td>
</tr>
</tbody>
</table>

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 $\text{ulv}$ 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 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 $\text{ulv\_qrit}$) to the matrix $L$ computed with one power iteration and no refinement in each stage; cf. table 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.
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 $\beta \leq 1$ applied to $A$ (which is a standard operation in signal processing), i.e., given the UTV decomposition $A = UTV^T$, where $T$ is either $L$ or $R$, we want to compute the new UTV decomposition $\beta A \\
\begin{pmatrix}
\beta A \\

w^T
\end{pmatrix} = UTV^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^TV \end{pmatrix} V^T,
$$

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

$$
\begin{pmatrix}
\beta A \\

w^T
\end{pmatrix} = \begin{pmatrix} U_1 & 0 \\
0 & 1 \end{pmatrix} G \begin{pmatrix} T \\
0 \end{pmatrix} H^TV^T = (U_1u_2) \begin{pmatrix} T \\
0 \end{pmatrix} V^T = UTV^T.
$$

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 [51] 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, section 12.5.5] or [51] for more details.

The updating algorithm is implemented in the Matlab functions `ulvup` and `urvup`, and after all the Givens rotations have been applied, we normalize the columns of $V$ as recommended 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 $A = U_1TV^T$, where $T$ is either $L$ or $R$, compute the new UTV decomposition of the $(m-1) \times n$ submatrix $A(2:m,:)$, i.e.,

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

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

$$A = (U_1, u_2) \begin{pmatrix} T & 0^T \\ 0 & 1 \end{pmatrix} V^T.$$

If $T = R$, then we use the standard algorithm for downdating a QR factorization, cf. [26, section 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:

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

where the $(m-1) \times n$ matrix $\mathcal{U}$ has orthonormal columns. Then it follows immediately that the three matrices $\mathcal{U}, R, \mathcal{V}$ 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:

$$U_1G = \tilde{U} = \begin{pmatrix} v e_1^T \\ \tilde{U}(2:m,:) \end{pmatrix}, \quad G^T LH = \tilde{L}, \quad \tilde{V} = VH,$$

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

$$(\tilde{U}, u_2)\tilde{G} = \begin{pmatrix} 0^T \\ \mathcal{U} \end{pmatrix} \pm 1 \quad \text{and} \quad \tilde{G}^T \begin{pmatrix} \tilde{L} \\ 0^T \end{pmatrix} = \begin{pmatrix} \mathcal{L} \\ 0^T \end{pmatrix}.$$  

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

$$q = (T^T)^{-1} V^T w,$$

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 safer 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 et al. [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 $\text{rank}(A(2: m,:)) < \text{rank}(A)$, then

$$e_1 \in \mathcal{R}(U_1) \iff \|q\|_2 = 1.$$  

**Proof.** Since $\text{rank}(T) = \text{rank}(A)$, and since $\text{rank}(U_1(2: m,:)^T) = \text{rank}(A(2: m,:)) = \text{rank}(A) - 1$, we conclude that $U_1(2: m,:)^T$ must be rank-deficient. Now, from the CS decomposition [26, section 2.6.4] of $U_1$ it is clear that $\|q\|_2 = 1 \iff \text{rank}(U_1(2: m,:)^T) = n - 1$. Hence, $\text{rank}(A(2: m,:)) < \text{rank}(A)$ 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 $U_1$. \qed

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 considering 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 $e_1$ is 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 $V$ 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 $6 \times 4$ matrix such that

\[
\text{svd}(A) = (2.08, 1.03, 0.21, 1.24 \cdot 10^{-16}),
\]

\[
\text{svd}(A(2: m, :)) = (2, 1, 1.96 \cdot 10^{-16}, 8.11 \cdot 10^{-17}),
\]

and

\[
\text{svd}(U_1(2: m, :)) = (1, 1, 1, 2.41 \cdot 10^{-16}).
\]

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 $j = 1: n$, $z \leftarrow z - U_1(:, j)(U_1(:, j)^T z)$; end
2. If $\|z\|_2 < 2^{-1/2}$ then
3. for $j = 1: n$, $z \leftarrow z - U_1(:, j)(U_1(:, j)^T z)$; 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 (11). 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 $A(2:m,:)$ become somewhat larger than the machine precision ($8.49 \cdot 10^{-7}$ and $10^{-9}$, 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, section 8.7.3], which yields information about the numerical rank and numerical subspaces of the matrix “quotient” $AB^{-1}$ when $B$ is invertible, and $AB^\dagger$ (where $B^\dagger$ is the pseudoinverse [26, section 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 the 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 as $A$ and $B$ have the same number of columns and $B$ has full column rank, the matrix quotient $AB^\dagger$ 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

$$A = U(ALV)^T, \quad B = UBLV^T,$$

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

\[
AB^\dagger = U_A L_A U_B^T.
\]

We see that the three matrices \( U_A, L_A, \) and \( U_B \) form the ULLV decomposition of \( AB^\dagger \), and this decomposition can always be made to reveal the numerical rank of \( AB^\dagger \) by means of ULV revealment steps. When this is the case, we say that the ULLV decomposition (13) 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 \( \text{rank}(B) = p < n \). The corresponding quotient ULV decomposition then takes the form

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

where \( L \) is now \( p \times p \); for more details about this version and its application in interference problems, see [30,39]. Other generalized UTV decompositions are discussed in [57] (matrix quotients of the form \( A^{-1} B \)) and [8] (a decomposition of the form \( A = U_A L_A V^T, \ B = U_B L_B V^T \)).

We note that the approximation bounds in theorem 1 immediately carry over to the numerical subspaces of the matrix quotient \( AB^\dagger \), 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 be derived from the low-rank ULV implementations luv and luv.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, \( Z = AL^{-1} \)).
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 $AB^\dagger$. Note that some of the Givens rotations in these steps are also 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 (13) 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 $\beta$ is included for completeness):

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

Now left and right Givens rotations are used to annihilate all but the leftmost element of the row $w^T V$ and, simultaneously, to 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

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

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

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

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

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

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

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

where $\tilde{z}$ is the vector of the leftmost element of the row $w^T V$ and its corresponding column in $L$. The transformation $Y$ is repeated for each row appended to $A$ and $B$.
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 $\tilde{L}$, $\tilde{L}_A$, and $I_n$. At this second intermediate stage, we have

$$
\begin{pmatrix}
\beta A
\end{pmatrix}_w =
\begin{pmatrix}
\tilde{U}_A \\
\eta e_1^T
\end{pmatrix}
\begin{pmatrix}
\tilde{L}_A^T \\
1
\end{pmatrix}
\tilde{L}\tilde{V}^T,
B = \tilde{U}_B\tilde{L}\tilde{V}^T,
$$

and now $\eta$ 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:

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

Hence, we arrive at

$$
\begin{pmatrix}
\beta A
\end{pmatrix}_w = \tilde{U}_A\tilde{L}_A\tilde{L}\tilde{V}^T,
B = \tilde{U}_B\tilde{L}\tilde{V}^T.
$$

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 (13) 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 [3,45], 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 $(U_A, u_2)$ is one. Then we formally write the ULLV decomposition as

$$
A = (U_A, u_2)
\begin{pmatrix}
L_A \\
0
\end{pmatrix}
\begin{pmatrix}
L \\
0
\end{pmatrix}^T V^T,
B = U_B(I_n, -\eta e_1)
\begin{pmatrix}
L \\
0
\end{pmatrix}^T V^T,
$$

where $\eta$ 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

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

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

$$\begin{pmatrix} \bar{v}e_1^T & u_2(1) \\ \bar{U}_A & u_2(2:m) \end{pmatrix} \begin{pmatrix} \tilde{L}_A & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0^T & \pm 1 \\ \overline{U}_A & 0 \end{pmatrix} \begin{pmatrix} \bar{T}_A & -s e_1 \\ \eta e_1^T & c \end{pmatrix}.$$ 

This relation defines the quantity $\eta$ used in the augmented expression for $B$. Finally, we apply the scaled transformation $Y$ (14) 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 $\eta$, and we obtain

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

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

We thus arrive at the expressions

$$A(2:m,:) = \overline{U}_A \bar{T}_A \tilde{L} \tilde{V}^T, \quad B = \tilde{U}_B \tilde{L} \tilde{V}^T.$$ 

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_dwa 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 $8 \times 5$ 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

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

\[
L = \begin{pmatrix}
0.45 & 0.06 & 0.62 \\
-2.40 & 0.38 & 2.53 \\
0.45 & 0.05 & -0.74 & 1.07 \\
-0.19 & 0.11 & -0.04 & -0.12 & 1.01
\end{pmatrix}.
\]

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

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

\[
\sigma_1 = 10, \quad \sigma_2 = 7, \quad \sigma_3 = 4, \quad \sigma_4 = 0.4, \quad \sigma_5 = 0.2.
\]

There is obviously a cluster of three large singular values, i.e., the rank is 3 with respect to the threshold \( \tau = 1 \). The \( 8 \times 5 \) 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 \( AB^\dagger \) are

\[
\sigma_1 = 10, \quad \sigma_2 = 5, \quad \sigma_3 = 0.5, \quad \sigma_4 = 0.2, \quad \sigma_5 = 0.1,
\]

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 2 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 \texttt{hulv} we compute \( A \)'s lower triangular factor in the ULV decomposition,

\[
\begin{pmatrix}
5.14 & 0.00 & 8.09 \\
0.87 & -1.48 & 6.73 \\
-9.5 \times 10^{-4} & 5.1 \times 10^{-4} & 9.7 \times 10^{-4} & 0.40 \\
-3.7 \times 10^{-4} & 2.2 \times 10^{-4} & 1.7 \times 10^{-4} & 3.2 \times 10^{-2} & 0.20
\end{pmatrix},
\]

and by means of \texttt{ullv} we compute the following \( L_A \)-factor in the ULLV decomposition:

\[
L_A = \begin{pmatrix}
5.13 & 0.00 & 9.74 \\
-1.2 \times 10^{-3} & 4.1 \times 10^{-3} & 0.50 \\
1.9 \times 10^{-5} & 2.3 \times 10^{-4} & 1.0 \times 10^{-2} & 0.19 \\
-8.6 \times 10^{-5} & 3.7 \times 10^{-5} & 2.6 \times 10^{-3} & -4.3 \times 10^{-2} & 0.10
\end{pmatrix}.
\]

Both triangular factors reveal the correct numerical rank of \( A \) and \( AB^\dagger \), respectively.

6. Overview of software

The following eight tables summarize the 46 Matlab 5.2 functions included in the package, of which details are given in the manual that accompanies the software.
UTV-based solvers

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tulv</td>
<td>Solves a numerically rank-deficient least squares problem using the rank-revealing ULV decomposition.</td>
</tr>
<tr>
<td>turv</td>
<td>Similar to tulv, except it uses the rank-revealing URV decomposition.</td>
</tr>
</tbody>
</table>

High-rank UTV algorithms

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hulv</td>
<td>Stewart’s rank-revealing ULV algorithm.</td>
</tr>
<tr>
<td>hulv_a</td>
<td>The alternative rank-revealing ULV algorithm.</td>
</tr>
<tr>
<td>hurv</td>
<td>Stewart’s rank-revealing URV algorithm.</td>
</tr>
<tr>
<td>hurv_a</td>
<td>The alternative rank-revealing URV algorithm.</td>
</tr>
</tbody>
</table>

Low-rank UTV algorithms

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lulv</td>
<td>Warm-started rank-revealing ULV algorithm.</td>
</tr>
<tr>
<td>lurv</td>
<td>Warm-started rank-revealing URV algorithm.</td>
</tr>
<tr>
<td>lulv_a</td>
<td>Cold-started rank-revealing ULV algorithm.</td>
</tr>
<tr>
<td>lurv_a</td>
<td>Cold-started rank-revealing URV algorithm.</td>
</tr>
</tbody>
</table>

Block QR refinement

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ulv_qrit</td>
<td>Refinement of $L$ in the ULV decomposition.</td>
</tr>
<tr>
<td>urv_qrit</td>
<td>Refinement of $R$ in the URV decomposition.</td>
</tr>
</tbody>
</table>

UTV up- and downdating

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ulv_dw</td>
<td>Downdate the rank-revealing ULV decomposition.</td>
</tr>
<tr>
<td>ulv_up</td>
<td>Update the rank-revealing ULV decomposition.</td>
</tr>
<tr>
<td>ulv_win</td>
<td>Sliding window modification of the rank-revealing ULV decomposition.</td>
</tr>
<tr>
<td>urv_dw</td>
<td>Downdate the rank-revealing URV decomposition.</td>
</tr>
<tr>
<td>urv_up</td>
<td>Update the rank-revealing URV decomposition.</td>
</tr>
<tr>
<td>urv_win</td>
<td>Sliding window modification of the rank-revealing URV decomposition.</td>
</tr>
</tbody>
</table>

ULLV algorithms

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ullv</td>
<td>Compute a high-rank revealing ULLV decomposition.</td>
</tr>
<tr>
<td>ullv_dw_a</td>
<td>Downdate $A$ in the rank-revealing ULLV decomposition.</td>
</tr>
<tr>
<td>ullv_dw_b</td>
<td>Downdate $B$ in the rank-revealing ULLV decomposition.</td>
</tr>
<tr>
<td>ullv_up_a</td>
<td>Update $A$ in the rank-revealing ULLV decomposition.</td>
</tr>
<tr>
<td>ullv_up_b</td>
<td>Update $B$ in the rank-revealing ULLV decomposition.</td>
</tr>
</tbody>
</table>

RRQR algorithms

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hrrqr</td>
<td>Chan–Foster high-rank RRQR algorithm.</td>
</tr>
<tr>
<td>lrrqr</td>
<td>Chan–Hansen low-rank RRQR algorithm.</td>
</tr>
<tr>
<td>trrqr</td>
<td>Solves a numerically rank-deficient least squares problem using the RRQR decomposition.</td>
</tr>
</tbody>
</table>
Misc. tools

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>app_giv</code></td>
<td>Apply a Givens rotation (from the left or right).</td>
</tr>
<tr>
<td><code>app_hous</code></td>
<td>Apply a Householder reflection (from the left or right).</td>
</tr>
<tr>
<td><code>ccvl</code></td>
<td>Estimation of the smallest singular value via the Cline–Conn–Van Loan algorithm.</td>
</tr>
<tr>
<td><code>gen_giv</code></td>
<td>Determine a $2 \times 2$ Givens rotation matrix.</td>
</tr>
<tr>
<td><code>gen_hous</code></td>
<td>Determine a Householder reflection matrix.</td>
</tr>
<tr>
<td><code>inviter</code></td>
<td>Estimation of the smallest singular value via inverse iteration.</td>
</tr>
<tr>
<td><code>lanczos</code></td>
<td>Estimation of the largest singular value via Lanczos bidiagonalization.</td>
</tr>
<tr>
<td><code>mgdr</code></td>
<td>Modified Gram–Schmidt expansion step with reorthogonalization.</td>
</tr>
<tr>
<td><code>powiter</code></td>
<td>Estimation of the largest singular value via the power method.</td>
</tr>
<tr>
<td><code>ullv.csne</code></td>
<td>Corrected semi-normal equations expansion step (for ULLV).</td>
</tr>
<tr>
<td><code>ullv.rdef</code></td>
<td>Deflate one row of $L_A$ in the ULLV decomposition.</td>
</tr>
<tr>
<td><code>ullv.ref</code></td>
<td>Refine one row of $L_A$ in the ULLV decomposition.</td>
</tr>
<tr>
<td><code>ulv.cdef</code></td>
<td>Deflate one column of $L$ in the ULV decomposition.</td>
</tr>
<tr>
<td><code>ulv.csne</code></td>
<td>Corrected semi-normal equations expansion step (for ULV).</td>
</tr>
<tr>
<td><code>ulv.rdef</code></td>
<td>Deflate one row of $L$ in the ULV decomposition.</td>
</tr>
<tr>
<td><code>ulv.ref</code></td>
<td>Refine one row of $L$ in the ULV decomposition.</td>
</tr>
<tr>
<td><code>urv.cdef</code></td>
<td>Deflate one column of $R$ in the URV decomposition.</td>
</tr>
<tr>
<td><code>urv.csne</code></td>
<td>Corrected semi-normal equations expansion step (for URV).</td>
</tr>
<tr>
<td><code>urv.rdef</code></td>
<td>Deflate one row of $R$ in the URV decomposition.</td>
</tr>
<tr>
<td><code>urv.ref</code></td>
<td>Refine one column of $R$ in the URV decomposition.</td>
</tr>
</tbody>
</table>

In addition we provide 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]. See the accompanying manual for more details.

Acknowledgements

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.

References


