Downdating a Rank-Revealing URV Decomposition*

Yuan-Jye Jason Wu†

November 6, 1996

Abstract

The rank-revealing URV decomposition is a useful tool for the subspace-tracking problem in digital signal processing. Updating the decomposition is a stable process. However, downdating a rank-revealing URV decomposition can be unstable because the R factor is ill-conditioned. In this article, we review some existing downdating algorithms for the full-rank URV decomposition in the absence of the U factor and develop a new combined algorithm. The combined algorithm has the merits of low cost and no intermediate breakdown, so the downdate is always computable in floating-point arithmetic. For the rank-revealing URV decomposition, we develop a two-step method that applies full-rank downdating algorithms to the signal and noise parts separately without using hyperbolic rotations. We prove that Park and Eldén's reduction algorithm and the combined algorithm have relational stabilities for both full-rank and rank-revealing cases. We demonstrate the efficiency and accuracy of our combined algorithm on ill-conditioned problems.

1 Introduction

Subspace tracking is an important subject in many applications of digital signal processing. Suppose that d signals are impinging on an array of m sensors and $m > d$. With the presence of noise, after the sensors collect n snapshots, $n \geq m$, we will have an $n \times m$ data matrix X whose singular values satisfy

$$\sigma_1 \geq \cdots \geq \sigma_d >> \sigma_{d+1} \geq \cdots \geq \sigma_m > 0 \ .$$

In this case, we say that the data matrix X has numerical rank d. The problem of subspace tracking is to estimate the numerical rank d and bases for the subspaces corresponding to $\{\sigma_1, \ldots, \sigma_d\}$ and $\{\sigma_{d+1}, \ldots, \sigma_m\}$ in the m-dimensional space.

Computing the singular value decomposition of X or the eigendecomposition of $X^H X$ is a common and natural choice for solving this problem. However, both methods require a large computational burden to update the estimates when the data matrix X incorporates

*This work was supported by NSF Grant CCR 91-15568.

†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439. jwu@mcs.anl.gov

an incoming data sample. Stewart [1] introduced a *rank-revealing URV decomposition* so that the data matrix can be expressed as

$$X = U \begin{bmatrix} R \\ 0 \end{bmatrix} V^H = U \begin{bmatrix} R_s & F \\ 0 & G \\ 0 & 0 \end{bmatrix} [V_s \; V_n] \,, \tag{1}$$

where

U and V are unitary matrices,
R_s is an upper triangular matrix of order d,
G is an upper triangular matrix of order $m-d$,
$\inf(R_s) \approx \sigma_d$, and
$\|G\|_F^2 + \|F\|_F^2 \approx \sigma_{d+1}^2 + \cdots + \sigma_m^2$.

The rank-revealing property of the R factor provides the information for rank estimation, and the orthonormal columns of the matrices V_s and V_n form bases for the signal and noise subspaces respectively. The matrix U is usually not saved because of its large order. Taking advantage of a fast $O(m^2)$ numerically stable updating algorithm, one can easily compute a decomposition of a new data matrix resulting from appending an incoming row of data. Several URV-based algorithms for finding a signal's direction-of-arrival have been proposed and have shown efficient and effective performance [2, 3].

However, in some applications the data matrix X is collected by the *rectangular windowing* method to reduce the effect of earlier data. As shown in Figure 1, the rectangular windowing method uses the most recent n samples of data. In this case, a reliable downdating algorithm is required to compute a rank-revealing URV decomposition of a matrix resulting from removing the first row of X.

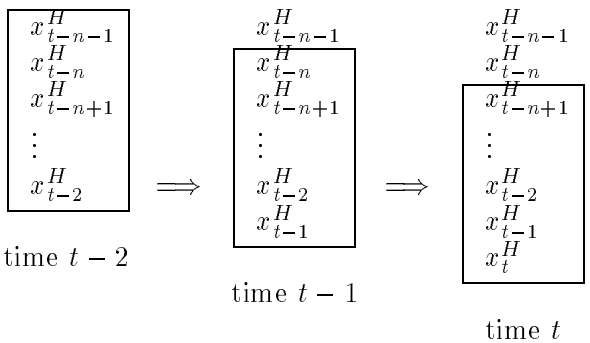


Figure 1: The shift of the n size window frame

Downdating algorithms when the matrix U is available are well studied and numerically stable (e.g., [4] and [5]). Unfortunately, without U, it is difficult to have a numerically stable downdating algorithm. The LINPACK algorithm [6], the CSNE algorithm [7], Chambers' algorithm [8], and the reduction algorithm [9] either break down or are sensitive to the condition number of R. Our goal is to design a new algorithm that will not break down and that will be more stable than those algorithms.

In this article, we first review existing downdating algorithms for a full-rank URV decomposition and develop a new combined algorithm (§2). For a rank-revealing URV decomposition, we review a two-step method that applies those full-rank downdating algorithms to the signal and noise parts separately (§3). We then show the relational stability for the combined algorithm (§4). Experimental results are given in §5. Finally, we state our conclusions in §6.

2 Full-Rank Downdating Algorithms

Consider a data matrix X with full column rank and a URV decomposition,

$$X = \begin{bmatrix} x^H \\ \widehat{X} \end{bmatrix} = U \begin{bmatrix} R \\ 0 \end{bmatrix} V^H,$$

where x^H is the first row of the data matrix X, R is an upper triangular matrix of order m, and U and V are unitary matrices of order n and m respectively. Our goal is to find an $m \times m$ upper triangular matrix T and unitary matrices $\widehat{U}$ and $\widehat{V}$ of order $n-1$ and m respectively such that

$$\widehat{X} \;=\; \widehat{U} \begin{bmatrix} T \\ 0 \end{bmatrix} \widehat{V}^H.$$

By partitioning U and $\widehat{U}$ as

$$U = \begin{bmatrix} \underset{m+1}{U_1} & \underset{n-m-1}{U_2} \end{bmatrix} \quad \text{and} \quad \widehat{U} = \begin{bmatrix} \underset{m}{\widehat{U}_1} & \underset{n-m-1}{\widehat{U}_2} \end{bmatrix}, \tag{2}$$

the decompositions can be written as

$$\begin{bmatrix} \widehat{X} \\ x^H \end{bmatrix} \;=\; W U_1 \begin{bmatrix} R \\ 0^H \end{bmatrix} V^H, \tag{3}$$

$$\widehat{X} \;=\; \widehat{U}_1 T \, \widehat{V}^{\,H}, \tag{4}$$

where W is the permutation matrix that exchanges the first and last rows. Then we substitute (4) into (3) and obtain

$$\begin{bmatrix} \widehat{U}_1 & 0 \\ 0^H & 1 \end{bmatrix} \begin{bmatrix} T \\ x^H \widehat{V} \end{bmatrix} \widehat{V}^H = W U_1 \begin{bmatrix} R \\ 0^H \end{bmatrix} V^H,$$

or

$$\begin{bmatrix} T \\ x^H \widehat{V} \end{bmatrix} = \begin{bmatrix} \widehat{U}_1^H & 0 \\ 0^H & 1 \end{bmatrix} W U_1 \begin{bmatrix} R \\ 0^H \end{bmatrix} V^H \widehat{V} \; .$$

Let

$$Q^H = \begin{bmatrix} \widehat{U}_1^H & 0 \\ 0^H & 1 \end{bmatrix} W U_1 \;\text{ and }\; P = V^H \widehat{V}.$$

We have

$$\begin{bmatrix} T \\ x^H V P \end{bmatrix} = Q^H \begin{bmatrix} R \\ 0^H \end{bmatrix} P. \tag{5}$$

Therefore, the downdating problem is equivalent to finding an $m \times m$ upper triangular matrix T and unitary matrices Q and P that satisfy (5).

Note that if $P = I$, downdating the URV decomposition is exactly the same as downdating a QR decomposition. However, as we shall show later, the presence of the matrix P will give us flexibility for deriving a more stable algorithm.

We first review several existing algorithms. Then we design a new combined algorithm. Throughout this article, the matrix Z denotes the product XV and $z^H = x^H V$ is the first row of Z.

2.1 LINPACK and CSNE Algorithms

We begin with algorithms that do not apply plane rotations on the right. In this case, $P = I$ and $R^H R$ forms a Cholesky factorization of the matrix $Z^H Z$. The original problem can be restated as finding an upper triangular matrix T such that

$$T^H T = R^H R - z z^H \ ,$$

which amounts to downdating a Cholesky factorization. The method in the LINPACK package [10] analyzed by Stewart [6] is a popular choice for solving this problem.

The strategy of the LINPACK algorithm is to recover u^H, the first row of the matrix U, in order to directly form a new U_1 in (2) with the structure

$$\begin{bmatrix} \mu & 0^H \\ 0 & \widehat{U}_1 \end{bmatrix},$$

where $|\mu| = 1$. Since the original matrix X has full rank, the matrix R is also full rank, and $[u_1 \cdots u_m]$, the first m components of u, can be uniquely determined by solving the triangular system $[u_1 \cdots u_m]R = z^H$.

The only constraint for the last $(n - m)$ columns of the matrix U is that they form an orthonormal basis for the left null subspace of X. Without loss of generality, we can choose the last $(n - m - 1)$ components of the first row of the matrix U to be zeros. Then we are free to choose $[\alpha\, 0 \cdots 0]$ as the last $(n - m)$ components of u^H, where

$$\alpha = \sqrt{1 - \| \, [u_1 \cdots u_m] \, \|_2^2} \ .$$

Now, we determine a sequence of plane rotations Q_k, $k = m, \ldots, 1$ of order $m+1$ in the $(k, m+1)$ plane such that

$$[u_1 \cdots u_m \ \alpha] Q_m \cdots Q_1 = [0 \cdots 0 \ \mu] \ . \tag{6}$$

Then the downdated triangular matrix T results from computing

$$(Q_m \cdots Q_1)^H \begin{bmatrix} R \\ 0^H \end{bmatrix} = \begin{bmatrix} T \\ z^H \end{bmatrix} \ . \tag{7}$$

We now state the LINPACK algorithm formally.

Algorithm 2.1 LINPACK

1. Compute $[u_1 \cdots u_m]$ by solving $[u_1 \cdots u_m]R = z^H$.

2. Compute $\alpha = \sqrt{1 - \| [u_1 \cdots u_m] \|_2^2}$.

3. Determine plane rotations $Q_m, \ldots, Q_1$ satisfying (6).

4. Compute the downdated triangular matrix T by using (7).

Let one flop be one floating point operation ($+, -, *$, or $/$). The LINPACK algorithm requires $4m^2 + O(m)$ flops, resulting from $m^2 + O(m)$ flops in triangular solving and $3m^2 + O(m)$ flops in plane rotations.

It is possible that the matrix R is ill-conditioned. For example, as the signal-to-noise ratio (SNR) or the sensor-to-signal ratio (m/d) increases, the smallest singular value of the matrix R tends to zero. Under floating-point arithmetic, a breakdown might occur in the LINPACK algorithm when there is a negative computed value under the square root at Step 2. To have a more accurate result, Björck, Park, and Eldén [7] developed a method called corrected seminormal equations (CSNE), using the original data matrix in the refinement of $[u_1 \cdots u_m]$ and α.

Let $\bar{u}^H$ be the computed result at Step 1 in the LINPACK algorithm. Since R is nonsingular, there is a vector w such that $Rw = \bar{u}$. The CSNE algorithm is based on the seminormal equations

$$R^H R w = Z^H e_1 \ ,$$

where e_1 is the first unit vector of length n. To apply one step of refinement, we need to find a vector δw such that

$$R^H R \delta w = Z^H r \ ,$$

where $r = e_1 - Zw$ is the residual. Let $\delta \bar{u} = R\delta w$. Thus we have corrected vectors $w_c = w + \delta w$ and $\bar{u}_c = \bar{u} + \delta \bar{u}$. For the correction of the scalar α, we have

$$\begin{aligned} \alpha_c &= \|u - \bar{u}_c\|_2 \\ &= \|U^H e_1 - U^H U \begin{bmatrix} R \\ 0 \end{bmatrix} w_c\|_2 \\ &= \|e_1 - Z w_c\|_2 \ . \end{aligned}$$

We now state the algorithm formally.

Algorithm 2.2 CSNE

1. Compute $\bar{u}$ by solving $R^H \bar{u} = z$.

2. Compute w by solving $Rw = \bar{u}$, and compute the residual $r = e_1 - Zw$.

3. Compute $\delta \bar{u}$ by solving $R^H \delta \bar{u} = Z^H r$, and let $\bar{u} = \bar{u} + \delta \bar{u}$.

4. Compute δw by solving $R\delta w = \delta \bar{u}$.

5. Compute $\alpha_c = \|r - Z\delta w\|_2$.

6. Determine plane rotations $Q_1, \ldots, Q_m$ satisfying (6).

7. Compute the downdated triangular matrix T by using (7).

There are four linear triangular systems to solve and three matrix-vector multiplications. The CSNE algorithm needs $6mn + 7m^2 + O(m)$ flops.

2.2 Chambers' Algorithm

Chambers' algorithm [8] is another method that does not apply right rotations. The idea is quite simple. With $P = I$, if we premultiply both sides in (5) by the unitary matrix Q, we have the updating problem

$$\begin{bmatrix} R \\ 0^H \end{bmatrix} = Q \begin{bmatrix} T \\ z^H \end{bmatrix} .$$

Then we examine the updating process and reverse it to obtain a solution to our downdating problem.

The most common way for solving this updating problem is to apply a sequence of left plane rotations Q_k, $k = 1, \ldots, m$ of order $m+1$ in the $(k, m+1)$ plane to eliminate the row vector z^H. Note that each rotation modifies only one row of T to compute the corresponding row of R. Therefore, we can reverse each rotation process and recover the matrix T row by row.

For Q_1, the rotation process can be expressed as

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ 0 & \bar{z}_2 & \cdots & \bar{z}_m \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ z_1 & z_2 & \cdots & z_m \end{bmatrix} , \tag{8}$$

where

$$c = \frac{t_{11}}{\sqrt{t_{11}^2 + z_1^2}} , \text{ and } s = \frac{z_1}{\sqrt{t_{11}^2 + z_1^2}} . \tag{9}$$

Now suppose that $[r_{11} \cdots r_{1m}]$ and $[z_1 \cdots z_m]$ are known. Our goal is to compute $[t_{11} \cdots t_{1m}]$ and $[\bar{z}_2 \cdots \bar{z}_m]$.

Since $r_{11} = \sqrt{t_{11}^2 + z_1^2}$, we have

$$t_{11} = \sqrt{r_{11}^2 - z_1^2} .$$

Once t_{11} is known, the scalars c and s are computed by (9). From the first row in (8) for computing $r_{1i}, i = 2, \ldots, m$, we have

$$t_{1i} = (r_{1i} - s z_i)/c . \tag{10}$$

Applying the result in (10) to the second row in (8), we obtain

$$\bar{z}_i = c z_i - s t_{1i} . \tag{11}$$

Repeating this process will yield the downdated upper triangular matrix T one row at a time.

The algorithm is formally stated as the following.

Algorithm 2.3 CHAMBERS

$(z^H = x^H V)$
For $k = 1, 2, \ldots, m$

1. Compute $t_{kk} = \sqrt{r_{kk}^2 - z_k^2}$.

2. If $k < m$

Compute $c = t_{kk}/r_{kk}$ and $s = z_k/r_{kk}$.
Compute $(t_{k,k+1} \cdots t_{km})$ by using (10).
Replace $(z_{k+1} \cdots z_m)$ by $(\bar{z}_{k+1} \cdots \bar{z}_m)$ in (11).
End if

End for

Chambers' algorithm requires only $3m^2 + O(m)$ flops. However, the algorithm breaks down when the argument of the square root at Step 1 is nonpositive for $k < m$, and there is no way to recover. When the breakdown happens at $k = m$, this implies that t_{mm} is quite small, and Park and Eldén [9] suggest letting $t_{mm} = 0$. Thus the matrix T possibly has rank one less than the matrix R. It will be proved in Section 4 that setting t_{mm} to zero gives an acceptable relative error bound.

2.3 Reduction Algorithm

We now introduce an algorithm described by Park and Eldén [9] that applies right rotations. The reduction algorithm works on the problem (5) directly. We first determine a sequence of right plane rotations P_k, $k = 1, \ldots, m-1$ of order m in the $(k, k+1)$ plane such that

$$[z_1 \cdots z_{m-1}\ z_m] P_1 \cdots P_{m-1} = [0 \cdots 0\ \|z\|_2] \ .$$

Each P_k reduces number of nonzeroes by one. When we apply P_k to the matrix R, we create a nonzero entry at the $(k+1, k)$ position in $[R^H\ 0^H]^H$. So we use a corresponding left plane rotation Q_k^H in the $(k, k+1)$ plane to eliminate that nonzero entry. Therefore, the matrix

$$Q_{m-1}^H \cdots Q_1^H \begin{bmatrix} R \\ 0 \end{bmatrix} P_1 \cdots P_{m-1} \tag{12}$$

remains upper triangular. The resulting matrix in the above equation is equal to the downdated triangular matrix T except at position (m, m), and t_{mm} can be computed by simply taking the square root of the difference between the square of the (m, m)-entry in (12) and $\|z\|_2$.

We now state the algorithm formally.

Algorithm 2.4 REDUCTION

$(z^H = x^H V)$

1. For $k = 1, 2, \ldots, m-1$

1.1. Compute a right rotation P_k to eliminate z_k with z_{k+1}, and apply it to R and V.

1.2. Compute a left rotation Q_k to eliminate $r_{k+1,k}$ with r_{kk}.

1.3. Let $[t_{kk} \ \cdots \ t_{km}] = [r_{kk} \ \cdots \ r_{km}]$.

End for

2. Compute $t_{mm} = \sqrt{r_{mm}^2 - \|z\|_2^2}$.

The reduction algorithm requires $12m^2 + O(m)$ flops. Again, the argument in the square root at Step 2 might be negative under floating-point arithmetic. We let $t_{mm} = 0$ if this occurs.

2.4 Combined Algorithm

None of the algorithms considered so far is ideal. The LINPACK algorithm will stop when a breakdown occurs. Even with one step of refinement, the CSNE algorithm will lead to an inaccurate result if the computed u is far from accurate. Thus, the LINPACK-type algorithms are unreliable if R is ill-conditioned.

Chambers' algorithm has an attractive computational cost, but there is a risk of intermediate breakdown. On the other hand, the reduction algorithm avoids intermediate breakdown but is more expensive. However, since both algorithms reduce the problem size one by one and compute the downdated triangular matrix T row by row, we can combine Chambers's algorithm and the reduction algorithm in order to obtain low cost and no intermediate breakdown.

The idea is to apply Chambers' algorithm first. If a breakdown occurs at $k < m$, we adopt one reduction step to reduce the problem size by one and then apply Chambers' algorithm until the next breakdown. If breakdown occurs at $k = m$, we let $t_{mm} = 0$. We now state the algorithm.

Algorithm 2.5 COMBINED

$(z^H = x^H V)$

1. For $k = 1, 2, \ldots, m-1$

1.1. Compute $\rho = r_{kk}^2 - z_k^2$.

1.2. If $\rho > 0$

% perform one step of Chambers' algorithm %

Compute $t_{kk} = \sqrt{\rho}$.

Compute $c = t_{kk}/r_{kk}$ and $s = z_k/r_{kk}$.

Compute $(t_{k,k+1} \ \cdots \ t_{km})$ by (10).

Replace $(z_{k+1} \ \cdots \ z_m)$ by $(\bar{z}_{k+1} \ \cdots \ \bar{z}_m)$ in (11).

1.3. Else

% perform one step of the reduction algorithm%

Compute a right rotation P_k to eliminate z_k with z_{k+1}, and apply it to R and V.
Compute a left rotation Q_k to eliminate $r_{k+1,k}$ with r_{kk}.
Let $[t_{kk} \ \cdots \ t_{km}] = [r_{kk} \ \cdots \ r_{km}]$.
End if

End for

2. Let $t_{mm} = \begin{cases} 0 & \text{if } r_{mm}^2 - z_m^2 \le 0 \\ \sqrt{r_{mm}^2 - z_m^2} & \text{if } r_{mm}^2 - z_m^2 > 0 \end{cases}$.

The complexity of the combined algorithm lies between $3m^2$ and $12m^2$, depending on how many reduction steps it takes.

3 Rank-Revealing Downdating Algorithms

We now consider downdating a rank-revealing URV decomposition. Referring to (1), the matrix R has the data of the signal (R_s) well separated from that of the noise (F and G), and we need to preserve this signal-noise (or large-small) structure for the downdated triangular matrix T. Therefore, we change our problem equation (5) to

$$Q^H \begin{bmatrix} R_s & F \\ 0 & G \\ 0^H & 0^H \end{bmatrix} P = \begin{bmatrix} T_s & B \\ 0 & C \\ z_s^H P_s & z_n^H P_n \end{bmatrix},$$

where

$$z^H = \begin{bmatrix} z_s^H & z_n^H \\ d & m-d \end{bmatrix}, \quad \text{and} \quad P = \begin{bmatrix} P_s & 0 \\ 0 & P_n \\ d & m-d \end{bmatrix} \begin{matrix} \\ \\ d \\ m-d \end{matrix}.$$

We cannot directly apply the reduction step in the combined algorithms to the rank-revealing case because the presence of the matrix P might mix the signal and noise data. The LINPACK, CSNE, and Chambers' algorithms have no risk of mixing signal and noise but could produce an inaccurate result or a breakdown because R is ill-conditioned.

Park and Eldén [9] give a simple and direct method called the two-step procedure to solve this problem. They consider only the LINPACK, CSNE, and reduction algorithms. A similar method is also studied by Barlow and Zha [11]. They suggest applying one of the algorithms for the full-rank problem to compute the signal (T_s) and noise (C) parts separately. The only additional work required is a connection task: we have to compute B and modify z_n^H after computing T_s. Then we can patch these two parts together to form the downdated triangular matrix T.

Note that the left plane rotations that do not involve the vector z_s^H in downdating the signal part are applied to the matrix F directly and there is no need to update z_n^H. However, for those rotations that update the vector z_s^H, we need an algorithm to perform the corresponding computation on z_n^H and F.

Park and Eldén choose hyperbolic rotations as the connection algorithm. For the LINPACK and CSNE algorithms, d hyperbolic rotations are required. Only one such rotation is needed for the reduction algorithm. However, the hyperbolic rotation is not recommended because it is not backward stable [12] [13].

In contrast to hyperbolic rotations, Equations (10) and (11) in Chambers' algorithm give an alternative way to perform the two-step method, computing $[T_s \; B]$ and modifying z_n^H simultaneously. Chambers' algorithm and the hyperbolic rotations differ only in the formula to modify z_n^H. The left rotations in (12) for the reduction algorithm also are applied to the matrix F directly. Therefore, the only connection task left in the combined algorithm is to modify the last row of the corrected F when t_{dd} is assigned to be 0.

The assignment occurs when we want to find a unitary matrix Q_d^H and vectors $\tilde{z}_n^H$ and $[t_{dd} \; b_d^H]$ such that

$$Q_d^H \begin{bmatrix} r_{dd} & f_d^H \\ 0 & G \\ 0 & \tilde{z}_n^H \end{bmatrix} = \begin{bmatrix} t_{dd} & b_d^H \\ 0 & G \\ \xi & \bar{z}_n^H \end{bmatrix}, \tag{13}$$

where the downdated vector $[\xi \; \bar{z}_n^H]$ results from the previous $d-1$ downdating step. We let $t_{dd} = 0$ if there is a negative argument in $\sqrt{r_{dd} - \xi}$ caused by $r_{dd} \approx \xi$. The same assignment is also needed in the reduction algorithm for downdating the signal part. In this case, Park and Eldén let $\tilde{z}_n^H = \bar{z}_n^H$ and $b^H = f^H$ after the zero assignment, and go on downdating the noise part.

We have a different approach after observing (13) closely. The zero assignment makes Q_d^H to be a plane rotation with 90 degree rotation, or a permutation matrix, so that $b_d^H = \tilde{z}_n^H$. In this case, further downdating to the noise part is not necessary because the next downdated vector is exactly equal to a row of the matrix B. We just let $b^H = 0$, $C = G$, and stop the downdating right here. Thus the combined algorithm can be applied to the rank-revealing case without using hyperbolic rotations.

In Section 5, we will show that the combined algorithm plus the zeroes assignment make a good connection between the signal and noise parts. Since we do not require any additional computation, the complexity of the combined algorithm for the rank-revealing case is still $O(m^2)$. Therefore, we have an algorithm that makes the downdate always computable in floating-point arithmetic.

We note that in the rank-revealing case, since one row is deleted from the original data matrix, the resulting triangular matrix T_s can have numerical rank degeneracy. We examine the resulting matrix T_s by applying the deflation algorithm defined in [1] after performing each downdate. If T_s is rank deficient, we repartition the matrix, reducing the dimension of T_s.

4 Error Analysis

In contrast to the methods in which the matrix U is available [4, 5], none of the downdating algorithms that we consider is backward stable in the classical sense [14]. In fact, Björck, Park, and Eldén [7] state that no algorithm using the matrix R only to compute the required entries of the matrix U can be backward stable. However, Stewart [6] found an special error property called *relational* or *mixed stability* for these algorithms. Furthermore,

Stewart [13] showed that relational stability can be preserved after a sequence of updates and downdates. He also proved that if the final leading principal matrix T_s in the sequence is well conditioned, it will be computed accurately. Based on this analysis, our goal is to verify the relational stability of the combined algorithm. Throughout this section, a tilde will denote a result computed in floating-point arithmetic. The quantities $\|A\|$ and $\|x\|$ will denote the Frobenius norm of a matrix A and the Euclidean norm of a vector x respectively. We study the first-order perturbation analysis only and suppress the higher-order terms. The relation symbol $\lesssim$ denotes less than or equal to without considering the second- and higher-order terms.

Suppose that $\tilde{T}$ is the computed downdated triangular matrix. Relational stability ensures that there exists an $(m+1) \times m$ matrix E satisfying

$$\|E\| \lesssim k_m \|R\| \epsilon_M \ , \tag{14}$$

and unitary matrices $\widehat{Q}$ and $\widehat{P}$ such that

$$\widehat{Q}^H \begin{bmatrix} R \\ 0 \end{bmatrix} \widehat{P} = \begin{bmatrix} \tilde{T} \\ z^H \widehat{P} \end{bmatrix} + E \ . \tag{15}$$

Here ϵ_M is the machine relative precision and k_m is a constant depending on m and the computer arithmetic. For convenience, we let $y^H = z^H \widehat{P}$ and express E as

$$E = \begin{bmatrix} \Delta \tilde{T} \\ \Delta y^H \end{bmatrix} \ .$$

From (15), we can understand why these algorithms are not backward stable, because the error matrix E is dependent not only on R and z but also on the result $\tilde{T}$.

It has been shown in (14) that,

- $k_m = m^2/2 + 9m\sqrt{m} + O(m)$, for the LINPACK algorithm [6],
- $k_m = 4m\sqrt{m}$, for Chambers' algorithm [12].

On the other hand, algorithms involving hyperbolic rotations do not have relational stability because the parameter k_m in (14) is not bounded and depends on the tangents of rotation angles [12]. Therefore, the two-step method using hyperbolic rotations is not relationally stable.

Our next task is to prove that the reduction algorithm has relational stability. We adopt the notation in [14] that $fl\,(a)$ represents the floating-point representation of a. Operations in floating-point arithmetic are based on the following rules:

1. $fl\,(a * b) = (a * b)(1+\epsilon)$,
2. $fl\,(a/b) = (a/b)(1+\epsilon)$,
3. $fl\,(a \pm b) = a(1+\epsilon_1) \pm b(1+\epsilon_2)$,
4. $fl\,(\sqrt{a}) = \sqrt{a}(1+\epsilon)$,

where $|\epsilon|, |\epsilon_1|, |\epsilon_2| \leq \epsilon_M$. For convenience, we denote

$$fl^{\,2}((a+b)+c) = fl\,(fl\,(a+b)+c)\ .$$

Each step of the combined algorithm uses either Chambers' algorithm or the reduction algorithm to reduce the problem size by one. Thus, we need only to prove relational stability for the reduction algorithm.

The main computation in the reduction algorithm is plane rotation. Therefore, we begin with an error analysis for computing right plane rotations. At Step 1.1 in Algorithm 2.4, we compute a sequence of plane rotations $\tilde{P}_1, \ldots, \tilde{P}_{m-1}$ so that

$$\tilde{y}^H = fl^{\,m-1}((\cdots(z^H\tilde{P}_1)\cdots)\tilde{P}_{m-1})\ ,$$

where $\tilde{y}$ is a multiple of the mth unit vector. Wilkinson [14, pp. 135–138] showed that, for any z, there exists a sequence of exactly orthogonal matrices $\widehat{P}_1, \ldots, \widehat{P}_{m-1}$ independent of z such that

$$\|\Delta y^H\| \equiv \|\tilde{y}^H - z^H\widehat{P}_1\cdots\widehat{P}_{m-1}\| \lesssim 6(m-1)\|z\|\epsilon_M\,. \tag{16}$$

Next, we apply these right rotations to the matrix R and compute corresponding left plane rotations $\tilde{Q}_1, \ldots, \tilde{Q}_{m-1}$ so that

$$\tilde{T}' = fl^{\,2m-2}(\tilde{Q}_{m-1}^H(\cdots(\tilde{Q}_1^H(R\tilde{P}_1))\cdots\tilde{P}_{m-1}))\ . \tag{17}$$

(Here the left rotations are of order m, which is one less than those in (5) since we apply them to the matrix R only.) Note that the matrix $\tilde{T}'$ is equal to the matrix $\tilde{T}$ except in the (m,m)-entry. As Wilkinson [14, p. 141] pointed out, the order of pre- and postmultiplications affects only the second-order term in error analysis. For convenience, we derive an error bound for the case in which the left rotations are applied after applying all the right rotations, though the right and left rotations are applied alternately in the reduction algorithm.

Let

$$R' = fl^{\,m-1}((\cdots(R\tilde{P}_1)\cdots)\tilde{P}_{m-1})\ .$$

By an argument similar to the derivation of (16) and norm property, we have

$$\|R' - R\widehat{P}_1\cdots\widehat{P}_{m-1}\| \lesssim 6(m-1)\|R\|\epsilon_M\ . \tag{18}$$

Furthermore, there also exists a sequence of exactly orthogonal matrices $\widehat{Q}_1$,..., $\widehat{Q}_{m-1}$ such that

$$\|\tilde{T}' - \widehat{Q}_{m-1}^H\cdots\widehat{Q}_1^H R'\| \lesssim 6(m-1)\|R'\|\epsilon_M\ , \tag{19}$$

Applying the triangular inequality to (18), we have

$$\|R'\| \leq (\sqrt{m} + 6(m-1)\epsilon_M)\|R\|\ , \tag{20}$$

where the $\sqrt{m}$ comes from taking the Frobenius norm of a unitary matrix. Therefore, combining (18), (19), and (20), we have

$$\begin{aligned}
\|\Delta\tilde{T}'\| &\equiv \|\tilde{T}' - \widehat{Q}_{m-1}^H\cdots\widehat{Q}_1^H R\widehat{P}_1\cdots\widehat{P}_{m-1}\| \\
&\leq \|\tilde{T}' - \widehat{Q}_{m-1}^H\cdots\widehat{Q}_1^H R'\| + \|\widehat{Q}_{m-1}^H\cdots\widehat{Q}_1^H(R' - R\widehat{P}_1\cdots\widehat{P}_{m-1})\| \\
&\lesssim 6(m-1)\|R'\|\epsilon_M + \sqrt{m}\|R' - R\widehat{P}_1\cdots\widehat{P}_{m-1}\| \\
&\lesssim 6(m-1)\epsilon_M(\sqrt{m} + 6(m-1)\epsilon_M)\|R\| + 6(m-1)\sqrt{m}\epsilon_M\|R\|\ .
\end{aligned}$$

Neglecting the ϵ_M^2 term, we have that

$$\|\Delta\tilde{T}'\| \lesssim 12(m-1)\sqrt{m}\epsilon_M\|R\|. \tag{21}$$

Now, in Step 2 of the reduction algorithm, we compute $\tilde{t}_{mm}$ from the (m,m)-entry of $\tilde{T}'$ (updated R) and $\tilde{y}_m$ (approximate 2-norm of z^H) using the equation

$$\tilde{t}_{mm} = \{[(\tilde{t'}_{mm} + \epsilon_1)^2(1+\epsilon_3) - (\tilde{y}_m + \epsilon_2)^2(1+\epsilon_4)](1+\epsilon_5)\}^{\frac{1}{2}}(1+\epsilon_6) \ ,$$

where

$$\begin{aligned} |\epsilon_1| &\lesssim 12(m-1)\sqrt{m}\epsilon_M\|R\| \text{ from (21)}, \\ |\epsilon_2| &\lesssim 6(m-1)\|z\|\epsilon_M \text{ from (16), and} \\ |\epsilon_3|, |\epsilon_4|, |\epsilon_5|, |\epsilon_6| &\leq \epsilon_M \text{ from floating-point operations rules.} \end{aligned}$$

Simplifying the above equation using the fact that $\|z\| \leq \|R\|$ and neglecting the ϵ_M^2 term, we characterize an error bound for $\tilde{t}_{mm}$ by

$$|\Delta\tilde{t}_{mm}| \lesssim [12(m-1)\sqrt{m}+2]\epsilon_M\|R\| \ . \tag{22}$$

Note that $\tilde{t}_{mm}$ should be nonnegative in the reduction algorithm. If there is a breakdown at the final step, it means that zero is within the bounded interval

$$[\tilde{t}_{mm} - (12(m-1)\sqrt{m}+2)\epsilon_M\|R\| \ , \ \tilde{t}_{mm} + (12(m-1)\sqrt{m}+2)\epsilon_M\|R\|] \ .$$

Thus, Park and Eldén's suggestion to put a zero when a breakdown occurs is acceptable.

Consequently, combining (16), (21), and (22), we derive a relational error bound for the reduction algorithm as

$$\begin{aligned} \|E\| &\leq \sqrt{\|\Delta\tilde{T}'\|^2 + |\Delta\tilde{t}_{mm}|^2 + \|\Delta\tilde{y}^H\|^2} \\ &\lesssim [12(m-1)\sqrt{m} + O(m)]\epsilon_M\|R\| \ . \end{aligned} \tag{23}$$

Therefore, we have shown that the combined algorithm (Algorithm 2.5) has relational stability.

Finally, since the two-step procedure use either left rotations or permutations (if a zero is assigned), the error bound in (23) holds for the rank-revealing case. Thus the rank-revealing combined algorithm also has relational stability.

5 Experimental Results

In this section, we show some experimental results for the rank-revealing downdating problem. Several combinations of full-rank algorithms can be applied to the signal and noise parts. However, considering the properties and complexity for each algorithm, we choose the following three combinations as our test algorithms:

Phase	Algorithm A	Algorithm B	Algorithm C
Signal	LINPACK/CSNE	Reduction	Combined
Connection	Hyperbolic	Hyperbolic	Chambers'/Zeroes
Noise	LINPACK/Reduction	Reduction	Combined

Note that the LINPACK algorithm cannot be present alone in any phase because it must be with a backup algorithm to recover from a possible breakdown at Step 2.

We construct a 100×8 test matrix K whose entries are taken from a uniform distribution in $(0, 1)$. Some portions of the matrix K are multiplied by scalars γ and δ to make varied numerical ranks. Then we multiply K on the right by a random unitary matrix. The size of the window function is 12.

To estimate the numerical rank, we need a tolerance described in [1]. The tolerance is an upper bound for the sum of squares of the singular values in the noise part and works like a barrier that separates the signal and noise parts. The numerical rank d is chosen as the smallest integer such that the norm of the resulting matrix C is less than the tolerance.

Suppose that the sizes of the noise collected in sensors are roughly the same. It has been shown in [3] that the sum of the squares of the $(m - d)$ smallest singular values of the data matrix sampled by the rectangular windowing method satisfies

$$\sigma_{d+1}^2 + \cdots + \sigma_m^2 \approx (m - d)\epsilon^2 * (\text{window size}) \ ,$$

where ϵ is the noise size. Therefore, in our tests, the tolerances are chosen as

$$tol_u = \psi_u * \delta * \sqrt{12(8 - d)} \ , \text{ for the updating algorithm,}$$
$$tol_d = \psi_d * \delta * \sqrt{12(8 - d + 1)} \ , \text{ for the deflation algorithm.}$$

The factors ψ_u and ψ_d are chosen by users to control the the accuracy of the approximate signal subspace. In our tests, the factor ψ_u is set to 1 and the factor ψ_d is chosen to make all three test algorithms give correct ranks.

Suppose that we partition the covariance matrix of the data matrix Z as

$$A = Z^H Z = \begin{bmatrix} A_s & A_c \\ A_c^H & A_n \end{bmatrix} \ ,$$

where A_s is of order d. We test the accuracy of the signal part by computing the relative error norm

$$\frac{\|A_s - \tilde{T}_s^H \tilde{T}_s\|_F}{\|A_s\|_F} \ ,$$

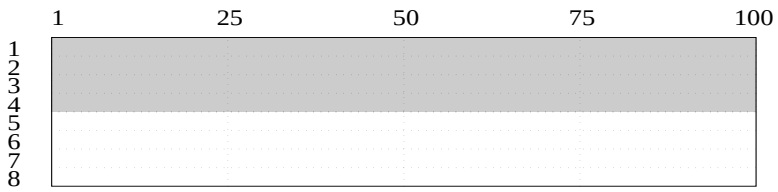
where $\tilde{T}_s$ is the computed T_s. Let $Z = W\Sigma Y^H$ be the singular value decomposition of the matrix Z, where W and Y are unitary matrices. For the accuracy of the noise part, we compute the sum of the sines of the canonical angles between the subspaces spanned by the last $8 - d$ columns of the matrices V and Y. Finally, we measure the relative error norm of the covariance matrix

$$\frac{\|A - \tilde{T}^H \tilde{T}\|_F}{\|A\|_F} \ ,$$

where $\tilde{T}$ denotes the computed T. All computations use double-precision IEEE floating-point arithmetic.

To make a fair comparison, we ran 50 trials for each test. The average costs over 50 trials, 88 downdates per trial, for each test are given in Table 1.

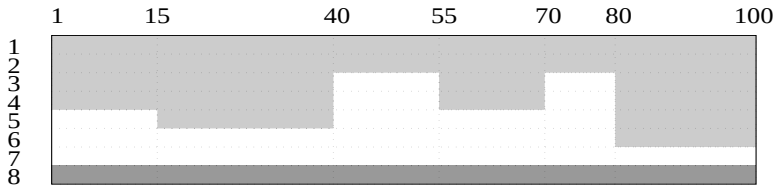
Test 1 & 2: Our first test matrix has a fixed numerical rank of 4. The test matrix K^H is constructed as



where the gray area is multiplied by $\delta = 10^{-4}$ for Test 1 and by $\delta = 10^{-8}$ for Test 2. The factor ψ_d is set to 4 and 8 for Test 1 and 2 respectively. Tables 2 and 3 show the average results of the rank estimates, the relative error norms, and the condition numbers of the matrix R. No breakdown occurs, so the LINPACK algorithm is always used in Algorithm A. All three algorithms give good results. However, Table 1 shows that the average cost of Algorithm C is less than that of the other two.

To make an ill-conditioned signal or noise part, we now increase the condition number of R_s or G by applying another scalar γ.

Test 3: Suppose that we have one signal stronger than others. The test matrix K^H looks like

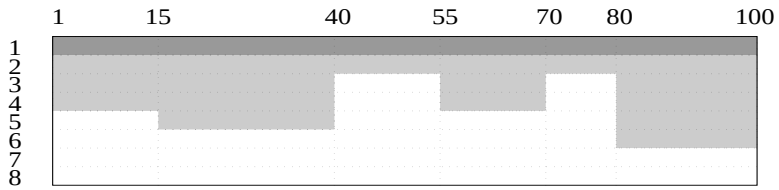


where the light gray area is multiplied by $\delta = 10^{-7}$ and the dark gray area is multiplied by $\gamma = 10^2$. The scalar γ makes R_s ill-conditioned around those positions with a sharp rank drop. We choose the factor $\psi_d = 20$. The results are given in Figure 2 and Figure 3.

In the first graph of Figure 2, we also mark the position where there is a breakdown. A "$*$" means that the CSNE algorithms is applied instead of the LINPACK algorithm. A "$+$" represents an assignment of 0 in the signal part in the reduction algorithm (no hyperbolic rotation is applied). A "$\circ$" shows that the combined algorithm assigns a 0 in the signal part and applies the plane rotations to eliminate the last row of the corrected F.

Algorithms A and B have a large relative error for the signal part when a breakdown occurs. Because of the ill-conditioned R_s, the solutions to the triangular linear systems in Algorithm A are less accurate, even if the CSNE algorithm corrects it by one step of refinement. For Algorithm B, the reduction steps in the signal part not only transfer the norm of the vector z_s^H to its last component but also transfer most of the energy of R_s to its last column. The enlargement of the arguments at Step 2 in Algorithm 2.4 increases the absolute error when we assign a 0 to t_{dd}. Algorithm C still has good performance in this test.

Test 4: We construct the test matrix K^H as



where the light gray area is multiplied by $\delta = 10^{-6}$ and the dark gray area is multiplied by $\gamma = 10^{-9}$. The factor ψ_d is set to 6. The scalar γ makes G ill-conditioned so that the data matrix behaves similarly to the one with large sensor-to-signal ratio (m/d).

Figure 4 shows the results. No breakdown occurs in any algorithm. However, the relative errors for the signal part of Algorithm A and B jump when the numerical rank increases. We find that the large errors actually start from the numerical rank degeneracy around position 26 shown in Figure 5. This is due to the ill-conditioned G and the inaccurately computed noise part when downdating by hyperbolic rotations. Again, Algorithm C shows good results.

6 Conclusions

We have presented a new algorithm, the combined algorithm, and shown its good performance on several ill-conditioned downdating problems. The combined algorithm has the following features:

- The work per downdate is $O(m^2)$.
- The algorithm is as efficient as Chambers' algorithm and does not break down.
- Since the algorithm does not use hyperbolic rotations, it has relational stability with the coefficient $k_m = 12(m-1)\sqrt{m}$ for the full-rank case and $k_m = 18(m-1)\sqrt{m}$ for the rank-revealing case.

We believe that the combined algorithm is suitable for real-time computations.

Acknowledgments

I thank Dianne P. O'Leary and Gail Pieper for very helpful comments.

References

[1] G. W. Stewart. An updating algorithm for subspace tracking. *IEEE Trans. Signal Processing*, 40:1535–1541, 1992.

[2] E. C. Boman, M. F. Griffen, and G. W. Stewart. Direction of arrival and the rank-revealing URV decomposition. In *Proceedings of ACASSP-91*, Washington, D.C., 1991. IEEE.

[3] K. J. Ray Liu, Dianne P. O'Leary, G. W. Stewart, and Yuan-Jye J. Wu. URV ESPRIT for tracking time-varying signals. *IEEE Trans. Signal Processing*, 42:3441–3448, 1994.

[4] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28:505–535, 1974.

[5] S. J. Olszanskyj and A. W. Bojanczyk. Compact Givens representation of the orthogonal factor in recursive linear squares. In John G. Lewis, editor, *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra*. SIAM, 1994.

[6] G. W. Stewart. The effects of rounding error on an algorithm for downdating a Cholesky factorization. *J. Institute for Mathematics and Appl.*, 23:203–213, 1979.

[7] Å. Björck, H. Park, and L. Eldén. Accurate downdating of least squares solutions. *SIAM J. Matrix Anal. and Appl.*, 15:549–568, 1994.

[8] J. M. Chambers. Regression updating. *J. American Statistical Association*, pages 744–748, 1971.

[9] H. Park and L. Eldén. Downdating the rank-revealing URV decomposition. *SIAM J. Matrix Anal. and Appl.*, 16:138–155, 1995.

[10] J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart. *LINPACK User's Guide*. SIAM, Philadelphia, 1979.

[11] J. L. Barlow and H. Zha. Stable algorithms for downdating two-sided orthogonal decompositions. Technical Report CSE–93–013, Department of Computer Science and Engineering, The Pennsylvania State University, 1993.

[12] A. W. Bojanczyk, R. P. Brent, P. Van Dooren, and F. R. De Hoog. A note on downdating the Cholesky factorization. *SIAM J. Scientific and Statistical Computing*, 8:210–221, 1987.

[13] G. W. Stewart. On the stability of sequential updates and downdates. Technical report, Inst. for Advanced Computer Studies Report TR-94-30, Computer Science Department Report TR-3238, University of Maryland, College Park, 1994.

[14] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, England, 1965.

	Algorithm A	Algorithm B	Algorithm C
Test 1	533	922	256
Test 2	533	922	524
Test 3	624	920	278
Test 4	571	920	260

Table 1: Average operations count (flops) for all tests

Algorithm	Ave. Rank Estimate	Ave. Error		Ave. Cond(R)
		Signal	Noise	
A	4	2.6937e-15	5.9723e-04	9.6484e+04
B	4	3.2455e-15	5.9723e-04	9.6484e+04
C	4	2.1222e-15	5.9723e-04	9.6484e+04

Table 2: Average results of the rank estimates, the signal and noise errors, and the condition numbers of R for Test 1 ($\delta = 10^{-4}$)

Algorithm	Ave. Rank Estimate	Ave. Error		Ave. Cond(R)
		Signal	Noise	
A	4	2.3847e-15	6.2704e-08	1.0911e+09
B	4	2.8806e-15	6.2704e-08	1.0911e+09
C	4	2.3357e-15	6.2704e-08	1.0911e+09

Table 3: Average results of the rank estimates, the signal and noise errors, and the condition numbers of R for Test 2 ($\delta = 10^{-8}$)

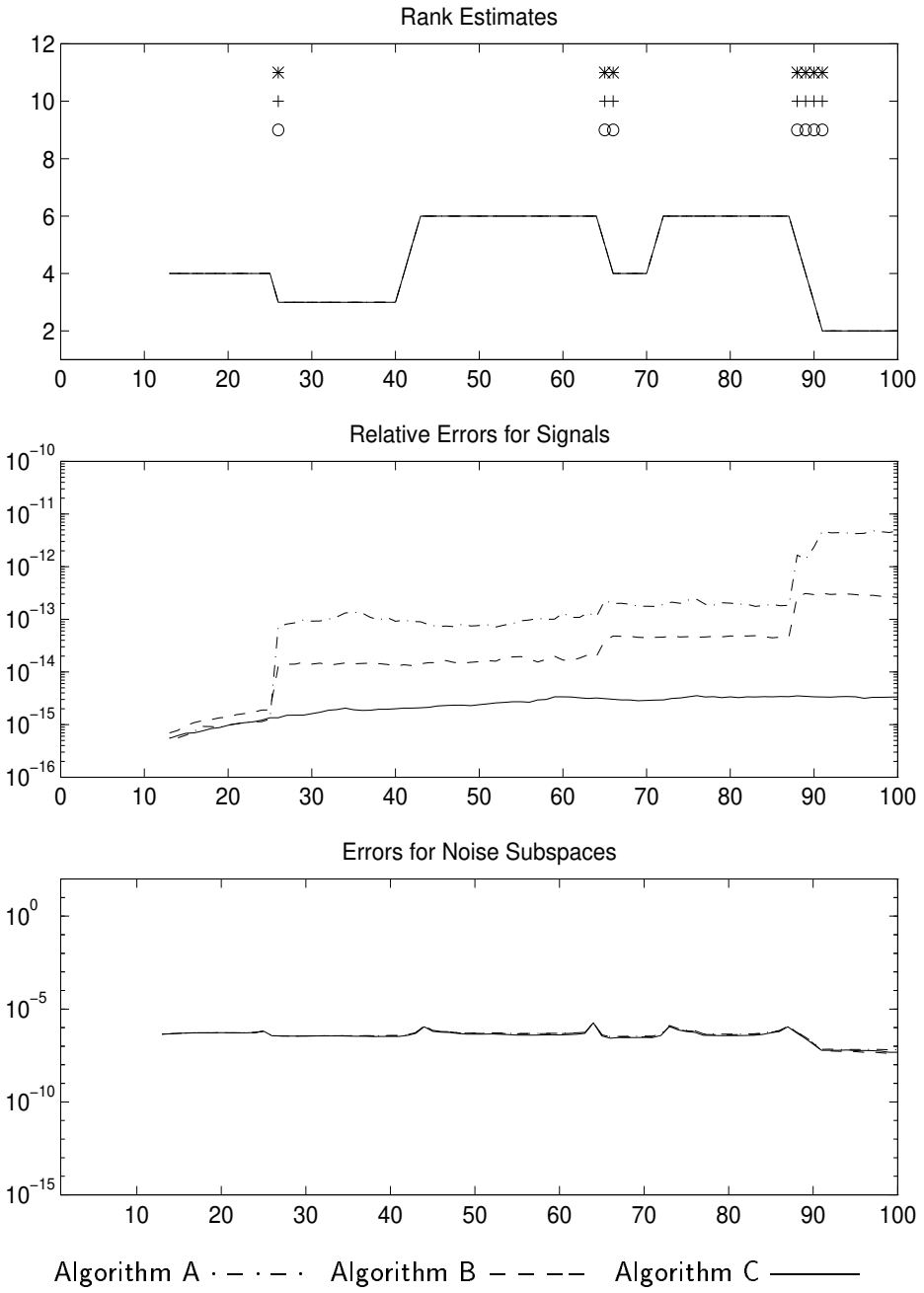


Figure 2: The plots of the estimated rank, the signal error, and the noise error vs. the window position for Test 3 ($\delta = 10^{-7}$ and $\gamma = 10^{2}$)

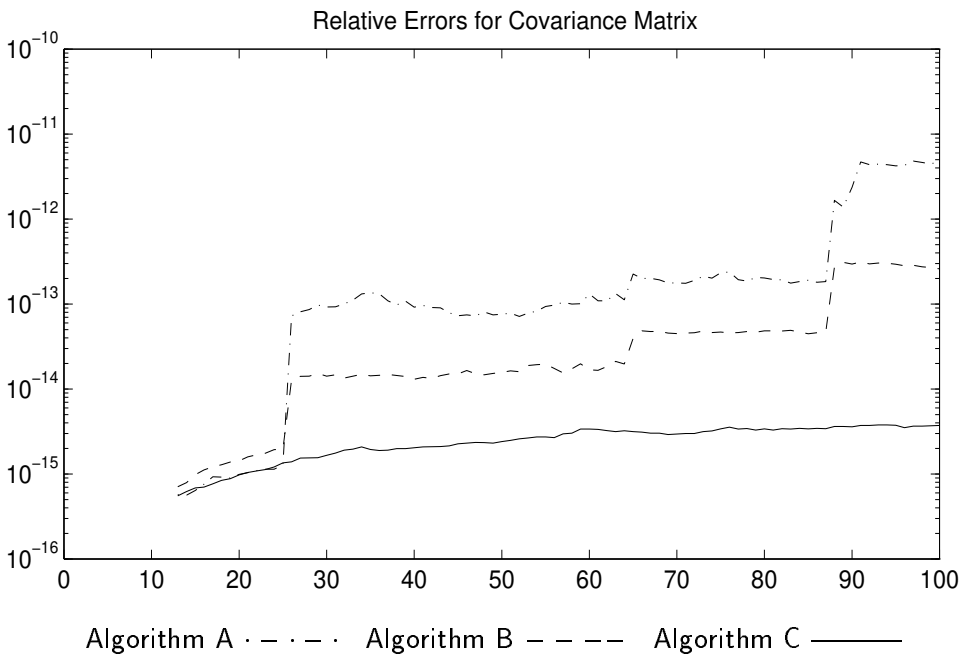


Figure 3: The plots of the relative errors for the covariance matrix vs. the window position for Test 3 ($\delta = 10^{-7}$ and $\gamma = 10^2$)

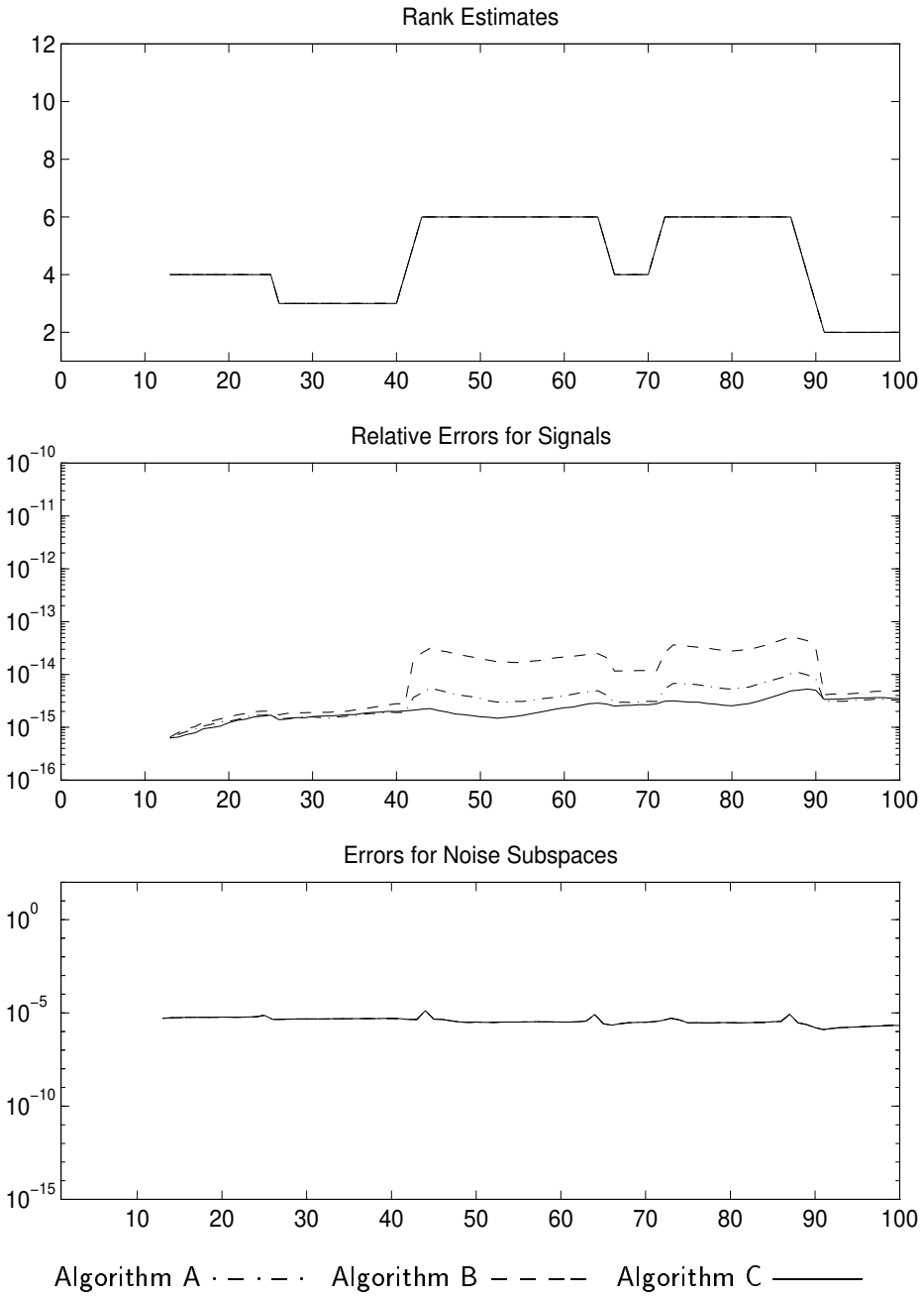


Figure 4: The plots of the estimated rank, the signal error, and the noise error vs. the window position for Test 4 ($\delta = 10^{-6}$ and $\gamma = 10^{-9}$)

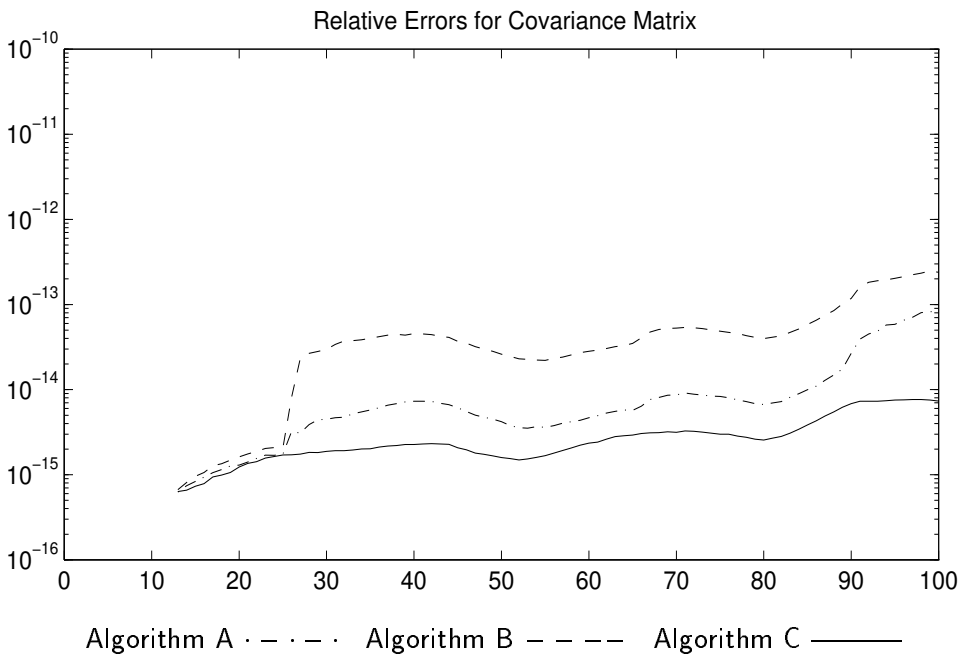


Figure 5: The plots of the relative errors for the covariance matrix vs. the window position for Test 4 ($\delta = 10^{-6}$ and $\gamma = 10^{-9}$)