DOWNDATING A RANK-REVEALING URV DECOMPOSITION*

YUAN-JYE JASON WU†

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 could be unstable because the R factor is ill-conditioned. In this paper, we review some existing downdating algorithms for the full-rank URV decomposition in the absence of U and develop a new combined algorithm. We also show that the combined algorithm has relational stability. For the rank-revealing URV decomposition, we review a two-step method that applies full-rank downdating algorithms to the signal and noise parts separately. We compare several combinations of the full-rank algorithms and demonstrate good performance of our combined algorithm.

Key words. rank-revealing factorization, downdating, URV decomposition

AMS(MOS) subject classifications. 65F20, 65F25, 65F30

August 31, 1995

1. Introduction. The rank-revealing URV decomposition [12] is a useful tool for subspaces tracking problems in digital signal processing. Updating the decomposition is a stable process requiring only $O(m^2)$ operations where m is the number of sensors. Applying this updating technique on data sampled by the exponential windowing method can have efficient and effective performance [4, 8].

In contrast to the exponential windowing method, some practical signal processing applications use the *rectangular windowing* method to collect the sample data matrix X. Since the sensors, or receivers, collect the data sequentially, a large set of data will be accumulated over time. Even with a small forgetting factor in the exponential windowing method, the earlier data might still distort or perturb certain estimates and lead to inaccurate results. For example, the location of a moving signal is better specified by the later data than by the earlier data, and it is better to reduce the effect of earlier data.

The rectangular windowing method multiplies the data x_i collected at time i, $i \leq t$ by a *window function* of size n defined as

$$w_{n,t}(i) = \begin{cases} 1 & \text{for } i = t, t-1, \ldots, t-n+1 \\ 0 & \text{otherwise} \end{cases} .$$

Thus the earlier data are truncated, and the data matrix will be always the same size. This function works like a window frame that only admits n pieces of data, and we shift it forward to use the most recent n samples of data. The shift is illustrated in Figure 1.

Let m be the length of the data vector. At time t, to compute a URV decomposition of the data matrix in the window frame requires addition and deletion of a row. Thus if there is a downdating algorithm that computes a URV decomposition of the matrix resulting from deleting the first data in the window at time $t-1$ and only requires $O(m^2)$ time, then applying the downdating and updating algorithms sequentially will yield a new URV decomposition in $O(m^2)$ time.

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

† Applied Mathematics Programs, University of Maryland, College Park, MD 20742. yunu@cs.umd.edu

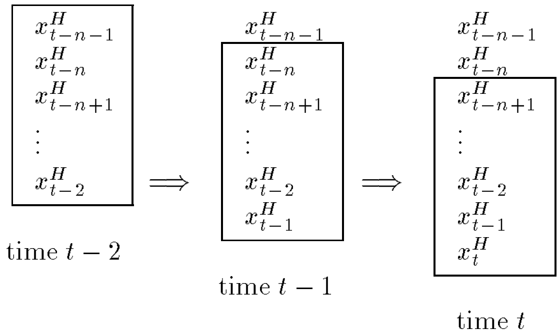


FIG. 1. *The shift of the n size window frame*

Consider an $n \times m$ data matrix X, where $n \geq m$. Then there exist unitary matrices U and V of order n and m respectively and an upper triangular matrix R of order m such that

$$X = \begin{bmatrix} x^H \\ \widehat{X} \end{bmatrix} = U \begin{bmatrix} R \\ 0 \end{bmatrix} V^H, \tag{1}$$

where x^H is the first row of the data matrix X. Let $Z = XV$ and $z^H = x^H V$. Suppose that we only consider downdating algorithms using two-sided orthogonal transformations. Then the downdating problem is to find an $m \times m$ upper triangular matrix T and unitary matrices Q and P such that

$$Q^H \begin{bmatrix} R \\ 0^H \end{bmatrix} P = \begin{bmatrix} T \\ z^H P \end{bmatrix}. \tag{2}$$

Let W be a permutation matrix that shifts the last row to be the first row. The product of two unitary matrices U and $\mathrm{diag}(QW, I_{n-m-1})$ will yield a new unitary matrix with the special structure

$$\begin{bmatrix} \mu & 0^H \\ 0 & \widehat{U} \end{bmatrix}, \tag{3}$$

where $|\mu| = 1$. Therefore, with $\widehat{V} = VP$, we have

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

as a URV decomposition of $\widehat{X}$.

For those methods described in [7] and [9], since the matrix U is explicitly saved, it can be easily modified to the form in (3). However, the matrix U is seldom saved in most applications because the window size is large. Therefore, we only discuss those algorithms that do not use U. In this paper, we first review some existing downdating algorithms for the full-rank URV decomposition and develop a new combined algorithm in §2. For the rank-revealing URV decomposition, we review a two-step method that applies full-rank downdating algorithms to the signal and noise parts separately in §3. We discuss the relational stability for those downdating algorithms in §4. The experimental results will be given in §5. Finally, we state conclusions in §6.

2. Full-rank algorithms. We first review several existing algorithms. Assume that the given data matrix X has full rank m. Then we design a new combined algorithm.

2.1. LINPACK and CSNE algorithms. Let us start from the algorithms without applying right plane rotations, i.e., $P = I$ in (2). In this case, $R^H R$ 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

$$T^H T = R^H R - zz^H ,$$

which amounts to downdating a Cholesky factorization. The method in the LINPACK package [6] described by Stewart [11] for downdating a Cholesky factorization is a popular choice to solve this kind of problem.

Note that $U[R^H 0]^H$ is a QR factorization of Z. The strategy of the LINPACK algorithm is to compute u^H, the first row of the matrix U, explicitly in order to form the new U factor with the structure in (3). 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 $[u_1 \cdots u_m]R = z^H$.

Actually, the first m components of each row of U can be computed in a similar way, and the first m columns of U are well determined. Therefore, if we partition U as

$$U = [\ \underset{m}{U_1} \quad \underset{n-m}{U_2}\ ] ,$$

then the only constraint for the last $(n - m)$ components of u^H is that they are the first components of vectors that form an orthonormal basis for the orthogonal complement of U_1. In order to simplify the calculation in downdating, we are free to choose $[\alpha\ 0 \cdots 0]$ as the last $(n - k)$ 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

$$[u_1 \cdots u_m\ \alpha]Q_m \cdots Q_1 = [0 \cdots 0\ \mu] , \tag{4}$$

with $|\mu| = 1$. Then the downdated triangular matrix T results from computing

$$(Q_m \cdots Q_1)^H \begin{bmatrix} R \\ 0^H \end{bmatrix} = \begin{bmatrix} T \\ z^H \end{bmatrix} . \tag{5}$$

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 (4).
4. Compute the downdated triangular matrix T using (5).

Let one flop be one 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. In order to have a more accurate result, Björck, Park, and Eldén [2] 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 $R\delta w = \delta\bar{u}$ and let $r = r - Z\delta w$.
5. Compute $\alpha_c = \|r\|_2$.
6. Determine plane rotations $Q_1, \ldots, Q_m$ satisfying (4).
7. Compute the downdated triangular matrix T using (5).

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 [5] also avoids applying right rotations. The idea is quite simple. With $P = I$ if we multiply (2) 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 only modifies 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 computations can be expressed as

$$\left[\begin{array}{cccc} r_{11} & r_{12} & \cdots & r_{1m} \\ 0 & \bar{z}_2 & \cdots & \bar{z}_m \end{array}\right] = \left[\begin{array}{cc} c & s \\ -s & c \end{array}\right] \left[\begin{array}{cccc} t_{11} & t_{12} & \cdots & t_{1m} \\ z_1 & z_2 & \cdots & z_m \end{array}\right], \tag{6}$$

where

$$c = \frac{t_{11}}{\sqrt{t_{11}^2 + z_1^2}}, \text{ and } s = \frac{z_1}{\sqrt{t_{11}^2 + z_1^2}}. \tag{7}$$

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 first have

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

Once t_{11} is known, the scalars c and s are computed by (7). From the first row in (6) for computing $r_{1i}, i = 2, \ldots, m$, we have

$$t_{1i} = (r_{1i} - s z_i)/c\,. \tag{8}$$

Applying the result in (8) to the second row in (6), we obtain

$$\bar{z}_i = c z_i - s t_{1i}\,. \tag{9}$$

Therefore, we have reduced the problem size by one with a new updated (or downdated) vector $[\bar{z}_2 \cdots \bar{z}_m]$. Repeating this process will yield the downdated upper triangular matrix T.

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 $(t_{k,k+1} \,\cdots\, t_{km})$ using (8).
  - Replace $(z_{k+1} \,\cdots\, z_m)$ by $(\bar{z}_{k+1} \,\cdots\, \bar{z}_m)$ in (9).
  - 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 non-positive for $k < m$, and there is no way to recover. When the breakdown happens at $k = m$, we know that t_{mm} is quite small and Park and Eldén [10] 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 this assumption is within an acceptable relative error bound.

2.3. Reduction algorithm. We now introduce an algorithm that applies right rotations, the reduction algorithm, described by Park and Eldén [10]. The reduction algorithm works on the problem (2) 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 the length of the downdated vector and reduces the problem size by one. When we apply P_k to the matrix R, we create a nonzero entry at the $(k+1, k)$ position. So we need a corresponding left plane rotation $\bar{Q}_k^H$ of order m in the $(k, k+1)$ plane to eliminate that nonzero entry. Therefore, the matrix

$$\bar{Q}_{m-1}^H \cdots \bar{Q}_1^H R P_1 \cdots P_{m-1} \tag{10}$$

remains upper triangular. Consequently, the problem size becomes only one and the downdated vector becomes a multiple of the last unit vector. The resulting matrix in the above equation is equal to the downdated triangular matrix T except at position (m, m). It is similar to the result in Chambers' algorithm after performing $m-1$ loops. Thus t_{mm} can be computed simply by taking the square root of the difference of the squares of the (m, m)-entry in (10) 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 $\bar{Q}_k$ 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 the algorithm breaks down.

2.4. Combined algorithm. 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. Actually, the LINPACK-type algorithms still depend on the condition number of R.

Chambers' algorithm has an attractive computational cost, but there is a risk of breakdown. On the other hand, with higher flops, the reduction algorithm has the advantage of avoiding breakdown. However, both algorithms have a common property: i.e., they reduce the problem size and compute the downdated triangular matrix T row by row. This suggests combining Chambers's algorithm and the reduction algorithm in order to obtain low cost and no breakdown.

The idea is to apply Chambers' algorithm first. If a breakdown occurs at $k < m$, we adopt one reduction step from the reduction algorithm to reduce the problem size by one. Then reapply Chambers' algorithm until the next breakdown. If the breakdown happens at $k = m$, we still let $t_{mm} = 0$. Since all the equations needed are derived in the previous subsections, 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 (8).

Replace $(z_{k+1} \ \cdots \ z_m)$ by $(\bar{z}_{k+1} \ \cdots \ \bar{z}_m)$ in (9).

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 $\bar{Q}_k$ 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 $t_{mm} = \begin{cases} 0 & \text{if } r_{mm}^2 - z_m^2 \leq 0 \\ \sqrt{r_{mm}^2 - z_m^2} & \text{if } r_{mm}^2 - z_m^2 > 0 \end{cases}$.

Step 1.2 is from Chambers' algorithm and step 1.3 is from the reduction algorithm. The complexity of the combined algorithm lies between $3m^2$ and $12m^2$, depending on how many reduction steps it takes.

3. Rank-revealing algorithms. Suppose that the data matrix X has numerical rank $d, d < m$, i.e. its singular values satisfy

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

Then Equation (1) is a rank-revealing URV decomposition of X if the matrix R has the form

$$R = \begin{bmatrix} R_s & F \\ 0 & G \end{bmatrix},$$

where

R_d 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

$||G||_F^2 + ||F||_F^2 \approx \sigma_{d+1}^2 + \cdots + \sigma_m^2$.

Since the matrix R has the data of the signal (R_s) and the noise (F and G) well separated, we have to preserve this signal-noise (or large-small) structure for the downdated triangular matrix T. Therefore, we change (2) 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

$$z^H = [\begin{array}{cc} z_s^H & z_n^H \\ d & m-d \end{array}], \quad \text{and} \quad P = [\begin{array}{cc} P_s & P_n \\ d & m-d \end{array}].$$

We can not directly apply both the reduction and 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. However, the large-small structure usually implies that R is ill-conditioned and leads to an inaccurate result or a breakdown.

Park and Eldén [10] give a simple and direct method called the two-step procedure to solve this problem. They consider only the LINPACK, CSNE, and reduction algorithms. Similar work is also studied by Barlow and Zha [1]. Since we already have algorithms for the full-rank problem, they suggest applying one of these methods to compute the signal (T_s) and noise (C) parts separately in order to keep the large-small structure unchanged. The only additional work required is a connection task. After computing T_s, we have to compute B and modify z_n^H for downdating the noise part. Then we can patch these two parts up to form the downdated triangular matrix T.

Note that those plane rotations that do not involve the vector z_s^H in downdating the signal part are also applied to the matrix F directly. Therefore, whenever we apply a rotation in a plane containing 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. Suppose that we have a plane rotation Q_1^H with rotation factors (c_1, s_1) applied to z_s^H and the first row of R. Denoting the unknown vectors by a bar, the corresponding computations on z_n^H and f^H (the first row of F) can be expressed as

$$\begin{bmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{bmatrix} \begin{bmatrix} f^H \\ \bar{z}_n^H \end{bmatrix} = \begin{bmatrix} \bar{f}^H \\ z_n^H \end{bmatrix}. \tag{11}$$

From the second row of the above equation, we have

$$\bar{z}_n^H = \frac{-s_1}{c_1} f^H + \frac{1}{c} z_n^H .$$

Substituting this result into the first row in (11), we obtain

$$\bar{f}^H = \frac{1}{c_1} f^H + \frac{-s_1}{c_1} z_n^H .$$

Therefore, we have a hyperbolic rotation

$$H_1 = \begin{bmatrix} 1/c_1 & -s_1/c_1 \\ -s_1/c_1 & 1/c_1 \end{bmatrix}$$

such that

$$\begin{bmatrix} \bar{f}^H \\ \bar{z}_n^H \end{bmatrix} = H_1 \begin{bmatrix} f^H \\ z_n^H \end{bmatrix} .$$

Since each plane rotation in downdating the signal part has a corresponding hyperbolic rotation, we have to save all the rotation factors (c_i, s_i) in the full-rank algorithm. Then we use the hyperbolic rotations to obtain the matrix B and the modified downdated vector for the noise part. For the LINPACK and CSNE, algorithms, we require d hyperbolic rotations. There is only one needed in the reduction algorithm.

However, the hyperbolic rotation is not recommended since it is not backward stable [3] [13]. Furthermore, if a breakdown occurs at the last step of the reduction algorithm, the assignment of 0 to t_{dd} implies a plane rotation with 90 degree rotation and the hyperbolic rotation cannot be completed. Park and Eldén leave Z_n^H and the last row of the corrected F unchanged and go on downdating the noise part.

In contrast to hyperbolic rotations, Equations (8) and (9) in Chambers' algorithm give an alternative way to perform the two-step method. Actually, it computes $[T_s\ B]$ and modifies z_n^H simultaneously. On the other hand, those left rotations in (10) for the reduction algorithm also are applied to the matrix F directly. This implies that the combined algorithm can be applied to the two-step method without using hyperbolic rotations. Note that the only difference between Chambers' algorithm and the reduction algorithm is the formula to modify the downdated vector for the noise part.

The combined algorithm still has the same trouble on the last row of the corrected F as in the reduction algorithm when t_{dd} is assigned to be 0. The assignment happens because rounding errors make the last component of the corrected z_s^H larger than the corrected r_{dd}. This means that the two scalars are within a small error bound and the assignment is an natural choice. Once the assignment is made, we can eliminate the last row of the corrected F with the diagonal entries of G by multiplying a sequence of left rotations.

This connection process using plane rotations refines the resulting matrix B and increases the norm of the matrix C so that the downdated triangular matrix T is more like a diagonal block matrix. Furthermore, the enlargement of the diagonal entries of G will increase the use of Chambers' algorithm instead of the reduction step in downdating the noise part and reduce some operation costs. In section 5, we will show the combined algorithm plus plane rotations make a good connection between the signal and noise parts.

Since we only apply a few more plane rotations, the complexity of the combined algorithm for the rank-revealing case is still $O(m^2)$. Therefore, we have an algorithm which will not break down in floating-point arithmetic so that the downdate is always computable.

One remark has to be noted in the rank-revealing case. Since one row is deleted from the original data matrix, it is possible that the resulting triangular matrix T_s has numerical rank degeneracy. We examine the resulting matrix T_s by applying the deflation algorithm defined in [12] 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 [7] and [9], none of the downdating algorithms introduced in this chapter is backward stable in the classical sense [14]. In fact, Björck, Park and Eldén [2] stated that no algorithm using the matrix R only to compute the required entries of the matrix U can be backward stable. However, Stewart [11] 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. Through out 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 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

$$\|E\| \lesssim k_m \|R\| \epsilon_M \ , \tag{12}$$

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

$$\widehat{Q}^H \begin{bmatrix} R \\ 0 \end{bmatrix} \widehat{P} = \begin{bmatrix} \tilde{T} \\ z^H \widehat{P} \end{bmatrix} + E \ . \tag{13}$$

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

It has been shown that in (12),

- $k_m = m^2/2 + 9m\sqrt{m} + O(m)$, for the LINPACK algorithm [11],
- $k_m = 4m\sqrt{m}$, for Chambers' algorithm [3].

On the other hand, algorithms involving hyperbolic rotations do not have relational stability because the parameter k_m in (12) is not bounded and depends on the tangents of rotation angles [3]. 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 $|\epsilon|, |\epsilon_1|, |\epsilon_2| \le \epsilon_M$. 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 only need 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

$$\|\Delta y^H\| \equiv \|\tilde{y}^H - z^H \widehat{P}_1 \cdots \widehat{P}_{m-1}\| \lesssim 6(m-1)\|z\|\epsilon_M \,. \tag{14}$$

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

$$\tilde{T}' = fl^{\,2m-2}(\tilde{Q}^H_{m-1}(\cdots(\tilde{Q}^H_1(R\tilde{P}_1))\cdots\tilde{P}_{m-1})) \,. \tag{15}$$

(Here the left rotations are of order m which is one less than those in (2) 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 post-multiplications effects 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 (14) and norm property, we have

$$\|R' - R\widehat{P}_1 \cdots \widehat{P}_{m-1}\| \lesssim 6(m-1)\|R\|\epsilon_M \,. \tag{16}$$

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

$$\|\tilde{T}' - \widehat{Q}^H_{m-1} \cdots \widehat{Q}^H_1 R'\| \lesssim 6(m-1)\|R'\|\epsilon_M \,, \tag{17}$$

Applying the triangular inequality to (16), we have

$$\|R'\| \leq (\sqrt{m} + 6(m-1)\epsilon_M)\|R\| \,, \tag{18}$$

where the $\sqrt{m}$ comes from taking the Frobenius norm of a unitary matrix. Therefore, combining (16),(17), and (18), we have

$$\begin{aligned}
\|\Delta\tilde{T}'\| &\equiv \|\tilde{T}' - \widehat{Q}^H_{m-1} \cdots \widehat{Q}^H_1 R \widehat{P}_1 \cdots \widehat{P}_{m-1}\| \\
&\leq \|\tilde{T}' - \widehat{Q}^H_{m-1} \cdots \widehat{Q}^H_1 R'\| + \|\widehat{Q}^H_{m-1} \cdots \widehat{Q}^H_1 (R' - R\widehat{P}_1 \cdots \widehat{P}_{m-1})\| \\
&\lesssim 6(m-1)\|R'\|\epsilon_M + \sqrt{m}\|R' - R\widehat{P}_1 \cdots \widehat{P}_{m-1}\| \\
&\lesssim 6(m-1)\epsilon_M(\sqrt{m} + 6(m-1)\epsilon_M)\|R\| + 6(m-1)\sqrt{m}\epsilon_M\|R\|
\end{aligned}$$

Neglecting the ϵ_M^2 term, we have that

$$\|\Delta\tilde{T}'\| \lesssim 12(m-1)\sqrt{m}\epsilon_M\|R\|. \tag{19}$$

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

$$
\begin{aligned}
|\epsilon_1| &\lesssim 12(m-1)\sqrt{m}\epsilon_M\|R\| \text{ from (19)},\\
|\epsilon_2| &\lesssim 6(m-1)\|z\|\epsilon_M \text{ from (14), and}\\
|\epsilon_3|,\ |\epsilon_4|,\ |\epsilon_5|,\ |\epsilon_6| &\leq \epsilon_M \text{ from floating-point operations rules.}
\end{aligned}
$$

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

$$
|\Delta\tilde{t}_{mm}| \lesssim [12(m-1)\sqrt{m}+2]\epsilon_M\|R\| \,. \tag{20}
$$

Note that $\tilde{t}_{mm}$ should be non-negative 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 (14), (19), and (20), we derive a relational error bound for the reduction algorithm as

$$
\begin{aligned}
\|E\| &\leq \sqrt{\|\Delta\tilde{T}'\|^2 + |\Delta\tilde{t}_{mm}|^2 + \|\Delta\tilde{y}^H\|^2}\\
&\lesssim [12(m-1)\sqrt{m} + O(m)]\epsilon_M\|R\| \,. \qquad (21)
\end{aligned}
$$

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

Finally, we check the algorithm for the rank-revealing case. The additional work is to eliminate the last row of the corrected F. We only need to apply $m-d$ more left rotations in (15). This adds $6(m-d)\sqrt{m}$ to the coefficients in (17), (19), and (20). So the final coefficient in (21) becomes $18(m-1)\sqrt{m}+O(m)$. The rank-revealing combined algorithm also has relational stability.

5. Experimental results. In this section, we show some experimental results using the two-step method for the rank-revealing downdating problem. There are several combinations from those full-rank algorithms that 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	Combined /Plane
Noise	LINPACK/Reduction	Reduction	Combined

Note that the LINPACK algorithm cannot be present alone in any phase because of its breakdown at step 2. We have to use a backup algorithm to recover, like the CSNE and reduction algorithms in Algorithm A.

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.

In order to estimate the numerical rank, we need a tolerance described in [12]. 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 [8] 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

$tol\_u = \psi_u * \delta * \sqrt{12(8-d)}$, for the updating algorithm,

$tol\_d = \psi_d * \delta * \sqrt{12(8-d+1)}$, 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

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

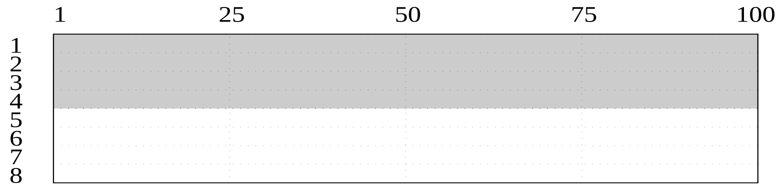
where $\tilde{T}_s$ is the computed T_s. Let $Z = W\Sigma Y^H$ be the singular value decomposition of the matrix Z. For the accuracy of the noise part, we compute the sum of the sins of the canonical angles between the subspaces spanned by the last $8-d$ columns of the matrices V and Y. Finally, we show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

In order to make a fair comparison, we ran 50 trials for each test and show the average results. The average costs over 50 trials, 88 downdates per trial, for each test are given in Table 3.

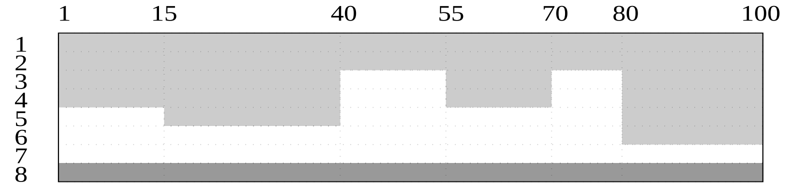
Test 1 & 2: Our first test matrix has a fixed numerical rank of 4. The test matrix K^H is constructed as



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. Table 1 and 2 show the average results of the rank estimates, the relative error norms, and the condition numbers of the matrix R. No breakdown occurred, so the LINPACK algorithm is always used in Algorithm A. All three algorithms give good results. However, Table 3 shows that the average cost of Algorithm C is less than the other two.

In order 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

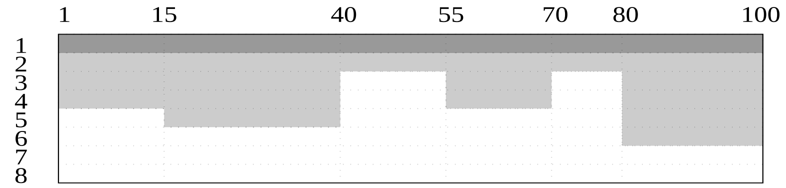


where the light gray area is multiplied by $\delta = 10^{-7}$ and the dark gray area is multiplied by $\gamma = 10^2$. The scalar γ makes R_s ill-conditioned around those positions with a sharp rank drop. We choose the factor $\psi_d = 20$. The results are given in Figure 2 and Figure 3.

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 and 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 is similar 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.

7. Acknowledgements. I thank my thesis advisor, Dianne P. O'Leary, for her very helpful comments.

REFERENCES

[1] 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.

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

[3] A. W. Bojanczyk, R. P. Brent, P. V. Dooren, and F. R. D. Hoog, *A note on downdating the Cholesky factoration*, SIAM Journal on Scientific and Statistical Computing, 8 (1987), pp. 210–221.

[4] E. C. Boman, M. F. Griffen, and G. W. Stewart, *Direction of arrival and the rank-revealing URV decomposition*, in Procedings of ACASSP-91, Washington, DC, 1991, IEEE.

[5] J. M. Chambers, *Regression updating*, Journal of the American Statistical Association, (1971), pp. 744–748.

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

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

[8] K. J. R. Liu, D. P. O'Leary, G. W. Stewart, and Y.-J. J. Wu, *URV ESPRIT for tracking time-varying signals*, IEEE Trans. Signal Processing, 42 (1994), pp. 3441–3448.

[9] S. J. Olszanskyj and A. W. Bojanczyk, *Compact Givens representation of the orthogonal factor in recursive linear squares*, in Proceedings of the fifth SIAM conference on Applied Linear Algebra, J. G. Lewis, ed., SIAM, 1994.

[10] H. Park and L. Eldén, *Downdating the rank-revealing URV decomposition*, SIAM Journal on Matrix Anal. and Appl., 16 (1995), pp. 138–155.

[11] G. W. Stewart, *The effects of rounding error on an algorithm for downdating a Cholesky factorization*, Journal of the Institute for Mathematics and Applications, 23 (1979), pp. 203–213.

[12] ———, *An updating algorithm for subspace tracking*, IEEE Transactions on Signal Processing, 40 (1992), pp. 1535–1541.

[13] ———, *On the stability of sequential updates and downdates*, tech. 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	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 1

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 2

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

	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 3

Average operations count (flops) for all tests.

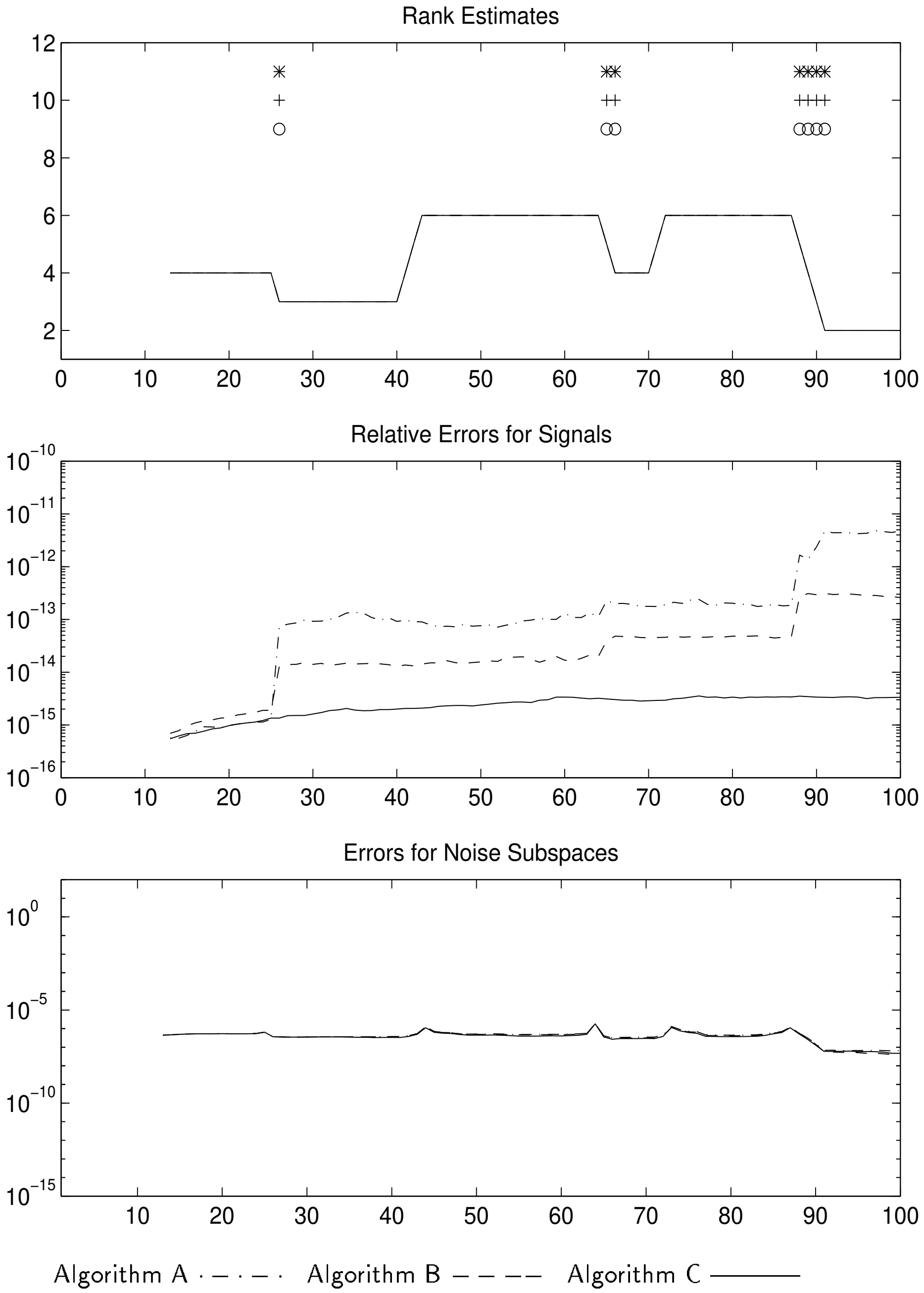


FIG. 2. *The plots of the estimated rank, the signal error, and the noise error vs. the window position for Test 3 ($\delta = 10^{-7}$ and $\gamma = 10^2$).*

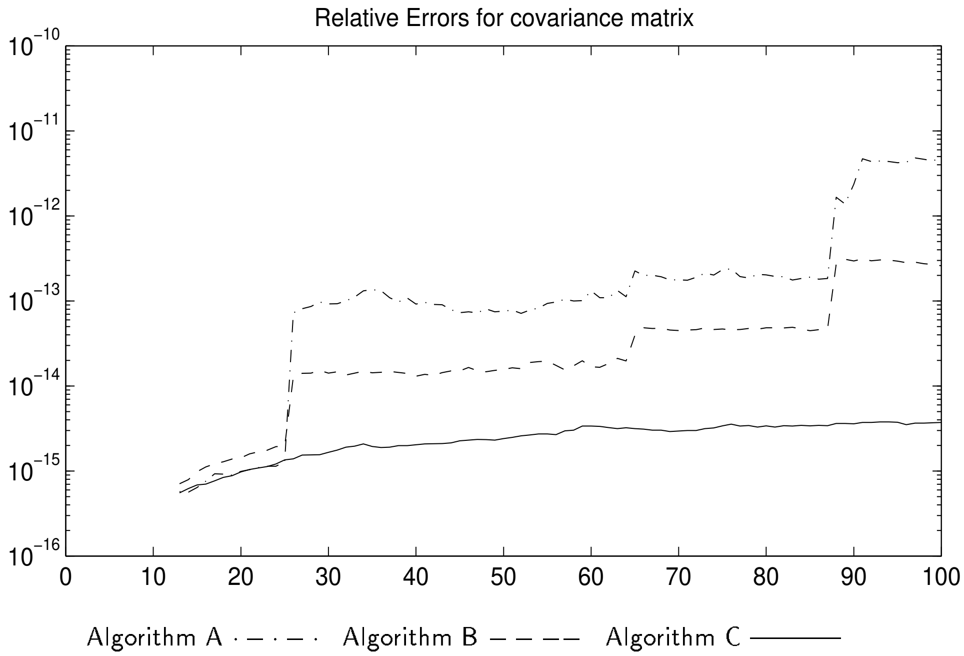


FIG. 3. *The plots of the relative errors for the covariance matrix vs. the window position for Test 3 ($\delta = 10^{-7}$ and $\gamma = 10^2$).*

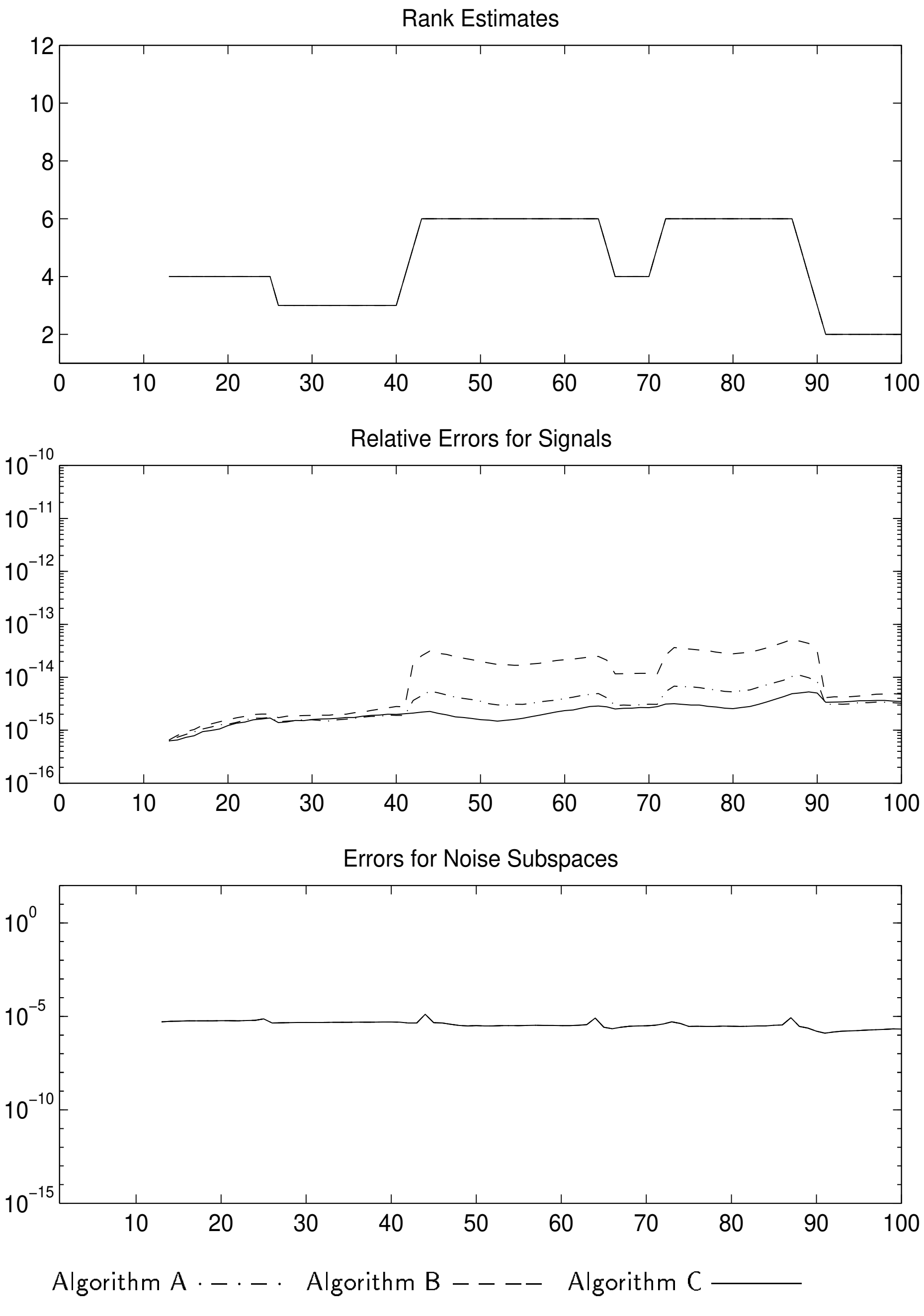


FIG. 4. *The plots of the estimated rank, the signal error, and the noise error vs. the window position for Test 4 ($\delta = 10^{-6}$ and $\gamma = 10^{-9}$).*

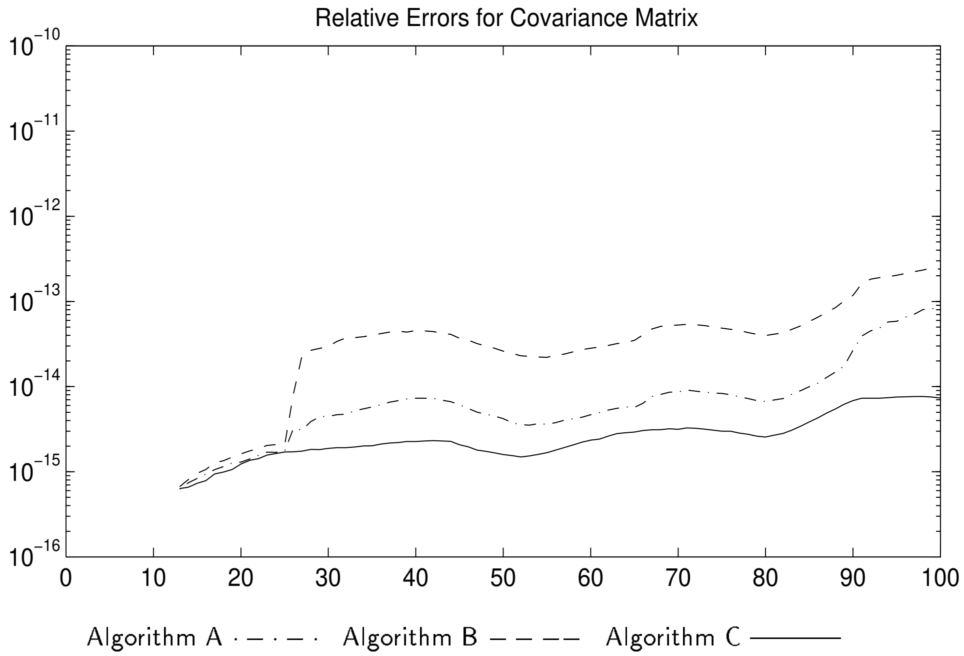


FIG. 5. *The plots of the relative errors for the covariance matrix vs. the window position for Test 4 ($\delta = 10^{-6}$ and $\gamma = 10^{-9}$).*