SIAM J. MATRIX ANAL. APPL.
Vol. 22, No. 2, pp. 479–500

UPDATING A GENERALIZED URV DECOMPOSITION[*]

MICHAEL STEWART[†] AND PAUL VAN DOOREN[‡]

Abstract. An updating scheme for a quotient type generalization of a URV decomposition of two matrices is introduced. This decomposition allows low complexity updating as rows are added to two rectangular matrices, determining the dimension of three distinct subspaces. One of these subspaces is the intersection of the range space of the two matrices—information which leads to a potential application in subspace algorithms for system identification.

Key words. generalized SVD, URV decomposition, system identification

AMS subject classifications. 65F15, 65F20

PII. S0895479897320460

1. Introduction.

The quotient singular value decomposition of two matrices, A and B, with an equal number of rows m, has been described in several ways. The justification for the name is most obvious when A and B are both square and of the same size and when A has full rank. Suppose an application demands knowledge of the singular values of $A^{-1}B$. It is well known in the context of the generalized eigenvalue problem, in which the goal is to find the eigenvalues of $A^{-1}B$, that the best approach is to compute the eigenvalues by applying orthogonal transformations to A and B without explicitly computing $A^{-1}B$. This also applies to the singular value problem and, instead of computing $A^{-1}B$, a more reasonable approach is to directly compute invertible X and orthogonal V_A and V_B such that

$$X^{-1}AV_A = \Sigma_A, \qquad X^{-1}BV_B = \Sigma_B,$$

where Σ_B and Σ_A are diagonal. This solves the problem, since

$$A^{-1}B = V_A \Sigma_A^{-1} \Sigma_B V_B^T$$

is clearly the singular value decomposition of $A^{-1}B$. If X is required to be orthogonal, then the best that can be done is to make Σ_A and Σ_B triangular. An appropriate choice of orthogonal X, V_A, and V_B guarantees that $\Sigma_A^{-1}\Sigma_B$ will be diagonal.

More generally, when A and B are possibly rank deficient $m \times n_a$ and $m \times n_b$ matrices, the generalized SVD [10, 13] has been defined by

$$X^{-1}AV_1 = \begin{array}{c}\begin{bmatrix}\Sigma_A \\ 0\end{bmatrix}\end{array}\begin{array}{l} r \\ m-r\end{array}, \qquad X^{-1}BV_2 = \begin{bmatrix}\Sigma_B \\ 0\end{bmatrix}\begin{array}{l} r \\ m-r\end{array}, \tag{1.1}$$

where

$$\Sigma_A = \begin{bmatrix} I_A & & \\ & S_A & \\ & & 0_A \end{bmatrix}, \qquad \Sigma_B = \begin{bmatrix} 0_B & & \\ & S_B & \\ & & I_B \end{bmatrix}$$

[*]Received by the editors April 23, 1997; accepted for publication (in revised form) by L. Eldén September 11, 1998; published electronically August 9, 2000. This work was supported by ARPA grant 60NANB2D1272 and NSF grant CCR-9209349.
http://www.siam.org/journals/simax/22-2/32046.html
[†]Computer Sciences Laboratory, RSISE, Australian National University, Canberra, ACT 0200, Australia (stewart@discus.anu.edu.au).
[‡]CESAME, Université Catholique de Louvain, Louvain-la-Neuve, Belgium (vdooren@anma.ucl.ac.be).

with diagonal positive definite S_A and S_B satisfying $S_A^2 + S_B^2 = I$ and where r is the rank of $\begin{bmatrix} A & B \end{bmatrix}$. The partitionings are such that S_A and S_B are the same size, r_3. The identity matrices I_B and I_A are $r_2 \times r_2$ and $(r_1 - r_3) \times (r_1 - r_3)$, where r_1 is the rank of A. The zero blocks 0_A and 0_B are $(r - r_1) \times (n_a - r_1)$ and $(r - r_2 - r_3) \times (n_b - r_2 - r_3)$. The decomposition reveals that r_3 is the dimension of the intersection of the range spaces of A and B.

In the rectangular case in which A has full rank the decomposition reveals the singular values of $A^\dagger B$, where $A^\dagger$ denotes the pseudoinverse of A. If A is rank deficient, then the decomposition reveals singular values associated with a quotient formed from the B-weighted pseudoinverse of A [4, 2].

An early development of the generalized SVD was given in [10]. A general description suitable for adaptation to a URV decomposition is as follows: the $m \times (n_a + n_b)$ matrix $\begin{bmatrix} A & B \end{bmatrix}$ is decomposed as

$$U^T \begin{bmatrix} A \parallel B \end{bmatrix} V = U^T \begin{bmatrix} A \parallel B \end{bmatrix} \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix} = \left[\begin{array}{cc||ccc} R_{11} & 0 & S_{13} & R_{14} & 0 \\ 0 & 0 & R_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right], \tag{1.2}$$

where R_{11} is $r_1 \times r_1$ and upper triangular with full rank and R_{23} is $r_2 \times r_2$ and upper triangular with full rank. The rectangular block

$$R_{14} = \begin{bmatrix} \hat{R}_{14} \\ 0 \end{bmatrix}$$

is $r_1 \times r_3$ with full column rank r_3, and $\hat{R}_{14}$ is square and upper triangular. Clearly the first r_3 columns of U form a basis for the intersection of the range spaces. Since the rank of A is clearly r_1 and the rank of B is $r_2 + r_3$, this decomposition reveals the same rank information as the quotient SVD.

Further, if we define $\hat{R}_{11}$ to be the $r_3 \times r_3$ leading principal submatrix of R_{11} and

$$\hat{R}_{11}^{-1} \hat{R}_{14} = V_{11} \Sigma_R V_{14}^T$$

to be the SVD of $\hat{R}_{11}^{-1} \hat{R}_{14}$, then

$$\hat{R}_{11} V_{11} = \hat{R}_{14} V_{14} \Sigma_R^{-1},$$

and consequently there exists an orthogonal U_R such that $U_R^T \hat{R}_{11} V_{11}$ and $U_R^T \hat{R}_{14} V_{14}$ are both upper triangular. Applying U_R, V_{11}, and V_{14} to the relevant rows and columns of (1.2) will maintain the structure of (1.2) while ensuring that $\hat{R}_{11}^{-1} \hat{R}_{14}$ will be diagonal. Note that the singular values of $\hat{R}_{11}^{-1} \hat{R}_{14}$ are not changed by these further orthogonal transformations.

If U and V are required to be orthogonal, then this is the most condensed form one can obtain. However, if we define

$$X^{-1} = \begin{bmatrix} R_{11}^{-1} & -R_{11}^{-1} S_{13} R_{23}^{-1} & 0 \\ 0 & R_{23}^{-1} & 0 \\ 0 & 0 & I \end{bmatrix} U^T,$$

then

$$X^{-1} \begin{bmatrix} A \parallel B \end{bmatrix} V = \left[\begin{array}{ccc||ccc} I & 0 & 0 & 0 & \hat{R}_{11}^{-1} \hat{R}_{14} & 0 \\ 0 & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

If the original orthogonal transformations were chosen so that $\hat{R}_{11}^{-1}\hat{R}_{14}$ is diagonal and positive definite, then clearly $\hat{R}_{11}^{-1}\hat{R}_{14} = S_A^{-1}S_B$. Thus up to permutations and scaling by S_A^{-1} we recover the quotient SVD as presented in [10] and we can conclude that the quotient singular values of A and B are the singular values of $\hat{R}_{11}^{-1}\hat{R}_{14}$ even when $\hat{R}_{11}^{-1}\hat{R}_{14}$ is not diagonal. This is the justification for viewing the decomposition as a quotient type generalization of the URV decomposition.

In this paper we do not require diagonality of $\hat{R}_{11}^{-1}\hat{R}_{14}$ and we present an algorithm to efficiently update a rank revealing decomposition that is related to (1.2) when rows are added to the matrices A and B. An obvious application is recursive identification of MIMO systems. The algorithm in [7] requires the intersection of the range spaces of two matrices and may be adapted for use with the decomposition. A summary of the main idea will be presented in section 5. Further details are in [12].

Other papers have considered updating for a quotient generalization of the ULV decomposition [5] in the case in which A and B are $n_a \times m$ and $n_b \times m$ with $n_a, n_b \geq m$ and the update involves the addition of rows to A and B. In the formulation chosen in this paper in which the matrices have an equal number of rows, this is equivalent to updating under the addition of columns to A and B. The method was first proposed in [5] for the case in which A has full rank and extended in [6] to the rank deficient case. A natural application for these decompositions is in prewhitening of colored noise in signal processing [4]. Because of the assumptions on the dimensions of the matrices in [5] and [6] and the difference between updating under the addition of columns and rows, the algorithms considered in this paper are substantially different from the previous work on generalized ULV decompositions.

The set of all rank deficient matrices is a subset of measure zero in the set of all matrices. If $m \geq n_a + n_b$, then A and B can have nonempty range space intersection only when $\begin{bmatrix} A & B \end{bmatrix}$ is rank deficient. Moreover, A or B can be rank deficient only when $\begin{bmatrix} A & B \end{bmatrix}$ is rank deficient. It follows that when $m \geq n_a + n_b$ the decomposition (1.2) has the form

$$U^T \left[A \parallel B\right] V = U^T \left[A \parallel B\right] \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix} = \left[\begin{array}{c||c} R_{11} & S_{13} \\ 0 & R_{23} \\ 0 & 0 \end{array}\right] \tag{1.3}$$

except on a measure zero set. This represents the special case of (1.2) when $r_1 = n_a$, $r_2 = n_b$, and $r_3 = 0$. Because of numerical errors or noisy data, we can always expect a quotient URV to have the trivial structure (1.3).

Thus an exact quotient URV gives no real information about the relation between the numerical range spaces of A and B. Instead of computing the exact quotient URV, we will attempt to compute a rank-revealing decomposition that shows when small perturbations give a nontrivial quotient URV structure of the type (1.2). The perturbations we allow will take the form of small nonzero elements in some of the blocks of (1.2) that were previously zero. We constrain U and V to be orthogonal and drop the requirement that $\hat{R}_{11}^{-1}\hat{R}_{14}$ must be diagonal. The result is

$$U^T \left[A \parallel B\right] \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix} = \left[\begin{array}{cc||ccc} R_{11} & E_{12} & S_{13} & R_{14} & E_{15} \\ 0 & E_{22} & R_{23} & E_{24} & E_{25} \\ 0 & E_{32} & 0 & F_{34} & E_{35} \\ 0 & E_{42} & 0 & 0 & F_{45} \\ 0 & F_{52} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right]. \tag{1.4}$$

The blocks R_{11} and R_{23} are square and upper triangular. In (1.2), R_{14} was upper triangular with full column rank. To make the updating easier, we modify this in (1.4): we allow R_{14} to have potentially large elements arranged in the form of an upper triangular matrix that has had its columns reversed. The rest of the elements can be nonzero but are constrained to be below a prespecified tolerance. The elements below the cross diagonal are kept small enough that they will not cause the block to become rank deficient when the triangular part R_{14} is kept suitably well conditioned by an appropriate method for deflating small singular values. Letting r represent a potentially large element and e represent an element which is small relative to the tolerance, the 4×3 case of R_{14} looks like

$$R_{14} = \begin{bmatrix} r & r & r \\ r & r & e \\ r & e & e \\ e & e & e \end{bmatrix}.$$

Although the structure of R_{14} may seem odd, it turns out that both the permuted triangular structure and the possibility of having small nonzeros below the cross diagonal significantly simplify the updating algorithm.

In further examples, we will follow the convention used for describing the structure of R_{14}. An r will always represent a potentially large element and an e will represent an element which is small relative to the tolerance. The algorithm will keep the elements which should be small from growing inappropriately.

Each F block of the decomposition is an upper triangular matrix with norm of the order of the tolerance. Each E block is an arbitrary matrix, also with norm of the order of the tolerance. The S block is an arbitrary matrix. With sufficiently small E and F blocks, the decomposition gives estimates of the numerical range spaces of A and B, along with an estimate of the numerical intersection in the form of the basis provided by the first r_3 columns of U.

The justification for the small nonzero blocks in (1.4) is that they allow us to find a nontrivial quotient URV structure associated with a slightly perturbed pair of matrices. The locations of these blocks are chosen to facilitate updating. The algorithm will be designed to perform deflations of small singular values using a tolerance that will keep these elements suitably small.

However, in some applications this might not be sufficient. If we wish to reliably identify the most rank deficient nearby quotient SVD structure corresponding to the smallest ranks for A, B, and $\begin{bmatrix} A & B \end{bmatrix}$, then it is natural to expect these perturbations, and hence the magnitudes of the E and F blocks, to be not much larger than the smallest perturbations to A and B required to give a quotient URV structure of the form (1.2). Inordinately large elements in this blocks might cause the tolerance to be reached too early in the deflation process, leading to an overestimate of the ranks of A and B and an underestimate of the intersection dimension.

Unfortunately, standard quotient QR and URV algorithms fail by this standard and the method of this paper suffers from a similar problem. The difficulty centers on the fact that R_{23} is a part of a URV decomposition,

$$\begin{bmatrix} R_{23} & E_{24} & E_{25} \\ 0 & F_{34} & E_{35} \\ 0 & 0 & F_{45} \end{bmatrix}, \tag{1.5}$$

that estimates a numerical rank for $P_A^{\perp} B$. The exact decomposition (1.2) reveals the exact rank of $P_A^\perp B$. When the range space of A is sufficiently sensitive to perturbations, small perturbations of A can lead to large changes in the small singular values of $P_A^\perp B$ that correspond to the exactly zero singular values shown in the unperturbed (1.2). Thus, even when dealing with small perturbations to a matrix pair with the exact structure (1.2), the E and F blocks in (1.5) might be significantly larger than the original perturbations to the data.

We illustrate the problem with the matrix pair

$$A = \begin{bmatrix} 1 & 0 \\ 0 & \delta \\ 0 & \epsilon \\ 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \tag{1.6}$$

where $0 < \epsilon < \delta < 1$. We suppose that δ is significantly smaller than 1 but that it is large enough that A can be considered to have full numerical rank. We assume that the perturbing quantity ϵ is small enough that it is of the same order as the tolerance used in rank decisions. A perturbation of norm ϵ to A clearly results in two full rank matrices with an exact one-dimensional row subspace intersection.

Consider the orthogonal transformation given by the QR factorization of A,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\delta}{\sqrt{\delta^2+\epsilon^2}} & \frac{\epsilon}{\sqrt{\delta^2+\epsilon^2}} & 0 \\ 0 & \frac{-\epsilon}{\sqrt{\delta^2+\epsilon^2}} & \frac{\delta}{\sqrt{\delta^2+\epsilon^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & \delta & 1 & 0 \\ 0 & \epsilon & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & \sqrt{\delta^2+\epsilon^2} & \frac{\delta}{\sqrt{\delta^2+\epsilon^2}} & 0 \\ 0 & 0 & \frac{-\epsilon}{\sqrt{\delta^2+\epsilon^2}} & 0 \\ 0 & 0 & 0 & 1 \end{array}\right].$$

This decomposition gives the SVD of $P_A^\perp B$. The smallest singular value is $\epsilon/\sqrt{\delta^2+\epsilon^2}$. If δ is sufficiently small, we would conclude that $P_A^\perp B$ has full rank. This would imply that $r_i = 0$, so that the algorithm completely misses the possibility that there is a nontrivial range space intersection achieved by matrices within $O(\epsilon)$ of A and B. The end result is a misleadingly partitioned quotient URV that fails to reveal an interesting and potentially useful feature of A and B.

Sensitivity in rank decisions is fundamental to any generalized URV or generalized QR algorithm that starts with an estimate of the range space of A and proceeds with a rank decision for $P_A^\perp B$, including the methods described in [9, 1]. The algorithms in [5, 6] are somewhat different in that they make rank decisions on a matrix with singular values equal to the generalized singular values of A and B. Since generalized singular values can be sensitive to perturbations [8], the rank decisions in these methods can also be difficult.

Although the updating problem considered here is more involved, the basic tools needed to update this decomposition have already been developed for the problem of updating a URV decomposition in [11]. The algorithm can be broken into two stages. The first restores the form of the decomposition when new rows are added to A and B. After this stage of the update, the decomposition has the same general form, but the triangular R matrices are potentially of different size and might no longer have full rank. The second stage looks for small singular values of the R blocks and recursively deflates these blocks, using the scheme described in [11], until they have full rank.

When it is obvious after representing the new rows in the bases provided by V_1 and V_2 that the new information does not increase the ranks of any of the full rank blocks, parts of the updating algorithm are not needed. This is essentially the same simplification as appears in [11]. To avoid giving the details of too many special cases,

$$
\begin{array}{cccc|cc||cc|cc|cc}
r & r & r & r & r & r_2 & r & r & r & r & r & r_1 \\
\hline
r & r & r & r & e & e & r & r & r & r & e & e \\
0 & r & r & r & e & e & r & r & r & e & e & e \\
0 & 0 & r & r & e & e & r & r & e & e & e & e \\
0 & 0 & 0 & r & e & e & r & r & e & e & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & r & r & e & e & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & r & e & e & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & 0 & 0 & e & e & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & e & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & 0_1 & e \\
\hline
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0_2 & e & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

FIG. 2.1. *An example.*

we deal only with the most general and the most difficult case here. This algorithm applies in every contingency, but if it is immediately obvious that the new data will not significantly change the estimates of the subspaces, then some of the steps are unnecessary. We will be more precise about which steps can be skipped in section 3.

2. The updating algorithm. We first describe how to restore the general structure after new rows are added to A and B, leaving the discussion of deflation for section 3. We also ignore initialization issues by assuming that at some stage the decomposition has already been computed and we are simply interested in computing the update. This does not evade the description of an essential step since the algorithm applies in degenerate cases when the sizes of some of the triangular matrices are zero (although this might involve the elimination of superfluous steps). Thus, the process can be initialized by setting the decomposition to zero, setting the unitary matrices to the identity, and starting the algorithm with the first rows of A and B. It could also start at some later point by applying a more conventional generalized SVD algorithm to compute the initial decomposition.

If two rows, a^T and b^T, are added to A and B, respectively, and each row of the old matrix is weighted by $0 < \alpha < 1$, then

$$
\begin{bmatrix} 1 & 0 \\ 0 & U^T \end{bmatrix}
\left[\begin{array}{c||c} a^T & b^T \\ \alpha A & \alpha B \end{array}\right]
\begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix}
=
\left[\begin{array}{cc||ccc}
a_1^T & a_2^T & b_3^T & b_4^T & b_5^T \\
R_{11} & E_{12} & S_{13} & R_{14} & E_{15} \\
0 & E_{22} & R_{23} & E_{24} & E_{25} \\
0 & E_{32} & 0 & F_{34} & E_{35} \\
0 & E_{42} & 0 & 0 & F_{45} \\
0 & F_{52} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{array}\right], \tag{2.1}
$$

where a_i^T and b_i^T are the obvious partitionings of $a^T V_1$ and $b^T V_2$. The blocks of the decomposition shown here are the same as those shown in (1.4) but weighted by α. The problem is to update the orthogonal matrices U and V to restore the structure of the decomposition and to deal with possible rank changes in the R matrices.

To illustrate how we can restore the appropriate structure, we take as an example Figure 2.1. This shows an extra row added to the top of a matrix that has the general form described in (1.4). We may efficiently restore the original structure through the application of sequences of Givens rotations of the form

$$G = \begin{bmatrix} c & s \\ -s & c \end{bmatrix},$$

where $c^2 + s^2 = 1$, to the appropriate rows and columns of the matrix in Figure 2.1. A description of how to compute Givens rotations to introduce zeros in a numerically reliable manner may be found in [3].

When showing the updating of the decomposition, the approach taken here for dealing with Givens rotations is to mark the elements that are to be eliminated with a number indicating their order and to give any additional information in the text. Such information includes whether the rotation acts on the row or column containing the marked element, along with which other row or column the rotation also acts on. In this algorithm, the rotations will always act on a row or column that is adjacent to the numbered row or column in addition to the numbered row or column itself. The identity of the adjacent row or column is not always explicitly mentioned in the text, but the examples should make this detail of the procedure clear.

Many of the rotations will destroy the structure of a block of the decomposition so that it is sometimes necessary to apply additional rotations to fix this damage. The elements that must be eliminated to fix the structure will be marked with the same number as the rotation that originally did the damage. Occasionally there will be a sequence of two such fixes beyond the original rotation. The fix will always be applied on the opposite side of the rotation which originally did the damage. Thus damage caused by rotations applied on the left are fixed by rotations applied on the right and vice versa. All such rotations can be easily spotted, since they always correspond to either a marked zero element or one of the small elements of the R_{14} block—elements for which no rotation would be needed if they had not been made potentially large by another rotation.

In Figure 2.1 the numbered elements represent two sequences of right rotations to zero the marked elements. Rotation 1 acts on the column of the numbered element and the preceding column to eliminate the marked element, destroying the triangularity of the F_{45} block. It can be restored after the Givens rotation is applied from the right by a rotation from the left. Rotation 2 destroys the triangular structure of the F_{52} block. This can also be maintained through the appropriate use of a left rotation. In a more general setting, rotations 1 and 2 would each be replaced by multiple rotations that are intended to zero all but the first element of a_2^T and b_5^T, respectively, and, after each rotation, it would be necessary to apply an additional rotation to fix F_{52} and F_{45}

The result of these rotations is the matrix shown in Figure 2.2. Now that rotations have been applied to concentrate large elements from the new rows into a region in which they can damage at most two columns, we can take advantage of the permuted triangular structure of the overall decomposition and apply a sequence of rotations that are essentially the same as those used in QR updating. Each numbered rotation, except for those in R_{14}, acts on the numbered row and the preceding row to introduce the necessary zeros. The only additional complication is the need to preserve the triangular structure of R_{14}. Figure 2.2 marks the elements to be eliminated by left rotations. Each left rotation operates on the row marked and the previous row to eliminate the marked element. The first r_3 rotations, rotations 1 and 2 in this example, will destroy the structure of the R_{14} block. For rotation 1, we may fix the damage to R_{14} by using a right rotation on the numbered column and the one before it and then

$$\begin{array}{cccc|cc||cc|cc|cc}
r & r & r & r & r & 0 & r & r & r & r & r & 0 \\
\hline
r_1 & r & r & r & e & e & r & r & r & r & e_1 & e \\
0 & r_2 & r & r & e & e & r & r & r & e_2 & e & e \\
0 & 0 & r_3 & r & e & e & r & r & e & e & e & e \\
0 & 0 & 0 & r_4 & e & e & r & r & e & e & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & r_5 & r & e & e & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & r_6 & e & e & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & 0 & 0 & e_7 & e & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0_2 & e_8 & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0_1 & e_9 & e \\
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & 0 & e_{10} \\
\hline
0 & 0 & 0 & 0 & e_{11} & e & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & e_{12} & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

FIG. 2.2. *A sequence of left rotations.*

$$\begin{array}{ccccc|c||ccc|cc|c}
r & r & r & r & r & e & r & r & r & r & r & e \\
0 & r & r & r & r & e & r & r & r & r & e & e \\
0 & 0 & r & r & r & e & r & r & r & e & e & e \\
0 & 0 & 0 & r & r & e & r & r & r & e & e & e \\
0 & 0 & 0 & 0 & r & e & r & r & r & e & e & e \\
\hline
0 & 0 & 0 & 0 & r_6 & e & 0 & r & r & e & e & e \\
0 & 0 & 0 & 0 & r_5 & e & 0 & 0 & r & e & e & e \\
0 & 0 & 0 & 0 & r_4 & e & 0 & 0 & 0 & e & e & e \\
\hline
0 & 0 & 0 & 0 & r_3 & e & 0 & 0 & 0 & 0 & e & e \\
0 & 0 & 0 & 0 & r_2 & e & 0 & 0 & 0 & 0 & 0 & e \\
0 & 0 & 0 & 0 & r_1 & e & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & e & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

FIG. 2.3. *Left rotations to repartition R_{11}.*

fix the damage to F_{34} by using a left rotation. We allow rotations 3 and 4 to fill the first column of R_{14} with potentially large elements, eventually moving down to add a potentially large column to the beginning of E_{24}, which results in the pattern shown in Figure 2.3.

At this point, it is necessary to repartition the matrix to prepare for a deflation. The reason for the peculiar reversed triangular structure of R_{14} becomes clear. This figure shows a sequence of elements to be eliminated by left rotations. These rotations will add extra elements into the subdiagonals of R_{23}, F_{34}, and F_{45} to give a matrix that has essentially the same form as the original decomposition. This matrix suggests a natural repartitioning. We expand the size of the square blocks R_{11} and R_{23} by one. The rest of the matrix can be repartitioned along these lines, except that an extra column is added onto the right of R_{14}. The new partitioning is marked in Figure 2.3 and in Figure 2.4. In Figure 2.4 it is clear that the general form of the decomposition has been restored.

The matrix has now been repartitioned so that it has its original form, but it is possible that some of the R blocks will not have full rank. To finish the update we need a general procedure to take a matrix of the correct form, (1.4), and determine

$$\begin{array}{ccccc|c||ccc|cc|c}
r&r&r&r&r&e&r&r&r&r&r&e\\
0&r&r&r&r&e&r&r&r&r&e&e\\
0&0&r&r&r&e&r&r&r&e&e&e\\
0&0&0&r&r&e&r&r&r&e&e&e\\
0&0&0&0&r&e&r&r&r&e&e&e\\ \hline
0&0&0&0&0&e&r&r&r&e&e&e\\
0&0&0&0&0&e&0&r&r&e&e&e\\
0&0&0&0&0&e&0&0&r&e&e&e\\ \hline
0&0&0&0&0&e&0&0&0&e&e&e\\
0&0&0&0&0&e&0&0&0&0&e&e\\ \hline
0&0&0&0&0&e&0&0&0&0&0&e\\ \hline
0&0&0&0&0&e&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0
\end{array}$$

FIG. 2.4. *The repartitioned matrix with the correct form restored.*

an appropriate size for all triangular matrices. The procedure is the deflation method introduced in [11], applied to R_{11}, R_{23}, and R_{14}, along with additional rotation to fix any damage done to the structure by the deflations.

An implementation of this algorithm in MATLAB code is given in the appendix. This implementation deals with special cases ($r_i = 0$ for $i = 1, 2, 3$ and/or A or B has full rank) which were glossed over in the description of the algorithm. For the most part, these special cases involve omitting only the unnecessary steps. To avoid producing an overwhelming quantity of code, the deflation procedures described in the next section are hidden in function calls. The implementation of these deflations is fairly straightforward, given an understanding of the basic methods of the updating algorithm. We will not present codes for these functions. Finally, we note that the algorithm can be made more efficient by comparing components of the new rows with the tolerance and avoiding certain steps (and the deflations) when the new rows cannot change the rank of A or B.

3. The deflation process.

The deflation process for each block proceeds by finding a small singular value associated with the block, using knowledge of an associated singular vector to apply transformations forcing the rightmost column to have elements only of the order of this singular value, and continuing the process recursively on a smaller triangular matrix. The overall process proceeds by recursively deflating R_{11} until an appropriate rank is found, then deflating R_{23} in a similar manner, and then finally deflating R_{14}. We will describe the deflations in this order. The basic idea behind the procedure in [11] is to find a vector $\|w\|_2 = 1$ such that

$$\|R_{11}w\| \approx \sigma_{\min}(R_{11}),$$

where $\sigma_{\min}(R_{11})$ denotes the smallest singular value of R_{11}. The literature on condition estimation contains reliable methods which find such a w with $O(r_1^2)$ complexity; the precise method is not particularly important for our explanation of the procedure.

If $\|R_{11}w\|$ is small enough to be considered a null vector within the tolerance, then R_{11} is nearly rank deficient and must be deflated. A sequence of rotations is constructed, zeroing the elements of w in order until the last component is reached. While at the same time applying left rotations, also to R_{11} and in the manner described in [11] to preserve triangularity, we obtain

$$R_{11} = \begin{bmatrix} r & r & r & r & e \\ 0 & r & r & r & e \\ 0 & 0 & r & r & e \\ 0 & 0 & 0 & r & e \\ 0 & 0 & 0 & 0 & e \end{bmatrix}.$$

Then pattern of e elements holds since

$$\sigma_{\min}(R) \approx \|Rw\| = \|\hat{U}^T R \hat{V} \hat{V}^T w\| = \|\hat{U}^T R \hat{V} e_n\|,$$

while $\hat{U}^T R \hat{V} e_n$ is the last column of the new R which results from this deflation procedure. As described in relation to Figure 2.2, while it is convenient we fix the effects of the left rotation on R_{14}, but after a certain point, we let the first column of R_{14} fill with large elements. The result of this is shown in Figure 3.1. The matrix has to be repartitioned to make R_{11} smaller. This can be done by eliminating the elements marked in Figure 3.1 with left rotations which act on the numbered row and the row just before it. The result is Figure 3.2.

The effect of the deflating R_{11} on the sizes of the other R blocks is simple to see. Each time the size of R_{11} decreases, the size of R_{23} potentially increases and the size of R_{14} potentially decreases.

The deflation of R_{23} is performed next and the deflation of R_{14} last. The effects of the deflation on R_{23} are very simple to deal with: none of the rotations damage the overall structure of the decomposition, so all that is needed is the standard deflation procedure from [11], resulting in Figure 3.3. A sequence of left rotations needed to produce zeros in the last column of S_{13} so that it can become the first column of R_{14} is shown, together with right rotations needed to fix the effect of these on R_{11}. Thus, each deflation results in a decrease in the size of R_{23} and an increase in the size of R_{14}. If the ranks are to be restored to their original values, then it will be necessary to carry out two deflations on R_{23}. The result of these two deflations, with the corresponding repartitioning for R_{23} and R_{14}, is shown in Figure 3.4. The assumption that the ranks return to their original values is not essential to the algorithm, and it is adopted here only for convenience on the grounds that the algorithm will typically operate in steady state. The method of deflation is general and applies even without this assumption.

The deflation process for R_{14} is similar, although it is worth taking note of minor differences imposed by the odd structure of the block. First, the methods for finding w usually involve a back substitution. Here we ignore the small nonzero elements in attempting to find

$$\|R_{14} w\| \approx \sigma_{\min}(R_{14}).$$

Assuming that we have such a w, we apply rotations to introduce zeros into all but the last element and apply these rotations, together with rotations to fix the permuted triangular structure of R_{14}. Further rotations will be needed to fix R_{11} and F_{34}. The result of single deflation will be

$$R_{14} = \begin{bmatrix} r & r & e \\ r & r & e \\ r & e & e \\ e & e & e \end{bmatrix}.$$

$$
\begin{array}{ccccc|c||ccc|cc|c}
r & r & r & r & e & e & r & r & r & r & r & e \\
0 & r & r & r & e & e & r & r & r & r & e & e \\
0 & 0 & r & r & e & e & r & r & r & r & e & e \\
0 & 0 & 0 & r & e & e & r & r & r & r & e & e \\
0 & 0 & 0 & 0 & e & e & r & r & r & r & e & e \\
\hline
0 & 0 & 0 & 0 & 0 & e & r_1 & r & r & r & e & e \\
0 & 0 & 0 & 0 & 0 & e & 0 & r_2 & r & r & e & e \\
0 & 0 & 0 & 0 & 0 & e & 0 & 0 & r_3 & r & e & e \\
\hline
0 & 0 & 0 & 0 & 0 & e & 0 & 0 & 0 & r_4 & e & e \\
0 & 0 & 0 & 0 & 0 & e & 0 & 0 & 0 & 0 & e_5 & e \\
\hline
0 & 0 & 0 & 0 & 0 & e & 0 & 0 & 0 & 0 & 0 & e_6 \\
\hline
0 & 0 & 0 & 0 & 0 & e & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

FIG. 3.1. *After a deflation of R_{11}.*

$$
\begin{array}{cccc|cc||cccc|c|c}
r & r & r & r & e & e & r & r & r & r & r & e \\
0 & r & r & r & e & e & r & r & r & r & e & e \\
0 & 0 & r & r & e & e & r & r & r & r & e & e \\
0 & 0 & 0 & r & e & e & r & r & r & r & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & r & r & r & r & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & r & r & r & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & 0 & r & r & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & r & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & 0 & e \\
\hline
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & e & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

FIG. 3.2. *After deflation of R_{11} and repartitioning.*

$$
\begin{array}{cccc|cc||cccc|c|c}
r & r & r & r & e & e & r & r & r & r & r & e \\
0 & r & r & r & e & e & r & r & r & r & e & e \\
0 & 0_2 & r & r & e & e & r & r & r & r_2 & e & e \\
0 & 0 & 0_1 & r & e & e & r & r & r & r_1 & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & r & r & r & e & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & r & r & e & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & 0 & r & e & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & e & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & 0 & e \\
\hline
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & e & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

FIG. 3.3. *After a deflation of R_{23}.*

$$\begin{array}{cccc|cc||cc|ccc|c}
r & r & r & r & e & e & r & r & r & r & r & e \\
0 & r & r & r & e & e & r & r & r & r & e & e \\
0 & 0 & r & r & e & e & r & r & r & e & e & e \\
0 & 0 & 0 & r & e & e & r & r & e & e & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & r & r & e & e & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & r & e & e & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & 0 & 0 & e & e & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & e & e & e \\
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & e & e \\
\hline
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & 0 & e \\
\hline
0 & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & e & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

FIG. 3.4. *After two deflations of R_{23}.*

A sequence of left rotations will easily transform this to

$$R_{14} = \begin{bmatrix} r & r & e \\ r & e & e \\ e & e & e \\ e & e & e \end{bmatrix},$$

and the damage that the left rotations do to R_{11} can easily be fixed by right rotations. This completes the deflation of all of the triangular blocks.

Since many of the basic principles were illustrated by appealing to an example, it is worth noting that the deflation process for each block is quite general and does not depend on the sizes of the blocks. The deflation might have to be done for each block several times, but because each deflation returns the matrix to its correct form, they can be performed recursively for each block until the proper ranks are determined. As explained above, the deflation process is first applied recursively to R_{11} until its rank has been determined, and then we do the same for R_{23} and finally R_{14}.

There is one difficulty that has not yet been mentioned. The code given in the appendix assumes that prior to the update $r_1 \geq r_3$. This is generally not a problem, but if the size of R_{23} drops sufficiently after several deflations, then r_3 will temporarily increase by a corresponding amount before R_{14} is deflated. After the first such deflation on R_{23} for which this is a problem, we will have R_{14} of the form

$$R_{14} = \begin{bmatrix} r & r & r & r \\ r & r & r & e \\ r & r & e & e \end{bmatrix}.$$

Right rotations should be applied to compress this into the first 3 columns. The effect on F_{34} may be fixed with left rotations. This will keep $r_3 = r_1$ while the appropriate size of R_{23} is being determined.

The algorithm in this paper is essentially an extension of the URV algorithm as described in [11] and it inherits the simplicity with which the URV decomposition can deal with updates that do not increase any of the ranks. If after applying V_1 and V_2 to the new row vectors it is apparent that none of the ranks increase, the generalized URV updating can proceed in a manner similar to the simplified URV algorithm by skipping the rotations shown in Figures 2.1 and 2.3 and by skipping the deflation of

R_{11} and R_{14}. Because of the way the rotations in Figure 2.2 change the partitioning of R_{23} and R_{14} it will still be necessary to do a deflation on R_{23}. This is all in contrast to the ULV algorithm in which substantial additional computations are needed to avoid a deflation.

In the sample code in the appendix, we have hidden the deflations in functions which are not presented in this paper. Since the deflations involve fewer special cases and parallel the method of [11] more closely than the update, we have left them out.

4. Complexity. The decomposition has a fairly involved structure and, from the description given here, it might be assumed that the algorithm is computationally intensive. In fact, considering the difficulty of the problem, this is not the case; the computational complexity is surprisingly reasonable. The exact numbers will depend on r_1, r_2, and r_3 in addition to the number of columns in A and B. To simplify matters for comparison, we assume ranks which are reasonable in the context of the identification algorithm of [7]. In particular, we assume that A and B each have $2i$ columns and that $r_1 = i + n$, $r_2 = i$, and $r_3 = n$. We assume that i is slightly larger than n. If left rotations are not accumulated to form U (knowledge of U is not required by the updating algorithm), then the complexity involved in updating this decomposition when none of the ranks change is at worst $325i^2 + 120in + 6n^2$ flops. If it is apparent in the first stage of the update that the new rows do not increase the ranks of the R blocks, then the update can be computed with much lower complexity.

These numbers look very bad, but when the level of difficulty inherent in the problem is taken into account, they are quite reasonable. If i is only slightly larger than n, so that the difference can be absorbed into lower-order terms, then the complexity is $451(4i)^2/16$. Thus, since the decomposition involves a matrix combining both A and B with a total of $4i$ columns, the worst-case complexity involved is really expressed more reasonable as approximately $28(4i)^2$ flops.

This is certainly large when compared with the updating of a QR decomposition of a matrix of the same size. The QR decomposition involves only $3(4i)^2$ flops, but an ordinary URV decomposition is a different matter. Just to compute a URV decomposition of A involves $71i^2 + 6in + 3n^2$ flops. Again, assuming i and n are close and ignoring lower-order terms, the complexity is roughly $5(4i)^2$. Thus updating the quotient URV is about a factor of three more costly than computing URV decompositions for A and B separately. Depending on the ranks involved, it is often not much more costly than updating the URV decomposition of a single matrix with $4i$ columns.

The generalized URV decomposition is similar in spirit to the generalized QR factorization of [9]. However, a generalized QR factorization does not lend itself to updating. In terms of computational complexity, the use of the URV updating method is justifiable only when updates are needed. The method is not competitive for finding the subspaces associated with the generalized SVD of a single matrix.

5. An application to system identification. Consider the state space model

$$x_{k+1} = A_k x_k + B_k u_k, \qquad (5.1)$$

$$y_k = C_k x_k + D_k u_k,$$

where x_k is $n \times 1, u_k$ is $m \times 1$, and y_k is $p \times 1$.

Assuming we have observations of the input and output vectors, u_k and y_k, the identification problem is to find an order, n, and time-varying system matrices $\{A_k, B_k, C_k, D_k\}$ that satisfy, or approximately satisfy, (5.1) for some $n \times 1$ state sequence x_k.

If the output vectors are generated by a time-invariant model $\{A, B, C, D\}$ and observations are corrupted by noise, we want the estimated model to converge to $\{A, B, C, D\}$ or to some model $\{SAS^{-1}, SB, CS^{-1}, D\}$ given by a change of basis for x_k and having identical input/output behavior.

More realistically, it is often assumed that the state space model is slowly time-varying and that there is small noise on the observed input and output vectors. Under those circumstances, we wish to provide an algorithm to track variations in the model.

The generalized URV decomposition applies naturally to a system identification algorithm developed in [7]. The approach can be characterized by two steps: find an estimate of the state sequence x_k, and then obtain the system matrices from the least squares problem

$$\begin{aligned} &\begin{bmatrix} x_{k+i+j-1} & \cdots & x_{k+i+1} \\ y_{k+i+j-2} & \cdots & y_{k+i} \end{bmatrix} W_{j-1} \\ &\quad = \begin{bmatrix} A_j & B_j \\ C_j & D_j \end{bmatrix} \begin{bmatrix} x_{k+i+j-2} & \cdots & x_{k+i} \\ u_{k+i+j-2} & \cdots & u_{k+i} \end{bmatrix} W_{j-1}, \end{aligned} \tag{5.2}$$

where W_j is a diagonal weighting matrix defined by

$$W_j = \begin{bmatrix} 1 & 0 \\ 0 & \alpha_j W_{j-1} \end{bmatrix}$$

for $|\alpha_j| < 1$ and $W_1 = 1$. The index k is the time at which observations begin and $k + i + j - 1$ is the time at which the latest observations have been made. Indices k and i are fixed, but j grows as more observations are made. To keep the notation compact, the indexing of the system matrices will show only the dependence on j, although $\{A_j, B_j, C_j, D_j\}$ will depend on observation up to $u_{k+i+j-1}$ and $y_{k+i+j-1}$.

Define the $mi \times j$ block Toeplitz matrices

$$U_K = \begin{bmatrix} u_{k+j-1} & u_{k+j-2} & \cdots & u_k \\ u_{k+j} & u_{k+j-1} & \cdots & u_{k+1} \\ \vdots & \vdots & & \vdots \\ u_{k+j+i-2} & u_{k+j+i-3} & \cdots & u_{k+i-1} \end{bmatrix},$$

$$Y_k = \begin{bmatrix} y_{k+j-1} & y_{k+j-2} & \cdots & y_k \\ y_{k+j} & y_{k+j-1} & \cdots & y_{k+1} \\ \vdots & \vdots & & \vdots \\ y_{k+j+i-2} & y_{k+j+i-3} & \cdots & y_{k+i-1} \end{bmatrix},$$

and

$$T_k = \begin{bmatrix} U_k \\ Y_k \end{bmatrix}.$$

The following theorem from [7] provides a means for generating an appropriate sequence of state vectors.

THEOREM 5.1. *Let the vectors u_k and y_k be generated by*

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k, \\ y_k &= Cx_k + Du_k, \end{aligned}$$

where the rank of

$$\begin{bmatrix} C^T & A^TC^T & \cdots & (A^T)^{n-1}C^T \end{bmatrix} \tag{5.3}$$

is n.

Let

$$X_k = \begin{bmatrix} x_{k+j-1} & x_{k+j-2} & \cdots & x_k \end{bmatrix}$$

and

$$X_{k+i} = \begin{bmatrix} x_{k+i+j-1} & x_{k+i+j-2} & \cdots & x_{k+i} \end{bmatrix}.$$

For $i \geq n$, if rank(X_k)= rank(X_{k+i}) $= n$ and the matrices

$$\begin{bmatrix} U_k \\ X_k \end{bmatrix}, \qquad \begin{bmatrix} U_{k+i} \\ X_{k+i} \end{bmatrix}, \qquad \begin{bmatrix} U_k \\ U_{k+i} \\ X_k \end{bmatrix} \tag{5.4}$$

all have full rank $mi + n$, $mi + n$, and $2mi + n$, respectively, then T_k and T_{k+i} both have rank $mi + n$ and the intersection of the span of the rows of T_k and T_{k+i} has dimension n. Further, there is a basis, X, of the intersection for which

$$X = \begin{bmatrix} x_{k+i+j-1} & x_{k+i+j-2} & \cdots & x_{k+i} \end{bmatrix},$$

and different bases for this space correspond to state vector sequences of models with equivalent input/output behavior under a transformation of the form

$$\{SAS^{-1}, SB, CS^{-1}, D\}.$$

The rank condition on (5.3) implies the observability of the linear system; without this assumption the full information contained in the state sequence will not be seen in the output and any identification scheme can be expected to fail. The condition on the rank of X_k and X_{k+i} implies that the input fully excites all modes of the system. This is also a standard and necessary assumption in system identification.

The rank assumption involving (5.4) is stronger: it clearly implies the rank condition on X_k and X_{k+i}. The key point is to make sure that U_k and U_{k+i} have full rank and to make sure that X_k is not contained in their span. A full rank condition on the inputs is standard in identification. The joint condition on X_k is less standard, but it can be verified that it is satisfied generically and the probability that it fails decreases when j is increased. More details about the implications of these assumptions together with a proof of the theorem may be found in [7].

If $\alpha \neq 1$, we look for the intersection of the span of the rows of T_kW_j and $T_{k+i}W_j$. In that case, the theorem shows that the intersection is the weighted state vector sequence required by (5.2).

The generalized URV algorithm can be used to update the intersection of the range spaces of T_k^T and T_{k+i}^T as new observations are made and new rows are added to the matrices. This leaves the solution of (5.2) to complete the identification process. It is possible to efficiently update the QR decomposition needed to solve (5.2) while updating the generalized URV decomposition. Further details are contained in [12].

In order to show the effectiveness of the decomposition, we consider the system defined by

$$A = \begin{bmatrix} .4 & 0 & .8 \\ .4 & .4 & -.4 \\ .4 & 0 & .4 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 2 \\ 3 & 1 \\ -4 & 2 \end{bmatrix},$$

$$C = \begin{bmatrix} 0 & -1 & 0 \\ 1 & -2 & -1 \end{bmatrix}, \qquad D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The system can be verified to be stable with its largest eigenvalue having magnitude strictly less than 1. The observability condition is also easily verified.

We generated a sequence of input vectors u_k with elements that were randomly generated according to zero mean normal distribution with variance 1. An initial state vector was chosen as $x_1 = 0$. A sequence of output vectors, y_k, was generated by (5.1).

Before applying the identification algorithm, each component of the input and output vectors was perturbed by zero mean unit variance normal noise scaled by .01, resulting in a noise component two orders of magnitude below the signal component. The tolerance for deflation of all three triangular blocks was set to an absolute value of .5. To give an idea of the relative significance of this tolerance, the data matrices satisfy

$$\left\| \begin{bmatrix} T_1^T & T_{i+1}^T \end{bmatrix} \right\| \approx 100$$

for $j = 50$. The value of this norm for $j = 20$ is approximately 50. We used $i = 3$ and data were taken for $j = 24, \ldots, 50$.

For each j the generalized URV decomposition correctly identified the order $n = 3$. When the identified model was driven by the inputs u_k starting with the earliest identified state vector x_{i+1}, the difference between the outputs produced by the identified model and the original model was of the same order of magnitude as the noise. We define

$$Y = \begin{bmatrix} y_{i+1} & y_{i+2} & \cdots & y_{100} \end{bmatrix}$$

as a matrix of original, unperturbed outputs and

$$\hat{Y}_j = \begin{bmatrix} \hat{y}_{i+1} & \hat{y}_{i+2} & \cdots & \hat{y}_{100} \end{bmatrix}$$

as the matrix of simulated outputs produced by the system identified using j columns of T_1 and T_{i+1}. The errors

$$\frac{\|Y - \hat{Y}_j\|_2}{\|Y\|_2}$$

are shown in Figure 5.1.

Clearly the algorithm is successful in handling this level of noise. However, if we increase the noise level by a factor of 10, the algorithm fails dramatically; it is not possible to find a tolerance for which the ranks r_2 and r_3 are estimated reliably. Nevertheless the rank r_1 and the sum $r_2 + r_3 = r_1$ are both estimated reliably for a choice of absolute tolerance of 1.

The problem is the inherent sensitivity of the generalized SVD computation as characterized by a simple perturbation analysis in section 1. For rank estimation, the