An Updating Algorithm for On-line MIMO System Identification

Michael Stewart
Coordinated Science Laboratory
University of Illinois
Urbana, IL USA
stewart@monk.csl.uiuc.edu

Paul Van Dooren
CESAME
Université Catholique de Louvain
Louvain-la-Neuve
Belgium
vandooren@anma.ucl.ac.be

ABSTRACT. This paper describes the application of a generalized URV decomposition to an on-line system identification algorithm. The algorithm updates estimates of a state space model with $O(n^2)$ complexity.

KEYWORDS. Identification, MIMO Systems, Updating, URV.

Introduction

Identification of a state space model for a MIMO system from input/output data is a computationally intensive problem. A reliable algorithm was given in [1], but as presented it depends on the SVD to make crucial rank decisions and identify subspaces. Unfortunately, there have been no exact algorithms proposed for updating the SVD when input/output measurements are added which are faster than $O(n^3)$. An approximate approach to SVD updating which might be considered for use here was developed in [2]. However the problem is really more difficult than just updating one SVD. What is desired is the intersection of the range spaces of two matrices. In [1] this is computed using two SVD's, and they both must be updated simultaneously.

The URV decomposition, [4], is an easily updated decomposition which, in some applications, may be used to replace the SVD. The fact that an intersection of the range spaces

of two matrices is required suggests that a generalization of the URV decomposition along the lines of [3] might be helpful. Such a decomposition was introduced in [5] along with an $O(n^2)$ updating algorithm. This paper will give a brief description of the decomposition and show how it can be used as part of an on-line identification algorithm.

Given a sequence of $m \times 1$ input vectors, $u(k)$, we assume that the sequence of $l \times 1$ output vectors, $y(k)$, are generated from the state space equations,

$$\begin{aligned} x(k+1) &= A_k x(k) + B_k u(k) \\ y(k) &= C_k x(k) + D_k u(k). \end{aligned} \tag{1}$$

Assuming we have observations of the input and output vectors, the identification problem is to find an order, n, and time-varying matrices, $\{A_k, B_k, C_k, D_k\}$, which satisfy (1) for some $n \times 1$ state sequence, $x(k)$. Generally it is assumed that the state space model is slowly time-varying. We then wish to provide an algorithm which will track the model.

The algorithm uses the same basic approach developed in [1]. It can be summarized in two steps: find an estimate of the state sequence and then obtain the system matrices from the least squares problem

$$\begin{aligned} \begin{bmatrix} x(k+i+j-1) & \cdots & x(k+i+1) \\ y(k+i+j-2) & \cdots & y(k+i) \end{bmatrix} W_{j-1} = \\ \begin{bmatrix} A_j & B_j \\ C_j & D_j \end{bmatrix} \begin{bmatrix} x(k+i+j-2) & \cdots & x(k+i) \\ u(k+i+j-2) & \cdots & u(k+i) \end{bmatrix} W_{j-1}, \end{aligned} \tag{2}$$

where W_j is a diagonal weighting matrix defined by

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

for $|\alpha| < 1$ 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 observations are made. To keep the notation compact, the indexing of the system matrices will show only the dependence on j, though $\{A_j, B_j, C_j, D_j\}$ will depend on observations up to $u(k+i+j-1)$ and $y(k+i+j-1)$

An appropriate exponentially weighted state vector sequence can be determined from the intersection of the row space of two Toeplitz matrices. Define the $(m+l)i \times j$ block Toeplitz matrix

$$T(k) = \begin{bmatrix} u(k+j-1) & u(k+j-2) & \cdots & u(k) \\ y(k+j-1) & y(k+j-2) & \cdots & y(k) \\ \vdots & \vdots & & \vdots \\ u(k+j+i-2) & u(k+j+i-3) & \cdots & u(k+i-1) \\ y(k+j+i-2) & y(k+j+i-3) & \cdots & y(k+i-1) \end{bmatrix}.$$

If $T_1 = T(k)$ and $T_2 = T(k+i)$ then in the time invariant case, the intersection of the row spaces of T_1 and T_2 generically has dimension n, the order of the model (1) generating $y(k)$ from $u(k)$. [1]

If the rows of some X form a basis for the intersection then the columns of X are a

sequence of state vectors for a time invariant model generating $y(k)$ from $u(k)$. A proof of this fact can be found in [1].

If we compute the intersection of the row spaces of $T_1 W_j$ and $T_2 W_j$ and let X denote the basis for this space, then we use X as the exponentially weighted state vector sequence,

$$X = \left[\begin{array}{cccc} x(k+i+j-1) & x(k+i+j-2) & \cdots & x(k+i) \end{array} \right] W_j. \tag{3}$$

The decomposition of [5] can be used to track the intersection of the row spaces. The contribution of this paper is to show how the system matrices can be obtained efficiently at the same time.

1 The Decomposition

This section will deal with the T matrices in transposed form so that the problem becomes one of tracking column spaces as rows are added.

The decomposition has the form

$$U^T \left[\begin{array}{c|c} W_j T_1^T & W_j T_2^T \end{array} \right] \left[\begin{array}{cc} V_1 & 0 \\ 0 & V_2 \end{array} \right] = \left[\begin{array}{cc|ccc} R_{11} & E_{12} & S_{13} & R_{14} & E_{15} \\ 0 & E_{22} & R_{23} & E_{24} & E_{25} \\ 0 & E_{32} & 0 & F_{34} & E_{35} \\ 0 & E_{42} & 0 & 0 & F_{45} \\ 0 & F_{52} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right]. \tag{4}$$

where R_{11}, R_{23} and R_{14} are upper triangular and full rank. R_{11} and R_{23} are square.

Each F block is an upper triangular matrix with norm less than the tolerance. Each E block is an arbitrary matrix with norm less than the tolerance. The S block is an arbitrary matrix. If this is the case, then the decomposition gives estimates of the range spaces of $W_j T_1^T$ and $W_j T_2^T$. In fact, it can be shown that if the E and F blocks are zero, then the first columns, U_1, of U corresponding to the number of columns in R_{14} are a basis for the intersection of the range space of $W_j T_1^T$ and $W_j T_2^T$. Details concerning decompositions of this type can be found in [3].

If we partition V_2 in a manner which matches the decomposition

$$V_2 = \left[\begin{array}{ccc} V_{23} & V_{24} & V_{25} \end{array} \right]$$

and assume that the E and F blocks are zero then

$$U_1 R_{14} = W_j T_2^T V_{24}$$

and the full rank property of R_{14} imply that $W_j T_2^T V_{24}$ is also a basis for the intersection. This fact makes it possible to avoid storing U and will avoid the problem of growing memory storage as rows are added to $W_j T_1^T$ and $W_j T_2^T$.

Details on updating the decomposition can be found in [5]. A brief summary of relevant features will be given here. We assume that the decomposition has already been computed and we are interested in having the decomposition for the same matrices, but with an added row. The process can be initialized by setting U and V to the identity and letting

the decomposition equal zero.

If two rows, a^T and b^T are added to $W_j T_1^T$ and $W_j T_2^T$ respectively and each row of the old matrix is weighted by $0 < \alpha < 1$ then we wish to restore the form of (4) to

$$\left[\begin{array}{cc} 1 & 0 \\ 0 & U^T \end{array} \right] \left[\begin{array}{c|c} a^T & b^T \\ \alpha W_j T_1^T & \alpha W_j T_2^T \end{array} \right] \left[\begin{array}{cc} V_1 & 0 \\ 0 & V_2 \end{array} \right]. \tag{5}$$

This looks much like (4), but with an additional row along the top. 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. The key feature of the algorithm as presented in [5] which has a bearing on the system identification algorithm is that the structure of (5) can be restored by applying plane rotations from the left which operate on adjacent rows and plane rotations from the right which operate on adjacent columns.

The approach is similar to that of URV updating as given in [4]. The algorithm can be broken into two stages. The first updates the overall structure of the decomposition when new rows are added to $W_j T_1^T$ and $W_j T_2^T$. After the update, the decomposition has the same general form, but the triangular R matrices are possibly larger 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 [4] until they have full rank.

2 Updating the System Matrices

As mentioned earlier, the intersection of the range spaces of $W_j T_1^T$ and $W_j T_2^T$ is given by $W_j T_2^T V_{24}$ which also gives an estimate of the exponentially weighted state vectors, (3).

Thus the least squares problem can be written as

$$\begin{aligned} \left(\tfrac{1}{\alpha} \left[\begin{array}{cc} W_j T_2^T V_{24} & W_j U(j) \end{array} \right] \right) (2:j) \left[\begin{array}{cc} A_j^T & C_j^T \\ B_j^T & D_j^T \end{array} \right] = \\ \left(\left[\begin{array}{cc} W_j T_2^T V_{24} & W_{j-1} Y(j-1) \end{array} \right] \right) (1:j-1) \end{aligned} \tag{6}$$

where

$$U(j) = \left[\begin{array}{cccc} u(k+i+j-1) & u(k+i+j-2) & \cdots & u(k+i) \end{array} \right]^T$$

and

$$Y(j) = \left[\begin{array}{cccc} y(k+i+j-2) & y(k+i+j-3) & \cdots & y(k+i-1) \end{array} \right]^T.$$

We will give an updating scheme for the QR decomposition associated with the least squares problem which can carried out in conjunction with the decomposition updating to provide a solution to the system identification problem. It would be nice if the updating could be performed in $O(n^2)$, and in a sense this is possible. Unfortunately there is a problem: The system matrices will be updated through several intermediate stages during the process of updating the decomposition of $W_j T_1^T$ and $W_j T_2^T$. If at one of these stages R_{14} is ill conditioned, as would be expected prior to a deflation, then the least squares problem will also be ill conditioned. The connection between the conditioning of R_{14} and that of the least squares problem is obvious since R_{14} is, neglecting the small elements, the $n \times n$

principle submatrix of the QR decomposition for (6). This temporary ill conditioning can introduce large errors in the system matrices as updating is carried out.

Thus there are two possible approaches to updating the system matrices presented in this paper. The first is the numerically safe approach of updating the QR decomposition for (6) and then do a back substitution to get $\{A_j, B_j, C_j, D_j\}$. This avoids large errors due to a temporarily ill conditioned R_{14}, but it is, unfortunately, an $O(n^3)$ process. The other possibility is an $O(n^2)$ algorithm which updates $\{A_j, B_j, C_j, D_j\}$ as the generalized URV decomposition is updated.

Both approaches require the QR factorization of

$$\left(\frac{1}{\alpha}\left[\begin{array}{cc} W_j T_2^T V_{24} & W_j U(j) \end{array}\right]\right)(2:j)$$

so that will be dealt with first. As the generalized URV decomposition is updated, the size of $W_j T_2^T V_{24}$ can change due to changes in the size of R_{14} during the updating. This can be dealt with by computing the QR decomposition of the expanded matrix,

$$P = \left(\frac{1}{\alpha}\left[\begin{array}{cccc} W_j T_2^T V_{24} & W_j U(j) & W_j T_2^T V_{23} & W_j T_2^T V_{25} \end{array}\right]\right)(2:j).$$

We then obtain the required R factor as the $(n+m)\times(n+m)$ principal submatrix of the expanded R.

Suppose a row is added to $W_j T_1^T$ and $W_j T_2^T$. This corresponds to a row being added to P,

$$\left[\begin{array}{cc} 1 & 0 \\ 0 & Q_1^T \\ 0 & Q_2^T \end{array}\right]\left[\begin{array}{c} p^T \\ \alpha P \end{array}\right] = \left[\begin{array}{c} p^T \\ \alpha R \\ 0 \end{array}\right], \tag{7}$$

where

$$Q = \left[\begin{array}{cc} Q_1 & Q_2 \end{array}\right]$$

is a square orthogonal matrix and Q_1 has the same number of columns as P. If P has full column rank, then Q_1 is an orthogonal basis for the range space of P. Otherwise, range of P is contained in the range of Q_1.

To deal with the right hand side, we define

$$S = \left(\left[\begin{array}{cccc} W_j T_2^T V_{24} & W_j Y(j) & W_j T_2^T V_{23} & W_j T_2^T V_{25} \end{array}\right]\right)(1:j-1),$$

and keep track of the matrix $Q_1^T S$. When a row is added we get

$$\left[\begin{array}{cc} 1 & 0 \\ 0 & Q_1^T \end{array}\right]\left[\begin{array}{c} s^T \\ \alpha S \end{array}\right] = \left[\begin{array}{c} s^T \\ \alpha Q_1^T S \end{array}\right]. \tag{8}$$

Before any updating is done on the generalized URV decomposition, we can apply a standard QR updating to (7). The rotations which accomplish this are applied to (8) at the same time. Since it is not necessary to store Q, the memory required by this approach does not grow with time.

Once the QR decomposition has been updated, the generalized URV updating can be

performed. Each time a right rotation is performed and V_2 is updated, a corresponding right rotation is performed on P and S. The rotation performed on P destroys the QR decomposition of P. Since all of the right rotations which are used to update the generalized URV decomposition operate on adjacent columns, there are clearly three ways in which the QR decomposition can be damaged. The simplest is when the update to V_2 only affects one of the matrices V_{23}, V_{24}, or V_{25}. In this case the rotation operates on two adjacent columns of P and hence merely creates a single nonzero element on the subdiagonal of the R factor of P. To zero this element requires just one left rotation which is applied to both P and S.

The other possibilities are when the update to V_2 affects the last column of V_{24} and the first column of V_{25} or the first column of V_{24} and the last column of V_{23}. Since they do not correspond to adjacent columns of P, they create more nonzeros than the first case. To restore the QR decomposition after one such right rotation is an $O(n^2)$ process. Fortunately, the number of these rotations is bounded independently of n, so that the overall process is still $O(n^2)$. It can easily be shown that it is possible to deal with changes in the block sizes of P and S due to changes in the sizes of R_{14} and R_{23} in $O(n^2)$ by using similar techniques.

Once the URV updating has been completed and the QR decomposition of P has been maintained, we have the R factor for the least squares problem in the form of the $(n+m) \times (n+m)$ principal submatrix of the R factor of P. Similarly if we take as the right hand side the $(n+m) \times (n+l)$ principal submatrix of $Q_1^T S$, then we can do a triangular backsolve to find the system matrices.

There are three sorts of updates which must be performed on the least squares solution. The first is to deal with a new row which is added to the problem. If we look at the submatrix of P,

$$P_1 = \left(\frac{1}{\alpha} \left[\begin{array}{cc} W_j T_2^T V_{24} & W_j U(j) \end{array} \right] \right) (2:j)$$

and the submatrix of S

$$S_1 = \left(\left[\begin{array}{cc} W_{j-1} T_2^T V_{24} & W_j Y(j) \end{array} \right] \right) (1:j-1)$$

that define the least squares problem

$$P_1 \left[\begin{array}{cc} A_j^T & C_j^T \\ B_j^T & D_j^T \end{array} \right] = S_1,$$

then when a row is added, we would like to find the solution to the least squares problem

$$\left[\begin{array}{c} p^T \\ \alpha P_1 \end{array} \right] \left[\begin{array}{cc} A_j^T & C_j^T \\ B_j^T & D_j^T \end{array} \right] = \left[\begin{array}{c} s^T \\ \alpha S_1 \end{array} \right].$$

The normal equations are

$$(pp^T + \alpha^2 P_1^T P_1) \left[\begin{array}{cc} A_j^T & C_j^T \\ B_j^T & D_j^T \end{array} \right] = (ps^T + \alpha^2 P_1^T S_1).$$

Using the Sherman-Morrison-Woodbury formula, the solution can be written as

$$\left[\begin{array}{cc} A_j^T & C_j^T \\ B_j^T & D_j^T \end{array} \right] = (P_1^T P_1)^{-1} P_1^T S_1 + \frac{x}{\alpha^2} \left(\frac{p^T x}{\alpha + p^T x} s^T - x^T P_1^T S_1 \right).$$

where $x = (P_1^T P_1)^{-1} p$. The new least squares solution is just a rank one modification of the old solution. This modification can be computed in $O(n^2)$ flops using the R factor for P_1, the updating of which was described earlier.

Once the new row has been incorporated into the least squares solution, the QR decomposition for P with the new row added can be computed as described earlier. When that has been done, the generalized URV updating can commence with the QR decomposition being updated as V_2 is changed. The final part of the identification problem is to update $\{A_j, B_j, C_j, D_j\}$ as V_2 is changed and the partitioning is changed.

Rotations which only affect V_{23} and V_{25} will not affect the least squares solution. The simplest case in which something actually has to be done is one in which the rotation affects only V_{24}. Suppose some rotation V is applied to the state estimate portions of P_1 and S_1. Then the normal equations become

$$\begin{bmatrix} V & 0 \\ 0 & I_m \end{bmatrix}^T P_1^T P_1 \begin{bmatrix} V & 0 \\ 0 & I_m \end{bmatrix} \begin{bmatrix} A_j^T & C_j^T \\ B_j^T & D_j^T \end{bmatrix} = \begin{bmatrix} V & 0 \\ 0 & I_m \end{bmatrix}^T P_1^T S_1 \begin{bmatrix} V & 0 \\ 0 & I_l \end{bmatrix}$$

so the new solution is

$$\begin{bmatrix} A_j^T & C_j^T \\ B_j^T & D_j^T \end{bmatrix} = \begin{bmatrix} V & 0 \\ 0 & I_m \end{bmatrix}^T (P_1^T P_1)^{-1} P_1^T S_1 \begin{bmatrix} V & 0 \\ 0 & I_l \end{bmatrix}.$$

The same rotation which is applied to P_1 and S_1 can be applied to the right and left of the old solution to get the new system matrices.

Because the right rotations involved in updating the generalized URV decomposition always act on adjacent columns, we need only make special consideration of the case in which a rotation acts at the boundaries of one of the blocks of V_2. It turns out that such rotations always occur when there is a change in the size of the R_{14} block. They can be dealt with by viewing the process as adding either the last column of V_{23} or the first column of V_{25} to V_{24} and then performing a rotation which acts purely within V_{24}.

All that we need to deal with are rotations which act solely within one of the blocks V_{23}, V_{24} or V_{25}, which has already been covered, and the process of adding or deleting a column from the least squares problem. Each time R_{14} grows, we must bring a column from the $W_j T_2^T V_{23}$ or $W_j T_2^T V_{25}$ into the $W_j T_2^T V_{24}$ block of P. Since the $W_j T_2^T V_{24}$ and the $U(j)$ blocks of P define, the least squares problem, this amounts to adding a column to the least squares problem. The same thing applies to S. Similarly, whenever a column is removed from R_{14}, the least squares problem shrinks by a column.

Suppose we have the QR decomposition of P_1 with the column p appended,

$$\begin{bmatrix} P_1 & p \end{bmatrix} = \begin{bmatrix} Q_1 & q_2 & Q_3 \end{bmatrix} \begin{bmatrix} R_{11} & r_{12} \\ 0 & r_{22} \\ 0 & 0 \end{bmatrix},$$

and we have a solution to the least squares problem

$$\begin{bmatrix} P_1 & p \end{bmatrix} \begin{bmatrix} X_{11} & x_{12} \\ x_{21}^T & x_{22} \end{bmatrix} = \begin{bmatrix} S_1 & s \end{bmatrix}.$$

Then the solution satisfies

$$\begin{bmatrix} R_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix} \begin{bmatrix} X_{11} & x_{12} \\ x_{21}^T & x_{22} \end{bmatrix} = \begin{bmatrix} Q_1^T S_1 & Q_1^T s \\ q_2^T S_1 & q_2^T s \end{bmatrix}.$$

¿From this we get four equations which can be used to update the solution

$$R_{11}(X_{11} + R_{11}^{-1} r_{12} x_{21}^T) = Q_1^T S_1$$

$$R_{11} x_{12} + r_{12} x_{22} = Q_1^T s$$

$$r_{22} x_{21}^T = q_2^T S_1$$

and

$$r_{22} x_{22} = q_2^T.$$

¿From the first of these equations, it is clear that if we wish to delete a column, we get a solution to the new least squares problem $P_1 X = S_1$ of

$$X = X_{11} + R_{11}^{-1} r_{12} x_{21}^T \tag{9}$$

This can easily be computed in $O(n^2)$.

The reverse process, that of going from a solution, X, of the smaller problem, to the solution of the larger problem makes use of all four of the equations. First x_{22} can be computed from the last equation, x_{21} from the third, x_{12} from the second, and X_{11} using (9). The necessary products $Q_1^T s$, $q_2^T S_1$ and $q_2^T s$ will be available from the part of the identification algorithm which updates the QR decomposition and the transformed right hand side. Again, the whole process can be carried out in $O(n^2)$ flops.

Acknowledgments

This research was supported by the National Science Foundation under Grant CCR 9209349

References

[1] M. Moonen, B. De Moor, L. Vandenberghe and J. Vandewalle. On- and Off-line Identification of Linear State-space Models. *Int. J. Control* **49**, pp 219-232, 1989.

[2] M. Moonen. *Jacobi-type Updating Algorithms for Signal Processing, Systems Identification and Control.* PhD Thesis, Katholieke Universiteit Leuven, 1990.

[3] C. C. Paige. Some Aspects of Generalized QR Factorizations. In : M. G. Cox and S. Hammarling (Eds.), *Reliable Numerical Computation*, Oxford Univ. Press, pp. 73-91, 1990.

[4] G. W. Stewart. An Updating Algorithm for Subspace Tracking. *IEEE Transactions on Signal Processing* **40**, pp. 1535-1541, 1992.

[5] M. Stewart and P. Van Dooren, A QURV Updating Algorithm. In : J. G. Lewis (Ed.), *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra*, SIAM, pp. 269-273, 1994.