ALGORITHMS AND LAPACK-BASED SOFTWARE FOR SUBSPACE IDENTIFICATION

Vasile Sima

Research Institute for Informatics
Bd. Mareşal Averescu, Nr. 8–10
71316 Bucharest 1, Romania
vsima@roearn.ici.ro

Abstract

Basic algorithms and LAPACK-based Fortran software for multivariable system identification by subspace techniques are briefly described. Deterministic and combined deterministic-stochastic identification problems are dealt with using two approaches. A state space model is computed from input-output data sequences. Multiple data sequences, collected by possibly independent identification experiments, can be handled. Sequential processing of large data sets is provided as an option. Illustrative numerical examples are included.

1. Introduction

Computer-aided control system design is usually based on linear time-invariant (LTI) state space models. Surprisingly enough, multivariable state space system identification has only recently become a topic of intense research. In contrast with classical input-output (I/O) identification approaches, the newly proposed Subspace Model Identification (SMI) techniques essentially find a state sequence, or a column space approximation, and then determine the system matrices by solving some least squares problems. These techniques have promising advantages over the classical ones. One advantage is that there is no need for parameterizations, which are notoriously difficult to choose or analyse, or could led to ill-conditioned problems. Another advantage is that robust numerical linear algebra techniques, like QR factorization and singular value decomposition (SVD), can be largely used; this contrasts with the iterative optimization schemes required in the parametric model identification approach, documented e.g. in [1]. The attractiveness of SMI techniques is further increased by the small number of parameters (essentially only one) to be selected for determining the model structure, without any restriction on model generality. See, for instance, [2]–[7] for a further discussion of the SMI features.

MATLAB codes based on SMI algorithms have been recently developed, e.g. in [8]. For efficiency, it is clearly useful to implement some algorithms in Fortran, using the state-of-the-art, public-domain linear algebra package LAPACK. This allows to exploit the potential parallelism of many modern computer architectures.

This paper briefly describes the basic algorithms and LAPACK-based Fortran routines for multivariable system identification by subspace techniques. The developed software package [9] includes implementations of two classes of SMI techniques to identify an LTI system. The first one is referred to as the State Intersection (SI) class of SMI techniques. The implemented method, described in [2], is known as the N4SID (Numerical algorithm for Subspace State Space System IDentification) approach. The second one is referred to as the Multivariable Output Error state SPace (MOESP) class of techniques. The implemented MOESP methods are described in [3], [5] and [6]. Deterministic and combined deterministic-stochastic identification problems are dealt with. A state space model is computed from I/O data sequences. It is possible to handle multiple I/O sequences, each one being collected by a possibly independent identification experiment. Sequential processing of large data sets is provided as an option. See [9], for further details. In addition to the main identification facilities, auxiliary routines are available to detect dead-times and shift the I/O signals appropriately, estimate initial states, identify systems operating in closed-loop, or simulate discrete-time LTI systems.

The theoretical algorithms and their MATLAB realizations have been largely reorganized. For instance, the N4SID algorithm makes use of the inverse of a part of the triangular factor of an RQ factorization of a Hankel-like matrix constructed from I/O sequences. (This is needed before computing the SVD that gives the system order and then the quadruple of system matrices.) The MATLAB implementations solve several standard least squares problems to obtain the associated, so-called "oblique projection". While the new, LAPACK-based implementation only involves some orthogonal transformations for obtaining the various projections and residuals.

One objective of the Fortran implementation has been to use the LAPACK routines as much as possible. Preference has generally been given to the block variants and the Level III BLAS codes, which are responsible for efficient use of parallelism. Advantage was taken of the problem structure, whenever possible. For instance, the calculations have been organized such that singular value decompositions were required for triangular matrices only, and a dedicated, very compact and efficient routine has been written for that purpose. Moreover, both in theory and in the MATLAB implementations, the computation of the system matrices involves a very large matrix. Fortunately, this matrix can be brought to a special block-triangular structure. A structure-exploiting QR row compression scheme has been devised, reducing the storage requirements and computational effort.

Another related objective has been to allow performing the computations with as reduced memory requirements as possible, but without sacrificing the speed, when enough memory is available. For instance, there is an option for sequential calculation of the triangular factor in QR factorization of the Hankel-like matrix. The user can specify that I/O data are entered in batches (independent or not). Moreover, when there is not enough working memory for processing a given data batch, the codes automatically perform an inner sequential processing of that batch, to accommodate with the available memory space.

2. Basic Approaches

The basic LTI state space models considered are described by

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k, \\ y_k &= Cx_k + Du_k + v_k, \end{aligned} \qquad (1)$$

where x_k is the n-dimensional state vector at time k, u_k is the m-dimensional input (control) vector, y_k is the ℓ-dimensional output vector, $\{w_k\}$ and $\{v_k\}$ are state and output disturbance or noise sequences, and A, B, C, and D are real matrices of appropriate dimensions. The system order, n, and the quadruple of system matrices (A, B, C, D) are not known; the only available information is given by an upper bound, s, on n, and by the input and output data sequences, $\{u_k\}$ and $\{y_k\}$, $k = 1{:}t$ (i.e., for k taking values from 1 to a given t).

2.1. The State Intersection Approach

The main feature of the SI class of SMI techniques is the determination of the state sequence of the LTI system to be identified, or of an observer to reconstruct its state sequence, via the intersection of the row spaces of the Hankel-like matrices constructed from "past" and "future" I/O data. The basic idea was first described in [4]. An extension of this idea led to the N4SID algorithm developed in [2]. This algorithm identifies LTI state space models in the so-called innovation form, described by

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Kv_k, \\ y_k &= Cx_k + Du_k + v_k, \end{aligned} \qquad (2)$$

where $\{v_k\}$ is a zero-mean white noise sequence and $\{u_k\}$ is a deterministic input sequence (perfectly known to the user). Both the basic and extended variants of this class produce statistically consistent and efficient estimates when certain assumptions hold.

2.2. The MOESP Approach

The main feature of this class of SMI techniques is the determination of an extended observability matrix of the deterministic part of the model (1). The extended observability matrix Γ_s is given by

$$\Gamma_s = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{s-1} \end{bmatrix}.$$

The basic idea was first described in [5] and [6]. The simplest algorithm derived from this idea, called the ordinary MOESP scheme, allows to identify, in a statistically consistent and efficient way, LTI systems that can be described by (1), with $w_k \equiv 0$ and $\{v_k\}$ a zero-mean white noise sequence, independent of the input.

Extensions based on using past input quantities and/or reconstructed state variables as instrumental variables have been proposed, allowing to consistently identify a model in (1), for $w_k \equiv 0$, and $\{v_k\}$ a zero-mean arbitrary stochastic disturbance, independent of the input. The price paid for this increased applicability is that the estimates are no longer efficient. These variants are referred to as the PI or RS schemes, when past inputs or reconstructed state variables are used as instrumental variables, respectively.

When the additive disturbance w_k in (1) is generated by an innovation model of the form $w_k = Kv_k$, and v_k is a zero-mean white noise independent of the input, it is again possible to obtain both consistent and efficient estimates when, besides the past input, past output quantities are also used as instrumental variables. The scheme derived from this extension of ordinary MOESP scheme is known as the PO scheme [3].

For the deterministic-stochastic identification problem, it is possible to estimate the deterministic part by using the PO scheme, and the stochastic part, that is the noise covariance matrices, by using the SI approach [2]. Further, the corresponding Kalman gain K can be computed to allow the design of state observers [3].

The algorithms belonging to the MOESP class have the striking feature of being highly streamlined, in the sense that the sequence of computations performed by these schemes is almost independent from the type of problems to be analyzed. This has led to highly modular implementations of the algorithms within this class.

3. Basic Algorithms

Full algorithmic details cannot be given, due to lack of space. Only some key points will be illustrated.

3.1. Algorithmic Outline

The simplest algorithm corresponds to the ordinary MOESP scheme. For non-sequential data processing, the $N\times(m+\ell)s$ matrix $H = \left[\begin{array}{cc} U_{1,s,N}^T & Y_{1,s,N}^T \end{array}\right]$ is constructed, where N denotes the total number of samples that can be used (here, $N = t-s+1$), $U_{1,s,N}$ and $Y_{1,s,N}$ are block-Hankel matrices defined in terms of the input and output data, respectively, i.e.,

$$U_{1,p,N} = \left[\begin{array}{ccccc} u_1 & u_2 & u_3 & \cdots & u_N \\ u_2 & u_3 & u_4 & \cdots & u_{N+1} \\ u_3 & u_4 & u_5 & \cdots & u_{N+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_p & u_{p+1} & u_{p+2} & \cdots & u_{N+p-1} \end{array}\right],$$

and similarly for Y. A QR factorization, $H = QR$, is used for data compressing; then a SVD of the submatrix $R_{22} := R(ms+1\colon(m+\ell)s, ms+1\colon(m+\ell)s)$ reveals the order n of the system as the number of "non-zero" singular values. System matrices are finally computed from the right singular vectors of R_{22}, using the $ms\times(m+\ell)s$ upper trapezoidal submatrix of R (see the next subsection).

For sequential data processing, the QR factorization is done sequentially, by updating the upper triangular factor R. When all data have been compressed, the system order and system matrices are computed as in the previous case.

For other schemes, the things are similar, but more involved. For the PO scheme, R_{22} is given by

$$R_{22} := R(ms+1\colon(2m+\ell)s, ms+1\colon(2m+\ell)s).$$

For the N4SID scheme, the triangular factor R of the $N\times 2(m+\ell)s$ matrix $H = \left[\begin{array}{cc} U_{1,2s,N}^T & Y_{1,2s,N}^T \end{array}\right]$ is found ($N = t-2s+1$), an oblique projection O is computed in terms of some submatrices of the upper triangular factor R [2], and then a SVD of O gives the order n. System matrices are finally computed using the first n right singular vectors of O^T, and other submatrices of R, solving a linear algebraic system in a total least squares sense [10]. The covariance matrices are computed using the residuals of a least squares problem. The Kalman gain is obtained by solving a discrete-time algebraic matrix Riccati equation using the Schur vectors approach [11] for the dual of an optimal control problem.

To give further details, let us partition the triangular factor of H as $R = [U_p \;\; U_f \;\; Y_p \;\; Y_f]$, where the subscripts p and f stand for "past" and "future" data, respectively, and the four blocks have ms, ms, ℓs, and ℓs columns, respectively. Define $W_p = [U_p \;\; Y_p]$, and consider the residuals of the two least squares problems giving the oblique projection,

$$r_1 = W_p - U_f X_1, \qquad r_2 = Y_f - U_f X_2,$$

where X_1 and X_2 are the minimum norm least squares solutions of the following problems

$$\min \|U_f X - W_p\|_2, \qquad \min \|U_f X - Y_f\|_2,$$

respectively. Then, the oblique projection can be computed as $O = r_2^T Q_1 Q_1^T$, where Q_1 consists of the first $k = \mathrm{rank}(r_1)$ columns of the orthogonal matrix in the QR factorization of r_1. No least squares problems should be actually solved.

3.2. Computation of System Matrices

The description below shows how the MOESP approach can estimate the quadruple of system matrices (A, B, C, D) of the LTI state space model using the information provided by previous computations.

Let R be a matrix whose leading $ms\times(m+\ell)s$ submatrix contains some relevant data for MOESP algorithm, namely the upper triangular matrix R_{11}, and the matrix R_{12}. (For the ordinary MOESP scheme, it is just the leading submatrix of the triangular factor of the QR factorization of H.) Let U be the $\ell s\times\ell s$ matrix of right singular vectors.

The matrix C is readily obtained as the leading $\ell\times n$ submatrix of U. Denoting $U_1 = U(1\colon(s-1)\ell, 1\colon n)$, and $U_2 = U(\ell+1\colon\ell s, 1\colon n)$, the QR decomposition of U_1 is computed, and U_2 is updated accordingly,

$$U_1 = Q\left[\begin{array}{c} \widetilde{R} \\ 0 \end{array}\right], \quad U_2 \leftarrow Q^T U_2.$$

Then, matrix A is obtained by solving $\widetilde{R}A = U_2(1\colon n, :)$ for A.

Finding the B and D matrices is more involved. First, the matrix equation $KR_{11}^T = \widetilde{U}_2^T R_{12}^T$ is solved for K, where $\widetilde{U}_2 = U(:,n+1:\ell s)$, and K is an $(\ell s-n)\times ms$ matrix. The matrix K is stored in the workspace, `K`. Then, compute the triangular factor of the QR factorization of the $s(\ell s-n)\times(\ell s+m)$ structured matrix $\begin{bmatrix} Q & K_e \end{bmatrix}$, implicitly defined by

$$\begin{bmatrix} Q_{1s} & Q_{1,s-1} & Q_{1,s-2} & \cdots & Q_{12} & Q_{11} & K_1 \\ 0 & Q_{1s} & Q_{1,s-1} & \cdots & Q_{13} & Q_{12} & K_2 \\ 0 & 0 & Q_{1s} & \cdots & Q_{14} & Q_{13} & K_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & Q_{1s} & K_s \end{bmatrix},$$

where $Q_{1i} = \widetilde{U}_2(\ell(i-1)+1:\ell i,:)^T$ is $(\ell s-n)\times\ell$, for $i=1:s$, $K_e = \begin{bmatrix} K_1^T & K_2^T & \cdots & K_s^T \end{bmatrix}^T$, and $K_i = K(:,(i-1)m+1:im)$ is $(\ell s-n)\times m$, $i=1:s$. The matrix $\begin{bmatrix} Q & K_e \end{bmatrix}$ is triangularized in an array `R`, exploiting its structure (for efficient use of the memory space), so that on output, the upper triangle of `R` contains, in its leading $(\ell s+m)\times(\ell s+m)$ submatrix, the required triangular factor. The calculations need $0((\ell s+m)^3)$ floating point operations. They are briefly described below. The first block-row of Q, i.e. the $(\ell s-n)\times\ell s$ matrix $\begin{bmatrix} Q_{1s} & \cdots & Q_{12} & Q_{11} \end{bmatrix}$, is constructed in `R`. The submatrix Q_{1s} is triangularized using an orthogonal matrix S, and the transformation S^T is then applied to the matrix $\begin{bmatrix} Q_{1,s-1} & \cdots & Q_{11} \end{bmatrix}$. Hence,

$$S^T\begin{bmatrix} Q_{1s}\cdots Q_{11} \end{bmatrix} = \begin{bmatrix} R & P_{s-1} & P_{s-2} & \cdots & P_2 & P_1 \\ 0 & F_{s-1} & F_{s-2} & \cdots & F_2 & F_1 \end{bmatrix}.$$

The transformation S^T is also applied to each of the submatrices K_i of K_e, $i=1:s$. Denote $\begin{bmatrix} C_i^T & G_i^T \end{bmatrix}^T = S^T K_i$ $(i=1:s)$, where C_i has ℓ rows. (Finally, C_i is saved in $\mathtt{R}(\ell(i-1)+1:\ell i, \ell s+1:\ell s+m)$, and G_i is in $\mathtt{K}(\ell+1:\ell s-n, m(i-1)+1:mi), i=1:s$.) Then, the rows to be annihilated are put in $F(1:\ell s-n-\ell, 1:\ell s-\ell)$. (In an implementation, F could overwrite $\widetilde{U}_2$.)

Now, the structure of the transformed matrix is:

$$\begin{bmatrix} R & P_{s-1} & P_{s-2} & \cdots & P_2 & P_1 & C_1 \\ 0 & R & P_{s-1} & \cdots & P_3 & P_2 & C_2 \\ 0 & 0 & R & \cdots & P_4 & P_3 & C_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & R & P_{s-1} & C_{s-1} \\ 0 & 0 & 0 & \cdots & 0 & R & C_s \\ 0 & F_{s-1} & F_{s-2} & \cdots & F_2 & F_1 & G_1 \\ 0 & 0 & F_{s-1} & \cdots & F_3 & F_2 & G_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & F_{s-1} & G_{s-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 & G_s \end{bmatrix},$$

where the block-rows have been permuted, to better exploit the structure. The block-rows having R on the diagonal are dealt with successively in the array `R`. The F submatrices are stored in an auxiliary $(\ell s-n-\ell, \ell s-\ell)$ array `F`, as a block-row. The large matrix above is never explicitly constructed. Only its first s block-rows are constructed, one by one. To triangularize the transformed matrix, exploiting its structure, part of the preceding block-row is first copied in the subsequent block-row, and then the current submatrix F_{s-i} is annihilated using an orthogonal matrix that modifies the corresponding submatrix R (in the same block-column). The transformation is applied to the corresponding block-rows of the arrays `R`, `F` and `K` (for G).

Finally, after C_i, $i=1:s$, have been copied in the last block-column of the QR factor, $\mathtt{R}(1:\ell s, \ell s+1:\ell s+m)$, $i=1:s$, the remaining rows of the transformed G are then compressed in the trailing part of `R`, $\mathtt{R}(\ell s+1:\ell s+m, \ell s+1:\ell s+m)$. Now, compute the right singular vectors matrix, V, of the triangular factor `R`, of order $\ell s+m$. (V^T is stored in `R`.) Solve for Y (the total least squares solution) the matrix equation

$$V(\ell s+1:\ell s+m, \ell s+1:\ell s+m)^T Y^T = \\ -V(\ell s+1:\ell s+m, 1:\ell s)^T,$$

where Y is $\ell s\times m$. Then, matrix D is readily obtained, if needed, as the trailing $\ell\times m$ submatrix of Y.

After rearranging the block-rows of U_1 as follows

$$\begin{bmatrix} U_1(\ell(s-2)+1:\ell(s-1),:) \\ \vdots \\ U_1(\ell+1:2\ell,:) \\ U_1(1:\ell,:) \end{bmatrix},$$

the modified U_1 matrix, extended by the first $\ell s-\ell$ rows of Y on the right, is triangularized. Finally, the right singular vectors matrix, V, of the triangular factor is computed, and the matrix equation

$$V(n+1:n+m, n+1:n+m)^T B^T = \\ -V(n+1:n+m, 1:n)^T,$$

is solved for B (in a total least squares sense).

For the N4SID algorithm, a similar computational scheme is used.

4. LAPACK-Based Software

To get an idea on the complexity of the developed software system, it is worth mentioning that there are six RASP-IDENT driver routines, five auxiliary routines for system identification, ten SLICOT compatible driver routines, and nineteen SLICOT compatible computational and auxiliary routines. There are 32 directly called LAPACK routines, and several directly called BLAS routines (six BLAS 1, four BLAS 2, and four BLAS 3 routines).

4.1. Driver Routines

The following driver routines are available:

RPIMOE computes consistent and statistically efficient estimates of the matrices (A, B, C, D) in (1) when $w \equiv 0$ and v is zero-mean white noise independent of the input, using the ordinary MOESP scheme.

RPIMPI (or RPIMRS) computes consistent estimates of the matrices (A, B, C, D) in (1) when $w \equiv 0$ and v is zero-mean, but of arbitrary statistical color, using the ordinary MOESP scheme extended with instrumental variables based on past input quantities (or reconstructed state variables, respectively).

RPIMPO computes consistent and statistically efficient estimates of the matrices (A, B, C, D) in (1) when w is generated by an innovation model and v is zero-mean white noise independent of the input, using the ordinary MOESP scheme extended with instrumental variables based on past input and past output quantities.

RPIMKG performs the computations in RPIMPO routine, but also estimates the noise covariance matrices and the Kalman gain (to allow the design of a state observer as a state predictor), using the SI approach.

RPIMN4 computes consistent and statistically efficient estimates of the matrices (A, B, C, D) in (2) when v is a zero-mean white noise sequence, using the basic N4SID algorithm.

All driver routines have an option for sequential I/O data processing.

4.2. Computational and Auxiliary Routines

Some of the implemented computational and auxiliary routines are listed below:

IMMPID estimates a finite set of Markov parameters from I/O data (no constraints on input sequence to be white noise, or initial conditions be zero).

IMSHFT shifts the I/O data sequences to compensate for the dead-times in the input vector components.

IMICID estimates the initial state x_0 of a LTI system, given the matrix quadruple (A, B, C, D) and the input and output trajectories of the system.

IMCLID computes the parameters of a LTI system operating in closed-loop with a LTI controller.

RPIMIU computes the output trajectory of a LTI discrete-time system, given the input trajectory, the initial state and/or the state and output disturbances (or noise) trajectories.

5. Examples

First, consider the discrete-time system (A, B, C, D), due to M. Verhaegen, with the following matrices:

$$A = \begin{bmatrix} 1.5 & -0.7 \\ 1.0 & 0.0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0.5 \end{bmatrix},$$

and $D = 0$, whose output response $\{y_k\}$ is computed using a random (0,1) uniform distribution input trajectory $\{u_k\}$, $k = 1{:}\,120$ ($t = 120$), and zero initial state. The parameter s is taken as 10.

The computed singular values used to estimate the system order, rounded to five significant digits, are: 32.203, 14.612, 0, 0, 0, 0, 0, 0, 0, 0. The estimated system matrices (rounded to four digits after the decimal point) are:

$$\widehat{A} = \begin{bmatrix} .7831 & .5606 \\ -.2472 & .7169 \end{bmatrix}, \quad \widehat{B} = \begin{bmatrix} -3.5468 \\ -2.3707 \end{bmatrix},$$

$$\widehat{C} = \begin{bmatrix} -.5749 & .4382 \end{bmatrix}, \quad \widehat{D} = 0.$$

The full precision computed matrices coincide with those obtained by applying an appropriate similarity transformation to the original system. The relative output root-mean-square error is:

$$\left(\sum_{k=1}^{t} (y_k - \widehat{y}_k)^2 \right)^{1/2} \Bigg/ \left(\sum_{k=1}^{t} y_k^2 \right)^{1/2} = 1.13 \times 10^{-15}.$$

As a second example, consider the following system

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

where [12]:

$$A = \begin{bmatrix} .9856 & .1628 & 0 & 0 & 0 & 0 \\ -.1628 & .9856 & 0 & 0 & 0 & 0 \\ 0 & 0 & .8976 & .4305 & 0 & 0 \\ 0 & 0 & -.4305 & .8976 & 0 & 0 \\ 0 & 0 & 0 & 0 & .8127 & .5690 \\ 0 & 0 & 0 & 0 & -.5690 & .8127 \end{bmatrix},$$

$$B = \begin{bmatrix} .0011 & .0134 & -.0016 & -.0072 & .0011 & .0034 \end{bmatrix}^T,$$

$$C = \begin{bmatrix} 1.5119 & 0 & 2 & 0 & 1.5119 & 0 \\ 1.3093 & 0 & 0 & 0 & -1.3093 & 0 \end{bmatrix}, \quad D = 0,$$

$$E = \mathrm{diag}(.00049, .00599, .00073, .00322, .00048, .00151),$$

and $F = \mathrm{diag}(.05277, .05277)$, whose output response $\{y_k\}$ is computed using random (0,1) normal distribution of input, process noise and output noise trajectories $\{u_k\}$, $\{w_k\}$, and $\{v_k\}$, $k = 1{:}\,120$ ($t = 120$), and zero initial state. The parameter s is taken as 8.

The computed singular values, rounded to five significant digits, are: 3.6458, 1.7754, .75446, .68453, .44332, .34115, .32113, .28318, .25693, .24346, .19697, .17021, .12216, .094313, .091442, .076357. These values indicate that the system order can be taken as two. However, we forced the value $n = 6$, as for the given system. The estimated system matrices (rounded to three digits after the decimal point) are:

$$\widehat{A} = \begin{bmatrix} .961 & .376 & .079 & .173 & -.205 & .069 \\ -.098 & .963 & .625 & -.043 & -.100 & .051 \\ .019 & -.106 & .914 & .429 & -.217 & .246 \\ .013 & .060 & -.289 & .895 & .165 & .175 \\ .011 & .004 & -.064 & -.047 & -.835 & .518 \\ -.007 & -.008 & .020 & .098 & -.434 & -.639 \end{bmatrix},$$

$$\widehat{B} = \begin{bmatrix} .079 & .045 & .021 & .025 & .001 & .001 \end{bmatrix}^T,$$

$$\widehat{C} = \begin{bmatrix} .273 & -.457 & .318 & -.026 & .059 & -.116 \\ .197 & -.151 & .044 & -.495 & .409 & .063 \end{bmatrix},$$

and $\widehat{D} = 0$. The estimated Kalman gain matrix $\widehat{K}$ is

$$\widehat{K} = \begin{bmatrix} .9152 & 1.7073 \\ -.3287 & .5212 \\ .0628 & -.1079 \\ .0727 & -.0860 \\ -.0040 & -.2926 \\ -.0197 & .1059 \end{bmatrix}.$$

The relative output root-mean-square error is about 1.9, for each column of y. The same statistics for the filtered output error, for each column of y, are 0.46, and 0.58, respectively.

6. Conclusions

Basic algorithms and LAPACK-based Fortran software for multivariable system identification by subspace techniques have been briefly described. The underlying approaches seem very promising, and the software components proved robust, flexible, and easy of use, even when the available memory is quite limited. The software has been extensively tested; the results obtained using various algorithms (including their MATLAB versions) have been compared. New versions of the codes, with improved modularity and efficiency, are under development. Implementation of new algorithms is also considered.

7. Acknowledgments

The author is indebted to Andras Varga and Michel Verhaegen for many helpful discussions.

8. References

[1] L. Ljung, *System Identification Toolbox.* Natick, MA: The MathWorks, Inc., 1992.

[2] P. Van Overschee and B. De Moor, "N4SID: Two subspace algorithms for the identification of combined deterministic-stochastic systems," *Automatica*, vol. 30, no. 1, pp. 75–93, 1994.

[3] M. Verhaegen, "Identification of the deterministic part of MIMO state space models given in innovations form from input-output data," *Automatica*, vol. 30, no. 1, pp. 61–74, 1994.

[4] M. Moonen, B. De Moor, L. Vandenberghe, and J. Vandewalle, "On- and off-line identification of linear state-space models," *Int. J. Control*, vol. 49, no. 1, pp. 219–232, 1989.

[5] M. Verhaegen, and P. Dewilde, "Subspace Model Identification. Part 1: The output-error state- space model identification class of algorithms," *Int. J. Control*, vol. 56, no. 5, pp. 1187–1210, 1992.

[6] M. Verhaegen, "Subspace Model Identification. Part 3: Analysis of the ordinary output-error state-space model identification algorithm," *Int. J. Control*, vol. 58, no. 3, pp. 555–586, 1993.

[7] P. Van Overschee, *Subspace Identification : Theory – Implementation – Applications*, Katholieke Universiteit Leuven, Ph. D. Thesis, 1995.

[8] M. Verhaegen, "State Space Model Identification Toolbox," Delft University of Technology, Tech. Rep., Nov. 1993.

[9] V. Sima, and A. Varga, "RASP–IDENT Subspace Model Identification Programs," DLR – Oberpfaffenhofen, Tech. Rep. TR R888–94, Oct. 1994.

[10] S. Van Huffel, and J. Vandewalle, *The Total Least Squares Problem: Computational Aspects and Analysis.* Philadelphia, PA: SIAM, 1991.

[11] A. J. Laub, "A Schur method for solving algebraic Riccati equations," *IEEE Trans. Automat. Control*, vol. AC–24, no. 6, pp. 913–921, 1979.

[12] J. N. Juang *et al*, "Identification of observer/Kalman filter Markov parameters: Theory and experiments," *J. Guidance, Control and Dynamics*, vol. 16, no. 2, Mar.–Apr. 1993.