COMPUTING THE OBLIQUE PROJECTION IN SUBSPACE-BASED MULTIVARIABLE SYSTEM IDENTIFICATION

Vasile Sima

Research Institute for Informatics, Bd. Mareşal Averescu, Nr. 8-10, 71316 Bucharest 1, ROMANIA. E-mail: vsima@u3.ici.ro

Abstract. Recently proposed algorithms for multivariable system identification by subspace techniques involve the computation of the so-called "oblique projection" to estimate the system order and system matrices. The paper shows that this computation can be performed without actually solving any least-square problems, and that orthogonal transformations only should be applied to the original input-output data sequences. Both deterministic and combined deterministic-stochastic identification problems can be dealt with. *Copyright © 1998 IFAC*

Keywords. Computational methods, discrete-time systems, identification algorithms, least squares, singular value decomposition, subspace methods, system matrices, system order.

1. INTRODUCTION

Computer-aided design of control systems 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 schemes, 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 analyze, or could lead to ill-conditioned problems. Some other advantage will be that robust numerical linear algebra techniques, like QR factorization and singular value decomposition (SVD), can largely be applied; this is in contrast with the iterative optimization schemes required in the parametric model identification approach, documented e.g. in (Ljung 1992). The attractiveness of SMI techniques is even more exercised 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, (Moonen *et al.* 1989, Van Overschee 1995, Van Overschee and De Moor 1994, Verhaegen 1993*b*, Verhaegen 1994, Verhaegen and Dewilde 1992) for a further discussion of the SMI features.

There are two main 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 method described in (Van Overschee and De Moor 1994) 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, described in (Verhaegen 1993*b*, Verhaegen 1994, Verhaegen and Dewilde 1992).

MATLAB codes based on SMI algorithms have lately been developed, e.g. in (Verhaegen 1993*a*).

Some algorithms have been also implemented in Fortran, using the state-of-the-art, public-domain linear algebra package LAPACK (Anderson *et al.* 1995). This allows to exploit the potential parallelism of many modern computer architectures. Deterministic and combined deterministic-stochastic identification problems for discrete-time multivariable systems are dealt with. It is possible to handle multiple I/O sequences, each one being collected during a possibly independent identification experiment. There is an option for sequential processing of large data sets. Details about the Fortran-implemented algorithms are given in (Sima 1996*b*, Sima 1996*c*). In implementing algorithms, great attention was paid to developing new LAPACK-style codes for special QR or SVD, to exploit the particular structure of the problem, increase efficiency, and reduce the memory requirements. 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, L, of an LQ factorization of a Hankel-like matrix constructed from I/O sequences. The MATLAB implementations solve several standard least-squares problems to obtain the so-called associated "oblique projection". The new LAPACK-based implementation only involves some orthogonal transformations for obtaining various projections and residuals. While an LQ factorization is convenient from a theoretical point of view, in practice the QR factorization is to be preferred in computations. For instance, there is no code available for performing the LQ factorization with pivoting, and pivoting is needed for certain calculations where a rank decision should be made. The current MATLAB codes compute the lower triangular factor $L = R^T$, then determine the singular value decomposition of R^T, hence $R^T = U\Sigma V^T$, and use the singular values in Σ for finding the system order, n (by visually detecting the widest "gap" between two consecutive singular values), and the left singular vectors in U for determining system matrices. Our approach consists in computing $R = U\Sigma V^T$, and using the right singular vectors (columns in V^T).

2. BASIC APPROACHES

The basic LTI state space models considered are described by

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

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 SI Approach*

The main feature of the SI class of SMI techniques is the determination of either 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. An extension of the basic idea in (Moonen *et al.* 1989) led to the N4SID algorithm developed in (Van Overschee and De Moor 1994). This algorithm identifies LTI models in the innovation form, described by

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

where $\{v_k\}$ is a zero-mean white noise sequence and $\{u_k\}$ is a deterministic input sequence. 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 to determine an extended observability matrix of the deterministic part of the model (1). The extended observability matrix Γ_s is given by

$$\Gamma_s = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{s-1} \end{bmatrix}. \tag{3}$$

The basic idea was introduced in (Verhaegen and Dewilde 1992) and (Verhaegen 1993*b*). The simplest algorithm based on 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 on 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. This increased applicability is made at the risk of not having efficient estimates. 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 possible again to obtain both consistent and efficient estimates when, beside the past input, past output quantities are also used as instrumental variables.

3. BASIC ALGORITHMS

An important computational step of the N4SID class of algorithms is the determination of the so-called "oblique projection", which is used to estimate the system order and then the system matrices. The MATLAB codes for performing this computation usually solve three least-squares problems. The paper shows that the oblique projection can however be obtained without actually solving any least-squares problems, and that orthogonal transformations only should be applied to the original input-output data sequences. This result is important for efficiency and numerical accuracy reasons.

3.1 *Computation of Oblique Projection*

For the N4SID scheme, the $N \times 2(m+\ell)s$ block matrix $H = \left[U_{1.2s.N}^T \; Y_{1.2s.N}^T \right]$ is constructed, where N denotes the total number of samples that can be used (here, $N = t - 2s + 1$), and $U_{1.2s.N}$ and $Y_{1.2s.N}$ are block-Hankel matrices defined in terms of the input and output data, respectively, i.e.,

$$U_{1.p.N} = \begin{bmatrix} u_1 & u_2 & u_3 & \cdots & u_N \\ u_2 & u_3 & u_4 & \cdots & u_{N+1} \\ u_3 & u_4 & u_5 & \cdots & u_{N+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_p & u_{p+1} & u_{p+2} & \cdots & u_{N+p-1} \end{bmatrix},$$

and similarly for Y. A QR factorization, $H = QR$, is used for data compression, giving the triangular factor R, an oblique projection O is computed in terms of some submatrices of the upper triangular factor R of H (Van Overschee and De Moor 1994), and then a SVD of O reveals the order n of the system as the number of "non-zero" singular values. 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 (Van Huffel and Vandewalle 1991). 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 based on the Schur vectors approach for the dual of an optimal control problem (see, for example, (Sima 1996*a*, Ch. 3)).

Let us partition the triangular factor of H as

$$R = \left[U_p \; U_f \; Y_p \; Y_f \right],$$

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. These notations will be used below.

Let E, F, and G be three matrices with the same number of columns. The *(orthogonal) projection* of the row space of E onto the row space of F is defined as (Van Overschee and De Moor 1994, Van Overschee 1995)

$$E/\boldsymbol{F} = E\Pi_F = EF^T(FF^T)^\dagger F,$$

where Π_F is the *projection operator*, and the superscript † denotes the (Moore-Penrose) pseudo-inverse. Note that this definition slightly differs from the similar MATLAB notation `E/F`, which can be written as $EF^T(FF^T)^\dagger$. Indeed, in MATLAB, `X = B/A` is equivalent to `X = (A'\B')'`, and `Y = C\D` means to solve in Y the set of linear algebraic equations $CY = D$ in a least-squares sense. This can be performed by QR factorization with column pivoting or, preferably, by singular value decomposition. Hence, `X = B/A` corresponds to solving $A^T X^T = B^T$ (or $XA = B$) in a least-squares sense. Moreover, `(B/A)*A` is the approximation of B using the minimum norm least-squares solution, and, clearly, `B - (B/A)*A` is the residual of that solution.

The orthogonal projection of the row space of E onto the orthogonal complement of the row space of F is, clearly,

$$E/\boldsymbol{F}^\perp = E\Pi_F^\perp = E(I - \Pi_F) = E - E/\boldsymbol{F}.$$

```
L = triu(qr([U; Y]'))';
Lf = L( (2*m+1)*s+1: 2*(m+1)*s, :);
Lp = [L(1:m*s,:); L(2*m*s+1:(2*m+1)*s,:)];
Lu  = L(m*s+1: 2*m*s, 1: 2*m*s);
Lfp = [Lf(:, 1: 2*m*s) - ..
      (Lf(:, 1: 2*m*s)/Lu)*Lu, ..
       Lf(:, 2*m*s+1: 2*(m+1)*s)];
Lpp = [Lp(:, 1: 2*m*s) -  ..
      (Lp(:, 1: 2*m*s)/Lu)*Lu, ..
       Lp(:, 2*m*s+1: 2*(m+1)*s)];
if (norm(Lpp(:,(2*m+1)*s-2*1:(2*m+1)*s),..
        'fro')) < 1e-10
  O  = (Lfp*pinv(Lpp')')*Lp;
else
  O = (Lfp/Lpp)*Lp;
end
```

Fig. 1. MATLAB code for the computation of the oblique projection.

The *oblique projection* of the row space of E along the row space of F on the row space of G is defined as (Van Overschee and De Moor 1994, Van Overschee 1995)

$$E/_F \boldsymbol{G} = [E/\boldsymbol{F}^{\perp}][G/\boldsymbol{F}^{\perp}]^{\dagger}G.$$

With MATLAB notation, one obtains

$$E/_F \boldsymbol{G} = [E-(E/F)F][G-(G/F)F]^{\dagger}G.$$

The N4SID algorithm requires the oblique projection O,

$$O = Y_f/_{U_f} \boldsymbol{W}_p = [Y_f/\boldsymbol{U}_f^{\perp}][W_p/\boldsymbol{U}_f^{\perp}]^{\dagger}W_p.$$

In particular, the projection algorithms (e.g., (Verhaegen and Dewilde 1992)) use a special weighting matrix $\Pi_{U_f^{\perp}}$, and compute $O\Pi_{U_f^{\perp}}$.

The MATLAB code in Figure 1 is an adaptation of a typical oblique projection computation code, accompanying (Van Overschee and De Moor 1996). This code shows that O is found by solving three least-squares problems, two of them having the same coefficient matrix `Lu`.[1] There is the following correspondences between the MATLAB notation and previous mathematical notation:

$$\texttt{Lf} \equiv Y_f^T, \qquad \texttt{Lu} \leftrightarrow U_f^T,$$
$$\texttt{Lp} \equiv W_p^T, \qquad \texttt{Lfp} \equiv r_2^T, \qquad \texttt{Lpp} \equiv r_1^T$$

(`Lu` retains the non-zero part only); hence,

$$O\Pi_{U_f^{\perp}} = (r_2^T/r_1^T)r_1^T.$$

[1] Some codes replace the last two `Lp` matrices by `Lpp`.

that is, $O\Pi_{U_f^{\perp}}$ is the transpose of the least-squares approximation of r_2 in a least-squares problem having r_1 as coefficient matrix. Taking a QR factorization with column pivoting of r_1,

$$r_1 = Q\widetilde{R} = [\,Q_1 \;\; Q_2\,] \begin{bmatrix} \widetilde{R}_{11} \\ 0 \end{bmatrix},$$

where matrix $\widetilde{R}_{11}$ has full row rank k, and Q_1 has k columns, the required least-squares approximation of r_2 can be computed as $Q_1Q_1^Tr_2$. For numerical computations, an incremental condition estimation technique is used for determining the numerical rank of r_1. Summarizing, the weighted oblique projection can be obtained from

$$O\Pi_{U_f^{\perp}} = r_2^TQ_1Q_1^T.$$

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 actually be solved.

3.2 *Computation of System Matrices*

Let R be the triangular factor of the QR factorization of H, and let U be the matrix of right singular vectors of O^T. Let $\widehat{\Gamma}$ be the matrix formed by the first n columns of U. Clearly, $\widehat{\Gamma}$ is an estimate of Γ_s, hence from (3), it follows that $C \approx \widehat{\Gamma}_{1:\ell,:}$, $A \approx \widehat{\Gamma}^{\dagger}_{1:(s-1)\ell,:}\widehat{\Gamma}_{\ell+1:s\ell,:}$. Specifically, C is readily obtained as the leading $\ell \times n$ submatrix of U. Moreover, denoting $U_1 = U(1:(s-1)\ell, 1:n)$, and $U_2 = U(\ell+1:\ell s, 1:n)$, the QR decomposition of U_1 is computed, and U_2 is updated accordingly,

$$U_1 = Q\begin{bmatrix} \widetilde{R} \\ 0 \end{bmatrix}, \quad U_2 \leftarrow Q^TU_2.$$

Then, matrix A is obtained by solving the set of linear algebraic systems $\widetilde{R}A = U_2(1:n,:)$ for A.

The determination of the B and D matrices is more involved. Typical relations (corresponding to the MOESP approach) are given below. From the I/O equations (Van Overschee 1995, Van Overschee and De Moor 1996),

$$Y_{s+1,2s,N} = \Gamma_sX_{s,s,N} + H_sU_{s+1,2s,N},$$

with,

$$X_{s,s,N} = [\,x_s \;\; x_{s+1} \;\; \cdots \;\; x_{s+N-1}\,],$$

$$H_s = \begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{s-2}B & CA^{s-3}B & CA^{s-4}B & \cdots & D \end{bmatrix}.$$

it follows, with obvious notation

$$[\, K_1 \;\; K_2 \;\; \ldots \;\; K_s \,] = [\, Q_{11} \;\; Q_{12} \;\; \ldots \;\; Q_{1s} \,]\, H_s,$$

where K_i $(\ell s - n) \times m$, and Q_{1i} $(\ell s - n) \times \ell$, $i = 1{:}s$, or equivalently,

$$\begin{bmatrix} K_1 \\ K_2 \\ \vdots \\ K_s \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & \cdots & Q_{1,s-2} & Q_{1,s-1} & Q_{1s} \\ Q_{12} & Q_{13} & \cdots & Q_{1,s-1} & Q_{1s} & 0 \\ Q_{13} & Q_{14} & \cdots & Q_{1s} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Q_{1s} & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} I_\ell & 0 \\ 0 & \widehat{\Gamma}_{1:(s-1)\ell,:} \end{bmatrix} \begin{bmatrix} D \\ B \end{bmatrix}.$$

The relations above define a set of linear algebraic equations in B and D, which should be solved using a total least-squares approach. The form of these relations are, however, not suitable for an efficient implementation, and their special structure cannot be directly exploited. It is possible to employ a block-column permutation, to put the block columns of the matrix Q in the reverse order. The implemented approach is very briefly described below, assuming that K_i are already available. (Note that $K_i = K(:, (i-1)m + 1{:}im)$, and K can be computed by solving a set of triangular linear algebraic equations.)

Consider the $s(\ell s - n) \times (\ell s + m)$ structured matrix $[Q \;\; K_e]$, implicitly defined by

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

The matrix $[Q \;\; K_e]$ 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. Details of the basic computations performed to estimate the quadruple of system matrices (A, B, C, D) of the LTI state space model are summarized in (Sima 1996*b*, Sima 1996*c*). Similar calculations are used for implementing the N4SID algorithm.

3.3 *Details on Applying the Transformations*

The implemented calculations use the LAPACK abilities for applying orthogonal transformations represented in a factored form, without storing any orthogonal matrix. The transformations employed are products of elementary reflectors (Householder transformations), defined by $H = I - 2uu^T$, where u is a vector. The non-zero elements of u are stored just in the memory locations containing the numerical values which were zeroed, plus one additional location. There are LAPACK routines for pre- or postmultiplying a given matrix by a sequence of reflectors in forward or backward order (corresponding to applying either the orthogonal transformation, or its transpose, respectively).

The special structure of the problem can be exploited. This will be illustrated below. For $m = 1$, $\ell = 2$, $s = 3$, the matrices U_f and Y_f can be represented as

$$U_f = \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad Y_f = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & 0 & x \end{bmatrix},$$

with x denoting a possibly non-zero value. When computing the residual r_2 of the least-squares problem

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

a QR factorization of U_f is needed, $U_f = \bar{Q}\bar{R}$, where $\bar{Q}$ is an 18×18 orthogonal matrix. But $\bar{Q}$ can be expressed as $\bar{Q} = \mathrm{diag}(q, I_{2\ell s})$, where q is a product of three elementary reflectors, q_1, q_2, and q_3, which annihilates the subdiagonal elements of $(U_f)_{1:6,:}$. Specifically, q_1 is chosen to annihilate $(U_f)_{2:4,1}$, q_2 annihilates $(U_f)_{3:5,2}$, and q_3 annihilates $(U_f)_{4:6,3}$. These three reflectors modify the first six rows of Y_f only, and the same rows of W_p. Since it is possible that $\mathrm{rank}(U_f) < ms = 3$, another QR factorization with column pivoting (and incremental condition estimation) should be used for the upper triangular part of $\bar{R}$, and the corresponding orthogonal transformation should be applied to the modified Y_f like discussed above. Clearly, exploiting the structure is not important for small values of m, ℓ, and s, but it can lead to significant efficiency improvement for large values. In practice, s should take large values, especially

for stochastic systems. Similar structure-exploiting techniques are used for computing the QR factorization of r_1. Note that the first block-row of r_1 is already upper triangular, and its third block-row has the form $[0\ T]$, where T is upper triangular.

4. CONCLUSIONS

The determination of the "oblique projection" is an important computational step of the N4SID class of algorithms for estimating the system order and system matrices for multivariable discrete-time system identification by subspace techniques. A special form of this projection can be obtained without actually solving any least-squares problems. Only orthogonal transformations have to be applied to the original input-output data sequences. This is important both for efficiency and numerical accuracy reasons.

5. REFERENCES

Anderson, E., Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen (1995). *LAPACK Users' Guide: Second Edition*. SIAM, Philadelphia.

Ljung, L. (1992). *System Identification Toolbox*. The Math Works, Inc., Natick, MA.

Moonen, M., B. De Moor, L. Vandenberghe and J. Vandewalle (1989). On- and off-line identification of linear state space models. *Int. J. Control*. **49**, 219–232.

Sima, V. (1996*a*). *Algorithms for Linear-Quadratic Optimization*. Marcel Dekker, Inc., New York.

Sima, V. (1996*b*). Algorithms and LAPACK-based software for subspace identification. In: *Proceedings of The 1996 IEEE International Symposium on Computer-Aided Control System Design, September 15–18, 1996, Ritz-Carlton, Dearborn, Michigan, U.S.A.*, pp. 182–187. IEEE Control Systems Society.

Sima, V. (1996*c*). Subspace-based algorithms for multivariable system identification. *Studies in Informatics and Control*, **5**, 335–344.

Van Huffel, S. and J. Vandewalle (1991). *The Total Least Squares Problem: Computational Aspects and Analysis*. Vol. 9 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia.

Van Overschee, P. (1995). Subspace Identification : Theory – Implementation – Applications. PhD thesis. Katholieke Universiteit Leuven, Leuven.

Van Overschee, P. and B. De Moor (1994). N4SID: Two subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, **30**, 75–93.

Van Overschee, P. and B. De Moor (1996). *Subspace Identification for Linear Systems: Theory—Implementation—Applications*. Kluwer Academic Publishers, Boston.

Verhaegen, M. (1993*a*). State space model identification toolbox. Technical Report. Delft University of Technology, Delft.

Verhaegen, M. (1993*b*). Subspace model identification. Part 3: Analysis of the ordinary output-error state-space model identification algorithm. *Int. J. Control*, **58**, 555–586.

Verhaegen, M. (1994). Identification of the deterministic part of MIMO state space models given in innovations form from input-output data. *Automatica*, **30**, 61–74.

Verhaegen, M. and P. Dewilde (1992). Subspace model identification. Part 1: The output-error state-space model identification class of algorithms. *Int. J. Control*, **56**, 1187–1210.