# FAST SYSTEM IDENTIFICATION AND MODEL REDUCTION SOLVERS

**Vasile Sima** [*] **Peter Benner** [**]

[*] *National Institute for Research & Development in Informatics*
*Bd. Mareşal Averescu, Nr. 8–10, 011455 Bucharest 1, Romania*
*Fax: (+40)-21-316-1030*
*E-mail:* `vsima@ici.ro`
[**] *Fakultät für Mathematik, TU Chemnitz*
*D-09107 Chemnitz, Germany*
*Fax : (+49)-371-531-22509*
*E-mail:* `benner@mathematik.tu-chemnitz.de`

Abstract: Efficient numerical techniques for multivariable system identification and model reduction are investigated. The techniques are implemented in the system identification and model reduction toolboxes based on the Fortran 77 **S**ubroutine **L**ibrary **i**n **Co**ntrol **T**heory (SLICOT). Besides highly performant Fortran drivers and computational routines, these toolboxes provide MATLAB or Scilab interfaces, implementing several algorithmic approaches. Extensive numerical testing and comparisons with similar MATLAB tools show that the solvers in these toolboxes are reliable, efficient, and able to solve industrially relevant problems. *Copyright ©2007 IFAC*

Keywords: Identification Algorithms, Linear Multivariable Systems, Model Reduction, Numerical Algorithms, Parameter Estimation, Singular Value Decomposition

## 1. INTRODUCTION

The availability of powerful system identification and model reduction tools is very important in practice, since modern control techniques are increasingly dependent on suitable dynamical models. The numerical procedures need to be based on reliable and robust numerical software provided by well-tested and user-friendly software libraries. This paper summarizes the functional and computational performances of the MATLAB [1] identification and model reduction toolboxes based on the SLICOT Library (Benner *et al.*, 1999). [2]

The SLICOT-based MATLAB toolboxes provide a user-friendly interface to the highly efficient, robust, and portable Fortran 77 SLICOT routines. The MATLAB M-functions contained in these toolboxes are based on MEX-files calling the Fortran routines. While the M-functions are destined to all user categories, the more sophisticated and flexible MEX-functions are intended for expert users and software developers. Executable SLICOT MEX-files are provided for MATLAB running under WINDOWS 95–XP, Sun Solaris, and Linux.

Most of the SLICOT functionality is concerned with *linear time-invariant (LTI) systems in state-space form*:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t)+\mathbf{B}\mathbf{u}(t)\\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t)+\mathbf{D}\mathbf{u}(t),\end{aligned} \tag{1}$$

[1] MATLAB is a registered trademark of The MathWorks.

[2] The SLICOT (Subroutine Library in Control Theory) software library and the related CACSD tools based on SLICOT were developed within the Numerics in Control Network (NICONET) funded by the European Community BRITE-EURAM III RTD Thematic Networks Programme, see `http://www.icm.tu-bs.de/NICONET`. SLICOT can be used free of charge by academic users, see `http://www.slicot.org`.

in the continuous-time case, and

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k,\end{aligned} \tag{2}$$

in the discrete-time case, where $\mathbf{A}$ is $n \times n$, $\mathbf{B}$ is $n \times m$, $\mathbf{C}$ is $\ell \times n$, and $\mathbf{D}$ is $\ell \times m$. All matrices are assumed to be real.

Alternatively, an LTI system (1) or (2) may be represented by a rational transfer-function matrix (TFM),

$$\mathbf{G}(\lambda) = \mathbf{C}(\lambda\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}, \tag{3}$$

where $\lambda$ is the variable $s$ appearing in the Laplace transform, or the $z$ variable appearing in the $z$-transform, respectively.

Sections 2 and 3 provide a short description of the two SLICOT toolboxes emphasizing their potential in solving computationally challenging problems.

## 2. THE SYSTEM IDENTIFICATION TOOLBOX

Consider a discrete-time system in *innovation form*,

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{K}\mathbf{e}_k\,, \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k + \mathbf{e}_k\,,\end{aligned} \tag{4}$$

where $\mathbf{x}_k$ is the state vector at time $k$, $\mathbf{u}_k$ and $\mathbf{y}_k$ are the input and output vectors, respectively, $\{\mathbf{e}_k\}$ is a zero mean white noise sequence, uncorrelated with $\{\mathbf{u}_k\}$ and with the initial state of the system, and $\mathbf{K}$ is the *Kalman gain matrix*; $(\mathbf{A}, \mathbf{C})$ is assumed observable.

In *system identification* problems, the system order, $n$, and the quadruple of system matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ have to be determined (up to a system similarity transformation) using the input and output data sequences, $\{\mathbf{u}_k\}$ and $\{\mathbf{y}_k\}$, $k = 1 : \bar{N}$. In addition, the Kalman gain matrix $\mathbf{K}$ in (4) has often to be found.

The SLICOT calculations are based on *subspace identification* algorithms, which use a matrix $\mathbf{H}$, built from block-Hankel matrices, using part of the available input-output (I/O) data. Two approaches, MOESP (Multivariable Output Error state-space) (Verhaegen and Dewilde, 1992), and N4SID (Numerical algorithm for Subspace State Space System IDentification) (Van Overschee and De Moor, 1994), and their combination are used. The matrix $\mathbf{H}$ is given by

$$\begin{aligned}\mathbf{H} &= \left[\, \mathbf{U}^T_{q+1,q+s,N+q} \;\; \mathbf{U}^T_{1,q,N} \;\; \mathbf{Y}^T_{1,r+s,N} \,\right] \text{ (MOESP)}, \\ \mathbf{H} &= \left[\, \mathbf{U}^T_{1,q+s,N} \;\; \mathbf{Y}^T_{1,r+s,N} \,\right] \qquad\qquad \text{(N4SID)},\end{aligned}$$

with $\mathbf{H} \in \mathbb{R}^{N \times (mq + \ell r + (m+\ell)s)}$, where $N \le \bar{N} - s - \min(q, r) + 1$ (usually $N \gg mq + \ell r + (m+\ell)s$),

$$\mathbf{U}_{a,b,c} = \begin{bmatrix} \mathbf{u}_a & \mathbf{u}_{a+1} & \mathbf{u}_{a+2} & \cdots & \mathbf{u}_c \\ \mathbf{u}_{a+1} & \mathbf{u}_{a+2} & \mathbf{u}_{a+3} & \cdots & \mathbf{u}_{c+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{u}_b & \mathbf{u}_{b+1} & \mathbf{u}_{b+2} & \cdots & \mathbf{u}_{c+b-a} \end{bmatrix}, \tag{5}$$

and similarly for $\mathbf{Y}_{a,b,c}$. The latest version of the MATLAB function `n4sid` from the System Identification Toolbox (Ljung, 2000) can use different "prediction horizons" $s$, $q$, and $r$. But standard MOESP and N4SID algorithms, assumed below, use $q = r = s$, where $s$ denotes the "number of block rows", and $s$ usually satisfies $s \ge n$.

The first step in a subspace identification procedure is to perform a "data compression" by computing the upper triangular factor $\mathbf{R}$ of a QR factorization of the matrix $\mathbf{H}$, $\mathbf{H} = \mathbf{Q}\mathbf{R}$; the matrix $\mathbf{Q} \in \mathbb{R}^{N \times 2(m+\ell)s}$, satisfying $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_{2(m+\ell)s}$, is not needed subsequently. Parts of $\mathbf{R}$ are further used to estimate $n$ and system matrices. As many theoretical results concerning, e.g., consistency and normality of the estimates of the system matrices hold asymptotically, for $\bar{N} \to \infty$, the algorithms should work with large amounts of data, if available. The standard QR factorization algorithm could be too costly, because $\mathbf{H}$ can have very large dimensions. Since matrix $\mathbf{H}$ is highly structured, efficient data processing is possible by using the problem structure. Two such techniques implemented in the SLICOT Library are based on fast Cholesky and fast QR factorization algorithms, which exploit the special structure of the matrix $\mathbf{H}$.

The Cholesky factorization algorithm efficiently builds the inter-correlation matrix $\mathbf{W} = \mathbf{H}^T\mathbf{H}$, and then factors $\mathbf{W}$, assuming it is positive definite. (In the rare case when this algorithm fails, the usual QR factorization is automatically used.) For the standard N4SID approach, the block-Hankel matrices corresponding to the inputs and outputs are $\mathbf{H}_u = \mathbf{U}^T_{1,2s,N}$ and $\mathbf{H}_y = \mathbf{Y}^T_{1,2s,N}$, and $\mathbf{H} = \left[\mathbf{H}_u \;\mathbf{H}_y\right]$. The definitions extend to multiple I/O data batches. Actually, assuming that the latest batch to be processed is defined by $\mathbf{H}$, then $\mathbf{W} = \widetilde{\mathbf{W}} + \mathbf{H}^T\mathbf{H}$, where $\widetilde{\mathbf{W}}$ corresponds to the already processed data batches. Clearly, $\widetilde{\mathbf{W}} = 0$ if the first (or single) data batch is processed. Two lemmas in (Sima *et al.*, 2004; Sima, 2004) give cheap formulas for computing the symmetric block matrix $\mathbf{W}$ for the N4SID and MOESP approaches, respectively. In the later case, the matrix for the N4SID technique is built, and then it is transformed for the MOESP case.

Consider now the fast QR factorization algorithm for the N4SID approach. Define the *shift matrix* $\mathbf{Z} = \operatorname{diag}(\mathbf{Z}_u, \mathbf{Z}_y)$, where $\mathbf{Z}_u$ ($\mathbf{Z}_y$) is a $2s \times 2s$ block matrix with $m \times m$ ($\ell \times \ell$) blocks, all zero except for identity blocks on the superdiagonal. A lemma in (Sima *et al.*, 2004) shows that the symmetric matrix $\nabla\mathbf{W} = \mathbf{W} - \mathbf{Z}^T\mathbf{W}\mathbf{Z}$, called the *displacement* of $\mathbf{W}$ (Kailath and Sayed, 1995), has a low rank factorization. Specifically, $\nabla\mathbf{W}$ can be written as

$$\nabla\mathbf{W} = \mathbf{G}^T\Sigma\mathbf{G}, \quad \Sigma = \operatorname{diag}(\mathbf{I}_p, -\mathbf{I}_q), \tag{6}$$

where $p = q = m + \ell + 1$, hence $\nabla\mathbf{W}$ has the rank $2(m+\ell+1)$ at most. The matrix $\mathbf{G} \in \mathbb{R}^{2(m+\ell+1) \times 2(m+\ell)s}$ is called the *generator* of $\mathbf{W}$, and efficient techniques are available to build the generators and the factor

**R** (Kailath and Sayed, 1995; Mastronardi *et al.*, 2001). In practical calculations, the *generalized Schur algorithm* is used to find all rows of the Cholesky factor of **W**, $\mathbf{W} = \widetilde{\mathbf{W}} + \mathbf{H}^T\mathbf{H}$. Householder transformations and hyperbolic rotations are involved. Details are given in (Mastronardi *et al.*, 2001). For the MOESP approach, the same algorithm as for N4SID is used, but the first two block columns of the resulting upper triangular factor **R** are interchanged, and then retriangularized, exploiting the structure.

SLICOT System Identification Toolbox can also be used to identify Wiener systems. A discrete-time Wiener system has a state-space representation

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \qquad \mathbf{z}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k, \\ \mathbf{y}_k &= \mathbf{f}(\mathbf{z}_k) + \mathbf{e}_k, \end{aligned} \tag{7}$$

where $\mathbf{f}(\cdot)$ is a nonlinear vector function from $\mathbb{R}^\ell$ to $\mathbb{R}^\ell$. Briefly speaking, a Wiener system consists of a linear dynamic block followed by a static nonlinearity. The system identification problem for (7) requires to find an approximation of $\mathbf{f}(\cdot)$, besides estimating $n$, the quadruple $(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D})$, and the initial conditions. The implemented approach uses a state-space representation for the linear part and a single layer neural network to model the static nonlinearity. Fast subspace identification algorithms are employed for estimating the linear part, based on the available input-output data. The resulted state-space model is used for finding an approximate model of the nonlinear part by a Levenberg-Marquardt (LM) algorithm. Finally, the whole model is refined using a specialized, structure-exploiting LAPACK-based scaling-invariant LM algorithm. The output normal form is used to parameterize the linear part. The parameters corresponding to the nonlinear part come first in the global parameter vector. Using this ordering, the Jacobian matrices in the multi-output case ($\ell > 1$) have a block diagonal form, with an additional block column at the right. This structure is preserved in a QR factorization with column pivoting restricted to each block column. The rank deficient case is also covered. The Jacobian is computed analytically, for the nonlinear part, and numerically, for the linear part. The implementation is memory conserving and significantly faster than standard LM algorithms or specialized LM calculations based on conjugate gradients (without preconditioning) for solving linear systems. Details and numerical results are given, e.g., in (Sima, 2003*a*; Sima, 2003*b*).

### 2.1 Functionality of the Toolbox

Summarizing, SLICOT System Identification Toolbox includes SLICOT-based MATLAB and Fortran tools for linear and Wiener-type, time-invariant discrete-time multivariable systems. The approaches MOESP, N4SID, and their combination, are used to identify linear systems, and to initialize the parameters of the linear part of a Wiener system. The toolbox functionalities include:

- identification of linear discrete-time state-space systems $(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D})$;
- identification of state and output covariance matrices for such systems;
- estimation of the associated Kalman gain matrix;
- estimation of the initial state;
- conversion from/to a state-space representation to/from the output normal form;
- identification of discrete-time Wiener systems;
- computation of the output of Wiener systems.

Attractive features of this toolbox include:

- possible speed-up factors larger then 10 in comparison with the commonly used software tools;
- standard or fast techniques for data compression (exploiting the block-Hankel structure);
- fast estimation of system models of various, specified orders;
- ability to process multiple data batches;
- specialized, structure-exploiting LM algorithm using block QR factorization with pivoting;
- optional assessing the quality of the intermediate results using the associated condition numbers.

### 2.2 Performance Results

Performance evaluation of the system identification software has been performed using data sets from the DAISY collection, publicly available from the Internet site `www.esat.kuleuven.be/sista/daisy`. Accuracy and efficiency comparisons of the SLICOT linear systems identification software and the available subspace-based codes for 25 applications are presented in (Sima *et al.*, 2004). New results are given below, comparing the SLICOT code `slmoen4` and MATLAB function `n4sid` (Ljung, 2000). The main SLICOT-based call for standard calculations was

```
[sys,K,rcnd] = slmoen4(s,y,u,n,alg);
```

The results have been obtained on an Intel Pentium 4 computer, at 3 GHz, with 1 GB RAM, under Windows XP, with Compaq Visual Fortran V6.5 and optimized LAPACK and BLAS (available in MATLAB), under MATLAB 7.0.4.365 (R14) Service Pack 2.

Figure 1 shows a timing comparison of `slmoen4` with fast QR factorization (`alg` = 2) vs MATLAB 7.0.4 `n4sid` with standard QR factorization and default options, but with `order` = $n$, `'N4Weight'` = `'MOESP'`, and four cases (a)–(d) for the possible combinations of values for `'N4H'` := `'N4Horizon'` and `'Cov'` := `'CovarianceMatrix'` (see the caption). Here $n$ is the chosen order of the system, and $s$ is the number of block rows, set to the same values as for `slmoen4` (see, e.g., (Sima *et al.*, 2004)). Setting `'Cov'` = `'None'` avoids the calculation of the covariance information, reducing the execution times.

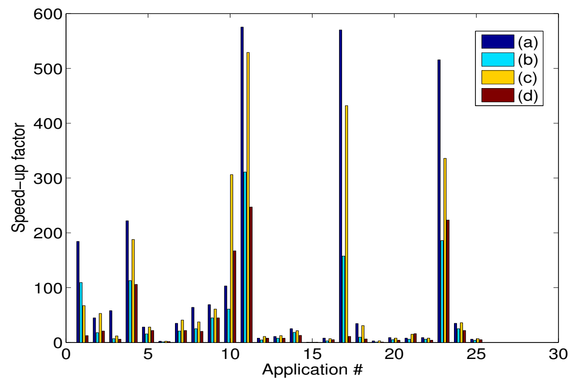

Fig. 1. `slmoen4` with fast QR versus MATLAB 7.0.4 `n4sid` with QR factorization and default options, but `order` $= n$, `'N4Weight'` = `'MOESP'`, and: (a) `'Cov'` = `[ ]`, `'N4H'` = `'Auto'`; (b) `'Cov'` = `'None'`, `'N4H'` = `'Auto'`; (c) `'Cov'` = `[ ]`, `'N4H'` $= [sss]$; (d) `'Cov'` = `'None'`, `'N4H'` $= [sss]$.

Specifically, Figure 1 shows the ratios between the execution times of `n4sid` and `slmoen4`—the speed-up factors—for each application and each case (a)-(d). Clearly, the use of a fast algorithm is very advantageous. The figure also illustrates the strong influence the `n4sid` options can have on the execution times. It should be mentioned that `n4sid` computes the covariance matrices of the estimated systems in the cases (a) and (c), while SLICOT function does not compute them. These results are shown since `'Cov'` = `[ ]` is the default value for the corresponding `n4sid` parameter. Similarly, `'Auto'` (as in cases (a) and (b)) is the default value for the parameter `'N4Horizon'`. But for large systems, the use of these default values involve significant computational effort. The largest speed-up factors have been usually obtained in case (a), normally followed by (c). Even if `'Cov'` = `'None'` (and `'N4H'` $= [sss]$, as for `slmoen4`), the SLICOT function can be much more efficient than `n4sid`, see the case (b) (and the case (d)) in Fig. 1. Moreover, the timing results for `slmoen4` (and other SLICOT functions) with `alg = 1` or `alg = 2` are essentially independent on the number of data samples $\bar{N}$, used for identification (see (Sima, 2004)). This is due to the use of fast factorization algorithms. On the other hand, the computing time for `n4sid`, which uses standard QR factorization of the block-Hankel-block matrix $\mathbf{H}$, grows linearly with $\bar{N}$, for large $\bar{N}$.

## 3. THE MODEL AND CONTROLLER REDUCTION TOOLBOX

SLICOT provides a wide variety of model (order) reduction techniques for LTI systems of the form (1) or (2). Model reduction is often used in system analysis and is mostly absolutely necessary for the computation of controllers. The reason is that modern and robust controllers like those based on LQG, $H_2$, or $H_\infty$ design techniques are themselves LTI systems of the form (1) or (2) with state dimension $N \geq n$, while in practice, only very low values $N$ should be used. The reduction of $n$ can be achieved using model reduction techniques, while $N$ can be directly reduced using *controller reduction techniques*, see, e.g., (Obinata and Anderson, 2001; Varga, 2001; Varga, 2003). In contrast to the MATLAB Control and Robust Control Toolboxes, SLICOT offers several controller reduction functions. Due to space limitations, here we will consider only continuous-time LTI systems although most SLICOT routines can be applied also to discrete-time systems.

Applying the Laplace transformation to (1) (with $\mathbf{x}(0) = \mathbf{0}$), the system dynamics are then given as $\mathbf{Y}(s) = \mathbf{G}(s)\mathbf{U}(s)$, where $\mathbf{U}, \mathbf{Y}$ are the Laplace transforms of $\mathbf{u}, \mathbf{y}$, respectively, and $\mathbf{G}$ is given in (3).

The aim of model reduction is to find an LTI system,

$$\begin{aligned} \dot{\hat{\mathbf{x}}}(t) &= \hat{\mathbf{A}}\hat{\mathbf{x}}(t) + \hat{\mathbf{B}}\mathbf{u}(t), \\ \hat{\mathbf{y}}(t) &= \hat{\mathbf{C}}\hat{\mathbf{x}}(t) + \hat{\mathbf{D}}\mathbf{u}(t), \end{aligned} \tag{8}$$

of order $r$, $r \ll n$, such that the associated TFM $\hat{\mathbf{G}}(s) = \hat{\mathbf{C}}(s\mathbf{I}_r - \hat{\mathbf{A}})^{-1}\hat{\mathbf{B}} + \hat{\mathbf{D}}$ approximates the original TFM $\mathbf{G}(s)$. This is motivated by the inequality $\|\mathbf{y} - \hat{\mathbf{y}}\|_2 \leq \|\mathbf{G} - \hat{\mathbf{G}}\|_\infty \|\mathbf{u}\|_2$, where $\|\cdot\|_2$ corresponds to the $L_2$-norm and $\|\cdot\|_\infty$ is the $H_\infty$-norm. Note that model and controller reduction for unstable systems is possible by an additive decomposition of the transfer-function into its stable and unstable parts or by a coprime factorization (where both factors are stable), see, e.g., (Varga, 2003).

Methods that attempt to minimize $\|\mathbf{G} - \hat{\mathbf{G}}\|$ are called *absolute error methods* while *relative error methods* try to minimize $\|\Delta_r\|$, where $\Delta_r$ is implicitly defined by $\mathbf{G} - \hat{\mathbf{G}} = \Delta_r \mathbf{G}$. Nevertheless, balanced truncation and related methods can be used to obtain good approximations using either one of these measures. An alternative is to use the Hankel (semi-)norm of (1) which is defined as the maximum Hankel singular value of $\mathbf{G}(s)$ for $s$ on the imaginary axis. Formulae for computing the best Hankel norm approximation to a given stable system $\mathbf{G}$ can for instance be found in (Antoulas, 2005; Obinata and Anderson, 2001) and references therein.

There is a vast variety of model reduction techniques serving different purposes; in case of linear systems, it seems that modal truncation and the related techniques of substructuring and static condensation, Padé and Padé-type approximations, and balancing-related truncation techniques play the most prominent role; see the recent monographs and surveys (Antoulas, 2005; Benner *et al.*, 2005; Benner *et al.*, 2006; Obinata and Anderson, 2001). SLICOT's model and controller reduction routines are all based on the latter approach. One reason is that SLICOT uses dense linear algebra while modal as well as Padé techniques have only advantages if large and sparse systems have to be reduced and $n$ is too large to use dense linear algebra—regarding their model reduction abilities, they are quite inferior to balancing-related techniques.

Balanced truncation is based on finding a state-space transformation which balances the controllability Gramian $\mathbf{P}$ versus the observability Gramian $\mathbf{Q}$ of the system (1). The Gramians are given as the solutions of the Lyapunov equations

$$\mathbf{AP}+\mathbf{PA}^T+\mathbf{BB}^T=0,\ \mathbf{A}^T\mathbf{Q}+\mathbf{QA}+\mathbf{C}^T\mathbf{C}=0. \quad (9)$$

A minimal and stable LTI system can be transformed by a state-space transformation (change of coordinates in state-space) such that the Gramians $\mathbf{P},\mathbf{Q}$ become equal and diagonal and the diagonal elements $\sigma_j$, $j=1,\ldots,n$, are monotonically decreasing positive real numbers, called the Hankel singular values of the LTI system (1). The reduced-order model is obtained by truncating the balanced state-space representation of the system at an order $r$ so that $\sigma_r > \sigma_{r+1}$. The so-obtained reduced-order model is stable and satisfies the error bound

$$\sigma_{r+1} \le \|\mathbf{G}-\hat{\mathbf{G}}\|_\infty \le 2\sum_{k=r+1}^{n} \sigma_k\,, \qquad (10)$$

which allows an adaptive choice of $r$ given an error tolerance. In the SLICOT toolbox, there are also several functions implementing balancing-related methods that can be used if other system properties are to be preserved (e.g., minimum-phase) or if controller reduction is desired. To overcome a disadvantage of balanced truncation of not preserving the steady-state performance, i.e., $\mathbf{G}(0) \neq \hat{\mathbf{G}}(0)$, it can be combined with singular perturbation approximation (SPA).

### *3.1 Functionality of the Toolbox*

The SLICOT Model and Controller Reduction Toolbox for MATLAB provides numerically reliable and efficient functions for balanced truncation, singular perturbation approximation, balanced stochastic truncation, frequency-weighting balancing, Hankel-norm approximation, coprime factorization, etc. The algorithms are based on methods with theoretically sound mathematical background described well in (Antoulas, 2005; Obinata and Anderson, 2001). The functionality of the toolbox includes

- order reduction for continuous-time and discrete-time LTI systems and controllers;
- order reduction for stable or unstable models/controllers;
- additive and relative error model reduction;
- frequency-weighted reduction with special stability/performance enforcing weights;
- coprime factorization-based reduction of state feedback and observer-based controllers.

The main features of the toolbox are:

- computational reliability using square-root and balancing-free accuracy enhancing techniques;

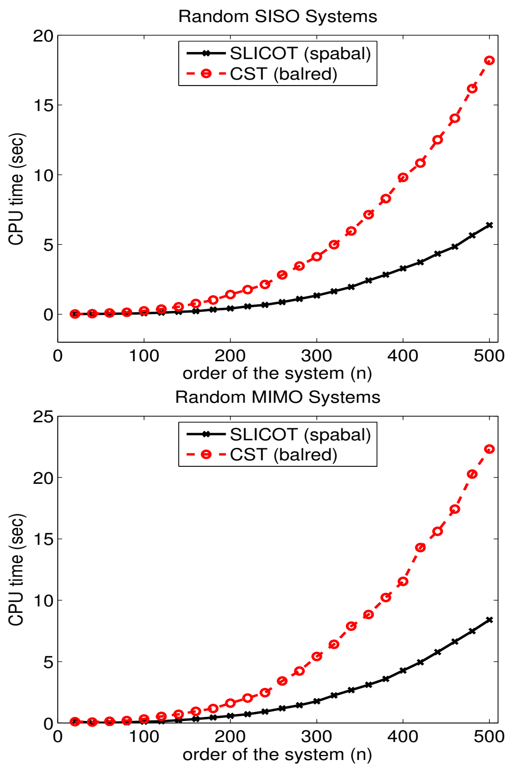

Fig. 2. Comparison of MATLAB functions for singular perturbation approximation for SISO (top) and MIMO (bottom, $m=5, \ell=10$) systems.

- high numerical efficiency, using latest developments and structure-exploiting algorithms;
- enhanced functionality, e.g, controller reduction.

For a more detailed description of the SLICOT Fortran 77 routines for model and controller reduction see (Varga, 2003).

### *3.2 Performance results*

In this section we present some typical results for functions from the SLICOT Model and Controller Reduction Toolbox. All tests are performed on a Toshiba notebook with 1.25 GB main memory, an Intel M processor at 1.1 Ghz, running MATLAB R2006a. We compare the functions for balanced truncation with matching DC gain, i.e., the balanced truncation method with singular perturbation approximation is used. This is the default in the function `balred` from the MATLAB Control System Toolbox (in the following, CST for short) and implemented as MATLAB function `spabal` in the SLICOT Model and Controller Reduction Toolbox. Note that `balancmr` from the MATLAB Robust Control Toolbox does not offer an option for matching DC gains and is thus not included in the comparison. It should be noted, though, that its performance is usually much worse than that of `balred`, see, e.g., (Benner, 2006).

We generate systems with $\mathbf{A},\mathbf{B},\mathbf{C}$ having normally distributed random entries. Figure 2 shows the CPU times needed for systems of increasing order for single-input/single-output (SISO, $m = \ell = 1$) and multi-input/multi-output (MIMO; here, $m = 5$, $\ell = 10$) systems. In contrast to `spabal`, `balred` does not offer to compute a reduced-order model satisfying a given error tolerance based on the computable error bound (10). Thus, in all cases, a reduced-order model of prescribed order $n_u + 5$ is computed, where $n_u$ is the number of unstable poles of the system. For all systems except for the smallest one tested ($n = 20$), `spabal` requires less than 40% of the CPU time of the already quite efficient routine `balred`. The accuracy of the reduced-order models is comparable for all routines and all tests.

Besides the often significant advantage in execution times shown above, the main advantage of the SLICOT Model and Controller Reduction Toolbox is the availability of frequency-weighted versions of balanced truncation methods for model and controller reduction. None of these are available in current MATLAB toolboxes. Also, as mentioned above, even for standard model reduction techniques, the SLICOT-based functions are sometimes more flexible and offer a better functionality.

## 4. CONCLUSIONS

The results show that the fast and reliable system identification and model/controller reduction algorithms and solvers included in the SLICOT toolboxes can be safely used, and they are significantly more efficient than the existing MATLAB codes.

## REFERENCES

Antoulas, A. C. (2005). *Approximation of Large-Scale Dynamical Systems*. SIAM Publications. Philadelphia, PA.

Benner, P. (2006). A MATLAB repository for model reduction based on spectral projection. In: *Proceedings of The 2006 IEEE Conference on Computer-Aided Control Systems Design (CACSD), October 4–6, 2006, Munich, Germany*, pp. 19–24. IEEE, Piscataway, NJ.

Benner, P., R. W. Freund, D. C. Sorensen, and A. Varga, Eds. (2006). *Special Issue on Order Reduction of Large-Scale Systems*. Vol. 415, issues 2–3 of *Linear Algebra and Its Applications*.

Benner, P., V. Mehrmann, and D. C. Sorensen, Eds. (2005). *Dimension Reduction of Large-Scale Systems*. Vol. 45 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin/Heidelberg, Germany.

Benner, P., V. Mehrmann, V. Sima, S. Van Huffel and A. Varga (1999). SLICOT — A subroutine library in systems and control theory. In: *Applied and Computational Control, Signals, and Circuits* (B. N. Datta, Ed.). Vol. 1, chapter 10. pp. 499–539. Birkhäuser, Boston.

Kailath, T. and A. Sayed (1995). Displacement structure: Theory and applications. *SIAM Review* **37**, 297–386.

Ljung, L. (2000). *System Identification Toolbox. For Use with* MATLAB. *User's Guide. Version 5*. The MathWorks, Inc. 3 Apple Hill Drive, Natick, MA.

Mastronardi, N., D. Kressner, V. Sima, P. Van Dooren and S. Van Huffel (2001). A fast algorithm for subspace state-space system identification via exploitation of the displacement structure. *J. Comput. Appl. Math.* **132**(1), 71–81.

Obinata, G. and B.D.O. Anderson (2001). *Model Reduction for Control System Design*. Communications and Control Engineering Series. Springer-Verlag, London, UK.

Sima, V. (2003*a*). Fast numerical algorithms for Wiener systems identification. In: *Analysis and Optimization of Differential Systems* (V. Barbu, I. Lasiecka, D. Tiba and C. Varsan, Eds.). pp. 375–386. Kluwer Academic Publishers. Boston/Dordrecht/London.

Sima, V. (2003*b*). Performance investigation of SLICOT Wiener systems identification toolbox. In: *Proceedings of The 13th IFAC Symposium on System Identification, SYSID 2003, August 27–29, 2003, Rotterdam, The Netherlands* (P.M.J. Van den Hof, B. Wahlberg and S. Weiland, Eds.). pp. 1345–1350. Omnipress.

Sima, V. (2004). Efficient data processing for subspace-based multivariable system identification. In: *Proceedings of IFAC Workshop on Adaptation and Learning in Control and Signal Processing (ALCOSP 2004), Aug. 30th – Sept. 1st, 2004, Yokohama, Japan*. pp. 871–876.

Sima, V., D. M. Sima and S. Van Huffel (2004). High-performance numerical algorithms and software for subspace-based linear multivariable system identification. *J. Comput. Appl. Math.* **170**(2), 371–397.

Van Overschee, P. and B. De Moor (1994). N4SID: Two subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica* **30**(1), 75–93.

Varga, A. (2001). Model reduction software in the SLICOT library. In: *Applied and Computational Control, Signals, and Circuits* (B.N. Datta, Ed.). Vol. 629 of *The Kluwer International Series in Engineering and Computer Science*. pp. 239–282. Kluwer Academic Publishers, Boston, MA.

Varga, A. (2003). Controller reduction using accuracy-enhancing methods. In: *Dimension Reduction of Large-Scale Systems* (P. Benner, V. Mehrmann and D. Sorensen, Eds.). Vol. 45 of *Lecture Notes in Computational Science and Engineering*. pp. 227–262. Springer-Verlag, Berlin.

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**(5), 1187–1210.