The SLICOT Toolboxes – a Survey

Peter Benner
Daniel Kressner
Vasile Sima

Abstract

SLICOT is a comprehensive numerical software package for control systems analysis and design. While based on highly performant Fortran routines, MATLAB and Scilab interfaces provide convenient access for users. In this survey, we summarize the functionality contained in the three SLICOT toolboxes for (i) basic tasks in systems and control, (ii) system identification, and (iii) model reduction. Several examples illustrate the use of these toolboxes for addressing frequent computational tasks.

Keywords: numerical methods, software, MATLAB*, matrix equations, system identification, model reduction*

I. INTRODUCTION

With the ever-increasing complexity of control systems, efficient computational methods for their analysis and design are becoming more and more important. These computational methods need to be based on reliable and robust numerical software provided by well-tested and user-friendly software libraries. This paper intents to give an overview of the MATLAB[1] toolboxes of SLICOT[2] for solving analysis and synthesis problems of modern and robust control.

P. Benner is with the Fakultät für Mathematik, TU Chemnitz, Germany (benner@mathematik.tu-chemnitz.de). D. Kressner is with the Department of Computing Science, Umeå University, Sweden (kressner@cs.umu.se) and was supported by a DFG Emmy Noether fellowship and in part by the Swedish Foundation for Strategic Research under the Frame Programme Grant A3 02:128. V. Sima is with the National Institute for Research & Development in Informatics, Romania (vsima@ici.ro).

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

[2]The SLICOT (Subroutine Library in Systems and 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`.

While the core of SLICOT consists of efficient, robust, and highly portable Fortran 77 routines, there are currently three MATLAB toolboxes providing a user-friendly interface:

1. Basic Systems and Control Toolbox
2. System Identification Toolbox
3. Model and Controller Reduction Toolbox

The MATLAB functions (M-functions) contained in these toolboxes are based on MEX gateways to the Fortran 77 routines. The MEX-functions are more difficult to use than the provided M-functions, but allow a greater flexibility. They are called by the M-functions. Executable SLICOT MEX-files are provided for MATLAB running under WINDOWS (95, 98, NT, ME, 2000, XP), Sun Solaris, and Linux.

While Section II describes the usage of the Basic Systems and Control Toolbox in some detail, Sections III and IV provide a broader description of the other two toolboxes emphasizing their advantages in solving computationally challenging problems.

II. The Basic Systems and Control Toolbox

Most of the functionality of SLICOT is concerned with *linear time-invariant (LTI) systems in state-space form.* In the continuous-time case, such a system takes the form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{1}$$

and in the discrete-time case,

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

In both cases, the system matrix A is $n \times n$, the input matrix B is $n \times m$, the output matrix C is $p \times n$, and the feedthrough (or input-output) matrix D is $p \times m$. In the following, we assume all matrices to be real.

MATLAB's Control System Toolbox provides the LTI object which allows to conveniently store and manipulate linear state-space systems. For example, a continuous-time LTI object `sys` is created by the command `sys = ss(A,B,C,D)` with the matrices A, B, C, D as in (1). The LTI object is supported by SLICOT, but it is worth mentioning that the complete functionality of SLICOT can be accessed (though in a slightly less convenient way) even if the control toolbox is not available.

An alternative way to represent LTIs is provided by the transfer function matrix (TFM), a rational matrix function obtained from (1) and (2) as

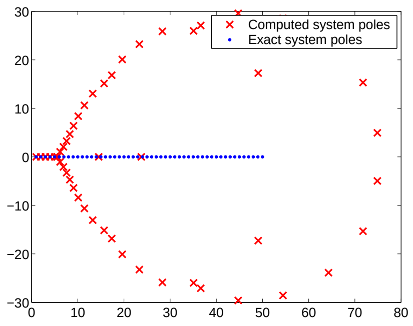
$$G(\lambda) = C(\lambda I - A)^{-1}B + D, \tag{3}$$

where λ is a variable appearing in the Laplace transform, in the continuous-time case, or the z variable appearing in the z-transform, in the discrete-time case. If $m, p \ll n$, the representation of G requires much less parameters ($O(n)$ for the coefficients of the nominator and denominator) than a state-space representation ($O(n^2)$ for the entries of the system matrix). However, SLICOT avoids the direct use of transfer functions in computations because of the severe numerical difficulties associated with this approach.

Example 1:
The following MATLAB *code converts a state-space system with diagonal system matrix having diagonal entries* $1, \ldots, 50$ *into a transfer function:*

```
A = diag(1:50); B = ones(50,1); C = ones(1,50);
sys = ss(A,B,C,0); tra = tf(sys);
```

The command `eig(tra)` *can be used to compute the poles of this transfer function (marked by the red crosses in the plot on the right). It turns out that the round-off error made during the conversion into a transfer function has severely perturbed the exact poles* $1, \ldots, 50$ *(marked by the blue dots).*



Extending the state equations in (1) and (2) to $E\dot{x}(t) = Ax(t) + Bu(t)$ and $Ex_{k+1} = Ax_k + Bu_k$, respectively, where E is an $n \times n$ matrix, leads to *linear descriptor systems*. The corresponding LTI object is created by the command `sys = dss(A,B,C,D,E)`. Several SLICOT functions are capable to deal with such generalized LTI systems.

A. *System Analysis*

This section presents a selection of exemplary SLICOT functions for analyzing intrinsic system properties.

A.1 Poles and Zeros

The function `polzer` computes the poles, zeros and normal rank of a standard or descriptor system.

Example 2: Consider a linear system with poles $-1/2, 1$, a zero $-1/2$ and normal rank 1:
`A = [4 3;-9/2 -7/2]; B = [ 1; -1 ]; C = [3 2]; D = 0; sys = ss(A,B,C,D);`
The command `[p,z,r] = polzer(sys)` *yields the following output:*

```
p =               z =               r =
    1.0000            -0.5000            1
   -0.5000
```

The function `polzer` can also be used to determine the Kronecker structure [22] of the system pencil $\left[\begin{array}{cc} A-\lambda E & B \\ C & D \end{array}\right]$ by providing additional output arguments.

A.2 Controllability and Observability

The *controllability staircase form* of a system is a reliable means to test controllability and identify uncontrollable poles. For this purpose, an orthogonal matrix U is computed such that

$$\left[\begin{array}{c|c} U^TAU & U^TB \\ \hline CU & D \end{array}\right] = \left[\begin{array}{cc|c} A_{11} & A_{12} & B_1 \\ 0 & A_{22} & 0 \\ \hline C_1 & C_2 & D \end{array}\right], \qquad (4)$$

where the subsystem $\left[\begin{array}{cc} A_{11} & B_1 \\ C_1 & D \end{array}\right]$ is controllable and A_{22} contains all uncontrollable poles. Moreover, A_{11} and A_{22} are in upper staircase form (e.g., in upper Hessenberg form for single-input systems), see [11] for more details. The original system is controllable if and only if A_{22} is void. The form (4) can be computed using the command `[syscf,Nc{,U,s}] = slconf(sys{,tol})`. Here, the LTI object `syscf` contains the system (4) and the integer `Nc` is the size of the controllable subsystem. The optional output arguments `U` and `s` provide the orthogonal transformation matrix and the block sizes of the staircase form, respectively. The optional input argument `tol` is a tolerance used to decide matrix ranks during the computation (by default `tol` $= n^2 \times$ `eps`).

Example 3: Applying `[syscf,Nc] = slconf(sys)` *with* `sys` *as in Example 2 yields the following output:*

```
a =                    b =                c =                        d =
            x1     x2                u1             x1        x2               u1
   x1        1   -7.5        x1  -1.414      y1  -0.7071     3.536        y1   0
   x2        0   -0.5        x2       0
```

```
Nc =
      1
```

This shows that the pole -0.5 is not controllable.

The *observability staircase form* of a system is given by

$$\left[\begin{array}{c|c} U^TAU & U^TB \\ \hline CU & D \end{array}\right] = \left[\begin{array}{cc|c} A_{11} & 0 & B_1 \\ A_{21} & A_{22} & B_2 \\ \hline C_1 & 0 & D \end{array}\right], \tag{5}$$

where U is orthogonal and the subsystem $\left[\begin{array}{cc} A_{11} & B_1 \\ C_1 & D \end{array}\right]$ is observable. The matrix A_{22} contains all unobservable poles. The syntax of the corresponding SLICOT function `slobsf` is analoguous to `slconf`.

By a combination of the controllability and observability staircase forms, any system can be transformed into the form

$$\left[\begin{array}{c|c} U^TAU & U^TB \\ \hline CU & D \end{array}\right] = \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & B_1 \\ 0 & A_{22} & A_{23} & B_2 \\ 0 & 0 & A_{33} & 0 \\ \hline 0 & C_2 & C_3 & D \end{array}\right]. \tag{6}$$

The subsystem $\left[\begin{array}{cc} A_{22} & B_2 \\ C_2 & D \end{array}\right]$, which is both controllable and observable, represents a minimal realization of the original system and can be computed with `slminr`.

A.3 Norms

The H_2 norm of a stable continuous-time system, computed with `slh2norm`, is given by

$$\|G\|_2 = \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} \|G(\imath\omega)\|_F^2 \,\mathrm{d}\omega},$$

where G is the TFM defined in (3) and $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. The L_2 norm (allowing unstable systems) is computed by providing an additional input flag: `slh2norm(sys,2)`. The H_∞-norm, computed with `slinorm(sys)`, is defined as

$$\|G\|_\infty = \sup_{\omega\in\mathbb{R}} \|G(\imath\omega)\|_2,$$

where $\|\cdot\|_2$ denotes the matrix 2-norm. An additional output `[ninf,fpeak] = slinorm(sys)` returns the peak frequency `fpeak` at which the gain $\|G(\imath\omega)\|_2$ attains $\|G\|_\infty$. In the discrete-time case, `slh2norm` and `slinorm` compute

$$\|G\|_2 = \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty}\|G(e^{\imath\omega})\|_F^2 \,\mathrm{d}\omega}, \qquad \|G\|_\infty = \sup_{\omega\in\mathbb{R}}\|G(e^{\imath\omega})\|_2,$$

respectively.

The Hankel norm of a stable system is computed using `slhknorm`. For an unstable system, `slhknorm` first separates the stable and unstable subsystems and returns the Hankel norm of the stable subsystem.

B. Linear Matrix Equations

The need for solving a Lyapunov equation

$$A^TX + XA = W, \tag{7}$$

where A is an $n\times n$ matrix and W is a symmetric $n\times n$ matrix, arises in several control applications, see for example Section IV. Provided that A has no eigenvalues on the imaginary axis, the solution X is unique and symmetric, and can be computed using the SLICOT function `sllyap` by typing `X = sllyap(A,W)`. Moreover, if A is stable and W is positive semi-definite then X is also positive semi-definite and henceforth admits a factorized representation $X = R^TR$. The function `slstly` can be used to directly compute the factor R from a factorized right-hand side $W = B^TB$: `R = slstly(A,B)`. Both functions provide additional input flags to solve a transposed version of (7) and to exploit additional structure in A. In this way, it is possible to solve the two Lyapunov equations

$$A(R_cR_c^T) + (R_cR_c^T)A = -BB^T, \qquad A^T(R_o^TR_o) + (R_o^TR_o)A = -C^TC$$

simultaneously with only one Schur factorization of A:

```
[Q,T] = schur(A);
Rc = Q * slstly(T,Q'*B,1,1); Ro = slstly(T,C*Q,1,0) * Q';
```

An additional output argument `[x,sep] = sllyap(A,C{,flag,trans})` returns the separation

$$\mathrm{sep}(A,-A^T) = \min_{\|X\|_F=1}\|A^TX + XA\|,$$

which provides a measure for the accuracy of the computed solution [13].

Table I summarizes all linear matrix equations that can be solved with SLICOT. See [21] for a performance comparison of these functions with other software.

TABLE I
Matrix equation solvers in SLICOT

Lyapunov equation	$A^TX + XA = W$	`[X{,sep}] = sllyap(A,W{,flag})`
	$AX + XA^T = W$	`[X{,sep}] = sllyap(A,W,flag,1)`
	$A^TR_o^TR_o + R_o^TR_oA = -C^TC$	`Ro = slstly(A,C{,flag})`
	$AR_cR_c^T + R_cR_c^TA^T = -BB^T$	`Rc = slstly(A,B,flag,1)`
Stein equation	$A^TXA - X = W$	`[X{,sep}] = slstei(A,W{,flag})`
	$AXA^T - X = W$	`[X{,sep}] = slstei(A,W,flag,1)`
	$A^T(R_o^TR_o)A - R_o^TR_o = -C^TC$	`Ro = slstst(A,C{,flag})`
	$A(R_cR_c^T)A^T - R_cR_c^T = -BB^T$	`Rc = slstst(A,B,flag,1)`
Sylvester equation	$AX + XB = C$	`X = slsylv(A,B,C{,flag})`
	$A^TX + XB^T = C$	`X = slsylv(A,B,C,flag,[1 1])`
	$AXB + X = C$	`X = sldsyl(A,B,C{,flag})`
	$A^TXB^T + X = C$	`X = sldsyl(A,B,C,flag,[1 1])`
Generalized Lyapunov equation	$A^TXE + E^TXA = W$	`[X{,sep}] = slgely(A,E,W{,flag})`
	$AXE^T + EXA^T = W$	`[X{,sep}] = slgely(A,E,W,flag,1)`
	$A^TR_o^TR_oE + E^TR_o^TR_oA = -C^TC$	`Ro = slgsly(A,E,C{,flag})`
	$AR_cR_c^TE^T + ER_cR_c^TA^T = -BB^T$	`Rc = slgsly(A,E,B,flag,1)`
Generalized Stein equation	$A^TXA - E^TXE = W$	`[X{,sep}] = slgest(A,E,W{,flag})`
	$AXA^T - EXE^T = W$	`[X{,sep}] = slgest(A,E,W,flag,1)`
	$A^TR_o^TR_oA - E^TR_o^TR_oE = -C^TC$	`Ro = slgsst(A,E,C{,flag})`
	$AR_cR_c^TA^T - ER_cR_c^TE^T = -BB^T$	`Rc = slgsst(A,E,B,flag,1)`

C. Quadratic Matrix Equations

A *continuous-time algebraic Riccati equation* takes the form

$$0 = Q + A^TX + XA - (L + XB)R^{-1}(L + XB)^T \tag{8}$$

where $Q \in \mathbb{R}^{n\times n}$ is symmetric, $R \in \mathbb{R}^{m\times m}$ is symmetric positive definite, and $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $L \in \mathbb{R}^{n\times m}$. Under rather mild assumptions (see, e.g., [15]), there exists a unique solution X that is symmetric. The command `X = slcares(A,Q,R,B,L)` computes this solution using Laub's Schur method [16]. The algorithm obtains $X = X_2X_1^{-1}$ from a basis $\left[\begin{smallmatrix} X_1 \\ X_2 \end{smallmatrix}\right]$ for the invariant subspace belonging to the stable eigenvalues of the Hamiltonian matrix associated with (8). These eigenvalues are returned in an additional output argument: `[X,ev] = slcares(A,Q,R,B,L)`. A third output argument can be specified to estimate the reciprocal condition number of the linear system $X_1^TX = X_2^T$, from which X is obtained.[3] Furthermore, a call `X = slcares(A,Q,G)` solves the following special case:

$$0 = Q + A^TX + XA - XGX.$$

[3]Another SLICOT function, `carecond`, can be used for estimating the conditioning of (8).

Similarly, the function `sldares` addresses the *discrete-time algebraic Riccati equation*

$$X = A^T XA - (L + A^T XB)(R + B^T XB)^{-1}(L + A^T XB)^T + Q, \tag{9}$$

and its special case

$$X = Q + A^T X(I + GX)^{-1}A.$$

The functions `slgcare` and `slgdare` address descriptor variants of (8) and (9), respectively. More detailed information on solving Riccati equations with SLICOT and a comparison with other software can be found in [9].

D. Structured Matrix Factorizations

SLICOT provides fast solvers for linear (least-squares) systems involving structured matrices, most notably block Toeplitz matrices:

$$T = \begin{bmatrix} T_0 & T_1 & \cdots & T_l \\ T_{-1} & T_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & T_1 \\ T_{-k} & \cdots & T_{-1} & T_0 \end{bmatrix} \tag{10}$$

with all blocks $T_{-k}, \ldots, T_{-1}, T_0, T_1, \ldots, T_l$ having the same dimension.

Given the first block column T_C of a square and symmetric positive definite block Toeplitz matrix T, the function `R = fstchol(TC)` computes an upper triangular matrix (Cholesky factor) R so that $T = R^T R$. When using `[R,X] = fstchol(TC,B)`, also the solution to the linear system $TX = B$ is computed. The function `X = fstsol(TC,B)` solves $TX = B$ directly, without requiring the storage of the factor R.

For a general block Toeplitz matrix T with first block column T_C and first block row T_R, the function `[Q,R] = fstqr(TC,TR)` computes a QR factorization $T = QR$, where Q is orthogonal and R is upper triangular. Let us assume that T has full column rank. Then `[Q,R,X{,Y}] = fstqr(TC,TR,B{,C})` additionally computes the matrix X that minimizes $\|TX - B\|_F$ and the matrix Y of minimal Frobenius norm that satisfies $T^T Y = C$. The call `[R{,X,Y}] = fstqr(TC,TR{,B,C})` omits Q and returns a banded sparse factor R if T itself is banded. The function `[X,Y] = fstlsq(TC,TR,B,C)` solves both least-squares problems directly, without requiring the storage of Q and R.

All functions mentioned above are based on variants of the generalized Schur algorithm and have a computational complexity that grows only quadratically as the matrix dimensions increase (in contrast to the cubic growth of general-purpose solvers), see [14] for more details. The function `fstmul(TC,TR,B)` computes the matrix product TB very efficiently via fast Hartley transforms.

E. Benchmark Collections

SLICOT provides five collections containing several academic and practical benchmark examples that allow an evaluation of the performance of a method as well as the implementation with respect to correctness, accuracy, and speed. Benchmarking gives also insight in the behavior of the method and its implementation in extreme situations, i.e., for problems where the limit of the possible accuracy is reached.

The MATLAB script `aredata` provides data for algebraic Riccati equations: 21 different examples for the continuous-time case (8) and 19 different examples for the discrete-time case (9). Some of these examples depend on parameters that allow to vary the dimension or tune certain properties, see [1], [2] for more details. The functions `ctdsx` and `dtdsx` generate benchmark examples for continuous-time and discrete-time LTI systems, respectively. The functions `ctlex` and `dtlex` generate benchmark examples for (generalized) continuous-time and discrete-time Lyapunov equations, respectively.

III. THE SYSTEM IDENTIFICATION TOOLBOX

The SLICOT System Identification Toolbox includes MATLAB and Fortran tools for linear and Wiener-type, time-invariant discrete-time multivariable systems. Subspace-based approaches MOESP (Multivariable Output-Error state SPace identification), N4SID (Numerical algorithms for Subspace State Space System IDentification), and their combination, are used to identify linear systems, and to initialize the parameters of the linear part of a Wiener system. All parameters of a Wiener system (whose nonlinear part is modeled as a single layer neural network) are then estimated using a specialized Levenberg-Marquardt algorithm. The main functionalities of the toolbox include:

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

Attractive features of this toolbox include:

• possible speed-up factors larger than 10 in comparison with commonly used software tools;
• standard or fast techniques for data compression (e.g., exploitation of block Hankel structure);
• fast estimation of system models of various, specified orders;
• ability to process multiple (possibly connected) data batches;
• specialized, structure-exploiting, Levenberg-Marquardt algorithm using block QR factorization with column pivoting;
• optionally, the quality of the intermediate results can be assessed by inspecting the associated condition numbers.

There are 14 M-functions for linear and Wiener systems identification. Four functions, `slmoesp`, `sln4sid`, `slmoen4`, and `slmoesm`, are method-oriented and enable to efficiently estimate models of various orders.

A. Performance Results

Extensive performance evaluation of the implemented system identification software has been performed using data sets from the DAISY collection, publicly available from the Internet site `http://www.esat.kuleuven.ac.be/sista/daisy`. Accuracy and efficiency comparisons of the SLICOT linear systems identification software and the available subspace-based techniques for 25 applications are presented in [20]. Figure 1 shows such a timing comparison [19] of the SLICOT `slmoen4` with fast QR factorization versus MATLAB 6.5.1 `n4sid` with standard QR factorization and default options, but with `order` $= n$, `'N4Weight'` = `'MOESP'`, and: (a) `'Cov'` := `'CovarianceMatrix'` = `[ ]`, `'N4H'` := `'N4Horizon'` = `'Auto'`; (b) `'Cov'` = `'None'`, `'N4H'` = `'Auto'`; (c) `'Cov'` = `[ ]`, `'N4H'` $= [s\,s\,s]$; (d) `'Cov'` = `'None'`, `'N4H'` $= [s\,s\,s]$. Here n is the chosen order of the system, and s is the number of block rows (see, e.g., [20]). Clearly, the use of fast algorithms is very advantageous. Results for Wiener system identification are presented, e.g., in [18]. Details about the algorithms and software tools related to the System Identification Toolbox are described in the SLICOT Working Notes 1998-6, 1999-3, 1999-19, 2000-4, and 2002-6, which can be found under `http://www.slicot.org/`.

IV. THE MODEL AND CONTROLLER REDUCTION TOOLBOX

Model (order) reduction is a common task within the simulation, control, and optimization of complex physical processes. SLICOT provides a wide variety of such reduction techniques

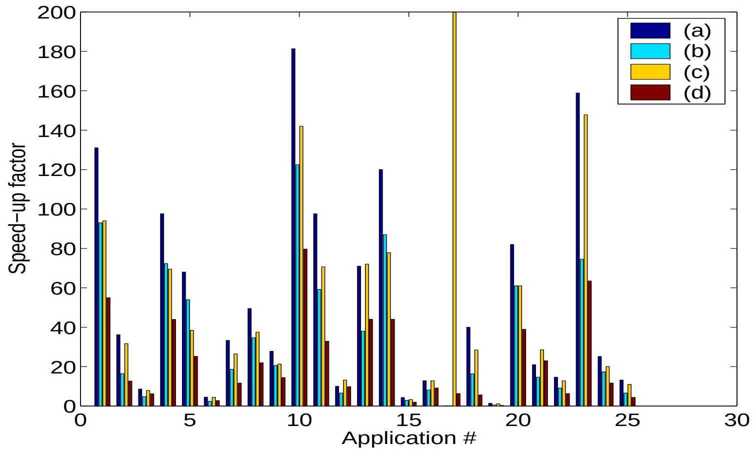


Fig. 1. `slmoen4` with fast QR versus MATLAB 6.5.1 `n4sid` with QR factorization and default options, but `order` $= n$, `'N4Weight'` = `'MOESP'`, and: (a) `'Cov'` := `'CovarianceMatrix'` = `[ ]`, `'N4H'` := `'N4Horizon'` = `'Auto'`; (b) `'Cov'` = `'None'`, `'N4H'` = `'Auto'`; (c) `'Cov'` = `[ ]`, `'N4H'` = $[s\,s\,s]$; (d) `'Cov'` = `'None'`, `'N4H'` = $[s\,s\,s]$.

for LTI systems of the form (1) or (2).

Order reduction is often an absolute necessity in the synthesis phase. The reason is that modern and robust controllers like those based on LQG, H_2, or H_∞ design techniques are themselves LTI systems of the form (1) or (2) with state-space dimension $N \geq n$. Noting that technologically, only very low-dimensions N should be used due to real-time, performance and fragility constraints, already for modest dimensions some reduction of n or N must be achieved. The reduction of the plant model, i.e., the reduction of n, can be achieved using model reduction techniques while the order N of the controller can be directly reduced using *controller reduction techniques*, see, e.g., [17], [23], [24]. In contrast to MATLAB's Control System and Robust Control Toolboxes, SLICOT offers several controller reduction functions.

To simplify the description, we focus on continuous-time systems in the following. Most of the SLICOT model and controller reduction functions can be applied to both continuous- and discrete-time LTI systems. In some cases, where a dedicated implementation for discrete-time systems is not available, the discrete-time system is first transformed to a continuous-time one using a bilinear transformation before it is reduced and then transformed back by the inverse bilinear transformation.

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

$$\begin{aligned} \dot{\hat{x}}(t) &= \hat{A}\hat{x}(t) + \hat{B}u(t), \\ \hat{y}(t) &= \hat{C}\hat{x}(t) + \hat{D}u(t), \end{aligned} \tag{11}$$

of order r, $r \ll n$, such that the associated TFM $\hat{G}(s) = \hat{C}(sI_r - \hat{A})^{-1}\hat{B} + \hat{D}$ approximates the original TFM $G(s)$. This is motivated by the relation $\|y - \hat{y}\|_2 = \|G - \hat{G}\|_\infty \|u\|_2$, where $\|\cdot\|_2$ corresponds to the L_2-norm and $\|\cdot\|_\infty$ is the H_∞-norm discussed in Section II-A.3. Note that the use of $\|\cdot\|_\infty$ requires the system to be stable. 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) and then applying methods for stable LTI systems to the resulting parts of the system that are described by stable transfer functions, see, e.g., [24].

Methods that are based on attempting to minimize $\|G - \hat{G}\|$ are called *absolute error methods* while *relative error methods* try to minimize $\|\Delta_r\|$, where Δ_r is implicitly defined by $G - \hat{G} = \Delta_r G$. For the H_∞-norm, the best approximation problem is unsolved so far. Nevertheless, balanced truncation and related methods can be used to obtain good approximations using this measure. An alternative is to use the Hankel (semi-)norm of (1) which is defined as the maximum Hankel singular value of the associated system. Formulae for computing the best approximation to a given stable system G can for instance be found in [3], [17] 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 [3], [4], [7], [8], [5], [12], [17]. 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, see, e.g., [6].

Balanced truncation is based on finding a state-space transformation which balances the controllability Gramian W_c versus the observability Gramian W_o of the system (1) given as the solutions of the Lyapunov equations

$$AW_c + W_cA^T + BB^T = 0, \qquad A^TW_o + W_oA + C^TC = 0. \tag{12}$$

The LTI system is balanced if $W_c = W_o = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$, where $\sigma_1 \geq \ldots \geq \sigma_n > 0$. The σ_j are called the Hankel singular values of the LTI system (1) and are system invariants. 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 reduced-order model has several desirable properties: stability is preserved and there exists a computable error bound

$$\sigma_{r+1} \leq \|G - \hat{G}\|_\infty \leq 2 \sum_{k=r+1}^{n} \sigma_k \tag{13}$$

that allows an adaptive choice of r given an error tolerance. In the SLICOT Model and Controller Reduction 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 the aim of the computation.

A disadvantage of balanced truncation is that it does not preserve steady-state performance (DC gain), i.e., $G(0) \neq \hat{G}(0)$. This can be overcome by combining balanced truncation with singular perturbation approximation (SPA, also called static condensation or balanced residualization). Most balancing-related methods can also be combined with SPA.

A. Functionality of the Toolbox

The SLICOT Model and Controller Reduction Toolbox employs theoretically sound and numerically reliable and efficient techniques, including balanced truncation, singular perturbation approximation, balanced stochastic truncation, frequency-weighting balancing, Hankel-norm approximation, coprime factorization, etc. For a detailed description of the SLICOT Fortran 77 routines for model and controller reduction see [24]. The mathematical background of most of the provided techniques is described well in [3], [17]. The toolbox includes

- order reduction for continuous-time and discrete-time LTI systems and controllers;
- order reduction for stable or unstable models/controllers;
- additive error model reduction;
- relative error model and controller 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;
- high numerical efficiency, using latest algorithmic developments and structure-exploiting algorithms;

TABLE II
M-functions in the SLICOT Model and Controller Reduction Toolbox.

M-Function	Functionality
`bta`	balanced truncation for the stable part (square-root balancing-free approach)
`btabal`	balanced truncation model reduction for the stable part (square-root approach)
`bta_cf`	balanced truncation model reduction of the coprime factors (square-root balancing-free approach)
`btabal_cf`	balanced truncation model reduction of the coprime factors (square-root approach)
`spa`	singular perturbation approximation based model reduction for the stable part (square-root balancing-free approach)
`spabal`	singular perturbation approximation based model reduction for the stable part (square-root approach)
`spa_cf`	singular perturbation approximation based model reduction of the coprime factors (square-root balancing-free approach)
`spabal_cf`	singular perturbation approximation based model reduction of the coprime factors (square-root approach)
`hna`	Hankel norm approximation based model reduction for the stable part (square-root approach)
`bst`	balanced stochastic truncation based model reduction
`fwbred`	frequency-weighted balancing related model reduction
`fwhna`	frequency-weighted Hankel-norm approximation
`fwbconred`	frequency-weighted balancing related controller reduction
`sfconred`	coprime factorization based state feedback controller reduction

- linear algebra based on LAPACK and the BLAS dictating the efficiency of the functions;
- flexibility and ease-of-use;
- enhanced functionality, e.g, for controller reduction;
- standardized interfaces.

Table II contains the list of implemented M-functions for model and controller reduction.

B. 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 Samsung M50 notebook with 1 GB main memory, an Intel M processor at 2.13 Ghz, running MATLAB R2006a.

In the first set of experiments we compare `bta` with the corresponding functions `balred` from the MATLAB Control System Toolbox and `balancmr` from the MATLAB Robust Control Toolbox. We generate systems with A, B, C having normally distributed random entries where A is modified to $A \leftarrow A - \mu I$ so that A is stable with stability margin roughly

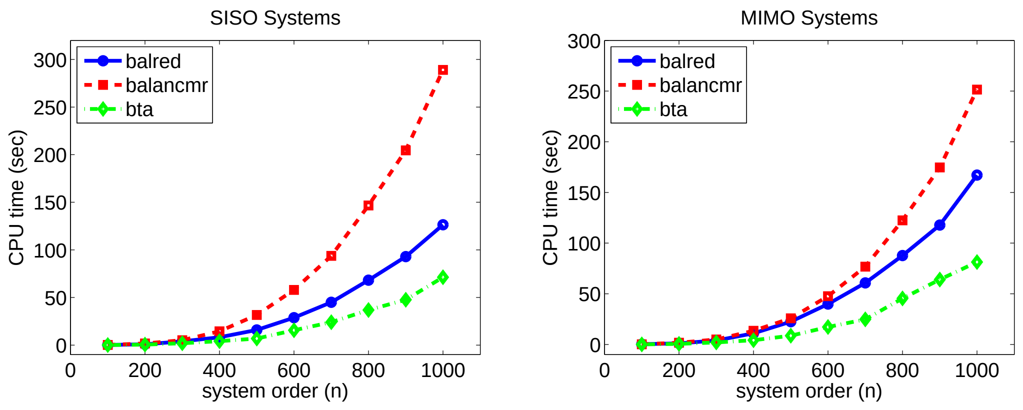


Fig. 2. Comparison of MATLAB functions for balanced truncation for SISO systems (left) and MIMO (right, $m = p = 10$) systems.

equal to 0.01. Figure 2 shows the CPU times needed for systems of increasing order for single-input/single-output (SISO, $m = p = 1$) and multi-input/multi-output (MIMO; here, $m = p = 10$) systems. In all cases, a reduced-order model of order $r = 20$ is computed. Obviously, `bta` is the fastest function in all cases, speeding up computations by about 40% compared to the already quite efficient routine `balred` (which improves significantly on previous MATLAB versions of balanced truncation like `balancmr`). Note that the accuracy of the reduced-order models is comparable for all routines and all tests.

Due to space limitations, we only present one further experiment, comparing the routines for balanced stochastic truncation `bst` from SLICOT and `bstmr` from the MATLAB Robust Control Toolbox. (Balanced stochastic truncation is not available in the MATLAB Control System Toolbox.) In balanced stochastic truncation, the observability Gramian used in balanced truncation is replaced by the solution of an associated algebraic Riccati equation. Balanced stochastic truncation belongs to the class of relative-error methods and preserves stability as well as minimum-phase. The left plot in Figure 3 shows the performance of both methods for stable SISO systems of increasing order. Again, the SLICOT function is significantly faster than the corresponding function from the MATLAB toolbox.

The right plot in Figure 3 gives an example for the accuracy that can be obtained by balanced stochastic truncation. Here, we use an example from the SLICOT model reduction collection [10]: ISS. The data describe a component of the international space station. Here, $n = 270$, $m = p = 3$. Shown is a part of the Bode plot for the errors $G(s) - \hat{G}(s)$: the magnitude of the error in the transfer function from the third input to the first output. (For

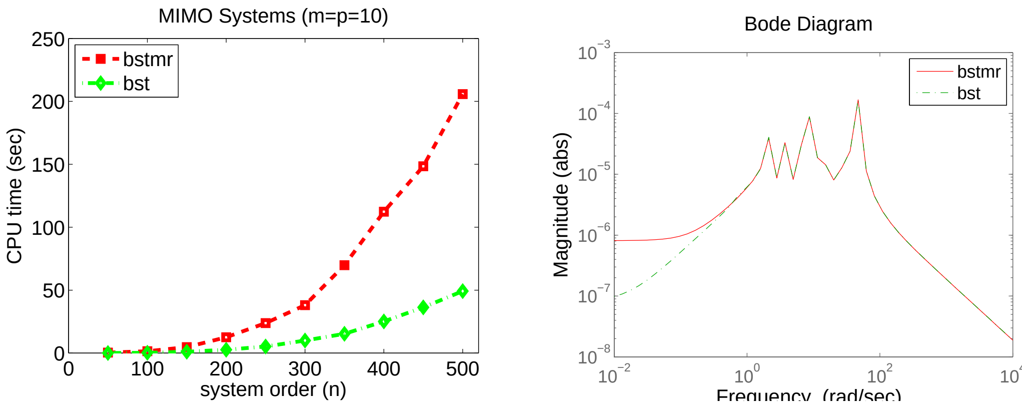


Fig. 3. Comparison of Matlab functions for balanced stochastic truncation: timings (left) and accuracy (right).

all other input-output-channels, there is no significant difference in the results.) Here, `bst` performs slightly better for lower frequencies.

It should be stressed that despite the significant advantages displayed in this section, 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. Thus, the SLICOT Toolbox provides a much more comprehensive functionality than available in Matlab so far.

References

[1] J. Abels and P. Benner. CAREX - a collection of benchmark examples for continuous-time algebraic Riccati equations (version 2.0). SLICOT working note 1999-14, WGS, 1999.
[2] J. Abels and P. Benner. DAREX - a collection of benchmark examples for discrete-time algebraic Riccati equations (version 2.0). SLICOT working note 1999-16, WGS, 1999.
[3] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM Publications, Philadelphia, PA, 2005.
[4] A. C. Antoulas, D. C. Sorensen, and S. Gugercin. A survey of model reduction methods for large-scale systems. *Contemp. Math.*, 280:193–219, 2001.
[5] Z. Bai. Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems. *Appl. Numer. Math.*, 43(1-2):9–44, 2002.
[6] P. Benner. Numerical linear algebra for model reduction in control and simulation. *GAMM Mitteilungen*, 29:275–296, 2006.
[7] P. Benner, R. W. Freund, D. C. Sorensen, and A. Varga, editors. *Special Issue on Order Reduction of Large-Scale Systems*, volume 415, issues 2–3 of *Linear Algebra and Its Applications*. June 2006.
[8] P. Benner, V. Mehrmann, and D. C. Sorensen, editors. *Dimension Reduction of Large-Scale Sys-*

tems, volume 45 of *Lecture Notes in Computational Science and Engineering.* Springer-Verlag, Berlin/Heidelberg, Germany, 2005.

[9] P. Benner and V. Sima. Solving algebraic Riccati equations with SLICOT. In *Proceedings of the 11th Mediterranean Conference on Control & Automation MED'03, June 18-20, 2003, Rhodes*, 2003.

[10] Y. Chahlaoui and P. Van Dooren. A collection of benchmark examples for model reduction of linear time invariant dynamical systems. SLICOT working note 2002-2, WGS, 2002.

[11] B. N. Datta. *Numerical Methods for Linear Control Systems Design and Analysis.* Elsevier Academic Press, 2003.

[12] R. W. Freund. Model reduction methods based on Krylov subspaces. *Acta Numer.*, 12:267–319, 2003.

[13] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, PA, second edition, 2002.

[14] D. Kressner and P. Van Dooren. Factorizations and linear system solvers for matrices with Toeplitz structure. SLICOT working note 2000-2, WGS, June 2000. Updated June 2001.

[15] P. Lancaster and L. Rodman. *The Algebraic Riccati Equation.* Oxford University Press, Oxford, 1995.

[16] A. J. Laub. A Schur method for solving algebraic Riccati equations. *IEEE Trans. Automat. Control*, AC-24:913–921, 1979.

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

[18] V. Sima. Performance investigation of SLICOT Wiener systems identification toolbox. In P. Van den Hof, B. Wahlberg, and S. Weiland, editors, *Proceedings of The 13th IFAC Symposium on System Identification, SYSID 2003, August 27–29, 2003, Rotterdam, The Netherlands*, pages 1345–1350. Omnipress, 2003.

[19] V. Sima. Computational experience with subspace identification tools. In *12th Mediterranean Conference on Control and Automation MED'04, June 6–9 2004, Kusadasi, Aydin / Turkey*, 2004.

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

[21] M. Slowik, P. Benner, and V. Sima. Evaluation of the linear matrix equation solvers in SLICOT, 2006. To appear in *Journal of Numerical Analysis, Industrial and Applied Mathematics.*

[22] A. Varga. Computation of Kronecker-like forms of a system pencil: applications, algorithms and software. In *Proc. of IEEE International Symposium on Computer Aided Control System Design, CACSD'96, Dearborn, MI*, pages 77–82, 1996.

[23] A. Varga. Selection of software for controller reduction. SLICOT working note 1999-18, WGS, 1999.

[24] A. Varga. Model reduction software in the SLICOT library. In B.N. Datta, editor, *Applied and Computational Control, Signals, and Circuits*, volume 629 of *The Kluwer International Series in Engineering and Computer Science*, pages 239–282. Kluwer Academic Publishers, Boston, MA, 2001.