European Society of Computational Methods in Sciences and Engineering (ESCMSE)

Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)
vol. 2, no. 1-2, 2007, pp. 11-34
ISSN 1790–8140

Evaluation of the Linear Matrix Equation Solvers in SLICOT[1]

Peter Benner[2]

Fakultät für Mathematik, TU Chemnitz,
D-09107 Chemnitz, Germany

Vasile Sima[3]

National Institute for Research & Development in Informatics,
Bd. Mareşal Al. Averescu, Nr. 8–10, 011455 Bucharest, Romania

Martin Slowik

Institut für Mathematik, MA 4-5, TU Berlin,
Straße des 17. Juni 136, D-10623 Berlin, Germany

Received 18 December, 2005; accepted in revised form 23 February, 2007

Abstract: We discuss solvers for Sylvester, Lyapunov, and Stein equations that are available in the SLICOT Library (**S**ubroutine **L**ibrary **I**n **CO**ntrol **T**heory). These solvers offer improved efficiency, reliability, and functionality compared to corresponding solvers in other computer-aided control system design packages. The performance of the SLICOT solvers is compared with the corresponding MATLAB solvers. This note can also serve as a guide to the SLICOT and SLICOT-based MATLAB solvers for Linear Matrix Equations.

Keywords: Computer-aided control system design, linear matrix equations, numerical algorithms, numerical linear algebra, Lyapunov equations, Sylvester equations

Mathematics Subject Classification: 93B40 Computational methods, 65F30 Other matrix algorithms

1 Introduction

Systems and control algorithms are widely used to model, simulate, and/or optimize industrial, economical, and biological processes. Systems analysis and design procedures often require the solution of general or special linear or quadratic matrix equations. Many high-level algorithms are based on these low-level kernels. For example, linear-quadratic optimal control, Kalman filtering, and H_2/H_∞-controller design procedures rely on the efficient solution of algebraic Riccati equations (AREs) [5, 9, 26, 30] while many observer design and stabilization procedures as well as Newton's method for AREs are based on the solution of linear matrix equations. In particular, this is also true

[1]Published electronically April 14, 2007
[2]E-mail: benner@mathematik.tu-chemnitz.de
[3]E-mail: vsima@ici.ro

for model reduction methods based on balancing system Gramians (see [2, 7, 28] and the references therein); as these Gramians are given as solutions of Lyapunov or related matrix equations, the applicability of these methods to large-scale problems mainly depends on the efficient solvability of linear matrix equations. There is a huge amount of theoretical results available both in systems and control, as well as in the linear algebra literature devoted to matrix equations and related topics. There are also a lot of associated software implementations, both commercial (e.g., in MATLAB[4] [24, 25]), copyrighted freeware (e.g., in the SLICOT Library [6, 27, 33], for academic use), or in the public domain (e.g., in Scilab [17]). The reliability, efficiency, and functionality of various solvers differ significantly from package to package.

This paper presents several solvers for linear matrix equations available in the SLICOT Library (**S**ubroutine **L**ibrary **I**n **CO**ntrol **T**heory), that provides Fortran 77 implementations of many numerical algorithms in systems and control theory, as well as standardized interfaces (gateways) to MATLAB and Scilab. Built around a nucleus of basic numerical linear algebra subroutines from the state-of-the-art software packages LAPACK [1], BLAS [10, 11, 23], and their counterparts for distributed memory computers, e.g., ScaLAPACK [8] and PBLAS, this library enables to exploit the potential of modern high-performance computer architectures.

We also present some performance improvements (concerning efficiency, reliability, and accuracy) offered by the SLICOT tools, in comparison with equivalent computations performed by the MATLAB functions included in the MATLAB Control System Toolbox. The results show that, at comparable or better accuracy, SLICOT computations are several times faster than MATLAB computations; moreover, the underlying problem structure is often better exploited. Note that from Release 14, MATLAB's linear matrix equation solvers are based on the SLICOT codes presented in this paper. Therefore this paper can also be seen as a performance analysis of the improved Control Toolbox functions. Still, SLICOT routines provide a larger functionality with respect to the generality of equations that can be solved and the evaluation of sensitivity and errors in the computed solutions than what has been included in the MATLAB toolbox. We will highlight these features in the next two sections. In Section 4, we describe the examples used for testing the linear matrix equation solvers. The evaluation of the numerical performance of the SLICOT linear equation solvers, given in Section 5, constitutes a major part of this paper.

2 Sylvester and Lyapunov Equations

Sylvester and Lyapunov equations are linear matrix equations. In a general setting, these equations can be defined as follows, where the notation $\mathrm{op}(M)$ denotes either the matrix M, or its transpose, M^T. Moreover, A, B, $\mathrm{op}(D)$, E, and F, are $n \times n$, $m \times m$, $m \times n$, $n \times n$, and $m \times m$ given matrices, respectively, C, G, and H are given matrices of appropriate dimensions, X and Y are unknown matrices of appropriate dimensions, and σ is a scaling factor, usually equal to one, but possibly set less than one, in order to prevent overflow in the solution matrix.

- Continuous-time and discrete-time Sylvester equations:

$$\mathrm{op}(A)\, X \pm X\, \mathrm{op}(B) = \sigma C, \tag{1}$$

$$\mathrm{op}(A)\, X\, \mathrm{op}(B) \pm X = \sigma C. \tag{2}$$

- Continuous-time and discrete-time[5] Lyapunov equations:

$$\mathrm{op}(A)^T X + X\, \mathrm{op}(A) = \sigma C, \tag{3}$$

$$\mathrm{op}(A)^T X\, \mathrm{op}(A) - X = \sigma C. \tag{4}$$

[4] MATLAB is a registered trademark of The MathWorks, Inc.

[5] The discrete-time Lyapunov equation is also called Stein equation.

- Stable non-negative definite continuous-time and discrete-time Lyapunov equations:

$$\begin{aligned} \mathrm{op}(A)^T X + X\,\mathrm{op}(A) &= -\sigma^2\,\mathrm{op}(D)^T\,\mathrm{op}(D)\,, \quad (5)\\ \mathrm{op}(A)^T X\,\mathrm{op}(A) - X &= -\sigma^2\,\mathrm{op}(D)^T\,\mathrm{op}(D)\,. \quad (6) \end{aligned}$$

- The generalized Sylvester equation:

$$AX - YB = \sigma G, \quad EX - YF = \sigma H, \tag{7}$$

  or the dual equation

$$A^T X + E^T Y = \sigma G, \quad XB^T + YF^T = -\sigma H. \tag{8}$$

- Generalized continuous-time and discrete-time Lyapunov equations:

$$\begin{aligned} \mathrm{op}(A)^T X\,\mathrm{op}(E) + \mathrm{op}(E)^T X\,\mathrm{op}(A) &= \sigma C, \quad (9)\\ \mathrm{op}(A)^T X\,\mathrm{op}(A) - \mathrm{op}(E)^T X\,\mathrm{op}(E) &= \sigma C. \quad (10) \end{aligned}$$

- Generalized stable continuous-time and discrete-time Lyapunov equations:

$$\begin{aligned} \mathrm{op}(A)^T X\,\mathrm{op}(E) + \mathrm{op}(E)^T X\,\mathrm{op}(A) &= -\sigma^2\,\mathrm{op}(D)^T\,\mathrm{op}(D)\,, \quad (11)\\ \mathrm{op}(A)^T X\,\mathrm{op}(A) - \mathrm{op}(E)^T X\,\mathrm{op}(E) &= -\sigma^2\,\mathrm{op}(D)^T\,\mathrm{op}(D)\,. \quad (12) \end{aligned}$$

The term "stable" in (5)–(6), (11)–(12) means that all eigenvalues of the matrix A (or of the matrix pencil $A - \lambda E$, for generalized stable Lyapunov equations) must have negative real parts, in the continuous-time case, or moduli less than one, in the discrete-time case. The solvers for stable Lyapunov equations compute the Cholesky factor U of the solution matrix X, i.e., $X = \mathrm{op}(U)^T\,\mathrm{op}(U)$, directly. Whenever feasible, the use of the stable solvers instead of the general ones is to be preferred, for several reasons, including the following

- the matrix product $\mathrm{op}(D)^T\,\mathrm{op}(D)$ need not be computed;
- definiteness of X is guaranteed.

Moreover, often the Cholesky factors themselves are actually needed, e.g., for model reduction based on balancing or for computing the Hankel singular values of a linear time-invariant system, see Subsection 3.1 below.

Let $\mathcal{E}(\mathcal{D},\mathcal{U}) = \mathcal{R}$ be a shorthand notation for any of the above equations, where $\mathcal{E}$, $\mathcal{D}$, $\mathcal{U}$, and $\mathcal{R}$ denote the corresponding equation formula, data, unknowns, and right hand side term, respectively. For general matrices, the solution is obtained by a *transformation method* (see, e.g., [30, page 144]). Specifically, the data $\mathcal{D}$ are transformed to some simpler forms, $\widetilde{\mathcal{D}}$ (usually corresponding to the real Schur form (RSF) of A, or generalized RSF of a matrix pair), the right hand side term is transformed accordingly to $\widetilde{\mathcal{R}}$, the *reduced equation*, $\mathcal{E}(\widetilde{\mathcal{D}},\widetilde{\mathcal{U}}) = \widetilde{\mathcal{R}}$, is solved in $\widetilde{\mathcal{U}}$, and finally, the solution of the original equation is recovered from $\widetilde{\mathcal{U}}$.

The methods implemented in SLICOT are basically the following: the Schur method (also known as Bartels–Stewart method) [4] for Sylvester equations (for A, B general, or in RSF), or Lyapunov equations (for A general, or in RSF), with the variant from [3] for the discrete-time case; the Hessenberg-Schur method [12, 16] for standard Sylvester equations, i.e., with $\mathrm{op}(M) = M$ (for A, B general, or at least one of A or B in RSF, and the other one in Hessenberg or Schur form, both either upper or lower); Hammarling's variant [18] of the Bartels–Stewart method for stable Lyapunov equations; and extensions of the above methods for generalized Sylvester [20] and

Lyapunov equations [13, 29]. The corresponding software can be considered as successor of the codes described in [14, 15].

Each of the matrix equations (1)–(12) may be re-written and solved as a system of linear algebraic equations. But this approach is inefficient for moderate or large values of n and m. For instance, the linear system corresponding to (3) or (4) has $n(n+1)/2$ equations and unknowns (exploiting the symmetry), hence its direct solution would involve about $n^6/12$ floating-point operations and $n^4/4$ memory locations. The transformation method enables to reduce these values to the order of n^3 and n^2, respectively. Assuming that the transformed matrices A, C, and the corresponding matrix X have the following form

$$A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \qquad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{12}^T & C_{22} \end{bmatrix}, \qquad X = \begin{bmatrix} X_{11} & X_{12} \\ X_{12}^T & X_{22} \end{bmatrix}, \tag{13}$$

with identical partitions, then the transformed Lyapunov equation (3), with $\mathrm{op}(M) = M$ and $\sigma = 1$, is equivalent to

$$\begin{aligned} A_{11}^T X_{11} + X_{11} A_{11} &= C_{11}, \\ A_{11}^T X_{12} + X_{12} A_{22} &= C_{12} - X_{11} A_{12}, \\ A_{22}^T X_{22} + X_{22} A_{22} &= C_{22} - A_{12}^T X_{12} - X_{12}^T A_{12}. \end{aligned}$$

The first and the third equations are again Lyapunov equations of smaller size, while the second is a Sylvester equation. This suggests a recursive solution algorithm. If A_{11} has order 1 or 2, corresponding to the first block in a real Schur form, then the first equation is equivalent to a linear system of order at most 3 (using symmetry). Its solution is substituted in the right-hand side of the second equation, which is solved in turn. Then, the third equation is solved in the same manner. The whole process can be seen as block-backsubstitution process. Note that this back-substitution process can be modified in order to obtain the Cholesky factor of the solution X of the "stable" matrix equations (5)–(6), (11)–(12) directly; see [18, 29, 34]. Many details about the numerical solution of the Lyapunov and Sylvester equations are given, e.g., in [30, section 2.3].

The ability to work with the $\mathrm{op}(\cdot)$ operator is important in many control analysis and design problems. For instance, the controllability Gramians can be defined as solutions of stable Lyapunov equations with $\mathrm{op}(A) = A^T$, while observability Gramians can be defined as solutions of stable Lyapunov equations with $\mathrm{op}(A) = A$. When both controllability and observability Gramians are needed (e.g., in model reduction computations as mentioned in the Introduction), then the same real Schur form of A can be used by a solver able to cope with $\mathrm{op}(\cdot)$, and this significantly improves the efficiency.

When solving any matrix equation, it is useful to have estimates of the *problem conditioning* and of the solution accuracy, e.g., *error bounds*. For instance, besides solving continuous-time or discrete-time Lyapunov equations, it is advisable to compute the *separation* of the matrices A and $-A^T$, or of A and A^T, respectively. The separation measures the sensitivity of the equation to perturbations in the data, and it is defined by

$$\mathrm{sep}(A^T, -A) = \min_{Z \neq 0} \frac{\|A^T Z + Z A\|}{\|Z\|} = \sigma_{\min}(P), \tag{14}$$

$$\mathrm{sepd}(A^T, A) = \min_{Z \neq 0} \frac{\|A^T Z A - Z\|}{\|Z\|} = \sigma_{\min}(P), \tag{15}$$

in the continuous-time or discrete-time case, respectively, where $\sigma_{\min}(P)$ is the minimal singular value of the matrix P, defined by

$$P = I_n \otimes A^T + A^T \otimes I_n, \tag{16}$$

$$P = A^T \otimes A^T - I_{n^2}, \tag{17}$$

respectively; the symbol $\otimes$ stands for the Kronecker product of two matrices, $X \otimes Y = (x_{ij}Y)$. Estimates of the separation quantities can be computed very efficiently.

The corresponding sensitivity measure for Sylvester equations is the separation of the matrices A and B, defined similarly as above. For generalized Sylvester equations one can optionally compute a "dif" estimate, $\mathrm{dif}[(A, E), (B, F)]$, which measures the separation of the spectrum of the matrix pair (A, E) from the spectrum of the matrix pair (B, F) [20].

Such measures, as well as condition number estimates and forward error bounds are returned by several routines of the SLICOT Library [31]. This allows to judge the reliability of the computed solution while the only way to obtain an error measure in other control packages is to compute the residual which can be misleading if the condition of the problem is large, see [19, Chapter 15]. Note that even the SLICOT-based solvers in the MATLAB Control Toolbox do not provide sensitivity or error estimates. We consider this as a significant advantage in reliability and functionality of the software available in SLICOT and in the SLICOT Basic Systems and Control Toolbox for use with MATLAB [27].

3 Solvers for Sylvester and Lyapunov Equations

Solvers for Sylvester and Lyapunov equations are available in all major control systems software packages. The specific high-level interfaces of these solvers in MATLAB and SLICOT are briefly described in the following paragraphs.

The MATLAB (Release 13) Control System Toolbox includes two solvers for continuous-time Lyapunov equations, one solver for discrete-time Lyapunov equations, and two solvers for continuous-time Sylvester equations, namely:

`lyap` / `lyap2`	solve continuous-time Lyapunov equations, if called with two arguments;
`dlyap`	solve discrete-time Lyapunov equations;
`lyap` / `lyap2`	solve continuous-time Sylvester equations, if called with three arguments.

Their usage is described in the following. The commands

```
X =  lyap(A, C);
X = dlyap(A, C);
```

compute the unique solution X of the continuous-time Lyapunov equation or of the Stein equation

$$AX + XA^T = -C, \tag{18}$$

$$AXA^T - X = -C. \tag{19}$$

Unfortunately, a closer look at the equations (3) and (18) or (4) and (19) reveals that the right hand side of the equation (18) and (19) differs in the sign as well as the transposed matrix A is used in a different way.

The command

```
X = lyap(A, B, C);
```

computes the unique solution X of the continuous-time Sylvester equation

$$AX + XB = -C. \tag{20}$$

The Sylvester equation (20) has again a different right hand side than (1).

The MATLAB command `lyap2` is used in the same way as `lyap`. This function, which uses a spectral decomposition, is fast and generally accurate, except when A and/or B have repeated eigenvalues.

To calculate the solution of the continuous-time Lyapunov equation or Sylvester equation, `lyap` (Release 13) first performs the real Schur decomposition and converts it afterwards to the complex form, while `lyap2` performs a spectral decomposition. The discrete-time Lyapunov equation is transformed to the continuous-time one.

There are no solvers for discrete-time Sylvester equations, or generalized Sylvester- and Lyapunov equations in MATLAB releases prior to Release 14.

The solvers `lyap` and `dlyap` in Release 14 are based on SLICOT routines. These functions can now be also used for solving generalized Lyapunov equations

$$AXE^T + EXA^T = -C, \tag{21}$$
$$AXA^T - EXE^T = -C, \tag{22}$$

using the commads

```
X =  lyap(A, C, [], E);
X = dlyap(A, C, [], E);
```

Moreover, the new commands, also based on SLICOT routines,

```
U =  lyapchol(A, B);
U = dlyapchol(A, B);
```

compute a Cholesky factorization $X = U^TU$ of the solution X of the Lyapunov matrix equations (18) and (19), respectively, with $C = BB^T$.

Similarly,

```
U =  lyapchol(A, B, E);
U = dlyapchol(A, B, E);
```

compute a Cholesky factorization $X = U^TU$ of the solution X of the generalized Lyapunov matrix equations (21) and (22), respectively, with $C = BB^T$.

The SLICOT Library contains 16 "user-callable" Fortran 77 routines for Sylvester and Lyapunov equations. There are routines computing estimates for condition numbers, and forward error bounds for Lyapunov equations, which enable to assess the accuracy of the results and the sensitivity of the equations to perturbations in the data. The library also includes several additional, "programmer-callable" routines. Detailed documentation of all these routines is available as HTML files at the SLICOT Web site accessible from the SLICOT hyperlink of the Web page

`http://www.slicot.org`

While the use of Fortran routines is more difficult, compared with user-friendly environments, like MATLAB, it enables to significantly increase the computational efficiency.

In order to enhance the user-friendliness of the efficient and reliable SLICOT Fortran routines, MATLAB or Scilab interfaces are provided for common control system analysis and design calculations, as shown below for Sylvester and Lyapunov equations. Two MEX-file implementations have been designed, `linmeq`, for standard linear matrix equations, and `genleq`, for generalized linear matrix equations. Another MEX-file, `arecond`, can be used to compute estimates of the reciprocal condition numbers and forward error bounds for Lyapunov and algebraic Riccati equations. These MEX-files call all SLICOT routines needed to perform the required task. The selection of the appropriate problem and solver is based on option parameters. For users' convenience, MATLAB functions are provided for each problem class. These MATLAB functions call the associated MEX-file. Executable MEX-files are built for common platforms: PC Windows 95/98/00/ME/NT (with Compaq Fortran compiler), Sun Solaris (with Fortran 95 compiler), or Linux (with g95 Fortran compiler).

The following MATLAB functions for Sylvester and Lyapunov-like equations are available:

`slsylv`	solve continuous-time Sylvester equations;
`sldsyl`	solve discrete-time Sylvester equations;
`sllyap`	solve continuous-time Lyapunov equations;
`slstei`	solve Stein equations;
`slstly`	solve stable continuous-time Lyapunov equations;
`slstst`	solve stable Stein equations;
`slgesg`	solve generalized Sylvester equations;
`slgely`	solve generalized continuous-time Lyapunov equations;
`slgest`	solve generalized Stein equations;
`slgsly`	solve stable generalized continuous-time Lyapunov equations;
`slgsst`	solve stable generalized Stein equations.

Details on the use of these MATLAB functions are given in the sequel.
The commands

```
X = slsylv(A, B, C, flag, trans, Schur);
X = sldsyl(A, B, C, flag, trans, Schur);
```

compute the unique solution X of a continuous-time Sylvester equation (1), or of a discrete-time Sylvester equation (2), respectively. The optional input parameter `flag` is a vector of length 2, which specifies the structure of A and/or B. The elements `flag(1)` and `flag(2)` refer to A and B, respectively. An input matrix is assumed to be quasi-upper triangular (or in real Schur form) if the corresponding element of `flag` is 1, and an input matrix is assumed to be upper Hessenberg if the corresponding element of `flag` is 2; otherwise, that matrix is a general matrix (default). The optional parameter `trans` specifies the operator $\mathrm{op}(\cdot)$ for the matrices A and B, as follows

$$\begin{aligned}
&\texttt{trans} = 0: \quad \mathrm{op}(A) = A; \quad \mathrm{op}(B) = B \quad \text{(default)};\\
&\texttt{trans} = 1: \quad \mathrm{op}(A) = A^T; \quad \mathrm{op}(B) = B^T;\\
&\texttt{trans} = 2: \quad \mathrm{op}(A) = A^T; \quad \mathrm{op}(B) = B;\\
&\texttt{trans} = 3: \quad \mathrm{op}(A) = A; \quad \mathrm{op}(B) = B^T.
\end{aligned} \tag{23}$$

The optional parameter `Schur` specifies the method to be used for the solution, as follows. If `Schur` = 1, the Hessenberg-Schur method is used by the solver, that is, one matrix is reduced to Hessenberg form, and the other matrix is reduced to Schur form (default); if `Schur` = 2, the Schur method is used, that is, both matrices are reduced to their Schur forms. If one or both matrices are already reduced to Schur/Hessenberg forms, this can be specified by `flag(1)` and `flag(2)`. For general matrices, the Hessenberg-Schur method is significantly more efficient than the Schur method.

The commands

```
[X, sep]  = sllyap(A, C, flag, trans);
[X, sepd] = slstei(A, C, flag, trans);
```

compute the unique symmetric solution X of a continuous-time Lyapunov equation (3) and of a discrete-time Lyapunov equation (4), respectively. If `flag` = 1, then A is assumed to be quasi upper triangular (or in RSF); otherwise, A is a general matrix (default). If `trans` = 0, then $\mathrm{op}(A) = A$ (default); otherwise, $\mathrm{op}(A) = A^T$. The optional output parameter `sep` or `sepd` returns an estimate of the separation of the matrices A and $-A^T$, defined by (14), or of the separation of the matrices A and A^T, defined by (15), respectively.

The following two commands compute the Cholesky factor U of the unique symmetric positive semi-definite solution, $\mathrm{op}(U)^T\,\mathrm{op}(U)$, of a stable continuous-time Lyapunov equation (5), or a stable discrete-time Lyapunov equation (6), respectively,

```
U = slstly(A, B, flag, trans);
U = slstst(A, B, flag, trans);
```

where `flag` and `trans` are the optional parameters defined above for Lyapunov equations.

The command

```
[X, Y, dif] = slgesg(A, E, B, F, G, H, flag, trans);
```

computes the unique solutions (X, Y) of the generalized linear matrix equation pairs (7), if `trans` $= 0$ (default), or the "transposed" equation pairs (8), if `trans` $\neq 0$. The optional input parameter `flag` is a vector with two elements, characterizing the structure of the matrix pairs. Specifically, `flag(1)` and `flag(2)` refer to the matrix pairs (A, E) and (B, F), respectively. If `flag(i)` $= 1$, the matrix pair `i` is assumed to be in a generalized Schur form; otherwise, that pair is in a general form. Default value is `flag = [0,0]`, that is, both pairs (A, E), and (B, F) are in general forms. The optional output parameter `dif` returns an estimate of the quantity $\mathrm{dif}[(A, E), (B, F)]$, which generalizes the notion of separation of two matrices.

The commands

```
[X, sep] = slgely(A, E, C, flag, trans);
[X, sep] = slgest(A, E, C, flag, trans);
```

compute the unique symmetric solution X of a generalized continuous-time Lyapunov equation (9), and a generalized Stein (discrete-time Lyapunov) equation (10), respectively. If `flag` $= 1$, it is assumed that (A, E) is in generalized Schur form; otherwise, (A, E) is in general form (default). The optional output parameter `sep` returns the separation, $\mathrm{sep}(A, E)$.

Similarly, the following two commands compute a Cholesky factor U of the unique symmetric positive semi-definite solution $\mathrm{op}(U)^T\,\mathrm{op}(U)$ of a stable generalized continuous-time Lyapunov equation (11), or of a stable generalized discrete-time Lyapunov equation (12), respectively,

```
U = slgsly(A, E, D, flag, trans);
U = slgsst(A, E, D, flag, trans);
```

Remark 1 *Starting with Release 14,* MATLAB *uses the SLICOT codes for solving (generalized) Lyapunov, (generalized) Stein, and Sylvester equations. Included are also the SLICOT solvers for the stable Lyapunov and Stein equation, but not the generalized Sylvester equation solver. Besides extended functionality, the SLICOT-based functions offer better flexibility, e.g., by directly dealing with matrices in (generalized) Schur forms, or with the op(·) operator, while the new* MATLAB *functions still cannot exploit such particular structures. These two features can give much better efficiency and are both useful in many instances, like model reduction. An example is given below. In addition, SLICOT functions can compute condition number and forward error estimates, which are not available using the* MATLAB *Control Toolbox functions.*

3.1 Application: Computation of the controllability and observability Gramians

For a stable state-space realization (A, B, C), $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, of a continuous-time linear time-invariant dynamical system, the *controllability* and *observability Gramians* are given by the solution of the stable Lyapunov equations

$$AW_c + W_cA^T = -BB^T \quad \text{and} \quad A^TW_o + W_oA = -C^TC. \tag{24}$$

Similarly, in the discrete-time case, W_c and W_o are given by the solutions of the stable Stein equations

$$AW_cA^T - W_c = -BB^T \quad \text{and} \quad A^TW_oA - W_o = -C^TC. \tag{25}$$

For a stable matrix A, both W_c and W_o are positive definite. These Gramians are important in many applications, including balanced realizations and model reduction.

The MATLAB codes cannot directly exploit the special structure of the equations (24) and (25). Two RSFs need to be computed, for A and A^T; a possibly existing RSF cannot be directly used. On the other hand, SLICOT functions can be efficiently used to obtain these Gramians. For instance, the SLICOT code below computes their Cholesky factors for the discrete-time case:

```
[A, B, C, ev, U] = systra(2, A, B, C);
Uc = slstst(A, B, 1, 1);  Uc = U*Uc;
Uo = slstst(A, C, 1, 0);  Uo = Uo*U';
```

Specifically, $W_c = U_cU_c^T$ and $W_o = U_o^TU_o$, where U_c and U_o are the matrices returned in the variables `Uc` and `Uo`, respectively.

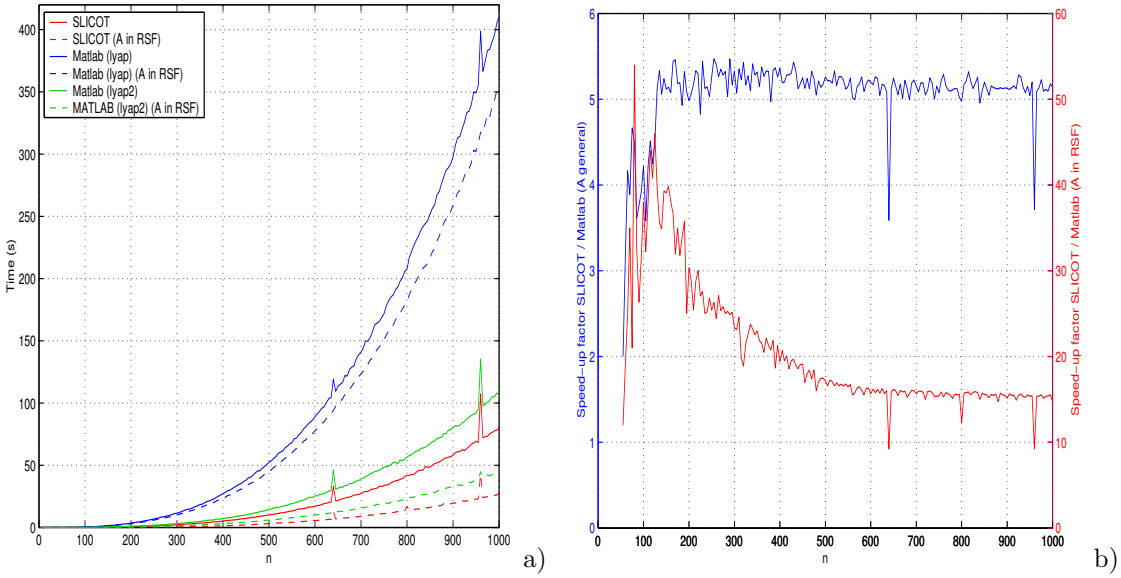


Figure 1: SLICOT `sllyap` versus MATLAB 6.5 `lyap` and `lyap2` for generated Lyapunov equations using Example 2 with the parameters $\lambda = -1.5$, $s = 1.01$. a) Timing comparison. b) Speed-up factor.

3.2 Conditioning and forward error estimates

The SLICOT-based MATLAB functions included in the SLICOT Basic Systems and Control Toolbox, but not available in MATLAB Control Toolbox, for condition and error estimation are briefly described in the following.

The commands

```
[rcnd, sep, ferr, T, U] = lyapcond(A, C, X, job, flag);
[rcnd, sep, ferr, T, U] = steicond(A, C, X, job, flag);
```

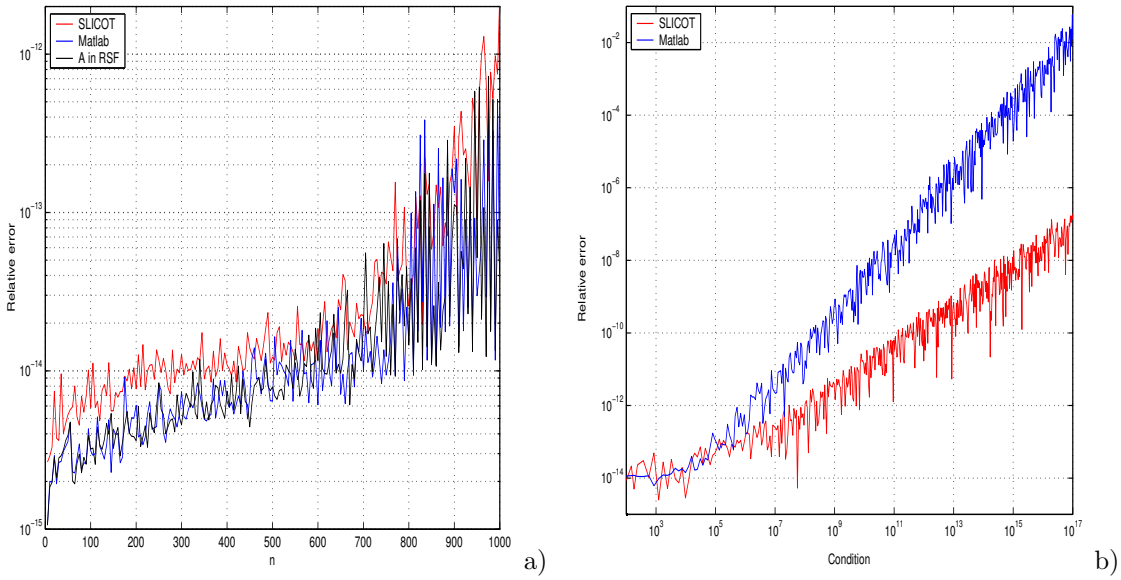


Figure 2: SLICOT `sllyap` versus MATLAB 6.5 `lyap` for generated Lyapunov equations using Example 2. a) Relative error comparison versus the dimension using the parameters $\lambda = -1.5$ and $s = 1.01$. b) Relative error versus the condition, using $\lambda = -2 \ldots -0.1$, $s = 1.5$, and $n = 10$.

compute estimates of the reciprocal of the condition number `rcnd`, separation `sep`, and forward error bound `ferr` for the continuous-time or discrete-time Lyapunov equation (3) or (4), respectively, where `X` is the (computed) solution. The optional input arguments `job` and `flag` specify the calculation to be performed. Specifically, `rcnd` and `sep` are computed if `job` $= 1$, `ferr` is computed if `job` $= 2$, and all these three values are computed if `job = 3`. If `flag(1)` $= 1$, the matrix A is assumed to be in a real Schur form T, and `C` and `X` contain the corresponding transformed matrices, i.e., $\bar{C} := U^T C U$ and $\bar{X} := U^T X U$, respectively, where U is the orthogonal matrix reducing A to T, $T = U^T A U$; otherwise, A is a general matrix. If `flag(2)` $= 0$, then $\mathrm{op}(A) = A$; otherwise, $\mathrm{op}(A) = A^T$. If `flag(3)` $= 0$, then the upper triangular part only of the matrix C should be given on input; otherwise, the lower triangular part of C should be given. Default values are `job` $= 1$ and `flag` $=$ [0,0,0]. Clearly, the output arguments `rcnd`, `sep`, and `ferr` should be specified only if requested by the value of the input argument `job`. Moreover, if `flag(1)` $= 2$ or 3, the output arguments `T` and `U`, if specified, contain the computed real Schur form T of A, and the orthogonal transformation matrix U. If `flag(1)` $= 1$ or `job` $= 1$, the condition estimator solves reduced Lyapunov equations in the iterative estimation process [31], without updating the right-hand sides and solutions. This scheme is very fast. The corresponding continuous-time or discrete-time Lyapunov equation is

$$\begin{aligned} \mathrm{op}(T)^T \bar{X} + \bar{X}\,\mathrm{op}(T) &= \bar{C}, \\ \mathrm{op}(T)^T \bar{X}\,\mathrm{op}(T) - \bar{X} &= \bar{C}, \end{aligned}$$

respectively.

4 Numerical Examples

This section briefly describes the examples which were used to test and compare speed and accuracy of the solvers for equations (1)–(12), implemented in MATLAB and SLICOT.

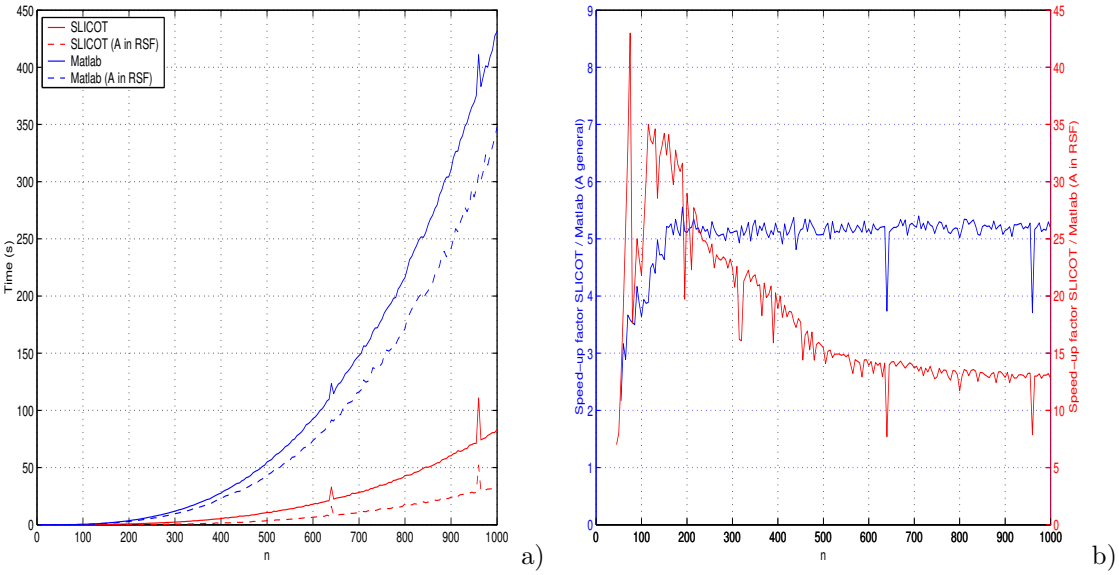


Figure 3: SLICOT `slstei` versus MATLAB 6.5 `dlyap` for generated Stein equations using Example 4 with the parameters $\lambda = -0.01$, $s = 1.005$. a) Timing comparison. b) Speed-up factor.

The examples are built in two steps, so that the solution is known explicitly. For Lyapunov equations, the auxiliary equations

$$A_0^T X_0 + X_0 A_0 = -C_0^T C_0, \tag{26}$$
$$A_0^T X_0 A_0 - X_0 = -C_0^T C_0, \tag{27}$$

are used in the first step, where A_0 is a diagonal matrix. C_0 can be a suitable row vector or a diagonal matrix. Because of this choice, the entries of the solutions can be computed easily as

$$(X_0)_{ij} = -\frac{(C_0)_i (C_0)_j}{(A_0)_{ii} + (A_0)_{jj}}, \qquad \text{for (26)}, \tag{28}$$
$$(X_0)_{ij} = -\frac{(C_0)_i (C_0)_j}{(A_0)_{ii}(A_0)_{jj} - 1}, \qquad \text{for (27)}. \tag{29}$$

In addition, another example of Lyapunov equations is chosen, where A_0 in (26) and (27) is a Jordan block matrix,

$$J = \begin{bmatrix} \lambda & 1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ & & & & \lambda \end{bmatrix}. \tag{30}$$

This matrix has only one real (multiple) eigenvalue, assuming $\lambda \in \mathbb{R}$. Again, the entries of the

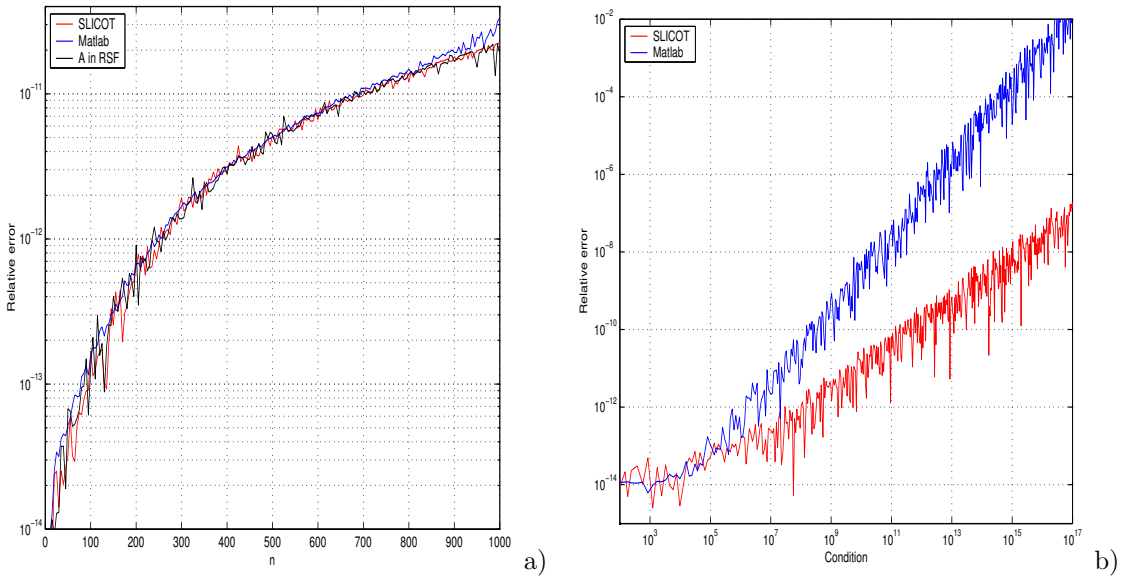


Figure 4: SLICOT `slstei` versus MATLAB 6.5 `dlyap` for generated Stein equations using Example 4. a) Relative error comparison versus the dimension using the parameters $\lambda = -0.01$ and $s = 1.005$. b) Relative error versus the condition, using $\lambda = -0.9 \ldots 0$, $s = 1.2$, and $n = 10$.

solution can be computed recursively as

$$(X_0)_{ij} = \begin{cases} -\dfrac{(C_0)_1(C_0)_1}{2\lambda}, & \text{if} \quad i = 1 \quad \text{and} \quad j = 1, \\ -\dfrac{(C_0)_1(C_0)_j + (X_0)_{1,j-1}}{2\lambda}, & \text{if} \quad i = 1 \quad \text{and} \quad j \in \{2, \ldots, n\}, \\ -\dfrac{(C_0)_i(C_0)_1 + (X_0)_{i-1,1}}{2\lambda}, & \text{if} \quad j = 1 \quad \text{and} \quad i \in \{2, \ldots, n\}, \\ -\dfrac{(C_0)_i(C_0)_j + (X_0)_{i-1,j} + (X_0)_{i,j-1}}{2\lambda}, & \text{if} \quad i, j \in \{2, \ldots, n\}, \end{cases} \tag{31}$$

for continuous-time Lyapunov equations and

$$(X_0)_{ij} = \begin{cases} -\dfrac{(C_0)_1(C_0)_1}{\lambda^2 - 1}, & \text{if} \quad i = 1 \quad \text{and} \quad j = 1, \\ -\dfrac{(C_0)_1(C_0)_j + \lambda\,(X_0)_{1,j-1}}{\lambda^2 - 1}, & \text{if} \quad i = 1 \quad \text{and} \quad j \in \{2, \ldots, n\}, \\ -\dfrac{(C_0)_i(C_0)_1 + \lambda\,(X_0)_{i-1,1}}{\lambda^2 - 1}, & \text{if} \quad j = 1 \quad \text{and} \quad i \in \{2, \ldots, n\}, \\ -\dfrac{(C_0)_i(C_0)_j + \lambda\,\left((X_0)_{i-1,j} + (X_0)_{i,j-1}\right)}{\lambda^2 - 1}, & \text{if} \quad i, j \in \{2, \ldots, n\}, \end{cases} \tag{32}$$

for discrete-time Lyapunov equations.

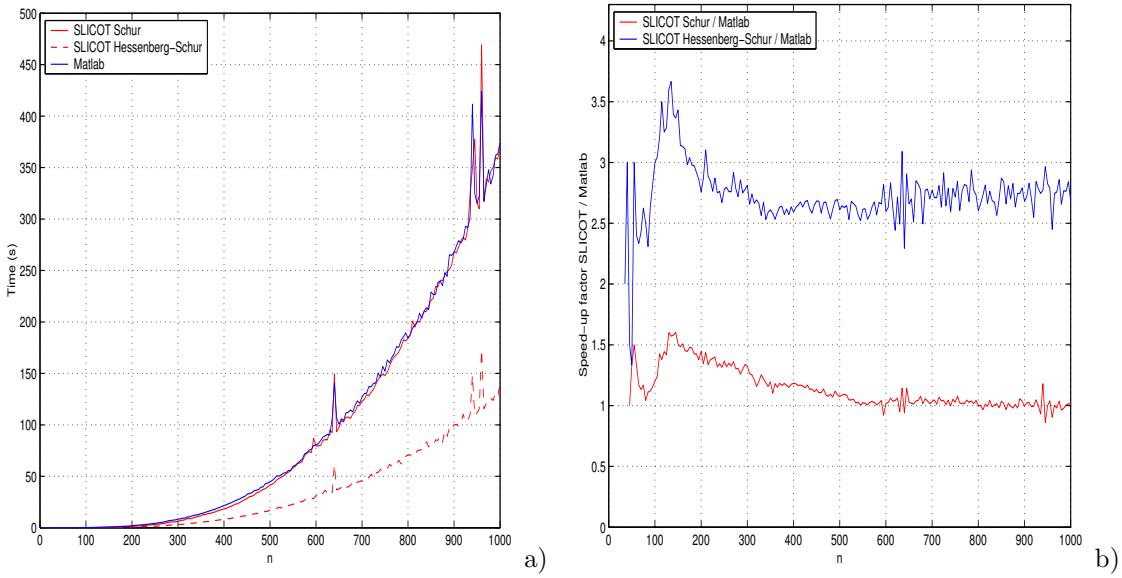


Figure 5: SLICOT `sldsyl` Hessenberg-Schur method and Schur method for generated discrete-time Sylvester equations versus MATLAB 6.5 `lyap` for the transformed continuous-time Sylvester equations using Example 8 with the parameters $a = 1.005$, $b = 1.01$, $s = 1.005$. a) Timing comparison. b) Speed-up factor.

In the same way, examples with known solution for Sylvester equations

$$A_0 X_0 + X_0 B_0 = -C_0, \tag{33}$$
$$A_0 X_0 B_0 + X_0 = -C_0, \tag{34}$$

are constructed, where A_0, B_0 are diagonal matrices. The entries of the solutions are given by

$$(X_0)_{ij} = -\frac{(C_0)_{ij}}{(A_0)_{ii} + (B_0)_{jj}}, \qquad \text{for (33)}, \tag{35}$$
$$(X_0)_{ij} = -\frac{(C_0)_{ij}}{(A_0)_{ii}(B_0)_{jj} + 1}, \qquad \text{for (34)}. \tag{36}$$

In the second step, a transformation matrix $T \in \mathbb{R}^{n \times n}$,

$$T = H_2 S H_1, \tag{37}$$

is constructed, where

$$H_1 = I_n - \frac{2}{n} e e^T, \qquad e = [1, 1, \ldots, 1]^T,$$
$$H_2 = I_n - \frac{2}{n} f f^T, \qquad f = [1, -1, \ldots, (-1)^{n-1}]^T,$$
$$S = \operatorname{diag}(1, s, \ldots, s^{n-1}), \qquad s > 1.$$

Using different values of the scalar s, it is possible to change the condition of T. The transformed matrices A, B, C, X are given by

$$A = T A_0 T^{-1}, \quad C = C_0 T^{-1}, \quad X = T^{-T} X_0 T^{-1}$$

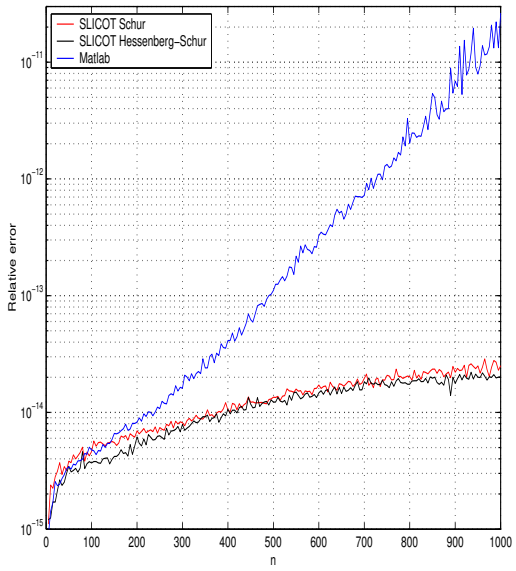


Figure 6: SLICOT `sldsyl` Hessenberg-Schur method and Schur method versus MATLAB 6.5 `lyap` for generated discrete-time and transformed continuous-time Sylvester equations using Example 8, respectively. The plot shows the relative error versus the dimension using the parameters $a = 1.005$, $b = 1.01$ and $s = 1.005$.

for the Lyapunov equations and by

$$A = T^{-T} A_0 T^T, \quad B = T B_0 T^{-1}, \quad C = T^{-T} C_0 T^{-1}, \quad X = T^{-T} X_0 T^{-1}$$

for the Sylvester equations. These transformations can be computed easily with high precision.

In order to get an example of a generalized Lyapunov equation, the following matrices are defined as in [21, 22], with $c = 2^{-t} - 1$ in the continuous-time case, and $c = 2^{-t}$ in the discrete-time case,

$$\begin{aligned} A &= cI_n + \mathrm{diag}(1, 2, \ldots, n) + U^T, \quad E = I_n + 2^{-t} U, \\ U &= \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 1 & \cdots & 1 & 0 \end{bmatrix}. \end{aligned} \tag{38}$$

Increasing the value of the parameter t tends to lower the accuracy of the numerically computed solution matrices.

For constructing examples for generalized stable Lyapunov equations and generalized Sylvester equations, the auxiliary equations

$$A_0^T X_0 E_0 + E_0^T X_0 A_0 = -C_0^T C_0, \tag{39}$$

$$A_0^T X_0 A_0 - E_0^T X_0 E_0 = -C_0^T C_0, \tag{40}$$

or

$$A_0 X_0 - Y_0 B_0 = G_0, \quad E_0 X_0 - Y_0 F_0 = H_0, \tag{41}$$

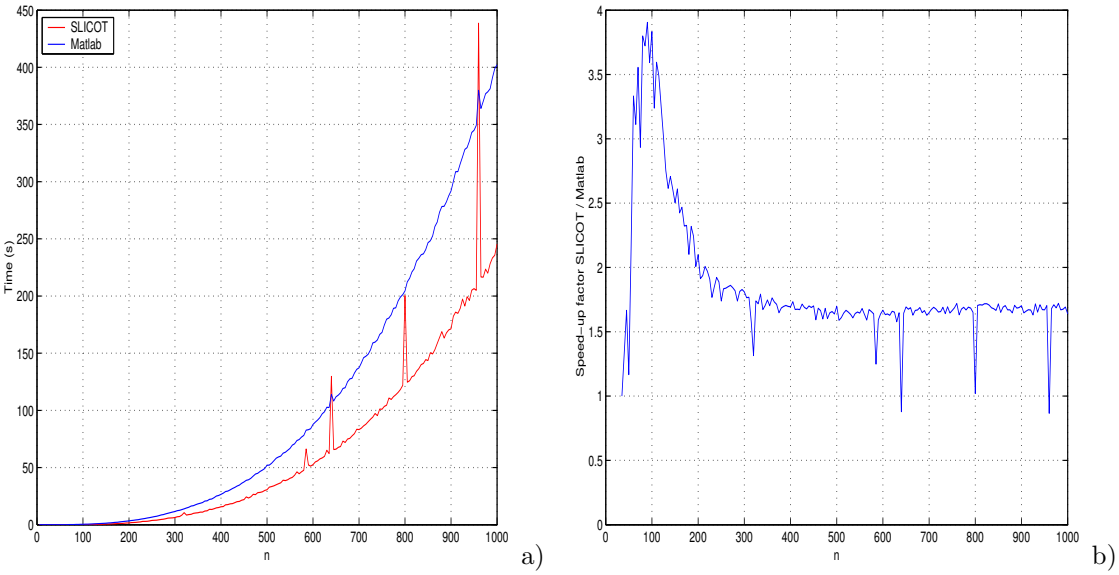


Figure 7: SLICOT `slgest` versus MATLAB 6.5 `dlyap` for generated generalized respectively transformed Stein equations using Example 10 with the parameter $t = 5$. a) Timing comparison. b) Speed-up factor. Note that the execution time which MATLAB needs for solving the generalized Stein equation does not take into account the time to obtain the transformed equations.

respectively, are used in the first step, where A_0, E_0 and B_0, F_0 are diagonal matrices. The entries of the solution can now be computed easily as

$$(X_0)_{ij} = -\frac{(C_0)_i(C_0)_j}{(A_0)_{ii}(E_0)_{jj} + (E_0)_{ii}(A_0)_{jj}}, \qquad \text{for (39)}, \tag{42}$$

$$(X_0)_{ij} = -\frac{(C_0)_i(C_0)_j}{(A_0)_{ii}(A_0)_{jj} - (E_0)_{ii}(E_0)_{jj}}, \qquad \text{for (40)}, \tag{43}$$

and

$$\begin{aligned}(X_0)_{ij} &= \frac{(F_0)_{jj}(G_0)_{ij} - (B_0)_{jj}(H_0)_{ij}}{(A_0)_{ii}(F_0)_{jj} - (E_0)_{ii}(B_0)_{jj}}, \\ (Y_0)_{ij} &= \frac{(E_0)_{ii}}{(F_0)_{jj}}(X_0)_{ij} - \frac{1}{(F_0)_{jj}}(H_0)_{ij} = \frac{(E_0)_{ii}(G_0)_{ij} - (A_0)_{ii}(H_0)_{ij}}{(A_0)_{ii}(F_0)_{jj} - (E_0)_{ii}(B_0)_{jj}},\end{aligned} \tag{44}$$

for (41). Using the transformation matrix T in (37), the matrices

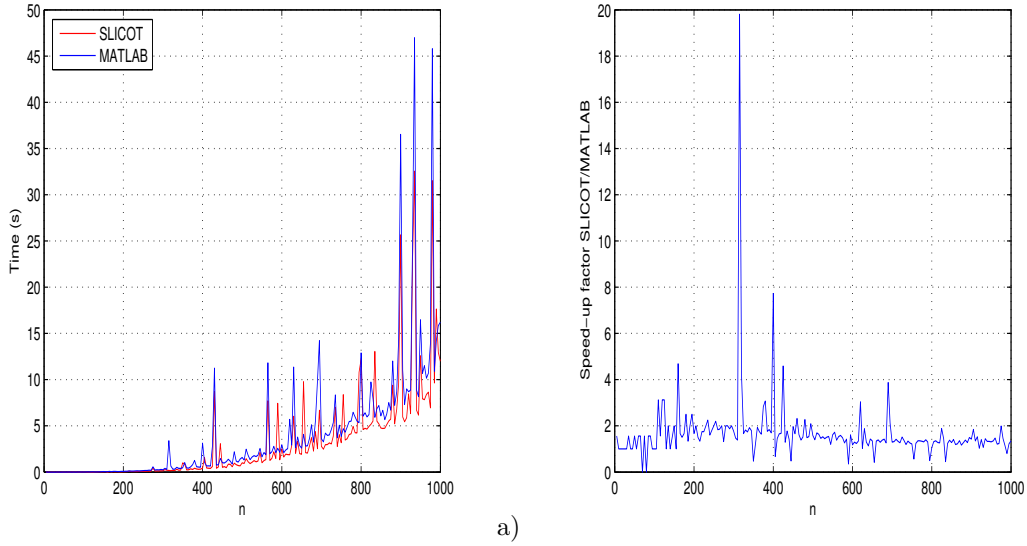
$$A = TA_0T^{-1}, \quad E = TE_0T^{-1}, \quad C = C_0T^{-1}, \quad X = T^{-T}X_0T^{-1},$$

for generalized Lyapunov equations, and

$$\begin{gathered}A = T^{-T}A_0T^T, \quad B = TB_0T^{-1}, \quad E = T^{-T}E_0T^T, \quad F = TF_0T^{-1}, \\ G = T^{-T}G_0T^{-1}, \quad H = T^{-T}H_0T^{-1}, \quad X = T^{-T}X_0T^{-1}, \quad Y = T^{-T}Y_0T^{-1},\end{gathered}$$

for generalized Sylvester equations, are constructed in the second step.

The definitions of various examples follow.



a) b)

Figure 8: SLICOT `slstst` versus MATLAB `dlyapchol` for the application from Subsection 3.1 for Example 3 with the parameters $a = 1.005$, $s = 1.005$, and $n = 5 : 5 : 1000$. a) Timing comparison. b) Speed-up factor.

Example 1 *Continuous-time Lyapunov equation with [21]*

$$\begin{aligned} A_0 &= \operatorname{diag}(-1, -a, -a^2, \ldots, -a^{n-1}), \qquad a > 1, \\ C_0 &= [\,1, 2, \ldots, n\,]. \end{aligned}$$

Example 2 *Continuous-time Lyapunov equation with*

$$\begin{aligned} A_0 &= J, \qquad \lambda < 0, \\ C_0 &= [\,1 \quad 0 \quad \ldots \quad 0\,]. \end{aligned}$$

Example 3 *Discrete-time Lyapunov equation with [22]*[6]

$$\begin{aligned} A_0 &= \operatorname{diag}(0, \frac{a-1}{a+1}, \frac{a^2-1}{a^2+1}, \ldots, \frac{a^{n-1}-1}{a^{n-1}+1}), \qquad a > 1, \\ C_0 &= [\,1, 0, \ldots, 0\,]. \end{aligned}$$

Example 4 *Discrete-time Lyapunov equation with*

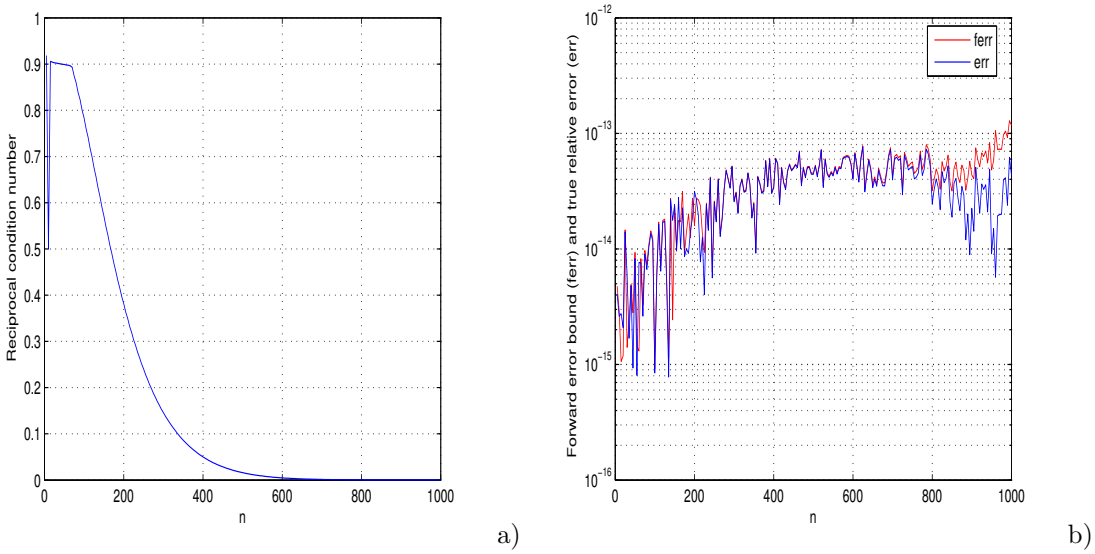
$$\begin{aligned} A_0 &= J, \qquad -1 < \lambda \leq 0, \\ C_0 &= [\,1 \quad 0 \quad \ldots \quad 0\,]. \end{aligned}$$

Example 5 *Stable continuous-time Lyapunov equation with A_0 as in Example 1 and*

$$C_0 = \operatorname{diag}[1, 2, \ldots, n].$$

Example 6 *Stable discrete-time Lyapunov equation with A_0 as in Example 3 and C_0 as in Example 5.*

[6] Note that [22] erroneously uses $C_0 = [\,1, 2, \ldots, n\,]$.



a) b)

Figure 9: SLICOT `slstst` versus MATLAB `dlyapchol` for solving the application from Subsection 3.1 for Example 3 with the parameters $a = 1.005$, $s = 1.005$, and $n = 5 : 5 : 1000$. a) Reciprocal condition numbers. b) Forward error bounds and true relative errors.

Example 7 *Continuous-time Sylvester equation with*

$$\begin{aligned} A_0 &= \mathrm{diag}(-1, -a, -a^2, \ldots, -a^{n-1}), \qquad a > 1, \\ B_0 &= \mathrm{diag}(-1, -b, -b^2, \ldots, -b^{n-1}), \qquad b > 1, \\ C_0 &= [\,1, 2, \ldots, n\,]. \end{aligned}$$

Example 8 *Discrete-time Sylvester equation with*

$$\begin{aligned} A_0 &= \mathrm{diag}(0, \frac{a-1}{a+1}, \frac{a^2-1}{a^2+1}, \ldots, \frac{a^{n-1}-1}{a^{n-1}+1}), \qquad a > 1, \\ B_0 &= \mathrm{diag}(0, \frac{b-1}{b+1}, \frac{b^2-1}{b^2+1}, \ldots, \frac{b^{n-1}-1}{b^{n-1}+1}), \qquad b > 1, \\ C_0 &= [\,1, 2, \ldots, n\,]. \end{aligned}$$

Example 9 *Generalized continuous-time Lyapunov equation with A and E defined in (38), the solution*

$$X = e\,e^T, \qquad e = [\,1 \quad 1 \quad \ldots \quad 1\,]^T,$$

and the corresponding right hand side computed as

$$C = A^T X E + E^T X A.$$

Example 10 *Generalized discrete-time Lyapunov equation with the same A, E, and X as in Example 9, and the corresponding right hand side computed as*

$$C = A^T X A - E^T X E.$$

Example 11 *Generalized stable continuous-time Lyapunov equation with A_0 as in Example 1 and*

$$\begin{aligned} E_0 &= \operatorname{diag}(1, e, e^2, \ldots, e^{n-1}), \qquad e > 1, \\ C_0 &= \operatorname{diag}[1, 2, \ldots, n]. \end{aligned}$$

Example 12 *Generalized stable discrete-time Lyapunov equation with A_0 as in Example 3 and E_0 and C_0 as in Example 11.*

Example 13 *Generalized Sylvester equation with*

$$\begin{aligned} A_0 &= \operatorname{diag}(1, a, a^2, \ldots, a^{n-1}), \qquad a > 1, \\ B_0 &= \operatorname{diag}(-1, -b, -b^2, \ldots, -b^{n-1}), \qquad b > 1, \\ E_0 &= \operatorname{diag}(-1, -e, -e^2, \ldots, -e^{n-1}), \qquad e > 1, \\ F_0 &= \operatorname{diag}(-1, -f, -f^2, \ldots, -f^{n-1}), \qquad f > 1, \\ G_0 &= -v^T v, \qquad H_0 = v^T v, \qquad v = [\,1, 2, \ldots, n\,]. \end{aligned}$$

Remark 2 *The parameters a (or λ), b, e, and f regulate the distribution of the eigenvalues of the matrices A_0, B_0, E_0, and F_0, respectively. This distribution is related to the condition of the respective Lyapunov or Sylvester equations and increasingly ill-conditioned examples can therefore be generated using the parameters. Even for small values of the parameters a and s in Examples 1 and 3, and moderate dimension n, the Lyapunov equations tend to be ill-conditioned. Furthermore, loosely speaking, the distribution of the eigenvalues of A in these examples is so widespread that it is difficult to find appropriate values for a and s avoiding a numerically indefinite solution matrix X, which precludes the calculation of the Cholesky factor of X. Note that in Examples 5 and 6 the solution X_0 is a diagonal matrix. Choosing small values for the parameter s, the eigenvalues of X_0 are approximations of the eigenvalues of the transformed matrix X.*

Remark 3 *Up to and including Release 13, no* MATLAB *solver for discrete-time Sylvester equations, or for generalized Lyapunov and Sylvester equations, was available. Since for Example 8, $\pm 1 \notin \Lambda(A), \Lambda(B)$, it is possible to transform the corresponding discrete-time Sylvester equation into the continuous-time Sylvester equation*

$$\widetilde{A}\,X + X\widetilde{B} = -\widetilde{C}, \tag{45}$$

where

$$\widetilde{A} = (A+I)^{-1}(A-I), \quad \widetilde{B} = (B+I)(B-I)^{-1}, \quad \text{and} \quad \widetilde{C} = 2(A+I)^{-1}C(B-I)^{-1}.$$

Similarly, Examples 9–13 are chosen in such a way that the generalized equations can be transformed into the standard Lyapunov and Sylvester equations. Hence, the generalized Lyapunov equations are transformed into

$$\widetilde{A}^T X + X\widetilde{A} = -\widetilde{C}, \tag{46}$$

$$\widetilde{A}^T X \widetilde{A} - X = -\widetilde{C}, \tag{47}$$

where $\widetilde{A} = A\,E^{-1}$ and $\widetilde{C} = E^{-T}\,C\,E^{-1}$, and the generalized stable Lyapunov equations are transformed into

$$\widetilde{A}^T X + X\widetilde{A} = -\widetilde{C}^T\widetilde{C}, \tag{48}$$

$$\widetilde{A}^T X \widetilde{A} - X = -\widetilde{C}^T\widetilde{C}, \tag{49}$$

where $\widetilde{A} = A\,E^{-1}$ *and* $\widetilde{C} = C\,E^{-1}$. *In order to solve the generalized Sylvester equation via* `lyap` *in* MATLAB *6.5 (Release 13), the second equation is solved for* Y *and* Y *is substituted in the first equation. Then, the following continuous-time Sylvester equation*

$$\widetilde{A}X + X\widetilde{B} = \widetilde{C}, \qquad Y = EXF^{-1} - HF^{-1}, \tag{50}$$

is obtained, where

$$\widetilde{A} = -E^{-1}\,A, \quad \widetilde{B} = F^{-1}\,B, \quad \textit{and} \quad \widetilde{C} = -E^{-1}\,G + E^{-1}\,HF^{-1}\,B.$$

Table 1: Comparative results using MATLAB 6.5

Example	SLICOT speed-up		Accuracy	Accuracy
	general	RSF		vs condition
1	2.3–1.5	20–8	comparable	comparable
2	2.0–5.0	45–15	comparable	+5 digits
3	2.4–1.6	15–8	–1 digit	+4 digits
4	2.0–5.5	43–13	comparable	+5 digits
5	3.0–1.7	–	comparable	comparable
6	3.2–1.8	–	comparable	+4 digits
7	1.5–2.0	–	comparable	–
8	3.5–2.6	–	+3 digits	–
9	3.5–1.8	–	± 1–2 digits	+9 digits
10	3.8–1.7	–	comparable	comparable
11	1.4–0.7	–	comparable	+4 digits
12	1.4–0.8	–	comparable	+7 digits
13	1.0–0.9	–	comparable	+4 digits X +5 digits Y

5 Numerical Results

This section presents typical performance results for some components of the SLICOT Library, called via the associated gateways. These results show that SLICOT routines often outperform MATLAB calculations. While the accuracy is comparable, sometimes better and sometimes slightly worse, the gain in efficiency by calling SLICOT routines can be significant. Note that the results have been obtained by timing in MATLAB the equivalent computations using the MATLAB commands

`t_begin = cputime;` *command*; `t_end = cputime - t_begin;`

Even better efficiency is to be expected by calling the SLICOT Fortran routines directly (not through gateways).

The first part of this section shows some SLICOT results in comparison with MATLAB 6.5 (R13). The second part of this section shows new SLICOT results in comparison with MATLAB 7.0.4 Release 14 Service Pack 2 (R14 SP 2).

The SLICOT–MATLAB 6.5 comparison indirectly indicates the improvements obtained in MATLAB 7 by incorporating the SLICOT routines. The calculations have been done on a PC computer with an AMD processor at 1.8 GHz, with 1 Gb memory and the relative machine precision $\epsilon = 2.22 \times 10^{-16}$, using Intel Fortran Compiler V7.0, optimized BLAS provided by MATLAB, and MATLAB 6.5 (R13). More detailed results are presented in [32].

Table 2: Parameter values used for Examples 1–13

Example	Parameters	Parameters (varying condition)
1	$a = 1.005$, $s = 1.005$	a, $s = 1.02 \cdots 4.33$
2	$\lambda = -1.5$, $s = 1.01$	$\lambda = -2 \cdots -0.1$, $s = 1.5$
3	$a = 1.005$, $s = 1.005$	a, $s = 1.005 \cdots 3.8$
4	$\lambda = -0.01$, $s = 1.005$	$\lambda = -0.9 \cdots 0$, $s = 1.2$
5	$a = 1.0129$, $s = 1.001$	$a = 5$, $s = 1.02 \cdots 4.4$
6	$a = 1.001$, $s = 1.01$	$a = 1.5$, $s = 1.01 \cdots 5.5$
7	$a = 1.03$, $b = 1.008$, $s = 1.001$	–
8	$a = 1.005$, $b = 1.01$, $s = 1.005$	–
9	$t = 5$	$t = 1.05 \cdots 46$
10	$t = 5$	$t = 1.1 \cdots 36$
11	$a = 1.005$, $e = 1.003$, $s = 1.01$	$a = 1.1$, $e = 1.01$, $s = 1.51 \cdots 7$
12	$a = 1.005$, $e = 1.003$, $s = 1.01$	$a = 1.1$, $e = 1.01$, $s = 1.51 \cdots 8$
13	$a = 1.001$, $b = 1.002$, $e = 1.003$, $f = 1.004$, $s = 1.01$	$a = 1.001$, $b = 1.002$, $e = 1.003$, $f = 1.004$, $s = 1.03 \cdots 11$

Table 1 summarizes the results using MATLAB 6.5. The calculations have been performed for $n = 5 : 5 : 1000$ (in MATLAB notation). The two, dash-separated values for the speed-up factors of SLICOT versus MATLAB functions correspond to small orders (usually for $n \leq 300$) and larger orders (usually for $n \geq 500$), respectively. The values of the speed-up factors for $300 < n < 500$ are normally in between the two specified values. The second column gives the results for general matrices, while the third column gives the results (when available) for matrices in real Schur form (RSF). The speed-up factors for Examples 7 and 8 are given for the Hessenberg-Schur method. When the Schur method is used, the SLICOT and MATLAB functions have comparable speed. An isolated dash (–) means that the corresponding test has not been tried. The last two columns in the table compare the SLICOT and MATLAB 6.5 accuracy; "$+x$ digits" means that the SLICOT functions gain up to x digits in accuracy, and similarly for "$-x$ digits". The results in the last column show the accuracy obtained for a fixed size $n = 10$ and increasing condition numbers (achieved by varying some parameters in the respective examples). Often, the MATLAB function `lyap2` has been comparable with the SLICOT functions concerning the speed, but the accuracy was sometimes significantly lower. When needed, the RSF of A and B have been computed using the SLICOT function `systra`.

It has to be mentioned that the execution time needed by the MATLAB functions for solving the transformed problems (see Remark 3), neither takes into account the time to generate the transformed equation nor the time to calculate the Cholesky factor of the solution, if needed.

For completeness, Table 2 lists the specific parameter values used in the tests.

Figure 1 shows the execution times for the SLICOT function `sllyap` and the MATLAB functions `lyap` and `lyap2` (left plot), and the speed-up factor of `sllyap` versus `lyap` (right plot), for solving Lyapunov equations generated with Example 2. The peaks at $n = 640$ and $n = 960$ are caused by cache effects. The SLICOT solver is much faster than `lyap` (up to 5 times as fast for general A and up to 45 times as fast for A in RSF), while the accuracy, shown in Figure 2 a), is comparable. The MATLAB solver `lyap2` is only about 25 % slower than `sllyap`, but `lyap2` is not useful for solving the Lyapunov equations generated with Example 2 because its relative error is bigger than 1. It is important to mention that `lyap2` will not work when the matrix A has multiple eigenvalues. Since the accuracy of `sllyap` and `lyap` is nearly the same if A is given in RSF, the result of `sllyap` is

shown only. Note that with the chosen parameters, the condition number varies between 1 and 1.5×10^6. In Figure 2 b) the comparison of the relative error versus the condition is shown using Example 2 with the parameters $\lambda = -2 \ldots -0.1$, $s = 1.5$, $n = 10$. The difference between `sllyap` and `lyap` is marginal, if the condition varies between 1 and 10^6, but SLICOT gains up to five digits in accuracy, if the condition is further increased.

Similarly, Figure 3 and Figure 4 show the efficiency and accuracy of the SLICOT discrete-time Lyapunov solver `slstei` for Example 4.

It is remarkable that the speed-up factor increases with n from 2 to 5.5 until $n = 200$, and stays at this level for higher dimensions. If A is in RSF, the speed-up factor even varies between 13 and 43. The accuracy of `slstei` is comparable and sometimes better than that of `dlyap`, see Figure 4 a), while the condition number of this Stein equation for the chosen parameters varies between 10^1 and 10^8. Since the accuracy of the SLICOT and MATLAB solvers is nearly the same if A is in RSF, the behaviour of `slstei` is only presented. Using Example 4 with the parameters $\lambda = -0.9 \ldots 0$, $s = 1.2$, $n = 10$, SLICOT gains five digits in accuracy, see Figure 4 b).

Figure 5 shows the execution times for SLICOT function `sldsyl` using the Schur and the Hessenberg-Schur method and the MATLAB function `lyap` solving the transformed equation (see Remark 3), and the speed-up factor of `sldsyl` versus `lyap` for solving discrete-time Sylvester equations generated with Example 8 using the parameters shown in Table 2. The Hessenberg-Schur method is up to 3.5 times faster than `lyap`, while the Schur method implemented in `sldsyl` is as fast as `lyap`. Figure 6 shows the relative error versus the dimension. While the accuracy of SLICOT Schur and the Hessenberg-Schur methods is comparable, `sldsyl` gains up to three digits versus `lyap` (applied to the transformed continuous-time Sylvester equation).

Figure 7 reveals the execution times for the SLICOT generalized Stein solver `slgest` and the MATLAB function `dlyap` solving the transformed equation, and the speed-up factor of `slgest` versus `dlyap`, for solving generalized Stein equations generated with Example 10 using the value $t = 5$ for the parameter. The efficiency of `slgest` is up to 3.8 times better than that of `dlyap`. Besides the usual peaks at $n = 640$ and $n = 960$, there are two additional peaks at $n = 590$ and $n = 800$. The reason is that the SLICOT generalized linear matrix equation solvers need two data matrices, A and E, while standard solvers use only one data matrix, A. The relative errors versus the dimension and versus the condition (for $t = 1.1 \ldots 36$, $n = 10$), are similar, and therefore they are not shown.

In the last part of this section, new SLICOT results or comparisons with MATLAB 7 are given. Specifically, the numerical results have been obtained on an Intel Pentium 4 computer at 3 GHz, with 1 GB RAM, with the relative machine precision $\epsilon \approx 2.22 \times 10^{-16}$, using Windows XP (Service Pack 2) operating system, the Compaq Visual Fortran V6.5 compiler and MATLAB 7.0.4.365 (R14 SP 2).

The application from Subsection 3.1 is considered for Example 3, with $B = C^T$, and parameters $a = 1.005$, $s = 1.005$. The SLICOT code shown there, involving calls to `systra` and `slstst`, has been used for computing the controllability and observability Gramians. The results for $n = 5 : 5 : 1000$ have been compared with those obtained using the corresponding MATLAB 7.0.4 code, involving calls to `dlyapchol` with argument `A` in a real Schur form (returned by `systra`). Actually, `dlyapchol` is based on the same SLICOT routines which are called by `slstst`. This can be seen by computing the relative errors of the Gramians computed with SLICOT and MATLAB, for all 200 problems solved (with $n = 5 : 5 : 1000$). Let `errc` and `erro` be the two vectors (of length 200) of computed relative errors in the controllability and observability Gramians, respectively. Their Euclidean norms resulted to be `1.5e-14` and `4.17e-17`. But contrary to `slstst`, `dlyapchol` cannot exploit the available real Schur form of A. Consequently, the MATLAB code needs more time to solve the problems.

Figure 8 shows the execution times needed by the two calls of the SLICOT function `slstst` and two calls of the corresponding SLICOT-based MATLAB function `dlyapchol` and the speed-up

factor of SLICOT versus MATLAB calls for solving the problems described above. The execution times seem to be somewhat influenced by other tasks performed by the operating system, but the general behavior is that SLICOT calls provides faster execution. The mean value of the SLICOT versus MATLAB speed-up factor is 1.66. There are only 11 problems (out of 200) for which MATLAB was slightly faster.

Figure 9 shows the reciprocal condition numbers and forward error bounds, as well as the true relative errors in the maximum norm, for the calculation of the observability Gramians for the same problems and the same parameters. The command used is

```
[rcnd, sep, ferr] = steicond(A, -C'*C, Uo'*Uo, 3);
```

where `A` is not reduced to RSF. For the chosen parameters, the problems are well-conditioned, as shown by the values of the reciprocal condition numbers and forward error bounds. The smallest reciprocal condition number is approximately `2.4e-5` (for $n = 1000$). The forward error bounds agree well with the true relative errors. Note that these calculations cannot be performed with the current functions available in the MATLAB toolboxes.

6 Conclusions

We have discussed easy-to-use solvers from the SLICOT Library for various linear matrix equations from systems and control theory. Based on Fortran 77 codes implementing state-of-the-art numerical algorithms, the high-level MATLAB or Scilab interfaces offer extended functionality, and improved efficiency and comparable reliability as compared to existing software tools. This is illustrated by numerous numerical results. Even though parts of the SLICOT solvers have been integrated into MATLAB's Release 14 version of the Control Toolbox, there are still advantages in using the original SLICOT tools as a larger class of matrix equations can be solved (generalized Sylvester equations), specialized calculations for matrices in a (generalized) real Schur form can be performed, and the sensitivity and accuracy of numerical computations can be accessed via condition and error estimators.

Acknowledgement

This work was supported in part by the DFG Research Center "Mathematics for Key Technologies" (MATHEON) in Berlin. The authors also thank the anonymous reviewer for suggestions for improving the paper.

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, LAPACK Users' Guide, third ed., Software · Environments · Tools (SIAM, Philadelphia, PA, 1999).

[2] A. Antoulas, Approximation of Large-Scale Dynamical Systems (SIAM Publications, Philadelphia, PA, 2005).

[3] A. Y. Barraud, A numerical algorithm to solve $A^T XA - X = Q$, IEEE Trans. Automat. Control **AC-22**, 883–885 (1977).

[4] R. H. Bartels and G. W. Stewart, Algorithm 432: Solution of the matrix equation $AX + XB = C$, Comm. ACM **15**, 820–826 (1972).

[5] P. Benner, Computational methods for linear-quadratic optimization, Supplemento ai Rendiconti del Circolo Matematico di Palermo, Serie II, No. 58, 21–56 (1999).

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

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

[8] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. Whaley, ScaLAPACK Users' Guide (SIAM, Philadelphia, PA, 1997).

[9] B. Datta, Numerical Methods for Linear Control Systems (Elsevier Academic Press, 2004).

[10] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms, ACM Trans. Math. Softw., **16** 1–17, 18–28 (1990).

[11] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, Algorithm 656: An extended set of Fortran Basic Linear Algebra Subprograms, ACM Trans. Math. Softw., **14** 1–17, 18–32 (1988).

[12] W. Enright, Improving the efficiency of matrix operations in the numerical solution of stiff ordinary differential equations, ACM Trans. Math. Softw. **4** 127–136 (1978).

[13] M. Epton, Methods for the solution of $AXD - BXC = E$ and its application in the numerical solution of implicit ordinary differential equations, BIT, **20** 341–345 (1980).

[14] J. Gardiner, A. Laub, J. Amato, and C. Moler, Solution of the Sylvester matrix equation $AXB + CXD = E$, ACM Trans. Math. Software, **18** 223–231 (1992).

[15] J. Gardiner, M. Wette, A. Laub, J. Amato, and C. Moler, Algorithm 705: A Fortran-77 software package for solving the Sylvester matrix equation $AXB^T + CXD^T = E$, ACM Trans. Math. Software **18** 232–238 (1992).

[16] G. H. Golub, S. Nash, and C. F. Van Loan, A Hessenberg-Schur method for the problem $AX + XB = C$, IEEE Trans. Automat. Control **AC–24** 909–913 (1979).

[17] C. Gomez (ed.), Engineering and Scientific Computing with Scilab (Birkhäuser, Boston, 1999).

[18] S. J. Hammarling, Numerical solution of the stable, non-negative definite Lyapunov equation, IMA J. Numer. Anal., **2** 303–323 (1982).

[19] N. Higham, Accuracy and Stability of Numerical Algorithms (SIAM Publications, Philadelphia, PA, 1996).

[20] B. Kågström and P. Poromaa, LAPACK-style algorithms and software for solving the generalized Sylvester equation and estimating the separation between regular matrix pairs, ACM Trans. Math. Softw., **22** 78–103 (1996).

[21] D. Kreßner, V. Mehrmann, and T. Penzl, CTLEX – A Collection of Benchmark Examples for Continuous-time Lyapunov Equations, SLICOT Working Note 1999-6. Available from `http://www.slicot.org`.

[22] D. Kreßner, V. Mehrmann, and T. Penzl, DTLEX – A Collection of Benchmark Examples for Discrete-time Lyapunov Equations, SLICOT Working Note 1999-7. Available from `http://www.slicot.org`.

[23] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, Basic Linear Algebra Subprograms for Fortran usage, ACM Trans. Math. Softw., **5** 308–323 (1979).

[24] The MathWorks, Control System Toolbox User's Guide. For use with MATLAB (The MathWorks, Inc., Natick, MA, 1998).

[25] The MathWorks, Using MATLAB. Version 5 (The MathWorks, Inc., Natick, MA, 1999).

[26] V. Mehrmann, The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution, Lecture Notes in Control and Information Sciences Vol. 163 (Springer-Verlag, Heidelberg, 1991).

[27] NICONET Society, SLICOT Basic Systems and Control Toolbox. See `http://www.slicot.org/start.php?site=slbasic` (2005).

[28] G. Obinata and B. Anderson, Model Reduction for Control System Design, Communications and Control Engineering Series (Springer-Verlag, London, 2001).

[29] T. Penzl, Numerical solution of generalized Lyapunov equations, Adv. Comp. Math., **8** 33–48 (1997).

[30] V. Sima, Algorithms for Linear-Quadratic Optimization, Pure and Applied Mathematics Vol. 200 (Marcel Dekker, Inc., New York, NY, 1996).

[31] V. Sima, P. Petkov, and S. Van Huffel, Efficient and reliable algorithms for condition estimation of Lyapunov and Riccati equations, in: Proceedings CD of the Fourteenth International Symposium of Mathematical Theory of Networks and Systems MTNS-2000, Perpignan, France, June 19–23, 2000, Session CS 2C, Algebraic Systems Theory (3), June 22, 10 pages.

[32] M. Slowik, P. Benner, and V. Sima, Evaluation of the Linear Matrix Equation Solvers in SLICOT, SLICOT Working Note 2004-1. Available from `http://www.slicot.org`.

[33] S. Van Huffel and V. Sima, SLICOT and control systems numerical software packages, in: Proceedings of the 2002 IEEE International Conference on Control Applications and IEEE International Symposium on Computer Aided Control System Design, CCA/CACSD 2002, September 18–20, 2002, Scottish Exhibition and Conference Centre, Glasgow, Scotland, U.K. (Omnipress, 2002) pp. 39–44.

[34] A. Varga, A note on Hammarling's algorithm for the discrete Lyapunov equation, Sys. Control Lett., **15** 273–275 (1990).