SOLVING LINEAR MATRIX EQUATIONS WITH SLICOT

V. Sima[*], **P. Benner**[†]

[*] National Institute for Research & Development in Informatics, Bd. Al. Averescu, Nr. 8–10, 011455 Bucharest, Romania;
email: `vsima@iciadmin.ici.ro`; fax +40-21-224-0539
[†] Institut für Mathematik, MA 4-5, TU Berlin, Straße des 17. Juni 136, D-10623 Berlin, Germany;
email: `benner@math.tu-berlin.de`; fax +49-30-314-79706

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

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.

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. 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[1] [14, 13]), copyrighted freeware (e.g., in the SLICOT Library [4, 18]), or in the public domain (e.g., in Scilab [9]). 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 [6, 7, 12], and their counterparts for distributed memory computers, e.g., ScaLAPACK [5] and PBLAS, this library enables the user to exploit the potential of modern high-performance computer architectures.

The paper also presents some performance improvements (concerning efficiency, reliability, and accuracy) offered by the SLICOT tools, in comparison with equivalent computations performed by some MATLAB functions included in the MATLAB nucleus or in the 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. Also note that SLICOT routines often offer a higher functionality than corresponding MATLAB functions. In particular, condition and forward error estimates can be computed in many cases.

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

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, 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[2] Lyapunov equations:

$$\mathrm{op}(A)^T X + X\,\mathrm{op}(A) = \sigma C\,; \tag{3}$$
$$\mathrm{op}(A)^T X\,\mathrm{op}(A) - X = \sigma C\,; \tag{4}$$

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

$$\mathrm{op}(A)^T X + X\,\mathrm{op}(A) = -\sigma^2\,\mathrm{op}(D)^T\,\mathrm{op}(D)\;; \tag{5}$$
$$\mathrm{op}(A)^T X\,\mathrm{op}(A) - X = -\sigma^2\,\mathrm{op}(D)^T\,\mathrm{op}(D)\;; \tag{6}$$

- Generalized Sylvester equation:

$$\begin{aligned} AX - YB &= \sigma G\,, \\ EX - YF &= \sigma H\,, \end{aligned} \tag{7}$$

or the "transposed" equation

$$\begin{aligned} A^T X + E^T Y &= \sigma G\,, \\ XB^T + YF^T &= -\sigma H\,; \end{aligned} \tag{8}$$

[2] the discrete-time Lyapunov equation is also called Stein equation

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

$$\mathrm{op}(A)^{\,T} X \,\mathrm{op}(E) \,+\, \mathrm{op}(E)^{\,T} X \,\mathrm{op}(A) \,=\, \sigma C\,; \quad (9)$$
$$\mathrm{op}(A)^{\,T} X \,\mathrm{op}(A) \,-\, \mathrm{op}(E)^{\,T} X \,\mathrm{op}(E) \,=\, \sigma C\,; \quad (10)$$

- Generalized stable continuous-time and discrete-time Lyapunov equations, which have the same form as (9) and (10), respectively, but with the right-hand side replaced by $-\sigma^2 \,\mathrm{op}(D)^{\,T} \,\mathrm{op}(D)$.

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., [16, 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) [3] for Sylvester equations (for A, B general, or in RSF), or Lyapunov equations (for A general, or in RSF), with the variant from [2] for the discrete-time case; the Hessenberg-Schur method in [8] 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 [10] of the Bartels–Stewart method for stable Lyapunov equations; and extensions of the above methods for generalized Sylvester [11] and Lyapunov equations [15].

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), then the same real Schur form of A can be used by a solver able to cope with $\mathrm{op}(\cdot)$, and this would significantly improve the efficiency.

The term "stable" 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 or for computing the Hankel singular values of the system.

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

$$\mathrm{sep}(A^T, -A) = \min_{Z \neq 0} \frac{\|A^T Z + ZA\|}{\|Z\|} = \sigma_{\min}(P)\,, \quad (11)$$
$$\mathrm{sepd}(A^T, A) = \min_{Z \neq 0} \frac{\|A^T ZA - Z\|}{\|Z\|} = \sigma_{\min}(P)\,, \quad (12)$$

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

$$P \;=\; I_n \otimes A^T + A^T \otimes I_n\,, \quad (13)$$
$$P \;=\; A^T \otimes A^T - I_{n^2}\,, \quad (14)$$

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) [11].

Such measures, as well as condition number estimates and forward error bounds are returned by several routines of the SLICOT Library [17]. 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 big. We consider this as a significant advantage in reliability and functionality of the software available in SLICOT.

3 Sylvester and Lyapunov Equation Solvers

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 Control System Toolbox includes two solvers for Lyapunov and Sylvester equations. Their use is shown below, using MATLAB commands, and comments explaining their function:

```
X = lyap(A, C);       % solves AX + XA^T = -C.
X = dlyap(A, C);      % solves AXA^T - X = -C.
X = lyap(A, B, C);    % solves AX + XB = -C.
```

There is no discrete-time Sylvester solver.

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 via the SLICOT hyperlink on the NICONET (*Numerics in Control Network*) homepage

```
http://www.win.tue.nl/niconet.
```

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. These MEX-files call all SLICOT routines needed to perform the required task. The selection of the appropriate problem and solver is made using 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 provided on the SLICOT ftp site for PC platforms under Windows 95/98/00/ME/NT, or for Sun Solaris platforms (with Fortran 95). Linux versions are under current development.

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 Lyapunov equations.
`slstei`	Solve Stein equations.
`slstly`	Solve stable Lyapunov equations.
`slstst`	Solve stable Stein equations.
`slgesg`	Solve generalized linear matrix equation pairs.
`slgely`	Solve generalized Lyapunov equations.
`slgest`	Solve generalized Stein equations.
`slgsly`	Solve stable generalized 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

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

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 (11), or of the separation of the matrices A and A^T, defined by (12), 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, D, flag, trans);
U = slstst(A, D, 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 pair (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 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, sep(A, E).

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

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

4 Numerical results

This subsection presents typical performance results for some components of the SLICOT Library, called via the associated gateways. The calculations have been done on an IBM PC computer at 500 MHz, with 128 Mb memory, using Compaq Visual Fortran V6.5, non-optimized BLAS, and MATLAB 6.1 (R12). These results show that SLICOT routines often outperform MATLAB calculations. While the accuracy is comparable, and sometimes better, 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. Even better efficiency is to be expected by calling the SLICOT Fortran routines directly (not through gateways), and similar accuracy/efficiency improvements are possible for other SLICOT computations. Better results would be obtained using optimized BLAS libraries.

Figure 1 shows the execution times for SLICOT function `slstei` and MATLAB function `dlyap` (upper plot), and the speed-up factor of `slstei` compared to `dlyap` (bottom plot), for solving randomly generated Stein equations with known solutions, and $A \in \mathbb{R}^{n\times n}$, for $n = 30 : 30 : 300$. (The matrices A and X were generated with `rand`, X was made symmetric, and then the corresponding right-hand side matrix C was computed and symmetrized.) Figure 2 plots the relative residuals and relative errors for the same examples. Figure 3 shows the execution times and relative errors for solving Stein equations with A in RSF ($n = 30 : 30 : 300$). Clearly, the SLICOT function is much faster, since it can exploit the problem structure, the speed-up factors varying between 5 and 21. The relative residuals and relative errors of the solutions are even much better than those obtained with `dlyap`.

The efficiency of the SLICOT continuous-time Lyapunov solver is similar, but the relative residuals and errors are comparable with those for the corresponding MATLAB function `lyap`; see Figures 4–5.

Analogous results are also obtained for the Sylvester solvers in SLICOT and MATLAB; see Figures 6–7.

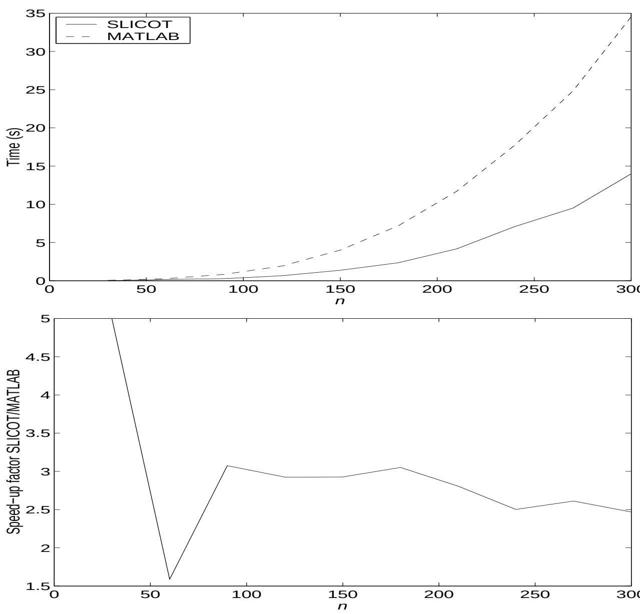

Figure 1: SLICOT `slstei` versus MATLAB `dlyap` for random Stein equations with $n = 30 : 30 : 300$. Timing comparison (top) and speed-up factor (bottom).

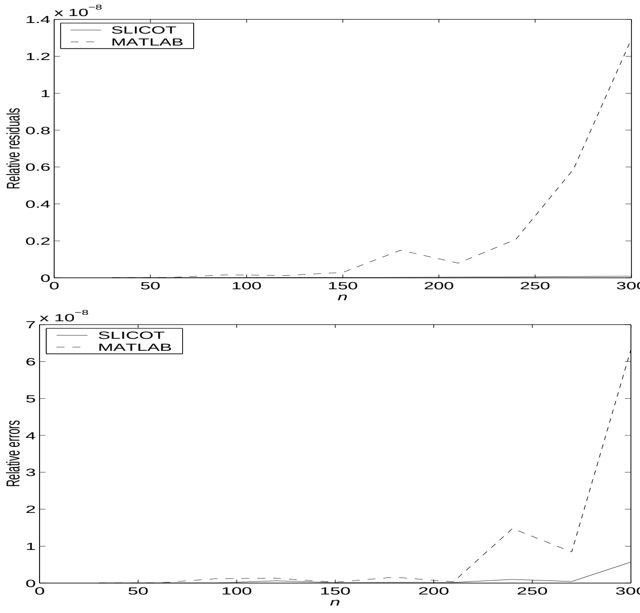

Figure 2: SLICOT `slstei` versus MATLAB `dlyap` for random Stein equations with $n = 30 : 30 : 300$. Relative residuals (top) and relative errors (bottom) comparisons.

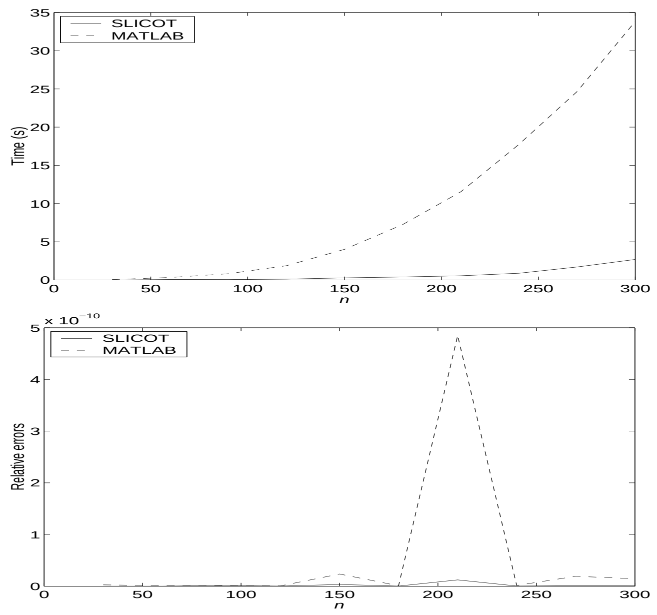

Figure 3: SLICOT `slstei` versus MATLAB `dlyap` for random Stein equations with A in real Schur form, $n = 30 : 30 : 300$. Timing (top) and relative errors (bottom) comparisons.

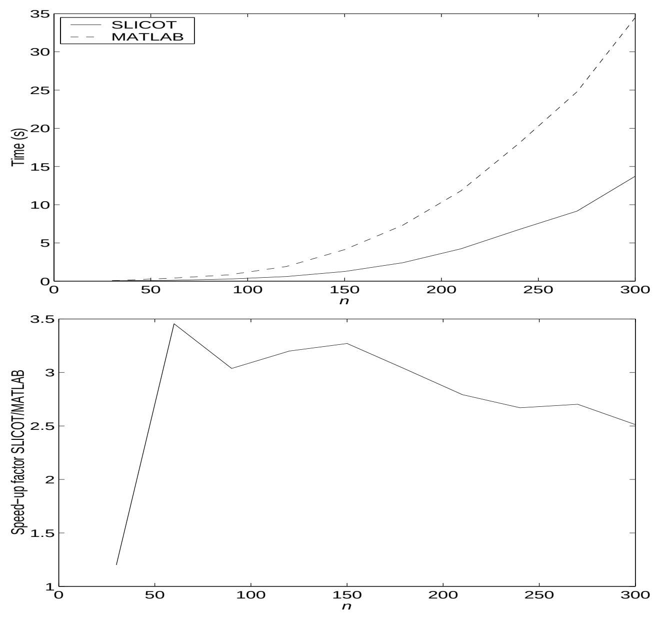

Figure 4: SLICOT `sllyap` versus MATLAB `lyap` for random Lyapunov equations with $n = 30 : 30 : 300$. Timing comparison (top) and speed-up factor (bottom).

5 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 reliability and efficiency compared to the existing software tools. This

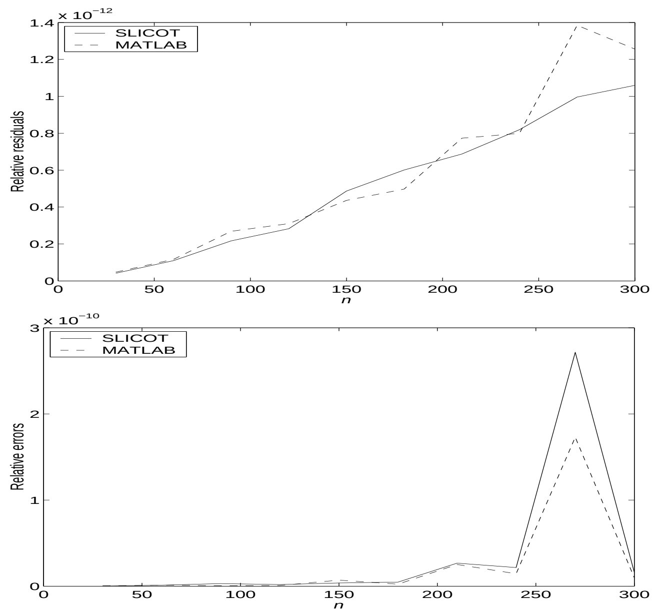

Figure 5: SLICOT `sllyap` versus MATLAB `lyap` for random Lyapunov equations with $n = 30 : 30 : 300$. Relative residuals (top) and relative errors (bottom) comparisons.

is illustrated by the results for several numerical examples.

Acknowledgments

This work was supported in part by the European Community BRITE-EURAM III *Thematic Networks Programme NICONET* (project BRRT–CT97-5040) and the DFG Research Center "Mathematics for Key Technologies" in Berlin.

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 Edition*, SIAM, Philadelphia, (1999).

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

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

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

[5] 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. Wha-

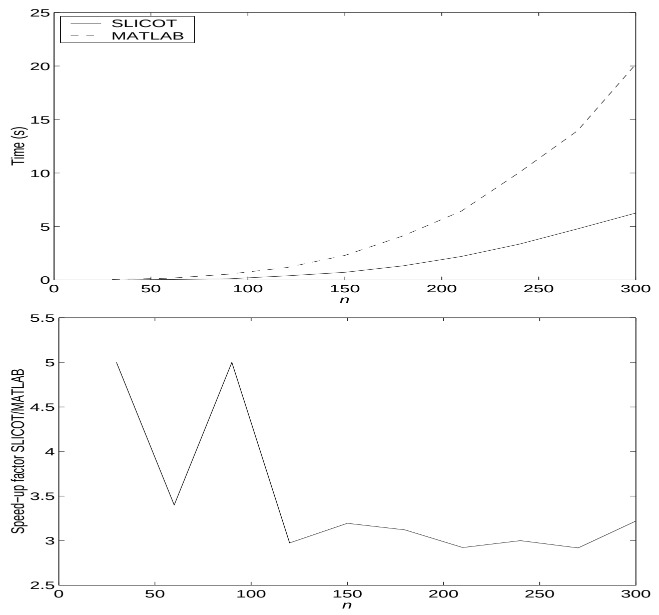

Figure 6: SLICOT `slsylv` versus MATLAB `lyap` for random Sylvester equations, $n = 30 : 30 : 300$, $m = n/2$. Timing comparison (top) and speed-up factor (bottom).

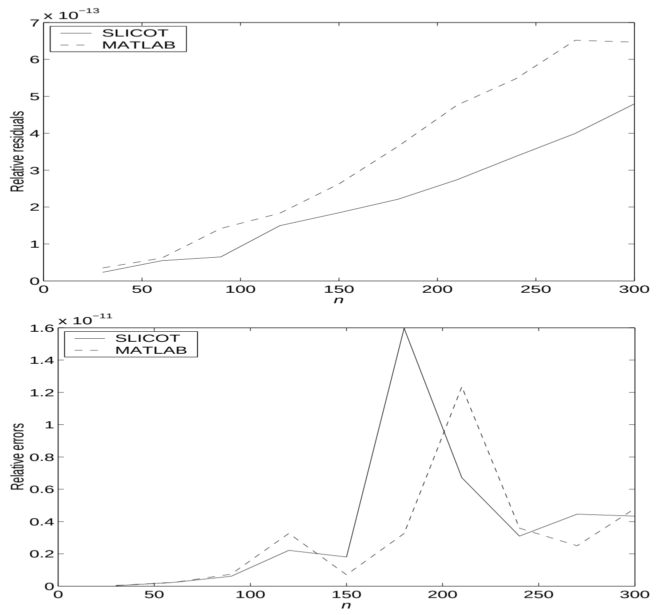

Figure 7: SLICOT `slsylv` versus MATLAB `lyap` for random Sylvester equations with $n = 30 : 30 : 300$, $m = n/2$. Relative residuals (top) and relative errors (bottom) comparisons.

ley. *ScaLAPACK Users' Guide*, SIAM, Philadelphia, (1997).

[6] 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**, pp. 1–17, 18–28, (1990).

[7] 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**, pp. 1–17, 18–32, (1988).

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

[9] C. Gomez, Ed., *Engineering and Scientific Computing with SciLab*, Birkhäuser, Boston, (1999).

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

[11] B. Kågström, 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**, pp. 78–103, (1996).

[12] 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**, pp. 308–323, (1979).

[13] The MathWorks, Inc, 24 Prime Park Way, Natick, Mass., 01760–1500. *Control System Toolbox User's Guide*, (1998).

[14] The MathWorks, Inc, 24 Prime Park Way, Natick, Mass., 01760–1500. *Using MATLAB. Version 5*, (1999).

[15] T. Penzl. "Numerical solution of generalized Lyapunov equations", *Advances in Comp. Math.*, **8**, pp. 33–48, (1998).

[16] V. Sima. *Algorithms for Linear-Quadratic Optimization*, volume 200 of *Pure and Applied Mathematics: A Series of Monographs and Textbooks*, Marcel Dekker, Inc., New York, (1996).

[17] 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, June 19–23, 2000, Perpignan, France, (2000).

[18] S. Van Huffel, V. Sima. "SLICOT and control systems numerical software packages", In P.R. Kalata, Ed., *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, Glasgow, U.K., Omnipress, pp. 39–44, (2002).