Performance of Lyapunov Solvers on Dedicated SLICOT Benchmarks Collections

Vasile Sima
National Institute for Research & Development in Informatics
Bucharest, Romania
vsima@ici.ro

***Abstract*—Lyapunov equations are often encountered in control theory and its applications, including system balancing, model and controller order reduction, and stability analyses. An accuracy-enhancing solver for standard and generalized continuous- and discrete-time Lyapunov equations is investigated in this paper. It has been derived by specializing a solver for algebraic Riccati equations based on Newton's method. The conceptual algorithm and some implementation details are summarized. The numerical results obtained by solving sets of examples of increasing dimension and numerical difficulty from the SLICOT benchmark collections for Lyapunov equations are analyzed and compared to the solutions computed by the state-of-the-art MATLAB and SLICOT solvers. The results show that most often the new solver is more accurate, sometimes by several orders of magnitude, than its competitors, and requires only a small increase of the computational effort.**

***Index Terms*—linear multivariable systems, Lyapunov equation, numerical algorithms, software, stability**

I. Introduction

Lyapunov equations are often encountered in control theory and its applications. Their solution is needed, for instance, to compute balanced (stochastic) realizations [1], [2], or to find reduced order models for systems or controllers [3]–[7]. Both mentioned applications may resort to the Hankel singular values, which are important input-output invariants of stable linear systems. Lyapunov equations also appear in Newton-like methods for solving algebraic Riccati equations (AREs), by finding the Newton direction at each iteration. Moreover, these equations are used in stability analyses of dynamical systems. For instance, consider an autonomous linear time-invariant system

$$\lambda(x(t)) = Ax(t),\ t \geq 0,\ x(0) = x_0, \tag{1}$$

where $x(t) \in \mathbb{R}^n$, and $\lambda(x(t))$ is the differential operator, $\mathrm{d}x(t)/\mathrm{d}t$, for a continuous-time system, and the advance difference operator, $\lambda(x(t)) = x(t+1)$, for a discrete-time system. This system is asymptotically stable if and only if for any symmetric positive definite matrix Y, there is a positive definite matrix X, the unique solution of a Lyapunov equation

$$A^TX + XA = -Y, \tag{2}$$

or

$$A^TXA - X = -Y, \tag{3}$$

for a continuous- or discrete-time system (1), respectively, where T denotes the matrix transposition. Discrete-time Lyapunov equations are also called *Stein equations*.

More general Lyapunov equations are considered in this paper. Specifically, the equations

$$\mathrm{op}(A)^T X\, \mathrm{op}(E) + \mathrm{op}(E)^T X\, \mathrm{op}(A) = -Y, \tag{4}$$

and

$$\mathrm{op}(A)^T X\, \mathrm{op}(A) - \mathrm{op}(E)^T X\, \mathrm{op}(E) = -Y, \tag{5}$$

where $A,\ E \in \mathbb{R}^{n\times n}$, and the operator $\mathrm{op}(M)$ is either M or M^T, are called *generalized* continuous- and discrete-time Lyapunov equations, respectively. A necessary condition for their nonsingularity is that both A and E, for (4), or either A or E, for (5), are nonsingular. Without loss of generality, it may be assumed that E is nonsingular for (5), because of the symmetric role of A and E in this equation. Stable Lyapunov equations are those for which $\Lambda(AE^{-1}) \in \mathbb{C}_-$, for (4), or $\rho(AE^{-1}) < 1$, for (5), where $\mathbb{C}_-$ is the open left half of the complex plane, and $\Lambda(M)$ and $\rho(M)$ are the spectrum and the *spectral radius* (i.e., the maximum moduli of the eigenvalues) of the matrix M, respectively. These stable equations have a unique positive-semidefinite solution, X. Then, X can be written as $X = U^TU$, where U is the Cholesky factor of X. For an identity matrix E, $E = I_n$, the *standard* Lyapunov equations are obtained. Both forms of $\mathrm{op}(\cdot)$ appear for finding the Hankel singular values of a dynamical system. For a generalized system,

$$E\lambda(x(t)) = Ax(t) + Bu(t),\ y(t) = Cx(t), \tag{6}$$

there are two closely related generalized Lyapunov equations,

$$APE^T + EPA^T = -BB^T,\ A^TQE + E^TQA = -C^TC, \tag{7}$$

in the continuous-time case, and

$$APA^T - EPE^T = -BB^T,\ A^TQA - E^TQE = -C^TC, \tag{8}$$

in the discrete-time case, where Q and R are the *controllability* and *observability Gramians*, respectively. The *Hankel singular values* are defined as the nonnegative square roots of the eigenvalues of the matrix product QP. For a stable system, this product has theoretically only nonnegative eigenvalues.

This work was partially supported by the Institutional research programme PN 1819 "Advanced IT resources to support digital transformation processes in the economy and society — RESINFO-TD" (2018), project PN 1819-01-01, "New research in complex systems modelling and optimization with applications in industry, business and cloud computing", funded by the Ministry of Research and Innovation, Romania.

Numerical computations performed without taking into account the symmetry and semidefiniteness of the solutions, might result in nonsymmetric and indefinite Gramians, due to accumulated rounding errors. Consequently, some computed Hankel singular values might appear as negative or even complex numbers, which is a nonsense. This proves the importance of ensuring reliability and accuracy of the computations. For this application, it is preferable to use the algorithm in [8], which delivers the Choleky factors R_c and R_o of the Gramians, $P = R_c R_c^T$, $Q = R_o^T R_o$, with R_c and R_o upper triangular. Moreover, the matrix products BB^T and $C^T C$ are not performed, but B and C are directly used. Then, the singular values of the product $R_c R_o$ are the Hankel singular values of the system. Therefore, they are numerically guaranteed to be real nonnegative.

This paper investigates the performance of some Lyapunov solvers in terms of accuracy and efficiency. Besides the state-of-the-art solvers from the Control System Toolbox [9] and SLICOT Library [10]–[12] (www.slicot.org), a new SLICOT-based, accuracy-enhancing solver is used. This solver has been originally destined to compute solutions of AREs using Newton's method [13]–[15], but its functionality has been recently extended to accurately solve (generalized) Lyapunov equations. All these solvers use the algorithm in [16] and its generalization [17], both implemented in SLICOT. The general algorithm in [17] reduces the matrix pair (A, E) to a real Schur-triangular form [18] by orthogonal transformations, updates the right hand side, solves the reduced Lyapunov equation, and transforms back the result to the solution of the original equation. Many computational details for standard Lyapunov equations are given, e.g., in [19]. The procedure is similar, but A is reduced to a real Schur form.

The use of the op operator as an option allows to compute only once the real Schur-triangular form (or real Schur form, when $E = I_n$) of the pair (A, E) (or of A) for finding both controllability and observability Gramians. No transposition is needed. For using the accuracy-enhancing solver, it is necessary to solve one equation using the original pair (A, E); the solver returns the Schur-triangular form of this pair, which can then be used as input for solving the second equation.

The paper is structured as follows. Section II briefly describes the underlying algorithm. Section III presents some numerical results for solving sets of examples from the SLICOT CTLEX [20] and DTLEX [21] benchmark collections for Lyapunov equations. Section IV concludes the paper.

II. Underlying Algorithm

Lyapunov equations are special cases of AREs, which miss the quadratic or rational matrix term in the continuous- or discrete-time case, respectively. Therefore, some algorithms for solving AREs could be, in principle, specialized to solve Lyapunov equations. The algorithms based on Newton's method, with or without line search [13]–[15], [22], proved to be very successful in improving the accuracy of the solutions for Riccati equations. Recently, the author adapted the Newton-based ARE solver to Lyapunov equations. The conceptual algorithm is summarized as follows.

Algorithm ALyap: Accuracy-enhancing Lyapunov solver

Input: The coefficient matrices E, A, and Y.
Output: The solution X_k of (4) or (5).

```
Set $X_0 = 0$.
FOR $k = 0, 1, \dots, k_{\max}$, DO
  1) Compute the *residual* $\mathcal{R}(X_k)$ defined by
       $\mathcal{R}(X_k) := \mathrm{op}(A)^T X_k \,\mathrm{op}(E) + \mathrm{op}(E)^T X_k \,\mathrm{op}(A) + Y,$
       $\mathcal{R}(X_k) := \mathrm{op}(A)^T X_k \,\mathrm{op}(A) - \mathrm{op}(E)^T X_k \,\mathrm{op}(E) + Y,$
     for a continuous- or discrete-time system, respectively. If
     convergence is decided, return $X_k$. If non-convergence
     is detected, return a warning or error indicator value.
  2) Solve in $L_k$ the continuous- or discrete-time generalized
     (or standard, if $E = I_n$) Lyapunov equation (9) or (10),
     respectively,
       $\mathrm{op}(A)^T L_k \,\mathrm{op}(E) + \mathrm{op}(E)^T L_k \,\mathrm{op}(A) = -\mathcal{R}(X_k),$   (9)
       $\mathrm{op}(A)^T L_k \,\mathrm{op}(A) - \mathrm{op}(E)^T L_k \,\mathrm{op}(E) = -\mathcal{R}(X_k).$   (10)
  3) Update $X_{k+1} = X_k + L_k$.
END
```

Actually, Algorithm ALyap iteratively refines a solution of a Lyapunov equation using the residual matrix at iteration k, $-\mathcal{R}(X_k)$, in the right hand side. Currently, (9) or (10) are solved using the best available algorithms for equations with dense matrices [16], [17], implemented in the SLICOT Library in the style of the DTRSYL routine from LAPACK [23]. But other algorithms can be used by making straightforward modifications. In implementation, Algorithm ALyap is refined for greater efficiency. Specifically, it exploits the fact that the same matrices A and E appear at each iteration. Therefore, A and E are reduced to the real Schur-triangular form at iteration $k = 0$, using two orthogonal transformations, Q and Z, namely

$$\widetilde{A} = Q^T A Z, \; \widetilde{E} = Q^T E Z, \tag{11}$$

where $\widetilde{A}$ is in real Schur form (i.e., block upper triangular with 1×1 or 2×2 diagonal blocks), and $\widetilde{E}$ is upper triangular. Then, the right hand side is transformed as

$$\widetilde{\mathcal{R}}(X_k) = Z^T \mathcal{R}(X_k) Z, \text{ or } \widetilde{\mathcal{R}}(X_k) = Q^T \mathcal{R}(X_k) Q, \tag{12}$$

for $\mathrm{op}(M) = M$, or $\mathrm{op}(M) = M^T$, respectively. A reduced equation, (13) or (14),

$$\mathrm{op}(\widetilde{A})^T \widetilde{L}_k \,\mathrm{op}(\widetilde{E}) + \mathrm{op}(\widetilde{E})^T \widetilde{L}_k \,\mathrm{op}(\widetilde{A}) = -\widetilde{\mathcal{R}}(X_k)\,, \tag{13}$$
$$\mathrm{op}(\widetilde{A})^T \widetilde{L}_k \,\mathrm{op}(\widetilde{A}) - \mathrm{op}(\widetilde{E})^T \widetilde{L}_k \,\mathrm{op}(\widetilde{E}) = -\widetilde{\mathcal{R}}(X_k)\,, \tag{14}$$

respectively, is solved for $\widetilde{L}_k$. Finally, $\widetilde{L}_k$ is transformed back to the solution of the original equation,

$$L_k = Q \widetilde{L}_k Q^T, \text{ or } L_k = Z \widetilde{L}_k Z^T, \tag{15}$$

for $\mathrm{op}(M) = M$, or $\mathrm{op}(M) = M^T$, respectively.

The most efficient technique for the iterative improvement of a Lyapunov equation solution is to iterate on the solutions of the reduced equations, without transforming the right hand sides with (12) and back transforming with (15) at each iteration, but only at the first and the last iterations, respectively.

The main termination criterion for the iterative process is defined based on a *normalized residual*, $r_k := r(X_k)$, and a tolerance τ. Specifically, if

$$r_k := \|\mathcal{R}(X_k)\|_F / \max(1, \|X_k\|_F) \leq \tau, \qquad (16)$$

where $\|M\|_F$ denotes the Frobenius norm of the matrix M, the calculations are successfully terminated at iteration k, and X_k is the computed solution. If $\tau \leq 0$ is given on input, it is replaced by a default tolerance, evaluated by one of the formulas below for (4) and (5), respectively,

$$\tau = \min\{\,\varepsilon_M\sqrt{n}\big(2\,\|A\|_F\|E\|_F + \|Y\|_F\big), \sqrt{\varepsilon_M}/10^3\,\},$$
$$\tau = \min\{\,\varepsilon_M\sqrt{n}\big(\|A\|_F^2 + \|E\|_F^2 + \|Y\|_F\big), \sqrt{\varepsilon_M}/10^3\,\},$$

where ε_M is the relative machine precision. The aim of introducing the second operand of min in the two formulas above was to prevent a premature convergence decision for equations with very large norms for A, E, and/or Y. For such equations, the termination criterion based on (16) might not hold in a reasonable number of iterations, if X has a small norm. However, a good approximate solution might actually be found in few iterations. Therefore, the MATLAB-style *relative residual*, $r_r(X_k)$, defined as the ratio between $\|\mathcal{R}(X_k)\|_F$ and the sum of the Frobenius norms of the matrix terms in (4) or (5), is also used as a termination criterion. However, it is not tested at each iteration, to limit the increase of additional computational effort. Moreover, updating X_k in the last step of Algorithm ALyap is done only if $\|L_k\|_F > \varepsilon_M\|X_k\|_F$. Otherwise, the iterative process ends with the computed solution X_k.

III. Numerical Results

This section presents part of the results of an extensive performance investigation of the Newton-based accuracy-enhancing Lyapunov solver in comparison to the Control System Toolbox and SLICOT Library solvers. (The reference to this solver as a "Newton-based" one is due to its straightforward derivation from the corresponding ARE solver.) The calculations have been performed in double precision on a 64-bit Intel Core i7-3820QM portable computer (2.7 GHz, 16 GB RAM), using Intel Visual Fortran Composer XE 2015 and MATLAB 8.6.0.267246 (R2015b). A MATLAB executable MEX-function has been built using SLICOT subroutines and MATLAB-provided optimized LAPACK and BLAS libraries.

Although many tests have been performed with randomly generated matrices, the results reported in the sequel are obtained using the SLICOT benchmark collections for Lyapunov equations, CTLEX [20] and DTLEX [21], for (generalized) continuous- and discrete-time systems, respectively. The short notation TLEX will be used for both CTLEX and DTLEX examples. These benchmarks have been explicitly designed to support evaluating the performance of a solution method and of its implementation with respect to correctness, accuracy, and speed, or to compare different numerical methods and investigate their behavior in difficult situations. Although planned to include four groups of examples, currently these collections contain just group 4 — parameter-dependent examples of scalable size — with four examples in each collection.

TABLE I
Details of TLEX Examples. Default Values Are Put into Parantheses.

Example	Order	Parameter(s)
4.1	$n \geq 2$ (10)	$r > 1$ (1.5) $s > 1$ (1.5)
4.2 CTLEX DTLEX	$n \geq 2$ (10)	$s > 1$ (1.5) $\lambda < 0$ (-0.5) $\lvert\lambda\rvert < 1$ (-0.5)
4.3	$n \geq 2$ (10)	$t \geq 0$ (10)
4.4	$n = 3q$, $q \geq 1$ (10)	$t \geq 1$ (1.5)

TABLE II
Parameters Used for the TLEX Examples.

Example	Order	Parameter(s)
4.1	`list_n` = 5 : 5 : 20	`list_r` = 1.1 : 0.2 : 1.9 `list_s` = 1.1 : 0.2 : 1.9
4.2 CTLEX DTLEX	`list_n` = 5 : 5 : 20	`list_s` = 1.1 : 0.2 : 1.9 `list_l` = −2 : 0.2 : −0.2 `list_l` = −0.9 : 0.2 : 0.9
4.3	`list_n` = 5 : 5 : 20	`list_t` = 1 : 1 : 30
4.4	`list_n` = 15 : 15 : 60	`list_t` = 1.1 : 0.2 : 9.9

One parameter of the TLEX examples is the order n. Other parameters may be chosen to set the *numerical condition* of the problem, which affects the accuracy of the computed solution and its sensitivity to small perturbations in the input data. All parameters have default values. Table I summarizes the main information about the TLEX examples. The examples 4.1 and 4.2 represent stable standard Lyapunov equations, while the examples 4.3 and 4.4 represent generalized Lyapunov equations (stable for 4.4). Moreover, the solutions of the equations for examples 4.1 and 4.3 are known, since they can be computed with machine accuracy.

TLEX examples have been used for several values of the order n and of the other parameters. For examples 4.1, 4.2, and 4.3, n took the values 5, 10, 15, and 20. For example 4.4, q took the same values, so n has been set to 15, 30, 45, and 60. Table II summarizes all tried parameter values. The MATLAB notation $i = k : l : m$ is used, i.e., i takes the values k, $k + l$, $k + 2l$, …, m.

The figures presented in the sequel have as abscissa axes the specific example index in a series, generated by two or three nested loops. For instance, the series of examples 4.1 is defined by a loop for `n = list_n`, followed by a loop for `r = list_r`, and then by a loop for `s = list_s`. For example 4.2, the order of the loops is `list_n`, `list_l`, and `list_s`, and for examples 4.3 and 4.4, the order is `list_n` and `list_t`.

The performance analysis program also estimated the reciprocal condition number, `rcond`, of the Lyapunov equations, using the SLICOT-based MATLAB functions `lyapcond` and `steicond`, for standard continuous- and discrete-time Lyapunov equations, respectively. For generalized Lyapunov equations, the same functions are called for the matrices A

and Y replaced by $E^{-1}A$ and $E^{-T}YE^{-1}$, respectively. Note that these estimators are using the exact solutions of the Lyapunov equations, when known, or the solutions, denoted X_m, computed by the corresponding MATLAB functions `lyap` or `dlyap`, otherwise. The chosen sequence of values for n and other parameters for each example produces a zigzaggy variation of `rcond`.

For TLEX 4.1 and CTLEX 4.4 series of examples, the equations with an estimated reciprocal condition number smaller than $\varepsilon_M^{1/2} \approx 1.49 \cdot 10^{-8}$ have been excluded; this way, 75, 77, and 85 examples remained for comparisons. Similarly, for TLEX 4.2 series, the equations with `rcond` $< 10^{-14}$ have been excluded, retaining 171 and 135 examples. No examples have been excluded for TLEX 4.3 series. No example in the DTLEX 4.4 series had a too small `rcond`, but the examples with $\|X_m\|_F > 0.4/\varepsilon_M \approx 1.8 \cdot 10^{15}$ have been excluded, since then the normalized residuals could artificially have a very small value. This way, 132 examples remained.

To assess the accuracy of the computed solutions, the relative errors are used when the true solution, X, is known (i.e., for TLEX examples 4.1 and 4.3). Otherwise, the normalized residuals defined in (16) are used. The *relative error* of a computed solution $\hat{X}$ is evaluated using the formula

$$e(\hat{X}) := \|\hat{X} - X\|_F / \max(1, \|X\|_F).$$

Fig. 1 shows the relative errors for CTLEX example 4.1 with various parameters, using MATLAB, SLICOT, and Newton-based solvers. For most examples, the Newton-based solver is the most accurate, but for 19 examples with large n and significantly ill-conditioned, this solver obtained slightly larger relative errors than MATLAB function `lyap`. Fig. 2 plots the number of iterations for the Newton-based solver and reciprocal condition numbers for the generated equations. Most examples need less than four iterations. Fig. 3 presents the elapsed CPU times for these examples. The SLICOT solver is the fastest, seconded by the Newton-based solver, with very few exceptions.

Fig. 4 shows similarly the behavior for CTLEX example 4.2. Newton-based solver is clearly the winner in terms of normalized residuals; like in Fig. 3, with few exceptions, it is faster than `lyap`, and not much slower than the SLICOT solver.

Fig. 5 plots the main results for CTLEX example 4.3. Newton-based solver shows an excellent accuracy, with relative error near ε_M, while the other solvers, and especially `lyap`, may lose even more than half of the machine precision, and are strongly influenced by the problem conditioning. The improvement is sometimes by many orders of magnitude (even by 10 orders). In most cases, two iterations are enough to attain this performance.

For CTLEX example 4.4, the three solvers have comparable behavior concerning the normalized residuals, but Newton-based solver is slightly better. Its speed is comparable with that of `lyap` for examples with order $n = 15$, but larger for a bigger order.

The behavior for discrete-time examples is similar. Newton-based solver is (much) more accurate for the first 35 examples, but also for several other examples from the DTLEX 4.1 series, see Fig. 6. Only for 14 examples, this solver obtained slightly larger relative errors than MATLAB function `dlyap`. For most cases, two iterations have been enough; consequently, Newton-based solver needed a CPU time placed in between the CPU times of the other two solvers, see Fig. 7.

Again, the new solver produced the smallest normalized

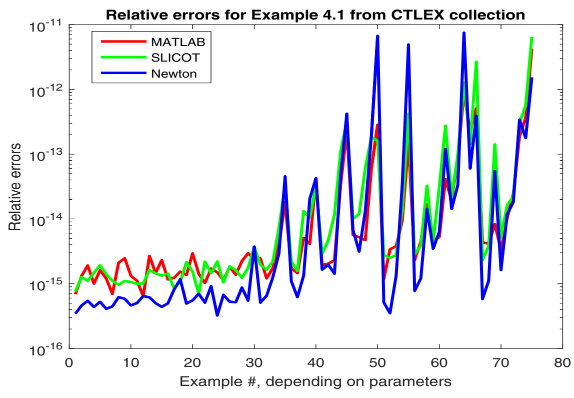

Fig. 1. Relative errors for CTLEX example 4.1 with various parameters, using MATLAB, SLICOT, and Newton-based solvers.

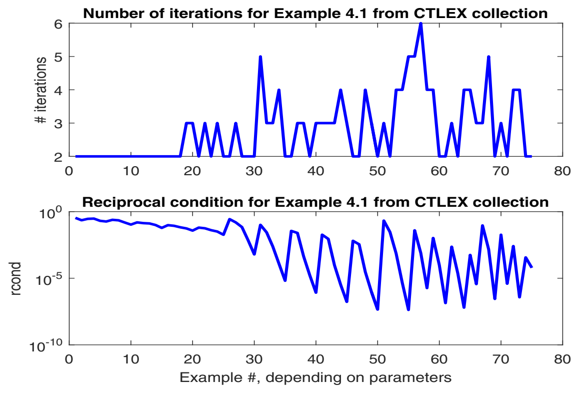

Fig. 2. Number of iterations for Newton-based solver and reciprocal condition numbers for CTLEX example 4.1 with various parameters.

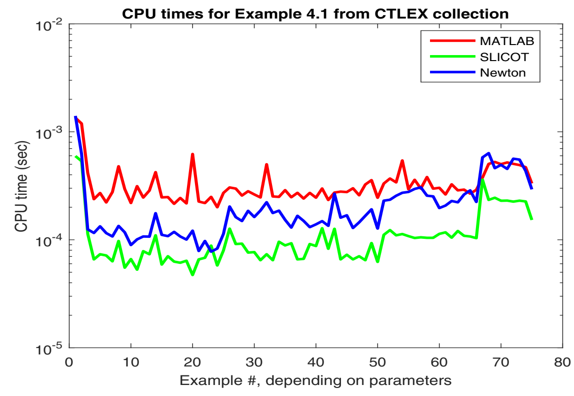

Fig. 3. Elapsed CPU time for CTLEX example 4.1 with various parameters, using MATLAB, SLICOT, and Newton-based solvers.

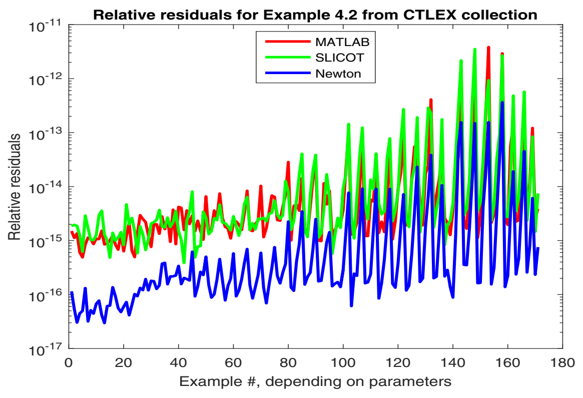

Fig. 4. Normalized residuals for CTLEX example 4.2 with various parameters, using MATLAB, SLICOT, and Newton-based solvers.

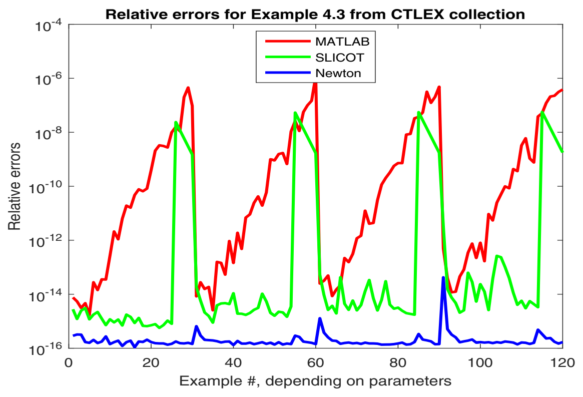

Fig. 5. Relative errors for CTLEX example 4.3 with various parameters, using MATLAB, SLICOT, and Newton-based solvers.

residuals for DTLEX 4.2 series, see Fig. 8. Only for 8 examples in this series, Newton-based solver obtained slightly larger normalized residuals than dlyap. The new solver is also a clear winner in accuracy for DTLEX examples 4.3 and, especially, 4.4, as shown in Fig. 9 and Fig. 10. Only for one example in the DTLEX 4.3 series, with condition number about $1.5 \cdot 10^{-12}$, Newton-based solver obtained a slightly larger relative error than dlyap. It was more accurate than dlyap for all DTLEX 4.4 examples, sometimes by several orders of magnitude.

IV. Conclusion

An accuracy-enhancing solver for standard and generalized continuous- and discrete-time Lyapunov equations, derived by specializing a solver for algebraic Riccati equations based on Newton's method, has been investigated. The conceptual algorithm and some implementation details have been summarized. The implementation uses the best algorithms for solving Lyapunov equations with dense matrices, based on the orthogonal reduction to the real Schur(-triangular) form. This reduction is performed only at the first iteration of the computational process. The numerical results obtained by solving sets of examples of increasing dimension and numerical difficulty from the SLICOT benchmark collections for standard and generalized continuous- and discrete-time Lyapunov equations are discussed and compared to the solutions computed by the state-of-the-art MATLAB and SLICOT solvers. The results show that most often the new solver is more accurate, sometimes by several orders of magnitude, than its competitors, without a significant increase of the computational effort.

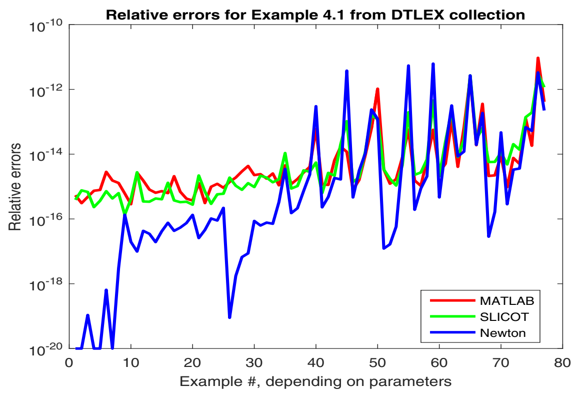

Fig. 6. Relative errors for DTLEX example 4.1 with various parameters, using MATLAB, SLICOT, and Newton-based solvers.

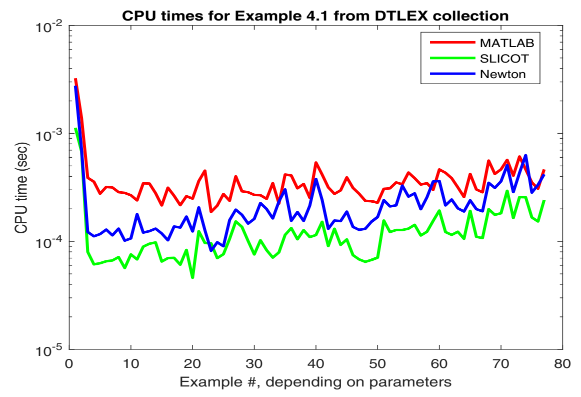

Fig. 7. Elapsed CPU time for DTLEX example 4.1 with various parameters, using MATLAB, SLICOT, and Newton-based solvers.

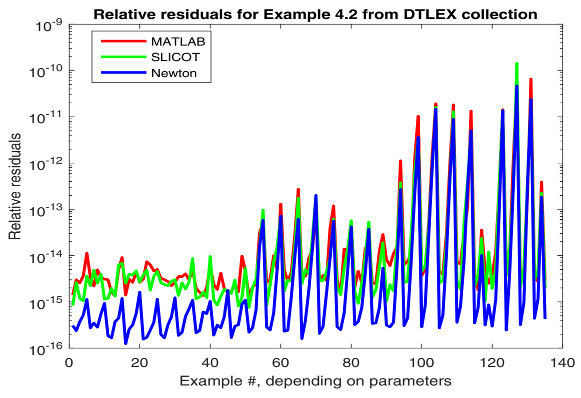

Fig. 8. Normalized residuals for DTLEX example 4.2 with various parameters, using MATLAB, SLICOT, and Newton-based solvers.

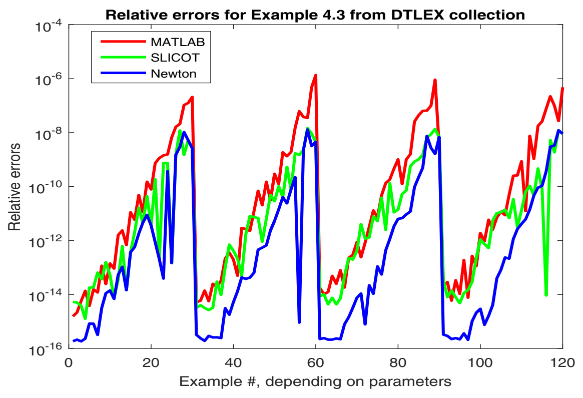

Fig. 9. Relative errors for DTLEX example 4.3 with various parameters, using MATLAB, SLICOT, and Newton-based solvers.

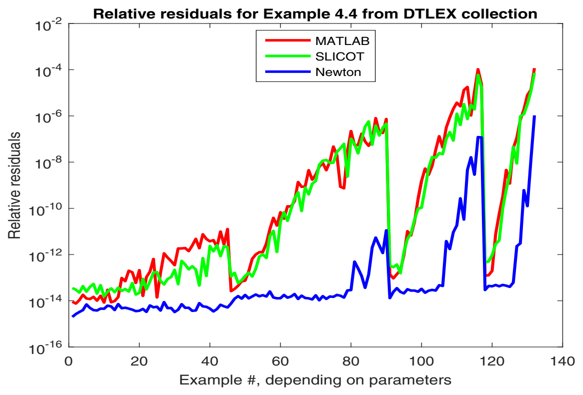

Fig. 10. Normalized residuals for DTLEX example 4.4 with various parameters, using MATLAB, SLICOT, and Newton-based solvers.

References

[1] M. Green, "Balanced stochastic realization," *Lin. Alg. Appl.*, vol. 98, pp. 211–247, 1988.

[2] R. Peeters, B. Hanzon, and M. Olivi, "Balanced realizations of discrete-time stable all-pass systems and the tangential Schur algorithm," in *Proceedings of the European Control Conference*, 31 August–3 September 1999, Karlsruhe, Germany, 1999.

[3] J. M. Badía, P. Benner, R. Mayo, and E. S. Quintana-Ortí, "Parallel algorithms for balanced truncation model reduction of sparse systems," in *Applied Parallel Computing*, ser. Lecture Notes in Computer Science, vol. 3732/2006. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 267–275.

[4] C.-A. Lin and T.-Y. Chiu, "Model reduction via frequency-weighted balanced realization," *Control Theory and Advanced Technology*, vol. 8, pp. 341–351, 1992.

[5] M. G. Safonov and R. Y. Chiang, "A Schur method for balanced-truncation model reduction," *IEEE Trans. Automat. Contr.*, vol. AC–34, pp. 729–733, 1989.

[6] M. S. Tombs and I. Postlethwaite, "Truncated balanced realization of a stable non-minimal state-space system," *Int. J. Control*, vol. 46, pp. 1319–1330, 1987.

[7] Y. Liu and B. D. O. Anderson, "Singular perturbation approximation of balanced systems," *Int. J. Control*, vol. 50, pp. 1379–1405, 1989.

[8] S. J. Hammarling, "Numerical solution of the stable, non-negative definite Lyapunov equation," *IMA J. Numer. Anal.*, vol. 2, no. 3, pp. 303–323, 1982.

[9] "Control System Toolbox User's Guide. Version 9," The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA, 01760–2098, 2011.

[10] 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*, B. N. Datta, Ed. Boston, MA: Birkhäuser, 1999, vol. 1, chapter 10, pp. 499–539.

[11] P. Benner, D. Kressner, V. Sima, and A. Varga, "Die SLICOT-Toolboxen für Matlab," *at—Automatisierungstechnik*, vol. 58, no. 1, pp. 15–25, Jan. 2010.

[12] S. Van Huffel, V. Sima, A. Varga, S. Hammarling, and F. Delebecque, "High-performance numerical software for control," *IEEE Control Syst. Mag.*, vol. 24, no. 1, pp. 60–76, Feb. 2004.

[13] V. Sima and P. Benner, "Numerical investigation of Newton's method for solving continuous-time algebraic Riccati equations," in *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO-2014)*, 1-3 September, 2014, Vienna, Austria, J.-L. Ferrier, O. Gusikhin, K. Madani, and J. Sasiadek, Eds., vol. 1. SciTePress — Science and Technology Publications, Portugal, 2014, pp. 404–409.

[14] V. Sima, "Computational experience with a modified Newton solver for continuous-time algebraic Riccati equations," in *Informatics in Control, Automation and Robotics*, ser. Lecture Notes in Electrical Engineering, J.-L. Ferrier, O. Gusikhin, K. Madani, and J. Sasiadek, Eds. Switzerland: Springer International Publishing, 2015, vol. 325, ch. 3, pp. 55–71.

[15] V. Sima and P. Benner, "Numerical investigation of Newton's method for solving discrete-time algebraic Riccati equations," in *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics (ICINCO-2018)*, 29–31 July, 2018, Porto, Portugal, K. Madani and O. Y. Gusikhin, Eds. SciTePress — Science and Technology Publications, Portugal, 2018, in press.

[16] R. H. Bartels and G. W. Stewart, "Algorithm 432: Solution of the matrix equation $AX + XB = C$," *Comm. ACM*, vol. 15, no. 9, pp. 820–826, 1972.

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

[18] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MA: The Johns Hopkins University Press, 1996.

[19] V. Sima, *Algorithms for Linear-Quadratic Optimization*, ser. Pure and Applied Mathematics: A Series of Monographs and Textbooks, E. J. Taft and Z. Nashed (Series editors), vol. 200. New York: Marcel Dekker, Inc., 1996.

[20] D. Kressner, V. Mehrmann, and T. Penzl, "CTLEX—A collection of benchmark examples for continuous-time Lyapunov equations," SLICOT Working Note 1999-6, Jun. 1999. [Online]. Available: www.slicot.org.

[21] ——, "DTLEX—A collection of benchmark examples for discrete-time Lyapunov equations," SLICOT Working Note 1999-7, Jun. 1999. [Online]. Available: www.slicot.org.

[22] P. Benner, "Accelerating Newton's method for discrete-time algebraic Riccati equations," in *Mathematical Theory of Networks and Systems, Proceedings of the MTNS-98 Symposium held in Padova, Italy, July, 1998*, A. Beghi, L. Finesso, and G. Picci, Eds. Padova, Italy: Il Poligrafo, 1998, pp. 569–572.

[23] 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*, ser. Software · Environments · Tools. Philadelphia: SIAM, 1999.