Efficient Solution of Coupled Lyapunov Equations via Matrix Sign Function Iteration

Peter Benner
Zentrum für Technomathematik
Fachbereich 3/Mathematik und Informatik
Universität Bremen
D–28334 Bremen
Germany
E-mail: benner@math.uni-bremen.de
Fax: +49 – 421 – 2184863

José M. Claver* Enrique S. Quintana-Ortí*
Departamento de Informática
Universidad Jaume I
Campus Penyeta Roja
12071–Castellón
Spain
E-mails: claver@inf.uji.es, quintana@inf.uji.es
Fax: +34 – 64 – 345848

Abstract

The controllability and observability Gramians of time-invariant linear systems are given by the solutions of two coupled Lyapunov equations. These Gramians play an essential role in many areas of control theory such as computing Hankel singular values, system balaning, and related model reduction algorithms. We investigate the numerical solution of coupled Lyapunov equations by iterations for the matrix sign function, and demonstrate by numerical examples the efficiency of the resulting algorithm on modern computer architectures. As the Cholesky factors of the Gramians are often required rather than the Gramians themselves, special emphasis is given on the direct computation of these factors without forming the Gramians explicitly.

Key words: numerical methods, coupled Lyapunov equations, controllability Gramian, observability Gramian, matrix sign function.

1 Introduction

Let a stable realization of a linear time-invariant system be given in generalized state-space form, i.e.,

$$\begin{array}{rcl} E\dot{x}(t) & = & Ax(t) + Bu(t), \quad x(0) = x^0, \\ y(t) & = & Cx(t), \end{array} \tag{1}$$

where $A, E \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$, E is nonsingular, and $A - \lambda E$ is a stable matrix pencil. Note that these assumptions imply the nonsingularity of A.

*Partially supported by the CICYT Project # TIC96-1062-C03-C03.

In linear control problems governed by first-order ordinary differential equations (ODE), usually $E = I_n$, where I_n denotes the identity of order n. The case $E \neq I_n$ appears if the control problem is governed by a second-order ODE (e.g., [8]) or a descriptor system (e.g., [17]), or if the underlying first-order ODE comes from a finite-element discretization of a partial differential equation (PDE) (e.g., [3, 21]).

In principle, (1) can be transformed to standard state-space form by inverting E and working with the system defined by $(E^{-1}A, E^{-1}B, C)$. Nevertheless, this may introduce large rounding errors in case E is ill-conditioned. Moreover, in many of the above-mentioned applications, E is a sparse matrix while its inverse may be full. As we show in Section 2, the additional cost for our algorithms caused by using the generalized rather than the standard state-space form, basically comes from matrix multiplications with E. This additional cost is negligible if E has only a small number of nonzero entries.

The controllability and observability Gramians, W_c and W_o, respectively, are given via the solutions X and Y of the *generalized Lyapunov equations*

$$AXE^T + EXA^T + BB^T = 0, \tag{2}$$
$$A^TYE + E^TYA + C^TC = 0, \tag{3}$$

and the relations

$$W_c = X, \qquad W_o = E^TYE. \tag{4}$$

An immediate consequence of Lyapunov stability theory (see, e.g., [14, Section 13.1]) is that, under the given assumptions, W_c and W_o are both positive semidefinite. Moreover, if $(E^{-1}A, E^{-1}B)$ is controllable, then W_c is nonsingular and hence definite; if $(E^{-1}A, C)$ is observable, the same holds for W_o.

2 Solving Lyapunov Equations via Sign Function Iterations

The *sign function* of a matrix $Z \in \mathbb{R}^{n\times n}$ can be defined as follows. Let $\sigma(Z) \cap \imath\mathbb{R} = \emptyset$, where $\sigma(Z)$ denotes the spectrum of Z and $\imath := \sqrt{-1}$. Denote the Jordan decomposition of Z by

$$Z \;=\; S \begin{bmatrix} J^- & 0 \\ 0 & J^+ \end{bmatrix} S^{-1}, \tag{5}$$

where $\sigma(J^-) \subset \mathbb{C}^-$, $\sigma(J^+) \subset \mathbb{C}^+$. Here, $\mathbb{C}^-$ denotes the open left, $\mathbb{C}^+$ the open right half complex plane. If $J^- \in \mathbb{R}^{k\times k}$ and $J^+ \in \mathbb{R}^{(n-k)\times(n-k)}$, then

$$\operatorname{sign}(Z) \;:=\; S \begin{bmatrix} -I_k & 0 \\ 0 & I_{n-k} \end{bmatrix} S^{-1}. \tag{6}$$

Note that $\operatorname{sign}(Z)$ is unique and independent of the order of the eigenvalues in the Jordan decomposition of Z (see, e.g., [13, Section 22.1]). Many other equivalent definitions for $\operatorname{sign}(Z)$ can be given; see, e.g., [12].

A generalization of the matrix sign function for a matrix pencil $Z - \lambda Y$ was given by Gardiner and Laub [7] in case Z and Y are both nonsingular. They consider the iteration

$$\begin{aligned} &Z_0 := Z. \\ &\texttt{FOR } k = 0,1,2,\ldots \textit{ until convergence} \\ &\quad Z_{k+1} \leftarrow \frac{1}{2c_k}\left(Z_k + c_k^2 Y Z_k^{-1} Y\right), \end{aligned} \tag{7}$$

where the scalar c_k is chosen in order to accelerate the convergence of the iteration. It can be proved that $\lim_{k\to\infty} Z_k =: Z_\infty$ exists and $\operatorname{sign}\left(Y^{-1}Z\right) = Y^{-1}Z_\infty$; see [7].

Now consider a generalized Lyapunov equation of the form

$$A^T X E + E^T X A + Q \;=\; 0, \tag{8}$$

where $A, E, Q, X \in \mathbb{R}^{n\times n}$, $Q = Q^T$, and $X = X^T$ is the sought-after solution. Assuming $A - \lambda E$ is stable and A, E are nonsingular, we can apply iteration (7) to the matrix pencil

$$Z - \lambda Y \;:=\; \begin{bmatrix} A & 0 \\ Q & -A^T \end{bmatrix} - \lambda \begin{bmatrix} E & 0 \\ 0 & E^T \end{bmatrix}. \tag{9}$$

It can be shown that in this case,

$$Z_\infty \;:=\; \begin{bmatrix} -E & 0 \\ 2E^T X E & E^T \end{bmatrix} \tag{10}$$

and the solution X of (8) can be obtained from Z_∞ by solving two linear systems of equations; see [3, 4, 7, 20].

Remark 2.1 *In case $E = I_n$, the solution X can be read off directly from the lower left $n \times n$ block of Z_∞ and (7) reduces to the standard Newton iteration for the computation of the matrix sign function as introduced in [20].*

In [3, 4] (and in [20] for the case $E = I_n$) it is observed that the iteration (7) greatly simplifies when applied to $Z - \lambda Y$ from (9). Rather than one iteration with $2n \times 2n$ matrices Z_k, it is sufficient to consider two iterations with $n \times n$ matrices:

$$\begin{aligned} &A_0 := A,\; Q_0 := Q. \\ &\texttt{FOR } k = 0,1,2,\ldots \textit{ until convergence} \\ &\quad A_{k+1} \leftarrow \frac{1}{2c_k}\left(A_k + c_k^2 E A_k^{-1} E\right), \\ &\quad Q_{k+1} \leftarrow \frac{1}{2c_k}\left(Q_k + c_k^2 E^T A_k^{-T} Q_k A_k^{-1} E\right). \end{aligned} \tag{11}$$

Denoting the limits of this iteration by

$$A_\infty := \lim_{k\to\infty} A_k, \qquad Q_\infty := \lim_{k\to\infty} Q_k,$$

we obtain

$$Z_\infty \;=\; \begin{bmatrix} A_\infty & 0 \\ Q_\infty & -A_\infty^T \end{bmatrix}, \tag{12}$$

and hence, $X = E^{-T} Q_\infty E^{-1}/2$. In [4] these ideas were used to design and investigate numerical algorithms for the solution of generalized Lyapunov equations of the form (8).

The iteration (7) can be applied to (3) directly by setting $Q_0 := C^T C$ in order to obtain the observability Gramian as $W_o = Q_\infty/2$. (Recall that from (4), $W_o = E^T Y E$ if Y is the solution of (3).) That is, for computing the observability Gramian, we do not need to invert E.

For (2), we have to replace A by A^T and E by E^T while setting $Q_0 := BB^T$. Rewriting the above iteration accordingly, we obtain

$$\begin{aligned} &\tilde{A}_0 := A^T,\; P_0 := BB^T. \\ &\texttt{FOR } k = 0,1,2,\ldots \textit{ until convergence} \\ &\quad \tilde{A}_{k+1} \leftarrow \frac{1}{2c_k}\left(\tilde{A}_k + c_k^2 E^T \tilde{A}_k^{-1} E^T\right), \\ &\quad P_{k+1} \leftarrow \frac{1}{2c_k}\left(P_k + c_k^2 E \tilde{A}_k^{-T} P_k \tilde{A}_k^{-1} E^T\right). \end{aligned} \tag{13}$$

such that the controllability Gramian is given by $W_c \;=\; E^{-1} P_\infty E^{-T}/2$. Here, we have to solve two matrix equations which can be achieved by only one factorization (e.g., LU or QR) of E. Note that the accuracy of the computed controllability Gramian will be affected by the conditioning of E; for more details, see [4]. It is easy to see that for the iterates in (13) we have

$\tilde{A}_j = A_j^T$ if the A_j denote the iterates of (11). Hence, W_c and W_o can be computed simultaneously by

$$
\begin{aligned}
&A_0 := A,\ P_0 := BB^T,\ Q_0 := C^TC.\\
&\texttt{FOR } k = 0,1,2,\dots \textit{ until convergence}\\
&\qquad A_{k+1} \leftarrow \frac{1}{2c_k}\left(A_k + c_k^2 E A_k^{-1} E\right),\\
&\qquad P_{k+1} \leftarrow \frac{1}{2c_k}\left(P_k + c_k^2 E A_k^{-1} P_k A_k^{-T} E^T\right),\\
&\qquad Q_{k+1} \leftarrow \frac{1}{2c_k}\left(Q_k + c_k^2 E^T A_k^{-T} Q_k A_k^{-1} E\right),
\end{aligned}
\tag{14}
$$

and setting

$$
W_c = \frac{1}{2}E^{-1}P_\infty E^{-T}, \quad W_o = \frac{1}{2}Q_\infty. \tag{15}
$$

The choice of the scaling factor c_k and the stopping criterion are discussed in Section 3.

Of course, if the solutions of the coupled Lyapunov equations (2) and (3) are required rather than the Gramians, then we still have to solve $E^TYE = Q_\infty$ for Y. This can be achieved by using the factorization of E already computed when solving $EXE^T = P_\infty$.

One step of Iteration (14) requires an LU factorization of A_k, solving two linear systems, and several matrix multiplications. This matrix-multiplication richness makes the method attractive for computer architectures that can take advantage of block-partitioned algorithms, and specially for parallel distributed architectures. Note that if E is a sparse matrix, this can be employed when implementing the matrix products with E in order to reduce the cost of the iteration. Moreover, if A and E commute, then A_k and E commute, too. In that case, we can save one linear system solve since $EA_k^{-1} = A_k^{-1}E$. This is sometimes the case, e.g., if (1) comes from the discretization of distributed parameter systems as in [3, 21].

Remark 2.2 *For $E = I_n$, iteration (14) has been used in [15] as the first step in a balanced model reduction algorithm.*

We have already noted in Section 1 that under the given assumptions, the solutions X and Y of (2) and (3), respectively, are positive semidefinite, and if the underlying system is controllable and observable, then both matrices are positive definite. Hence, there exist full-rank factorizations $X = X_1X_1^T$ and $Y = Y_1^TY_1$ where $\text{rank}\,(X_1) = \text{rank}\,(X)$ and $\text{rank}\,(Y_1) = \text{rank}\,(Y)$. In case X and Y are nonsingular and X_1, Y_1^T are chosen lower triangular, these are the Cholesky factorizations of X and Y, respectively. In the following, we will call X_1, Y_1 Cholesky factors, regardless of their rank.

Often, the Cholesky factor is required rather than the solution X itself, e.g., [9, 18, 22]. Hammarling's method for solving Lyapunov equations [10, 23, 19] of the form (3) computes this factor without forming the product C^TC and the solution X explicitly. The advantage of this approach is that the condition number of X can be up to the square of that of its Cholesky factor X_1. Hence, a significant increase in accuracy can be observed using X_1 instead of X if X is ill-conditioned.

In particular, if the eigenvalues and eigenspaces of the product W_cW_o are desired (as in Hankel singular values computation, system balancing, and model reduction), they can be computed via the singular value decomposition (SVD) of V_oV_c, where $W_c = V_cV_c^T$, $W_o = V_o^TV_o$ are Cholesky factorizations [22]: if $V_oV_c = U\Sigma V^T$ is the SVD of V_oV_c with $\Sigma = \text{diag}(\sigma_1,\dots,\sigma_n)$, then $\sigma(W_cW_o) = \{\sigma_1^2,\dots,\sigma_n^2\}$ and the right eigenspace of W_cW_o is spanned by the columns of V_cV. Moreover, the SVD of V_oV_c can be computed without forming the product explicitly [11], thereby avoiding all problems caused by forming products of matrices and thus multiplying their condition numbers.

In [16] (for $E = I_n$) and [4] (for $E \neq I_n$), the iteration (11) was modified to obtain the Cholesky factors rather than the solutions themselves. The basic idea is that if $Q = C^TC$ and $C_0 = C$, the iterations for the symmetric matrices Q_k can be written in factored form as

$$
\begin{aligned}
Q_{k+1} &= C_{k+1}^TC_{k+1}\\
&= \frac{1}{2c_k}\begin{bmatrix} C_k \\ c_kC_kA_k^{-1}E \end{bmatrix}^T \begin{bmatrix} C_k \\ c_kC_kA_k^{-1}E \end{bmatrix},
\end{aligned}
$$

yielding

$$
C_{k+1} \leftarrow \frac{1}{\sqrt{2c_k}}\begin{bmatrix} C_k \\ c_kC_kA_k^{-1}E \end{bmatrix}. \tag{16}
$$

Analogous, for Iteration (13) with $B_0 := B$ we obtain

$$
B_{k+1} \leftarrow \frac{1}{\sqrt{2c_k}}[\,B_k,\ c_kEA_k^{-1}B_k\,]. \tag{17}
$$

Using (16), (17), the work space required to store the B_k's, C_k's is doubled in each iteration step. This can be avoided by computing in each iteration step an LQ factorization of B_{k+1} and a QR factorization of C_{k+1},

$$
B_{k+1} = [\,L_{k+1}\,0\,]U_{k+1}, \quad C_{k+1} = V_{k+1}\begin{bmatrix} R_{k+1} \\ 0 \end{bmatrix}.
$$

As

$$
B_{k+1}B_{k+1}^T = L_{k+1}L_{k+1}^T, \quad C_{k+1}^TC_{k+1} = R_{k+1}^TR_{k+1},
$$

it is sufficient to store the triangular factors in B_{k+1}, C_{k+1}. The orthogonal factors need not be accumulated, but still the amount of work required in each iteration step is increased by these factorizations. In [4] we therefore propose a compromise: first, fix the available workspace for the C_k's (here also for the B_k's). A reasonable amount is an $2n \times n$ array (or $n \times 2n$ for the B_k's) as the rank of the required Cholesky factors Y_1 (and X_1) cannot exceed n. Therefore, we may use (16) and (17) as long as $C_k \in \mathbb{R}^{p_k \times n}$ for $p_k \leq n$ and $B_k \in \mathbb{R}^{n \times m_k}$ for $m_k \leq n$, and then switch to working with the triangular factors obtained by the factorizations. Hence, we perform $k \leq \log_2 \frac{n}{m}$ iterations with (17) and $l \leq \log_2 \frac{n}{p}$ iterations with (16) before starting to compute factorizations in each step and work only with the triangular factors. If convergence is achieved before the switch, we have to compute final factorizations to obtain the Cholesky factors.

3 Implementation Details and Numerical Experiments

In order to accelerate the convergence, we employ in our algorithms the *determinantal scaling* [6, 7], where the scalar c_k in iteration (7) is set to

$$c_k \leftarrow |\det(Y^{-1} Z_k)|^{-1/r} \qquad (\text{if } Z, Y \in \mathbb{R}^{r \times r})$$

From the definition of Z and Y in (9), we have for iteration (14),

$$c_k \leftarrow |\det(E^{-1} A_k)|^{-1/n}.$$

Other choices of c_k are possible; for a comparison see [2].

From (10) and (12) we obtain $A_\infty = -E$. This suggests the stopping criterion

$$\|A_k + E\| \quad \leq \quad tol \cdot \|E\|,$$

for a suitable norm and a user-defined tolerance *tol*. This criterion is easy to check and does not require additional computations or workspace. As suggested in [3], in our implementations we employ the 1–norm for ease of computation, and $tol = 10 \cdot n \cdot \sqrt{\varepsilon}$ (ε denotes the machine precision). This technique avoids possible stagnation of the iteration due to ill-conditioning of the sign function matrix. Once the criterion is satisfied, due to the quadratic convergence of the Newton iteration, two additional iterations are usually enough to reach the attainable accuracy.

The experiments reported here demonstrate the computational performance of the investigated algorithms. The accuracy of Lyapunov equation solvers based on matrix sign function computations usually is as good as can be expected from the conditioning of the problem; see [4] for details and numerical results regarding accuracy.

All our numerical experiments were performed using Fortran 77 and IEEE double-precision arithmetic on an IBM SP3 platform ($\varepsilon \approx 2.2 \times 10^{-16}$). This platform consists of 80 RS6000 nodes at 120 MHz and 256 RAM MBytes per node. We made use of the vendor supplied BLAS (*essl*) and LAPACK libraries [1]. In our tests one node achieved around 200 Mflops (*Millions of floating-point arithmetic operations per second*) for the matrix product (routine `DGEMM`).

We construct both single-input single-output (SISO) problems ($m = p = 1$), and multiple-input multiple-output (MIMO) problems (with $n = m = p$). The matrix pencils (A, E) are generated with random entries, and eigenvalues uniformly distributed in $[-10, 0)$. The right-hand side matrices B and C are generated with random integer entries.

Our first experiment compares the performance of the coupled Lyapunov equation solvers based on direct methods (Bartels-Stewart's method [19] and Hammarling's algorithm [9]) and the methods based on the Newton iteration for the sign function as proposed in (14). We use the following notation for the solvers:

- `xxBS`: Bartels-Stewart's method.
- `xxHA`: Hammarling's algorithm.
- `xxNE`: Newton iteration for the explicit solutions.
- `xxNC`: Newton iteration for the Cholesky factors.

Characters (`xx`) specify whether the solver deals with the standard (`ST`) or the generalized (`GE`) equation. In all figures, we employ

- dashed lines for `STBS` ('$*$') and `GEBS` ('$\times$');
- dotted lines for `STHA` ('$*$') and `GEHA` ('$\times$');
- solid lines for `STNE` ('o') and `GENE` ('+');
- dashed/dotted lines for `STNC` ('o') and `GENC` ('+').

Figures 1 and 2 report, respectively, the execution time of the standard and generalized coupled Lyapunov solvers. The direct methods are slower than our iterative solvers based on the matrix sign function by a factor of up to 10. The number of iterations of the matrix sign function iterations was 10–15 in all our experiments. This is enough to achieve convergence in most cases. The performance of the solvers that compute the Cholesky factors strongly depends on the number of inputs/outputs of the system. For SISO systems, solvers `STNC` and `GENC` are more efficient. The cost of these solvers grows rapidly as m and p get larger. Note that many discretization schemes for distributed parameter systems yield SISO systems; see, e.g., [21].

We have also developed parallel codes, based on the matrix sign function iterations, for the coupled Lyapunov equations. Our solvers employ the ScaLAPACK parallel matrix algebra library [5].

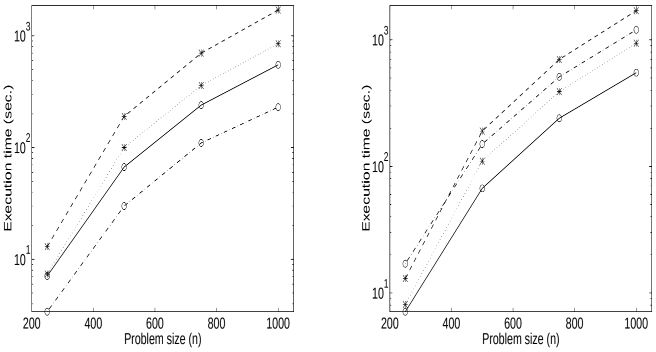

Figure 1: Execution time of the standard coupled Lyapunov equation solvers for SISO (left) and MIMO (right) systems.

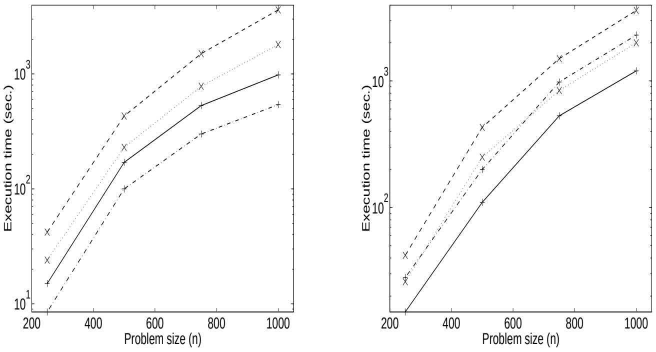

Figure 2: Execution time of the generalized coupled Lyapunov equation solvers for SISO (left) and MIMO (right) systems.

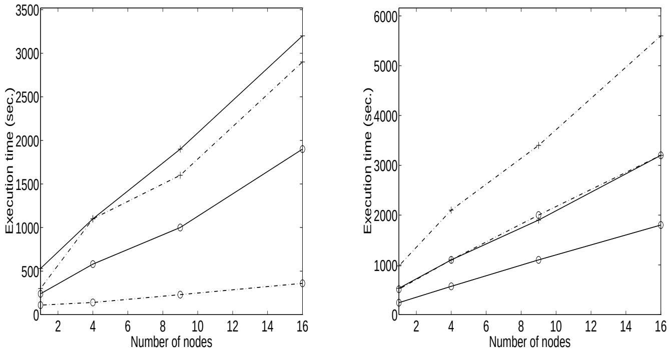

Figure 3: Execution time of the (parallel) coupled Lyapunov equation solvers for SISO (left) and MIMO (right) systems; n/q fixed at 750.

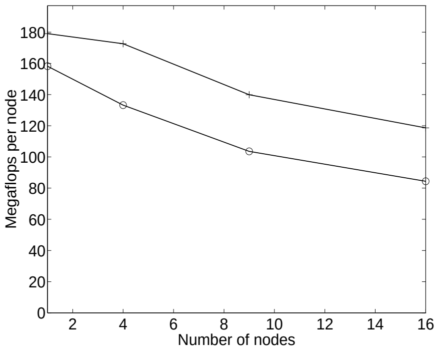

Figure 4: Mflops per node of the coupled Lyapunov equation solvers for MIMO systems; n/q fixed at 750.

We compare next the performance of the parallel solvers based on the matrix sign function. We do not include the direct methods in the comparison since the current version of ScaLAPACK does not provide the appropriate matrix kernels (e.g., the QZ algorithm).

In Figure 3 we report the execution time of the serial solvers and the parallel solvers on a $q \times q$, $q = 2, 3, 4$, mesh of nodes. In the experiment, we fix the ratio $n/q = 750$. The figure shows that, for SISO systems, the solvers for the Cholesky factor are more efficient, though they loose their efficiency in case the number of inputs/outputs is large.

Our final experiment analyzes the degree of parallelism of the solvers. For this purpose, we fix $n/q = 750$ (e.g., $n = 3000$ for $q = 4$) and compute the Mflops per node of the algorithms. We only report the results for algorithms `STNE` and `GENE` (see Figure 4). This ratio allows to measure the communication overhead of the parallel algorithms; for instance, the efficiency of algorithm `GENE` (compared to its serial version) is, respectively, 0.96, 0.78, and 0.66 on 4, 9, and 16 nodes. Compared to the serial routine for the matrix product (`DGEMM`), these solvers attain an efficiency of 0.86, 0.69, and 0.59 on 4, 9, and 16 nodes.

4 Concluding Remarks

We have presented numerical methods for the solution of coupled Lyapunov equations by means of matrix sign function iterations. Our algorithms can be employed to obtain explicit solutions of the coupled Lyapunov equations, their Cholesky factors, or the Gramians of an associated linear time-invariant system.

The numerical results show that the solvers based on the matrix sign function outperform traditional direct methods based on reduction to condensed forms on serial architectures. Moreover, the experimental results on a parallel distributed architecture show the efficiency of the algorithms and report this method as an efficient approach for solving large-scale problems.

Acknowledgements. We want to express our gratitude to the Argonne National Laboratory for the use of the IBM SP3.

References

[1] E. Anderson et al., *LAPACK Users' Guide*, SIAM, Philadelphia, PA, 2nd ed., 1994.

[2] Z. Bai and J. Demmel, *Design of a parallel nonsymmetric eigenroutine toolbox, Part I*, in Proceedings of the Sixth SIAM Conference on parallel Processing for Scientific Computing, R. S. et al, ed., 1993.

[3] P. Benner, *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*, Logos–Verlag, Berlin, Germany, 1997.

[4] P. Benner and E. S. Quintana-Ortí, *Solving stable generalized Lyapunov equations with the matrix sign function*, Tech. Rep. SFB393/97-23, Fak. f. Mathematik, TU Chemnitz, 09107 Chemnitz, FRG, 1997.

[5] L. S. Blackford et al., *ScaLAPACK Users' Guide*, SIAM, Philadelphia, PA, 1997.

[6] R. Byers, *Solving the algebraic Riccati equation with the matrix sign function*, Linear Algebra Appl., 85 (1987), pp. 267–279.

[7] J. Gardiner and A. Laub, *A generalization of the matrix-sign-function solution for algebraic Riccati equations*, Internat. J. Control, 44 (1986), pp. 823–832.

[8] J. D. Gardiner, *Stabilizing control for second-order models and positive real systems*, AIAA J. Guidance, Dynamics and Control, 15 (1992), pp. 280–282.

[9] S. J. Hammarling, *Newton's method for solving the algebraic Riccati equation*, NPL Report DITC 12/82, National Physical Laboratory, Teddington, Middlesex TW11 OLW, U.K., 1982.

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

[11] M. T. Heath, A. J. Laub, C. C. Paige, and R. C. Ward, *Computing the SVD of a product of two matrices*, SIAM J. Sci. Statist. Comput., 7 (1987), pp. 1147–1159.

[12] C. Kenney and A. Laub, *The matrix sign function*, IEEE Trans. Automat. Control, 40 (1995), pp. 1330–1348.

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

[14] P. Lancaster and M. Tismenetsky, *The Theory of Matrices*, Academic Press, Orlando, 2nd ed., 1985.

[15] W. Lang and U. Lezius, *Numerical realization of the balanced reduction of a control problem*, in Progress in Industrial Mathematics at ECMI94, H. Neunzert, ed., John Wiley & Sons Ltd and B.G. Teubner, New York and Leipzig, 1996, pp. 504–512.

[16] V. Larin and F. Aliev, *Construction of square root factor for solution of the Lyapunov matrix equation*, Sys. Control Lett., 20 (1993), pp. 109–112.

[17] V. Mehrmann, *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution*, no. 163 in Lecture Notes in Control and Information Sciences, Springer-Verlag, Heidelberg, July 1991.

[18] B. C. Moore, *Principal component analysis in linear systems: Controllability, observability, and model reduction*, IEEE Trans. Automat. Control, AC-26 (1981), pp. 17–32.

[19] T. Penzl, *Numerical solution of generalized Lyapunov equations*, to appear in Adv. Comp. Math., 1997.

[20] J. Roberts, *Linear model reduction and solution of the algebraic Riccati equation by use of the sign function*, Internat. J. Control, 32 (1980), pp. 677–687.

[21] I. Rosen and C. Wang, *A multi–level technique for the approximate solution of operator Lyapunov and algebraic Riccati equations*, SIAM J. Numer. Anal., 32 (1995), pp. 514–541.

[22] M. G. Safonov and R. Y. Chiang, *Model reduction for robust control: A Schur relative error method*, Int. J. Adapt. Cont. and Sign. Proc., 2 (1988), pp. 259–272.

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