Zentrum für Technomathematik
Fachbereich 3 – Mathematik und Informatik

Numerical Solution of Schur Stable Linear Matrix Equations on Multicomputers

Peter Benner Enrique S. Quintana-Ortí
Gregorio Quintana-Ortí

Report 99–13

Berichte aus der Technomathematik

Report 99–13 November 1999

Numerical Solution of Schur Stable Linear Matrix Equations on Multicomputers*

Peter Benner† Enrique S. Quintana-Ortí‡ Gregorio Quintana-Ortí§

November 8, 1999

Abstract

We investigate the parallel performance of numerical algorithms for solving discrete Sylvester and Stein equations as they appear for instance in discrete-time control problems, filtering, and image restoration. We assume that the coefficient matrices of the equations are stable with respect to the unit circle. The methods used here are the squared Smith iteration and the sign function method on a Cayley transformation of the original equation. For Stein equations with semidefinite right-hand side these methods are modified such that the Cholesky factor of the solution can be computed directly without forming the solution matrix explicitly. We report experimental results of these algorithms on distributed-memory multicomputers.

1 Introduction

We study the numerical solution of the *discrete Sylvester equation*

$$AXB - X + C = 0, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$, $C \in \mathbb{R}^{n\times m}$, and $X \in \mathbb{R}^{n\times m}$ is the sought-after solution of (1). A particular instance of equation (1) is the *Stein equation*, often also referred to as *discrete Lyapunov equation*, which is a symmetric version of (1) and takes the form

$$AXA^T - X + C = 0, \tag{2}$$

where $A, C \in \mathbb{R}^{n\times n}$, $C = C^T$, and $X \in \mathbb{R}^{n\times n}$. That is, (2) results from (1) by setting $m = n$, $B = A^T$ and $C = C^T$. It easily follows that if there exists a unique solution X to (2), then this solution has to be symmetric.

Equations of the form (1), (2) play a fundamental role in linear control and filtering theory for discrete-time systems; see, e.g., [32, 29, 37]. Equations of the form (1) also have to be solved in methods for restoration of noisy images; see, e.g., [13]. Throughout this paper we

*Partially supported by the DAAD programme *Acciones Integradas Hispano-Alemanas.* Enrique S. Quintana-Ortí and Gregorio Quintana-Ortí were also supported by the Spanish CICYT Project TIC96-1062-C03-03.

†Zentrum für Technomathematik, Fachbereich 3 – Mathematik und Informatik, Universität Bremen, D–28334 Bremen, Germany; e-mail: benner@math.uni-bremen.de.

‡Departamento de Informática, Universidad Jaime I, 12080 Castellón, Spain; e-mail: quintana@inf.uji.es.

§Departamento de Informática, Universidad Jaime I, 12080 Castellón, Spain; e-mail: gquintan@inf.uji.es.

assume equations (1) and (2) to be *Schur stable*, that is, if $\rho(A)$ denotes the spectral radius of a square matrix A, then we assume $\rho(A) < 1$ and $\rho(B) < 1$. In case of the Stein equation this is equivalent to the usual Schur stability of the matrix A, i.e., $\sigma(A) \subset \{z \in \mathbb{C} : |z| < 1\}$, where $\sigma(A)$ denotes the spectrum of A. Defining the operator $\mathrm{vec} : \mathbb{R}^{k\times\ell} \to \mathbb{R}^{k\ell}$ mapping any $M \in \mathbb{R}^{k\times\ell}$ to

$$\mathrm{vec}(M) = [\, m_{11}, m_{21}, \ldots, m_{k,1}, m_{12}, \ldots, m_{k,2}, \ldots, m_{k,\ell}\,]^T, \tag{3}$$

we can rewrite (1) as a system of linear equations

$$(B^T \otimes A - I_{nm})\mathrm{vec}(X) = -\mathrm{vec}(C). \tag{4}$$

Here, $\otimes$ denotes the *Kronecker product*; see, e.g., [30] for details. If we set $W := B^T \otimes A - I_{nm}$, $x := \mathrm{vec}(X)$, and $c := -\mathrm{vec}(C)$ then (4) can be written in the standard form

$$Wx = c \tag{5}$$

and the spectrum of the coefficient matrix W is given by

$$\sigma(W) = \{\lambda_i\mu_j - 1 \,|\, \lambda_i \in \sigma(A), \mu_j \in \sigma(B),\ i = 1, \ldots, n,\ j = 1, \ldots, m\}.$$

From the stability assumption used in this paper it follows that W is nonsingular. Hence (5) and thereby (4), (1) as well as the Stein equation (2) have unique solutions; see, e.g., [30].

We will also consider the *generalized Stein equation*

$$AXA^T - EXE^T + C = 0, \tag{6}$$

with A, X, C as above and $E \in \mathbb{R}^{n\times n}$ nonsingular. This equation and its solution inherits all its mathematical properties from (2) as it is equivalent to (2) if pre-multiplied with E^{-1} and post-multiplied by E^{-T}. Such a transformation may introduce large rounding errors into the data if E is ill-conditioned and should be avoided for practical computations if possible. The equation (6) is Schur stable if the matrix pencil $A - \lambda E$ has all its eigenvalues in the open unit disk, i.e., under the given assumptions if $E^{-1}A$ is Schur stable.

Schur stable equations of the form (1) arise for instance in the design of optimal regulators for the reference tracking problem (see, e.g., [37]) and in model reduction methods (see, e.g., [36]). (Generalized) Stein equations with Schur stable coefficient matrix A (or matrix pencil $A - \lambda E$) appear in many computational problems for linear control systems. For example, such an equation has to be solved in each step of Newton's method for (generalized) discrete-time algebraic Riccati equations [25, 32]. Moreover, the *controllability Gramian* W_c and *observability Gramian* W_o of a discrete-time linear time-invariant (LTI) system of the form

$$x_{k+1} = Ax_k + Bu_k, \quad y_k = Cx_k, \qquad k = 0, 1, 2, \ldots, \qquad x_0 = x^{(0)},$$

are given by the solutions of the corresponding Stein equations

$$AW_cA^T - W_c + BB^T = 0, \qquad A^TW_oA - W_o + C^TC = 0. \tag{7}$$

The Gramians of LTI systems play a fundamental role in many analysis and design problems for LTI systems such as computing balanced, minimal, or partial realizations, the Hankel singular values, the Hankel norm or H_2-norm of the system, and model reduction. Often,

the *Cholesky factors* of the solutions to the above equations are needed, i.e., one is interested in computing the factors in the Cholesky factorizations $W_c = LL^T$ and $W_o = R^T R$. Hence, special algorithms are designed to compute these factors without ever forming the solution matrices explicitly.

The need for parallel computing in this area can be seen from the fact that already for a system with state-space dimension $n = 1000$, the corresponding Stein equations represent a set of linear equations with about half a million unknowns (exploiting the symmetry of X). Moreover, in some applications as the reference tracking problem mentioned above, often on-line solutions of these equations are required. Even for small state-space dimension n this means that a system of linear equations with a few thousand unknowns has to be solved in a few milli- or even microseconds. We assume here that the coefficient matrices are dense. If sparsity of matrices is to be exploited, other computational techniques have to be employed.

The standard methods for solving equations (1) or (2) are based on the Bartels-Stewart method or Hessenberg-Schur methods; see [3, 19, 37]. These algorithms in a first stage reduce the coefficient matrices A and B to quasi-upper triangular form by the QR algorithm. (For the Hessenberg-Schur method, B needs only to be transformed to upper Hessenberg form.) Hence, in order to use these methods on parallel computers, it is necessary to have an efficient parallelization of the QR algorithm. However, several experimental studies report the difficulties in parallelizing the double implicit shifted QR algorithm on parallel distributed multiprocessors; see, e.g., [11, 18, 23, 39]. The algorithm presents a fine granularity which introduces a loss of performance due to communication start-up overhead (latency). Besides, traditional data layouts (column/row block scattered) lead to an unbalanced distribution of the computational load. A different approach relies on a block Hankel distribution, which improves the balancing of the computational load [23]. Attempts to increase the granularity by employing multishift techniques have been recently proposed in [12, 24]. Nevertheless, the parallelism and scalability of these algorithms are still far from those of matrix multiplications, matrix factorizations, triangular linear systems solvers, etc.; see, e.g., [10] and the references given therein.

For the generalized Stein equation, the situation is even worse. If this equation is to be solved directly using a Bartels-Stewart type algorithm without transforming it to the standard case (2) as proposed in [16, 17, 34], then the QZ algorithm (see, e.g., [20]) has to be employed for the initial reduction of the matrix pair (A, E) to generalized Schur form. We are not aware of any attempt to parallelize the QZ algorithm. From the above considerations and taking into account the even more complex computational work in this algorithm, even worse parallelism and scalability than for the QR algorithm are to be expected.

For these reasons we will use methods that are purely based on easy to parallelize computational kernels. These are the squared Smith iteration applied to (1) or (2) and the sign function method applied to the Sylvester or (generalized) Lyapunov equations resulting from a Cayley transformation of (1), (2), or (6), respectively. The employed algorithms will be reviewed in Section 2. In Section 3 we show how to use the squared Smith iteration in order to compute the Cholesky factor of the solution of a semidefinite Stein equation without computing the solution matrix explicitly. The algorithms considered here are implemented using the kernels in libraries BLACS, PBLAS, and ScaLAPACK [10]. This ensures the portability of the codes across a wide variety of platforms for distributed memory computing. The computational performance of the implemented algorithms with respect to accuracy, execution time as well as scalability will be reported in Section 4. Some final remarks are given in Section 5.

2 Numerical Algorithms

2.1 The Smith Iteration

Consider (1) and recall that (2) is a special instance of (1). We can rewrite equation (1) in fixed point form, $X = AXB + C$, and form the fixed point iteration

$$X_0 := C, \qquad X_{k+1} = C + AX_kB, \quad k = 0, 1, 2, \ldots.$$

Then this iteration converges to X if $\rho(A)\rho(B) < 1$, i.e., convergence is guaranteed under the given assumptions. The convergence rate of this iteration is linear. A quadratically convergent version of this fixed point iteration is suggested in [14, 38]. Setting $X_0 := C$, $A_0 := A$, and $B_0 := B$, this iteration can be written as

$$\begin{array}{lll} A_0 := A, & B_0 := B, & X_0 := C, \\ X_{k+1} := A_kX_kB_k + X_k, & & \\ A_{k+1} := A_k^2, & B_{k+1} := B_k^2, & \end{array} \qquad k = 0, 1, 2, \ldots. \tag{8}$$

The above iteration is referred to as the *squared Smith iteration*. In case (2) is to be solved with the Smith iteration, the iterative scheme given in (8) simplifies as no sequence of B_k has to be computed.

The most appealing feature of the squared Smith iteration regarding its parallel implementation is that all the computational cost comes from matrix products. These are known to be highly parallelizable; see, e.g., [10].

The convergence theory of the Smith iteration derived in [38] yields that for $\rho(A)\rho(B) < 1$ there exist real constants $0 < \mu$ and $0 < \rho < 1$ such that

$$\|X - X_k\|_2 \leq \mu\|C\|_2(1-\rho)^{-1}\rho^{2^k}. \tag{9}$$

This shows that the method converges for all equations with Schur stable coefficient matrices A and B. One of them may even be only *almost stable*, i.e., have eigenvalues on the unit circle. Nevertheless, if any of the coefficient matrices A or B is highly non-normal such that $\|A\|_2 > 1$ or $\|B\|_2 > 1$, then overflow may occur in the early stages of the iteration due to increasing $\|A_k\|_2$ or $\|B_k\|_2$ although eventually, $\lim_{k\to\infty} A_k = 0$ and $\lim_{k\to\infty} B_k = 0$ if $\rho(A) < 1$ and $\rho(B) < 1$.

We can derive a conservative bound on the number of iteration steps of the squared Smith iteration before overflow can occur as follows. Let $\alpha := \|A\|_2$, $\beta := \|B\|_2$. We can scale the equation such that $\|C\|_2 = 1$ by setting $\tilde{C} = C/\|C\|_2$ and $\tilde{X} = X/\|C\|_2$. As for any matrix M, $\max_{i,j} |m_{i,j}| \leq \|M\|_2$ [20, Chapter 2], overflow does not occur as long as $\|X_k\| < r_{\max}$, where $r_{\max}$ is the overflow threshold of the arithmetic used. (For example, in IEEE double precision arithmetic, $\log_2(r_{\max}) = 1024$.) It is easy to see that

$$X_k = \sum_{j=0}^{2^k-1} A^jCB^j. \tag{10}$$

Overflow will occur if one of the terms A^jCB^j overflows. It is rather unlikely that the sum will overflow if all the terms remain bounded below $r_{\max}$. As we assume $\|C\|_2 = 1$, we can derive

bounds from the highest order term in (10) requiring $\|A_k\|_2 \leq \sqrt{r_{\max}}$ and $\|B_k\|_2 \leq \sqrt{r_{\max}}$. Hence, for $\alpha \geq 1$, $\beta \geq 1$ we obtain the bounds

$$k \;<\; \min\left\{\log_2\left(\frac{\log_2(r_{\max})}{2(1+\log_2(\alpha))}\right), \log_2\left(\frac{\log_2(r_{\max})}{2(1+\log_2(\beta))}\right)\right\}. \tag{11}$$

Note that the bounds in (11) may be very conservative. On the other hand, (11) only requires *a priori* knowledge and can thus be computed before staring the iteration. As the 2-norm computation is in general too expensive, other norms have to be employed if the bounds from (11) are to be checked. The Frobenius norm can be used without changes if α, β are replaced by $\|A\|_F$, $\|B\|_F$, respectively (where the scaling has then also to be performed using the Frobenius norm). Using $\|M\|_2^2 \leq \|M\|_1\|M\|_\infty$ [20, Chapter 2], (11) transforms to

$$k \;<\; \min\left\{\log_2\left(\frac{\log_2(r_{\max})}{1+\log_2(\|A\|_1\|A\|_\infty)}\right), \log_2\left(\frac{\log_2(r_{\max})}{1+\log_2(\|B\|_1\|B\|_\infty)}\right)\right\}. \tag{12}$$

Another way to get an easy computable bound on the element growth in A^{2^k} is via the *departure from normality* which is defined as (see [20, Chapter 7.1])

$$\eta_A := \eta(A) = \sqrt{\|A\|_F^2 - \operatorname{trace}(A^2)}. \tag{13}$$

For any Schur decomposition $A = Q(D+N)Q^H$ with unitary Q, diagonal matrix of eigenvalues D, and nilpotent, strictly upper triangular matrix N, we have $\|A\|_2 \leq \|A\|_F = \|D+N\|_F$ due to the unitary invariance of the Frobenius and spectral norms. Moreover, $\|N\|_F = \eta_A$ regardless of the choice of Q (see [20, Chapter 7.1]). Using $N^j = 0$ for all $j \geq n-1$ we obtain

$$\|A^{2^k}\|_2 \leq \|A^{2^k}\|_F \leq \|A\|_F^{2^k} = \|D+N\|_F^{2^k} \leq \sum_{j=0}^{\min\{n-1,2^k\}} \binom{2^k}{j} \eta_A^j. \tag{14}$$

An analogous expression gives a bound for the growth in B. The right-hand side expression is easy to evaluate and can hence be checked without significant effort as A^2 has to be computed in the first iteration step anyway. The bounds implied by (14) can be checked in each iteration and the iteration can be stopped if they indicate that $\|A^{2^k}\|_2$ or $\|B^{2^k}\|_2$ will overflow in the next step.

Note that all these bounds do not take rounding errors into account. In case the spectral radii of A or B are close to one it can happen that rounding errors will cause $\rho(A_k) > 1$ or $\rho(B_k) > 1$ and hence the whole convergence theory breaks down. In that case, overflow may occur even if the bounds derived above give no indication for this. This usually happens only if there are *defective* eigenvalues very close to one and almost never occurs in practice. (In control theory this means that the underlying system is highly sensitive to perturbations and can therefore be considered as ill designed.)

The spectral norm (and other norms as well) of a matrix and its departure from normality can sometimes be significantly be reduced using balancing [33, 20]. A balancing procedure computes a diagonal scaling matrix D such that the rows and columns of $\tilde{A} := D^{-1}AD$ have almost equal 1-norms. If $\tilde{C} := DCD$ and $\tilde{X} := DXD$, then the solution of (2) can be retrieved from the solution of the equivalent Stein equation $\tilde{A}^T\tilde{X}\tilde{A} - \tilde{X} + \tilde{C} = 0$. This approach usually yields no advantage for the discrete Sylvester equation (1) as the simultaneous balancing of A and B is seldom possible.

If the generalized Stein equation (6) is to be solved by the Smith iteration, one has to invert either A or E in order to transform it to a standard equation as in (2). We therefore consider this approach not any further and propose in this case to use the sign function method which will be reviewed in the next section.

2.2 The Sign Function Method

The sign function method was first introduced in 1971 by Roberts [35] for solving algebraic Riccati equations. Roberts also shows how to solve *Sylvester equations* of the form

$$\hat{A}X - X\hat{B} + \hat{C} = 0, \qquad \hat{A} \in \mathbb{R}^{n\times n},\ \hat{B} \in \mathbb{R}^{m\times m}\ \hat{C} \in \mathbb{R}^{n\times m}, \tag{15}$$

via the matrix sign function if both $\sigma(\hat{A})$ and $\sigma(-\hat{B})$ are contained in the open left half complex plane, i.e., if $\hat{A}$ and $-\hat{B}$ are *Hurwitz stable*.

Let $Z \in \mathbb{R}^{n\times n}$ have no eigenvalues on the imaginary axis and denote by $Z = S \begin{bmatrix} J^- & 0 \\ 0 & J^+ \end{bmatrix} S^{-1}$ its Jordan decomposition with $J^- \in \mathbb{C}^{k\times k}$, $J^+ \in \mathbb{C}^{(n-k)\times(n-k)}$ containing the Jordan blocks corresponding to the eigenvalues in the open left and right half planes, respectively. Then the *matrix sign function* of Z is defined as $\operatorname{sign}(Z) := S \begin{bmatrix} -I_k & 0 \\ 0 & I_{n-k} \end{bmatrix} S^{-1}$. Note that $\operatorname{sign}(Z)$ is unique and independent of the order of the eigenvalues in the Jordan decomposition of Z. Many other equivalent definitions for $\operatorname{sign}(Z)$ can be given. For more details see, e.g., the survey paper [28].

The sign function can be computed via the Newton iteration for the equation $Z^2 = I$ where the starting point is chosen as Z, i.e.,

$$Z_0 := Z, \qquad Z_{k+1} := \left(Z_k + Z_k^{-1}\right)/2, \qquad k = 0, 1, 2, \ldots. \tag{16}$$

It is shown in [35] that $\operatorname{sign}(Z) = \lim_{k\to\infty} Z_k$ and moreover that

$$\operatorname{sign}\left(\begin{bmatrix} \hat{A} & \hat{C} \\ 0 & \hat{B} \end{bmatrix}\right) + I_{n+m} = 2\begin{bmatrix} 0 & X \\ 0 & I \end{bmatrix}, \tag{17}$$

i.e., under the given assumptions, (15) can be solved by applying the iteration (16) to $Z_0 := \begin{bmatrix} \hat{A} & \hat{C} \\ 0 & \hat{B} \end{bmatrix}$. The same algorithm was later again derived in [4, 27].

The computation of the sign function via (16) only requires basic numerical linear algebra tools like matrix multiplication, inversion and/or solution of linear systems. These computations are implemented efficiently on most parallel architectures and, in particular, ScaLAPACK [10] provides easy to use and portable computational kernels for these operations. Hence, the sign function method is an appropriate tool to design and implement efficient and portable numerical software for distributed memory parallel computers.

If we apply the *Cayley transformation*

$$c(A) = (A - I_n)^{-1}(A + I_n) \tag{18}$$

to A from (2), then the Stein equation (2) is equivalent to the Lyapunov equation (15) with $\hat{A} = -\hat{B}^T = c(A)$ and $\hat{C} = 2(A - I_n)^{-1}C(A - I_n)^{-T}$. The transformation of the discrete Sylvester equation (1) to a (continuous) Sylvester equation as in (15) is achieved analogously by setting $\hat{A} := c(A)$, $\hat{B} := -(B + I_m)(B - I_m)^{-1}$, and $\hat{C} := 2(A - I_n)^{-1}C(B - I_m)^{-1}$. Hence the discrete Sylvester and Stein equations can be solved by applying the sign function

iteration to the Sylvester and Lyapunov equations, respectively, resulting from the Cayley transformation.

In [35] it is observed that applying the Newton iteration (16) to the matrix $\left[\begin{smallmatrix} \hat{A} & \hat{C} \\ 0 & \hat{B} \end{smallmatrix}\right]$ and exploiting the block-triangular structure of all matrices involved, (16) boils down to

$$\begin{aligned} A_0 &:= \hat{A}, & A_{k+1} &:= \tfrac{1}{2}\left(A_k + A_k^{-1}\right), \\ B_0 &:= \hat{B}, & B_{k+1} &:= \tfrac{1}{2}\left(B_k + B_k^{-1}\right), \qquad k = 0, 1, 2, \ldots, \\ C_0 &:= \hat{C}, & C_{k+1} &:= \tfrac{1}{2}\left(C_k + A_k^{-1} C_k B_k^{-1}\right), \end{aligned} \tag{19}$$

and hence from (17) it follows that $X = \frac{1}{2}\left(\lim_{k\to\infty} C_k\right)$. For the Cayley-transformed Stein equation, the iteration for the B_k's equals the one for the A_k's and hence can be omitted. Moreover, in that case all C_k's are symmetric as $B_k = A_k^T$ which can also be exploited to save some arithmetic and work space. We will compare the Smith iteration for (1) and (2) to the iteration (19) applied to the Cayley-transformed equations in Section 4.

Other iterative schemes for computing the sign function like the Newton-Schulz iteration or Halley's method (see, e.g., [28]) can also be implemented efficiently to solve Sylvester and Lyapunov equations and can therefore also be used for Cayley-transformed discrete Sylvester and Stein equations; details of the resulting algorithms are reported in [8].

The generalized Stein equation (6) can be transformed to a Lyapunov equation via the *generalized Cayley transformation*

$$c(A, E) = ((\mu A + E), (A - \mu E)), \qquad |\mu| = 1.$$

In order to keep computations real, the shift parameter μ must satisfy $\mu = \pm 1$. It is well known (see, e.g., [15]) that for a Schur stable matrix pencil $A - \lambda E$, both $(\mu A + E)$ and $(A - \mu E)$ are nonsingular and $(A - \mu E)^{-1}(\mu A + E)$ is Hurwitz stable. Hence we use in the following without loss of generality $\mu = 1$. Now let $\tilde{A} := A + E$, $\tilde{E} := A - E$, and $\tilde{C} := 2C$. Then, (6) is equivalent to

$$\tilde{A} X \tilde{E}^T + \tilde{E} X \tilde{A}^T + \tilde{C} = 0. \tag{20}$$

This is a Hurwitz stable generalized Lyapunov equation with the same solution as (6). Note that no matrix inverses or multiplications are needed to derive (20) from (6). The numerical solution of generalized Lyapunov equations is considered in [7]. It is shown there that the generalized sign function iteration of [15] applied to (20) simplifies to

$$\begin{aligned} A_0 &\leftarrow \tilde{A}, & A_{k+1} &\leftarrow \tfrac{1}{2}\left(A_k + \tilde{E} A_k^{-1} \tilde{E}\right), \\ C_0 &\leftarrow \tilde{C}, & C_{k+1} &\leftarrow \tfrac{1}{2}\left(C_k + \tilde{E}^T A_k^{-T} C_k A_k^{-1} \tilde{E}\right), \end{aligned} \qquad \text{for } k = 0, 1, 2, \ldots \tag{21}$$

and that $X = \frac{1}{2}\tilde{E}^{-T}\left(\lim_{k\to\infty} C_k\right)\tilde{E}^{-1}$. Hence we can solve (6) by applying the generalized Cayley transformation to the coefficient matrices and using the algorithms and implementations reported in [5, 7]. The generalized Stein equation (6) can also be reduced further to a standard Lyapunov equation, i.e., $B = -A^T$ in (15), by multiplying (2) from the left with $\tilde{E}^{-1}$ and from the right by $\tilde{E}^{-T}$. The resulting Lyapunov equation with $\hat{A} = (A - E)^{-1}(A + E)$ and $\hat{C} = 2(A - E)^{-1} C (A - E)^{-T}$ can then be solved using the standard Newton iteration (19).

2.3 Iterative Refinement

The computed solution of a system of linear equations can often be improved by iterative refinement; see, e.g., [20, Section 3.53]. As the discrete Sylvester and (generalized) Stein equations represent also systems of linear equations in nm and $\frac{n(n+1)}{2}$,respectively, unknowns, this can be used here as well to refine an approximate solution.

Let $\tilde{X}$ be an approximate solution to (1), (2), or (6) and define the *defect* $N := X - \tilde{X}$ as well as the *residual* R of the respective equation. Then N satisfies the corresponding defect equations

$$\begin{aligned} ANB - N &= R := A\tilde{X}B - \tilde{X} + C, && (22)\\ ANA^T - N &= R := A\tilde{X}A^T - \tilde{X} + C, && (23)\\ ANA^T - ENE^T &= R := A\tilde{X}A^T - E\tilde{X}E^T + C. && (24) \end{aligned}$$

These equations can then again be solved via any of the methods proposed above where a combination of the methods is possible, i.e., the approximation $\tilde{X}$ and the defect N can be computed using different methods. A (hopefully) improved solution of the original equation is then obtained by $X := \tilde{X} + N$.

Note that for the methods considered here, the convergence rate only depends on the spectrum of A, B, or $A - \lambda E$ and hence the same number of iterations needed to solve the original equation is required to solve the defect equation. In other words, iterative refinement doubles the cost of the computation in contrast to standard linear systems where for instance, the LU decomposition of the coefficient matrix can be reused when solving the defect equation [20, Section 3.5.3].

Therefore and as an improvement is not to be expected if $\tilde{X}$ is sufficiently close to the exact solution, iterative refinement should only be used if the computed solution is not accurate enough. This can be decided, using, e.g., the *relative residual* defined for the equations under consideration (see [26, Section 15.2]) as

$$\begin{aligned} r_r &:= \frac{\|R\|}{\|A\|\|B\|\|\tilde{X}\| + \|\tilde{X}\| + \|C\|} && \text{(for (1))}, && (25)\\ r_r &:= \frac{\|R\|}{\|A\|\|A^T\|\|\tilde{X}\| + \|\tilde{X}\| + \|C\|} && \text{(for (2))}, && (26)\\ r_r &:= \frac{\|R\|}{(\|A\|\|A^T\| + \|E\|\|E^T\|)\|\tilde{X}\| + \|C\|} && \text{(for (6))}, && (27) \end{aligned}$$

where R denotes the corresponding residuals as defined in (22)–(24). As for a numerically backward stable method, the relative residual should be close to the machine precision ε (see [26, Section 1.10] and the references given there), one should require for the approximate solution that $r_r \leq c\varepsilon$ where c is a constant depending mildly on the dimensions n, m and also on the chosen norm. For instance, for (2), $c = 10\sqrt{n}$ seems to work very well. Note that if $\tilde{X}$ yields already a small relative residual, then iterative refinement will usually only improve the computed solution if the residual is computed with extended precision, see [20, Section 3.53].

3 Solving Semidefinite Stein Equations for the Cholesky Factor

If the "right-hand side" C of the generalized Stein equation (6) is positive semidefinite, then so is the solution matrix X and hence can be factored as $X = LL^T$. We can therefore write (6) as

$$ALL^TA^T - ELL^TE^T + BB^T = 0, \tag{28}$$

where $B \in \mathbb{R}^{p\times n}$ with $BB^T = C$. (For (2), set $E = I_n$ in (28).) The factor $L \in \mathbb{R}^{n\times \ell}$ is called the Cholesky factor of the solution. Usually, $\ell = n$ such that L is a square, possibly singular, matrix [21, 22, 40]. Here we will also use "Cholesky factor" to denote a full rank factor of the solution, i.e., $\operatorname{rank}(L) = \operatorname{rank}(X) = \ell \leq n$. This has several advantages

The equation (28) is called a *semidefinite generalized Stein equation*. In many applications, the Cholesky factor L of X is required rather than the solution X itself, e.g., most model reduction algorithms for discrete-time systems based on the system Gramians as defined in (7) use their Cholesky factors, see [36] and the references given therein. Hammarling's algorithm [21, 22, 34, 40] computes this factor without forming the product BB^T or 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 L. Hence, a significant increase in accuracy can often be observed working with L instead of X if X is ill-conditioned. Moreover, if $\ell \ll n$, usually significant savings in computational work is obtained by using the full-rank Cholesky factor rather than X for subsequent computations [9].

The sign function method applied to the (generalized) Lyapunov equation resulting from the Cayley-transformation of the (generalized) Stein equation can be modified to compute the (full-rank) Cholesky factor of X directly; see [31, 5, 7, 9]. Here we will show that this is also possible (and even simpler) for the squared Smith iteration. As the generalized Stein equation has to be rewritten as a standard Stein equation in order to apply the iteration (8), we will focus here on equations of the form (2), i.e., set $E = I_n$ in (28).

Setting $C = BB^T$, the iteration for X in (8) can be re-written as

$$\begin{aligned} L_0 &\leftarrow B, \\ L_{k+1}L_{k+1}^T &\leftarrow L_kL_k^T + A_k(L_kL_k^T)A_k^T = [\,L_k,\ A_kL_k\,] \begin{bmatrix} L_k^T \\ L_k^TA_k^T \end{bmatrix}, \quad \text{for } k = 0, 1, 2, \ldots. \end{aligned} \tag{29}$$

In each step of the resulting algorithm the current iterate L_k is augmented by A_kL_k such that

$$L_{k+1} := [\,L_k,\ A_kL_k\,]$$

The computational cost for the k-th iteration step of such a procedure is $2(2^kp)n^2$, where p is the number of columns of B. This compares to $3n^3$ flops for each iteration step for the L_k in (8).

The above approach requires to double in each iteration step the workspace needed for the iterates L_k. We will outline two approaches that limit the required workspace to a fixed size.

As the rank of the solution X of (28) and hence of its Cholesky factor R can not be predicted by the rank of B, the implementation of Hammarling's algorithm in [34] requires a work array of dimension at least $n \times n$ for B if it is supposed to be overwritten by R. This

suggests to use (29) only as long as $2^k p$ is less than $n/2$ which is also the bound for which the original iteration (8) becomes cheaper than (29). This bound is given by

$$k \;>\; \left\lfloor \log_2 \frac{n}{p} \right\rfloor, \tag{30}$$

where $\lfloor\, x \,\rfloor$ denotes the integer part of x.

If k has reached the bound given above (which is the case for $k = 0$ if $p > n/2$), we propose to form the augmented matrix $\tilde{L}_{k+1} = [\, L_k,\; A_k L_k \,] \in \mathbb{R}^{n \times 2\ell_k}$, where $L_k \in \mathbb{R}^{n \times \ell_k}$ with $\ell_0 = p$. Then compute its LQ factorization

$$\tilde{L}_{k+1} \;=\; [\, \underbrace{\hat{L}_{k+1}}_{\ell_{k+1}},\; \underbrace{0}_{2\ell_k - \ell_{k+1}} \,] U_{k+1},$$

where $\ell_{k+1} := \operatorname{rank}(\tilde{L}_{k+1})$. It follows that $L_{k+1} L_{k+1}^T := \tilde{L}_{k+1} \tilde{L}_{k+1}^T = \hat{L}_{k+1} \hat{L}_{k+1}^T$ and hence we can set $L_{k+1} := \hat{L}_{k+1}$. Note that in order to obtain the Cholesky factor of X, a QR factorization of L_{k+1} has to be computed at convergence even if k does not reach the bound in (30). A similar, but less efficient version of this iteration is considered in [1, Section 3.1]. In order to determine the rank of $\tilde{L}_{k+1}$ correctly, it is more reasonable to employ a LQ factorization with row pivoting or even a rank-revealing LQ factorization (see, e.g., [20, Chapter 5] and the references therein). In that case L_{k+1} is obtained as the left $n \times \ell_{k+1}$ part of the product of a permutation matrix Π_{k+1} and the lower triangular matrix L_{k+1}, i.e.,

$$\begin{bmatrix} (\Pi_{k+1})_{11} & (\Pi_{k+1})_{12} \\ (\Pi_{k+1})_{21} & (\Pi_{k+1})_{22} \end{bmatrix} \begin{bmatrix} (\hat{L}_{k+1})_{11} & 0 \\ (\hat{L}_{k+1})_{21} & 0 \end{bmatrix} \;=\; [\, L_{k+1},\; 0 \,].$$

An alternative approach is to compute a rank-revealing LQ factorization of the augmented matrix in each iteration step, starting from L_0 obtained by a rank-revealing LQ factorization of B. As $\operatorname{rank}(X)$ may be up to n, this requires a work space of size up to $2n \times n$. On the other hand, this approach is preferable when the semidefinite Lyapunov equation is solved during some model reduction algorithms for large-scale systems. As the (numerical) rank of the Cholesky factors is then often much smaller than n, the additional work of performing LQ factorizations is well counter-balanced by keeping the number of columns of L_{k+1} small. This approach can also be used to compute low-rank approximations to the full-rank factor by either increasing the tolerance threshold for determining the numerical rank or by fixing the allowed number of columns in L_k.

Remark 3.1 *There is a possibility that in practice, the iteration (29) terminates but we have* $\operatorname{rank}(L_k) < \operatorname{rank}(X)$. *Suppose that A_k has converged to zero before L_k has converged to L (and hence, before we have achieved* $\operatorname{rank}(L_k) = \operatorname{rank}(X)$*). Then the second part of the iteration (29) will stagnate, i.e., L_k will not be changed in the subsequent iterations. But as from (9),*

$$\|X - X_k\| = \|LL^T - L_k L_k^T\| \to 0 \qquad \text{for} \quad k \to \infty,$$

the "low-rank" approximation L_k is already a very good approximation of L such that further changes in the rank of L_k do not affect the error in the solution itself. This can be interpreted as having computed the Cholesky factor with respect to the numerical *rank of L.*

Premature convergence as considered above has never occurred during our numerical experiments. In order to avoid such an undesired termination, one might monitor $\operatorname{rank}(L_k)$*, do*

another iteration step if at convergence this rank has still changed during the final two steps, and continue until it remains constant. On the other hand, early convergence with a low rank L_k may prove useful in connection with model reduction algorithms as mentioned above.

4 Performance Results

In this section we compare the accuracy and performance of several solvers for discrete Sylvester and Stein equations of the forms (1) and (2). (As equations of the form (6) are solved using the generalized Lyapunov equation resulting from a Cayley transformation, we do not discuss this method here. Accuracy and performance results for generalized Lyapunov equations can be found in [5, 7].) All the experiments were performed on a PC cluster of 25 nodes, connected with a *myrinet* cross-bar switch. Each node consists of an Intel Pentium-II processor at 300MHz and 128MBytes of RAM. The algorithms were coded in Fortran-77, using IEEE double-precision arithmetic ($\varepsilon \approx 2.2 \times 10^{-16}$), a tuned BLAS library for Intel Pentium-II processors, and the LAPACK 2.0 [2], BLACS 1.1, PBLAS 2.0α, and ScaLAPACK 1.6 libraries [10]. The experiments regarding numerical accuracy and serial performance were obtained on one node of the cluster, using serial tuned implementations of the solvers.

4.1 Numerical Accuracy

We first analyze the reliability of our Stein solvers by means of several numerical examples. Specifically, we compare the following Stein solvers:

- `SB03PD`. The Bartels-Stewart method for the Stein equation as implemented in the *Subroutine Library in Control Theory* – SLICOT[1] [6]. The method is numerically backward stable and hence gives a lower bound for the accuracy that any numerically reliable method should obtain.
- `DGEDLSM`. The squared Smith iteration for the Stein equation as given in (8).
- `DGEDLSG`. The sign function method applied to the Lyapunov equation resulting from the Cayley-transformation of the Stein equation.

Example 4.1 *In this example $\hat{A}$ has the following structure:*

$$A = U^T \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} U, \quad A_{11} = \begin{bmatrix} 1-\alpha & \alpha & & \\ & 1-\alpha & \alpha & \\ & & 1-\alpha & \alpha \\ \alpha/2 & & & 1-\alpha \end{bmatrix}, \quad A_{22} = -A_{11}^T,$$

where A_{12} is a 4×4 random matrix (not necessarily symmetric), U is a random orthogonal matrix, and $C = I_8$. As α approaches 0, the eigenvalues of A get closer to the unit circle. We evaluate the accuracy of the solvers using the normalized residual $\|A\tilde{X}A^T - \tilde{X} + C\|_1/\|C\|_1$, with $\tilde{X}$ the computed solution. Table 1 reports that `DGEDLSM` *and* `DGEDLSG` *obtain less accurate results than* `SB03PD`*. However, by applying iterative refinement, the results obtained by this method (denoted as* `DGEDLSM-ref.`*) constantly outperform those of* `SB03PD`*. As $\alpha \to 0$, even*

[1] Available from `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/SLICOT`.

with iterative refinement there is no hope for algorithm DGEDLSG *due to the increasing ill-conditioning of the Cayley transformed matrix* $c(A)$ *that has to be inverted during the Newton iteration for the sign function. Note that iterative refinement does not improve the solution obtained by* SB03PD *as the computed solution already has a small relative residual.*

α	SB03PD	DGEDLSM	DGEDLSM-ref.	DGEDLSG	DGEDLSG-ref.
10^{-1}	1.403×10^{-13}	8.504×10^{-14}	2.828×10^{-14}	1.427×10^{-12}	2.220×10^{-14}
10^{-2}	1.236×10^{-12}	2.874×10^{-11}	3.695×10^{-13}	2.759×10^{-10}	3.020×10^{-13}
10^{-3}	2.127×10^{-11}	5.441×10^{-9}	2.956×10^{-12}	1.9747×10^{-8}	3.160×10^{-10}
10^{-4}	3.160×10^{-10}	1.091×10^{-6}	6.321×10^{-11}	4.608×10^{-6}	5.730×10^{-11}
10^{-5}	2.103×10^{-9}	4.781×10^{-5}	2.692×10^{-10}	1.755×10^{-4}	6.352×10^{-9}
10^{-6}	2.782×10^{-8}	5.205×10^{-3}	5.180×10^{-9}	4.813×10^{-2}	1.625×10^{-4}

Table 1: Normalized residuals for Example 4.1.

Example 4.2 *We generated Stein equations of order* n *from 100 to 1000. A stable matrix* A *was generated by dividing a matrix with random entries uniformly distributed in* $[0, 1]$ *by the 1-norm of that matrix. The solution matrix was then generated to be symmetric and positive semidefinite as* $X = G^T G$*, with a random uniform matrix* G*. Finally,* C *was set to* $C := X - AXA^T$*.*

The relative errors, $\|X - \tilde{X}\|_1 / \|X\|_1$*, were similar for solvers* SB03PD *and* DGEDLSM*. For the largest problem sizes, the solutions obtained by* DGEDLSM *used to be two digits more accurate than those obtained by* SB03PD*. No iterative refinement was required in any of these random experiments.*

Similar accuracy results were obtained for the discrete Sylvester equation solvers, and the Stein equation solver for the Cholesky factor obtained from (29) as denoted by DGEDLSC.

4.2 Serial performance

In this subsection we investigate the performance of the serial Stein and discrete Sylvester equation solvers.

The matrices in the following experiments were generated as described in Example 4.2 (in case of the discrete Sylvester equation, B was generated as A). Although the execution time of the iterative solvers depends on the number of iterations necessary for convergence, we always perform a fixed amount of 10 iterations. Our experiments showed than in practice, 8–10 iterations are enough for convergence.

The left-hand plot in Figure 1 reports the execution time of the solvers for Stein equations of size n varying from 100 to 1000. Here we consider the execution time of the serial routine SB03PD as the unit time and we report how much "faster" are the iterative solvers DGEDLSM and DGEDLSG. Routine DGEDLSM consistently requires only 60–65% of the execution time of SB03PD. Routine DGEDLSG requires (except for the smaller problem sizes) around 90–95% of the execution time of SB03PD.

In the right-hand plot of Figure 1 we fix $n = 500$ and report the execution time of the solvers for the Cholesky factor of semidefinite Stein equations with m varying from 1 to 500. We compare the SLICOT routine SB03PD and the specific routine to obtain the Cholesky

factor, SB03OD (Hammarling's method), with the iterative solver DGEDLSC. DGEDLSC is more efficient than SB03PD and SB03OD as long as m <100.

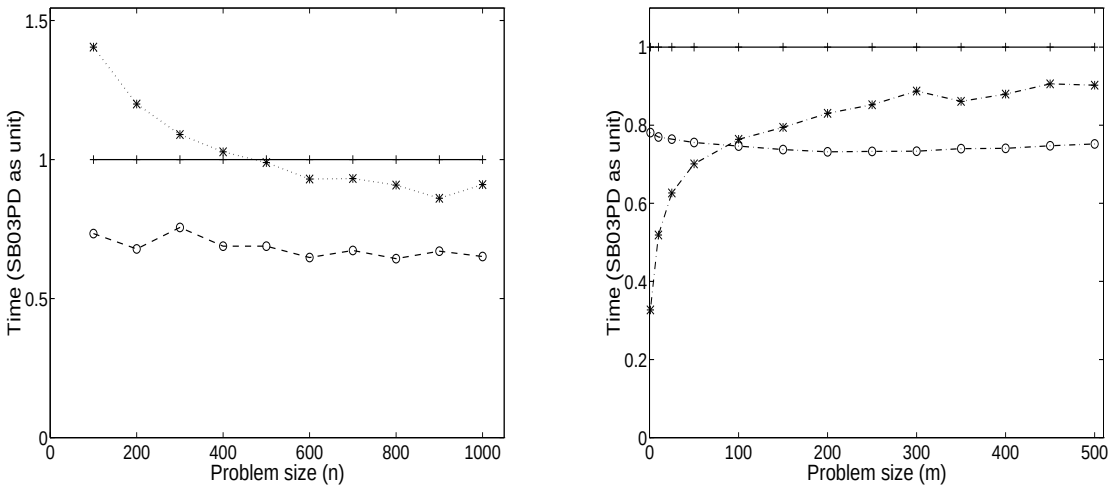

Figure 1: Execution times of the serial Stein equation solvers. Legend: "——+——" = SB03PD, "– – ∘ – –" = DGEDLSM, "· · · ∗ · · ·" = DGEDLSG, "– · – ∘ – · –" = SB03OD, and "– · – ∗ – · –" = DGEDLSC.

In our next experiment, we analyze the performance of the discrete Sylvester solver. As currently there is no appropriate solver in SLICOT, we estimate its cost by scaling the execution time of the Stein solver SB03PD proportionally by the theoretical cost of the Hessenberg-Schur method for the discrete Sylvester equation [19]. We compare the results with those of the following two iterative solvers:

- DGEDSSM. The squared Smith iteration for the discrete Sylvester equation.
- DGEDSSG. The sign function method applied to the Sylvester equation resulting from the Cayley-transformation of the discrete Sylvester equation.

Figure 2 reports the execution time of the solvers for discrete Sylvester equations of size n varying from 100 to 1000, and $m = n$ (left) or $m = n/2$ (right).

When $m = n$, we obtain results similar to those obtained for the Stein equation, with the execution time of iterative solvers DGEDSSM and DGEDSSG below the estimated time for the Hessenberg-Schur method. For $m = n/2$ the picture is quite different; in this case routine DGEDSSG clearly requires a larger execution time than the Hessenberg-Schur method and DGEDSSM.

4.3 Parallel performance

Our next experiment is designed to analyze the scalability and performance of the parallel Stein equation solvers PDGEDLSM, PDGEDLSG, and PDGEDLSC, and the discrete Sylvester equation solvers PDGEDSSM and PDGEDSSG. (Following the naming convention in ScaLAPACK we use the prefix "P-" for the parallel versions of the routines.) No parallel implementation of the Bartels-Stewart algorithm is included as that requires a parallel kernel for solving

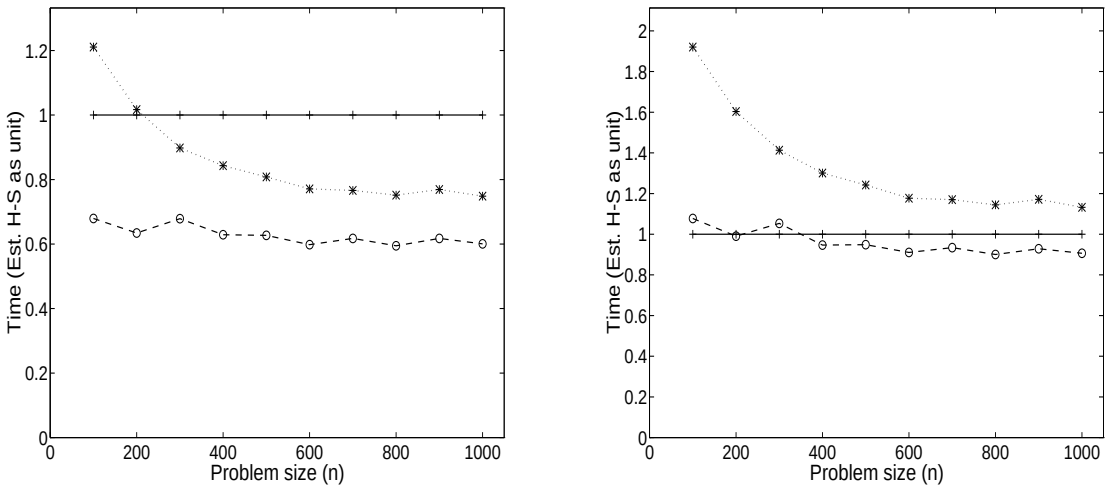

Figure 2: Execution times of the serial discrete Sylvester equation solvers; $m = n$ (left) and $m = n/2$ (right). Legend: "—+—" = Estimated cost for Hessenberg-Schur (H-S) method, "$- - \circ - -$" = `DGEDSSM`, and "$\cdots * \cdots$" = `DGEDSSG`.

Stein/discrete Sylvester equations with the coefficient matrices reduced to real Schur form which is not available in the current version of ScaLAPACK (version 1.6).

Figure 3 reports the MFlops rate per node (millions of floating-point arithmetic operation per second) of the serial and the parallel implementations. We evaluate the parallel algorithms on p=4, 9, 16, and 25 nodes and we set n so that $n/\sqrt{p}$ is constant and equal to 1000. Our parallel algorithms were evaluated using several distribution block sizes (32 was the optimal in our experiments) and both square and rectangular logical topologies. The results in the figure shows a high scalability of the parallel routines as the performance remains almost constant as p is increased.

5 Concluding Remarks

We have described iterative algorithms for solving discrete Sylvester and Stein equations on parallel distributed memory architectures. Our experiments for Stein equations with stable random matrices show similar numerical accuracies for the iterative solvers and the numerically stable Bartels-Stewart method. For coefficient matrices with spectra well-separated from the unit circle the Smith iteration is faster by a factor of about 1.5 than the Bartels-Stewart method.

The Smith iteration basically consists of products of matrices. The experimental results on a PC cluster show the scalability and efficiency of this algorithm, both for discrete Sylvester and Stein equations. The Newton iteration for the matrix sign function only requires scalable parallel kernels. The performance of this algorithm is only slightly worse than that of the Smith iteration.

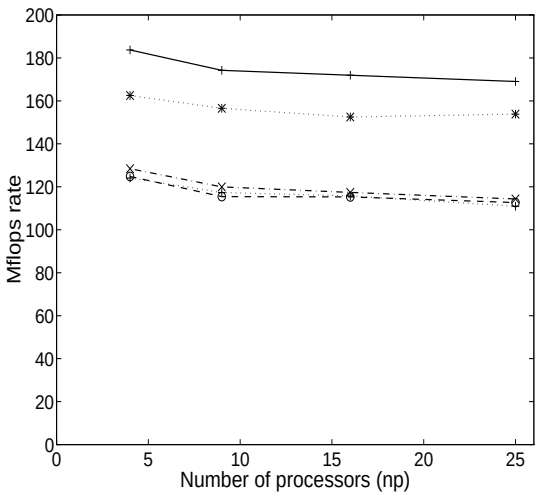

Figure 3: Performance of the parallel Stein and discrete Sylvester equation solvers with $n/\sqrt{p} = 1000$. Legend: "—+—" = `PDGEDSSM`, "– – ∘ – –" = `PDGEDSSG`, "· · · ∗ · · ·" = `PDGEDLSM`, "– · –x – ·–" = `PDGEDLSG`, and "· · · + · · ·" = `PDGEDLSC`.

References

[1] F.A. Aliev and V.B. Larin. *Optimization of Linear Control Systems: Analytical Methods and Computational Algorithms*, volume 8 of *Stability and Control: Theory, Methods and Applications*. Gordon and Breach, 1998.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, second edition, 1995.

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

[4] A. N. Beavers and E. D. Denman. A new solution method for the Lyapunov matrix equations. *SIAM J. Appl. Math.*, 29:416–421, 1975.

[5] P. Benner, J.M. Claver, and E.S. Quintana-Ortí. Parallel distributed solvers for large stable generalized Lyapunov equations. *Parallel Processing Letters*, 9(1):147–158, 1999.

[6] 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, editor, *Applied and Computational Control, Signals, and Circuits*, volume 1, chapter 10, pages 499–539. Birkhäuser, Boston, MA, 1999.

[7] P. Benner and E.S. Quintana-Ortí. Solving stable generalized Lyapunov equations with the matrix sign function. *Numer. Algorithms*, 20(1):75–100, 1999.

[8] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Solving linear matrix equations via rational iterative schemes. In preparation.

[9] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Balanced truncation model reduction of large-scale dense systems on parallel computers. Berichte aus der Technomathematik, Report 99–07, FB3 – Mathematik und Informatik, Universität Bremen, 28334 Bremen (Germany), September 1999. Available from `http://www.math.uni-bremen.de/zetem/berichte.html`.

[10] 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.C. Whaley. *ScaLAPACK Users' Guide*. SIAM, Philadelphia, PA, 1997.

[11] D. Boley and R. Maier. A parallel QR algorithm for the unsymmetric eigenvalue problem. Technical Report TR-88-12, University of Minnesota at Minneapolis, Department of Computer Science, Minneapolis, MN, 1988.

[12] K. Braman, R. Byers, and R. Mathias. The multi-shift QR-algorithm: Aggressive deflation, maintaining well focused shifts, and level 3 performance. Preprint 99-05-01, Department of Mathematics, University of Kansas, Lawrence, KS 66045-2142, May 1999. Available from `http://www.math.ukans.edu/~reports/1999.html`.

[13] D. Calvetti and L. Reichel. Application of ADI iterative methods to the restoration of noisy images. *SIAM J. Matrix Anal. Appl.*, 17:165–186, 1996.

[14] E.J. Davison and F.T. Man. The numerical solution of $A'Q + QA = -C$. *IEEE Trans. Automat. Control*, AC-13:448–449, 1968.

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

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

[17] J.D. Gardiner, M.R. Wette, A.J. Laub, J.J. Amato, and C.B. 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.

[18] G.A. Geist, R.C. Ward, G.J. Davis, and R.E. Funderlic. Finding eigenvalues and eigenvectors of unsymmetric matrices using a hypercube multiprocessor. In G. Fox, editor, *Proc. 3rd Conference on Hypercube Concurrent Computers and Appl.*, pages 1577–1582, 1988.

[19] 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.

[20] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, second edition, 1989.

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

[22] S.J. Hammarling. Numerical solution of the discrete-time, convergent, non-negative definite Lyapunov equation. *Sys. Control Lett.*, 17:137–139, 1991.

[23] G. Henry and R. van de Geijn. Parallelizing the QR algorithm for the unsymmetric algebraic eigenvalue problem: myths and reality. *SIAM J. Sci. Comput.*, 17:870–883, 1997.

[24] G. Henry, D.S. Watkins, and J.J. Dongarra. A parallel implementation of the nonsymmetric QR algorithm for distributed memory architectures. LAPACK Working Note 121, University of Tennessee at Knoxville, 1997.

[25] G.A. Hewer. An iterative technique for the computation of steady state gains for the discrete optimal regulator. *IEEE Trans. Automat. Control*, AC-16:382–384, 1971.

[26] N.J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM Publications, Philadelphia, PA, 1996.

[27] W.D. Hoskins, D.S. Meek, and D.J. Walton. The numerical solution of $A'Q+QA=-C$. *IEEE Trans. Automat. Control*, AC-22:882–883, 1977.

[28] C. Kenney and A.J. Laub. The matrix sign function. *IEEE Trans. Automat. Control*, 40(8):1330–1348, 1995.

[29] V. Kučera. *Analysis and Design of Discrete Linear Control Systems.* Academia, Prague, Czech Republic, 1991.

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

[31] V.B. Larin and F.A. Aliev. Construction of square root factor for solution of the Lyapunov matrix equation. *Sys. Control Lett.*, 20:109–112, 1993.

[32] V. Mehrmann. *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution.* Number 163 in Lecture Notes in Control and Information Sciences. Springer-Verlag, Heidelberg, July 1991.

[33] B.N. Parlett and C. Reinsch. Balancing a matrix for calculation of eigenvalues and eigenvectors. *Numer. Math.*, 13:296–304, 1969.

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

[35] J.D. Roberts. Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *Internat. J. Control*, 32:677–687, 1980. (Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971).

[36] G. Schelfhout. *Model Reduction for Control Design.* PhD thesis, Dept. Electrical Engineering, KU Leuven, 3001 Leuven–Heverlee, Belgium, 1996.

[37] V. Sima. *Algorithms for Linear-Quadratic Optimization*, volume 200 of *Pure and Applied Mathematics.* Marcel Dekker, Inc., New York, NY, 1996.

[38] R.A. Smith. Matrix equation $XA+BX=C$. *SIAM J. Appl. Math.*, 16(1):198–201, 1968.

[39] G.W. Stewart. A parallel implementation of the QR algorithm. *Parallel Computing*, 5:187–196, 1987.

[40] A. Varga. A note on Hammarling's algorithm for the discrete Lyapunov equation. *Sys. Control Lett.*, 15(3):273–275, 1990.

Berichte aus der Technomathematik ISSN 1435-7968

`http://www.math.uni-bremen.de/zetem/berichte.html`

— Vertrieb durch den Autor —

Reports **Stand: 9. November 1999**

98–01. Peter Benner, Heike Faßbender:
An Implicitly Restarted Symplectic Lanczos Method for the Symplectic Eigenvalue Problem, Juli 1998.

98–02. Heike Faßbender:
Sliding Window Schemes for Discrete Least-Squares Approximation by Trigonometric Polynomials, Juli 1998.

98–03. Peter Benner, Maribel Castillo, Enrique S. Quintana-Ortí:
Parallel Partial Stabilizing Algorithms for Large Linear Control Systems, Juli 1998.

98–04. Peter Benner:
Computational Methods for Linear–Quadratic Optimization, August 1998.

98–05. Peter Benner, Ralph Byers, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí:
Solving Algebraic Riccati Equations on Parallel Computers Using Newton's Method with Exact Line Search, August 1998.

98–06. Lars Grüne, Fabian Wirth:
On the rate of convergence of infinite horizon discounted optimal value functions, November 1998.

98–07. Peter Benner, Volker Mehrmann, Hongguo Xu:
A Note on the Numerical Solution of Complex Hamiltonian and Skew-Hamiltonian Eigenvalue Problems, November 1998.

98–08. Eberhard Bänsch, Burkhard Höhn:
Numerical simulation of a silicon floating zone with a free capillary surface, Dezember 1998.

99–01. Heike Faßbender:
The Parameterized SR Algorithm for Symplectic (Butterfly) Matrices, Februar 1999.

99–02. Heike Faßbender:
Error Analysis of the symplectic Lanczos Method for the symplectic Eigenvalue Problem, März 1999.

99–03. Eberhard Bänsch, Alfred Schmidt:
Simulation of dendritic crystal growth with thermal convection, März 1999.

99–04. Eberhard Bänsch:
Finite element discretization of the Navier-Stokes equations with a free capillary surface, März 1999.

99–05. Peter Benner:
Mathematik in der Berufspraxis, Juli 1999.

99–06. Andrew D.B. Paice, Fabian R. Wirth:
Robustness of nonlinear systems and their domains of attraction, August 1999.

99–07. Peter Benner, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí:
Balanced Truncation Model Reduction of Large-Scale Dense Systems on Parallel Computers, September 1999.

99–08. Ronald Stöver:
Collocation methods for solving linear differential-algebraic boundary value problems, September 1999.

99–09. Huseyin Akcay:
Modelling with Orthonormal Basis Functions, September 1999.

99–10. Heike Faßbender, D. Steven Mackey, Niloufer Mackey:
Hamilton and Jacobi come full circle: Jacobi algorithms for structured Hamiltonian eigenproblems, Oktober 1999.

99–11. Peter Benner, Vincente Hernández, Antonio Pastor:
On the Kleinman Iteration for Nonstabilizable System, Oktober 1999.

99–12. Peter Benner, Heike Faßbender:
A Hybrid Method for the Numerical Solution of Discrete-Time Algebraic Riccati Equations, November 1999.

99–13. Peter Benner, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí:
Numerical Solution of Schur Stable Linear Matrix Equations on Multicomputers, November 1999.