Parallel Processing Letters

PARALLEL DISTRIBUTED SOLVERS FOR LARGE STABLE GENERALIZED LYAPUNOV EQUATIONS[1]

PETER BENNER
Zentrum für Technomathematik, Fachbereich 3/Mathematik und Informatik, Universität Bremen, D–28334-Bremen, Germany. E-mail: benner@math.uni-bremen.de

JOSÉ M. CLAVER and ENRIQUE S. QUINTANA-ORTI
Dpto. de Informática, Universitat Jaume I, Aptdo. 242, 12071-Castellón, Spain. E-mails: claver@inf.uji.es and quintana@inf.uji.es

Received (received date)
Revised (revised date)
Communicated by (Name of Editor)

ABSTRACT

In this paper we study the solution of stable generalized Lyapunov matrix equations with large-scale, dense coefficient matrices. Our iterative algorithms, based on the matrix sign function, only require scalable matrix algebra kernels which are highly efficient on parallel distributed architectures. This approach avoids therefore the difficult parallelization of direct methods based on the QZ algorithm. The experimental analysis reports a remarkable performance of our solvers on an IBM SP2 platform.

Keywords: Generalized Lyapunov matrix equations, mathematical software, matrix sign function, parallel distributed multiprocessors.

1. Introduction

Consider the *generalized Lyapunov equation*

$$A^T X E + E^T X A + Q \;=\; 0, \tag{1}$$

where $A, E, X, Q \in \mathbb{R}^{n \times n}$, $Q = Q^T$, and $X = X^T$ is the unknown matrix. Lyapunov equations are of fundamental importance in many analysis and synthesis algorithms in control theory. They arise naturally in linear control problems driven by linear autonomous first-order ordinary differential equations (ODE). As many methods of nonlinear control use the linear system obtained from a linearization of the nonlinear ODE around a working point, these methods also require the sound and efficient

[1] J.M. Claver and E.S. Quintana-Ortí were supported by CICYT Project TIC96-1062-C03-C03.

solution of equations of the form (1). The generalized equations of type (1) with $E \neq I_n$ ($n \times n$ identity matrix) arise naturally from control systems driven by second-order ODEs, descriptor systems, or partial differential equations (PDE). See, e.g., [9, 22, 24, 26, 29, 33] and the references given therein, to list only a few recent references. In particular, in recent years, model reduction for large-scale control problems has become one of the most important issues in systems and control theory. Most of the algorithms proposed so far need to solve one or more Lyapunov equations (1); see, e.g., [15, 27] and the overview given in [28]. Moreover, compression techniques for large-scale descriptor systems lead to dense, unstructured, and large equations of the form (1) that have to be solved in several fundamental control-theoretic computations, see, e.g., [30, 31].

Hereafter, we assume that E is nonsingular and hence, $A-\lambda E$ is a regular matrix pencil, that is $\det(A - \mu E) \not\equiv 0$ for some complex scalar μ ($\mu \in \mathbb{C}$). Additionally, we assume $\lambda_i + \lambda_j \neq 0$ for all $\lambda_i, \lambda_j \in \sigma(A, E)$, where

$$\sigma(A,E) := \{\lambda \in \mathbb{C} \cup \{\infty\} : \lambda = \alpha/\beta,\ \det(\beta A - \alpha E) = 0,\ with\ \lambda = \infty\ if\ \beta = 0\}$$

denotes the generalized spectrum of $A - \lambda E$. These assumptions guarantee (and are necessary) that (1) has a unique solution [20]. Moreover, they also imply the nonsingularity of A and that all eigenvalues of $A - \lambda E$ are finite.

The matrix pencil $A - \lambda E$ is called *stable* if all its eigenvalues are contained in the open left half plane, denoted by $\sigma(A, E) \subset \mathbb{C}^-$. This property holds for most applications we are interested in, and ensures the feasibility of our solvers based on the matrix sign function. The property will be assumed throughout this paper and the associated Lyapunov equation will be called *stable* Lyapunov equation (the anti-stable case, i.e., $\sigma(A, E) \subset \mathbb{C}^-$, can be treated analogously [4]). Moreover, if Q is positive/negative (semi-)definite, then the solution X of (1) is also positive/negative (semi-)definite [20, 23]. We say then that the equation is a *(semi-)definite* generalized Lyapunov equation.

Numerical solution methods for generalized Lyapunov equations are studied in [12, 23]. The methods investigated there are generalizations of the Bartels-Stewart method [3] and Hammarling's algorithm [14] introduced for standard Lyapunov equations ($E = I_n$). Note that Hammarling's algorithm is only applicable for (semi-)definite Lyapunov equations. The initial stage in all these methods is the application of the QZ algorithm [13] (or the QR algorithm [13] if $E = I_n$) to the matrix pencil $A - \lambda E$. This is followed by a quite less expensive back substitution process. The parallelization of back substitution stage on shared memory multiprocessors is analyzed in [18]. The need for parallel computing in this area can be seen from the fact that already for a system with state-space dimension $n = 1000$, (1) represents a set of linear equations with 505000 unknowns (having already exploited the symmetry of X). Systems of such a dimension driven by ODEs are not uncommon in chemical engineering applications, are standard for second order systems, and represent rather coarse grids when derived from the discretization of a PDE; see, e.g., [11, 21, 26].

Several experimental studies, based on block scattered distributions, report the

difficulties in parallelizing the double implicit shifted QR algorithm on parallel distributed multiprocessors [16]. An attempt to increase the granularity employs the multishift techniques [32, 17]. A different approach relies on a block Hankel distribution [16], which improves the balancing of the computational load. Nevertheless, the parallelism and scalability of these parallel QR algorithms are still far from those of matrix factorizations, triangular linear systems solvers, etc. (see, e.g., [5, 8]).

Although the parallelization of the QR algorithm has been thoroughly studied (see [16, 17] and the references therein), in order to solve (1) we need instead the QZ algorithm. We are not aware of any parallel implementation of this algorithm so far, probably due to its higher complexity. Moreover, since both the QR and the QZ algorithms are composed of the same type of fine-grain computations, similar parallelism and scalability results are to be expected from the QZ algorithm.

In this paper we study a different approach, based on the matrix sign function, for solving stable generalized Lyapunov matrix equations. The computation of the matrix sign function only requires well-known matrix kernels (matrix product and matrix inversion) which are highly efficient on parallel architectures [5]. We have chosen this method as it has proved its efficiency for some basic linear algebra computations involving medium-size matrices (i.e., of order $\mathcal{O}(10^3)$) and has therefore been chosen as one of the basic algorithms in ScaLAPACK [5]. Moreover, it has also shown its efficiency for parallel control-relevant computations; see, e.g., [21, 11].

In Section 2 we review the algorithms suggested in [4] for solving stable generalized Lyapunov equations with the matrix sign function. The algorithms and a brief study of the computational and communication cost are described in Section 3. In Section 4 we analyze the performance of our parallel solvers on an IBM SP2 parallel distributed architecture. Concluding remarks are given in Section 5.

2. Solving Lyapunov Equations with the Matrix Sign Function

In this section we briefly summarize two methods, presented in [4], for solving stable generalized Lyapunov equations by means of the matrix sign function.

The *matrix sign function* of a matrix $Z \in \mathbb{R}^{n\times n}$, with no eigenvalues on the imaginary axis, can be defined via the Jordan decomposition of Z,

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

where the Jordan blocks in $J^- \in \mathbb{R}^{k\times k}$ and $J^+ \in \mathbb{R}^{(n-k)\times(n-k)}$ satisfy $\sigma(J^-, I_k) \subset \mathbb{C}^-$ and $\sigma(J^+, I_{n-k}) \subset \mathbb{C}^+$. Then

$$\operatorname{sign}(Z) \;:=\; S \begin{bmatrix} -I_k & 0 \\ 0 & I_{n-k} \end{bmatrix} S^{-1}. \tag{2}$$

Note that $\operatorname{sign}(Z)$ is unique and independent of the order of the eigenvalues in the Jordan decomposition of Z; see, e.g., [19].

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

$$Z_0 \leftarrow Z, \qquad Z_{k+1} \leftarrow \left(Z_k + Z_k^{-1}\right)/2 \quad \text{for} \quad k = 0, 1, 2, \ldots \tag{3}$$

It is shown in [25] that $\operatorname{sign}(Z) = \lim_{k\to\infty} Z_k$.

Although the convergence of the Newton iteration is globally quadratic, the initial convergence may be slow. Accelerating the Newton iteration is possible, e.g., via *determinantal scaling* [6]

$$Z_k \leftarrow |\det(Z_k)|^{-\frac{1}{n}} Z_k.$$

A generalization of the matrix sign function method to a matrix pencil $Z - \lambda Y$ was given by Gardiner and Laub [10] in case Z and Y are nonsingular. They consider the iteration (with determinantal acceleration)

$$Z_0 \leftarrow Z, \quad Z_{k+1} \leftarrow \frac{1}{2c_k}(Z_k + c_k^2 Y Z_k^{-1} Y) \quad \text{for } k = 0, 1, 2, \ldots, \tag{4}$$

where $c_k = (\frac{|\det(Z_k)|}{|\det(Y)|})^{\frac{1}{n}}$. The sequence $\{Y_k\}_{k=0}^{\infty}$ converges to $Y \operatorname{sign}\left(Y^{-1}Z\right)$ [10].

2.1. The stable case

The sign function method was first introduced by Roberts [25] in order to solve stable Sylvester and Lyapunov matrix equations, with $E = I_n$, and algebraic Riccati equations.

In the generalized case, when $A - \lambda E$ is stable, we can use the generalized Newton iteration (4) applied to the matrix pencil

$$H - \lambda K = \begin{bmatrix} A & 0 \\ Q & -A^T \end{bmatrix} - \lambda \begin{bmatrix} E & 0 \\ 0 & E^T \end{bmatrix}.$$

Denoting the limit of the iteration (4) by H_∞, the solution of the Lyapunov equation (1) is given by the solution of the overdetermined and consistent set of linear equations (for details see [10, 4])

$$(H_\infty + K) \begin{bmatrix} I_n \\ -XE \end{bmatrix} = 0. \tag{5}$$

Exploiting the structure of the matrix pencil in $H - \lambda K$, the generalized Newton iteration can be simplified to

$$\begin{aligned} A_0 &\leftarrow A, & A_{k+1} &\leftarrow \tfrac{1}{2}\left(A_k + E A_k^{-1} E\right), \\ Q_0 &\leftarrow Q, & Q_{k+1} &\leftarrow \tfrac{1}{2}\left(Q_k + E^T A_k^{-T} Q_k A_k^{-1} E\right), \end{aligned} \qquad k = 0, 1, 2, \ldots \tag{6}$$

such that $X = \frac{1}{2} E^{-T} \left(\lim_{k\to\infty} Q_k\right) E^{-1}$. For $E = I_n$ this has already been observed by Roberts in [25] while the case $E \neq I_n$ is treated in [4].

Iteration (6) saves a considerable amount of workspace and computational cost compared to applying the generalized Newton iteration to $H - \lambda K$.

2.2. The stable and semidefinite case

The stable semidefinite generalized Lyapunov equation can be written as

$$A^T X E + E^T X A \pm C^T C = 0, \tag{7}$$

where $C \in \mathbb{R}^{r \times n}$. In this case, the solution matrix X can also be written in factored form, $X = \pm Y^T Y$, as X is semidefinite.

In many applications, the Cholesky factor Y of X is required rather than the solution X itself, see, e.g., [14, 15, 28]. A generalization of Hammarling's algorithm as proposed in [23] computes this factor without forming the product $C^T C$ and the solution X explicitly. The advantage of working with Y instead of X is that the condition number of X can be up to the square of that of Y. Hence, using Y, subsequent computations are usually performed with higher accuracy, in particular if X is ill-conditioned.

The method presented in the previous subsection can be modified in order to compute the Cholesky factor of X directly. Consider the iteration for Q in (6). Suppose $Q = C^T C$, this iteration can be re-written as

$$C_0 \leftarrow C, \quad C_{k+1}^T C_{k+1} \leftarrow \frac{1}{2} \begin{bmatrix} C_k \\ C_k A_k^{-1} E \end{bmatrix}^T \begin{bmatrix} C_k \\ C_k A_k^{-1} E \end{bmatrix} \quad \text{for } k = 0, 1, 2, \ldots. \tag{8}$$

Thus, in the resulting algorithm the current iterate C_k is augmented at each iteration by the product $C_k A_k^{-1} E$.

The implementation of Hammarling's algorithm in [23] requires a work array of dimension at least $n \times n$ for C if it is supposed to be overwritten by Y. This suggests to use (8) only as long as $2^k r < n/2$ which is also the bound for which the original iteration (6) becomes cheaper than (8) [4]. This bound is given by

$$k \; > \; \left\lfloor \log_2 \frac{n}{r} \right\rfloor := k_{switch}, \tag{9}$$

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

If k has reached the bound given above, we propose to form the augmented matrix $C_{k+1} = [\, C_k^T, \, (C_k A_k^{-1} E)^T \,]^T \in \mathbb{R}^{2s_k \times n}$, where $C_k \in \mathbb{R}^{s_k \times n}$ and $s_0 = r$. Then from its QR factorization,

$$\tilde{C}_{k+1} \; := \; U_{k+1} \tilde{R}_{k+1} \; = \; U_{k+1} \begin{bmatrix} R_{k+1} \\ 0 \end{bmatrix} \begin{matrix} \} r_{k+1} \\ \} 2s_k - r_{k+1} \end{matrix} ,$$

where $r_{k+1} := \operatorname{rank}\left(\tilde{C}_{k+1}\right)$, it follows that $C_{k+1}^T C_{k+1} = \frac{1}{2} R_{k+1}^T R_{k+1}$. Hence we can set $C_{k+1} := R_{k+1}/\sqrt{2}$ and $s_{k+1} := r_{k+1}$. Note that in order to obtain the Cholesky factor of X, a QR factorization of C_{k+1} has to be computed at convergence even if k does not reach the bound in (9). In order to determine the rank of $\tilde{C}_{k+1}$ correctly, it may be more reasonable to employ a QR factorization with column pivoting [13] or even a rank-revealing QR (RRQR) factorization [7]. This is also described in detail in [4].

We can employ the same stopping criterion in both cases. The convergence $\lim_{k \to \infty} A_k = -E$ suggests the stopping criterion

$$\|A_k + E\|_1 \; \leq \; tol \cdot \|E\|_1 \tag{10}$$

for a user-defined tolerance *tol*. An appropriate procedure in practice [4] is to use $tol = 10 \cdot n \cdot \sqrt{\varepsilon}$ (ε is the machine precision), and perform two additional iteration

steps after the stopping criterion is satisfied. Due to the quadratic convergence of the Newton iteration, this is usually enough to reach the attainable accuracy.

3. Serial and Parallel Algorithms

We first describe the serial generalized Lyapunov solvers based on the matrix sign function. Algorithm 1 is obtained by the iterative scheme in (6) employing determinantal scaling.

Algorithm 1 *[SIGE]*

Input: $A, E, Q \in \mathbb{R}^{n\times n}$ *with* $Q = Q^T$, $\sigma(A,E) \subset \mathbb{C}^-$.

Output: *Solution* $X \in \mathbb{R}^{n\times n}$ *of (1).*

1. *Compute* $\gamma_E = |det(E)|^{\frac{1}{n}}$ *by LU factorization of* E.
   $V = A, \quad X = Q.$
2. *WHILE* $\|V+E\|_1 > tol \cdot \|E\|_1$
   2.1. $A = LUP$ *by LU factorization with partial pivoting.*
   2.2. $\gamma_A = |\det(A)|^{\frac{1}{n}} = \prod_{j=1}^{n} |u_{jj}|^{\frac{1}{n}}$, $\gamma = \gamma_A/\gamma_E$.
   2.3. $W = P^T U^{-1} L^{-1} E$ *by forward and backward substitution.*
   2.4. $A = \frac{1}{2}\left(\frac{1}{\gamma}V + \gamma EW\right)$.
   2.5. $X = \frac{1}{2}\left(\frac{1}{\gamma}X + \gamma W^T X W\right)$.
   2.6. $V = A$.
   END WHILE
3. $E = LUP$ *by LU factorization with partial pivoting.*
   $X = \frac{1}{2} L^{-T} U^{-T} P X P^T U^{-1} L^{-1}$ *by forward and backward substitution.*

END

Algorithm 2 is obtained by the iterative scheme in (8).

Algorithm 2 *[SIGS]*

Input: $A, E \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{r\times n}$, $\sigma(A,E) \subset \mathbb{C}^-$.

Output: *Cholesky factor* Y *of the solution* $X \in \mathbb{R}^{n\times n}$ *of (7).*

1. *Compute* $\gamma_E = |det(E)|^{\frac{1}{n}}$ *by LU factorization of* E.
   $V = A, \quad Y = C, \quad k = 0.$
2. *WHILE* $\|V+E\|_1 > tol \cdot \|E\|_1$
   2.1. $A = LUP$ *by LU factorization with partial pivoting.*
   2.2. $\gamma_A = |\det(A)|^{\frac{1}{n}} = \prod_{j=1}^{n} |u_{jj}|^{\frac{1}{n}}$, $\gamma = \gamma_A/\gamma_E$.
   2.3. $W = P^T U^{-1} L^{-1} E$ *by forward and backward substitution.*
   2.4. $A = \frac{1}{2}\left(\frac{1}{\gamma}V + \gamma EW\right)$.
   2.5. $V = A, \quad \gamma = \sqrt{\gamma}, \quad k = k+1.$
   2.6. *IF* $k \le \lfloor \log_2 \frac{n}{r} \rfloor$ *THEN* $Y = \frac{1}{\sqrt{2}} \begin{bmatrix} \frac{1}{\gamma}Y \\ \gamma YW \end{bmatrix}$.
   ELSE $\frac{1}{\sqrt{2}} \begin{bmatrix} \frac{1}{\gamma}Y \\ \gamma YW \end{bmatrix} = U \begin{bmatrix} Y \\ 0 \end{bmatrix}$ *by QR factorization.*
   END IF
   END WHILE
3. $E = LUP$ *by LU factorization with partial pivoting.*

$Y = \frac{1}{\sqrt{2}} Y P^T U^{-1} L^{-1}$ *by forward and backward substitution.*

END

The most expensive computations involved in Algorithms SIGE and SIGS in terms of flops (floating-point arithmetic operations) are LU factorizations, triangular linear systems, and matrix products. Algorithm SIGS also involves a (rank-revealing) QR factorization. The QR factorization with column pivoting (QRP) [13] can be employed in practice as an RRQR factorization. Table 1 compares the computational costs of these algorithms in flops.

	SIGE	SIGS
Step 1	$\frac{2}{3}n^3$	$\frac{2}{3}n^3$
Step 2	$\frac{23}{3}n^3$	$\frac{14}{3}n^3 + 2^k rn^2 \ (k \leq k_{switch})$ $\frac{24}{3}n^3 \ (k > k_{switch})$
Step 3	$\frac{11}{3}n^3$	$\frac{2}{3}n^3 + 2rn^2$

Table 1. Flop counts per step for SIGE and SIGS; (k is the number of iterations).

Solving the generalized Lyapunov solvers with the Bartels-Stewart method [3] requires about $74n^3$ flops; in case $r \leq n$, Hammarling's algorithm [14] requires about $70n^3 + 2rn^2 + 2r^2n - r^3$ flops. Roughly speaking, ten iterations (Step 2) of (6) are about as expensive as solving (1) by the generalized Bartels-Stewart method [12, 23]. It can be observed that convergence of (6) or (8) often requires 7–10 iterations. All these methods require approximately the same amount of work space.

The parallelization of matrix algebra kernels on parallel distributed architectures has been actively analyzed in recent years. Matrix (LU) factorizations, triangular linear system solvers, and matrix products are highly parallel and scalable [5, 8]. The parallel performance of the QRP is similar to the BLAS-2 LU factorization.

Our parallel algorithms are implemented by means of the parallel matrix algebra building blocks in ScaLAPACK [5]. In this parallel library, the matrices are cyclically distributed by blocks among a $p_r \times p_c$ mesh of processors; for scalability purposes, we only employ square topologies ($p_r = p_c$). Our theoretical analysis of the parallel algorithms follows the performance model for the spectral division problem presented in [2]. The following problem and machine parameters are used:

- n: The dimension of the problem.
- $p = \sqrt{p} \times \sqrt{p}$: The dimension of the mesh of processors.
- τ: Cost of a flop.
- α: Cost of transfering a void message between two processors.
- β: Cost of transfering a floating-point number between two processors.

Approximate computation and communication costs for the building blocks involved in algorithms SIGE and SIGS are given in Table 2. Lower order expressions and load imbalance communication and computation costs due to the block distribution block size are neglected.

A theoretical performance model, based on the costs in Table 1, can be constructed for our parallel solvers. For example, the total cost of one iteration of

Block	Computation cost $\times \frac{n^3}{p}\tau$	Communication cost	
		Latency $\times\alpha$	Bandwith^{-1} $\times \frac{n^2}{\sqrt{p}}\beta$
LU factorization	$\frac{2}{3}$	$(6 + \log p)n$	$3 + \frac{1}{4}\log p$
Triang. solver	1	n	$1 + \frac{3}{4}\log p$
Matrix product	2	$(1 + \frac{1}{2}\log p)\sqrt{p}$	$1 + \frac{1}{2}\log p$
QRP factorization	$\frac{4}{3}$	$3n \log p$	$\frac{3}{4}\log p$

Table 1. Computation and communication cost per node for the building blocks.

solver SIGE is approximately

$$\frac{23}{3}\,\frac{n^3}{p}\,\tau \;+\; (8 + \log p)\, n\, \sqrt{p}\,\alpha \;+\; (8 + \tfrac{13}{4}\log p)\,\tfrac{n^2}{\sqrt{p}}\,\beta.$$

The communication-computation ratio

$$\frac{26}{3}\,\frac{n^3}{p}\,\tau \times \frac{1}{(8 + \log p)\, n\, \sqrt{p}\,\alpha \;+\; (8 + \frac{13}{4}\log p)\,\frac{n^2}{\sqrt{p}}\,\beta}$$

shows the efficiency of the parallel algorithm as the problem size is increased.

4. Experimental Results

In this section we compare the performance of the generalized Lyapunov matrix solvers. In our examples, the computed solution is obtained with the accuracy that could be expected from the conditioning of the problem as implied by the coefficient matrices A, E, Q and the solution X from (1). Roughly speaking, this condition number is proportional to the 2-norm condition number of $W := (E^T \otimes A^T) + (A^T \otimes E^T)$; for details and numerical examples demonstrating the numerical reliability of the proposed algorithms see [4].

All experiments were performed using Fortran 77 and IEEE double-precision arithmetic ($\varepsilon \approx 2.2 \times 10^{-16}$) on an IBM SP2 platform with 80 SP2 RS6000 nodes at 120 MHz, and 256 MBytes RAM per processor. In our tests, each node obtained around 200 Mflops for the matrix product (routine `DGEMM`). Internally, the nodes are connected by a TB3 high performance switch, with latency $\alpha = 31 \times 10^{-6}$ sec. and bandwith $\beta^{-1} \approx 90$ MBytes/sec. We use the native BLAS, and LAPACK, BLACS, and ScaLAPACK libraries [1, 5] to ensure the portability of the algorithms.

We generate random matrices $A = V_n \mathrm{diag}\,(\alpha_1, \ldots, \alpha_n)\, W_n$, $E = V_n W_n$, where the scalars α_i, $1 \le i \le n$, are uniformly distributed in $[-10, 0)$, W_n is an $n \times n$ lower triangular matrix with all unit entries, and V_n is an $n \times n$ matrix with unit entries on and below the anti-diagonal and all other entries equal to zero. Then, C is generated as a random $r \times n$ matrix and $Q = C^T C$. Notice that the convergence criteria used in our algorithms does not involve the right-hand side matrices C or Q. The execution time per iteration of the Lyapunov solvers based on the matrix sign function does not depend on the characteristics/structure of the matrix. Hence, this simple example allow us to analyze the degree of parallelism of our approach.

In our first experiment we compare the execution time of direct methods (the Bartels-Stewart method, BT-ST, and Hammarling's algorithm, HAMM) with the it-

erative Lyapunov solvers based on the matrix sign function. The execution time of the matrix sign function solvers depends on the number of iterations required to converge. In this experiment we perform 10 iterations of the matrix sign function schemes, so that the theoretical cost of the direct methods and the iterative methods is similar. Figure 1 shows that, in practice, serial solvers based on matrix sign function perform much better than expected due to the highly efficient implementation of their computational kernels. Our experiments reported a better performance of the serial solvers based on the matrix sign function even when 20 iterations were required.

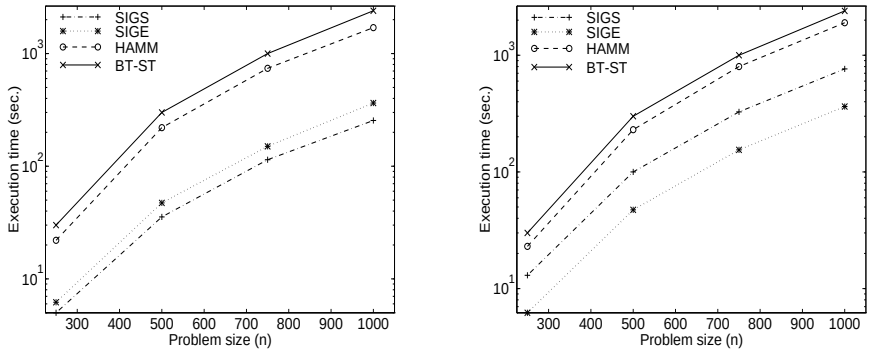


Fig. 2. Execution time of the serial generalized Lyapunov solvers with $r = 1$ (left) and $r = n$ (right) on 1 processor of the IBM SP2.

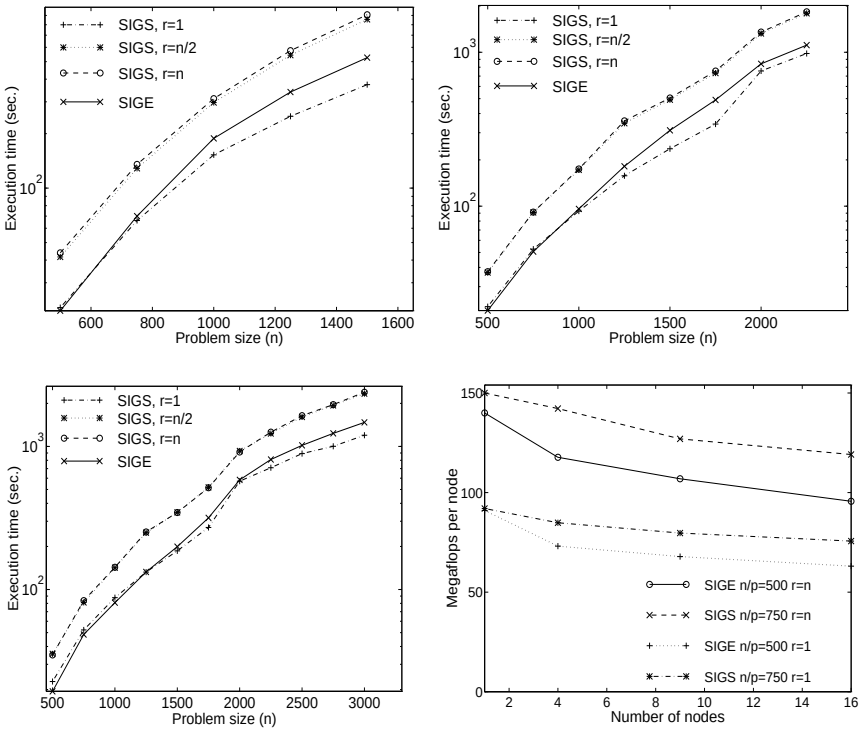
In our next experiment we evaluate the performance of the parallel solvers SIGE and SIGS. Unfortunately, we are not aware of any parallel implementation of the QZ algorithm and therefore a comparison with direct methods can not be given. In Figure 2 we report the execution time of the parallel solvers on 2×2, 3×3, and 4×4 processors, with $r = 1$, $n/2$, and n. In the semidefinite case, when the rank of Q is low ($r \ll n$), solving the Lyapunov equation for the Cholesky factor is more efficient than solving the equation for the explicit solution X. Otherwise, the high overhead of SIGS may only be justified by a significant gain in numerical accuracy.

Table 3 shows the speed-up of the parallel algorithms computed as the ratio between the parallel execution time on $\sqrt{p} \times \sqrt{p}$ processors and the serial execution time. The table reports acceptable speed-ups for 4 and 9 processors; the speed-up is however quite low for 16 processors due to the small size of the problems evaluated.

n	SIGE			SIGS $(r = 1)$			SIGS $(r = n)$		
	$p = 4$	$p = 9$	$p = 16$	$p = 4$	$p = 9$	$p = 16$	$p = 4$	$p = 9$	$p = 16$
500	3.3	3.3	3.6	2.5	2.9	3.1	2.1	2.1	2.0
750	3.5	4.9	5.1	3.1	4.5	5.0	2.5	3.2	3.2
1000	3.3	6.5	7.7	3.2	5.7	6.9	2.6	4.3	4.5

Table 1. Speed-up for SIGE and SIGS on $\sqrt{p} \times \sqrt{p}$ processors of the IBM SP2.

As we can not compute the serial execution time, and therefore the speed-up, for larger problems, we analyze instead the scalability of the parallel algorithms.



g. 3. Execution time on 2×2 (top left) 3×3 (top right), and 4×4 (bottom left) processors of the IBM SP2 and mflop ratio per node (bottom right).

We fix the memory requirements per node of the solvers to $n/p = 500$ and 750. We then obtain the megaflop ratio per node dividing the theoretical cost (in flops) of the algorithm by the execution time; the ratio considers both the computational and communication costs of the algorithms. The bottom right plot in Figure 2 reports the scalability of the algorithms. The performance is slightly degraded as the number of processors gets larger, and the results agree with those of the basic building blocks (LU factorization, triangular linear systems, etc.)

5. Concluding Remarks

We have studied the parallelism of two numerical methods for solving stable generalized Lyapunov matrix equations. Our new solvers, based on the matrix sign function, are currently the only feasible approach for solving these equations when the coefficient matrices are large and dense. Moreover, these algorithms only require scalable matrix algebra kernels which are highly efficient on parallel distributed architectures. We use standard libraries enhances the portability of the algorithms.

The experimental results on an IBM SP2 platform show the advantage of the serial algorithms over the standard approaches based on the QZ algorithms (the Bartels-Stewart method and Hammarling's algorithm) and the performance and

scalability of our new parallel solvers.

Acknowledgements

We thank the Mathematics and Computer Science Division at Argonne National Laboratory for the use of the IBM SP2.

References

1. E. Anderson et al. *LAPACK Users' Guide*, SIAM, Phil., PA, 2nd ed., 1994.
2. Z. Bai et al. *The spectral decomposition of nonsymmetric matrices on distributed memory parallel computers*, SIAM J. Sci. Comp., 18 (1997), pp. 1446–1461.
3. R. Bartels and G. Stewart, *Solution of the matrix equation $AX + XB = C$: Algorithm 432*, Comm. ACM, 15 (1972), pp. 820–826.
4. P. Benner and E. 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, Phil., PA, 1997.
6. R. Byers, *Solving the algebraic Riccati equation with the matrix sign function*, Lin. Alg. Appl., 85 (1987), pp. 267–279.
7. T. Chan, *Rank-revealing QR factorizations*, Lin. Alg. Appl., 88/89 (1987), pp. 67–82.
8. J. Dongarra, A. Sameh, and D. Sorensen, *Implementation of some concurrent algorithms for matrix factorization*, Parallel Comp., 3 (1986), pp. 25–34.
9. Z. Gajic and M. Qureshi, *Lyapunov Matrix Equation in System Stability and Control*, Academic Press, Math. in Sci. and Eng. Series, San Diego, CA, 1995.
10. J. Gardiner and A. Laub, *A generalization of the matrix-sign-function solution for algebraic Riccati equations*, Int. J. Control, 44 (1986), pp. 823–832.
11. ——, *Solving the algebraic Riccati equation on a hypercube multiprocessor*, in Hypercube Concurrent Comp. and Appl., Vol. II, G. Fox, ed., ACM Press, New York, NY, 1988, pp. 1562–1568.
12. J. Gardiner, A. Laub, J. Amato, and C. Moler, *Solution of the Sylvester matrix equation $AXB + CXD = E$*, ACM Trans. Math. Software, 18 (1992), pp. 223–231.
13. G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 2nd ed., 1989.
14. S. J. Hammarling, *Numerical solution of the stable, non-negative definite Lyapunov equation*, IMA J. Numer. Anal., 2 (1982), pp. 303–323.
15. U. Helmke and J. Moore, *Optimization and Dynamical Systems*, Springer-Verlag, London, 1994.
16. G. Henry and R. van de Geijn, *Parallelizing the QR algorithm for the unsymmetric algebraic eigenvalue problem: myths and reality*, SIAM J. Sci. Comp., 17 (1996), pp. 870–883.

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

18. A. S. Hodel and K. Poolla, *Parallel solution of large Lyapunov equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1189–1203.

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

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

21. A. Laub and J. Gardiner, *Hypercube implementation of some parallel algorithms in control*, in Advanced Computing Concepts and Tech. in Control Eng., M. Denham and A. Laub, eds., Springer-Verlag, Berlin, 1988, pp. 361–390.

22. 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, 1991.

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

24. P. Petkov, N. Christov, and M. Konstantinov, *Computational Methods for Linear Control Systems*, Prentice-Hall, Hertfordshire, UK, 1991.

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

26. 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.

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

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

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

30. A. Varga, *On stabilization methods of descriptor systems*, Sys. Control Lett., 24 (1995), pp. 133–138.

31. A. Varga and T. Katayama, *Computation of J–inner–outer factorizations of rational matrices*, Int. J. Robust and Nonlinear Cont., 7 (1997).

32. D. Watkins and L. Elsner, *Chasing algorithms for the eigenvalue problem*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 374–384.

33. K. Zhou, J. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice-Hall, Upper Saddle River, NJ, 1996.