Zentrum für Technomathematik
Fachbereich 3 – Mathematik und Informatik

Parallel Partial Stabilizing Algorithmus for Large Linear Control Systems

Peter Benner Maribel Castillo
Enrique S. Quintana-Ortí

Report 98–03

Berichte aus der Technomathematik

Report 98–03 Juli 1998

Parallel Partial Stabilizing Algorithms for Large Linear Control Systems

Peter Benner* Maribel Castillo† Enrique S. Quintana-Ortí‡

July 20, 1998

Abstract

In this paper we present parallel algorithms for stabilizing large linear control systems on multicomputers. Our algorithms first separate the stable part of the linear control system and then compute a stabilizing feedback for the unstable part. Both stages are solved by means of the matrix sign function which presents a high degree of parallelism and scalability.

The experimental results on an IBM SP2 platform show the performance of our approach.

Keywords: Linear systems, stabilization, Lyapunov equations, matrix sign function, mathematical software.

1 Introduction

Consider a continuous time-invariant linear control system

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ is the state matrix, and $B \in \mathbb{R}^{n\times m}$ is the input matrix. In case the spectrum (or set of eigenvalues) of the state matrix is in the open left half plane (denoted as $\Lambda(A) \subset \mathbb{C}^-$), system (1) is said to be stable. The stabilization problem consists in finding a feedback matrix $F \in \mathbb{R}^{m\times n}$ such that the input $u(t) = Fx(t)$, $t \geq 0$, yields a stable closed-loop system

$$\dot{x}(t) = (A + BF)x(t), \quad x(0) = x_0. \tag{2}$$

This problem has a solution if the matrix pair (A, B) is stabilizable, i.e., for all λ in the closed right half plane, $\text{rank}\,([A - \lambda I_n, B]) = n$ (hereafter, I_n denotes the identity matrix of order n). The stabilization problem arises in methods for the solution of parabolic partial differential equations [23], in control problems such as, e.g., the computation of an initial approximate solution in Newton's method for the algebraic Riccati equation, simple synthesis methods to design controllers, etc. [14, 15, 28].

There are two different approaches for solving the stabilization problem, these are, pole assignment methods or linear quadratic optimization algorithms. Pole assignment methods compute a feedback matrix such that the state matrix of the closed-loop system (2) has a prespecified spectrum. In this sense, the stabilization problem can be considered as a special type of pole assignment problem with a high degree of freedom in the design of the problem (the poles can be chosen anywhere in the open left half plane). This approach however presents several drawbacks. First, the pole assignment problem is probably an intrinsically ill-conditioned problem for systems of order larger than 10 [3, 19, 29, 30]; second, how to choose the poles to improve the conditioning

*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 Jaume I, E–12.071 Castellón, Spain. E-mail: castillo@inf.uji.es

‡Departamento de Informática, Universidad Jaume I, E–12.071 Castellón, Spain. E-mail: quintana@inf.uji.es

of the problem is still an open problem; and third, most of the pole assignment algorithms are based on QR algorithm-like procedures (see [3, 31, 35, 36] and the references therein) which are not well-suited for parallel computation and therefore too expensive for large control systems.

In the linear quadratic optimization approach an input $u(t)$, $t \geq 0$, is computed that minimizes

$$\mathcal{J}(u) = \min_{u(t)} \int_0^\infty (x(t)^T Q x(t) + u(t)^T R u(t)) dt$$

subject to (1); here, $Q = C^T C$ and $R = R^T$ can be any positive semidefinite and positive definite matrices, respectively. Under certain conditions (see, e.g., [27, 28]), it can be proved that the linear quadratic optimization problem has a unique solution of the form

$$u(t) = Fx(t) = -R^{-1} B^T X x(t), \tag{3}$$

where $X \in \mathbb{R}^{n \times n}$ is the unique symmetric positive semidefinite solution of the algebraic Riccati equation (ARE)

$$A^T X + XA - XBR^{-1}B^T X + Q = 0. \tag{4}$$

Moreover, the closed-loop system defined by this feedback matrix is stable. Here, the freedom in choosing Q and R can be used to obtain a small norm feedback matrix F and to guarantee that small perturbations will not make the closed-loop system unstable. Typically, $Q = 0_n$ (the null matrix of order n) is chosen and the ARE (4) then becomes an inverse Lyapunov equation

$$AY + YA^T - \tilde{Q} = 0, \tag{5}$$

where $\tilde{Q} = BR^{-1}B^T$ and $Y = X^{-1}$ (if X is singular then $Y = X^+$, the pseudo-inverse of X).

A recent study in [19] shows that in case $\Lambda(A) \subset \mathbb{C}^+$, B is a square nonsingular matrix, and $R = (B^T B)^{1/2}$ is chosen, $Q = 0_n$ is optimal in the sense that it minimizes the norm of F. This is however a too restrictive case and in practice we just set $Q = 0_n$, $R = I_n$, and obtain the feedback matrix from equations (3) and (5) [2, 26, 34]. This method is simple and does not introduce additional rounding errors.

The Bartels-Stewart method is one of the most well-known and efficient algorithms for solving Lyapunov equations of moderate dimension [6]. In this method, the coefficient matrix A is reduced in a first stage to a condensed form by means of the QR algorithm; then, in a second stage, the solution is obtained from the reduced equation by a backsubstitution procedure. The QR algorithm is known to present a lack of scalability [20, 21]; moreover, the degree of parallelism in this algorithm is not as high as that of the usual matrix algebra kernels (matrix factorizations, matrix products, etc.).

For large order Lyapunov equations with a stable (or antistable) coefficient matrix, Lyapunov solvers based on the matrix sign function are more appropriate [7] due to their high degree of parallelism. While in general the coefficient matrix of the Lyapunov equation in (5) is not stable, we can use a spectral division technique, also based on the matrix sign function, to reduce our stabilization problem to a subproblem which satisfies such a condition (see Section 3). This two-stage approach does not increase significantly the computational cost.

The paper is structured as follows. In Section 2 we define the matrix sign function and present three iterative schemes for its computation. In Section 3 we review how to use the matrix sign function to divide the spectrum of a matrix. In Section 4 we describe Lyapunov equation solvers based on the matrix sign function. In Sections 5 and 6 we combine these algorithms to provide two-stage stabilization methods and study the parallelization of the matrix sign function–based iterations from the theoretical and experimental points of view. Finally, in Section 7 we summarize our concluding remarks.

2 The Matrix Sign Function

There exist several definitions of the matrix sign function (see, e.g., [25] for an overview). For instance, let

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

be the Jordan decomposition of $Z \in \mathbb{R}^{n\times n}$ [17], where the Jordan blocks in $J^- \in \mathbb{C}^{k\times k}$ and $J^+ \in \mathbb{C}^{(n-k)\times(n-k)}$ contain, respectively, the eigenvalues of Z in the open left and right complex planes. In case Z has no pure imaginary eigenvalues, the *matrix sign function* is given by

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

Note that $\operatorname{sign}(Z)$ is unique and independent of the order of the eigenvalues in the Jordan decomposition of Z [27].

Choosing a starting point at Z and applying Newton's root-finding iteration to $Z^2 = I_n$, we obtain the Newton matrix sign function iteration

$$Z_0 \leftarrow Z, \quad Z_{j+1} \leftarrow \frac{1}{2}(Z_j + Z_j^{-1}), \quad j = 0, 1, 2, \ldots, \tag{7}$$

which converges to $\operatorname{sign}(Z) = \lim_{j\to\infty} Z_j$ [33]. The iterative scheme (7) requires $2n^3$ flops (floating-point arithmetic operations) per iteration.

The convergence of the Newton matrix sign function iteration is globally quadratic. Acceleration of the initial convergence is possible, e.g., via *determinantal scaling* [11],

$$Z_j \leftarrow c_j Z_j, \quad c_j = |\det(Z_j)|^{-\frac{1}{n}},$$

where $\det(Z_j)$ denotes the determinant of Z_j. The determinant of Z_j is obtained, with an $\mathcal{O}(n)$ flop cost, as a by-product when its inverse is computed. Other acceleration schemes are compared in [4].

There exist also other iterative schemes for the matrix sign function [24]; among these, *Halley's iteration*,

$$Z_0 \leftarrow Z, \qquad Z_{j+1} \leftarrow Z_j(3I_n + Z_j^2)(I_n + 3Z_j^2)^{-1}, \qquad j = 0, 1, 2, \ldots, \tag{8}$$

and the *Newton-Schulz iteration*,

$$Z_0 \leftarrow Z, \qquad Z_{j+1} \leftarrow \frac{1}{2} Z_j(3I_n - Z_j^2), \qquad j = 0, 1, 2, \ldots, \tag{9}$$

are specially appropriate for parallel distributed architectures as their computational cost is mainly due to matrix products. The higher computational cost of the Halley matrix sign function iteration (about $20n^3/3$ flops per iteration) might be balanced by its cubic convergence, but this is not the case in practice. The cost of the Newton-Schulz matrix sign iteration is approximately $4n^3$ flops per iteration. Moreover, this iteration is only guaranteed to converge if $\|Z_0^2 - I_n\| < 1$, for some suitable norm. Therefore, the Newton-Schulz matrix sign function iteration has to be combined with some initial iteration to obtain a globally convergent "hybrid" algorithm.

Iterations (7), (8), and (9) can be stopped when

$$\|Z_{j+1} - Z_j\| \le \gamma := c \cdot \sqrt{\varepsilon} \cdot \|Z_j\|,$$

for $\|\cdot\|$ a suitable norm, c a small order constant, and ε the machine precision. Once the stopping criterion is satisfied, two more iterations are carried out. Due to the quadratic (or cubic) convergence of the iterations we therefore ensure the maximum attainable accuracy.

3 Spectral Division with the Matrix Sign Function

The matrix sign function has proved useful in many spectral division problems, as $(I_n - \mathrm{sign}(Z))/2$ defines the skew projector onto the stable Z–invariant subspace parallel to the unstable subspace. In other words, if Z has k stable eigenvalues, $n-k$ unstable eigenvalues, and no eigenvalues on the imaginary axis, and

$$U^T\left((\mathrm{sign}(Z) - I_n)/2\right)\Pi = \begin{bmatrix} S_{11} & S_{12} \\ 0 & 0 \end{bmatrix}$$

is a rank-revealing QR decomposition [13], then U divides the spectrum of Z as follows

$$U^T Z U = \begin{bmatrix} Z_{11} & Z_{12} \\ 0 & Z_{22} \end{bmatrix}, \tag{10}$$

with $\Lambda(Z_{11}) \subset \mathbb{C}^-$, of order k, and $\Lambda(Z_{22}) \subset \mathbb{C}^+$, of order $n-k$. Here, the first k columns of U, associated with the stable part of the spectrum of Z, form an orthonormal basis for the stable invariant subspace of this matrix.

Recent results show that the matrix sign function is in practice an accurate approach for the computation of invariant subspaces [12, 7].

Consider we have to stabilize a system defined by the matrix pair (A, B), and assume U divides the spectrum of A as in (10). Applying this transformation to the matrix B, we obtain

$$U^T B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix},$$

where $B_1 \in \mathbb{R}^{k\times n}$ and $B_2 \in \mathbb{R}^{(n-k)\times n}$; the stabilization problem is then reduced to finding a feedback matrix $F_2 \in \mathbb{R}^{m\times n-k}$ that stabilizes the matrix pair (A_{22}, B_2). Note that matrix A_{22} is antistable ($\Lambda(A_{22}) \subset \mathbb{C}^+$); the complete feedback matrix is obtained as $F = (0_{m\times k}, F_2)U^T$.

4 Solving Lyapunov Equations with the Matrix Sign Function

In [33] Roberts introduced the use of the matrix sign function for solving stable (or antistable) Lyapunov equations. The solution of

$$A^T X + XA - Q = 0,$$

is computed by applying the Newton iteration (7) to the associated Hamiltonian matrix

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

corresponding to (5). The solution matrix X can then be determined from the stable H–invariant subspace given by the range of the projector $(I_{2n} - \mathrm{sign}(H))/2$. Roberts also shows in [33] that, when applied to H, the Newton matrix sign function iteration (7) can be simplified to

$$\begin{array}{lll} A_0 \leftarrow A, & A_{j+1} \leftarrow \frac{1}{2}\left(A_j + A_j^{-1}\right), & \\ Q_0 \leftarrow Q, & Q_{j+1} \leftarrow \frac{1}{2}\left(Q_j + A_j^{-T} Q_j A_j^{-1}\right), & j = 0, 1, 2, \ldots \end{array} \tag{11}$$

and that $X = \frac{1}{2}\lim_{j\to\infty} Q_j$. The sequences for A_j and Q_j require about $6n^3$ flops per iteration so that 5–6 iterations are as expensive as the Bartels–Stewart method [6].

The structure of H can also be exploited to obtain efficient variants for the Halley matrix sign function iteration (8),

$$\begin{array}{rcl} A_0 \leftarrow A, & A_{j+1} \leftarrow & A_j(3I_n + A_j^2)(I_n + 3A_j^2)^{-1}, \\ Q_0 \leftarrow Q, & Q_{j+1} \leftarrow & \left((A_j - 3A_{j+1})^T(A_j^T Q_j - Q_j A_j)\right. \\ & & \left. +Q_j(3I_n + A_j^2)\right)(I_n + 3A_j^2)^{-1}, \end{array} \quad j = 0, 1, 2, \ldots \tag{12}$$

and Newton-Schulz matrix sign function iteration (9),

$$\begin{array}{rcl} A_0 \leftarrow A, & A_{j+1} \leftarrow & \frac{1}{2}A_j(3I_n - A_j^2), \\ Q_0 \leftarrow Q, & Q_{j+1} \leftarrow & \frac{1}{2}\left(Q_j(3I_n - A_j^2)\right. \\ & & \left. -A_j^T(A_j^T Q_j - Q_j A_j)\right). \end{array} \quad j = 0, 1, 2, \ldots. \tag{13}$$

See [8] for details. The approximate computational costs per iteration for (12) and (13) are about $\frac{50}{3}n^3$ and $12n^3$ flops, respectively.

A stopping criterion for these iterations can be designed based on $\lim_{j\to\infty} A_j = I_n$ [7], e.g., we can employ

$$||A_{j+1} - I_n|| \leq c \cdot \sqrt{\varepsilon} \cdot ||A_j||,$$

and perform two additional iterations once it is satisfied.

5 Implementation Issues and Parallel Algorithms

The proposed stabilizing method can be algorithmically described as follows (see also [19, 22]).

Input: $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ such that (A, B) is stabilizable.

Output: $F \in \mathbb{R}^{m\times n}$ such that $A + BF$ is stable.

1. Compute $\operatorname{sign}(A)$.
2. Compute a rank-revealing QR decomposition
$$U^T\left((\operatorname{sign}(A) - I_n)/2\right)\Pi = \begin{bmatrix} S_{11} & S_{12} \\ 0 & 0 \end{bmatrix}.$$
3. Let $U = (U_1, U_2)$ be an analogous partition of U. Compute $A_{22} = U_2^T A U_2$ and $B_2 = U_2^T B$.
4. Solve the Lyapunov equation
$$A_{22}Y + YA_{22}^T - B_2 B_2^T = 0$$
5. Compute $F_2 = -B_2 Y^{-1}$; then, $F = (0, F_2)U^T$.

We may use iterative schemes (7), (8), or (9) to compute $\operatorname{sign}(A)$ in Stage 1, and the special variants in (11), (12), or (13) to solve the Lyapunov equation in Stage 4. In Stage 2, we can use a QR factorization with column pivoting to obtain an approximate rank-revealing decomposition [17]. Theoretically this orthogonal factorization may fail to reveal the rank, though in practice this is a reliable numerical tool [9, 32].

In case A has eigenvalues on the imaginary axis we can nevertheless apply the algorithm to $(A + \alpha I_n, B)$ for a small $\alpha > 0$. Thus, we divide the spectrum of A along the line $-\alpha i$ and stabilize those eigenvalues with real part larger than $-\alpha$. Choosing α carefully so that $A + \alpha I_n$ has no eigenvalues close to the imaginary axis we can avoid numerical difficulties associated with eigenvalues close to or on the imaginary axis. The same technique can also be used to obtain a certain degree of stability, i.e., $\Lambda(A + BF) \subset \{z \in \mathbb{C}; \operatorname{Re}(z) < -\alpha\}$ (here, $\operatorname{Re}(z)$ stands for the real part of z).

In control problems usually the state matrix only has a few unstable eigenvalues. The subsystem to stabilize in Stages 3, 4, and 5 is small and the cost of these stages is therefore negligible when compared to the cost of Stages 1 and 2. In our theoretical study we will neglect the cost for these last three stages and assume that the computation of sign(A) requires "iter" iterations.

Our algorithms are implemented using ScaLAPACK (scalable linear algebra package) and PBBLAS (parallel block basic linear algebra suprograms) [10]. These are public-domain parallel libraries for MIMD computers which can be run on any machine that supports either PVM [16] or MPI [18]. ScaLAPACK provides scalable parallel distributed kernels for many of the matrix algebra kernels available in LAPACK [1]. This library employs BLAS and LAPACK for serial computations and the BLACS (basic linear algebra communication subprograms) for communication.

The implementation of ScaLAPACK assumes a block-cyclic distribution scheme [10], which is mapped to a logical $p_r \times p_c$ grid of processes. Each process owns a collection of (`MB` $\times$ `NB`) blocks, which are locally and contiguously stored in a two-dimensional array in "column-major" order.

For scalability purposes, we only employ in our algorithms square topologies ($p_r = p_c$) with each process mapped onto a different processor. The following problem and machine parameters are used:

- n: Size of the problem.
- $p = \sqrt{p} \times \sqrt{p}$: Dimension of the square grid of processors.
- τ: Time of a double-precision floating-point arithmetic operation.
- α: Time required to communicate a zero-length message between two processors.
- β: Time required to communicate a double-precision floating-point number between two processors.

Following the performance model in [5], we present in Table 1 approximate computation and communication costs for the building blocks from ScaLAPACK [10] employed in our parallel stabilizing algorithms. We have neglected the lower order expressions in the table.

		Computation	Communication cost	
Block	operation	cost $\times \frac{n^3}{p}\,\tau$	Latency $\times\,\alpha$	Bandwidth^{-1} $\times \frac{n^2}{\sqrt{p}}\,\beta$
`PxGETRF`:	LU factorization	$\frac{2}{3}$	$(6+\ \log p)n$	$(3+\frac{1}{4}\log p)$
`PxTRSM`:	Triangular system solver	1	n	$(1+\frac{3}{4}\log p)$
`PxGETRI`:	Inverse from LU factors	$\frac{4}{3}$	$2n$	$(2+\frac{3}{2}\log p)$
`PxGEMM`:	Matrix product	2	$(1+\frac{1}{2}\log p)\sqrt{p}$	$(1+\frac{1}{2}\log p)$
`PxGEQPF`:	QR fact. with column pivot.	$\frac{4}{3}$	$3n\log p$	$\frac{3}{4}\log p$
`PxORMQR`:	Apply Householder transf.	2	–	$2\log p$

Table 1: Theoretical costs of the parallel building blocks.

In Table 2 we present a theoretical performance model for our parallel stabilizing algorithms.

We have observed moderate differences between the theoretical results obtained from our theoretical model and the experimental results; these differences are mainly due to several simplifications and approximations in the theoretical model; e.g., for simplicity, imbalance cost and lower order terms in computations and communications are neglected in the theoretical model.

Matrix sign function it.	Computation cost $\times \frac{n^3}{p}\,\tau$	Communication cost Latency $\times\,\alpha$	Communication cost Bandwith^{-1} $\times \frac{n^2}{\sqrt{p}}\,\beta$
Newton	$2 \times$ iter $+\ \frac{10}{3}$	$(8+\log p)n \times$ iter $+\ 3n\log p$	$(5+\frac{7}{4}\log p) \times$ iter $+\ \frac{11}{4}\log p$
Halley	$\frac{20}{3} \times$ iter $+\ \frac{10}{3}$	$((2+\log p)\sqrt{p}\ +\ (8+\log p)n)$ $\times$ iter $+\ 3n\log p$	$(7+\frac{11}{4}\log p) \times$ iter $+\ \frac{11}{4}\log p$
Newton-Schulz	$4 \times$ iter $+\ \frac{10}{3}$	$(2+\log p)\sqrt{p} \times$ iter $+\ 3n\log p$	$(2+\log p) \times$ iter $+\ \frac{11}{4}\log p$

Table 2: Theoretical costs of the parallel stabilizing algorithms.

6 Experimental Results

All our numerical experiments were performed using Fortran 77 and IEEE double-precision arithmetic on an IBM SP2 platform ($\varepsilon \approx 2.2 \times 10^{-16}$). We made use of the vendor supplied BLAS (*essl*) and LAPACK libraries. The IBM SP2 architecture consists of 80 SP2 RS6000 nodes at 120 MHz, and 256 MBytes RAM per processor. Internally, the nodes are connected by a TB3 high performance switch. The latency is 31 microseconds and the bandwith is about 90 MBytes/sec.

We have developed three parallel stabilizing algorithms which differ in the matrix sign function iteration employed both for separating the spectrum of A and solving the Lyapunov equation in the second stage. The following names are used for our algorithms:

- `PDGESTNE`. Newton iteration for the matrix sign function.
- `PDGESTNS`. Newton iteration for the matrix sign function followed by Newton-Schulz iteration once its convergence is guaranteed.
- `PDGESTHA`. Halley iteration for the matrix sign function.

In the serial case, we compare the performance of these algorithms with the traditional approach, based on the QR algorithm [17], for dividing the spectrum and solving Lyapunov equations (algorithm `DGESTQR`). Unfortunately a parallel implementation of this algorithm requires several basic matrix kernels which are not available in ScaLAPACK.

In our first experiment we generate a random matrix pair (A, B), with n=m=20 and entries normally distributed as N(0,1). In Figure 1 we report the distribution of the spectrum of A and $A+BF$ with F computed by means of algorithm `PDGESTNE`. The figure shows that, after applying the algorithm, the closed-loop system is stable. Actually, the stable eigenvalues of the original system are not modified while for each unstable eigenvalue, $\lambda = \alpha + \beta i \in \Lambda(A)$, $\alpha > 0$, we now have $\tilde{\lambda} = -\alpha + \beta i \in \Lambda(A+BF)$. No significant differences were found when algorithms `DGESTQR`, `PDGESTHA`, or `PDGESTNS` were employed.

We now analyze the influence of the number of stable eigenvalues, r, of the state matrix A in the execution time of the stabilizing algorithms. For this purpose, we generate random matrix pairs (A, B) with n=m=500, r=50, 150, ..., 450, and entries uniformly distributed in U[-1,1]. We then employ the serial versions of the stabilizing algorithms to compute the feedback matrix F. The execution time of the algorithms is reported in Figure 2. The experiment shows that the execution time is inversely proportional to the number of stable eigenvalues (the size of the Lyapunov equation that has to be solved in the second stage depends on the number of stable/unstable eigenvalues). An important decrease is observed when r varies from 50 to 250, while as r gets close to n, this reduction becomes negligible.

Next, we evaluate the performance of the parallel stabilizing algorithms based on the matrix sign function. Following the usual case in control problems, we generate random matrix pairs

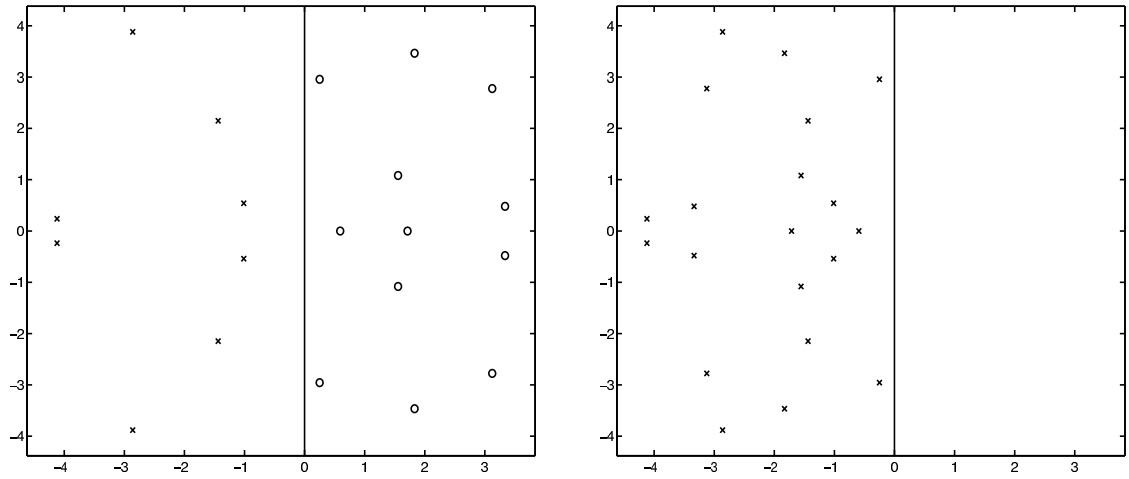

Figure 1: Distribution of $\Lambda(A)$ (left) and $\Lambda(A + BF)$ (right), with A and B random matrices (n=m=20), and F computed by means of algorithm PDGESTNE.

(A, B) with a large number of stable eigenvalues (about 99%). The stable and unstable eigenvalues are uniformly distributed in $[-10, 0)$ and $(0, 10]$, respectively. The cost of the parallel stabilizing algorithms is basically due to the cost of separating the spectrum of the stable and antistable part of the system as stabilizing the antistable part only requires the resolution of a small Lyapunov equation ($\mathcal{O}(n/100)$).

In Figure 3 we analyze the execution time of the parallel stabilizing algorithms on 1, 4, 9, and 16 processors. This figure shows that the best algorithms are those based on the Newton (PDGESTNE) and Newton-Schulz (PDGESTNS) matrix sign function iterations. Moreover, algorithm PDGESTNS presents better results when the number of processors is large.

We also analyze the scalability of the parallel stabilizing algorithms. For this purpose, we fix the dimension of the problem per node to $n/p = 1000$ and compute the Mflop ratio per node (millions of flops per second and node) of the algorithms. Figure 4 reports a high scalability of our parallel algorithms as there is only a slight decrease in the Mflop ratio as the number of processors is increased.

Finally, we define the approximate speed-up

$$\tilde{S}_p(n, p) = \frac{M(n \cdot \sqrt{p}, p)}{M(n, 1)} \cdot p,$$

where $M(n, p)$ is the Mflop ratio per node of an algorithm for solving a problem of size n using p processors. This speed-up can be computed for large n, and avoids perturbing effects of the cache size as the local size of the problem remains constant as p is increased. In case we keep n/p fixed at 1000 we obtain the "speed-ups" in Table 3. The results show that algorithm PDGESTHA presents the highest speed-ups. This is due to the greater number of matrix products in this iterative scheme, which are specially efficient on parallel distributed architectures. However, it should be noted that for the examples presented above the higher Mflop ratio and speed-up of the Halley iteration still does not balance its higher computational cost per iteration.

7 Concluding Remarks

We have presented parallel algorithms for the stabilization of large linear control systems. Our new solvers are based on the matrix sign function and can be employed to stabilize large systems with dense coefficient matrices.

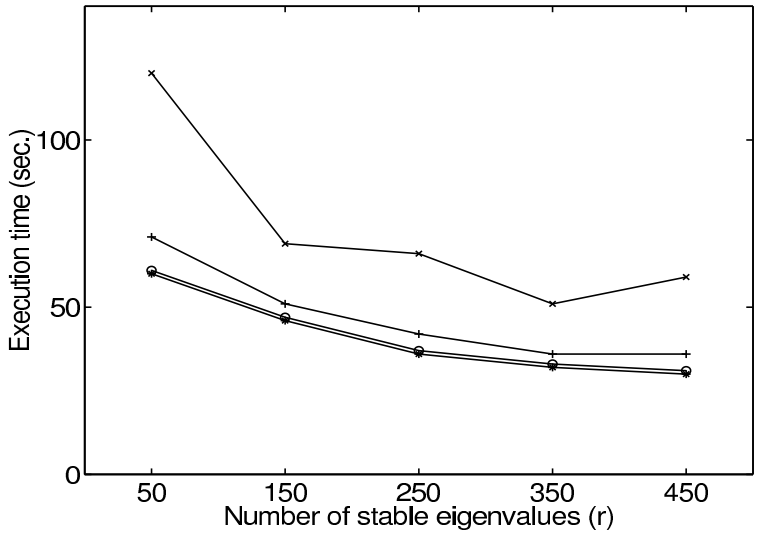

Figure 2: Execution time of the serial stabilizing algorithms, with A and B random matrices (n=m=500), on the IBM SP2. Legend: symbols "–.x.–" for DGESTQR, "– – ∗ – –" for PDGESTNE, "· · · o · · ·" for PDGESTNS, and "– + –" for PDGESTHA.

Algorithm	p=4	p=9	p=16
PDGESTNE	3.00	6.43	10.01
PDGESTNS	3.04	6.23	9.76
PDGESTHA	3.46	7.14	11.71

Table 3: Approximate "speed-ups" ($\tilde{S}_p(n,p)$) of the parallel stabilizing algorithms with n/p=1000.

The matrix sign function approach only requires scalable matrix algebra kernels which are highly efficient on parallel distributed architectures. The experimental results on an IBM SP2 platform show the performance of our new parallel solvers.

Acknowledgments

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

This research was supported by the *Acciones-Integradas Hispano-Alemanas* program. Maribel Castillo and Enrique S. Quintana-Ortí were partially supported by the CICYT project No. TIC96-1062-C03-C03.

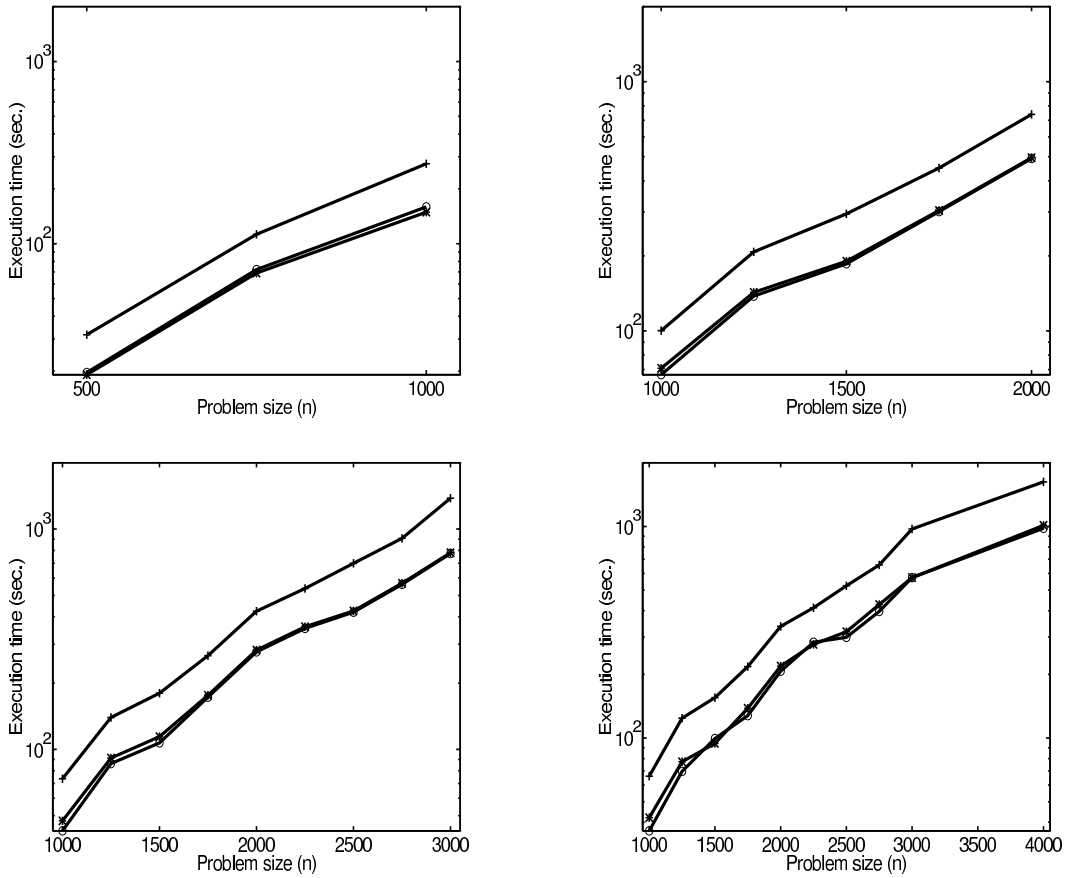

Figure 3: Execution time of the parallel stabilizing algorithms, with A and B random matrices (n=m, r=0.99 n) , on 1 processor (top left), 2×2 processors (top right), 3×3 processors (bottom left), and 4×4 processors (bottom right) of the IBM SP2. Legend: symbols "$--*--$" for PDGESTNE, "$\cdots o \cdots$" for PDGESTNS, and "$-+-$" for PDGESTHA.

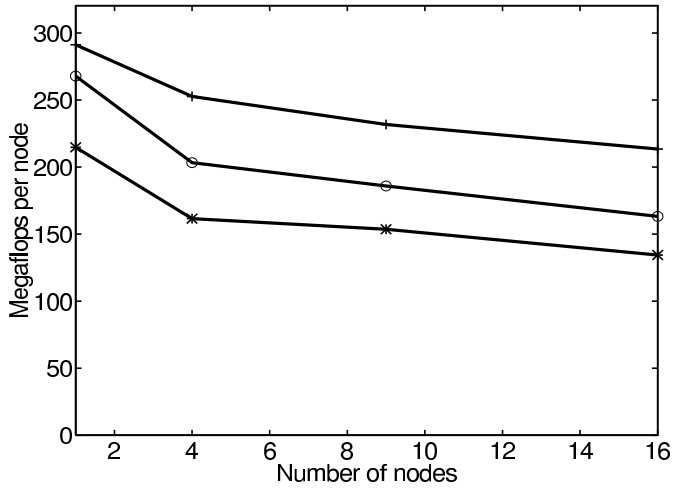

Figure 4: Mflop ratio per node of the parallel stabilizing algorithms on the IBM SP2 with n/p=1000. Legend: symbols "$--*--$" for PDGESTNE, "$\cdots o \cdots$" for PDGESTNS, and "$-+-$" for PDGESTHA.

References

[1] E. Anderson *et al. LAPACK Users' Guide*. SIAM, Philadelphia, PA, second edition, 1994.

[2] E. S. Armstrong. An extension of Bass' algorithm for stabilizing linear continuous constant systems. *IEEE Trans. Automat. Control*, AC–20:153–154, 1975.

[3] M. Arnold. Algorithms and conditioning for eigenvalue assignment. Master's thesis, Northern Illinois University, Dept. of Mathematical Sciences, DeKalb, Il, 1993.

[4] Z. Bai and J. Demmel. Design of a parallel nonsymmetric eigenroutine toolbox, Part I. In R.F. Sincovec et al, editor, *Proceedings of the Sixth SIAM Conference on parallel Processing for Scientific Computing*, 1993.

[5] Z. Bai, J. Demmel, J. Dongarra, A. Petitet, H. Robinson, and K. Stanley. The spectral decomposition of nonsymmetric matrices on distributed memory parallel computers. *SIAM J. Sci. Comput.*, 18:1446–1461, 1997.

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

[7] P. Benner and E.S. Quintana-Ortí. Solving stable generalized Lyapunov equations with the matrix sign function. Technical Report SPC 97_23, Fak. f. Mathematik, TU Chemnitz, 09107 Chemnitz, FRG, 1997.

[8] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Rational iterative schemes for the numerical solution of linear matrix equations. In preparation.

[9] C.H. Bischof and G. Quintana. Computing rank-revealing QR factorizations of dense matrices. *ACM Trans. Math. Software*, 1998. To appear.

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

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

[12] R. Byers, C. He, and V. Mehrmann. The matrix sign function method and the computation of invariant subspaces. *SIAM J. Matrix Anal. Appl.*, 18(3):615–632, 1997.

[13] T. Chan. Rank revealing QR factorizations. *Linear Algebra Appl.*, 88/89:67–82, 1987.

[14] L. Dai. *Singular Control Systems*, volume 118 of *Lecture Notes in Control and Inform. Sci.* Springer, New York, NY, 1989.

[15] V. Dragan and A. Halanay. *Stabilization of Linear Systems*. Birkhäuser, Basel, Switzerland, 1994.

[16] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, B. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine – A User's Guide and Tutorial for Network Parallel Computing*. MIT Press, 1994.

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

[18] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press, 1994.

[19] C. He and V. Mehrmann. Stabilization of large linear systems. In L. Kulhavá, M. Kárný, and K. Warwick, editors, *Preprints of the European IEEE Workshop CMP'94, Prague, September 1994*, pages 91–100, 1994.

[20] G. Henry and R. van de Geijn. Parallelizing the QR algorithm for the unsymmetric algebraic eigenvalue problem: myths and reality. *SIAM J. Sci. Comput.*, to appear.

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

[22] V. Hernández and E.S. Quintana. *Stabilizing large control linear systems on multicomputers*, volume 1215 of *Lecture Notes in Control and Information Sci.* Springer–Verlag, Berlin, 1997.

[23] H. Jarausch. *Zur numerischen Untersuchung von parabolischen Differentialgleichungen mit Hilfe einer adaptiven spektralen Zerlegung.* Habilitationsschrift, RWTH Aachen, Aachen (FRG), 1990.

[24] C. Kenney and A.J. Laub. Rational iterative methods for the matrix sign function. *SIAM J. Matrix Anal. Appl.*, 12:273–291, 1991.

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

[26] D. L. Kleinman. An easy way to stabilize a linear constant system. *IEEE Trans. Automat. Control*, AC-15:692, 1970.

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

[28] 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.

[29] V. Mehrmann and H. Xu. An analysis of the pole placement problem. I. The single-input case. *Electr. Trans. Num. Anal.*, 4:89–105, 1996.

[30] V. Mehrmann and H. Xu. Analysis of the pole placement problem II. The multi-input case. *Electr. Trans. Num. Anal.*, 5:77–97, 1997.

[31] G. Miminis and C.C. Paige. An algorithm for pole assignment of time invariant linear systems. *Internat. J. Control*, 35:341–354, 1982.

[32] G. Quintana, X. Sun, and C.H. Bischof. A Blas-3 version of the QR factorization with column pivoting. *SIAM J. Sci. Comput.*, 1998. To appear.

[33] 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.

[34] V. Sima. An efficient Schur method to solve the stabilization problem. *IEEE Trans. Automat. Control*, AC-26:724–725, 1981.

[35] A. Varga. A Schur method for pole assignment. *IEEE Trans. Automat. Control*, AC-27:517–519, 1981.

[36] A. Varga. A multishift Hessenberg method for pole assignment of single-input systems. *IEEE Trans. Automat. Control*, AC-41(12), 1996.

Berichte aus der Technomathematik ISSN 1435-7968

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

— Vertrieb durch den Autor —

Reports **Stand: 2. September 1998**

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.