Solving Large Sparse Lyapunov Equations on Parallel Computers*

José M. Badía[1], Peter Benner[2], Rafael Mayo[1], and Enrique S. Quintana-Ortí[1]

[1] Depto. de Ingeniería y Ciencia de Computadores, Universidad Jaume I,
12080–Castellón, Spain,
{badia,mayo,quintana}@icc.uji.es,
Tel.: +34-964-728257, Fax: +34-964-728486
[2] Institut für Mathematik, Technische Universität Berlin,
D-10623 Berlin, Germany,
benner@math.tu-berlin.de,
Tel.: +30-314-28035, Fax: +30-314-79706

Abstract. This paper describes the parallelization of the low-rank ADI iteration for the solution of large-scale, sparse Lyapunov equations. The only relevant operations involved in the method are matrix-vector products and the solution of linear systems. Experimental results on a cluster, using the SuperLU library, show the performance of this approach.

Key words: Lyapunov equations, ADI iteration, low-rank approximation, sparse linear systems.

1 Introduction

The *Lyapunov equation*

$$AX + XA^T + BB^T \ = \ 0, \tag{1}$$

where $A, X \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and X is the sought-after solution, plays a fundamental role in control theory; see, e.g., [1]. In this paper, we assume that the spectrum of A lies in the open left half plane. Under this assumption, there exists a unique solution, $X = X^T \geq 0$ [8], which can be factorized as $X = SS^T$; here, S is known as the "Cholesky" factor of the solution. Usually $m \ll n$ and often, S is of low (column) rank [9].

Numerical methods for the solution of small-scale, dense Lyapunov equations are introduced in [2,5]. These methods require the application of the QR algorithm, resulting in a parallelism poorly attractive [6]. The use of the matrix sign function [11] allows the solution of moderate-scale dense Lyapunov equations (n up to a few thousands) on parallel computers [3]. However, none of these algorithms are appropriate for large-scale sparse Lyapunov equations.

In this paper, we follow an approach based on the LR-ADI iteration [10], which provides a low-rank approximation of the Cholesky factor of the solution. Many large-scale applications, e.g. in model reduction and optimal control,

* Supported by the Fundació Caixa-Castelló/Bancaixa PI-1B2001-14.

employ this approximation rather than the solution. The LR-ADI solver only requires matrix operations such as the matrix-vector product and the solution of linear systems. The parallelization of these matrix operations on distributed-memory computers has been largely studied in the literature and several libraries are currently available like, e.g., SuperLU and MUMPS.

The rest of the paper is structured as follows. In Section 2 we briefly review the LR-ADI solver for large sparse Lyapunov equations. Details on the implementation and the parallelization are given in Section 3. Finally, experimental results and concluding remarks follow, respectively, in Sections 4 and 5.

2 LR-ADI Solver for Large Sparse Lyapunov Equations

The *cyclic low-rank alternating direction implicit* (LR-ADI) Lyapunov solver [10] benefits from the usual low-rank property of matrix B to provide low-rank approximations to the Cholesky factor of X. This iterative algorithm also includes a heuristic to determine a set of "shift" parameters to accelerate the convergence.

Specifically, given an "l–cyclic" set of complex shift parameters $\{p_1, p_2, \ldots\}$, $p_k = a_k + b_k i$, $p_k = p_{k+l}$, the LR-ADI iteration can be formulated as follows:

$$\begin{array}{ll} V_0 = (A + p_1 I_n)^{-1} B, & S_0 = \sqrt{-2\ a_1}\ V_0, \\ V_{k+1} = V_k - \delta_k (A + p_{k+1} I_n)^{-1} V_k, & \delta_k = p_{k+1} + \overline{p_k}, \\ S_{k+1} = [S_k\ ,\ \gamma_k V_{k+1}], & \gamma_k = \sqrt{a_{k+1}/a_k}, \end{array} \tag{2}$$

where I_n denotes the identity matrix of order n. On convergence, after $\tilde{k}$ iterations, a low-rank matrix $\tilde{S}$ of order $n \times \tilde{k}m$ is computed such that $\tilde{S}\tilde{S}^T$ approximates X. Matrix $\tilde{S}$ can be employed in many situations where the Cholesky factor S is required. Further details on the LR-ADI iteration are given in [10].

The only matrix operation required in the LR-ADI iteration is the solution of linear systems with a sparse coefficient matrix shifted by a certain (complex) scalar.

The performance of the iteration strongly depends on the selection of the shift parameters; see [10]. A heuristic procedure is proposed in [10] which employs approximations of certain eigenvalues of A. The procedure is based on an Arnoldi iteration with A and A^{-1}, and therefore only requires matrix-vector products and the solution of linear systems, respectively.

3 Implementation and Parallelization

The LR-ADI iteration (2) involves the same coefficient matrix in iterations k and $k+l$. Thus, when more than l iterations are required and sufficient storage space is available, a large computational cost is saved by computing factorizations of $(A+p_i I_n)$, $i = 1, 2, \ldots, l$, in advance and using these factors in the solution of the subsequent linear systems. This approach benefits from the use of direct solvers; besides, as all matrices have the same sparsity structure, the preliminary analysis phase, common in direct sparse solvers, needs to be performed only once.

In practice, the LR-ADI iteration produces a sequence of matrices V_k of decreasing norm [9]. A practical criterion for detecting the convergence of the iteration is therefore to stop when $\|V_k\|$ is small.

Although A is real, the iteration may require complex arithmetic in case any of the shift parameters is complex. For symmetric matrices all the shifts should be chosen to be real, using a Lanczos-based procedure, and the iteration only employs real arithmetic.

The parallelization of the solver requires parallel routines for the computation of the matrix-vector product, the solution of linear systems involving sparse (complex) matrices, and some other minor operations. Here we employ the SuperLU library [4] for the solution of linear systems. In the current version of SuperLU both the coefficient and the right-hand side matrices have to be completely stored in all processes (processors). We take advantage of this replication to easily implement an efficient parallel matrix-vector product. Future extensions of our parallel solvers will benefit from employing parallel kernels from other libraries like MUMPS, MCSPARSE, ScaLAPACK, etc.

4 Numerical Experiments

All the experiments presented in this section were performed on a cluster of Intel Pentium-II processors (300MHz, 128 MBytes of RAM), connected with a *Fast Ethernet* switch, using IEEE double-precision floating-point arithmetic. The BLAS routine `DGEMV` for the matrix-vector product achieved around 46 Mflops (millions of flops/sec.) on this architecture.

We report experimental results for the solution of large-scale Lyapunov equations (1) where A is tridiagonal stable matrix with random entries and B is a random vector ($m = 1$).

Figure 1 reports the execution time of the first LR-ADI iteration (2) using n_p processors. This iteration requires the solution of a complex linear system of order n with a single right-hand side, and several other minor computations (e.g., convergence criterion). A profile of the parallel algorithm showed that, when 4 processors were used on a problem of order n=150,000, the iteration spent roughly 75% in the LU factorization (including negligible times for equilibration, row and column permutations and symbolic factorization; almost half of this time was employed to distribute the matrices). The solution of the linear system from the LU factors used 24% of the time (including one iterative refinement step); the remaining time was spent in minor computations.

The figure also shows the convergence rate, measured as $\|V_k\|_1$, when l different shifts are used in the LR-ADI iteration and n=150,000. Actually, $k_p + k_m$ shifts are computed: $k_p = 2l$ are obtained using the Arnoldi iteration on A and $k_m = 2l$ with the Arnoldi iteration on A^{-1}. A selection procedure determines then the "best" l shifts to use in the iteration.

Solving a Lyapunov equation of order n=150,000 on 4 processors, with $k_p = k_m = 2l = 12$, required about 35 minutes; 39% of this time was spent in the computation of the shifts and the remaining 61% in the LR-ADI iteration.

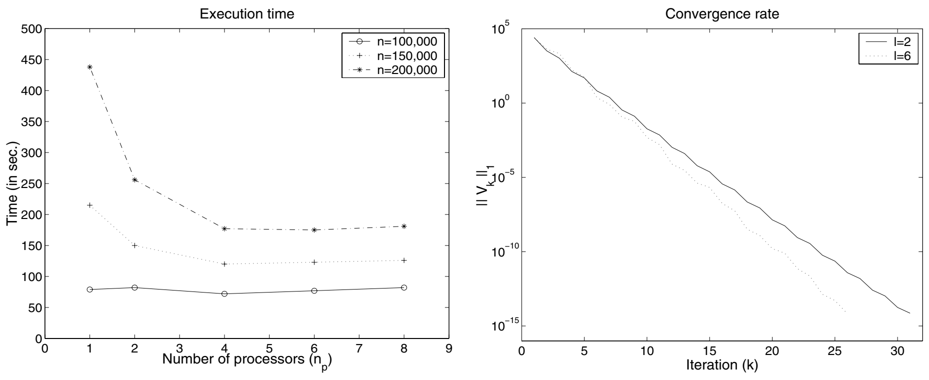

Fig. 1. Execution time of a single LR-ADI iteration (left) and convergence rate (right).

5 Concluding Remarks

We have described the parallelization of a numerical solver, based on the LR-ADI iteration, for large-scale, sparse Lyapunov equations on a cluster. The algorithm benefits from the availability and efficiency of parallel kernels for the matrix-vector product and the solution of sparse linear systems using direct methods.

References

1. B.D.O. Anderson and J.B. Moore. *Optimal Control – Linear Quadratic Methods.* Prentice-Hall, Englewood Cliffs, NJ, 1990.
2. R.H. Bartels and G.W. Stewart. Solution of the matrix equation $AX + XB = C$: Algorithm 432. *Comm. ACM*, 15:820–826, 1972.
3. P. Benner, J.M. Claver, and E.S. Quintana-Ortí. Parallel distributed solvers for large stable generalized Lyapunov equations. *Parallel Proc. Lett.*, 9:147–158, 1999.
4. J.W. Demmel, J.R. Gilbert, and X.S. Li. *SuperLU User's Guide*, 1999. Available from `http://www.nersc.gov/~xiaoye/SuperLU`.
5. S.J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA J. Numer. Anal.*, 2:303–323, 1982.
6. 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.
7. A.S. Hodel, B. Tenison, and K.R. Poolla. Numerical solution of the Lyapunov equation by approximate power iteration. *Linear Algebra Appl.*, 236:205–230, 1996.
8. P. Lancaster and M. Tismenetsky. *The Theory of Matrices.* Academic Press, Orlando, 2nd edition, 1985.
9. T. Penzl. Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. *Sys. Control Lett.*, 40(2):139–144, 2000.
10. T. Penzl. A cyclic low rank Smith method for large sparse Lyapunov equations. *SIAM J. Sci. Comput.*, 21(4):1401–1418, 2000.
11. 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.