SOFTWARE FOR THE GENERALIZED EIGENPROBLEM ON DISTRIBUTED MEMORY ARCHITECTURES *

MARK T. JONES AND PAUL E. PLASSMANN†

Abstract. The generalized eigenproblem is of significant importance in several fields. Generalized eigenproblems can be very large with matrices of order greater than one million for problems arising from three-dimensional finite element models. To solve such problems we are proposing a flexible software system for parallel distributed memory architectures. This software is based on the Lanczos algorithm with a shift-and-invert transformation. In this paper we briefly describe the prototype version of the software, present computational results, and indicate the status of the project.

1. Introduction. The solution of the symmetric generalized eigenvalue problem,

$$Kx = \lambda Mx, \tag{1}$$

where K and M are real, symmetric matrices, and either K or M is positive semi-definite, is of significant practical importance, especially in structural engineering as the vibration problem and the buckling problem [1]. The matrices K and M are either banded or sparse. Usually $m << n$ of the smallest eigenvalues of Equation 1 are sought, where n is the order of the system. The method of Lanczos [9], suitably altered for the generalized eigenvalue problem, has been shown to be useful for the efficient solution of Equation 1 [10].

The Lanczos algorithm for the generalized eigenproblem [10] has been shown to be effective on vector supercomputers and shared-memory parallel computers [5]. An effective software package, LANZ [4], for shared-memory parallel computers has been developed and is available from netlib. However, the LANZ software is not suitable for distributed-memory architectures such as the Intel DELTA, the Thinking Machines CM-5, and the IBM SP-1. The LANZ software uses a shared-memory model of computation that is not suitable for such architectures. Also, the linear system solution methods in LANZ are only appropriate for a small number of tightly-coupled processors (≤ 20) [5] [6].

In this paper we will discuss our plans for scalable software, provide brief prototype results, and give the current status of the software.

2. Distributed-Memory Software. The basic algorithm in the software is the block Lanczos algorithm with a spectral transformation as described in [3]. The reader is referred to [3] for a complete description of the algorithm. The majority of the parallel computation/communication in this algorithm occurs in the solution of a linear system,

$$(K - \sigma M)x = b, \tag{2}$$

and a matrix multiplication,

$$Mx = b. \tag{3}$$

The block variant of the Lanczos algorithm is chosen because it not only allows easy computation of eigenvalues of multiplicity up to the blocksize, but also improves the communication to computation ratio of a parallel implementation. The ratio is improved because the

* This work was supported in part by the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38.

† The address of the first author is: Computer Science Department, University of Tennessee, Knoxville, TN 37996. The address of the second author is: Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439.

block algorithm allows the inner products to take place as matrix-matrix operations and, more importantly, the operations in Equations 2 and 3 to be implemented using multiple right-hand sides. This core Lanczos algorithm is implemented in a reverse-communication style; this allows flexibility in implementing different approaches for Equations 2 and 3.

The class of architectures targeted for the software has from 20 to 1000 RISC processors. Each processor has its own memory and processors communicate via message passing. The communication rate is expected to be slow relative to the computation rate. Little consideration will be given to the topology; for example, the software will not be tailored to a hypercube architecture. The goal is to have a scalable algorithm/implementation, where for a fixed problem size per processor, the computation rate per node is constant and the memory requirements per node are constant.

The eigensolution will take place as part of a larger computation occurring on the parallel computer. In particular, the software is written under the assumption that the matrix has been partitioned and mapped in a "good" fashion onto the processors and that this partitioning and mapping will not be changed by the eigensolver. For example, if the underlying domain from which the matrix was generated was a spatial discretization of a PDE being solved with the finite element method, it is expected that a good partitioning and mapping already exists to allow efficient evaluation and assembly of the stiffness matrix. If such a partitioning has not been done, many algorithms and some software currently exist for this problem (see, for example, [11] [13]).

The most difficult part of the eigensolution to parallelize, as well as the most computationally expensive, is the solution of Equation 2. For this computation at least two options are planned. First, a preliminary interface to the iterative methods in the BlockSolve package [7] has been constructed. This package provides portable, parallel software for the conjugate gradient method preconditioned by incomplete factorization; the parallelism is obtained by employing a parallel graph coloring heuristic [8]. The interface to this package is relatively straightforward, but poor convergence can result when σ in Equation 2 is large relative to the smallest eigenvalue in Equation 1. This difficulty must be addressed.

Also planned is an interface to the parallel direct sparse factorization methods in the CAPSS package [12]. This package provides a parallel implementation of sparse Cholesky factorization. When σ is larger than the smallest eigenvalue in Equation 1, one could, at the risk of a loss of stability, use the LDL^T decomposition. To the best of the authors' knowledge, no general, numerically stable software for distributed memory architectures exists for sparse indefinite factorization. The interface to the CAPSS package is more difficult than to BlockSolve as one must address issues related to the suitability of the existing partitioning/mapping for sparse direct factorization.

3. Some Experimental Results. Using a prototype version of the core, reverse communication software combined with BlockSolve, the following results were obtained on the Intel DELTA for a practical vibration problem arising from a finite element model [2]:

p	n	nnz	Mflops/ Proc.	Total Mflops
64	7.7×10^4	1.6×10^7	4.88	312
128	1.6×10^5	3.4×10^7	5.00	640
256	3.2×10^5	6.8×10^7	4.87	1247
512	6.4×10^5	1.4×10^8	4.97	2545

Note that the problem size per processor is fixed as the number of processors varies. From the results one can see that, as desired, the computation rate per processor remained constant. In addition, note that the memory requirements per processor also remained constant.

4. Current Status. We currently have a portable, stable, but undocumented version of the core, reverse communication block Lanczos software running on distributed memory architectures. Additional testing must be performed on this software prior to release. A preliminary interface to BlockSolve is completed, but the interface to CAPSS remains to be worked out. In short we believe that we have a software plan that, based on prototype results, will result in scalable software that is of use to a large user community.

5. Acknowledgements. The authors thank Albert Danial of Purdue University who participated in the coding of the preliminary version of the parallel block Lanczos software.

REFERENCES

[1] C. P. Blankenship and R. J. Hayduk, *Potential supercomputer needs for structural analysis.* Presentation at the Second International Conference on Supercomputing (Santa Clara, CA), May 3-8 1987.

[2] T. Canfield, M. T. Jones, P. E. Plassmann, and M. Tang, *Thermal effects on the frequency response of piezoelectric crystals*, in New Methods in Transient Analysis, PVP-Vol. 246 and AMD-Vol. 143, New York, 1992, ASME, pp. 103–108.

[3] R. G. Grimes, J. G. Lewis, and H. D. Simon, *The implementation of a block shifted and inverted Lanczos algorithm for eigenvalue problems in structural engineering*, ETA-TR-39, Boeing Computer Services, Seattle, Washington, August 1986.

[4] M. T. Jones and M. L. Patrick, *LANZ: Software for solving the large sparse symmetric generalized eigenproblem*, Preprint MCS-P158-0690, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Il., 1990. Also available as ICASE Interim Report no. 12.

[5] ———, *The Lanczos algorithm for the generalized symmetric eigenproblem on shared-memory architectures*, Applied Numerical Mathematics, 12 (1993), pp. 377–389.

[6] ———, *Factoring symmetric indefinite matrices on high-performance architectures*, SIAM Journal on Matrix Analysis and Applications, 15 (1994), pp. 273–283.

[7] M. T. Jones and P. E. Plassmann, *BlockSolve v1.0: Scalable library software for the parallel solution of sparse linear systems*, ANL Report ANL-92/46, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1992.

[8] ———, *A parallel graph coloring heuristic*, SIAM Journal on Scientific and Statistical Computing, 14 (1993), pp. 654–669.

[9] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of Research of the National Bureau of Standards, 45 (1950), pp. 255–282.

[10] B. Nour-Omid, B. N. Parlett, T. Ericsson, and P. S. Jensen, *How to implement the spectral transformation*, Mathematics of Computation, 48 (1987), pp. 663–673.

[11] A. Pothen, H. D. Simon, and K.-P. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM Journal of Matrix Analysis and Applications, 11 (1990), pp. 430–452.

[12] P. Raghavan, *User's manual: CAPSS: A Cartesian parallel sparse solver.* National Center for Supercomputer Applications, University of Illinois at Urbana-Champaign, November 1993.

[13] R. D. Williams, *Performance of dynamic load balancing algorithms for unstructured mesh calculations*, Concurrency: Practice and Experience, 3 (1991), pp. 457–481.