DOE/ER/25122--T1

Project Summary

E.R. Jessup
University of Colorado
Department of Computer Science
Boulder, CO 80309-0430

This report summarizes the results of our project "Numerical Methods for the Unsymmetric Tridiagonal Eigenvalue Problem". It was funded by both by a DOE grant (#DE-FG02-92ER25122, 6/1/92–5/31/94, $100,000) and by an NSF Research Initiation Award (#CCR-9109785, 7/1/91–6/30/93, $46,564.) The publications resulting from that project during the DOE funding period are listed below. Two other journal papers ([11] and [10]) and two other conference papers ([6] and [3]) were produced during the NSF funding period. Most of the listed conference papers are early or partial versions of the listed journal papers.

Journal Publications

[J0] E.R. Jessup, D. Yang, and S.A. Zenios, *Parallel Decomposition of Structured Matrices Arising in Stochastic Programming Problems*, to appear in SIAM J. Opt.

[J1] S. Crivelli and E.R. Jessup, *Optimal Eigenvalue Computation on Distributed-Memory MIMD Multiprocessors*, to appear in Parallel Computing.

Conference Publications

[C1] S. Crivelli and E.R. Jessup, *A Programming Paradigm for Distributed-Memory Computers*, in *Parallel Processing for Scientific Computing*, ed. R. Sincovec, D. Keyes, M. Leuze, L. Petzold, and D. Reed, SIAM, 1993.

[C2] S. Crivelli and E.R. Jessup, *Optimal Eigenvalue Computation on a Mesh Multiprocessor*, in *Parallel Processing for Scientific Computing*, ed. R. Sincovec, D. Keyes, M. Leuze, L. Petzold, and D. Reed, SIAM, 1993.

[C3] E.R. Jessup, *On a Divide and Conquer Approach to the Nonsymmetric Eigenvalue Problem*, in *the Proceedings of Jornadas Panamericanas de Matematicas Aplicadas y Computationales*, ed. M. Lentini, 1993.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

HH

IIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER

Papers C3 and [10] builds on our paper [11], references [1, 4, 5], and our earlier work [8, 9] by discussing how how a divide and conquer eigensolver might be developed for the nonsymmetric eigenvalue problem. In papers C3 and [10], we show that, although the divide and conquer mechanism can be adapted in a straightforward way to solve the real nonsymmetric eigenproblem, most of the desirable characteristics of the other algorithms are lost.

The speed and accuracy of divide and conquer eigensolvers rely largely on the availability of a fast and globally convergent root-finder and on the prevalence and ease of deflation. However, there appears to be no equivalent root-finder for the nonsymmetric case, and deflation of the nonsymmetric problem may not lead to such substantial savings as in the symmetric case. Furthermore, if it is necessary to compute the left eigenvectors from the right eigenvectors to maintain accuracy, the eigenvector computation may become inefficient, especially in parallel. The greatest danger with the divide and conquer method, however, lies in its potential instability. Even if the original matrix is well-conditioned with respect to the eigenproblem, an ill-conditioned submatrix can be created at any level of updating. Given the obstacles in the way of an efficient and accurate nonsymmetric divide and conquer method, we do not intend to continue work on this approach at this time.

Papers C2, [3], and J1 discuss how to compute one eigenvalue of a symmetric tridiagonal matrix efficiently on hypercube and mesh-connected distributed-memory, MIMD multiprocessors. Understanding how best to compute one eigenvalue in parallel is the first step in understanding how best to compute an arbitrary number of eigenvalues in parallel. In our papers, we study and compare the costs of computing a single eigenvalue by serial bisection [7], parallel multisection [8, 12], and Swarztrauber's parallel bisection method [14] on the target machines. All of these methods derive from the Sturm sequence property of the determinants of the sequence of principal minors of a symmetric tridiagonal matrix [7, 15]. Multisection is implemented in parallel on p processors by simultaneously evaluating the Sturm sequence at p interior points of the search interval, one evaluation per processor. Swarztrauber's method uses the associative property of matrix multiplication to split the evaluation of the determinants among the different processors.

Our approach to this problem was directed by Simon's proof that bisection is not the optimal method for computing an eigenvalue on a single vector processor [13]. However, we show that his analysis does not extend directly to the computation of an eigenvalue on a distributed-memory MIMD multiprocessor. In particular, we show how the optimal number of sections (and processors) to use for bisection or multisection depends on variables such as the matrix size and certain parameters inherent to the machine. We also show that parallel multisection consistently outperforms Swarztrauber's parallel bisection on machines with significant communication costs.

Generalizing the results of papers C2, [3], and J1 to the computation of many eigenvalues leads to problems in processor load balance. Bisection can be implemented in parallel by assigning the computation of different eigenvalues to different processors. The processor workload then becomes unbalanced when the different eigenvalues take different amounts of time to compute. To keep the workload balanced, busy processors must split and redistribute their computing tasks to idle processors. As the eigenvalue computation tasks are generated asynchronously and unpredictably and also require unequal times to complete, parallel bisection represents a dynamic problem. Dynamic problems contrast with static problems for which all tasks are assigned to processors a priori and with quasi-dynamic problems where tasks are generated and redistributed in a way that is synchronous and predictable in phases.

As we studied the load balance problem, we began to recognize common patterns in the different schemes for balancing the processor workload. In particular, we noted that a dynamic programming problem consists five distinct phases involving different programming issues. These phases may occur in any number and order in a given program and can be named as follows. The **Partition** phase splits a task into subtasks and takes care of all aspects related to handling the tasks and the task queue structure. The **Map** phase distributes the subtasks among a virtual graph of processors interconnected by some convenient virtual topology. The **Embed** phase embeds the virtual interconnecting topology of the processors into the actual machine architecture. The **Solve** phase represents the sequential computation that is performed on each processor. The **Communicate** phase takes care of the interprocessor communication. In Paper C1, we describe the **PMESC** programming paradigm that guides program development according to those

phases. While we developed PMESC to handle dynamic problems, it applies equally well to static and quasi-dynamic problems.

We are beginning to develop a library of programming tools based on the PMESC paradigm for distributed-memory MIMD multiprocessors. This library is designed to fill the void of tools available to aid in the building of programs requiring dynamic processor scheduling.

Paper J0 covers our most recent work on parallel solution of stochastic programming problems. Solving the deterministic equivalent formulation of two-stage stochastic programs using interior point algorithms requires the solution of linear systems of the form

$$(AD^2A^T)dy = b.$$

The constraint matrix A has a dual, block-angular structure. In this paper, we develop a parallel matrix factorization procedure using the Sherman-Morrison-Woodbury formula, based on the work of Birge and Qi [2]. This procedure requires the solution of smaller, independent systems of equations. With the use of optimal communication algorithms and careful attention to data layout we obtain a parallel implementation that achieves near perfect speedup. We demonstrate scalable performance of our implementations on an Intel iPSC/860 hypercube and a Connection Machine CM–5.

Bibliography

[1] G.S. Ammar, L. Reichel, and D.C. Sorensen. An implementation of a divide and conquer method for the unitary eigenproblem. *ACM Trans. Math. Softw.*, 18:292ff., 1992.

[2] J. Birge and L. Qi. Computing block-angular Karmarkar projections with applications to stochastic programming. *Management Science*, 34, 1988.

[3] S. Crivelli and E.R. Jessup. Toward an efficient parallel implementation of the bisection method for computing eigenvalues. In Q. Stout and M. Wolfe, editors, *Proceedings of the Sixth Distributed Memory Computing Conference*, 1991.

[4] J.J.M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36:177–95, 1981.

[5] J.J. Dongarra and D.C. Sorensen. A fully parallel algorithm for the symmetric eigenvalue problem. *SIAM J. Sci. Stat. Comput.*, 8:s139–s154, 1987.

[6] G.A. Geist and E.R. Jessup. Parallel implementation of a nonsymmetric tridiagonal eigensolver. In D. Sorensen, editor, *Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*, 1991.

[7] W. Givens. Numerical computation of the characteristic values of a real symmetric matrix. Technical Report ORNL-1574, Oak Ridge National Laboratory, 1954.

[8] I.C.F. Ipsen and E.R. Jessup. Solving the symmetric tridiagonal eigenvalue problem on the hypercube. *SIAM J. Sci. Stat. Comput., Vol. 11, No. 2*, pages 203–229, 1990.

[9] E.R. Jessup. *Parallel Solution of the Symmetric Tridiagonal Eigenproblem.* PhD thesis, Dept of Computer Science, Yale University, 1989.

[10] E.R. Jessup. A case against a divide and conquer approach to the nonsymmetric eigenvalue problem. *Applied Numerical Mathematics,* 12:403–420, 1993.

[11] E.R. Jessup and D.C. Sorensen. A parallel algorithm for computing the singular value decomposition of a matrix. *SIAM J. Matrix Anal. Appl.,* 15:530–548, 1994.

[12] S. Lo, B. Phillipe, and A. Sameh. A multiprocessor algorithm for the symmetric tridiagonal eigenvalue problem. *SIAM J. Sci. Stat. Comput.,* 8:s155–s165, 1987.

[13] H.D. Simon. Bisection is not optimal on vector processors. *SIAM J. Sci. Stat. Comput.,* 10:205–209, 1989.

[14] P. Swarztrauber. A parallel algorithm for computing the eigenvalues of a symmetric tridiagonal matrix. *Math. Comp.,* 20:651–668, 1993.

[15] J.H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.