Contemporary Mathematics
Volume 180, 1994

Finite-Element/Newton Method for Solution of Nonlinear Problems in Transport Processes Using Domain Decomposition and Nested Dissection on MIMD Parallel Computers

M. REZA MEHRABI AND ROBERT A. BROWN

ABSTRACT. Finite-element discretizations of nonlinear steady-state problems describing transport processes are solved by Newton's method using a multilevel algorithm for direct factorization of the Jacobian matrix. The factorization algorithm is based on incomplete nested dissection and domain decomposition to distribute the equations to processors of a MIMD parallel computer, where the Jacobian matrix is concurrently factorized. Sample calculations for two flow problems show reasonable computation speeds and speedups on an Intel iPSC/860 hypercube. Moreover, the execution times for these codes compare favorably with the performances for conventional finite element software on a serial, vector supercomputer and for a highly parallel commercial program for solving incompressible flow problems on a MIMD parallel computer.

1. Introduction

The development of efficient and robust algorithms on parallel computer architectures is an outstanding problem for solution of the wide variety of problems in fluid flow and heat and mass transfer that are of interest in modeling materials processing and manufacturing. The mathematical models for this class of problems are typically composed of a group of differential equations, algebraic equations and integral constraints. For steady-state (time independent) models, finite element or finite difference discretization of these models leads to large sets of nonlinear differential equations with highly structured coupling between the

1991 *Mathematics Subject Classification.* Primary 65N55, 65Y05; Secondary 65N30, 65Y20.
This research was supported by the Advanced Research Projects Agency and the Microgravity Sciences and Applications Program of the National Aeronautics and Space Administration of the United States Government.
The final version of this paper will be submitted for publication elsewhere.

variables. These dependencies are naturally asymmetric: the linearization of the nonlinear equation sets leads to asymmetric Jacobian matrices. For serial, vector computers, Newton's method, coupled with direct LU-decomposition, has proven to be an efficient method for solution of these nonlinear equation sets, at least for the discretization of problems in two space dimensions, where the structure of the nonzero entries in the Jacobian matrix makes direct LU-decomposition reasonably efficient in terms of operation count and memory usage. Indeed, finite-element/Newton algorithms have been used to solve an enormous variety of transport problems and are the basis of several commercial computer codes used for this purpose. The popularity of this algorithm comes from the robustness of Newton's method for converging to the solution of extremely nonlinear problems and also from the availability of continuation methods, based on the computation of the factors of the Jacobian matrix [**27, 17**], for mapping out multiple solutions.

The goal of the development of the algorithm described in this paper is to extend finite-element/Newton algorithms for the solution of transport problems to MIMD parallel computers like the Intel iPSC/860 hypercube available to our research group. The major effort associated with this endeavor has been the development of a an LU-factorization algorithm for the solution of large, sparse, asymmetric linear equation sets based on domain decomposition and nested dissection for partitioning the equations to the processors of the MIMD computer. The algorithm used here and described in detail by Mehrabi and Brown [**22**] is based on dividing the geometrical domain recursively into subdomains and separators using nested dissection [**8, 9**] and assigning subdomains and associated separator data to each processor. The algorithm described in [**22**] extends the work of Lucas et al. [**19**] for symmetric, positive-definite matrices to avoid duplicate storage, to allow asymmetric matrices and to allow partial pivoting during factorization. The algorithm stores the lower and upper triangular forms and performs forward and backward solutions separately.

The LU-factorization algorithm is described in more detail in Section 2. Two test problems that arise in the solution of isothermal and nonisothermal incompressible flows are described in Section 3; these are the two-dimensional flow of an incompressible fluid in a lid-driven cavity and the natural convection motion of a Boussinesq fluid in a two-dimensional cavity with differentially heated lateral walls. Both problems have been widely used as test problems for incompressible flow calculations and become computationally difficult as the intensity of the convection is increased. This occurs with increasing the Reynolds number for the lid-driven cavity problem [**24, 21**] and with increasing the Grashof number for the thermal convection problem [**6**]. Because the finite element discretizations used to create the discrete equation sets are standard, we do not focus on the accuracy of the computations, but rather on their efficiency compared to implementation of the finite-element/Newton method on a serial, vector supercomputer and to a highly parallelizable spectral element code based on

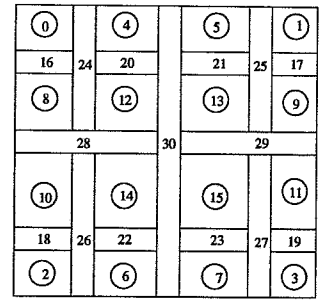

FIGURE 1. Nested dissection ordering of a rectangular two-dimensional domain. A binary reflected Gray code is used for numbering the subdomains to reduce hypercube communication [4].

pseudo-time-stepping to compute the steady-state solution.

2. Concurrent *LU*-Factorization and Storage: CFS

For simplicity, we consider a mathematical model for transport processes described in a two-dimensional geometrical region represented by the rectangular domain shown in Fig. 1. This region is partitioned into P quadrilateral subdomains, each of which is assigned to a processor of the MIMD parallel computer. The equations that correspond to each subdomain are ordered using incomplete nested dissection [9]. Then, each point on the elimination tree for the nested dissection ordering corresponds to a supernode or a group of equations [22].

In general, the mathematical model may be composed of differential and algebraic equations, integral constraints and boundary conditions. Finite element methods are used to define local approximations to the field variables and to discretize the partial differential equations and boundary conditions. The details of these discretizations are not important for the discussion here, as long as the approximations to the field variables have compact support within the elements. The discretized problem is a large set of nonlinear algebraic equations that is represented as

$$\mathbf{R}(\mathbf{x}) = \mathbf{0} \tag{2.1}$$

where $\mathbf{R} \in \mathbb{R}^N$ and $\mathbf{x} \in \mathbb{R}^N$, where N is the total dimension of the discrete field variables. The variables and equations are associated with specific processors according to $\mathbf{x} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, ..., \mathbf{x}^{(P)})$ and $\mathbf{R} = (\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{R}^{(3)}, ..., \mathbf{R}^{(P)})$, where the dimensions of the variables $\{\mathbf{x}^{(i)}\}$ and equations $\{\mathbf{R}^{(i)}\}$ associated with each processor depend on the number of finite elements allocated to it.

We solve eqs. (2.1) by Newton's method using direct *LU*-decomposition of the Jacobian matrix at each Newton iteration. The Jacobian matrix is written as a

partitioned matrix among the P-processors as

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}^{(ii)} & \mathbf{J}^{(ie)} \\ \mathbf{J}^{(ei)} & \mathbf{J}^{(ee)} \end{bmatrix} \tag{2.2}$$

where the Jacobian matrix is organized into data that is totally interior to each subdomain – the interior matrix $\{\mathbf{J}^{(ii)}, \mathbf{J}^{(ei)}, \mathbf{J}^{(ie)}\}$ – and data that corresponds to the borders shared by two or more subdomains, which are the separators in Fig. 1 – the exterior matrix $\mathbf{J}^{(ee)}$. The exterior matrix is partitioned among adjacent processors to minimize communication and memory requirements. The data structures used for storage of the interior and exterior matrices are described in [**22**]. Matrix data corresponding to each supernode is stored in block from to facilitate vectorization and pivoting during the LU-factorization.

The algorithm for the LU-decomposition of the matrix, eq. (2.2), described by Mehrabi and Brown [**22**] is operationally equivalent to direct computation and factorization of the Schur complement. The formation of the Schur complement corresponds to the LU-decomposition of the interior matrices of each processor $\{\mathbf{J}^{(ii)}\}$ and to the update of the exterior matrix $\mathbf{J}^{(ee)}$ by those factors. Data is communicated from each processor to form the exterior matrix using the *fan-in* method of Ashcraft et al. [**1**]. A hybrid *fan-in/fan-out* algorithm is used for updating the exterior matrices during factorization and subsequent updating. As discussed in [**22**], the fan-in and the hybrid fan-in/fan-out methods of update reduce the number of messages and the total message volume for LU-decomposition by approximately factors of 2 and 3 over the algorithm of Mu and Rice [**23**] developed using a grid-based, subtree-subcube assignment of exterior matrix elements to the processors.

The computer code was written in standard Fortran and C with only machine specific statements for send, receive and synchronization. The code was implemented on a 32-node Intel iPSC/860 hypercube with 8 Mbytes of memory per processor and compiled with Intel's *if77* and *icc* compilers. Hand-coded Level-1 BLAS by Kuck and Associates [**18**] were used to increase the speed. Each i860 node executes the LINPACK benchmark at 9.5 MFLOPS and the machine has a ratio of speed for communication to computation of 1:100, which lowers the parallel speedup when communication is large. Here the speedup is defined as the speed of a calculation on P-processors divided by the speed on a single processor of the same machine, where the single processor solves a problem of similar size to the problem of each of the P-processors. Below we report speedups for the LU-factorization, for forward and backward elimination steps of the solution of linear equation sets and for the formation of the matrix and forcing vector at each Newton iteration.

3. Test Problems and Results

The two test problems are shown schematically in Fig. 2. Each of these problems is described below.

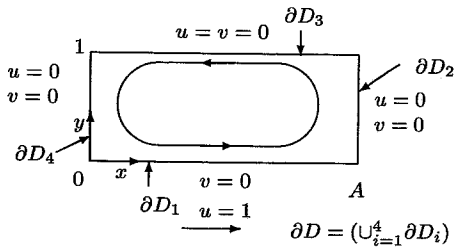

FIGURE 2a

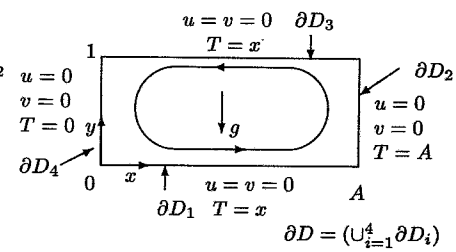

FIGURE 2b

FIGURE 2. Schematic diagrams of the flow geometries for the problems of flow in a (a) lid-driven cavity and (b) thermal convection in heated side-wall cavity. The boundary conditions for each flow problem also are shown.

3.1. Lid-Driven Cavity. Consider the steady-state incompressible flow of a Newtonian liquid in a two-dimensional cavity with the bottom surface moving, as shown in Fig. 2a. The Navier-Stokes equations are made dimensionless with the speed of the bottom surface and the height of the cavity to give the dimensionless equations

$$\text{Re}\, \mathbf{v} \cdot \nabla \mathbf{v} = \nabla^2 \mathbf{v} - \nabla p \tag{3.1}$$

$$\nabla \cdot \mathbf{v} = 0 \tag{3.2}$$

where the boundary conditions are shown in Fig. 2a. In addition to setting both components of velocity on each surface, the pressure is specified at a point to set a datum level. The problem is specified in terms of two dimensionless parameters: the Reynolds number Re, which scales the importance of fluid inertia, and the aspect ratio of the cavity A, which scales the cavity width to its height. Including a nonzero value of Re makes the problem nonlinear and the Jacobian matrix asymmetric. The calculations presented here are meant only as a demonstration of the parallelization of the algorithm; they were performed with parameter values of A=4 and Re=1. Calculations with large Re are feasible with the finite element meshes used here.

The eqs. (3.1)-(3.2) are discretized using a standard mixed finite element method for the Stokes problem [3]. The velocity components are interpolated using Lagrangian biquadratic polynomials defined on each element and the pressure by bilinear polynomials. The weak forms of the eqs. (3.1)-(3.2) are written using the isoparametric mapping to a unit element and the two-dimensional integrals are computed numerically using 9-point Gaussian quadrature. The Newton's iteration was started with the initial guess of no flow in the cavity and converged to an absolute tolerance of 10^{-12} in 4 iterations. A sample solution for a mesh of 36 elements in the vertical direction and 76 elements in the horizontal

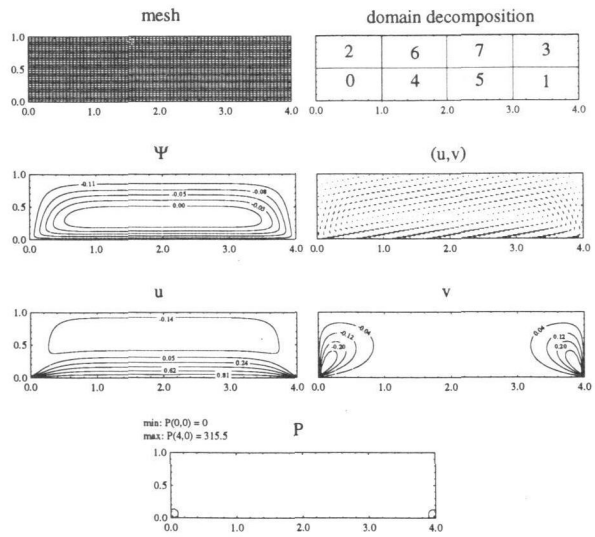

FIGURE 3. Sample mesh, subdomains, velocity components and pressure field for the flow in a lid-driven cavity. Calculations are for a 36×76 element mesh with N=25,000 degrees-of-freedom (DOF's) on 8 processors.

direction is shown in Fig. 3. Calculations with varying mesh size show a 5 percent oscillation in the maximum magnitude of the velocity due to singularities in the pressure field and its gradient at the corners where the moving and stationary walls meet. It is possible to circumvent the convergence problem caused by these singularities by changing the boundary conditions in the corners; however, we did not attempt to do so because we are only interested in the performance of the algorithm at this point, not in the details of the flow field.

The computation times for formulation of the Jacobian matrix and forcing vector and for LU-decomposition are shown in Fig. 4 for calculations with between 1 and 32 processors (P) and discretizations resulting in 4,000 to 80,000 degrees-of-freedom (DOF's). Two features are important. First, the time for calculation decreases proportionally to the number of processors, showing good parallel scaling for both operations. For the largest problem (N=80,000), the time for formulation (8.5 s) is a fraction of the time needed for LU-decomposition (28 s). The total time for a Newton iteration is the summation of these times with the time for forward and backward solution of the triangular equation sets. For the calculation with N=80,000, this is

$$\begin{aligned} \frac{\text{Total Time}}{\text{Newton Iteration}} &= 8.5 \text{ s (Formulation)} + 28 \text{ s } (LU\text{-Decomposition}) \\ &+ 1.1 \text{ s (Forward Elimination)} \\ &+ 0.6 \text{ s (Backward Elimination)} \\ &= 38.2 \text{ s} \end{aligned}$$

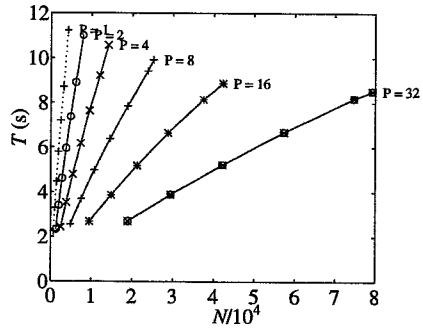

FIGURE 4a

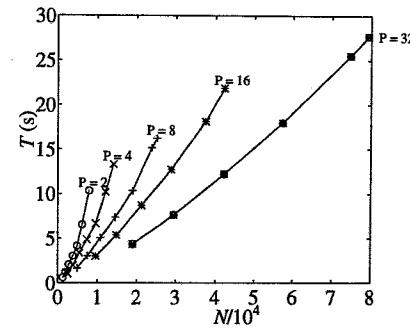

FIGURE 4b

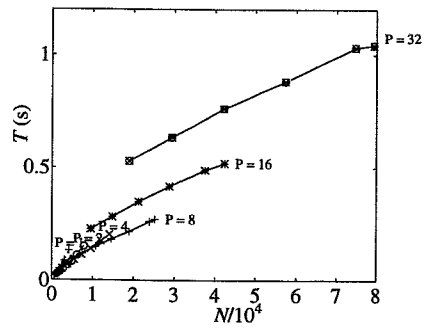

FIGURE 4c

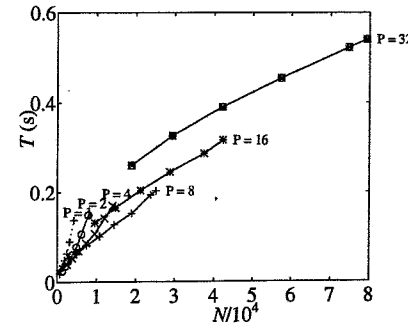

FIGURE 4d

FIGURE 4. Execution times for (a) formulation, (b) LU-decomposition, (c) forward elimination and (d) backward elimination for solution of the lid-driven cavity problem as a function of problem size (N) and number of processors (P).

where formulation represents 22 percent and LU-decomposition 73 percent; the other two steps represent 3 and 2 percent of the total computation time, respectively.

The low percentage time spent performing the forward and backward solution algorithms offsets the poor parallelization of these two algorithms, as is well known; e.g. see Lucas et al. [**19**]. As seen in Fig. 4, the performance of both algorithms degrades with increasing the number of processors for a constant problem size (N) because of the increased communication needed in these steps of the algorithm.

The lid-driven cavity problem was used as a benchmark to establish the performance of the finite-element/Newton method using the CFS algorithm for solution of linear equation sets relative to the finite-element/Newton method implemented using a frontal solution method [**16, 14, 15**] on a vector supercomputer. Two sets of calculations are reported. In the first, both the CFS and

TABLE 1. Comparison of finite-element/Newton method using the CFS LU-decomposition software and the serial frontal solution algorithm. The domain is dissected 6 times by CFS. Both programs are run on a single, dedicated processor of a Cray X-MP.

Quantity	Serial Frontal Code	CFS Code
Time for Newton Iteration (s)	162	144
Time for Solution (s)	900	780
Speed for LU-Decomposition (MFLOPS)	110	45

frontal algorithms are run on a single processor of the Cray X-MP computer at the MIT Supercomputer Facility to establish the advantages of the incomplete nested dissection algorithm relative to the frontal technique outside of the context of the parallel computer. In the second calculation, the CFS algorithm run on the Intel iPSC/860 is compared to the frontal algorithm running on the Cray X-MP.

The comparison between the frontal version of the finite-element/Newton method and the CFS implementation on the Cray X-MP is summarized in Table 1 for a calculation with a 68×68 mesh, giving N=42,000; the speeds for the two codes are included there. The speed for the frontal code is taken from diagnostic software supplied by Cray. The speed for the CFS code is calculated directly by counting the number of operations and dividing it by time. The frontal code uses Level-1 BLAS routines in the innermost loops of the LU-decomposition routine and runs at 110 MFLOPS on the Cray. By comparison, no explicit attempt has been made to vectorize the CFS code on the Cray; as a result it executes only at 45 MFLOPS. The test run by CFS is dissected 6 times, i.e. into 2^6 sections. Further dissection degrades this speed more because of smaller length of vectors and the larger overhead of integer addressing. Even though the CFS routine is not as fast as the frontal solver, it still executes in a slightly lower time, because of the lower operation count associated with the incomplete nested dissection ordering of the matrix [**11, 10**].

The comparison between the frontal version of the finite-element/Newton method running on the Cray X-MP and the CFS version running on the 32-processor Intel iPSC/860 for a discretization with N=80,000 (64×136 elements) is summarized in Table 2. Most significantly, the CFS version is almost 10 times faster for performing the same calculation. The Intel machine is performing at half of the maximum speed of 32×9.5=304 MFLOPS estimated from the LINPACK benchmark run on a single processor.

TABLE 2. Comparison of finite-element/Newton method using the CFS LU-decomposition software on a MIMD computer, where each subdomain is dissected 6 times, and the frontal implementation of LU-decomposition on a serial, vector supercomputer.

Quantity	Serial Frontal Code on a single processor of Cray X-MP	CFS Code on 32-Node Intel iPSC/860
Time for Newton Iteration (s)	273	38
Time for Solution (s)	1800	180

3.2. Thermal Convection in a Cavity with Heated Sidewalls. The geometry for this problem is shown schematically in Fig. 2b. The equations are made dimensionless with the height of the cavity, the temperature difference between the hot and cold walls and the buoyant velocity to yield the dimensionless momentum, continuity, and energy equations

$$\sqrt{\mathrm{Gr}}\,\mathbf{v}\cdot\nabla\mathbf{v} = \nabla^2\mathbf{v} - \nabla p + \sqrt{\mathrm{Gr}}\,\mathbf{e}_y\,T \tag{3.3}$$

$$\nabla\cdot\mathbf{v} = 0 \tag{3.4}$$

$$\sqrt{\mathrm{Gr}}\,\mathbf{v}\cdot\nabla T = \frac{1}{\mathrm{Pr}}\,\nabla^2 T \tag{3.5}$$

which are expressed in terms of the Grashof number (Gr), the Prandtl number (Pr) and the aspect ratio A of the cavity. In eq. (3.3), $\mathbf{e}_y$ is the unit vector in the y-direction. The boundary conditions for the velocity and temperature field are shown in Fig. 2b; here the temperature field along the horizontal surfaces has been specified as a linear function of position to match the temperatures at the two vertical surfaces; these conditions are the same as those used by others [**6**] in the specification of this problem as a test for algorithms for computation of natural convection.

Equations (3.3)-(3.5) are discretized using a standard mixed finite element method for the natural convection problem [**12**]. The velocity components and the temperature are interpolated using Lagrangian biquadratic polynomials defined on each element and the pressure by bilinear polynomials. The weak form of the eqs. (3.3)-(3.5) is constructed in a similar way to what was described for the lid-driven cavity problem.

Calculations were carried out using the parameter values of A=4, Pr=0.015 and Gr=20,000 to demonstrate the robustness of the algorithm for computing a highly nonlinear steady-state. Setting the Prandtl number on this low value makes the time scales for heat and momentum diffusion very different and makes this computation particularly hard for any algorithm based on a pseudo-time-stepping procedure for computing the steady-state solution. The Newton's iterations converge in 8 iterations starting with the converged solution for A=4,

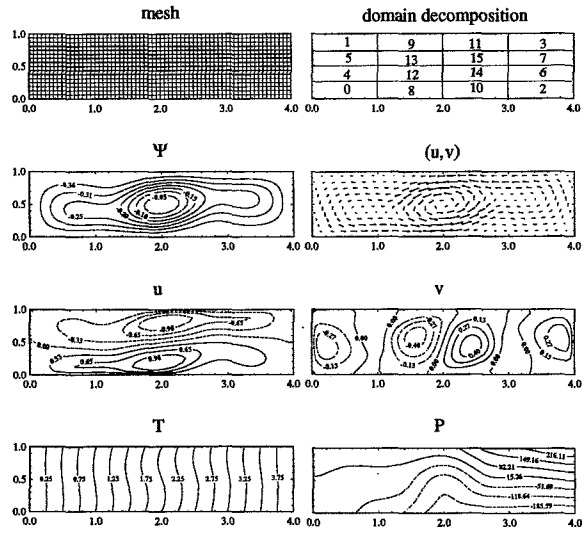

FIGURE 5. Sample mesh, subdomains, velocity components, temperature field and pressure field for the thermal convection in a cavity with heated sidewalls. Calculations are for a 16×64 element mesh with N=14,000 degrees-of-freedom (DOF's).

Pr=0.015, and Gr=5,000 as the initial approximation, which in turn converges in 5 iterations from the no flow state as the initial approximation. The solution contours computed with a mesh of 16×64 elements are shown in Fig. 5 and compare well with solutions reported by others [**5, 2**] for this same flow state.

The speedups associated with the calculations on the Intel iPSC/860 are shown in Fig. 6 for the formulation and LU-decomposition as a function of the DOF's and the number of processors P. The efficiency is approximately 0.85 for the formulation and 0.65 for LU-decomposition; both values are essentially independent of the problem size.

The computational efficiency of the finite-element/Newton method for solution of this thermal convection problem was compared directly to a highly optimized commercial incompressible flow code, Nekton, which uses spectral element discretizations coupled with a semi-implicit temporal operator splitting to compute steady-state solutions as the asymptotic limit of transients [**20, 25, 13, 26, 7**]. The operator splitting method requires only the solution of positive-definite linear equation sets at each time step. These systems are solved using a conjugate gradient algorithm with a two-level preconditioner. The results of the comparison tests run on the Intel iPSC/860 are summarized in Table 3. The CFS calculation is nearly 7 times faster. The comparison has been based on the solution of discretizations of similar size (N). For this discretization, the spectral element representation (11th-order polynomials for velocity and temperature

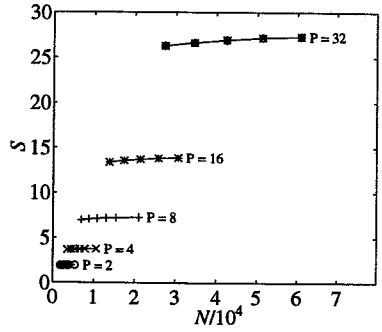

FIGURE 6a

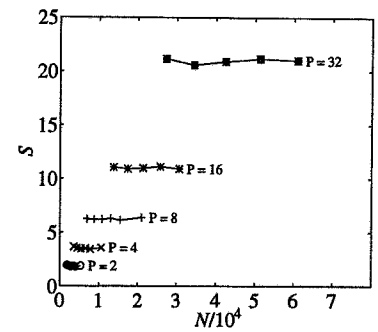

FIGURE 6b

FIGURE 6. Speedups for (a) formulation, and (b) *LU*-decomposition for solution of the thermal convection problem as a function of problem size (N) and number of processors (P).

and 9th-order for pressure) is much more accurate than the finite element approximation. However, the CFS algorithm also is applicable to spectral element discretizations and will yield improved computation rates because of the reduced size of the exterior matrix relative to the interior matrix. A direct comparison on spectral element calculations will be reported later. The two test calculations reported here make a reasonably fair comparison.

4. Discussion

The calculations presented here demonstrate that the finite-element/Newton method coupled with the CFS algorithm for *LU*-factorization is a robust and efficient method for the solution of two-dimensional nonlinear transport problems using MIMD parallel computers. We believe that the utility of the method will be even greater for problems with more complicated physicochemical processes, such as complicated kinetics, radiative heat transfer and nonlinear constitutive behavior, that are present in many materials processing problems of interest. The results show that the speedup associated with the *LU*-decomposition is not particularly high; it is shown to be 0.65 for 32-processors and estimated to be as low as 0.5 for 512-processors. However, the robustness of the *LU*-decomposition for solution of the linear equation set and the rapid convergence of Newton's iterations for nonlinear problems makes the finite-element/Newton method a viable algorithm for solving any problem that can be fit into memory of the MIMD computer. Current machine sizes limit this approach to large two-dimensional and small three-dimensional problems.

TABLE 3. Comparison of finite-element/Newton method based on CFS algorithm with Nekton, a spectral element code based on semi-implicit, pseudo-time-stepping method with preconditioned conjugate gradient method. Both calculations were run on Intel iPSC/860. Each subdomain is dissected 6 times by CFS. For this calculation Gr=1, Pr=0.01 and A=4.

Quantity	Nekton	Finite-Element/Newton CFS method
Steps/Iterations	100 time steps	3 Newton iterations
Mesh	8×16 spectral elements	48×96 finite elements
Order of Approximations	Velocity: 11 Temperature: 11 Pressure: 9	Velocity: 2 Temperature: 2 Pressure: 1
DOF's	58,000	61,000
Number of Processors	P=8*	P=32
Solution Time for P=32	800	120

* Solution time on P=32 estimated by dividing time on P=8 by 4.

Acknowledgements

The authors would like to thank Paul Fischer for providing the results by Nekton, and Todd Salamon and David Bornside for providing the results by the serial frontal algorithm.

REFERENCES

1. C. Ashcraft, S. Eisenstat, and J. Liu, *A fan-in algorithm for distributed sparse numerical factorization*, SIAM J. Sci. Statist. Comput. **11** (1990), 593–599.
2. M. Behnia and G. de Vahl Davis, *Fine mesh solutions using stream function-vorticity formulation*, Numerical Simulation of Oscillatory Convection in Low-Pr Fluids (Braunschweig, Germany) (B. Roux, ed.), Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, 1990, pp. 11–18.
3. G. F. Carey and J. T. Oden, *Finite elements*, vol. 6, Prentice Hall, Englewood Cliffs, NJ, 1986.
4. T. Chan and Y. Saad, *Multigrid algorithms on the hypercube multiprocessor*, IEEE Trans. on Computers **C-35** (1986), 969–977.
5. O. Daube and S. Rida, *Contribution to the GAMM workshop*, Numerical Simulation of Oscillatory Convection in Low-Pr Fluids (Braunschweig, Germany) (B. Roux, ed.), Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, 1990.
6. B. Roux (ed.), *Numerical simulation of oscillatory convection in low-Pr fluids*, Notes on Numerical Fluid Mechanics, vol. 27, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig, Germany, 1990.
7. P. F. Fischer, *Spectral element solution of the navier-stokes equations on high performance distributed-memory parallel processors*, Ph.D. thesis, MIT, Cambridge, MA, June 1989.
8. A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal. **10** (1973), 345–367.
9. A. George and J. W. H. Liu, *Computer solution of large sparse positive definite systems*, Prentice Hall, Englewood Cliffs, NJ, 1981.
10. A. George and E. Ng, *On the complexity of sparse QR and LU factorization of finite-*

element matrices, SIAM J. Sci. Statist. Comput. **9** (1988), 849–861.
11. A. George and H. Rashwan, *Auxiliary storage methods for solving finite element systems*, SIAM J. Sci. Statist. Comput. **6** (1985), 882–910.
12. P. M. Gresho, R. L Lee, S. T. Chan, and J. M. Leone, *A new finite element for Boussinesq fluids*, Proc. 3rd Intl. Conf. Finite Elements in Flow Problems (Alberta, Canada) (D. H. Norrie, ed.), Banff, 1980, pp. 204–215.
13. L. W. Ho and A. T. Patera, *A Legendre spectral element method for simulation of unsteady incompressible viscous free-surface flows*, Computer Methods in Applied Mechanics and Engineering **80** (1990), 355–366.
14. P. Hood, *Frontal solution program for unsymmetric matrices*, J. Num. Meth. Engng. **10** (1976), 379–399.
15. ———, *Note on frontal solution program for unsymmetric matrices*, J. Num. Meth. Engng. **11** (1977), 1055.
16. B. M. Irons, *A frontal solution program for finite element analysis*, Intl. J. Num. Meth. Engng. **2** (1970), 5–32.
17. H. B. Keller, *Numerical solution of bifurcation and nonlinear eigenvalue problems*, Applications of Bifurcation Theory (New York, San Francisco, London) (P. H. Rabinowitz, ed.), Academic Press, Inc., 1977, pp. 359–384.
18. Kuck and Associates, Inc., Champaign, IL, *CLASSPACK basic math library*, February 1992.
19. R. F. Lucas, T. Blank, and J. J. Tiemann, *A parallel solution method for large sparse systems of equations*, IEEE Trans. Computer-Aided Design **CAD-6** (1987), 981–991.
20. Y. Maday and A. T. Patera, *Spectral element methods for the navier-stokes equations*, State-of-the-Art Surveys in Computational Mechanics (New York, NY) (A. K. Noor and J. T. Oden, eds.), American Society of Mechanical Engineers, 1989.
21. A. Mahallati and J. Militzer, *Application of the piecewise parabolic finite analytic method to the three-dimensional cavity flow*, Numerical Heat Transfer, Part B **24** (1993), 337–351.
22. M. R. Mehrabi and R. A. Brown, *An incomplete nested dissection algorithm for parallel solution of finite element discretizations of partial differential equations.*, J. Sci. Computing **8** (1993), 373–387.
23. Mo Mu and J. R. Rice, *A grid-based subtree-subcube assignment strategy for solving partial differential equations on hypercubes*, SIAM J. Sci. Stat. Comput. **13** (1992), 826–839.
24. H. Nishida and N. Satofuka, *Higher-order solutions of square driven cavity flow using a variable-order multi-grid method*, Intl. J. Numer. Meths. Engng. **34** (1992), 637–653.
25. A. T. Patera, *A spectral method for fluid dynamics, laminar flow in a channel expansion*, Journal of Computational Physics **54** (1984), 468–488.
26. E. M. Ronquist, *Optimal spectral element methods for the navier-stokes equations*, Ph.D. thesis, MIT, Cambridge, MA, 1988.
27. Y. Yamaguchi, C. J. Chang, and R. A. Brown, *Multiple buoyancy-driven flows in a vertical cylinder heated from below*, Phil. Trans. R. Soc. Lond. **A312** (1984), 519–552.

Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139
E-mail address: reza@mit.edu and rab@mit.edu