ornl

ORNL/TM-12733

OAK RIDGE NATIONAL LABORATORY

MARTIN MARIETTA

Implementation of Parallel Matrix Decomposition for NIKE3D on the KSR1 System

Philip S. Su
Robert E. Fulton
Thomas Zacharia

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

ORNL/TM-12733

Metals and Ceramics Division

IMPLEMENTATION OF PARALLEL MATRIX DECOMPOSITION FOR NIKE3D ON THE KSR1 SYSTEM

Philip S. Su, Robert E. Fulton, and Thomas Zacharia

Date Published: June 1995

Prepared for
U.S. Department of Energy
Office of Basic Energy Sciences
KC 02 01 05 0

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6080
managed by
LOCKHEED MARTIN ENERGY SYSTEMS
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

CONTENTS

LIST OF FIGURES

IMPLEMENTATION OF PARALLEL MATRIX DECOMPOSITION FOR NIKE3D ON THE KSR1 SYSTEM*

Philip S. Su†, Robert E. Fulton‡, and Thomas Zacharia

ABSTRACT

New massively parallel computer architecture has revolutionized the design of computer algorithms and promises to have significant influence on algorithms for engineering computations. Realistic engineering problems using finite element analysis typically imply excessively large computational requirements. Parallel supercomputers that have the potential for significantly increasing calculation speeds can meet these computational requirements. This report explores the potential for the parallel Cholesky (U^TDU) matrix decomposition algorithm on NIKE3D through actual computations. The examples of two- and three-dimensional nonlinear dynamic finite element problems are presented on the Kendall Square Research (KSR1) multiprocessor system, with 64 processors, at Oak Ridge National Laboratory. The numerical results indicate that the parallel Cholesky (U^TDU) matrix decomposition algorithm is attractive for NIKE3D under multiprocessor system environments.

---

1. INTRODUCTION

The emergence of parallel computers has revolutionized approaches to numerical solutions of large-scale engineering problems. New algorithms are being developed by researchers in engineering and computer science in order to exploit the tremendous potential of parallel computing. Parallel computers have thus become important tools to solve complex and computationally intensive science and engineering problems.

---

*This research was supported in part by an appointment to the Oak Ridge National Laboratory Postdoctoral Research Associates Program administered jointly by the Oak Ridge National Laboratory and the Oak Ridge Institute for Science and Education and was sponsored by the Division of Materials Sciences, U.S. Department of Energy, under contract DE-AC05-84OR21400 with Lockheed Martin Energy Systems.

†Postdoctoral Research Associate, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6140.

‡Professor, School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0405.

Finite element methods are the basis for numerical solutions to engineering problems. Solving the associated simultaneous equations can take 50% of the total computation time. Realistic engineering problems using such analysis typically require excessively large computation times. Parallel supercomputers have the potential for significantly increasing calculation speeds in order to meet these computational requirements.

This report explores the parallel Cholesky (U^TDU) matrix decomposition algorithm on the NIKE3D finite element code for two-dimensional (2-D) and three-dimensional (3-D) nonlinear dynamic finite element structural system problems. The algorithm was implemented on the Kendall Square Research (KSR1) supercomputer system at Oak Ridge National Laboratory (ORNL). Numerical results indicate that the algorithm is highly adaptive to a multiprocessor system environment, and a significant speed-up is observed.

1.1 NIKE3D FINITE ELEMENT CODE

NIKE3D (ref. 1) is an implicit finite element code for analyzing the finite strain static and dynamic responses of 3-D inelastic solids, shells, and beams. The finite element formulation accounts for both material and geometric nonlinearities. The code is fully vectorized and available on several computer platforms. A number of material models are incorporated to simulate a wide range of material behavior, including elastoplasticity, anisotropy, creep, thermal effects, and rate dependence. Slideline algorithms model gaps and sliding along material interfaces, including interface friction and single surface contact. Interactive graphics and rezoning are available for analyses with large mesh distortions. Several nonlinear solution strategies are available, including full-, modified-, and quasi-Newton methods. The resulting system of simultaneous linear equations is either solved iteratively by an element-by-element method or directly by a factorization method, for which case bandwidth minimization is optional. Data may be stored either in or out of core memory to allow for large analysis.

As a Lagrangian implicit code, NIKE3D is ideally suited for quasi-static and low-rate dynamic problems of solids, shells, and beams. NIKE3D uses a small number of relatively large time steps. A coupled system of nonlinear algebraic equations is solved at each time step by a linearization and iteration procedure. A robust and efficient nonlinear solution strategy is an essential capability for the severely nonlinear problems typically solved by NIKE3D.

Ongoing parallel computation research and development activities at ORNL have focussed on exploring parallelization of NIKE3D's application on a multiprocessor system such as KSR1. Since the equation solving can use up to one-half of the total computation time, exploration of the parallel version of the popular direct matrix factorization algorithm on the multiprocessor system for the nonlinear problems should be the first step. The parallel Cholesky (U^TDU) matrix decomposition algorithm and skyline data storage scheme were used for the implementation.

1.2 KSR1 MULTIPROCESSOR SYSTEM

The KSR1 architecture is a multiprocessor system composed of a hierarchy of rings. The lowest level, ring:0, consists of a 34-slot backplane connecting 32 processing cells and 2 cells responsible for routing to the next higher layer ring, ring:1. A fully populated ring:1 is composed of the interconnecting cells from 32 ring:0 rings. A fully configured KSR1 multiprocessor system is composed of 2 layers containing 1024 processing cells along with 2 ring interconnecting cells on each ring:0. Figure 1 shows the hierarchical ring structure of the KSR1 multiprocessor system.

Each processor contains a 256-KB data cache and a 256-KB instruction cache. The on-board data and instruction caches are referred to as *subcaches.* A daughter board connected to each processing cell contains 32 MB of memory referred to as *local cache.* The word size of the KSR1 multiprocessor system is 64 bits, and all functional units are based on 64-bit operands. All of the local caches make up the ALLCACHE memory system. The KSR1 architecture and chip set are designed specifically to support a shared-memory multiprocessor. Reference 2 provides more detail on the actual implementation. Each processor has the peak performance of 40 MFLOPS. The KSR1 multiprocessor system at ORNL has 64 (2 ring:0 rings) processors.

2. PARALLEL MATRIX DECOMPOSITION

The parallel Cholesky (U^TDU) matrix decomposition algorithm uses a column-wise reduction scheme to compute upper triangular and diagonal factors of a sparse symmetric

ORNL-DWG 95-5901

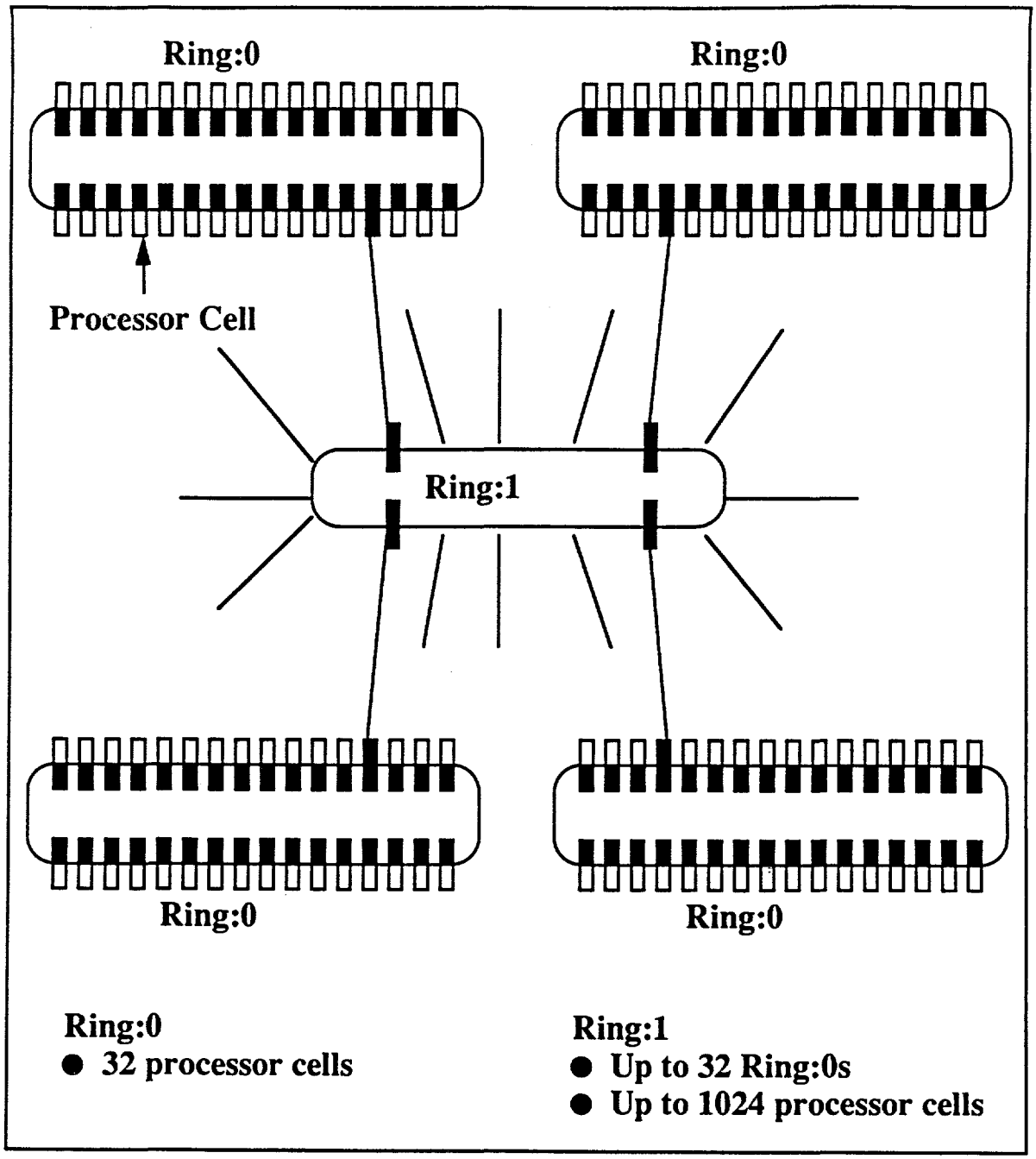

Fig. 1. The architecture of the KSR1 multiprocessor system.

matrix K (refs. 3, 4). The terms U_{ij} of the upper triangular matrix U may be obtained from:

$$L_{ji} = K_{ij} - \sum_{m=1}^{i-1} U_{mi} L_{jm} \qquad i=1,\ldots,j-1 \quad . \tag{1}$$

and

$$U_{ij} = \frac{L_{ji}}{D_{ii}} \qquad i=1,\ldots,j-1 \quad . \tag{2}$$

The above equations are evaluated for j = 1,...,n where n is the number of columns in K. The corresponding diagonal terms D_{jj} are computed from:

$$D_{jj} = K_{jj} - \sum_{m=1}^{j-1} L_{jm} U_{mj} \qquad j=1,\ldots,n \quad . \tag{3}$$

The decomposition for matrix K proceeds in a column-wise pattern when m, i, and j are used as inner, intermediate, and outer loop indices. Through the above algorithm, the factor terms may replace the corresponding matrix terms in memory as they are determined.

From Eqs. (1) through (3), it is evident that the envelope of the original matrix and the factor matrices are identical inside the band; however, zero terms may or may not become non-zero. In the case of finite element method applications, the sparsity of the factor matrix is generally less than the sparsity of the original matrix. This "fill-in" phenomenon is the reason why, in the present algorithm, all in-band zeros are stored and operated on. Hence, the total number of operations is not an absolute minimum but is time consuming for additional testing, and skipping would be required in the equation solution to skip "non-fill-in zeros."

The Cholesky (U^TDU) decomposition algorithm has a high level of inherent parallelism. In the present procedure, concurrence is realized on the column level, and the parallel granularity is proportional to the square of the half bandwidth. One processor computes all terms U_{ij} (i = 1,...,j-1) and the diagonal term D_{jj} in one particular column j using Eqs. (1) through (3). Concurrently, the other processors factorized the neighboring

columns j+1, j+2, etc. However, these tasks are not completely independent. For example, writing Eq. (1) for column j+1 yields:

$$L_{j+1,i} = K_{i,j+1} - \sum_{m=1}^{i-1} U_{mi} L_{j+1,m} \qquad i=1,\ldots,j \quad . \qquad (4)$$

For i = j, the term $U_{j-1,j}$ is required to evaluate Eq. (4). Hence, $L_{j+1,j}$ cannot be computed before the previous column is completely decomposed; this is an interprocess synchronization point.

If p denotes the number of processors, each column (i.e., each parallel task) has p-1 synchronization points. Software flags and lock variables are used to synchronize the decomposition procedure when the tasks reach the critical region at the bottom of the column. The distribution of columns among processors is done in a controlled scheduling loop, i.e., when a processor has finished its column, it automatically identifies and claims the next not-yet-decomposed column according to a specific pattern. Figure 2 (ref. 5) illustrates the parallel decomposition processing algorithm.

3. NUMERICAL EXAMPLES

The parallel Cholesky (U^TDU) matrix decomposition algorithm was implemented for two symmetric, perfect-banded matrices to explore the speed-up performance of the algorithm. Then, this algorithm was applied to the NIKE3D code and tested for two nonlinear, dynamic 2-D and 3-D finite element problems. The NIKE3D applications were run with in-core mode to eliminate the disk input/output timing. All the computations were performed on the KSR1 multiprocessor system, with 64 processors, at ORNL.

The KSR1 compilers provide two levels of optimization for the object code: one is the global optimization which is mainly composed of constant folding, common subexpression elimination, strength reduction, peephole optimization, global register allocation, pipeline scheduler, argument passing in registers, and subroutine inlining; the other is the automatic loop unrolling.[6] These two levels of optimization can reduce a lot of execution time for most of the serial finite element codes. The new subroutine which contains the parallel matrix decomposition uses both the global optimization and the automatic loop unrolling.

ORNL-DWG 95-5902

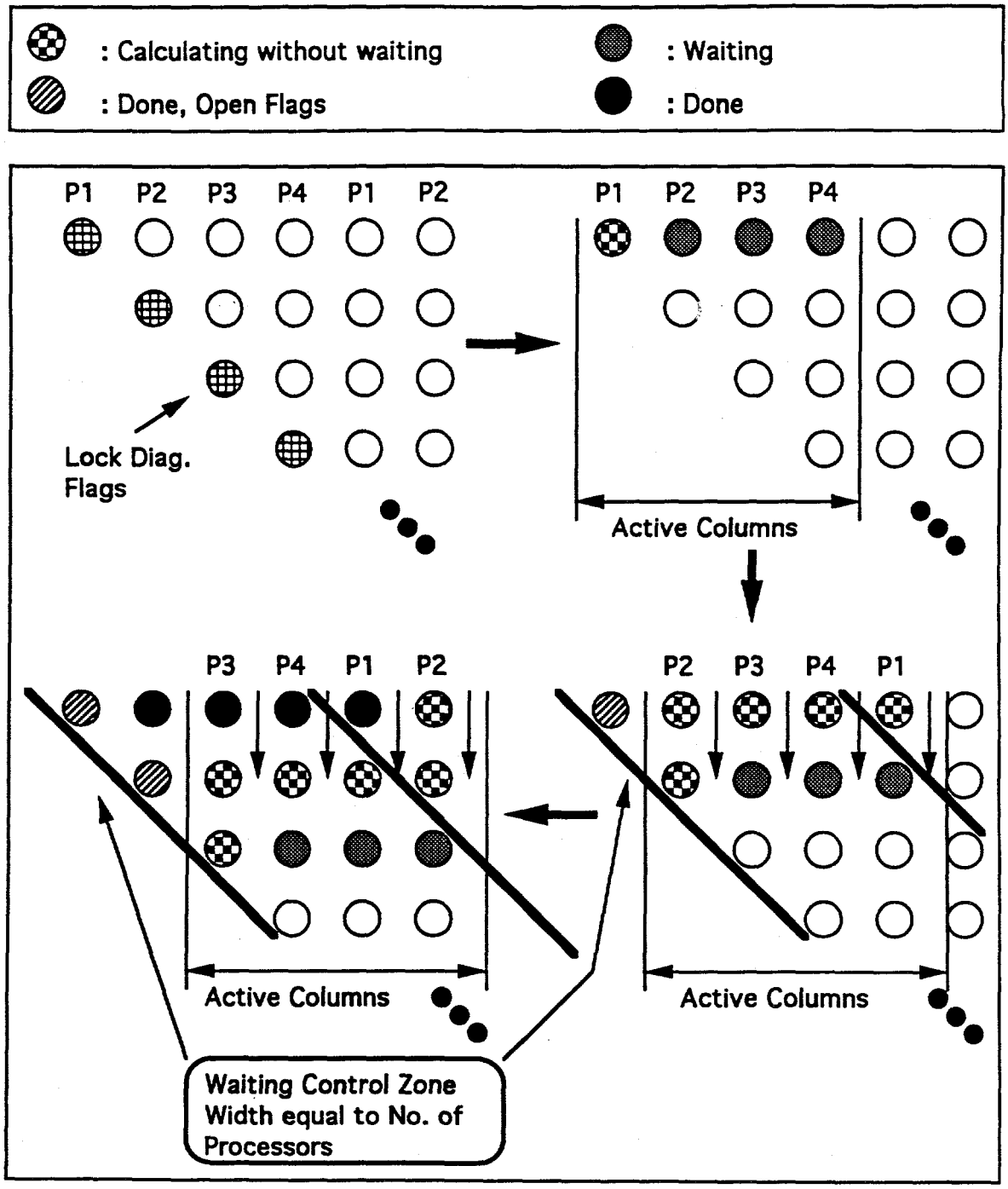

Fig. 2. The parallel decomposition processing algorithm.

However, the original NIKE3D code was developed several years ago, and all subroutines were not considered to be optimized on a supercomputer system such as KSR1. The KSR1 compilers can optimize NIKE3D code up to the second level for the 2-D shell elements case but not even for the first level for the 3-D solid elements case.

3.1 PARALLEL CHOLESKY (U^TDU) MATRIX DECOMPOSITION

There are several test results for the parallel Cholesky (U^TDU) matrix decomposition performance on the KSR1 multiprocessor system.[6] However, all those results are up to 28 processors which are within one ring:0 ring. The results shown here were implemented, up to 64 processors, which are across two ring:0 rings, and it is important to understand how the speed-up performance changes when the number of processors is across multiple ring:0 rings. Two test matrices used here are both symmetric and perfect banded. One has 6000 deg of freedom with 600 perfect half bandwidth. The other has 16,000 deg of freedom with 1600 perfect half bandwidth. Both matrices are stored by use of the skyline matrix storage scheme as described before. Figure 3 shows the speed-up performance for these two cases. The speed-up describes the speed advantage of the parallel algorithms compared to the central processing unit (CPU) time required in a single processor node. For the 6000 deg of freedom case, the performance drops off as the number of processors exceeds one ring:0 ring. This is primarily due to the delay in frequent references to shared data across two ring:0 rings. Also, the problem and bandwidth sizes are not big enough so that the computation can overcome the communication delay. The larger problem, 16,000 deg of freedom with 1600 perfect half bandwidth, is big enough to have a pretty good speed-up for up to 64 processors.

3.2 PIPE WHIP TEST PROBLEM

This transient dynamic analysis simulates the impact of two steel pipes.[1] The pipes have 3.3125 in. OD, 0.432 in. wall thickness, and are 50 in. long. The target pipe is supported with a fixed boundary condition at each end. The second pipe swings freely about one end with an angular velocity at impact of 50 rad/s. One-half of this symmetric problem is modeled. Shell elements are used in the model, and a slide surface (type 3) is included between the pipes. There are 1484 shell elements, 8985 deg of freedom, and 20 time steps are used with the step size of 5.0×10^{-5} s. The original serial matrix decomposition

ORNL-DWG 95-5903

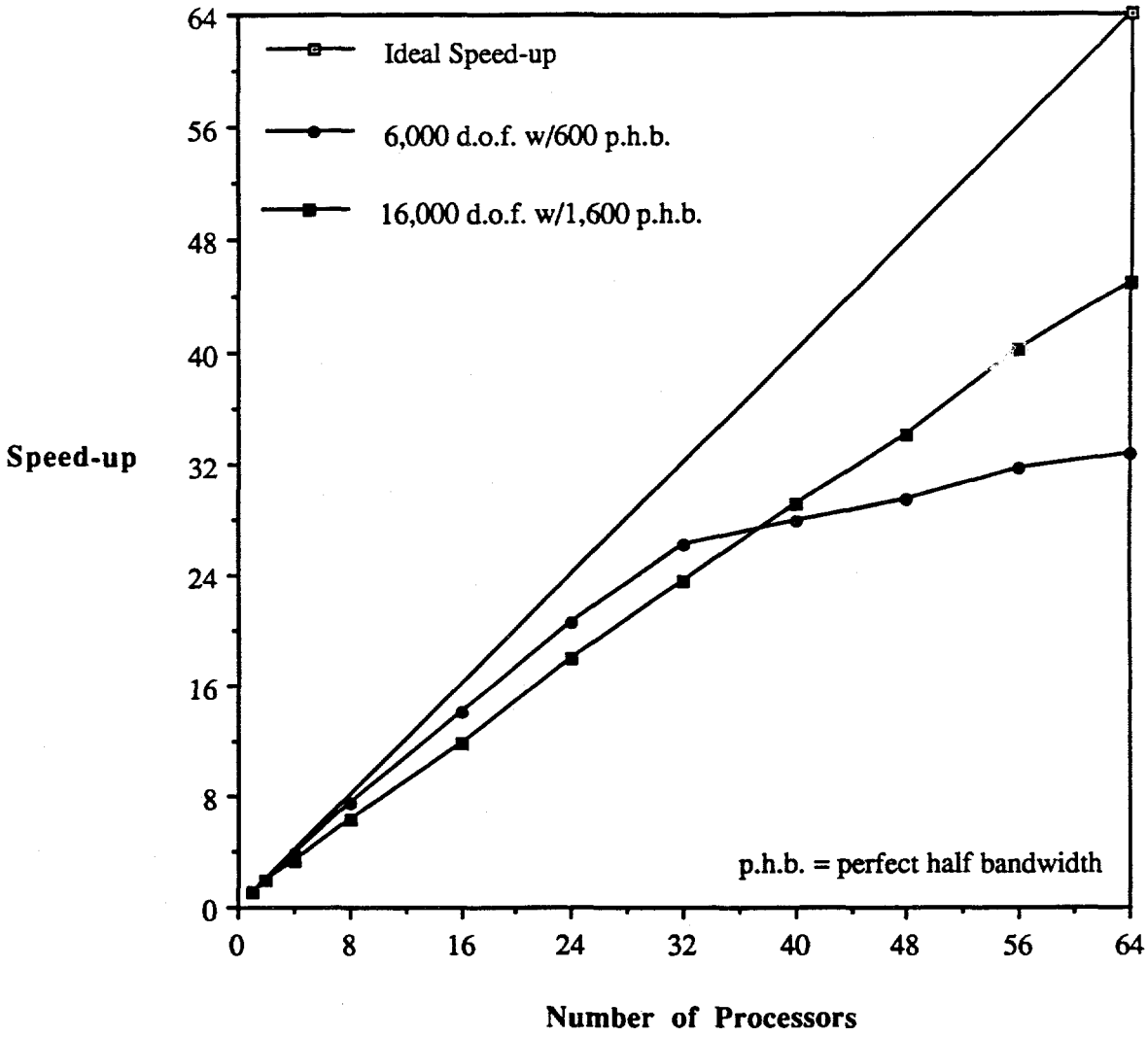

Fig. 3. The speed-up for parallel Cholesky matrix decomposition.

subroutine was substituted by the new parallel version subroutine. Also, all subroutines were recompiled with the KSR1 optimization compiler up to the first and second levels. The execution time of the optimized parallel version with one processor drops to 2950 s compared to 6709 s for the serial version without optimization compilation. Figure 4 (ref. 1) shows the deformed geometry of the pipe whip test problem at several stages of analysis. Figure 5 shows the speed-up performance for NIKE3D with the parallel Cholesky (U^TDU) matrix decomposition subroutine. The ideal speed-up curve was drawn by the fact that 56.2% of the code has been parallelized. The performance drops off as the number of processors exceeds 8 and drops off again as the number of processors exceeds 32. This is primarily due to the communication waiting of the highly skyline outline, uneven half bandwidth, and the use of two ring:0 rings.

3.3 BAR IMPACTING RIGID WALL TEST PROBLEM

This transient dynamic analysis simulates a copper bar of 6.4 mm diam and 32.4 mm length impacting a rigid wall with an initial axial velocity of 0.227 mm/μs (ref. 1). The bilinear elastic-plastic material model is used with isotropic hardening. A 90° segment of this axisymmetric cross section is modeled using 972 solid elements, 3552 deg of freedom, and 80 time steps with a step size of 1.0×10^{-6} s. Only the new parallel matrix decomposition subroutine was compiled with up to the second level optimization. The execution time for the parallelized version with one processor only reduces to 5470 s compared to 6060 s for the serial version. Figure 6 (ref. 1) shows the stages of deformation at different time periods. Figure 7 shows the speed-up performance for NIKE3D with the parallel Cholesky (U^TDU) matrix decomposition subroutine. The ideal speed-up curve was drawn by the fact that 12.6% of the code has been parallelized. The performance drops off as the number of processors exceeds 24 and drops off again as the number of processors exceeds 32. This is primarily due to the communication waiting of the skyline outline and the use of two ring:0 rings.

4. CONCLUSIONS

A parallel Cholesky (U^TDU) matrix decomposition algorithm and its application on the NIKE3D finite element code have been presented in this report. The results show that

ORNL-DWG 95-5904

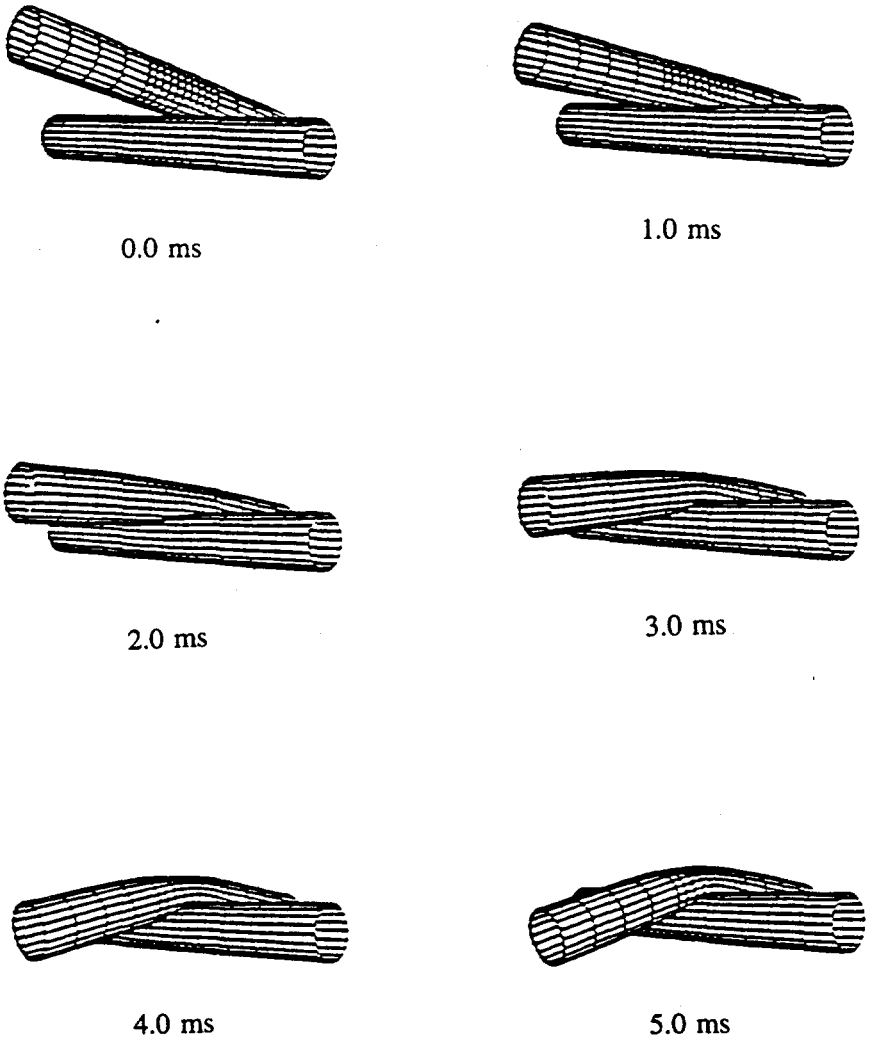

Fig. 4. The pipe whip test problem.

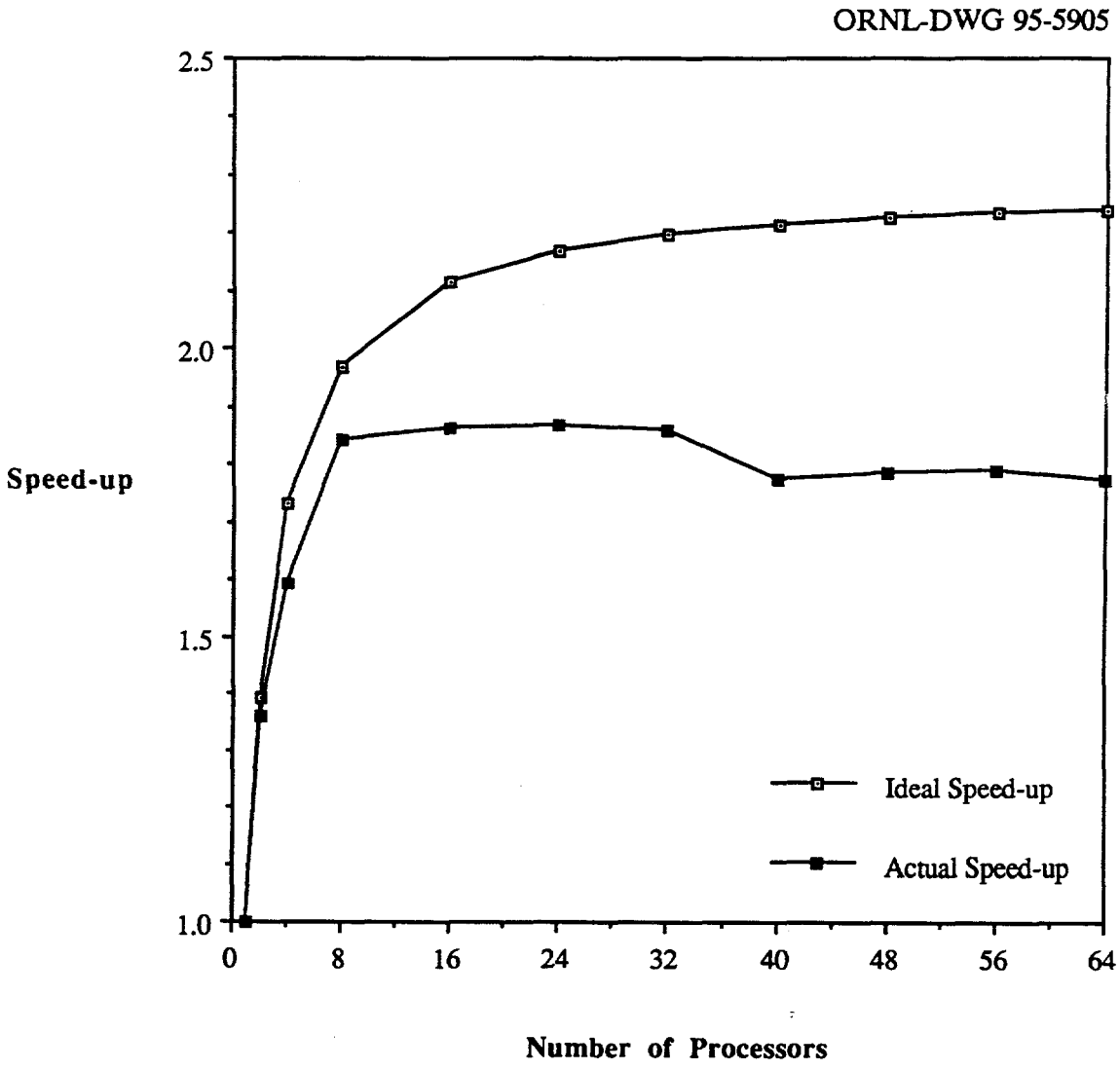

Fig. 5. Speed-up for the pipe whip test problem.

ORNL-DWG 95-5906

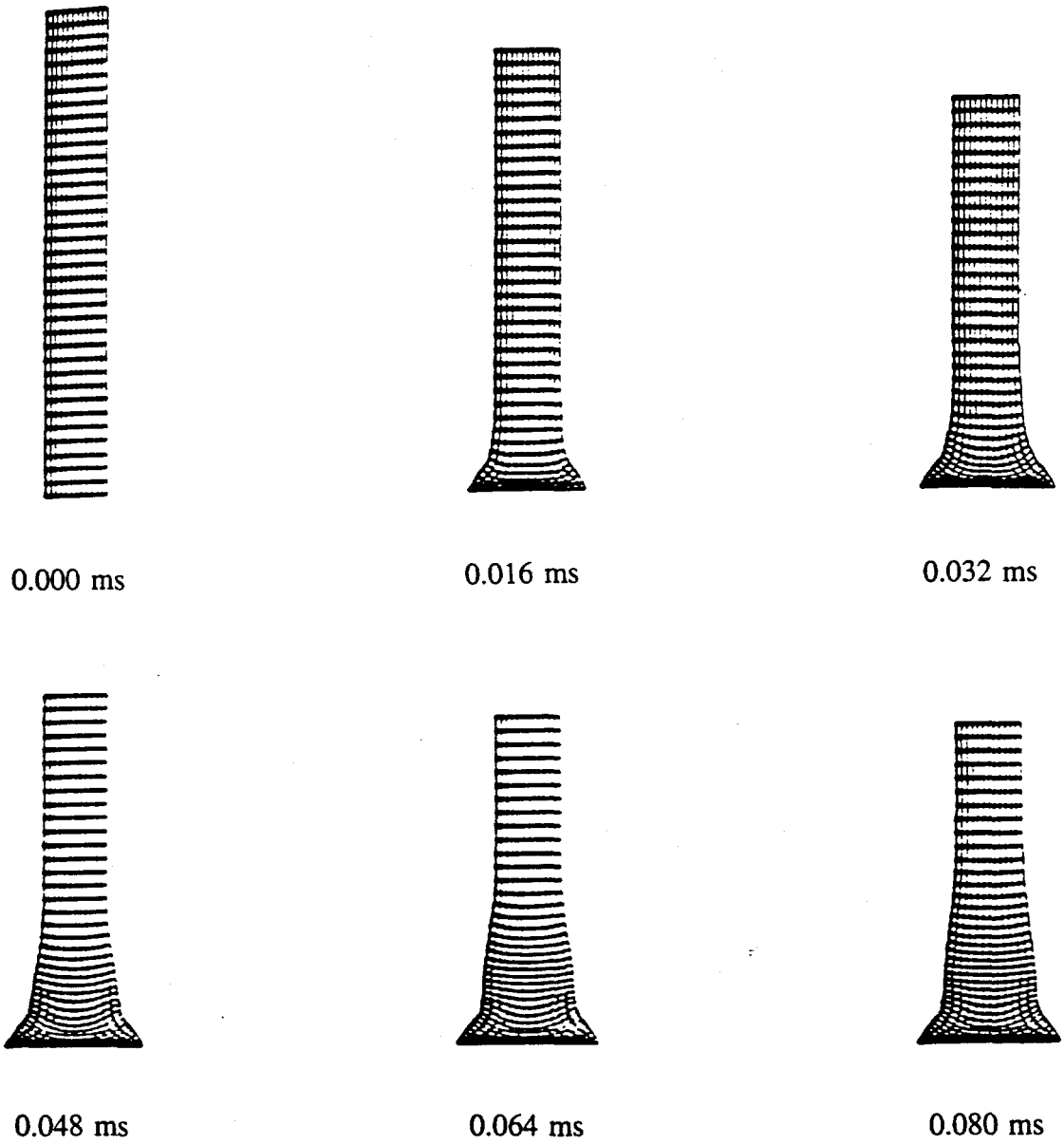

Fig. 6. The bar impacting rigid wall test problem.

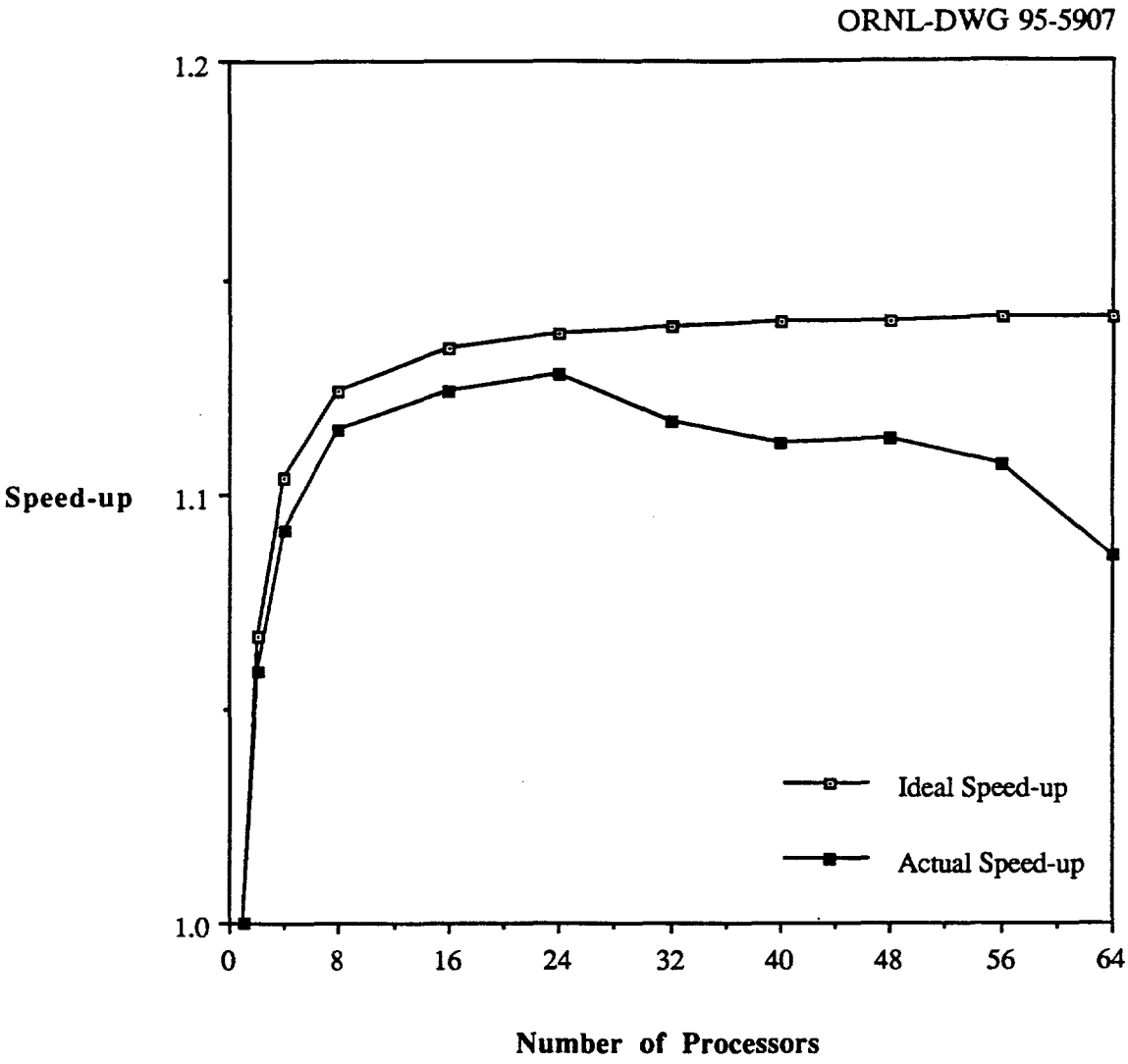

Fig. 7. Speed-up for the bar impacting rigid wall test problem.

the parallel Cholesky (U^TDU) matrix decomposition is attractive for a multiprocessor computer architecture with local- and shared-memory configurations. The implicit method-oriented finite element codes, like NIKE3D, are required to solve a very large set of equations within each time step. The parallelization of the matrix decomposition can save a lot of execution time, especially for complex problems which have huge numbers of degrees of freedom and need to use many time steps. The matrix outline, data storage scheme, and task assignment will affect the performance of the algorithm.

It is believed the results indicate that the parallel Cholesky (U^TDU) matrix decomposition algorithm can provide an attractive strategy for high-performance finite element computations on multiprocessor, local-, and shared-memory systems. Further experience with parallel sparse matrix algorithm would be desirable as well as the parallel element matrix generation, parallel global stiffness matrix assembly, and parallel force vector generation.

5. ACKNOWLEDGMENTS

The authors would like to thank Drs. Rajesh Aggarwal and Srđan Šimunović. The authors also thank Kathy Spence for editing, Glenda Carter for quality assurance review, and Donna Balltrip for final report preparation.

6. REFERENCES

1. B. N. Maker, R. M. Ferencz, and J. O. Hallquist, *NIKE3D: A Nonlinear, Implicit, Three-Dimensional Finite Element Code for Solid and Structural Mechanics - User's Manual*, Lawrence Livermore National Laboratory, Livermore, Calif., 1991.

2. *Kendall Square Research, KSR1 Principles of Operations*, KSR 8/1/91, Rev. 5.5, Kendall Square Research, Waltham, Mass., 1991.

3. D. Goehlich, "Design of Finite Element Systems for Parallel Computers," Ph.D. thesis, Georgia Institute of Technology, Atlanta, 1989.

4. K. N. Chiang, "Parallel Processing Approach for Crash Dynamic Analysis," Ph.D. thesis, Georgia Institute of Technology, Atlanta, 1989.

5. P. S. Su, "Parallel Subdomain Method for Massively Parallel Computers," Ph.D. thesis, Georgia Institute of Technology, Atlanta, 1992.

6. S. C. Chuang, "Parallel Methods for High-Performance Finite Element Methods Based on Sparsity," Ph.D. thesis, Georgia Institute of Technology, Atlanta, 1993.

INTERNAL DISTRIBUTION

1-2. Central Research Library
3. Document Reference Section
4-5. Laboratory Records Department
6. Laboratory Records, ORNL RC
7. ORNL Patent Section
8-10. M&C Records Office
11. H. W. Hayden, Jr.
12. G. M. Ludtka
13. R. E. Stoller
14-18. P. S. Su
19. S. Viswanathan
20-24. T. Zacharia
25. H. W. Foglesong (Consultant)
26. E. L. Menger (Consultant)
27. J. G. Simon (Consultant)
28. K. E. Spear (Consultant)

EXTERNAL DISTRIBUTION

29-33. GEORGIA INSTITUTE OF TECHNOLOGY, School of Mechanical Engineering, Atlanta, GA 30332-0405

R. E. Fulton

34. KENDALL SQUARE RESEARCH CORPORATION, 170 Tracer Lane, Waltham, MA 02154-1379

M. Kaufman

35. LAWRENCE LIVERMORE NATIONAL LABORATORY, Method Development Group, P.O. Box 808, L-122, Livermore, CA 94550

B. N. Maker

36. LIVERMORE SOFTWARE TECHNOLOGY CORPORATION, 2876 Waverley Way, Livermore, CA 94550

J. O. Hallquist

37. NASA LANGLEY, Research Center, Mail Stop 240, Hampton, VA 23681

O. O. Storaasli

38. UNIVERSITY OF TENNESSEE, JICS, Room 104, South College, Knoxville, TN 37996-1301

C. Halloy

39. DEPARTMENT OF ENERGY, OAK RIDGE OPERATIONS OFFICE, P.O. Box 2001, Oak Ridge, TN 37831

Assistant Manager, Energy Research and Development

40-41. DEPARTMENT OF ENERGY, OFFICE OF SCIENTIFIC AND TECHNICAL INFORMATION, P.O. Box 62, Oak Ridge, TN 37831

For distribution by microfiche as shown in DOE/OSTI-4500, Distribution Category UC-404 (Materials)