MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES
3 4456 0155148 1

ORNL/TM-10367

ornl

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

Implementing Fracture Mechanics Analysis on a Distributed-Memory Parallel Processor

J. A. Clinard
G. A. Geist

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
4500N ROOM 175
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish someone else to see this report, send in name with report and the library will arrange a loan.

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

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
NTIS price codes—Printed Copy: A02 Microfiche A01

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.

ORNL/TM-10367

Engineering Physics and Mathematics Division

Mathematical Sciences Section

IMPLEMENTING FRACTURE MECHANICS ANALYSIS ON A DISTRIBUTED-MEMORY PARALLEL PROCESSOR

J. A. Clinard *

G. A. Geist

Date Published - March 1987

* Engineering Technology Division

Research was supported by the Exploratory Studies Program of Oak Ridge National Laboratory and the Applied Mathematical Sciences Research Program of the Office of Energy Research, U. S. Department of Energy

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
operated by
Martin Marietta Energy Systems, Inc.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0155148 1

Table of Contents

Abstract v

1. Introduction 1

2. ORVIRT.PC 1

3. Finite-Element Fracture Analysis 2

4. Mapping Fracture Analysis to the Parallel Processor 2

5. The Multi-frontal Solution Technique 3

6. Example Problem 5

7. Future Directions 8

Acknowledgements 9

References 10

IMPLEMENTING FRACTURE MECHANICS ANALYSIS ON A DISTRIBUTED-MEMORY PARALLEL PROCESSOR*

J. A. Clinard

Engineering Technology Division
Oak Ridge National Laboratory

G. A. Geist

Engineering Physics and Mathematics Division
Oak Ridge National Laboratory

ABSTRACT

As part of an exploratory studies initiative at Oak Ridge National Laboratory, a parallel algorithm was developed and implemented for finite-element fracture mechanics. It was actually implemented on an Intel iPSC hypercube, although the algorithm was designed for any distributed-memory parallel computer with message passing primitives. A p-frontal method was developed for the solution of the equilibrium equations. This method required only $\log_2 p$ communication exchanges between processors during the solution. On four processors, the parallel code solved a sample problem 2.7 times faster than the serial code, ORVIRT.PC, from which it was derived.

* Research supported by the Exploratory Studies Program of Oak Ridge National Laboratory and the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems Inc.

IMPLEMENTING FRACTURE MECHANICS ANALYSIS ON A DISTRIBUTED-MEMORY PARALLEL PROCESSOR

1. INTRODUCTION

For more than a decade, ORNL research in fracture mechanics has centered around the NRC-sponsored Heavy-Section Steel Technology (HSST) program [1]. The HSST program is devoted to extending and developing technology for assessing the margin of safety against fracture of thick-walled steel pressure vessels used in light-water-cooled nuclear power reactors. The program couples materials and structures testing with analytical studies using finite-element fracture mechanics to determine the behavior and structural integrity of steel pressure vessels containing crack-like flaws. These efforts have been performed in the Engineering Technology and Metals and Ceramics Divisions of ORNL.

ORMGEN.PC/ORVIRT.PC [2,3], a fracture analysis system developed by the HSST program and designed to accommodate the limitations of microcomputers, offered an appropriate starting point for the parallel code development. The ORVIRT.PC program deals with 2-D geometries, linear elastic material behavior, and static loadings and contains a virtual crack extension technique [4,5] for evaluation of the crack-tip stress intensity parameters.

The fracture mechanics approach accepts that some flaws will be present in a structure, but assumes that conditions can be established to ensure that the flaws do not grow to an unacceptable size during the life of the structure. Life prediction in fracture mechanics requires calculation of crack-tip stress intensity parameters to quantify both stable crack growth and the conditions for unstable fracture in complex geometries under complex loading conditions.

In the linear elastic fracture mechanics (LEFM) model a cracked geometry is assumed. The structure is modeled in the usual finite-element manner except that special crack-tip elements, which allow for the proper variation in the near-tip stress and strain fields, are substituted at the crack-tip. After this pre-process modeling of the region near the crack-tip, the assembly and solution phases of the fracture mechanics analysis proceed in the same manner as standard finite-element structural analysis. The results are the nodal displacements, elemental stresses, and elemental strains in the model. The pertinent crack-tip stress intensity parameters are evaluated in a post-process step as a combination of volume integrals calculated only over the crack-tip region elements. Thus, the fracture mechanics evaluation is confined to the pre- and post-process steps, and the bulk of the computation is contained in the assembly and solution phases.

2. ORVIRT.PC

The ORVIRT.PC finite-element program was designed for fracture analysis on a microcomputer. The program uses modified versions of subroutines presented in the finite-element texts by Owen and Fawkes [7] and Hinton and Owen [8]. The frontal solver of [8] is used unmodified in ORVIRT.PC. The solver allows any number of degrees of freedom to be assigned to the nodes, and thus could accommodate applications ranging from 2-D and 3-D solid element models to plate and shell element models. ORVIRT.PC, however, is a single-element 2-D code using eight-node isoparametric quadrilateral elements exclusively (including the crack-tip region).

The eight-node element has special utility for elastic crack-tip modeling because the element has a $1/\sqrt{r}$ singularity in the stress and strain fields at the neighboring corner node [9,10] when the mid-side nodes are placed at the quarter-point positions. This allows

the appropriate LEFM solution to be found in the high-gradient crack-tip region with fewer elements than if ordinary nonsingular elements were used.

ORVIRT.PC permits linear thermoelastic stress and fracture mechanics analysis. (The material is assumed to be isotropic.) The virtual crack extension is based on a modified deLorenzi technique [4]. This technique allows thermal loadings as well as standard mechanical loadings, including crack-face pressure loadings. This crack extension technique has been shown to be accurate to within 1% for problems with closed-form solutions [3].

The virtual crack extension technique for parameter evaluation used in ORVIRT.PC can be shown [3] to reduce to the J-integral [11]. The technique offers an advantage over the J-integral because only volume integrals are evaluated over the crack region elements, as opposed to a mixture of volume and area integrals needed for the J-integral. The virtual crack extension technique (as well as the J-integral) is also applicable to nonlinear material behavior (plastic and/or viscoplastic). The application of parallel processing to nonlinear fracture mechanics problems is discussed in the final section.

3. FINITE-ELEMENT FRACTURE ANALYSIS

Over the last decade, numerical techniques such as the finite-element method have been established as powerful aids to fracture analysis. Solutions can now be obtained with confidence for complex linear and nonlinear engineering problems. However, these numerical solutions are often obtained at a considerable computing cost. When a high degree of accuracy is demanded, such as in nuclear power plant pressure-containing components and aerospace structures, simplified methods often cannot be relied upon and the engineer must resort to a detailed rigorous finite-element solution. There are several classes of problems, such as large 3-D nonlinear structural problems, that are so costly in practice that they are usually avoided.

The overall effectiveness of the finite-element structural/fracture analysis depends to a large degree on the numerical procedures used for the solution of the equilibrium equations. The accuracy of the analysis can be improved by using a more refined finite-element mesh. Therefore, the engineer tends to employ larger and larger finite-element systems to approximate the actual structure. This means the cost of the analysis and its practical feasibility depend on the efficiencies of the algorithms available for the solution of the resulting systems of equations. Because large and/or multiple systems must be solved, much research effort has gone into optimizing solution algorithms for sequential processors. (Dynamic or nonlinear structures require multiple solutions of their systems.)

Large complex structural/fracture analysis problems requiring a high degree of accuracy are the target for the current work. Mapping of key finite-element algorithms to the parallel processor and demonstration of cost savings of parallel algorithms over serial algorithms are the initial goals of this work. This work forms the foundation for additional labor required to produce a finite-element program that can routinely solve the target problems defined above.

4. MAPPING FRACTURE ANALYSIS TO THE PARALLEL PROCESSOR

The first problem to be addressed in mapping a large application code across several processors is the division of computational work. The computationally intensive routines in the application must be identified and a partitioning of the algorithm developed. In some applications this may be quite straightforward. For instance, if the computations involve the solution of many independent systems of linear equations, then it is reasonable to solve a few of these systems on each of p processors with an essentially serial code. Another consideration is whether the processors have access to shared memory. We

address the case where the processors have only local memory, such as in a hypercube multiprocessor. The restriction of using only local memory requires that the data be partitioned among the processors. Several methods of data partitioning have been investigated [12], with the most general conclusion being that efficient partitionings are usually spatial in nature.

The most computationally intensive routines in fracture analysis involve the assembly of the global stiffness matrix K and the subsequent solution of the equation $Ku=f$, where u is the deflection of the finite element mesh under loads f. For large meshes, these two steps routinely consume more than 90% of the total execution time. A natural partitioning then is to use the hypercube processors to perform these two steps and let the host processor do all the input/output. The crack-tip intensity parameters are evaluated by the host because the computation involved is small and because global information is required about the stresses around the cracks.

The second task is to decide on a method for performing the assembly and solution in parallel on the processors and from this to decide how to partition and assign the algorithm and data to each processor. The next section describes how this is implemented.

5. THE MULTI-FRONTAL SOLUTION TECHNIQUE

In the original serial code the elemental stiffness matrices are calculated and stored on secondary storage (disk). The solution of $Ku=f$ is performed by a frontal solver [6]. Thus at each step one elemental stiffness matrix is read from the disk and assembled into K. The frontal solver then performs some operations on K and writes parts of the factors back out to disk. Once the factors are determined they are read back in from the disk and used in the back substitution phase. At the end of this phase the solution vector u is known and the solver is finished. The advantage of a frontal solver is its small primary memory requirements. Only a small portion of the problem is in the main memory at any one time. The rest of the information is kept on disk. This allows computers to solve very large problems. In general, a frontal solution technique is more efficient than a band solution technique, another popular method for solving these types of problems, because it takes better advantage of the sparsity structure of the global stiffness matrix K. This often leads to a lower operation count for the solution.

There are several problems in implementing a parallel frontal solver on the hypercube processors, including the lack of access to secondary storage and the strictly sequential order that the frontal method uses to eliminate elements. One alternative is to have p fronts, all working simultaneously on different parts of the mesh, where p is the number of processors. The idea of having multiple fronts was proposed in [13] in the context of a serial algorithm and more recently in the context of a parallel algorithm for shared memory multiprocessors [14]. One advantage of having p fronts is that each processor can use the original frontal solver, avoiding the problem of programming the much more complicated multi-frontal algorithm. A second advantage of having p fronts is that it removes the necessity of following a sequential element elimination order. However, the lack of access to secondary storage remains. To work around this lack, a temporary storage vector can be built into the frontal code, allowing most of the existing software to be used. One necessary change is in the storage of all the elemental stiffness matrices. Even for small problems this requires more storage than is available on our hypercube processors. The solution is to generate these matrices one at a time as they are needed on the node, rather than storing them.

The p-front approach was designed so that each processor thinks it is solving an entire problem. In reality only the host processor knows what the whole problem looks like. The host reconstructs the mesh deflections, given the deflections computed by the individual processors. To solve its subproblem, each processor performs some

communication and some redundant calculations. The advantage of this approach is that the back substitution can be done without any communication and thus is very fast. This can be important in finite element problems where several load cases may be applied to the same mesh (such as in nonlinear problems). The communication and redundant calculations revolve around special elements, called *super boundary elements* (SBE's), created for this finite element approach.

The SBE's are formed from the shared nodes between processors. It is unlikely that they correspond to an actual element; thus their elemental stiffness matricies cannot be determined in the normal way. Instead, we use the fact that an SBE elemental stiffness matrix is some permutation of the frontal matrix on another processor. Pairs of processors assemble super boundary elements and then exchange them. This exchange is the only node-to-node communication required. At each stage there are $p/2$ exchanges, which can be performed in parallel. After each exchange, all the processors proceed normally, assembling and eliminating nodes from the super boundary element they received. This allows the solution to proceed to the end without developing special code to handle the shared nodes. In the p-front case there will be $\log_2 p$ stages during the course of the factorization since this is how many stages are required for one processor to receive information about all the other processors' fronts. One can think of it as combining all the fronts on all the processors, with nodes being eliminated during the process in order to keep the front as small as possible. This is in the same spirit as in the serial frontal method. Super boundary elements are the key to the efficient parallel implementation of the algorithm.

A difficult task in this approach is determining a good partitioning of the mesh into p blocks and reorganizing the global problem into p smaller complete problems. The original serial code (and initially the parallel code) requires the engineer to set up the data files in the way he wants the problem to be solved. The routines read the reorganizing information from a data file set up by the engineer. Automation of this reordering is under development and will be described in detail in a future report. The objective can be simply stated: the original problem must be reorganized into p tasks such that both the communication and the the operation count are small, while maximizing the parallelism among the p processors.

The amount of communication is reduced in two ways. First, only mesh information is passed between the host and the nodes instead of the entire stiffness matrix K. Second, the number of shared nodes between processors at each of the $\log_2 p$ stages must be minimized. The number of shared nodes is a property of how the original problem is partitioned and the order in which the exchanges are performed.

The operation count is a function of the front width and thus the diameter of each of the subproblems, which is dictated by the partitioning of the elements. Since K is sparse, the operation count can often be reduced by reordering the elimination of the element nodes. Initially, we have used a profile minimization ordering because it is the best ordering for a frontal solver. Recently there has been evidence that a minimum degree ordering [15] coupled with a multi-frontal solver can dramatically reduce the operation count and thus the execution time of finite element analysis codes. We have begun investigating the possibility of using a multi-frontal solver on each of the p problems. The main drawback is that these codes are complex and could easily double the size of the entire program.

The final constraint of maximizing the parallelism among the p tasks is accomplished by partitioning the original problem into p blocks of elements such that the number of operations in each block is approximately the same. An automatic way of partitioning a problem into p balanced pieces is under investigation and preliminary results indicate that

this can be accomplished in $O(p)$ time. The constraint on having only p blocks of elements is an attempt to keep communication costs reasonable. The next section will illustrate this reorganization on a small example problem and present results from runs on an Intel iPSC hypercube.

6. EXAMPLE PROBLEM

Figure 1 shows the configuration of the original problem, which arose from studies of crack formation in neutron hardened boiler plates exposed to thermal shocks. The problem contains 28 eight-node isoparametric elements, for a total of 101 nodes. Because the problem is small, we decided to implement it on only 5 processors (four hypercube processors and the host processor). This gives fewer than 10 elements in each of the four subproblems. If more processors were used, the subproblems would become trivial, but the communication would grow to be a dominant portion of the execution time. For large enough problems, many processors could be involved in the computation, but this will require automation of the problem reorganization.

In this small example we performed the reorganization manually as follows. First, the mesh was divided symmetrically down through the virtual crack. (Since the host will reconstruct the problem, it is not necessary that the crack actually exist in one of the subproblems). Second, the 14 elements in each half were separated into two blocks such that 6 elements were assigned to one processor and 8 elements were assigned to another processor. (It was noticed that assigning 7 elements to each processor balanced the computational load better but increased the volume of comunication by almost 30%.) This gave the partitioning of elements seen in figure 2, which minimizes the maximum front generated in any of the subproblems.

The next step in the reorganization is to determine the SBE's and the order in which they are exchanged. For each exchange, each processor is paired with a *partner* with whom it shares at least one node. The SBE of each processor is defined to be the set of all of its partner's shared nodes. The shape and size of the SBE's are affected by the order in which the partners are chosen, and these parameters are directly related to the communication volume and computation to communication ratio. The problem of choosing an exchange order is greatly simplified in this example since we are only using four processors and thus have only two SBE's per processor. In general, the criterion for choosing partners is that the number of shared nodes that can be eliminated due to all the exchanges should be maximized. This involves choosing partners according to how many common nodes they have in their respective fronts at this stage of the factorization. Once the exchange is performed, all these shared nodes will be eliminated simultaneously by the two processors. This procedure is repeated for each of the $\log_2 p$ steps. Note that even if we use 64 processors, there will be only 6 of these exchange steps regardless of the complexity of the finite element problem. Thus the search for good partners is often a small part of the reorganization algorithm.

The two SBE's of processor 1 are illustrated in figure 2. Processor 1's first partner is P3, and its second partner is P2. The SBE's resemble mesh separators as described in George and Liu [15]. In fact they are related, but because of required communication, the best choice for a separator may not be the best choice for an SBE.

An outline of an algorithm for determining SBE's and partners can be described as follows:

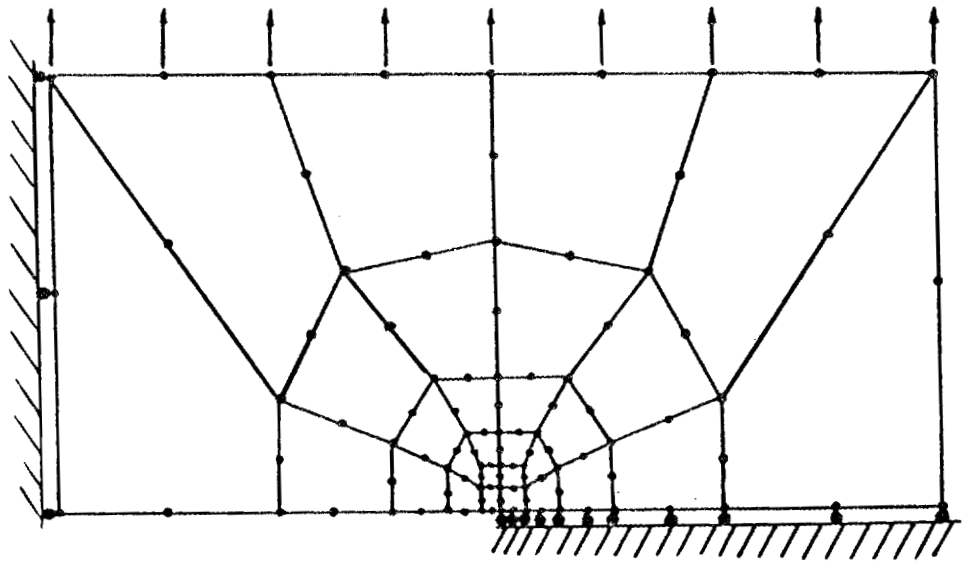

Figure 1. Finite Element Mesh of Simple Fracture Analysis Problem

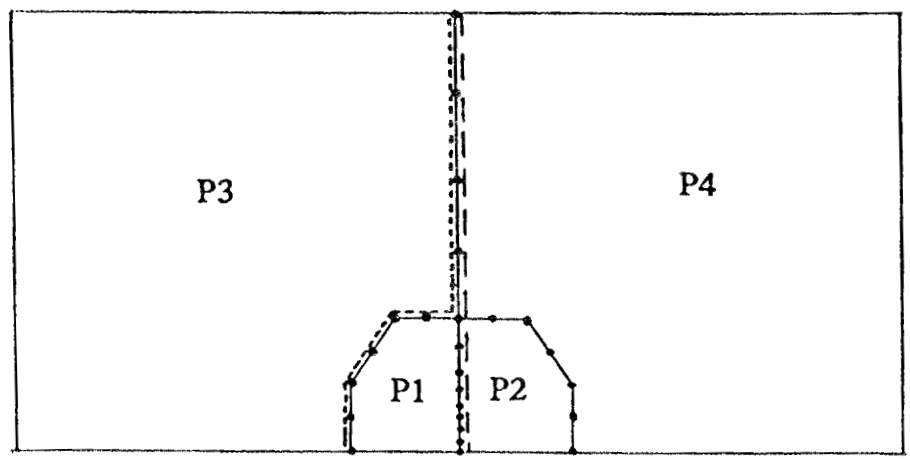

Figure 2. Partitioning of Sample Problem into 4 Blocks.
The two Super Boundary Elements of Processor 1
are highlighted. SBE 1 = --------
SBE 2 = — — —

```
Determine an initial mesh partition into p blocks
For log2 p steps
    loop over all processors
        determine their shared nodes and with whom
    choose partners to maximize eliminations
    loop over all processors
        assign SBE as the shared nodes of your partner
        mark all nodes in common in exchange as eliminated
End for
```

Once all the SBE's are determined, the p subproblems are complete and the ordering of the elements (or nodes) on each of these tasks must be done. Since this implementation uses the original frontal solver, the elements of each of these problems were ordered using a frontwidth reduction algorithm.

In summary there are three major stages to the reorganization of the original problem. First, the mesh must be partitioned into p blocks such that the size of the block separators is minimized and each of the blocks contains approximately the same amount of work. Second, the super-boundary-elements and partners must be determined for each of the subproblems so that the amount of computation between exchanges is balanced across the processors. Third, the individual subproblems must be numbered (ordered) to minimize the number of operations on each of them. Once the reorganization is complete, the host processor sends one subproblem to each hypercube processor.

Table 1. Performance Results for Sample Problem

Code	assemble / factor	solve	total time*
Serial on host	32.2	0.88	55 (sec)
Parallel w/4 nodes	7.6	0.27	24 (sec)

*Total time includes I/O, Host to Node communication, and fracture analysis.

The results from running the sample problem on four processors are encouraging. As seen in Table 1, the overall execution time for the problem has been decreased by more than 58% compared to the serial code, even though the parallel time includes such purely serial phases as host I/O, host-to-node communication, and the fracture analysis. The results from the two parts of the code that were parallelized, the assembly/factorization of K and the solution of $Ku=f$, are somewhat deceiving. Notice that the parallel assembly/factorization step performs 4.2 times faster than the serial step. This is because the multifrontal ordering has a lower operation count than the frontal ordering so the parallel code has an advantage beyond having more processors. A similar but less dramatic improvement can be seen in the solution time for the same reason. Also the assembly/factorization time is reduced by 24.6 seconds and the solution time is reduced by 0.6 seconds, whereas the reduction in total time is 31 seconds. The extra time savings occurs because the host processor performs useful work while the nodes are computing.

Although this test problem is quite small, when automated partitioning allows the solution of much larger problems the results are expected to improve even further, since

the number of nodes per processor grows much faster than the number of shared nodes as the size of the problem increases. There is some indication from our work on automatic partitioning that large problems are required to justify the use of many processors. Or conversely, using too many processors to solve a problem will slow down the solution process. For example, we have seen meshes with 2500 nodes that could be solved faster on 4 processors than on 16 processors.

7. FUTURE DIRECTIONS

There is a current need for finite-element fracture analysis of nonlinear problems. The mapping of the p-frontal solution technique to the parallel processor provides interesting and unusual opportunities for streamlining the solution to some nonlinear problem classes. Specifically, we will look at the class represented by the equilibrium equations

$$K\dot{u} = \dot{f} ,$$

where K is a function of u, and $\dot{u}$ and $\dot{f}$ are either increments of displacements and loads or rates of displacements and loads. The nonlinearity, $K = K(u)$, is composed of material and/or geometric nonlinearity.

The nonlinearity contributed by the material is due to the yielding phenomenon, the post-yield flow, and the hardening behavior of the ductile metal. Such metal exhibits elastic response as well as some form of history-dependent inelastic behavior. Classical plasticity theories, which consider elastic-plastic and elastic-viscoplastic response, provide useful models for the ductile metal. Material nonlinearity may be restricted to the vicinity of the crack-tip, producing a mildly nonlinear problem. In any case, the stress-strain relationships of all of the yielded finite elements must be altered to accommodate the appropriate plasticity of the material. Also, the singularity at the crack front approaches $1/r$ (instead of $1/\sqrt{r}$ for the elastic material), which can be easily achieved by modeling the tip by collapsed elements. Collapsed elements have corner nodes and mid-point nodes which initially share the same location. These elements deform with increasing load to allow for blunting of the crack.

Geometric nonlinearity is concerned with the effects of large geometry changes in the vicinity of the crack tip and consists almost entirely of large rotations that occur in the region. Although the approaches of the various available large deflection techniques differ, they basically involve considering the nonlinear strain-displacement relations and the changing geometry of the structure.

The incorporation of the nonlinear effects into the finite-element analysis is accounted for most often by incremental solution techniques that linearize the nonlinear problem for a sequence of loading steps. The dominant numerical techniques are based on Newton's method with various modifications depending on the severity of the nonlinearities. For simple contained plasticity problems, the techniques developed fall into two categories: the initial strain method and the tangent modulus method. They differ computationally depending on whether the effects of nonlinearities enter as pseudo-loads (initial strain), or the stiffness matrix is explicitly altered (tangent modulus). In either case, the static problem has no imposed time scale, and the only criterion involved in selection of the time step is accuracy in modeling nonlinear effects.

In solving the nonlinear problem on the parallel processor with the p-front approach, there is an advantage to avoiding reformulation of the stiffness matrix K because the back substitution can be done without any node-to-node communication. Thus, the initial strain solution technique is attractive because it allows nonlinear problem solutions without reformulation and refactorization of the stiffness matrix. Of course,

reformulation of $\dot{f}$ for each step of the solution cannot be avoided. While this potentially produces a requirement for node-to-node communication, if mapping of the problem onto the processors can place all nonlinearities within the boundaries of a single super-element, then the communication may be avoided. If the nonlinearities cannot be contained so neatly, then the necessary node-to-node communication can be accomplished through a simple extension of our algorithm.

Accurate solution by the initial strain approach requires smaller solution steps than in the tangent stiffness method. Even so, drifting of the solution can occur. This drifting can be monitored through calculation of an equilibrium imbalance vector. If the solution begins to drift, then the stiffness matrix needs to be reformulated and refactored along the steps of the solution process. Such a hybrid approach is mandatory in the case of geometric nonlinearity. All problems in this class require frequent updating of the stiffness matrix because of the need to update the geometry of the deformed body.

We have addressed only a subclass of nonlinear fracture problems that may be labeled as mildly nonlinear. It is not clear that methods for treating severely nonlinear cases (where initial strain methods tend to break down) will allow all the advantages of our present parallel approach.

ACKNOWLEDGEMENTS

We thank John Bryson, the author of ORVIRT.PC, for allowing us to use this code in our development effort.

References

1. Pugh, C. E., *Heavy-Section Steel Program Semiannual Progress Report for October 1985 - March 1986*, NUREG/CR-4219, Vol. 3, No. 1 (ORNL/TM-9593/V3&N1), March 1986.
2. Bryson, J. W. and Bass, B. R., *ORMGEN.PC: A Microcomputer Program for Automatic Mesh Generation of 2-D Crack Geometries*, NUREG/CR-4475, (ORNL-6250), in preparation.
3. Bryson, J. W., *ORVIRT.PC: A 2-D Finite Element Fracture Analysis Program for a Microcomputer*, NUREG/CR-4367 (ORNL-6208), Oct. 1985.
4. deLorenzi, H. G., "On the Energy Release Rate and the J-Integral for 3-D Crack Configuration," *International Journal of Fracture*, **19**, 1982, pp. 183-193.
5. Bass, B. R. and Bryson, J. W.,"Energy Release Rate Techniques for Combined Thermo-Mechanical Loading," *International Journal of Fracture*, **22**, 1983, R3-R7.
6. Irons, B. M., "A Frontal Solution Program," *International Journal of Numerical Methods in Engineering*, **2**, 1970, pp. 5-32.
7. Owen, D. R. J. and Fawkes, A. J., *Engineering Fracture Mechanics Numerical Methods and Applications*, Pinerridge Press, Swansea, U.K., 1983.
8. Hinton, E. and Owen, D. R. J., *Finite Element Programming*, Academic Press, New York, 1977.
9. Henshell, R. D. and Shaw, K. G.,"Crack-tip Elements are Unnecessary," *International Journal for Numerical Methods in Engineering*, **9**, 1975, pp. 495-507.
10. Barsoum, R. S., "On the Use of Isoparametric Finite Elements in Linear Fracture Mechanics," *International Journal for Numerical Methods in Engineering*, **10**, 1976, pp. 25-27.
11. Rice, J. R., "A Path Independent Integral and the Approximate Analysis of Strain Concentrations by Notches and Cracks," *ASME Journal of Applied Mechanics*, **35**, 1968, pp. 379-386.
12. Heath, M. T.,"Hypercube Applications at Oak Ridge National Laboratory," Hypercube Multiprocessors 1987, SIAM, Philadelphia, 1987.
13. Duff, I. S. and Reid, J. K., "The Multifrontal Solution of Independent Sparse Symmetric Linear Systems," *ACM Transactions on Math. Software*, **9**, 1983, pp. 302-325.
14. Duff, I. S., "Parallel implementation of multifrontal schemes," *Parallel Computing*, **3**, 1986, pp. 193-204.
15. George, A. and Liu, J. W. H., *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

INTERNAL DISTRIBUTION

1-5. J. A. Clinard
6-10. J. M. Corum
11-15. G. A. Geist
16-17. R. F Harbison
18-22. J. K. Ingersoll
23-27. F. C. Maienschein
28-32. H. E. Trammell
33-37. R. C. Ward
38. A. Zucker
39. P. W. Dickson (Consultant)
40. G. H. Golub (Consultant)
41. R. M. Haralick (Consultant)
42. D. Steiner (Consultant)
43. Central Research Library
44. K-25 Plant Library
45. ORNL Patent Office
46. Y-12 Technical Library /Document Reference Station
47. Laboratory Records - RC
48-49. Laboratory Records Department

EXTERNAL DISTRIBUTION

50. Dr. Donald M. Austin, Office of Scientific Computing, Office of Energy Research, ER-7, Germantown Building, U.S. Department of Energy, Washington, DC 20545
51. Lawrence J. Baker, Exxon Production Research Company, P.O.Box 2189, Houston, TX 77252-2189
52. Dr. Jesse L. Barlow, Department of Computer Science, Pennsylvania State University, University Park, PA 16802
53. Prof. Ake Bjorck, Department of Mathematics, Linkoping University, Linkoping 58183, Sweden
54. Dr. Bill L. Buzbee, C-3, Applications Support & Research, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545
55. Dr. Donald A. Calahan, Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109
56. Dr. Tony Chan, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
57. Dr. Jagdish Chandra, Army Research Office, P.O. Box 12211, Research Triangle Park, North Carolina 27709
58. Dr. Paul Concus, Mathematics and Computing, Lawrence Berkeley Laboratory, Berkeley, CA 94720
59. Dr. Jane K. Cullum, IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598
60. Dr. George Cybenko, Department of Computer Science, Tufts University, Medford, MA 02155
61. Dr. George J. Davis, Department of Mathematics, Georgia State University, Atlanta, GA 30303
62. Dr. Jack J. Dongarra, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

63. Dr. Stanley Eisenstat, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
64. Dr. Howard C. Elman, Computer Science Department, University of Maryland, College Park, MD 20742
65. Dr. Albert M. Erisman, Boeing Computer Services, 565 Andover Park West, Tukwila, WA 98188
66. Dr. Geoffrey C. Fox, Booth Computing Center 158-79, California Institute of Technology, Pasadena, CA 91125
67. Dr. Paul O. Frederickson, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545
68. Dr. Robert E. Funderlic, Department of Computer Science, North Carolina State University, Raleigh, NC 27650
69. Dr. Dennis B. Gannon, Computer Science Department, Indiana University, Bloomington, IN 47405
70. Dr. David M. Gay, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974
71. Dr. C. William Gear, Computer Science Department, University of Illinois, Urbana, Illinois 61801
72. Dr. Don E. Heller, Physics and Computer Science Department, Shell Development Co., P.O. Box 481, Houston, TX 77001
73. Dr. Robert E. Huddleston, Computation Department, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550
74. Dr. Ilse Ipsen, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
75. Dr. Harry Jordan, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309
76. Dr. Linda Kaufman, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974
77. Dr. Robert J. Kee, Applied Mathematics Division 8331, Sandia National Laboratories, Livermore, CA 94550
78. Dr. Richard Lau, Office of Naval Research, 1030 E. Green Street, Pasadena, CA 91101
79. Dr. Alan J. Laub, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106
80. Dr. Robert L. Launer, Army Research Office, P.O. Box 12211, Research Triangle Park, North Carolina 27709
81. Prof. Peter D. Lax, Director, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012
82. Dr. Michael R. Leuze, Computer Science Department, Box 1679 Station B, Vanderbilt University, Nashville, TN 37235
83. Dr. Joseph Liu, Department of Computer Science, York University, 4700 Keele Street, Downsview, Ontario, Canada M3J 1P3
84. Dr. Franklin Luk, Electrical Engineering Department, Cornell University, Ithaca, NY 14853
85. James G. Malone General Motors Research Laboratories, Warren, Michigan 48090-9055

86. Dr. Thomas A. Manteuffel, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545
87. Dr. Paul C. Messina, Applied Mathematics Division, Argonne National Laboratory, Argonne, IL 60439
88. Dr. Cleve Moler, Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006
89. Dr. Dianne P. O'Leary, Computer Science Department, University of Maryland, College Park, MD 20742
90. Maj. C. E. Oliver, Office of the Chief Scientist, Air Force Weapons Laboratory, Kirtland Air Force Base, Albuquerque, NM 87115
91. Dr. James M. Ortega, Department of Applied Mathematics, University of Virginia, Charlottesville, VA 22903
92. Prof. Chris Paige, Basser Department of Computer Science, Madsen Building F09, University of Sydney, N.S.W., Sydney, Australia 2006
93. Dr. John F. Palmer, NCUBE Corporation, 915 E. LaVieve Lane, Tempe, AZ 85284
94. Prof. Beresford N. Parlett, Department of Mathematics, University of California, Berkeley, CA 94720
95. Prof. Merrell Patrick, Department of Computer Science, Duke University, Durham, NC 27706
96. Dr. Robert J. Plemmons, Departments of Mathematics and Computer Science, North Carolina State University, Raleigh, NC 27650
97. Dr. John K. Reid, CSS Division, Building 8.9, AERE Harwell, Didcot, Oxon, England OX11 0RA
98. Dr. John R. Rice, Computer Science Department, Purdue University, West Lafayette, IN 47907
99. Dr. Garry Rodrigue, Numerical Mathematics Group, Lawrence Livermore Laboratory, Livermore, CA 94550
100. Dr. Donald J. Rose, Department of Computer Science, Duke University, Durham, NC 27706
101. Dr. Ahmed H. Sameh, Computer Science Department, University of Illinois, Urbana, IL 61801
102. Dr. Michael Saunders, Systems Optimization Laboratory, Operations Research Department, Stanford University, Stanford, CA 94305
103. Dr. Robert Schreiber, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180
104. Dr. Martin H. Schultz, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
105. Dr. David S. Scott, Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006
106. Dr. Lawrence F. Shampine, Mathematics Department Southern Methodist University Dallas, Texas 75275
107. Dr. Danny C. Sorensen, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
108. Prof. G. W. Stewart, Computer Science Department, University of Maryland, College Park, MD 20742

109. Capt. John P. Thomas, Air Force Office of Scientific Research, Building 410, Bolling Air Force Base, Washington, DC 20332

110. Prof. Charles Van Loan, Department of Computer Science, Cornell University, Ithaca, NY 14853

111. Dr. Robert G. Voigt, ICASE, MS 132-C, NASA Langley Research Center, Hampton, VA 23665

112. Dr. Andrew B. White, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545

113. Mr. Patrick H. Worley, Computer Science Department, Stanford University, Stanford, CA 94305

114. Dr. Arthur Wouk, Army Research Office, P.O. Box 12211, Research Triangle Park, North Carolina 27709

115. Dr. Margaret Wright, Systems Optimization Laboratory, Operations Research Department, Stanford University, Stanford, CA 94305

116. Office of Assistant Manager for Energy Research and Development, Department of Energy, Oak Ridge Operations Office, Oak Ridge, TN 37830

117-146. Technical Information Center