22nd IBERIAN LATIN-AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING
2nd Brazilian Congress on Computational Mechanics
NOVEMBER 7-9, 2001
Campinas, SP - Brazil

AN INTEGRATED PARALLEL SYSTEM FOR PROPAGATION OF ARBITRARY CRACKS IN SOLID MODELS

Célio O. Moretti
Túlio N. Bittencourt
Computational Mechanics Laboratory, Department of Structural and Foundation Engineering, Polytechnical School, University of São Paulo
Av. Prof. Almeida Prado, travessa 2, no. 83 - CEP 05508-900 - São Paulo – Brazil
Luiz F. Martha
Department of Civil Engineering and Technology Group on Computer Graphics - Tecgraf
Pontifical Catholic University of Rio de Janeiro - PUC-Rio
Rua Marquês de São Vicente, 225 - CEP 22453-900 - Rio de Janeiro – Brazil

Abstract. *To perform crack propagation in a solid model, a high computational power is required, mainly at three-dimensional mesh generation and structural analysis steps. At each crack propagation step, the mesh is rebuilt and a new structural analysis is performed. If a large scale cracked model is being analyzed, time consumed by mesh generation and analysis may be extremely large or even prohibitive in some cases. The main idea of the methodology presented in this work is to parallelize mesh generation and structural analysis procedures, and to integrate these procedures into a computational environment able to perform automatic arbitrary crack propagation. A parallel mesh generation algorithm has been developed. This algorithm is capable of generating three-dimensional meshes of tetrahedral elements in arbitrary domains with one or multiple embedded cracks. A finite element method program called FEMOOP, based on object oriented programming, has been adapted to implement the parallel features. The parallel strategy to solve the set of linear equations is based on an element-by-element scheme in conjunction with a gradient iterative solution. A program called FRANC3D, which is completely integrated with other components of the system, performs crack propagation and geometry updates. The entire system is described in this work and an application example is presented to show the performance and reliability of the crack propagation process.*

Keywords: *Parallel computing, Crack propagation, Finite Element Method, Mesh generation*

1. INTRODUCTION

Crack propagation simulation is an important topic in many fields, e.g., aeronautical engineering, material sciences, and geophysics. This type of simulation requires a high computational power, mainly at three-dimensional mesh generation and structural analysis steps. These steps usually spend a large amount of computing time. The main objective of this work is to provide a fast and accurate system for crack growth simulation in three-dimensional models. To do this, a parallel 3-D mesh generator and a parallel FEM analysis have developed and integrated with a program called FRANC3D (Martha,1989) (Carter,2000), responsible for crack propagation and geometry updates, establishing a parallel system to perform automatic crack growth simulation.

In the following sections, all components of the parallel system are presented, with special emphasis to parallel mesh generator and parallel FEM analysis components. A three steps crack propagation in a concrete dam model is presented as an example to demonstrate the performance and reliability of the parallel system.

2. THE PARALLEL SYSTEM

The parallel analysis system is comprised by a set of integrated programs, each of one responsible for a specific task: pre-processing, mesh generation, structural analysis and crack propagation. All components of this system are presented in the following sections.

2.1 Pre-processing

A program named FRANC3D (3D Fracture Analysis Code) (Martha,1989) (Carter,2000) is used in the pre-processing step. FRANC3D is a system that exploits graphical and processing resources of high performance workstations to perform the modeling and visualization of three-dimensional solids with arbitrary cracks. With an intuitive graphical interface (Figure 1), this program allows the modeling of complex 3D solids with a variety of crack shapes.

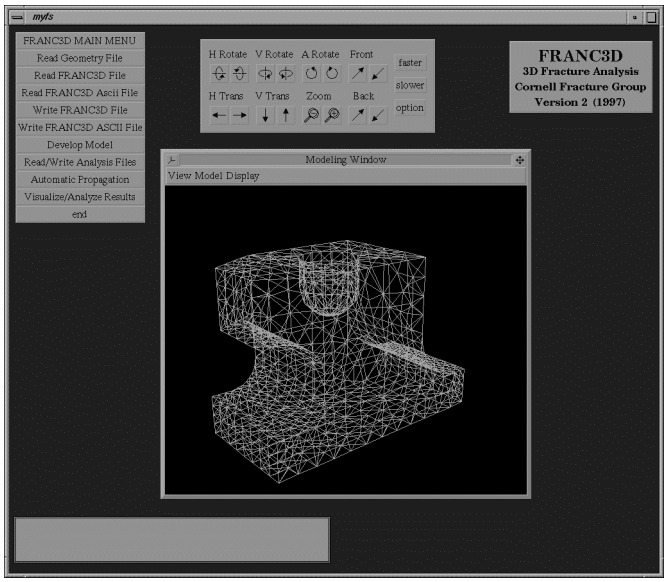

Figure 1 - FRANC3D graphical interface.

2.2 Parallel mesh generation

A sequential volumetric mesh generation algorithm has already developed and implemented by Cavalcante Neto, et al. (Cavalcante,2001). This algorithm is capable of generating three-dimensional meshes of tetrahedral elements in arbitrary domains with one or multiple cracks. The algorithm combines an advancing front technique with a recursive spatial decomposition technique, in this case an octree, to define the internal nodes, element sizes, and mesh transition. Various complex models have been meshed using this algorithm with good results (Cavalcante,2001).

The basic idea to parallelize this volumetric mesh generation algorithm is to divide the original domain into subdomains that could be meshed independently, without message passing among processors. The parallel algorithm uses a master/slave programming model: the master program is responsible for domain partitioning and final mesh assembly; and the slave programs are responsible for mesh generation in each subdomain. Briefly, the desirable features of the present parallel mesh generation algorithm are:

1. To generate subdomains from a given triangular surface mesh;
2. To generate interfaces between subdomains using triangular elements with sizes compatible to the sizes of given (input) surface mesh;
3. To generate consistent solid elements at the neighborhood of adjacent subdomains;
4. To generate subdomains with approximate mesh generation computation time.

In the following subsections, all steps of the mesh generation process are described.

2.2.1 Input data

A triangular surface mesh is the input data. In this work, this surface mesh is generated by FRANC3D, but any given triangular surface mesh can be used. There is a large number of triangular surface mesh generation algorithms (Lau,1996)(Lewis,1996) that can be used to generate this initial mesh.

2.2.2 Background mesh generation

In this work, an initial coarse solid mesh is generated (background mesh) from a given triangular surface mesh, using the existing sequential mesh generation algorithm. This background mesh is partitioned to generate subdomains. A desirable feature of the background mesh generation is to have a reduced processing time in comparison with complete mesh generation time.

The number of internal nodes created in the process of mesh generation is defined by the octree density. As a reduced processing time is wanted, a small octree density value, in comparison with a regular density value, is used. Using this small density value, a few internal nodes are created in the process of background mesh generation, resulting in a coarse mesh generated in a reduced time.

2.2.3 Background mesh partitioning

Subdomains are created from background mesh partitioning. To perform this partitioning, Metis library (Karypis,1997) is used. Metis library is a set of programs developed to perform

graph and mesh partitioning. To partition a finite element mesh, initially a corresponding graph is defined and is divided using Metis algorithms. These algorithms are based in a multilevel scheme for partitioning irregular graphs (Karypis,1998).

The background mesh partitioning is performed assigning elements to different subdomains. In this step, the original domain is divided into subdomains with approximately the same number of elements and with minimal neighboring interfaces. A desirable condition, but not necessary, is that created subdomains are continuous. Existence of discontinuous subdomains does not prevent mesh generation but can depreciate the performance of the parallel mesh generation algorithm.

2.2.4 Initial interface smoothing

In this step, an initial smoothing is applied to interfaces between subdomains. The implemented algorithm covers all elements at the boundary of a subdomain and verifies, for each element, whether the number of neighbor elements of an adjacent subdomain is greater than the number of neighbor elements of the same subdomain of the element. If that occurs, the element is transferred to the adjacent subdomain. The main goal is to smooth the interface between subdomains with a very low computational cost.

2.2.5 Interface refinement

Background mesh partitioning creates subdomains normally comprised by the original (input) triangular surface mesh and by the surface meshes that belong to interfaces between subdomains. The latter surface meshes result from the initially generated coarse mesh. For this reason, the size of their triangles is not compatible with size of the original surface triangles. To guarantee quality of the final solid mesh generated after assembling all subdomain meshes, an interface refinement is required.

A technique based on octree partitioning is used to guide the interface refinement. In this case, an octree with a density normally used to generate not coarse meshes is applied. The refinement procedure can be summarized as: for each interface triangle, if its area is greater than the corresponding octree cell face area, divide this triangle and the adjacent triangles creating a node in the middle of longest edge that does not belong to original surface mesh; repeat this process until all interface triangles are refined.

After applying this procedure, some triangles can present a sliver shape, i.e., with one edge much greater than other two edges. To prevent this, all triangles are traversed again and, if a sliver triangle is found, it is subdivided creating a node in the middle of longest edge (if possible). If not possible, i.e., the longest edge belongs to original (input) triangular surface mesh, the triangle remains unchanged.

2.2.6 Interface smoothing

After the interface refinement step, some triangles might not present a good shape. This condition could depreciate quality of final generated mesh. To avoid this, a Constrained Laplacian Smoothing Algorithm (Cannan,1998) is applied. This algorithm, in its simplest form, moves each node to the average coordinates of adjacent nodes. This technique works well in convex regions. To avoid distorted or inverted elements near concavities, constraints are applied to node movements.

2.2.7 Parallel mesh generation

After the interface refinement and smoothing phases, resulting subdomains are sent to processors. Each processor applies the sequential mesh generation algorithm (Cavalcante,2001) to the corresponding subdomain. Therefore, subdomain mesh is generated concurrently and independently, without message passing among processors.

2.2.8 Final mesh assembly

In this final step, the master program receives subdomain meshes and assemblies the complete solid mesh.

2.3 Parallel FEM Analysis

In the system presented here, the structural analysis is performed by a finite element program called FEMOOP (Finite Element Method - Object Oriented Programming) (Martha,1996), which is organized using object-oriented concepts (Fujii,1997) (Guimarães,1992). One of the most important advantages of the object-oriented programming is the code extensibility. This feature allows new implementations with minimum impact over the existent code. Another important feature of the program code is its portability, which allows an easy code adaptation to different platforms. These two features have allowed the adaptation of the original sequential code to a parallel environment and to different computational platforms. The parallel environment considered in this work is a distributed memory environment, which can be comprised by a local area network, a parallel computer or a multiprocessor machine. In a previous work (Moretti,1998), parallel analyses running in a local area network was presented. In this case, the local area network can be viewed as a virtual parallel machine with multiple processors and distributed memory. In the present work, the analyses were performed in a multiprocessor machine.

To adapt FEMOOP to the parallel computational environment, a new class has been created, which is responsible for data manipulation. Also, a series of new functions have been implemented into existent classes. The first step necessary to adapt FEMOOP to the parallel environment was the implementation of a library responsible for the message passing management. The main objective of this library is to the direct access message passing functions. This access facilitates an eventual change of the message passing manager or the addition of a new one. A change or addition of a message passing manager has impact only over the library code. This parallel procedure library contains all functions necessary to perform the message passing in a distributed memory environment. The main functions implemented here are responsible for sending and receiving messages among processors, for parallel process initialization, and for the identification of program type (either a master or a task program).

The parallel programming paradigm adopted here has been the master-slave model. In this model, the master is a separate program responsible for process spawning, initialization, reception and display of results, and timing of functions. The task (or slave) programs are executed concurrently and interact through message passing. Each task program is responsible for assembling finite element stiffness matrices and right-hand side forces of a subdomain. Through interactions between the master program and the task programs, the global solution is obtained.

In the following subsection, a parallel technique used to solve the linear system of equations

$$Ku = f \ , \tag{1}$$

that arises in the finite element method, is presented.

2.3.1 Element-by-element

The global stiffness matrix and the right-hand side vector that arise in a finite element method can be written as

$$K = \sum_{e=1}^{N} \hat{K}^e \ , \ \ f = \sum_{e=1}^{N} \hat{f}^e \ , \tag{2}$$

where $\hat{K}^e$ and $\hat{f}^e$ are the contribution from the finite element e and N is the total number of elements. $\hat{K}^e$ and $\hat{f}^e$ represent the local element contribution (K^e and f^e, respectively) expanded to the global structure size, i.e., the entries of K^e are mapped into corresponding global row and column locations with others entries in $\hat{K}^e$ equal to zero. Then Eq. (1) can be written as

$$\left(\sum_{e=1}^{N} \hat{K}^e \right) u = \left(\sum_{e=1}^{N} \hat{f}^e \right). \tag{3}$$

$\hat{K}^e$ is a very sparse matrix, but only the dense local element contribution K^e needs to be stored. K^e and f^e can be calculated concurrently and independently for elements e=1,2,…,N.

In this case, a PCG method has also been used to solve the linear system. The main calculations in this method are a matrix-vector product and vector dot products that are repeated at each iteration. These operations can be performed concurrently taking advantage of K^e and f^e characteristics. A global matrix-vector product of the form $Kv = w$ can be written as

$$Kv = \left(\sum_{e=1}^{N} \hat{K}^e \right) v = \left(\sum_{e=1}^{N} \hat{K}^e \hat{v}^e \right) = \sum_{e=1}^{N} \hat{w}^e = w \, . \tag{4}$$

Again, $\hat{v}^e$ and $\hat{w}^e$ represent the local element contribution expanded to system size and to perform the product only the non-zero terms are considered

$$w^e = A^e v^e \tag{5}$$

and each element matrix-vector product can be computed independently. The dot products are performed in a similar way, considering only the non-zero entries of the local vectors.

When the local calculations are completed, the processors send the local results of the matrix-product or dot product to the master program, which assembles the global result and

send it back to the processors. This process is repeated at each PCG iteration until the solution is obtained. Note that this scheme does not require any special mesh partitioning or element ordering.

2.4 Crack propagation

In this work, FRANC3D is used in the crack propagation step. FRANC3D is designed for modeling arbitrary crack growth in three-dimensional solid and shell structures. This program has capabilities for modeling multiple, non-planar, arbitrary shaped cracks. From FEM numerical results, stress intensity factors along the crack fronts are determined. Currently, crack fronts are propagated by assuming that plane strain equations are valid at discrete points along the crack front.

FRANC3D was completely integrated with the parallel mesh generator and parallel FEM analysis program, described above. Automatic propagation of an arbitrary 3-D crack in an arbitrary 3-D structure is possible through the use of the implemented programs.

3. EXAMPLE

In this section, a model of a concrete dam with a external half-penny shape crack is presented. Figure 2 shows the model and crack details. Three steps of crack growth are simulated.

A Dell PowerEdge computer with four Pentium III Xeon 500 MHz processors is used as computing environment, running Windows 2000 and Cygwin tools. MPI/Pro version 1.6.3 provides message passing for parallel computing.

Figure 3 shows detailed processing time for each crack propagation step: time consumed for parallel mesh generation and parallel FEM analysis, and total time consumed for each step, using 1 to 4 processors. Figure 4 shows total time consumed for simulate three steps of crack growth, using again 1 to 4 processors. Figure 5 presents the speed-up obtained for mesh generation, FEM analysis and complete simulation. Finally, Figure 6 shows final crack shape after three propagation steps.

Figures 3 and 4 show that the parallel system has a good scalability, reducing simulation time with increment of number of processors used.

Figure 5 shows that mesh generation has a very good speed-up, but speed-up of complete simulation is decreased by FEM analysis performance. This analysis performance can be explained by the fact that the number of generated elements increases with the number of processors used, increasing the number of degrees of freedom of the model. It has been shown in a previous work (Moretti,2000), that, if the analysis is performed for a model with a fixed number of elements, the present FEM parallel analysis program provides an excellent speed-up result. As a future work, it is necessary to reduce the number of generated elements when the number of subdomains is increased. This can be done modifying the interface refinement between subdomains.

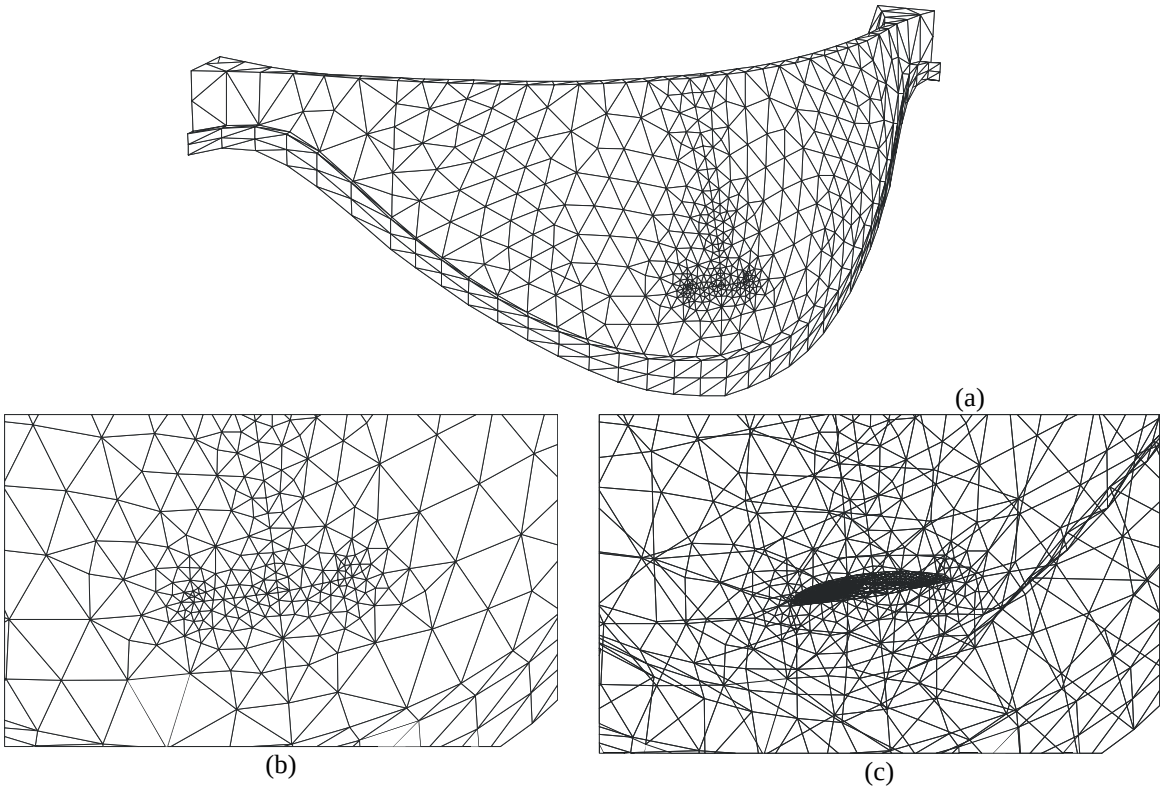

Figure 2 - Concrete dam and crack details: (a) entire model; (b) surface mesh of crack region; (c) mesh of initial crack.

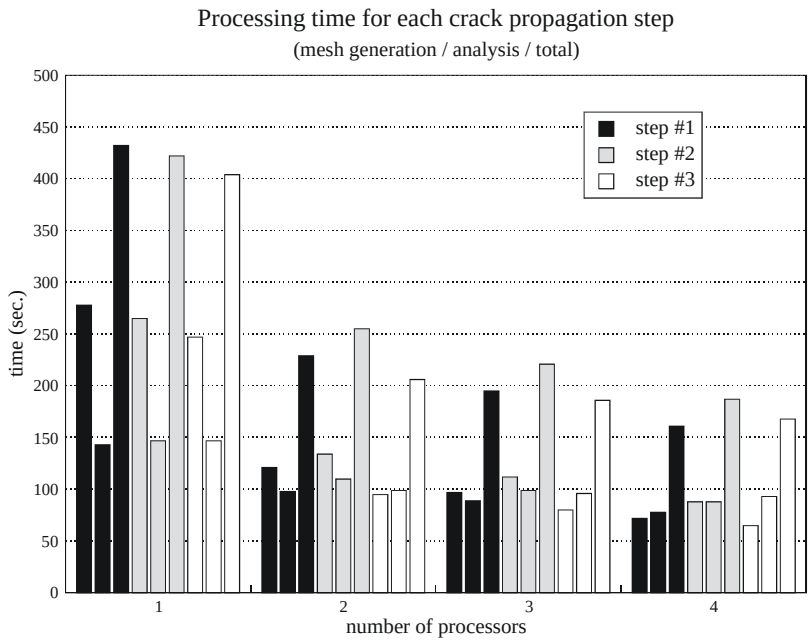

Figure 3 - Detailed processing time using 1 to 4 processors: mesh generation, analysis and total time consumed for each crack propagation step.

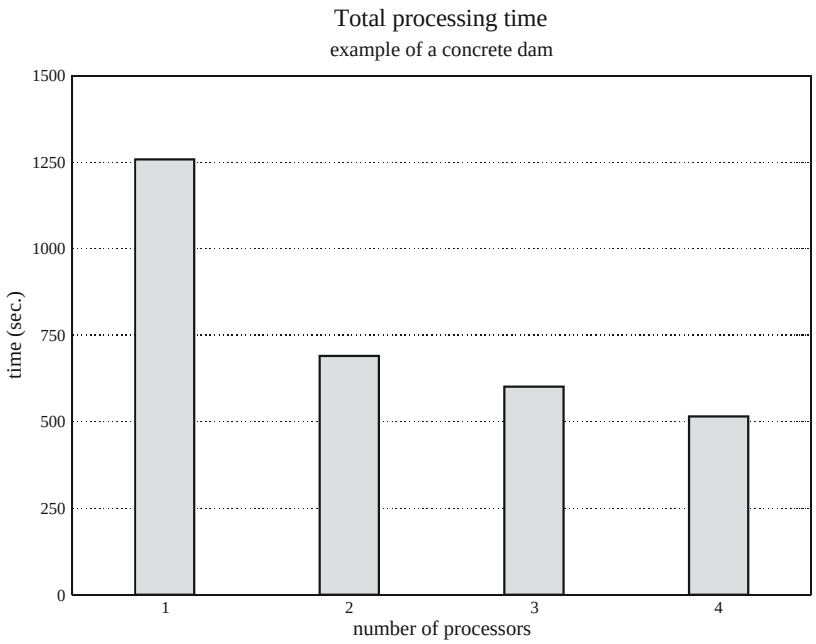

Figure 4 - Total processing time.

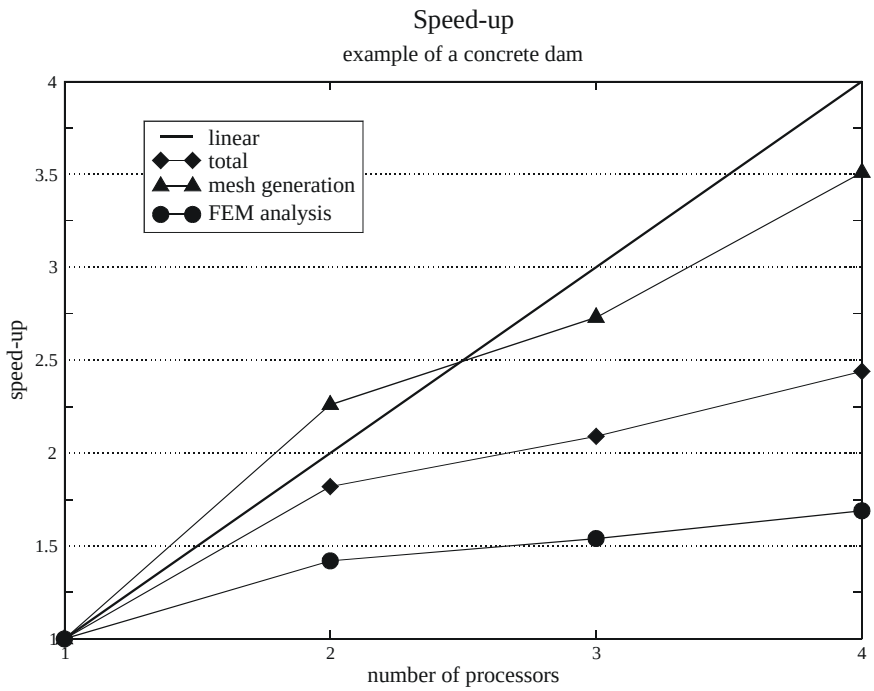

Figure 5 - Speed-up of mesh generation, FEM analysis and entire simulation. The linear speed-up represents a system with ideal scalability.

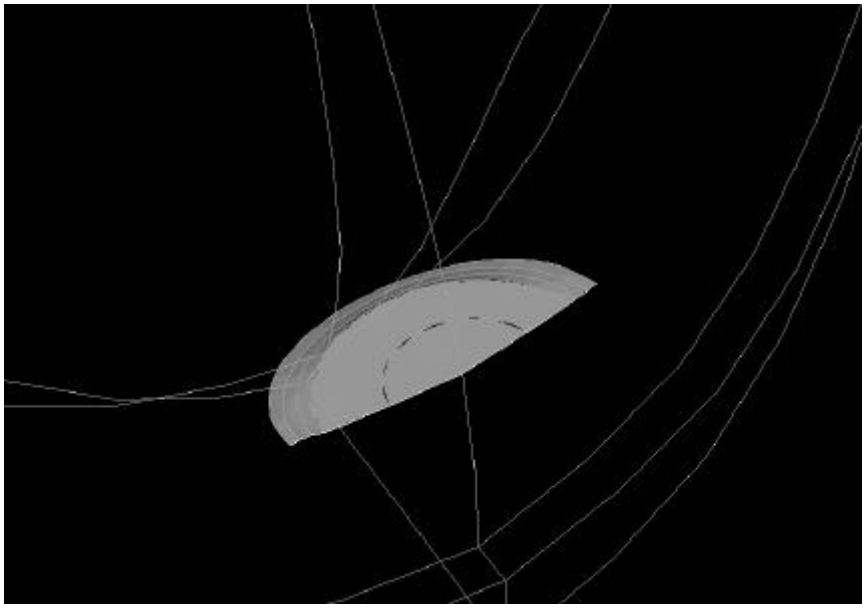

Figure 6 - Crack propagation after three steps.

4. CONCLUSIONS

In this work, a parallel system for automatic 3-D crack growth simulation was presented. The main objectives of this system are to improve speed of mesh generation and FEM analysis, allowing simulation of large scale cracked models. These objectives were attained through development of a parallel volumetric mesh generator and a parallel FEM analysis program, and integration of these programs with FRANC3D, creating a system capable to perform automatic crack growth simulation in a fast and accurate way. A system with these features can be a valuable tool for engineers and scientists to simulate realistic 3-D fracture processes.

REFERENCES

Cannan, S.A., Tristano, J.R., Staten, M.L., "An Approach to Combined Laplacian and Optimazation-based Smoothing for Triangular, Quadrilateral, and Quad-dominant Meshes", Technical Report, ANSYS, Inc., 1998.

Carter, B.J., Wawrzynek, P.A., Ingraffea, A.R., "An Automatic 3-D Crack Growth Simulation", International Journal for Numerical Methods in Engineering, v. 47, pp. 229-253, 2000.

Cavalcante Neto, J.B., Wawrzynek, P.A., Carvalho, M.T.M., Martha, L.F., Ingraffea, A.R., "An Algorithm for Three-Dimensional Mesh Generation for Arbitrary Regions with Cracks", Engineering with Computers, v. 17, no. 1, pp. 75-91, 2001.

Fujii, G., Análise de Estruturas Tridimensionais: Desenvolvimento de uma Ferramenta Computacional Orientada para Objetos, Dissertação de Mestrado, Dep. de Engenharia de Estruturas e Fundações (PEF), Escola Politécnica, USP, 1997.

Guimarães, L.G.S., Menezes, I.F.M., Martha, L.F.: Object Oriented Programming Discipline for Finite Element Analysis Systems (in Portuguese), Proceedings of XIII CILAMCE, Porto Alegre, RS, Brasil, Vol. 1, pp. 342-351, 1992.

Karypis, G., Kumar, V., "Metis – a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices – version 3.0.3", Technical report, University of Minnesota, 1997.

Karypis, G., Kumar, V., “A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs”, Technical Report 95-035, University of Minnesota, 1998.

Lau, T.S., Lo, S.H.: Finite Element Mesh Generation Over Analytical Curved Surfaces, Computers and Structures, vol. 59, pp. 301-309, 1996.

Lewis, R.W., Zheng, Y., Gethin, D.T.: Three-dimensional unstructured mesh generation: Part 2, Surface Meshes, Comput. Math. Appli. Mech., vol. 134, pp. 269-284, 1996.

Martha, L.F., “Topological and Geometrical Modeling Approach to Numerical Discretization and Arbitrary Crack Propagation in Three-Dimensions”, PhD Dissertation, School of Civil Engineering, Cornell University, 1989.

Martha, L.F., Menezes, I.F.M., Lages, E.N., Parente Jr., E., Pitangueira, R.L.S.: An OOP Class Organization for Materially Nonlinear Finite Element Analysis, Joint Conference of Italian Group of Computational Mechanics and Ibero-Latin American Association of Computational Methods in Engineering, Padova, Italy, Sep. 1996, pp. 229-232, 1996.

Moretti, C.O., Bittencourt, T.N., Martha, L.F.: A Low Cost Distributed System for FEM Parallel Structural Analysis, VECPAR'98 - 3rd International Meeting on Vector and Parallel Processing, Porto, Portugal, June 21-23, pgs 1063-1075, 1998.

Moretti, C.O.; Cavalcante Neto, J.B.; Bittencourt, T.N.; Martha, L.F., “A Parallel Environment for Three-Dimensional Finite Element Method Analysis”, Developments in Engineering Computational Technology, B.H.V. Topping (Editor), Civil-Comp Press, Edinburgh, UK, ISBN 0-948749-70-9, pp. 283-287, 2000.