DYNELF

A computer program for DYNamic three-dimensional ELastic deformation with Frictional sliding on faults,

under development by Dudley Joe Andrews of USGS.

The Fortran source code is not generally available. It has been given to a few colleagues under the following conditions.

1. You may not give a copy of the program source code to anyone else. All requests for a copy must be directed to Dudley Joe Andrews.
2. The program is under continuing development. No standard version is available for distribution. Particular coding is required to set up each particular problem. Andrews will not make later versions compatible with your modifications.
3. Any publication using calculations done with Dynelf must acknowledge Dudley Joe Andrews, but you will *not* list him as a co-author.

In addition to these conditions, the USGS requires the following disclaimer statement:

"Although this program has been used by the USGS, no warranty, expressed or implied, is made by the USGS or the United States Government as to the accuracy and functioning of the program and related program material nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the USGS in connection therewith."

BACKGROUND DISCUSSION OF DYNELF

The 3D dynamic code Dynelf may be called either a finite difference or a finite element code. The finite differences in space in Dynelf are formulated to be equivalent to finite elements, although someone familiar with finite element codes would not readily recognize them as being so.

In order to understand Dynelf, I suggest that you start with my 2D finite difference code Scoot, and read the appendices in Andrews (1973) and Andrews and Ben-Zion (1997). Scoot uses triangular simplex finite elements. Velocity and displacement are defined at the nodes of the mesh. In a simplex element (which is a triangle in 2D) strain and stress are uniform in the interior of each element. Although Scoot uses finite elements in space, I call it a finite difference code for three reasons.

(1) I do not formulate a stiffness matrix and do not invert a matrix equation to find a static solution. The solution is stepped through time with explicit equations.

(2) A uniform mesh is used with indices simply related to Cartesian co-ordinates. A finite element code with a non-uniform mesh is more complicated because of indirect addressing and the required construction of a pointer array associating element vertices with node points.

(3) Because simplex elements are used, I can regard velocity and stress as the fundamental variables in Scoot. Displacement and force are the fundamental variables in a finite element method. Using velocity and stress as fundamental variables, moment tensor sources and plastic yielding can be handled simply by adjusting stress. Equivalent body force must be calculated and saved if displacement and force are the fundamental variables.

In 1996 when I started developing the 3D code Dynelf, I had a number of design decisions to make.

Coding is very simple for a staggered grid scheme in which different components of velocity and different components of stress are defined at different points in space. It is easy to increase the order of accuracy to fourth order in space, while keeping it second order in time. There is no doubt that the fourth-order staggered-grid method is the most efficient and accurate method to propagate a wave to a distance that is a large number of wavelengths. Accurate boundary conditions are not simple, however, with a fourth-order staggered grid. (Or one could say that simple implementations of boundary conditions are not accurate.) To calculate earthquake sources, the fault boundary condition is more important than propagation to large distances, so I decided not to use a staggered-grid scheme.

A straightforward generalization of my 2D code to 3D would be to use 3D simplex elements, which are tetrahedra. I seriously considered this option. Brad Aagaard at Caltech has written such a code. Such a code is still a very attractive option in my mind. With this option the fault boundary condition could be the traction-at-split-node (TSN) method (Andrews, 1999), which I am now using in Dynelf. I did not choose the option of tetrahedral elements simply because I wanted a simple relationship between node indices and Cartesian co-ordinates.

I chose to use quadrilateral isoparametric elements (Hughes, 1987, Chapter 3). Stress is not uniform inside an element, and the fundamental variables are displacement and force at nodes. A few years earlier, about 1993, I had worked with Jay Melosh's finite element code TECTON. It is very similar to the code in the appendix of Hughes' book. I modified it to step through time explicitly. I also modified the calculation in the interior of elements. Instead of using a local stiffness matrix, I did the calculation in several distinct steps. From displacement at nodes, find strain at the integration points, then find stress, then find force at the nodes. I found that this formulation, mathematically equivalent to the local stiffness matrix, required many fewer arithmetic operations. I adopted this formulation in subroutine box in Dynelf. I gained more efficiency by moving the integration points to the nodes of the element and by specializing the element to be a rectangular box. The equations in subroutine box are equivalent to and 100 times more efficient that the general isoparametric finite element equations. My objective in designing Dynelf was to have all components of velocity, displacement, and force defined at the same node points and to achieve an efficiency comparable to a finite difference method. The equations in subroutine box can be recognized as finite difference equations. The number of finite difference expressions that are calculated is 8 times larger than in a simple staggered grid scheme.

The book by Hughes will not be helpful in understanding my code, unless you want to check my claim that subroutine box is equivalent to an isoparametric bilinear element. To get a triangular element, Hughes prefers to collapse nodes of a quadrilateral, which is not the same as a simplex element. If you want to read about simplex elements, find a different textbook.

In order to calculate a dipping fault, I split rectangular box elements into triangular wedge elements. I wrote the equations in subroutine wedge by analogy with subroutine box using intuition about the finite difference expressions. I guess that a 2D triangular face of the wedge is a simplex element and that a 2D rectangular face is a bilinear element. I did not derive the equations rigorously from those assumptions. I checked subroutine wedge by comparing a test calculation done on a dipping fault with the same test on a vertical fault.

REFERENCES

Andrews, D. J., Tectonic stress release by underground explosions, *Bull. Seism. Soc. Am.*, **63**, 1375-1391, 1973.

Andrews, D. J. and Y. Ben-Zion, Wrinkle-like slip pulse on a fault between different materials, *J. Geophys. Res.*, **102**, 553-571, 1997.

Andrews, D. J., Test of two methods for faulting in finite difference calculations, *Bull. Seism. Soc. Am.*, **89**, 931-937, 1999.

Hughes, T. J. R., *The Finite Element Method*, Prentice-Hall Inc., 1987.