308 --- Q200311260001

Scientific Notebook No. 188: Continued from Scientific Notebook No. 183: Finite Element Simulation of Slip on Bare Mountain Fault--Effects on Seismicity Assessment for the Proposed Yucca Mountain Repository Sites (08/19/1996 through 07/15/1999)

Good luck OFOEGBU
CNWRA

The Boorum & Pease® Quality Guarantee

The materials and craftsmanship that went into this product are of the finest quality. The pages are thread sewn, meaning they're bound to stay bound. The inks are moisture resistant and will not smear. And the uniform quality of the paper assures consistent rulings, excellent writing surface and erasability. If, at any time during normal use, this product does not perform to your expectations, we will replace it free of charge. Simply write to us:

Boorum & Pease Company
71 Clinton Road, Garden City, NY 11530
Attn: Marketing Services

Any correspondence should include the code number printed at the bottom of this page as well as the book title stamped at the bottom of the spine.

One Good Book Deserves Many Others.

Look for the complete line of Boorum & Pease® Columnar, Journal, and Record books. Custom-designed books also available by special order. For more information about our Customized Book Program, contact your office products dealer. See back cover for other books in this series.

Made in U.S.A.

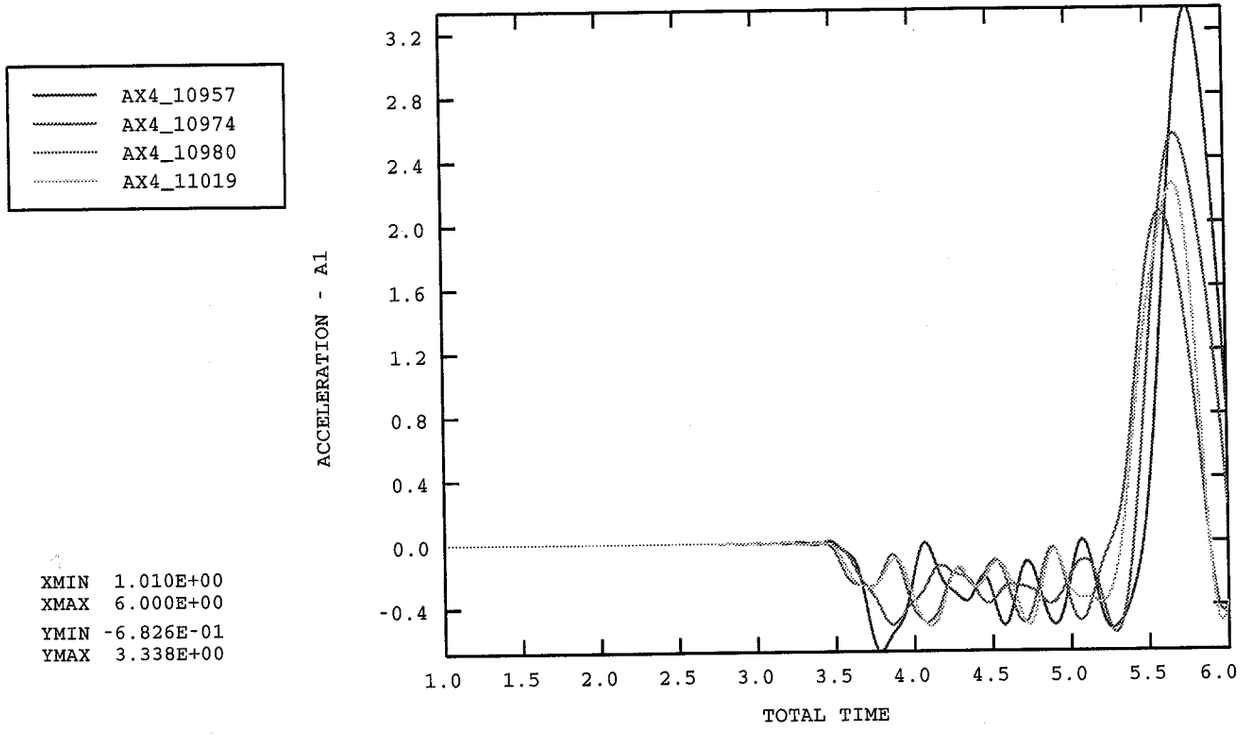
(3)

Contents

Page

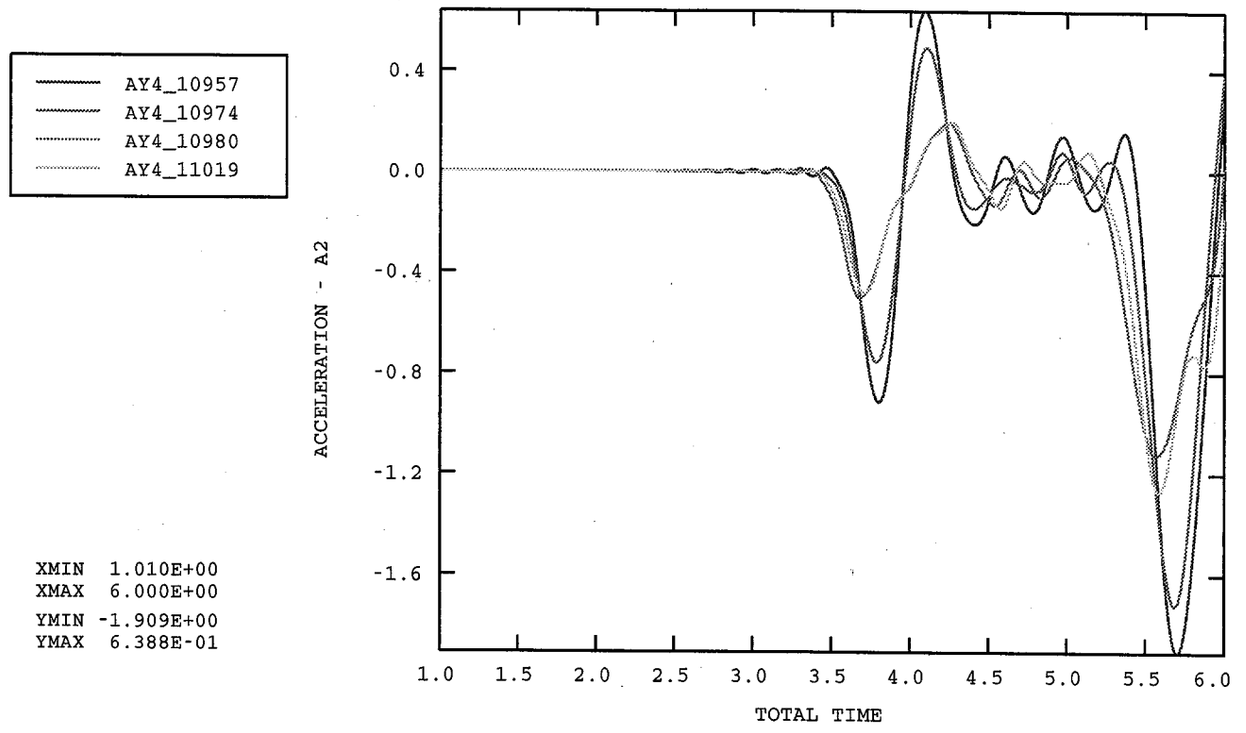
Continued from Notebook #183 GW

ABAQUS



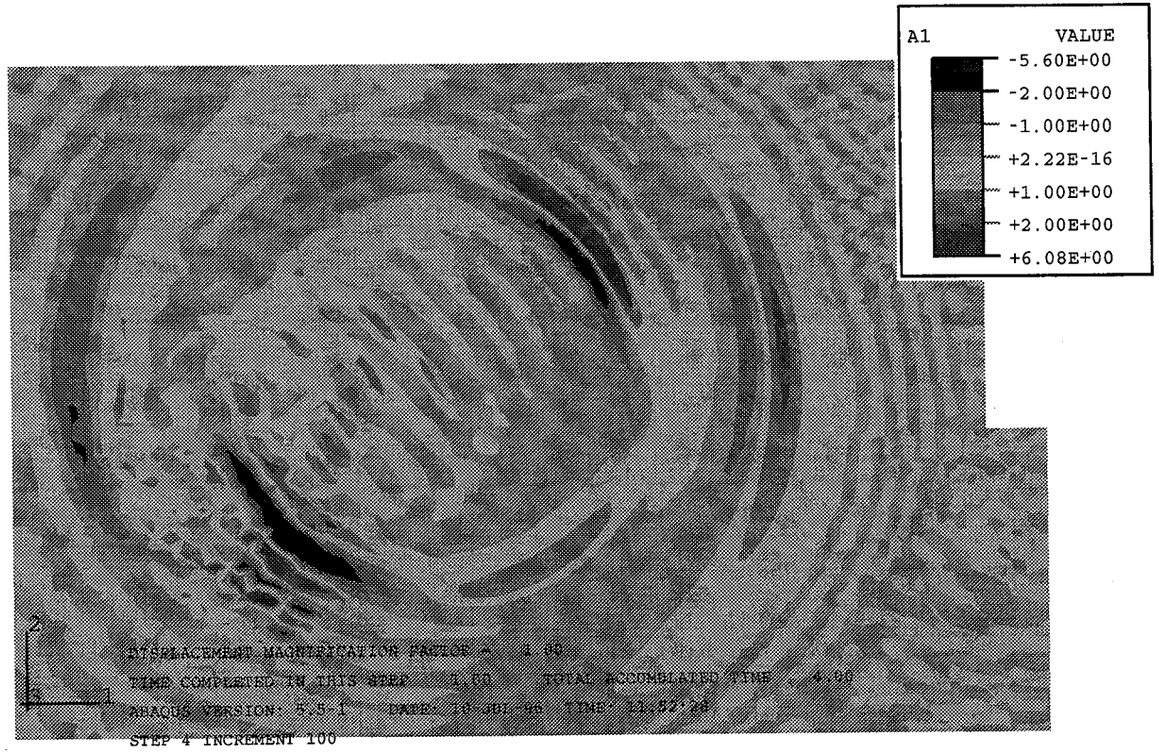
XMIN 1.010E+00
XMAX 6.000E+00
YMIN -6.826E-01
YMAX 3.338E+00

ABAQUS

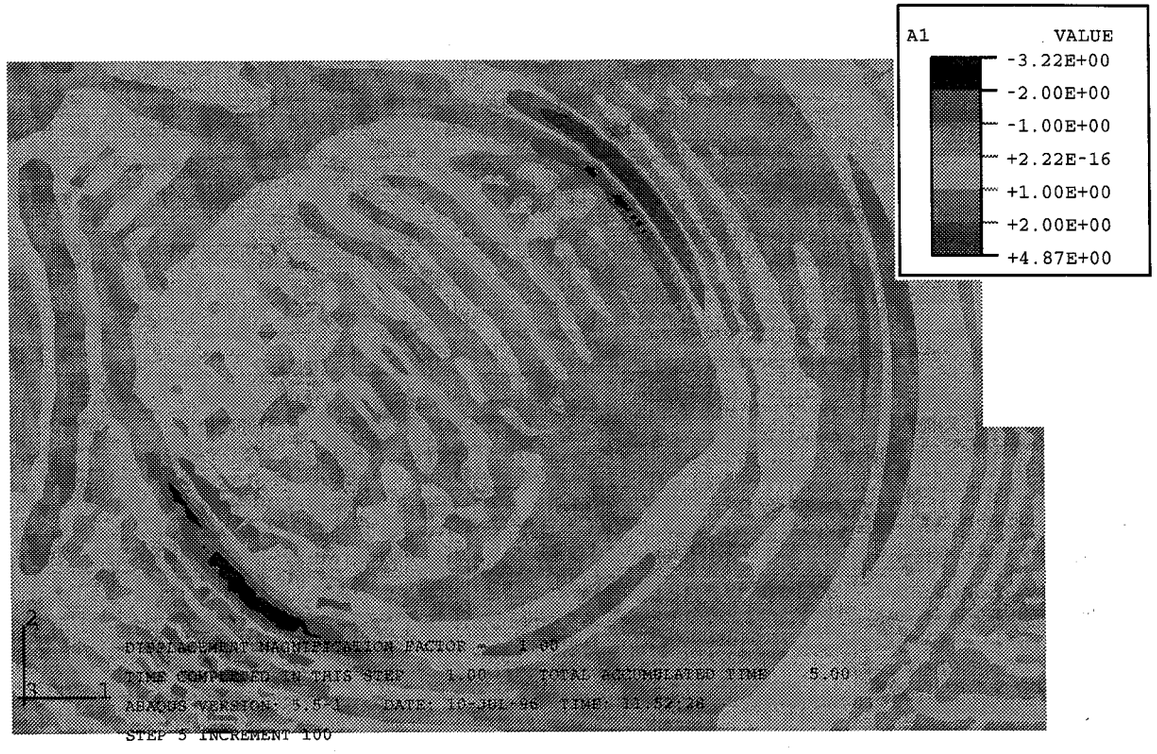


XMIN 1.010E+00
XMAX 6.000E+00
YMIN -1.909E+00
YMAX 6.388E-01

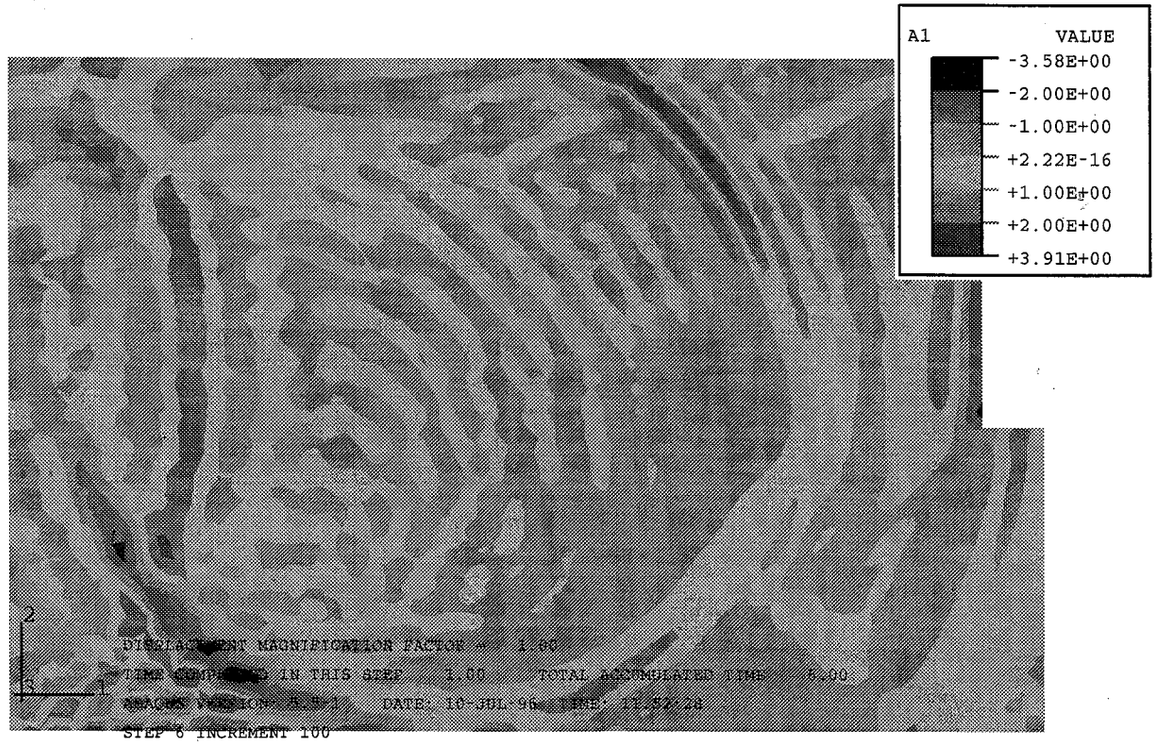
ABAQUS


A1
VALUE
-5.60E+00
-2.00E+00
-1.00E+00
+2.22E-16
+1.00E+00
+2.00E+00
+6.08E+00
2
3
1
DISPLACEMENT MAGNIFICATION FACTOR = 1.00
TIME COMPLETED IN THIS STEP 1.00 TOTAL ACCUMULATED TIME 4.00
ABAQUS VERSION: 5.5-1 DATE: 10-JUL-96 TIME: 11:52:28
STEP 4 INCREMENT 100

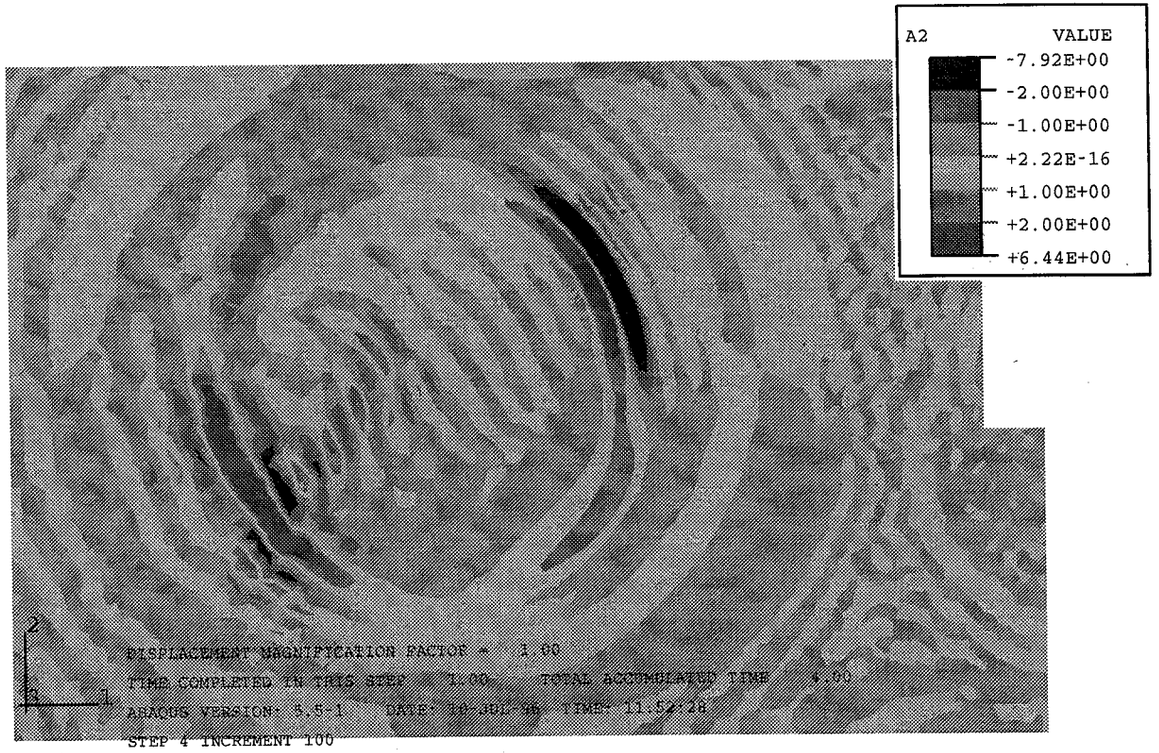
ABAQUS


A1
VALUE
-3.22E+00
-2.00E+00
-1.00E+00
+2.22E-16
+1.00E+00
+2.00E+00
+4.87E+00
DISPLACEMENT MAGNIFICATION FACTOR = 1.00
TIME COMPLETED IN THIS STEP 1.00 TOTAL ACCUMULATED TIME 5.00
ABAQUS VERSION: 5.5-1 DATE: 10-DEC-96 TIME: 11:52:28
STEP 3 INCREMENT 100
2
3
1

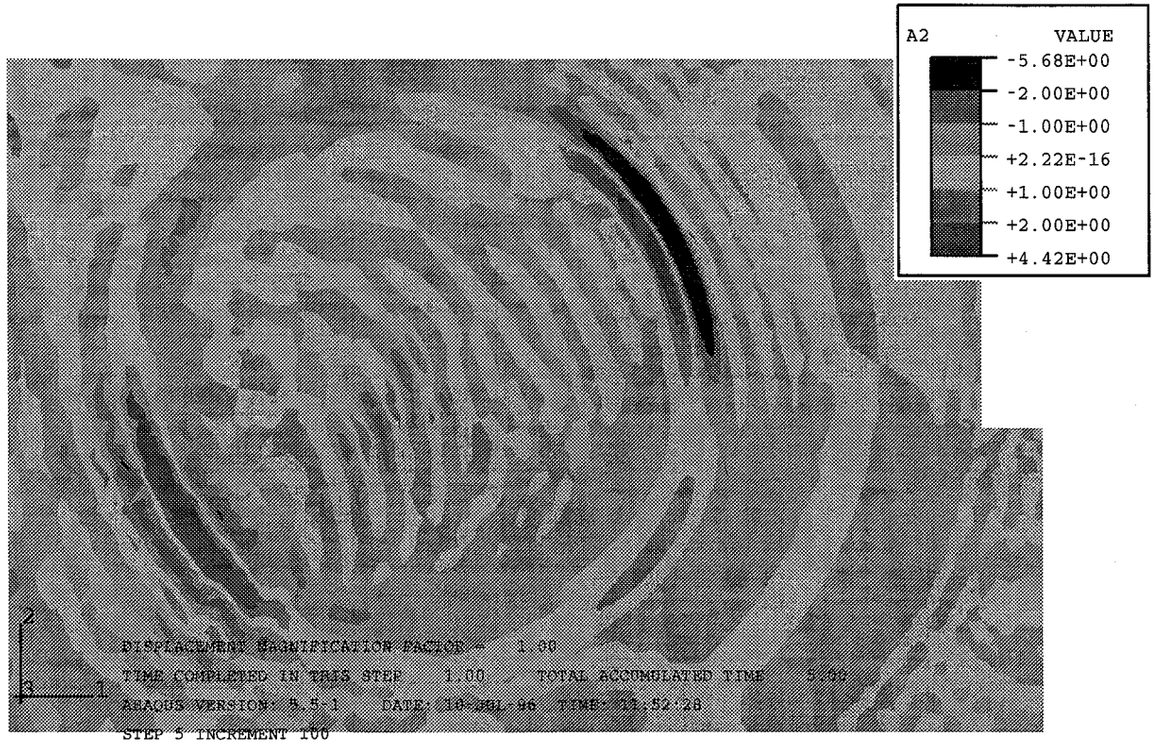
ABAQUS


A1
VALUE
-3.58E+00
-2.00E+00
-1.00E+00
+2.22E-16
+1.00E+00
+2.00E+00
+3.91E+00
2
3
1
DISPLACEMENT MAGNIFICATION FACTOR = 1.00
TIME COMPLETED IN THIS STEP 1.00 TOTAL ACCUMULATED TIME 6.00
ABAQUS VERSION: 5.8-1 DATE: 10-JUL-98 TIME: 11:52:28
STEP 6 INCREMENT 100

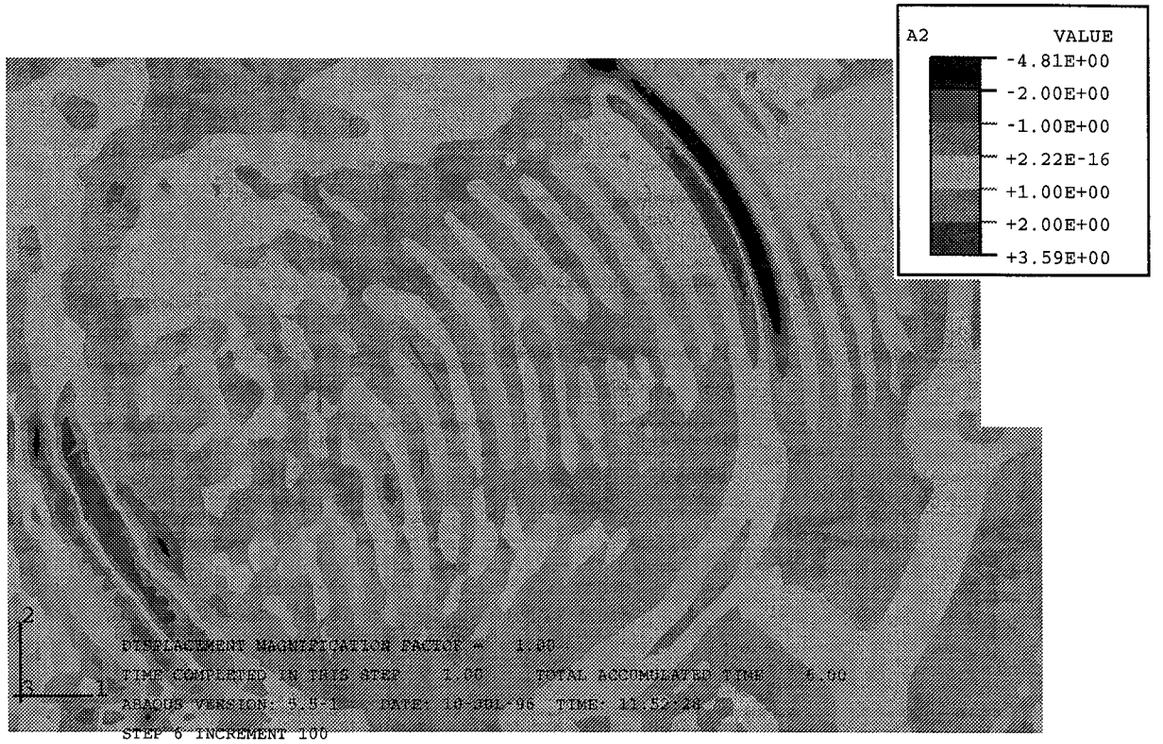
ABAQUS


A2
VALUE
-7.92E+00
-2.00E+00
-1.00E+00
+2.22E-16
+1.00E+00
+2.00E+00
+6.44E+00
2
3
1
DISPLACEMENT MAGNIFICATION FACTOR = 1.00
TIME COMPLETED IN THIS STEP = 1.00 TOTAL ACCUMULATED TIME 4.00
ABAQUS VERSION: 5.5-1 DATE: 10-JUL-96 TIME: 11:52:18
STEP 4 INCREMENT 100

ABAQUS


A2
VALUE
-5.68E+00
-2.00E+00
-1.00E+00
+2.22E-16
+1.00E+00
+2.00E+00
+4.42E+00
DISPLACEMENT MAGNIFICATION FACTOR = 1.00
TIME COMPLETED IN THIS STEP 1.00 TOTAL ACCUMULATED TIME 5.00
ABAQUS VERSION: 5.5-1 DATE: 10-JUL-96 TIME: 11:52:28
STEP 5 INCREMENT 100

ABAQUS


A2
VALUE
-4.81E+00
-2.00E+00
-1.00E+00
+2.22E-16
+1.00E+00
+2.00E+00
+3.59E+00
2
3
1
DISPLACEMENT MAGNIFICATION FACTOR = 1.00
TIME COMPLETED IN THIS STEP 2.00 TOTAL ACCUMULATED TIME 6.00
ABAQUS VERSION: 5.5-1 DATE: 10-JUL-96 TIME: 11:52:28
STEP 6 INCREMENT 100

11 Attempts at Controlling Energy Reflections from Model Boundaries to Achieve Longer Monitoring Times

Modifications to the basic model (i.e., m05Base) were necessitated by observations (reported in section 10.5 on 7/19/96, page 146 of Notebook 183) that: (1) fault slip and ground motions need to be monitored for longer than 5 seconds following the simulated seismic event; and (2) reflections of acceleration wave fronts from the sides and bottom of the model back into the area of interest occur within approximately 4-5 seconds of the seismic event. Consequently, the model geometry was modified so that longer analyses could be performed and so as to eliminate boundary reflections.

11.1 Use of Infinite Elements

The first attempt was called m06Base and made use of ABAQUS infinite elements along the side and bottom boundaries of the model. The infinite elements do not permit additional boundary conditions to be applied to them so the "rollers" were removed from the sides and bottom of the model. Unfortunately, this configuration provided no restraint against rigid-body motion of the entire model because in the ABAQUS formulation of two-dimensional infinite elements, the nodal displacements at "infinity" are not fixed to be zero as they are for three-dimensional infinite elements. As a result, the entire model was free to translate within the xy plane (i.e., the plane containing the model) which resulted in numerical instability.

A new model, m07Base, was constructed from m06Base by arbitrarily assigning no displacement boundary conditions to 4 nodes within the main model zone. It was hoped that this would effectively "pin" down the model, prevent the rigid-body translations, and therefore eliminate some of the numerical instability. However, this restraint proved to be insufficient. Consequently, the use of infinite elements was abandoned.

11.2 Energy Absorbing Boundaries

Since the infinite elements were unsuccessful, it was decided to modify m05Base by adding an additional 15 km of material to the sides and bottom of the model (referred to as m08Base). The additional material was made into three rows of 5 km thick elements, with the innermost two rows assigned material properties identical to the footwall and hangingwall regions (i.e., linear-elastic with very small damping). The

KJS.

outermost row was assigned linear-elastic properties with a very large damping coefficient. This region is called the "absorbent boundary." While the absorbing layer did prevent reflections from the model boundaries, the sharp transition from the main model mesh (approximately 500 by 500 m elements) to the new 5 km wide elements of the new region produced its own reflection. This new problem appears to be solely the result of the rapid change in mesh size.

Consequently, a further modification (m09Base) was setup such that 5 km of new material was added beyond the model zone of m05Base. This new material was divided into 5 rows of elements that get progressively larger toward the moel boundaries, with the innermost row composed of elements that are nearly the same size as in the main model zone. The 4 innermost rows are assigned material properties identical to the main footwall and hangingwall regions (i.e., linear elastic with minor damping). The outermost row is assigned a linear elastic material with very large damping (referred to as the "absorbent boundary"). Until further notice, all subsequent models will make use of m09Base as the starting condition.

11.3 Results of analysis cases sc01, sc02, sc03, and sc04 for m09Base

11.3.1 Case sc01 (shallow-focus earthquake on Fault F0)

Case sc01 was monitored for 10 seconds after the simulated seismic event (total time of 11 seconds). Maximum ground accelerations were observed at the group 3 monitoring nodes (midway between faults F0 and F1) approximately 2.5 seconds following the seismic event. The maximum horizontal acceleration of 2.087 m/s^2 (0.21 g) occurred at node 2110 (depth of 500 m), while the maximum vertical acceleration of 4.290 m/s^2 (0.44 g) occurred at node 2072 (on the surface). The average group 3 accelerations were 0.16 and 0.34 g, horizontal and vertical respectively.

For fault F0, the maximum displacement of 4 m took place at a depth of 1.98 km and a down-dip fault length of 2.29 km, and the surface displacement 10 seconds after the event was 2.92 m. The down-dip rupture length for F0 varied from 15 to 31 km and the average fault displacement ranged from 1.4 down to 0.7 m. Measurable slip did not occur on faults F1 and F2 in repsonse to the shallow earthquake on F0.

Simulated earthquake magnitudes of 6.9 and 6.8 were calculated for model sc01 using the empirical formulas of Wells & Coppersmith (1994) for maximum surface displacement and down-dip rupture length, respectively. A magnitude range of 6.4 to 6.7

was calculated from the seismic moment (Kanamori & Anderson 1975) by estimating a rupture area range (see previous discussion of rupture area calculation).

11.3.2 Case sc02 (intermediate-focus earthquake on Fault F0)

Case sc02 was monitored for 12 seconds after the simulated seismic event (total time of 13 seconds). Maximum ground accelerations were observed at group 4 nodes (between faults F1 and F2) approximately 4 seconds after the seismic event. The maximum horizontal acceleration of 4.691 m/s^2 (0.48 g) occurred at node 10974 (depth of 250 m), while the maximum vertical acceleration of 5.388 m/s^2 (0.55 g) occurred at node 10957 (on the surface). The average group 4 accelerations were 0.38 and 0.41 g, horizontal and vertical respectively.

For fault F0, the maximum displacement of 11.6 m took place at a depth of 6.37 km and a down-dip fault length of 7.36 km, and the surface displacement 12 seconds after the event was 4.12 m. The down-dip rupture length for F0 varied from 21 to 31 km and the average fault displacement ranged from 3.6 m down to 2.62 m. Maximum and surface displacements of 0.13 and 0.09 m, respectively, were measured on fault F1. For fault F2, maximum and surface displacements of 0.40 and 0.03 m, respectively, were observed.

Simulated earthquake magnitudes of 7.0 were calculated for model sc02 using the empirical formulas of Wells & Coppersmith (1994) for both maximum surface displacement and down-dip rupture length. A magnitude range of 6.9 to 7.2 was calculated from the rupture area range.

11.3.3 Case sc03 (deep-focus earthquake on steep segment of Fault F0)

Case sc03 was monitored for 12 seconds after the simulated seismic event (total time of 13 seconds). Maximum ground accelerations were observed at group 5 nodes (right of fault F2) approximately 6 seconds after the seismic event. The maximum horizontal acceleration of 4.820 m/s^2 (0.49 g) occurred at node 3687 (depth of 250 m), while the maximum vertical acceleration of 4.932 m/s^2 (0.50 g) occurred at node 3669 (on the surface). The average group 5 accelerations were 0.43 and 0.39 g, horizontal and vertical respectively.

For fault F0, the maximum displacement of 13.9 m took place at a depth of 10.03 km and a down-dip fault length of 11.58 km, and the surface displacement 12 seconds after the event was 2.39 m. The down-dip rupture length for F0 varied from 21 to 31 km

and the average fault displacement ranged from 3.6 m down to 2.6 m. Maximum and surface displacements of 0.13 and 0.02 m, respectively, were measured on fault F1. For fault F2, maximum and surface displacements of 0.26 m were observed.

Simulated earthquake magnitudes of 6.9 and 7.0 were calculated for model sc03 using the empirical formulas of Wells & Coppersmith (1994) for maximum surface displacement and down-dip rupture length, respectively. A magnitude range of 6.8 to 7.2 was calculated from the rupture area range.

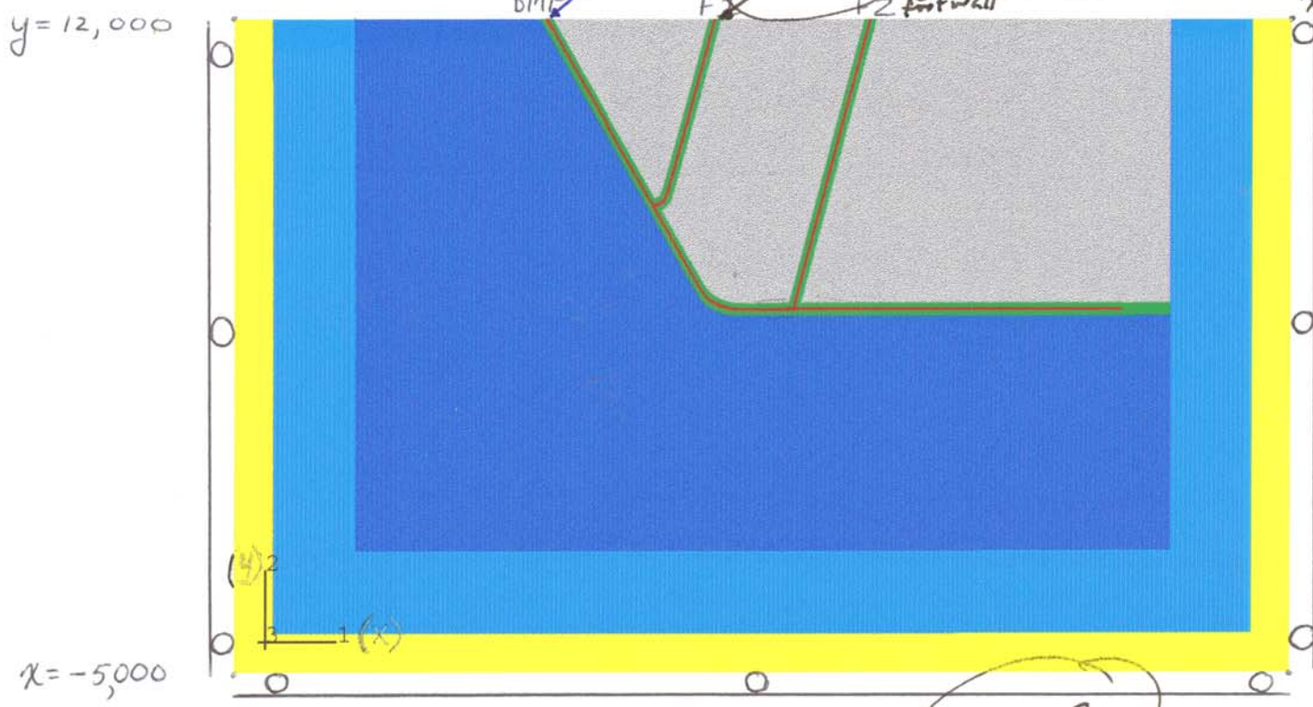
11.3.4 Case sc04 (deep-focus earthquake on detachment segment of Fault F0)

Case sc04 was monitored for 6 seconds after the simulated seismic event (total time of 7 seconds). This model could not be monitored further because the horizontal component of the acceleration was so strong as to overwhelm the absorbing boundary at the base of the model and reflect back into the zone of interest. Maximum ground accelerations were observed at group 4 nodes approximately 5 seconds after the seismic event. The maximum horizontal acceleration of 17.640 m/s^2 (1.80 g) and the maximum vertical acceleration of 1.798 m/s^2 (0.26 g) occurred at node 10957 (on the surface). The average group 4 accelerations were 1.17 and 0.18 g, horizontal and vertical respectively.

For fault F0, the maximum displacement of 4.43 m took place at a depth of 11.95 km and a down-dip fault length of 16.29 km, and the surface displacement 6 seconds after the event was 0.05 m. The down-dip rupture length for F0 varied from 21 to 29 km and the average fault displacement ranged from 0.72 m down to 0.57 m. Maximum and surface displacements of 0.10 and 0.06 m, respectively, were measured on fault F1. For fault F2, maximum and surface displacements of 0.07 and 0.02 m, respectively, were observed.

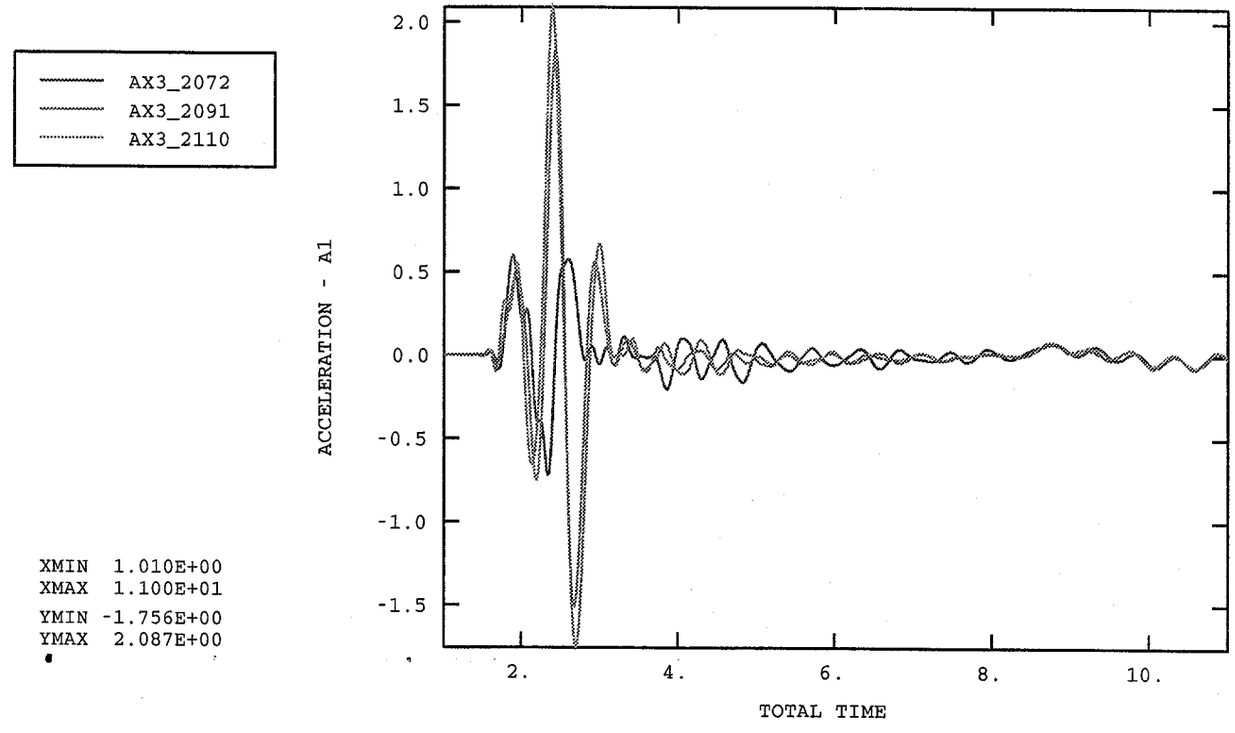
Simulated earthquake magnitudes of 5.7 and 7.0 were calculated for model sc04 using the empirical formulas of Wells & Coppersmith (1994) for maximum surface displacement and down-dip rupture length, respectively. A magnitude range of 6.4 to 6.7 was calculated from the rupture area range.

ABAQUS



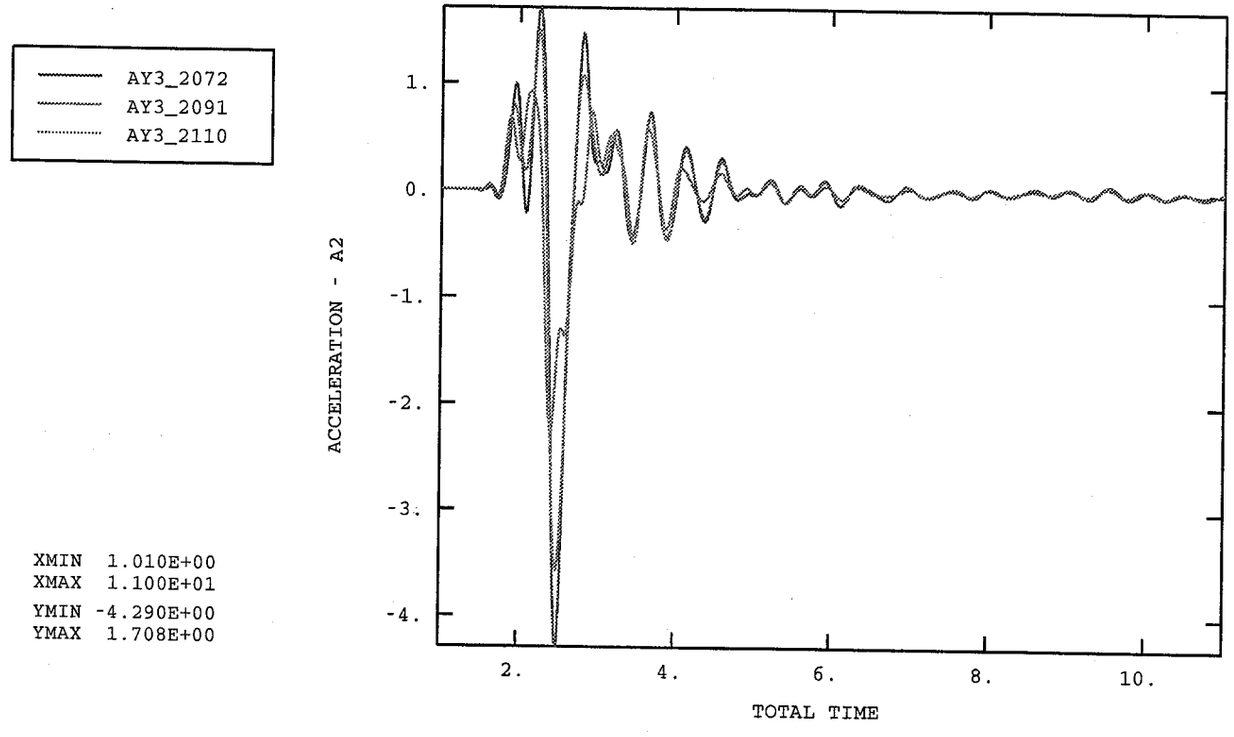
Top of F2 hangingwall at x = 21,500 m
Top of F2 footwall hangingwall @ x = 21,600 m
GW
Red line 100 m thick

ABAQUS



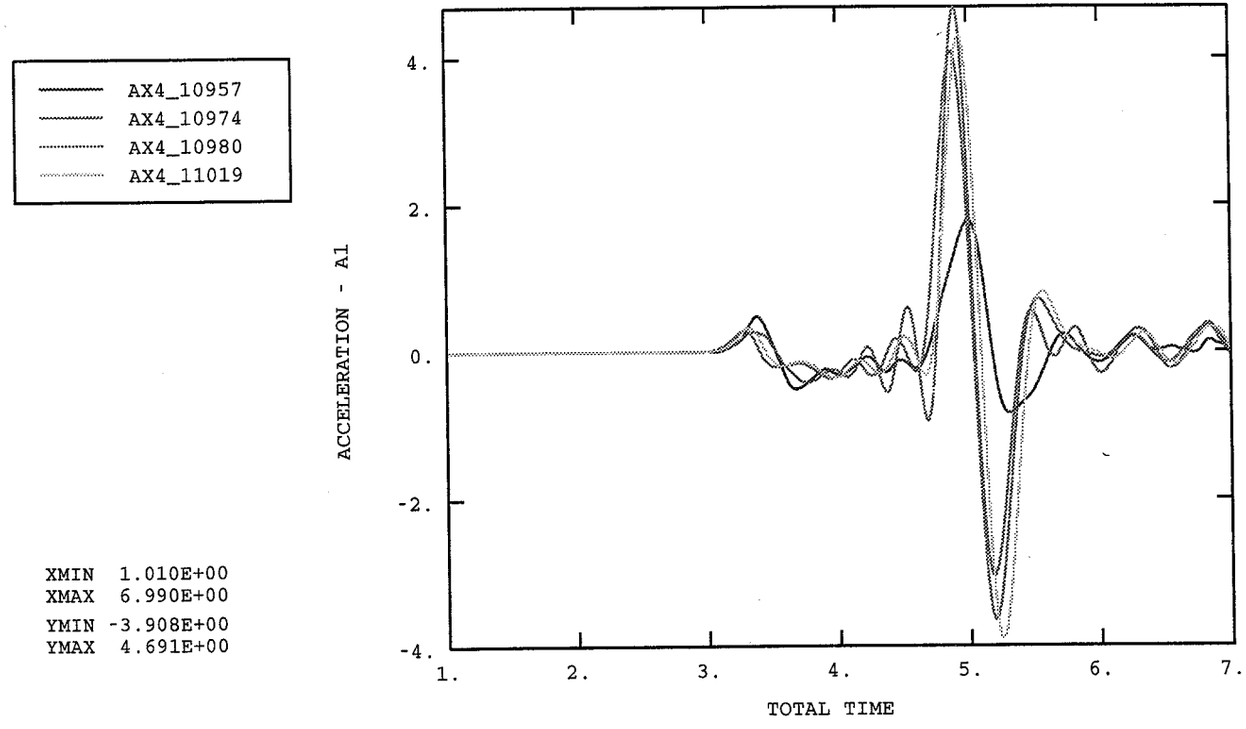
XMIN 1.010E+00
XMAX 1.100E+01
YMIN -1.756E+00
YMAX 2.087E+00

ABAQUS



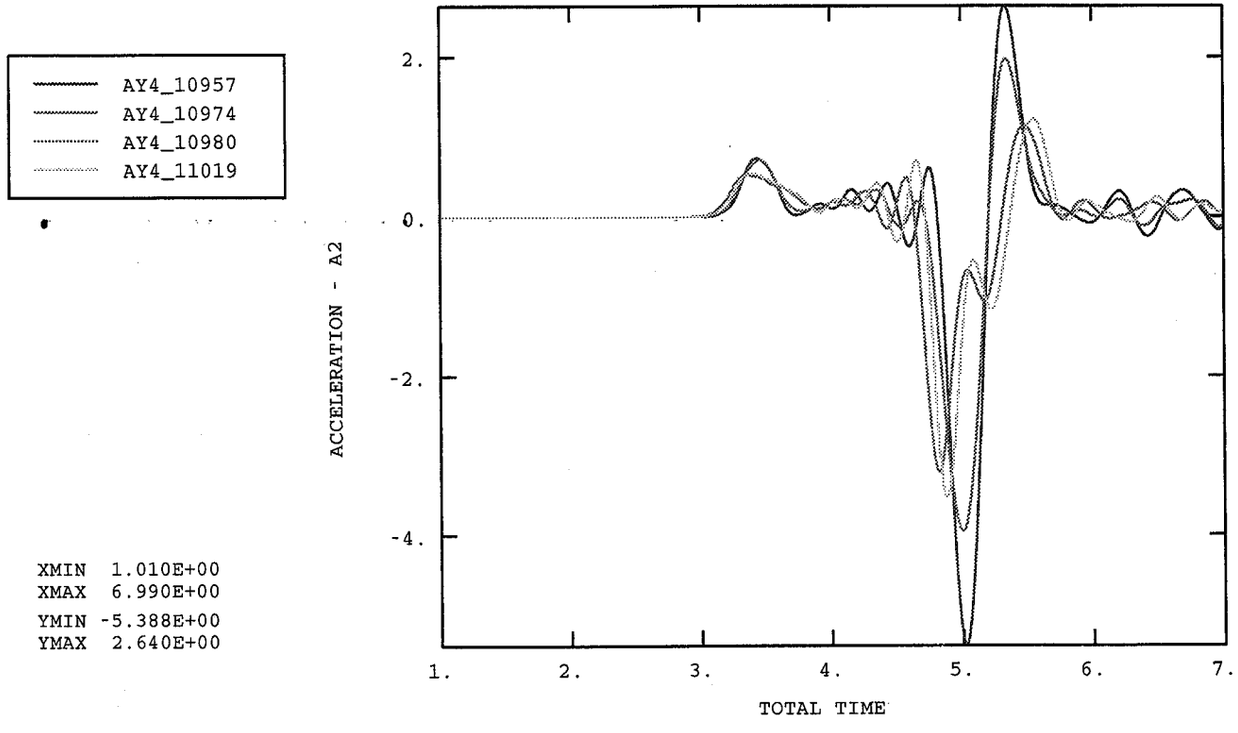
XMIN 1.010E+00
XMAX 1.100E+01
YMIN -4.290E+00
YMAX 1.708E+00

ABAQUS



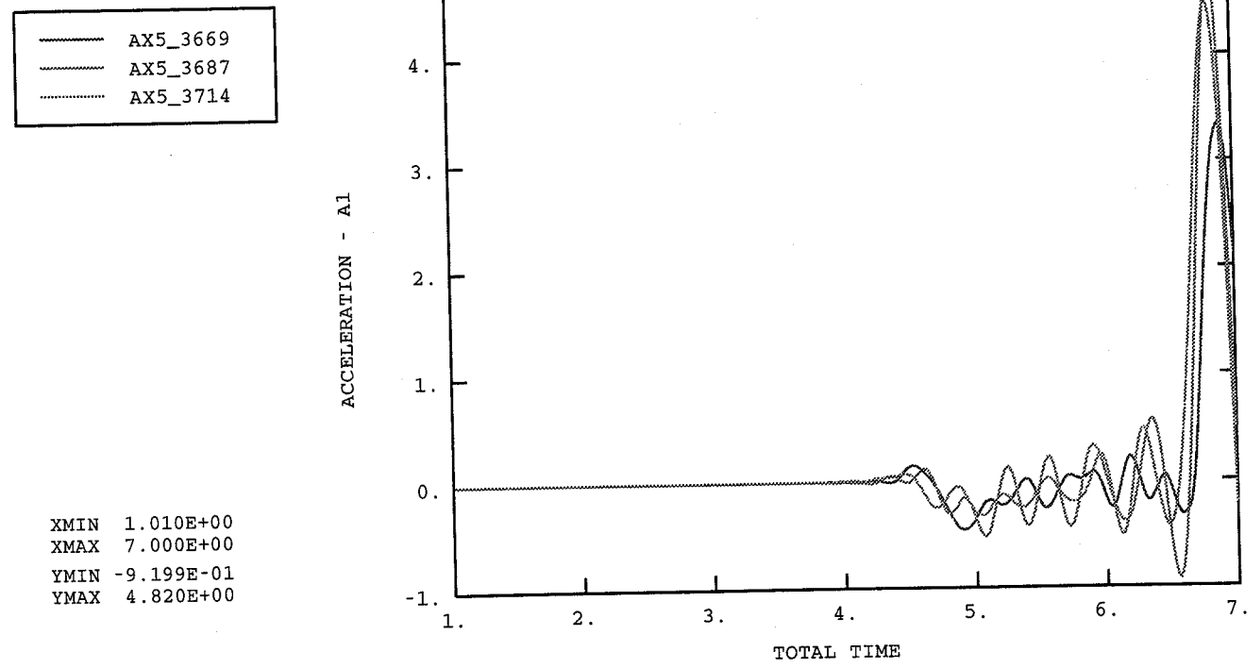
XMIN 1.010E+00
XMAX 6.990E+00
YMIN -3.908E+00
YMAX 4.691E+00

ABAQUS



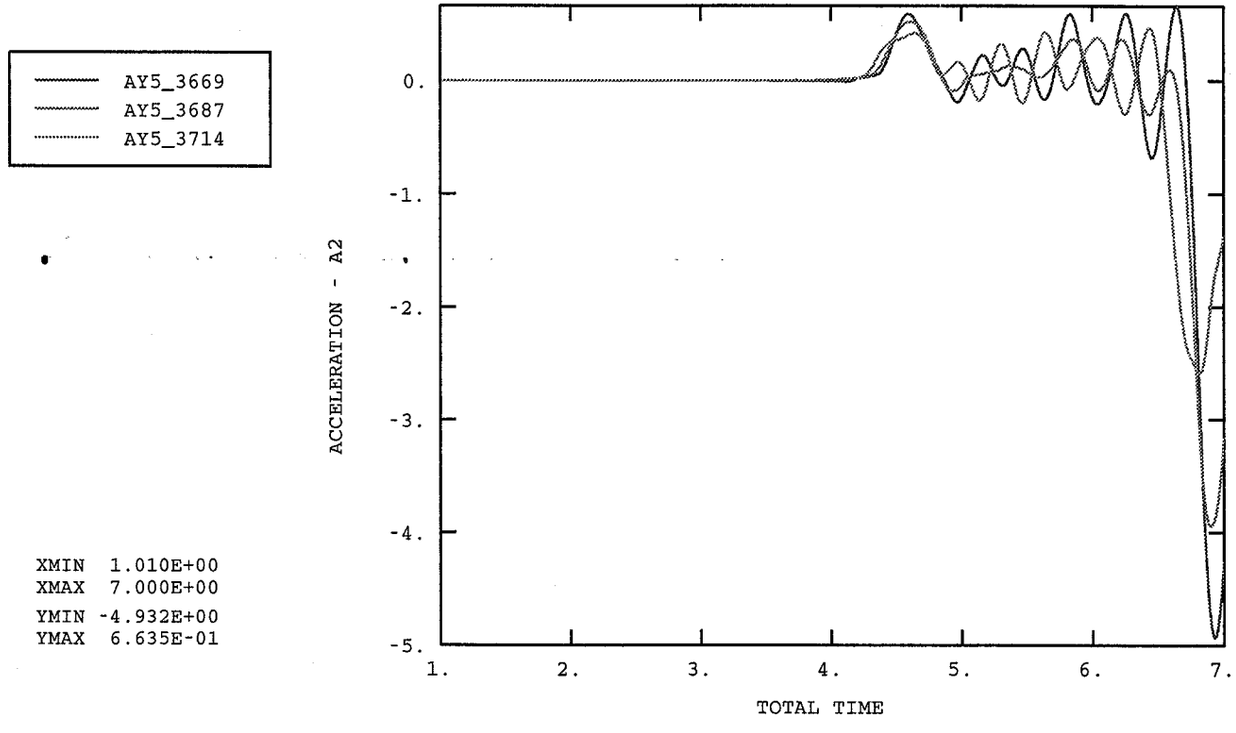
XMIN 1.010E+00
XMAX 6.990E+00
YMIN -5.388E+00
YMAX 2.640E+00

ABAQUS



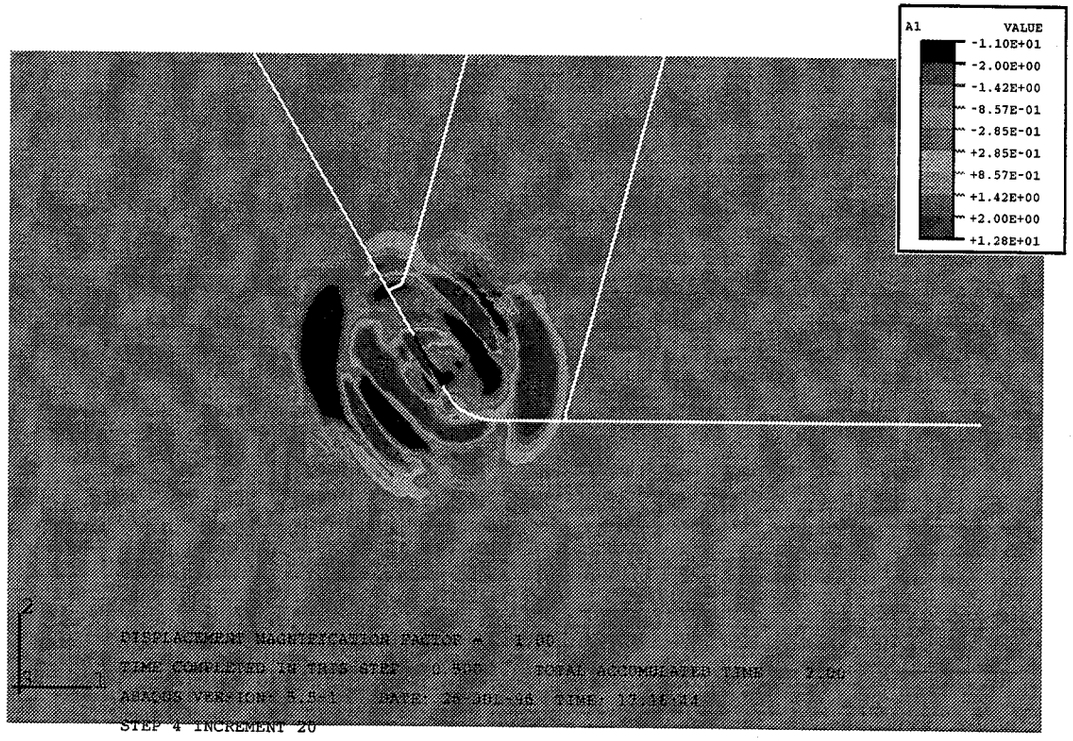
XMIN 1.010E+00
XMAX 7.000E+00
YMIN -9.199E-01
YMAX 4.820E+00

ABAQUS

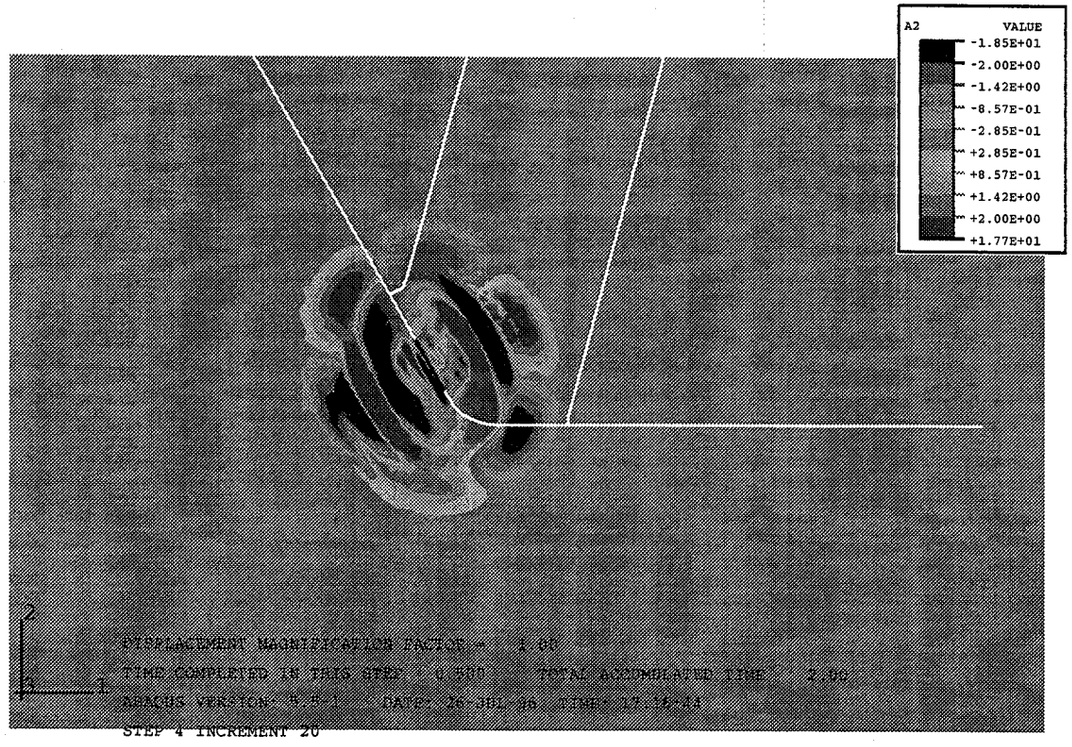


XMIN 1.010E+00
XMAX 7.000E+00
YMIN -4.932E+00
YMAX 6.635E-01

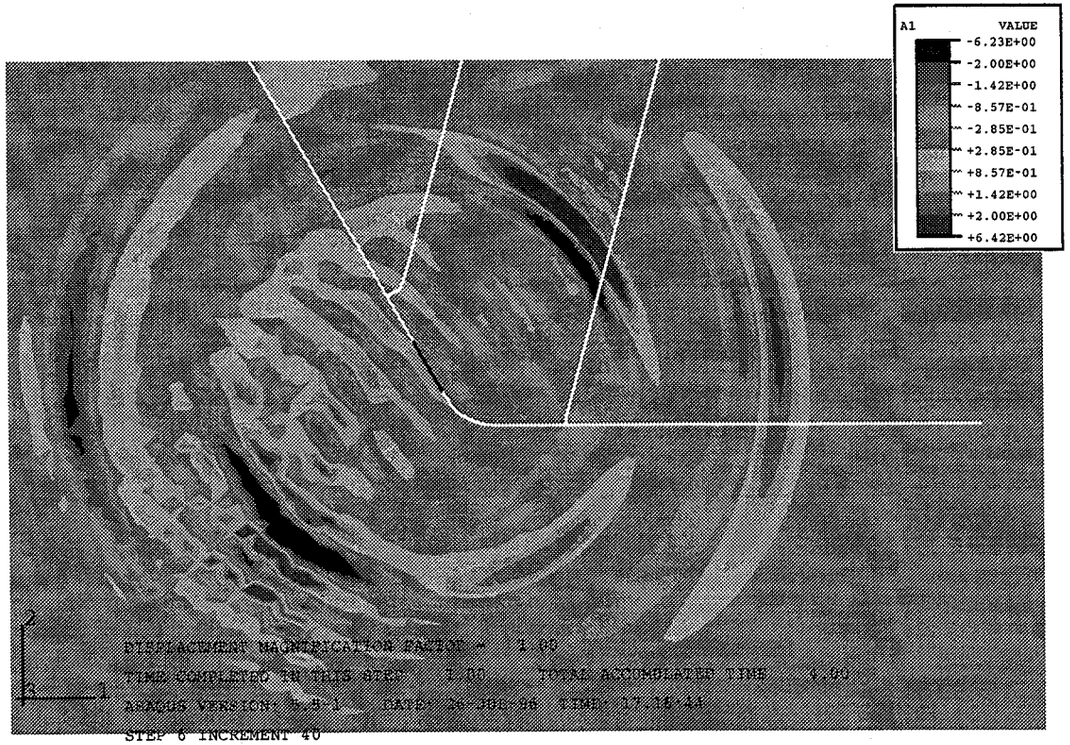
ABAQUS


A1
VALUE
-1.10E+01
-2.00E+00
-1.42E+00
-8.57E-01
-2.85E-01
+2.85E-01
+8.57E-01
+1.42E+00
+2.00E+00
+1.28E+01
DISPLACEMENT MAGNIFICATION FACTOR = 1.00
TIME COMPLETED IN THIS STEP 0.500 TOTAL ACCUMULATED TIME 2.00
ABAQUS VERSION: 5.5-1 DATE: 28-JUL-98 TIME: 17:16:14
STEP 4 INCREMENT 20

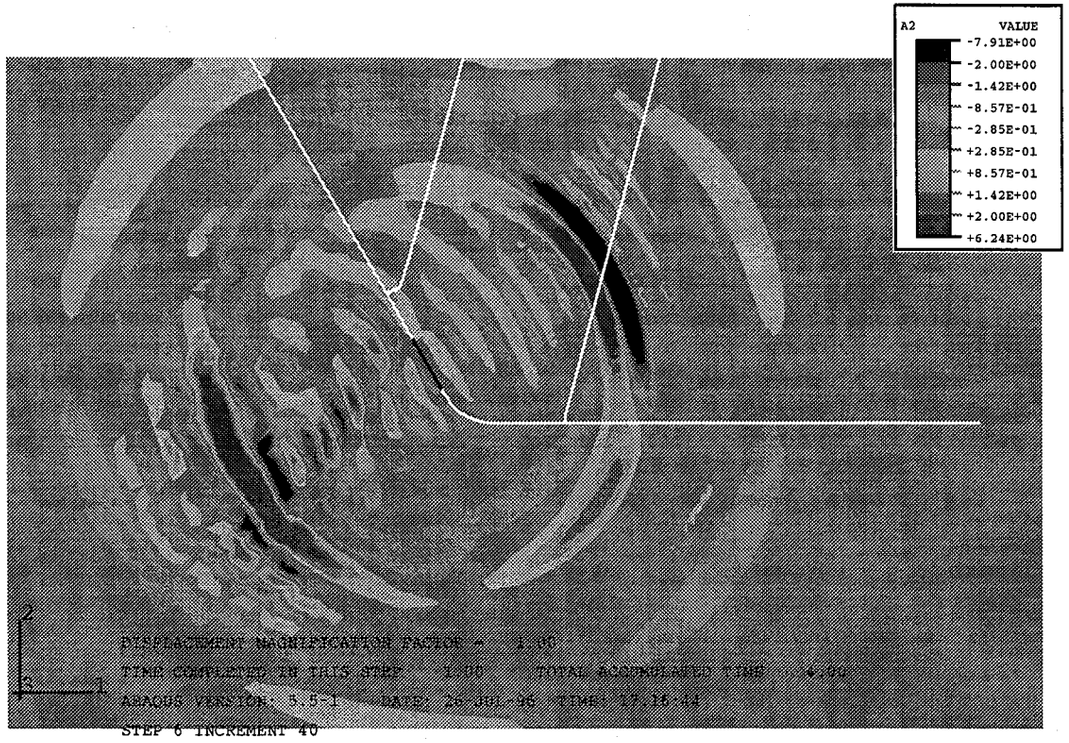
ABAQUS


A2
VALUE
-1.85E+01
-2.00E+00
-1.42E+00
-8.57E-01
-2.85E-01
+2.85E-01
+8.57E-01
+1.42E+00
+2.00E+00
+1.77E+01
2
3
1
DISPLACEMENT MAGNIFICATION FACTOR = 1.00
TIME COMPLETED IN THIS STEP = 0.500 TOTAL ACCUMULATED TIME = 2.00
ABAQUS VERSION: 5.5-1 DATE: 26-JUL-96 TIME: 17:16:14
STEP 4 INCREMENT 20

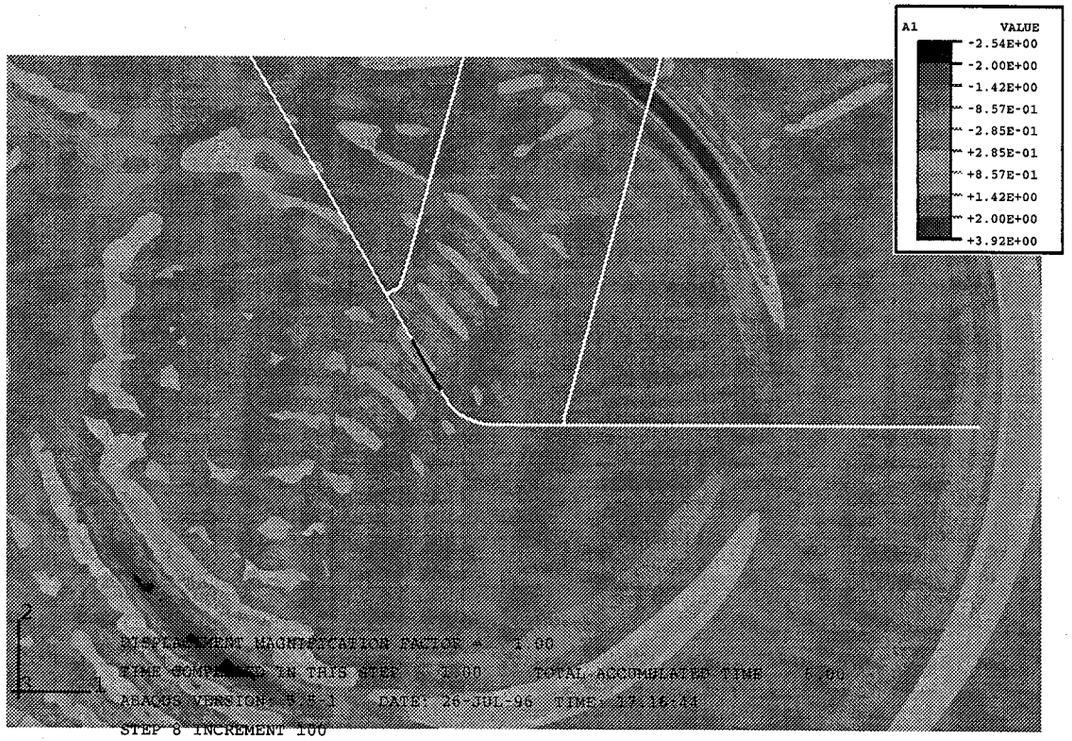
ABAQUS


A1
VALUE
-6.23E+00
-2.00E+00
-1.42E+00
-8.57E-01
-2.85E-01
+2.85E-01
+8.57E-01
+1.42E+00
+2.00E+00
+6.42E+00
2
3
1
DISPLACEMENT MAGNIFICATION FACTOR = 1.00
TIME COMPLETED IN THIS STEP 1.00 TOTAL ACCUMULATED TIME 1.00
ABAQUS VERSION: 6.8-1 DATE: 26 JUN 08 TIME: 17:15:44
STEP 6 INCREMENT 40

ABAQUS


A2
VALUE
-7.91E+00
-2.00E+00
-1.42E+00
-8.57E-01
-2.85E-01
+2.85E-01
+8.57E-01
+1.42E+00
+2.00E+00
+6.24E+00
DISPLACEMENT MAGNIFICATION FACTOR = 1.00
TIME COMPLETED IN THIS STEP 1.00 TOTAL ACCUMULATED TIME 4.00
ABAQUS VERSION: 5.5-1 DATE: 26-JUL-96 TIME: 17:16:44
STEP 6 INCREMENT 40

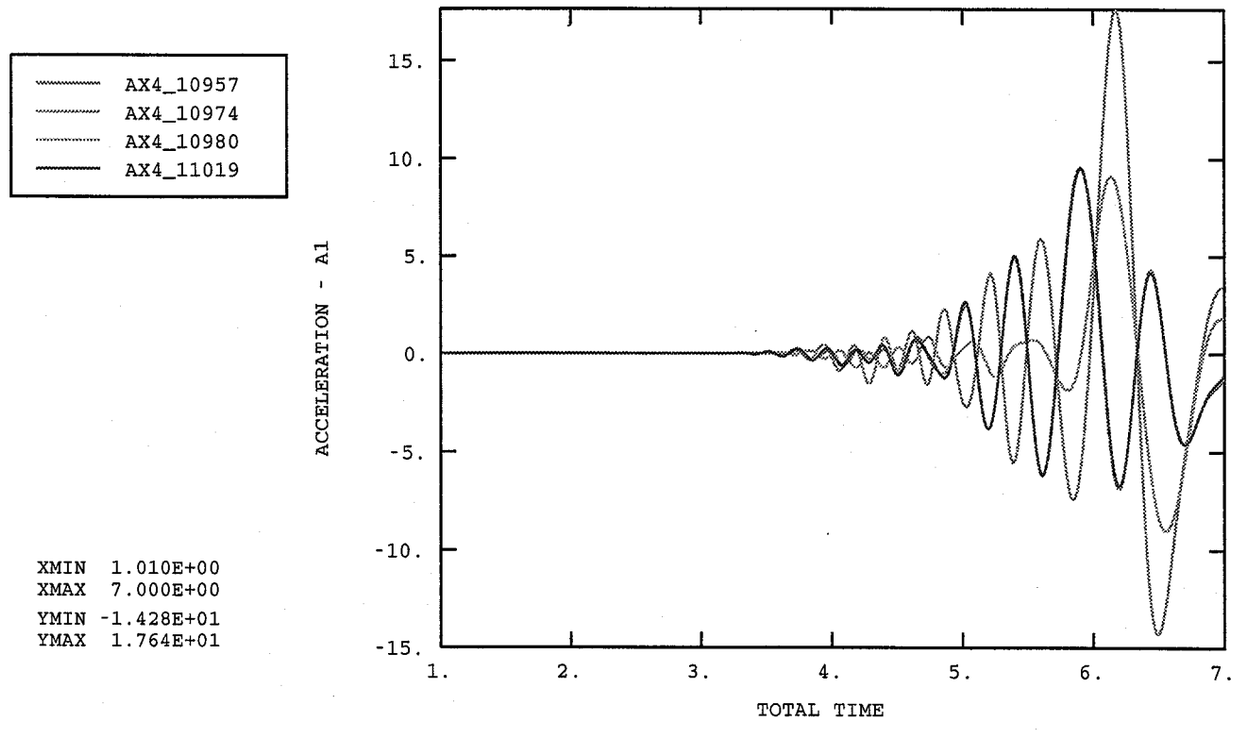
ABAQUS


A1
VALUE
-2.54E+00
-2.00E+00
-1.42E+00
-8.57E-01
-2.85E-01
+2.85E-01
+8.57E-01
+1.42E+00
+2.00E+00
+3.92E+00
DISPLACEMENT MAGNIFICATION FACTOR = 1.00
TIME COMPLETED IN THIS STEP = 2.00 TOTAL ACCUMULATED TIME 8.00
ABAQUS VERSION: 5.5-1 DATE: 26-JUL-96 TIME: 17:16:44
STEP 8 INCREMENT 100
2
1

ABAQUS

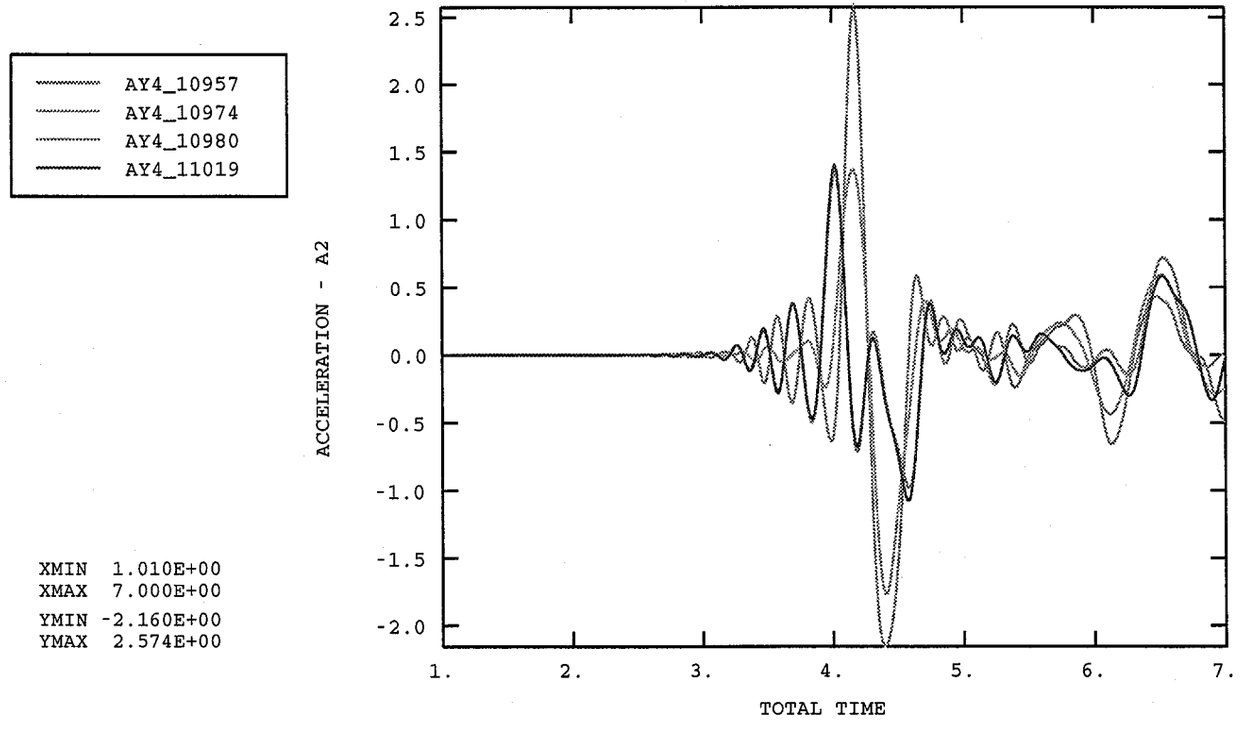

A2
VALUE
-4.85E+00
-2.00E+00
-1.42E+00
-8.57E-01
-2.85E-01
+2.85E-01
+8.57E-01
+1.42E+00
+2.00E+00
+3.21E+00
DISPLACEMENT MAGNIFICATION FACTOR = 1.00
TIME COMPLETED IN THIS STEP 1.00 TOTAL ACCUMULATED TIME 6.00
ABAQUS VERSION: 5.5-1 DATE: 26-JUL-96 TIME: 17:15:14
STEP 8 INCREMENT 100

ABAQUS



XMIN 1.010E+00
XMAX 7.000E+00
YMIN -1.428E+01
YMAX 1.764E+01

ABAQUS



XMIN 1.010E+00
XMAX 7.000E+00
YMIN -2.160E+00
YMAX 2.574E+00

12 Detailed Acceleration Monitoring

This section documents a change in the manner in which near-ground accelerations are monitored in the finite element models. Previously, (refer to section 7.7 of notebook 183) accelerations were monitored at 3 to 5 nodes in each of 7 groups spaced across the model. To provide a better comparison of model derived accelerations with those obtained with published attenuation functions, it was decided to montior the acceleration at every surface node across the model as well as at all nodes at a depth of approximately 300 m. The surface nodes are defined by the node-set "ndtop" which is defined in the *m09Base.res* file. The 300 m deep nodes (depth actually varies from about 200 to 400 m) are defined in the node-set "nd300dep" which is defined in each model input file (e.g., sc09, sc10, etc.). An example of this new node-set defintion is given here:

```
**********************************************************************
**********************************************************************
      ***
      *** Define nd300dep nodeset
      ***   Nodes at depth of 300 +/- 100 m (i.e., 200-400 m)
      ***
      *NSET, NSET=nd300dep
          2,      55,     108,     161,     214,     267,    1326,    1376,
       1426,    1476,    1526,    1576,    1644,    1718,    1792,    1866,
       1940,    2014,    2086,    2087,    2088,    2089,    2090,    2091,
       2099,    2107,    2122,    2159,    3678,    3679,    3680,    3681,
       3682,    3683,    3684,    3685,    3686,    3687,    3688,    3689,
       3690,    3692,    3693,    3694,    3695,    5490,    5491,    5492,
       5493,    5494,    6807,    6860,    6913,    6966,    7019,    7072,
       7125,    7178,    7231,    7284,   10309,   10364,   10454,   10542,
      10632,   10722,   10804,   10887,   10974,   11053,   11142,   11225,
      11319,   11396,   11482,   11556,   14366,   14419,   14472,   14525,
      14578,
      ***
      *** Define combined nodeset (ndtop & nd300dep)
      ***
      *NSET, NSET=ndcomb
      ndtop,
      nd300dep,
      ***
      ***
**********************************************************************
**********************************************************************
```

The last *NSET command simply groups together ndtop and nd300dep into a single node set called ndcomb so that writing the accelerations to the results file is easier.

12.1 Acceleration Extraction from ABAQUS History File

This ABAQUS journal file, *max-accel.jnl*, extracts the maximum horizontal (a1) and vertical (a2) accelerations from the nodes on the surface (ndtop) and at about 300-m depth (nd300dep).

```
**********************************************************************
**********************************************************************
** Script "max-accel.jnl"
**
** Change result filename for each model!
**
result, file=sc06
**
**
set, reportfile=ndtop.a1
readcurve, name=test, var=a1, node=ndtop
define curve, name=abs, operation=abs
test
define curve, name=max, operation=maximum
abs
print curve
max

delete curve, name=all curves
y

set, reportfile=ndtop.a2
readcurve, name=test, var=a2, node=ndtop
define curve, name=abs, operation=abs
test
define curve, name=max, operation=maximum
abs
print curve
max

delete curve, name=all curves
y

set, reportfile=nd300.a1
readcurve, name=test, var=a1, node=nd300dep
define curve, name=abs, operation=abs
test
define curve, name=max, operation=maximum
abs
print curve
max

delete curve, name=allcurves
y

set, reportfile=nd300.a2
readcurve, name=test, var=a2, node=nd300dep
define curve, name=abs, operation=abs
test
define curve, name=max, operation=maximum
abs
print curve
max

end
**********************************************************************
**********************************************************************
```

This file can be used for any source case as long as the "result, file=name" line is changed for a particular case. As it is written, this journal file can be used with ABAQUS in batch mode. Four example, the following line would be used for model case sc06:

abaqus55 post restart=sc06 input=max-accel.jnl

Four output files are generated (ndtop.a1, ndtop.a2, nd300.a1, nd300.a2) that are then used to generate acceleration profiles as described below.

12.2 Creation of Acceleration Profile

The program, *AccProfile.c*, takes the surface and 300-m depth accelerations, and sorts and combines them with the x-y nodal coordinates to create a file that can be used to plot an accleration profile.

```
**********************************************************************
**********************************************************************
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

struct Point{
  int node;
  float x,y,maxacc;
  struct Point *next;
};

int GetMaxAcc(float* amax, int numNodes);
int GetNodeNums(int* nodeNums,char* buf);
int InsertAcc(struct Point *first,int node,float amax);

main(int argc, char* argv[])
{

  char buf[121];
  int *nodeNums,node,numNodes,maxNumNodes=4;
  int i;
  float *amax,x,y;
  struct Point *FirstPoint,*LastPoint,*currentPoint;
  FILE *Fxy;

/*************************************************************
******
  AccProfile.c

  This code prepares horizontal profile of maximum acceleration
  using data output from an ABAQUS/Post PRINT CURVE command
  (where each curve is a time history of maximum acceleration
prepared
  by applying ABAQUS/Post function DEFINE CURVE with
OPERATION=ABS and
  OPERATION=MAXIMUM to selected acceleration history data.

  The x- and y-coordinates of the nodal points are read from
file name
  given as command-line argument

  author                    G. I. Ofoegbu
  Date                      August 21 1996
  System Requirement:       ANSI C under UNIX
                            Compile with acc on SUN OS (cc
will fail)
*************************************************************
********/

/* Process command-line argument to determine xy-file name */

  if (argc < 2)
    DumpAndQuit("Usage: ProgramName xyFileName < accFileName >
outputfile");

  Fxy = fopen(argv[1],"r");
  if (!Fxy){
    sprintf(buf,"Unable to open file %s",argv[1]);
    DumpAndQuit(buf);
  }
  amax = (float *)malloc(maxNumNodes*sizeof(float));
  nodeNums = (int *)malloc(maxNumNodes*sizeof(int));
  if (!amax || !nodeNums)
    DumpAndQuit("Memory allocation error");
```

```
/* Read coordinates file and store nodes sorted in increasing x
order */

  FirstPoint = LastPoint = NULL;
  while (fgets(buf,120,Fxy))
    if (sscanf(buf,"%d %f %f %*f %*f",&node,&x,&y) == 3){
      currentPoint = (struct Point *)malloc(sizeof(struct
Point));
      if (!currentPoint)
        DumpAndQuit("Memory allocation failure");
      currentPoint->node = node;
      currentPoint->x = x;
      currentPoint->y = y;
      StorePoint(currentPoint,&FirstPoint,&LastPoint);
    }
  fclose(Fxy);
  if (!FirstPoint){
    sprintf(buf,"No valid input line found in file
%s",argv[1]);
    DumpAndQuit(buf);
  }

/*
  Read acceleration-data file.
  For each heading line found:
     extract node numbers from heading;
     get maximum acceleration data,
     stepping through file to end of current data block;
     and insert acceleration values into Points data structure.
*/

  for (; ;){
    if (!gets(buf))
      break;
    if (IsHeading(buf)){                        /* Found heading
*/
      numNodes = GetNodeNums(nodeNums,buf);
      if (!numNodes) DumpAndQuit("Heading-line error ... No
nodes found");
      else if (numNodes > maxNumNodes){
        sprintf(buf,
         "Max number of nodes allowed is %d, but %d was found",
         maxNumNodes,numNodes);
        DumpAndQuit(buf);
      }
      if (!GetMaxAcc(amax,numNodes))
        DumpAndQuit("No acceleration data found following a
heading line");
      for (i=0; i<numNodes; i++)
        if (!InsertAcc(FirstPoint,nodeNums[i],amax[i])){
          sprintf(buf,
           "Node %d in acceleration file not found in xy file",
           nodeNums[i]);
          DumpAndQuit(buf);
        }
    }
  }

/*  Print sorted data */

  PrintPoints(FirstPoint);
}

int GetMaxAcc(float *amax, int numNodes)
{
  char buf[121];
  int i,found;
  float a0,a1,a2,a3,time;

/*************************************************************
*******
  This code module reads the acceleration file to the end of
the
  current block (marked by a line that does not contain the
expected
  number of float variables). The acceleration values extracted
from
  the last valid input line are the maximum.
  The code provides for up to four acceleration values per line
*************************************************************
*******/

/* Read and discard the first two lines */

  found = 0;
  gets(buf);
  gets(buf);

  switch (numNodes){

    case 1:                 /* Case of 1 acceleration value per
line */
      for (; ;){
        if (!gets(buf) || (sscanf(buf,"%f %f",&time,&a0) != 2))
          return(found);

        amax[0] = a0;
        found = 1;
      }

    case 2:                 /* Case of 2 acceleration values per
line */
      for (; ;){
        if (!gets(buf) || (sscanf(buf,"%f %f %f",&time,&a0,&a1)
!= 3))
          return(found);

        amax[0] = a0;
        amax[1] = a1;
        found = 1;
      }

    case 3:                 /* Case of 3 acceleration values per
line */
      for (; ;){
        if (!gets(buf) ||
          (sscanf(buf,"%f %f %f %f",&time,&a0,&a1,&a2) != 4))
          return(found);

        amax[0] = a0;
        amax[1] = a1;
        amax[2] = a2;
        found = 1;
      }

    case 4:                 /* Case of 4 acceleration values per
line */
      for (; ;){
        if (!gets(buf) ||
          (sscanf(buf,"%f %f %f %f %f",&time,&a0,&a1,&a2,&a3)
!= 5))
          return(found);

        amax[0] = a0;
        amax[1] = a1;
        amax[2] = a2;
        amax[3] = a3;
        found = 1;
      }

    default:        /* Invalid number of nodes passed to
function */

      sprintf(buf,"Function GetMaxAcc cannot process %d nodes",
        numNodes);
      DumpAndQuit(buf);

  }
}

int GetNodeNums(int *nodeNums,char *buf)
{

  char c,*name="999999";
  int i,j,k,nodes,maxdigit;

  maxdigit = 6;
  k = 0;

/*************************************************************
***
  This code module extracts node numbers from the heading line.
  The maximum number of digits in a node number is maxdigit.
*************************************************************
**/

  nodes = 0;
  i=0;
  for (c=buf[0]; c!='\0'; c=buf[i]){
    if (!isdigit(c))
      ++i;
    else{
      j = 0;
      name[j] = c;
      for (; ;){
        c = buf[++i];
        if (isdigit(c)){
          if (++j == maxdigit)
            DumpAndQuit(
              "Maximum number of digits exceeded in function
GetNodeNums");
          name[j] = c;
        }
        else{           /* Collected last digit of current
number */
          name[++j] = '\0';
          nodeNums[k++] = atoi(name);
          nodes++;
          break;
        }
      }
    }
  }

  return(nodes);
}

StorePoint(struct Point *current,struct Point **first,struct
Point **last)
{
  struct Point *old,*p;

  p = *first;

  if (!*last){          /* current is the very first item in
the list */
    current->next = NULL;
    *last = current;
    *first = current;
    return;
  }

  old = NULL;
  while (p){
    if (IsBelow(current,p)){
      old = p;
      p = p->next;
    }
    else{
      if (old) {      /* insert current at this point in the
list */
        old->next = current;
        current->next = p;
        return;
      }
      current->next = p;    /* current becomes new first
element */
      *first = current;
      return;
    }
  }

  (*last)->next = current;  /* current becomes new last element
*/
  current->next = NULL;
  *last = current;
}

PrintPoints(struct Point *first)
{

  if (!first){
    printf("\t***\tNothing to print\t***\n");
    return;
  }

  printf("%10s%12s%12s%12s\n","Node","x (km)","Depth
(m)","maxAcc (g)");

  while (first){
    printf("%10d%12.2f%12.2f%12.4f\n",
      first->node,
      1.0e-3*first->x,
      12000. - first->y,
```

```
      (first->maxacc)/9.8
    );
    first = first->next;
  }
}

int InsertAcc(struct Point *first,int node,float amax)
{
  int found=1,notfound=0;

  while (first){
    if (node == first->node){
      first->maxacc = amax;
      return(found);
    }
    first = first->next;
  }
  return(notfound);
}

int IsBelow(struct Point *current,struct Point *p)
{
  int below,notBelow;
  float delta,xTol;

/*
   This code module determines whether first-named Point is
down-dip, i.e.,
   to the right, or up-dip (to the left) of second-named Point
*/

  below = 1;
  notBelow = 0;
  xTol = 0.05;

  delta = current->x - p->x;
  if (delta > xTol)
    return(below);

  return(notBelow);
}

int IsHeading(char *s)
{
  int i,isHead=1,notHead=0;
  char c;

  if ((s[0] != '*') || (s[1] != '*'))
    return(notHead);

  i = 2;
  for (c=s[2]; c!='\0'; c=s[i]){
    if (isspace(c))
      ++i;                          /* Discard white-space
characters */
    else if (c=='M' && s[i+1]=='A' && s[i+2]=='X')
      return(isHead);
    else
      return(notHead);
  }
  return(notHead);
}

DumpAndQuit(char *s)
{
  printf("\n *** %s ***\n\n",s);
  exit(0);
}
```

****

****

In addition to the four files generated by *max-accel.jnl*, files containing the node coordinates (ndtop.dat, nd300.dat) are necessary and can be generated from ABAQUS Post using the "print nodes, original" command. An executable file, *AccProfile.exe*, was compiled from *AccProfile.c*. A batch file, *accel-profile* was used to generate 4 output files that contain the node number, x coordinates (in km), y coordinate (depth in m), and maximum acceleration for that node (in g).

****

****

```
AccProfile.exe ndtop.dat < ndtop.a1 > ndtop.a1.out
AccProfile.exe ndtop.dat < ndtop.a2 > ndtop.a2.out
AccProfile.exe nd300.dat < nd300.a1 > nd300.a1.out
AccProfile.exe nd300.dat < nd300.a2 > nd300.a2.out
```

****

****

This is an example of a partial file containing the maximum horizontal accelerations for the surface nodes:

****

****

Node	x (km)	Depth (m)	maxAcc (g)
14367	-5.00	0.00	0.0000
14385	-4.20	0.00	0.0496
14420	-3.39	0.00	0.1046
14438	-2.78	0.00	0.1262
14473	-2.17	0.00	0.1271
14491	-1.71	0.00	0.1280
14526	-1.24	0.00	0.1302
14544	-0.89	0.00	0.1327
14579	-0.54	0.00	0.1297
14597	-0.27	0.00	0.1318
6806	0.00	0.00	0.1282
6841	0.28	0.00	0.1294
6859	0.56	0.00	0.1281
6894	0.84	0.00	0.1314
6912	1.12	0.00	0.1295
6947	1.40	0.00	0.1317
6965	1.68	0.00	0.1294
7000	1.96	0.00	0.1308

7018	2.24	0.00	0.1254
7053	2.52	0.00	0.1292
7071	2.80	0.00	0.1220
7106	3.08	0.00	0.1256
7124	3.36	0.00	0.1192
7159	3.64	0.00	0.1233
7177	3.92	0.00	0.1145
7212	4.20	0.00	0.1173
7230	4.48	0.00	0.1089
7265	4.76	0.00	0.1082
7283	5.04	0.00	0.0992
7318	5.32	0.00	0.0980
5488	5.60	0.00	0.0983
5487	5.95	0.00	0.1057
5486	6.29	0.00	0.1074
5485	6.56	0.00	0.1193
5484	6.82	0.00	0.1199
5483	7.02	0.00	0.1148
5482	7.22	0.00	0.1117
5481	7.37	0.00	0.1120
5480	7.52	0.00	0.1152
5479	7.63	0.00	0.1156
266	7.75	0.00	0.1119
248	7.80	0.00	0.1121
213	7.85	0.00	0.1104
195	7.90	0.00	0.1093
160	7.95	0.00	0.1083
142	8.00	0.00	0.1364
107	8.05	0.00	0.1586
89	8.10	0.00	0.1591
54	8.15	0.00	0.1564
36	8.20	0.00	0.1540
1	8.25	0.00	0.1523
2083	8.49	0.00	0.1510
2082	8.74	0.00	0.1550
2081	9.00	0.00	0.1632
2080	9.27	0.00	0.1782
2079	9.55	0.00	0.1889
2078	9.84	0.00	0.1944
2077	10.15	0.00	0.2096
2076	10.46	0.00	0.2094
2075	10.80	0.00	0.1900
2074	11.13	0.00	0.1558
2073	11.50	0.00	0.1288
2072	11.87	0.00	0.0922
2071	12.20	0.00	0.0795
2070	12.54	0.00	0.0961
2069	12.85	0.00	0.1464

****

****

12.3 Model Combinations Analyzed

Once the new acceleration monitoring procedure was established, a series of different models were analyzed by varying the source depth (shallow, intermediate or deep on the steep segment of fault F0) as well as the source size. Source size is controlled by the length of fault segment over which the shear stress pulse (refer to section 8 of notebook 183) is applied. For the intermediate and deep sources, the source size was varied from 0.5 km to 1 km to 2 km. The table below summarizes each of the models and gives the file names that were used to generate each of the runs. All files are archived as required for Quality Assurance purposes. As noted earlier, all runs make use of the file *m09Base.res* as the starting point.

File Name	Source Depth	Source Size (km)
sc13	Shallow	2.0
sc09	Intermediate	2.0
sc10	Intermediate	1.0
sc11	Intermediate	0.5
sc06	Deep	2.0
sc07	Deep	1.0
sc08	Deep	0.5

12.4 Additional Model with Smaller Shear Stress Pulse

To better evaluate the effect of the shear stress pulse that simulates the earthquake rupture, model sc06 was re-analyzed with one addition. The *AMPLITUDE command shown below was added to the sc06 file so that the peak of the applied shear stress pulse goes up to 1.10 rather than 1.25. The duration of the shear stress pulse remains 0.5 seconds.

```
**********************************************************************
**********************************************************************
      ***
      ***
      *AMPLITUDE,NAME=NEWLOAD,TIME=STEP TIME,DEF=TABULAR
      0.0,0.0,0.25,1.10,0.5,0.95,5.0,0.95
      ***
      ***
**********************************************************************
**********************************************************************
```

Preliminary analysis of the results suggests that overall ground accelerations are somewhat smaller but displacement on fault F0 (surface, maximum and average) is unchanged.

12.5 Additional Model without Antithetic Normal Faults

To better evaluate the effect of the two antithetic normal faults (F1 and F2) on the near-ground accelerations, model sc06 was re-analysed using *m10Base.res*. The only difference between *m10Base.res* and *m09Base.res* is that the material description for Faults F1 and F2 were changed from "RUBBLE" to "LEROCK". This effectively makes the hangingwall one homogeneous linear elastic material.

1997 May 13

Acceleration - Contour Plots

Acceleration-contour plots representing the distributions of vertical and horizontal accelerations are being extracted from the results for the following cases:

Case ID	Job Name	Job Step	Time from end of Static Analysis Step (s)
DS20	sc06	4	1.0
		6	3.0
		8	5.0
DS10	sc07	4	1.0
	sc07b	7	3.0
		9	5.0
DS05	sc08	4	3.0
		6	5.0
IS20	sc09	3	1.0
		5	2.5
		7	4.5
IS10	sc10	3	1.0
		5	3.0
		7	5.0
IS05	sc11	3	1.0
		5	3.0
		7	5.0
SS20	sc13	3	1.0
		5	3.0

In the above table, column "Case ID" is assigned value DSnn, ISnn or SSnn, where DS = Deep Source, IS = Intermediate Source, and SS = Shallow source; and number nn is 20, 10, or 05, representing source sizes of 2.0, 1.0 or 0.5 km respectively (Also see table at top of p. 33).

Contours of vertical and horizontal acceleration are plotted by reading the following file into an ABAQUS/POST session, after issuing the command

set, hard = HardCopyFileName.

GTD

```
set, fill=on
set, c level=8
set, c max =2
set, c min = -2
set, c legend size=0.25
set, c legend=(0.79, 0.99)
set, outline=off

restart,step=4
cont,v=a1
pause
cont,v=a2
pause
restart,step=6
cont,v=a1
pause
cont,v=a2
pause
restart,step=8
cont,v=a1
pause
cont,v=a2
```

Modified according to "Job Step" column in table on previous page.

Thereafter the Hardcopy File is processed using ABAQUS/PLOT to obtain postscript files ~~A~~.

Postscript files of model geometry showing appropriate source locations are superimposed on the contour plots using C code AddFaultGeom.c (reproduced on p. 36). The ~~fo~~ source-geometry plots are read from the following files

Case ID	Name of source-geometry file
DS20	geomDs20.ps
DS10	geomDs10.ps
DS05	geomDs05.ps
IS20	geomIS(GTD)20.ps
IS10	geomIs10.ps
IS05	geomIs05.ps
SS20	geomSs20.ps

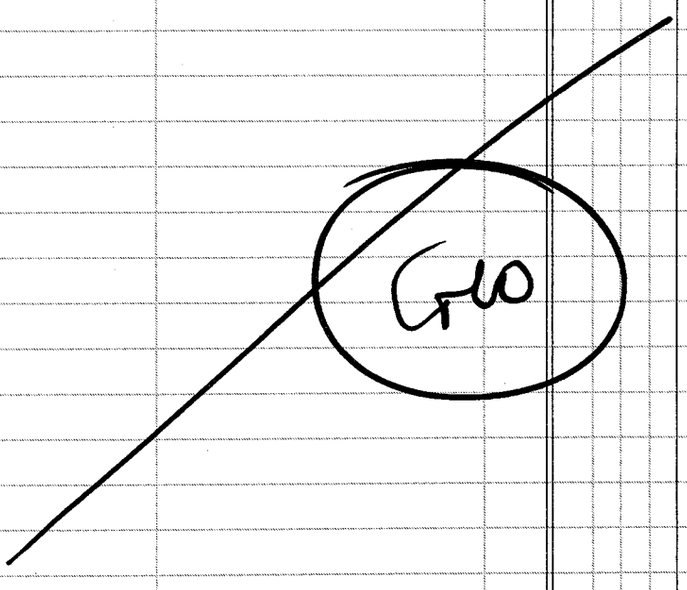
```
/*************************************************************************
  AddFaultGeom.c

  This code superimposes a color PostScript plot of fault geometry (showing
  induced-earthquake source) on a color PostScript contour-plot of calculated
  results (such as ground acceleration).

  The following variable file names should be assigned appropriate values
  before compiling the code.

  contourFile                Full name of file containing results plot
  geomFile                   Full name of fault-geometry file

  The combined plot (color PostScript) will be dumped onto the screen unless
  redirected to a user-named file to be displayed on a PostScript device

  Author:              G.I. Ofoegbu
  Date:                July 01 1996
  Revised:             May 13 1996
                       user is now prompted to supply names for
                       contourFile, geomFile, and outFile.
*************************************************************************/

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

int PenNumber(char* s);
int GetFileName(char* name, char* prompt);

main()
{

  char buf[151];
  char *contourFile="0123456789012345",
       *geomFile="0123456789012345",
       *outFile="0123456789012345";
  int pen;
  FILE *Fin, *Fout;

/* Get file names */

  if (!GetFileName(contourFile,"Name of contour file") ||
    !GetFileName(geomFile,"Name of geometry file"))
      return;

  if (!GetFileName(outFile,"Name of output file")) outFile = NULL;

  Fin = fopen(contourFile,"r");
  if (!Fin) DumpAndQuit(strcat("Unable to open file ", contourFile));
  Fout = fopen(outFile,"w");
  if (!Fout) DumpAndQuit(strcat("Unable to open output file ", outFile));

  while (fgets(buf,150,Fin))
    if (!strncmp(buf,"ep",2)) break;
    else if (strncmp(buf,"logo",4))
      fputs(buf,Fout);

  fclose(Fin);

  Fin = fopen(geomFile,"r");
  if (!Fin) DumpAndQuit(strcat("Unable to open file ", geomFile));

  for(;;){
    if (!fgets(buf,150,Fin))
      break;
    if (!strncmp(buf,"ep",2)){ /* Found ep; copy to end-of-file and exit */
      fputs(buf,Fout);
      while (fgets(buf,150,Fin)) fputs(buf,Fout);
      break;
    }
    if (!strncmp(buf,"/pen",4)) fputs(buf,Fout);
    else if (!strncmp(buf,"pen",3) && (pen=PenNumber(&buf[3]))<=4 && pen!=3){
      fputs(buf,Fout);
      while(fgets(buf,150,Fin)){
        if (!strncmp(buf,"pen",3) && ((pen=PenNumber(&buf[3]))>4 || pen==3))
          break;
        else fputs(buf,Fout);
      }
    }
  }
}

int PenNumber(char* s)
{
  char pen[3];
  int i;

  if (!isdigit(s[0])) return(20);
  pen[0] = s[0];
  if (isdigit(s[1])) pen[1]=s[1];
  else pen[1] = '\0';

  pen[2] = '\0';
  return(atoi(&pen[0]));
}

int GetFileName(char *name, char *prompt)
{
  printf("%s (15 char max) or return to quit: ",prompt);
  if (!gets(name) || strlen(name)==0)
    return(0);

  return(1);
}

DumpAndQuit(s)
char *s;
{
  printf("\n *** %s ***\n\n",s);
  exit(0);
}
```

June 19 1997

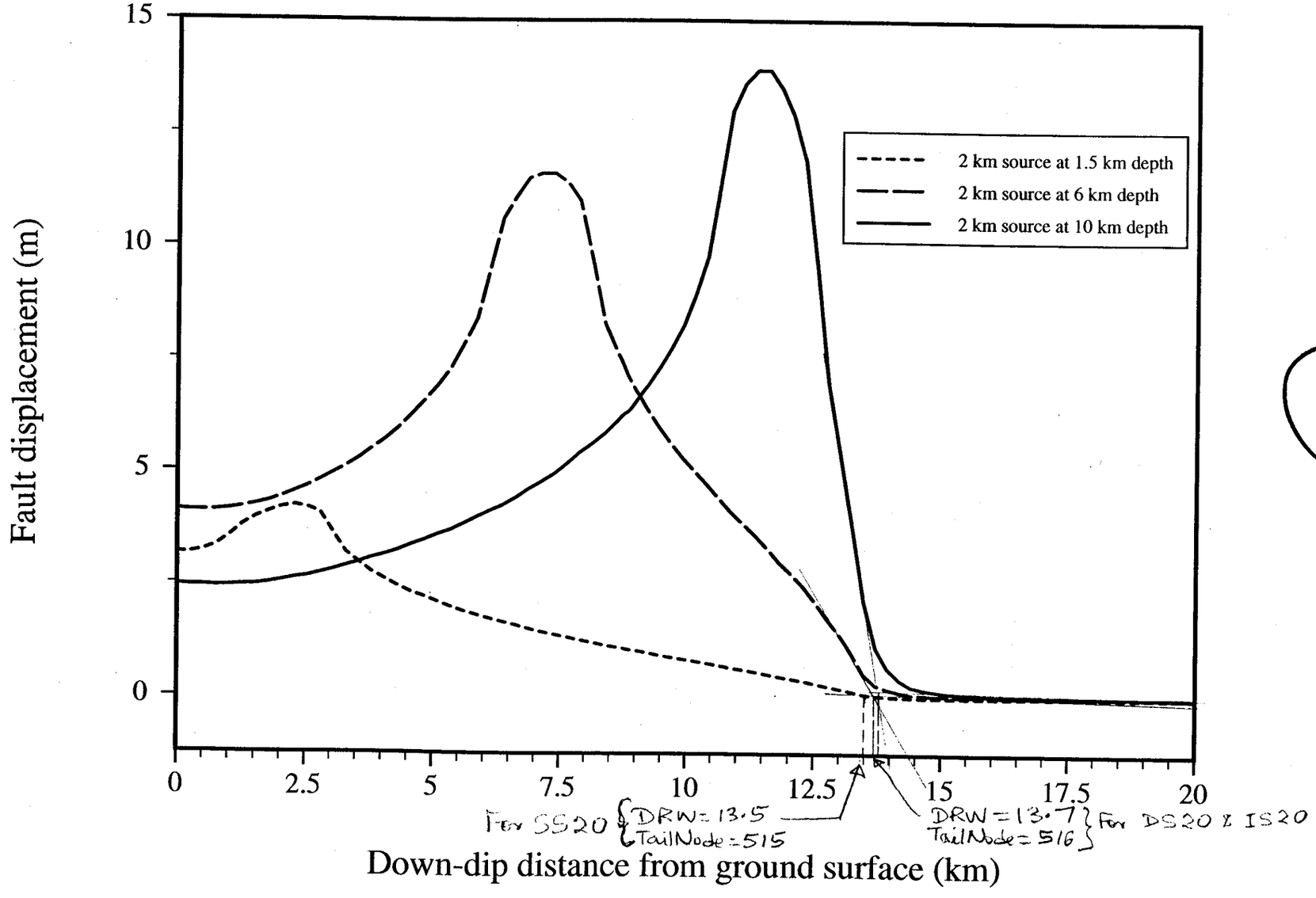
Evaluation of Down-Dip Rupture Width

The method used earlier to evaluate down-dip rupture width (DRW), based on cut-off fault displacement of 0.0, 0.5, or 1.0 cm (see p. 142 of Notebook #183), is difficult to apply consistently because of erratic variation of fault displacement along the detachment segment of FO. Fault displacement is on the order of m and 10 m along the steep segment of the fault, but decreases sharply to order of ~~cm~~ GLO mm in the detachment segment. It is believed, based on the result of previous studies, that the small magnitudes of fault slip in the detachment segment most likely occurred aseismically ~~whereas slip in~~ GLO in response to seismic slip on the steep segments. Consequently, rupture contributions from the detachment segment should not be included in the evaluation of fault-rupture parameters for seismic-risk assessment.

As a result, it was decided that the tip of seismic rupture ~~should be taken to~~ GLO corresponds to the sharp break in the fault-displacement profile, i.e., the point on the fault where the fault displacement profile changes from a steep curve to an essentially flat curve (see three figures on p. 38-40).

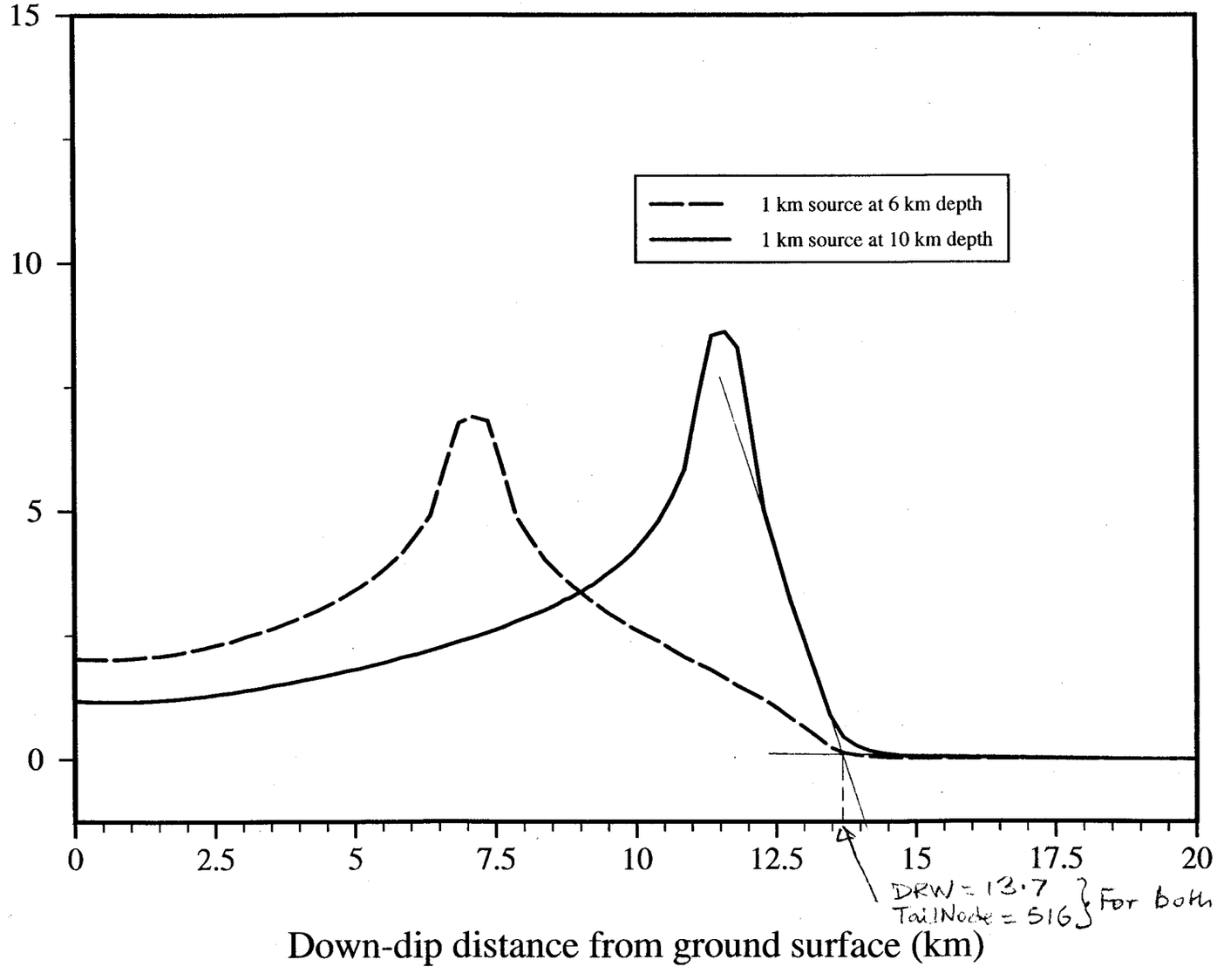


Tip Line Analyses for 2 km Sources

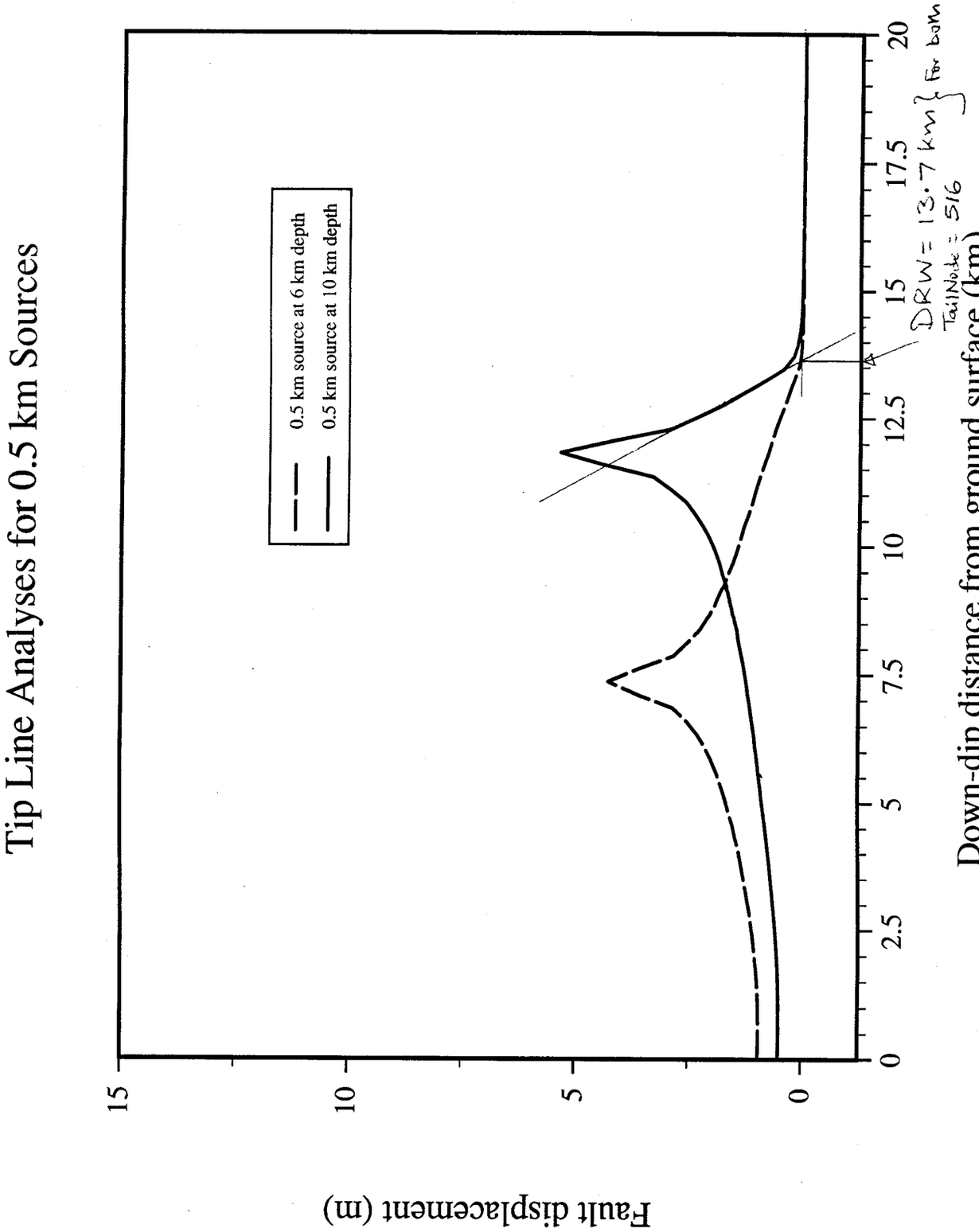


GW0

Tip Line Analyses for 1 km Sources



GW



The node closest to the * seismic-rupture tip determined as illustrated on pages 38, 39, and 40 was used in the calculation of average fault displacement and slip energy. The calculation procedures are documented in the C codes on pages 41-44.

Average Fault Displacement (See bottom of P. 40 for explanation)

```
/*****************************************************************************
AvgFaultDisp.c

This code calculates the average fault displacement along a fault,
as integral of down-dip fault-displacement profile (trapezoidal rule)
divided by down-dip rupture width. It uses output of code FaultDisp.c
stored in a file scNNf0disp.dat where NN is a user-supplied integer
(modelNumber) specified as command-line argument to run the code.

Author:                         G. I. Ofoegbu
Date:                           July 25 1996
System Requirement:             ANSI C under UNIX
                                Compile with acc on SUN OS (cc will fail)

Revised:                        January 10 1997
Revised:                        June 18 1997

                                User required to supply tailNode as second
                                command-line argument, and is prompted to
                                confirm input-file name and tailNode
                                interpreted by the code.
*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Point{
  int node;
  float fdisp,flen;
  struct Point *next;
};

void DumpAndQuit(char* s);

main(int argc, char* argv[])
{

  char buf[81],cnode[5],*fileName="scNNf0disp.dat";
  int node,tailNode,modelNumber;
  float fdisp,flen,avgDisp;
  float u0,u1,d0,d1,area;
  float zeroDisp=5.0e-3;
  FILE *Fin;
  struct Point *firstPoint,*lastPoint,*p;

/* Process command-line argument for input-file name and tailNode */

  if (argc < 3)
    DumpAndQuit("Usage: ProgramName modelNumber tailNode");

  modelNumber = atoi(argv[1]);
  sprintf(fileName,"sc%02df0disp.dat",modelNumber);
  Fin = fopen(fileName,"r");
  if (!Fin)
    DumpAndQuit(strcat("Unable to open file ",fileName));
  if (!(tailNode = atoi(argv[2])))
    DumpAndQuit(strcat("Tail node not found in string ",argv[2]));
  printf("Input file = %s\n",fileName);
  printf("Tail node = %d\n",tailNode);
  printf("Enter 'y' to confirm or anything else to quit: ");
  gets(buf);
  if (buf[0] != 'y')
    return;

  firstPoint = lastPoint = NULL;
  while (fgets(buf,80,Fin))
    if (sscanf(buf,"%d %*f %f %*f %f",&node,&flen,&fdisp) == 3){
      p = (struct Point *)malloc(sizeof(struct Point));
      if (!p) DumpAndQuit("Memory allocation error");
      p->node = node;
      p->flen = flen;
      p->fdisp = fdisp;
      p->next = NULL;
      if (!firstPoint) firstPoint = p;     /* First point in line */
      else if (!lastPoint){                /* Second point in line */
        lastPoint = p;
        firstPoint->next = p;
      }
      else{                          /* Attach current point to end of line */
        lastPoint->next = p;
        lastPoint = p;
      }
    }

  if (!firstPoint){
    sprintf(buf,"No valid input line found in file %s", fileName);
    DumpAndQuit(buf);
  }

/*****************************************************************************
            This section modified Jan 10 1997
            *********************************
  for (;;){
    printf("Press return to end or Enter new tailNode: ");
    gets(cnode);
    if (!(tailNode = atoi(cnode))) break;

    p = firstPoint;
    numNodes = 0;
    totalDisp = 0.0;
    area = 0.0;
    u0 = p->fdisp;
    d0 = p->flen;

    while (p){
      numNodes++;
      totalDisp += p->fdisp;
      u1 = p->fdisp;
      d1 = p->flen;
      area += (d1-d0)*(u0+u1)/2.0;
      if (p->node == tailNode){
        avgDisp = totalDisp/numNodes;
        printf("\n\t Avg fault disp by arithmetic mean = %.2f m\n\n",avgDisp);
        avgDisp = area/d1;
        printf("\n\t Avg fault disp by integration = %.2f m\n\n",avgDisp);
        break;
      }
      u0 = u1;
      d0 = d1;
      p = p->next;
```

```
    }
    if (!p)
      printf("\n\tNode %d was not found in file %s\n\n",tailNode,fileName);
  }
*********************************************************************/

  p = firstPoint;
  area = 0.0;
  u0 = p->fdisp;
  d0 = p->flen;

  while (p){
    u1 = p->fdisp;
    d1 = p->flen;
    area += (d1-d0)*(u0+u1)/2.0;
    if ((p->node) == tailNode){ /* Stop when tailNode is encountered */
      avgDisp = area/d1;
      printf("\t Avg D_f = %.2f m with tailNode %d from file  %s\n",
        avgDisp,tailNode,fileName);
      break;
    }
    u0 = u1;
    d0 = d1;
    p = p->next;
  }
  if (!p)
    printf(
      "\n\tTail node %d not found in %s \n\n",tailNode,fileName);
}

void DumpAndQuit(char *s)
{
  printf("\n *** %s ***\n\n",s);
  exit(0);
}
```

```
/*************************************************************************
SlipEnergy.c

This code computes the mechanical energy (MJ) required to generate
the fault-displacement distribution calculated for any of the
Bare Mountain Fault test models. It uses output of code
FaultDisp.c, stored in a file scNNf0disp.dat where NN is a
user-supplied integer (modelNumber) specified as command-line
argument to run the code.

Author:                         G. I. Ofoegbu
Date:                           January 07 1997
System Requirement:             ANSI C under UNIX
                                Compile with acc on SUN OS (cc will fail)

Revised:                        June 18 1997

                                User required to supply variable tailNode
                                as second command-line argument, and is
                                prompted to confirm input-file name and
                                tailNode interpreted by the code.
                                Variable tailNode is used to identify
                                fault tip
*************************************************************************/

#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

float ShearStress(float dip, float kmDepth);
void DumpAndQuit(char* s);

main(int argc, char* argv[])
{
  char buf[81],*fileName="scNNf0disp.dat";
  int node,modelNumber,tailNode;
  float depth,fLen,dip,fDisp;
  float u1,u2,w1,w2,tau1,tau2;
  float energy;
/*  float zeroSlip=5.0e-3;  No longer used, as of June 18 1997 */
  FILE *Fin;

/*
   Process command-line argument for to determine input file name
   and tailNode
*/

  if (argc < 3)
    DumpAndQuit("Usage: ProgramName modelNumber tailNode");

  modelNumber = atoi(argv[1]);
  sprintf(fileName,"sc%02df0disp.dat",modelNumber);
  Fin = fopen(fileName,"r");
  if (!Fin)
    DumpAndQuit(strcat("Unable to open file ",fileName));
  if (!(tailNode = atoi(argv[2])))
    DumpAndQuit(strcat("Tail node not found in string ",argv[2]));
  printf("Input file = %s\n",fileName);
  printf("Tail node = %d\n",tailNode);
  printf("Enter 'y' to confirm or anything else to quit: ");
  gets(buf);
  if (buf[0] != 'y')
    return;

  fgets(buf,80,Fin);    /* Read and discard one-line heading */

  if (!fgets(buf,80,Fin) ||
    sscanf(buf,"%d %f %f %f %f",&node,&depth,&fLen,&dip,&fDisp) != 5)
      DumpAndQuit(strcat("Error reading 1st data line in ",fileName));
  u1 = fDisp;
  w1 = fLen*1000.;    /* Convert down-dip width from km to m */
  tau1 = ShearStress(dip,-depth);  /* convert depth to positive value
                                      and calculate shear strength */

  energy = 0.0;
  while (fgets(buf,80,Fin)){
    if (sscanf(buf,"%d %f %f %f %f",&node,&depth,&fLen,&dip,&fDisp) != 5)
      DumpAndQuit(strcat("Invalid input line read from ",fileName));

/********************** Following option removed on June 18 1997 ***
    if (fDisp < zeroSlip)
      break;
*************************************************************************/
    u2 = fDisp;
    w2 = fLen*1000.;    /* Convert down-dip width from km to m */
    tau2 = ShearStress(dip,-depth);  /* convert depth to positive value
                                        and calculate shear strength */
    energy += (0.5*(tau1*u1 + tau2*u2)*(w2 - w1));

/* Next two code lines added on June 18 1997  */

    if (node == tailNode)
      break;

    u1 = u2;
    w1 = w2;
    tau1 = tau2;
  }

  printf("Slip energy = %12.3E MJ\t from file %s; tailNode %d\n",
    energy,fileName,tailNode);
}


float ShearStress(float dip, float kmDepth)
{
  float pi,alpha,ca,sa,nStress;
  float sGradkm=25.0,fricAngle=47.0,hsigRatio=0.25,cohesion=2.7;

/*
  This code module computes shear stress for incipient slip
  at depth kmDepth in km on a surface with local dip of dip
  based on Mohr-Coulomb slip criterion.
  The friction angle for the surface is fricAngle,
  the vertical-stress gradient is sGradkm per km depth,
  and ratio of horizontal to vertical stress is hsigRatio.
*/

  pi = 4.0*atan(1.0);
  alpha = pi*(0.5 + dip/180.0);
```

Slip Energy (See p. 40 for explanation)

GVO

SlipEnergy.c p. 2 of 2

Listing for Goodluck Ofoegbu

```
  ca = cos(alpha);
  sa = sin(alpha);
  nStress = sGradkm*(hsigRatio*ca*ca + sa*sa)*kmDepth;
  return(cohesion + nStress*tan(pi*fricAngle/180.0));
}

void DumpAndQuit(char *s)
{
  printf("\n *** %s ***\n\n",s);
  exit(0);
}
```

GVO

Values of down-dip rupture width (and associated tip node), average fault displacement and slip energy that resulted from the procedures described on pages 37-44 are summarized below:

Case ID	File #	Tip Node	DRW (km)	Davg (m)	Energy (10^6 MJ/m)
SS20	Sc13	515	13.5	1.88	1.162
* IS05	Sc11	516	13.7	1.48	1.428
IS10	Sc10	516	13.7	2.91	2.736
IS20	Sc02	516	13.7	5.76	5.421
DS05	Sc08	516	13.7	1.38	1.791
DS10	Sc07	516	13.7	2.84	3.607
DS20	Sc03	516	13.7	5.59	7.094

It is worth noting that a DRW of 13.7 km on a 60° fault corresponds to a rupture depth of 11.86 km. The detachment segments of fault FO occur at 12 km depth, which implies that the entire steep segment of the fault ruptured seismically, but seismic rupture was arrested by the detachment.

Moment Magnitude of Simulated Earthquake

These rupture parameters were used to estimate earthquake magnitudes, using the C code documented below.

GLO

Listing for Goodluck Ofoegbu

```
/*****************************************************************
MagEstimates.c

The following code calculates earthquake magnitudes from
empirical formulas, hard-coded values of surface fault displacement,
average fault displacement, down-dip rupture width, and ratio
of rupture width to rupture length.

Author:                         G. I. Ofoegbu
Date:                           January 10 1997
System Requirement:             ANSI C under UNIX
                                Compile with acc on SUN OS (cc will fail)

Revised:                        June 18 1997

                                Fault-rupture parameters re-defined
                                as described below
*****************************************************************/

#include <stdio.h>
#include <math.h>

main()
{
  char *model[]={"SS20","IS05","IS10","IS20","DS05","DS10","DS20"};
  int i,numModels=7;
  float sDisp[]={3.16,0.95,2.03,4.12,0.50,1.18,2.39};

/*****************************************************************
  Values for averageDisplacement and downDipRuptureWidth modified
  on June 18 1997 because of change in method of determining location
  of fault tip.

  float aDisp[]={1.32,1.06,1.80,2.94,0.99,1.77,2.93};
  float wd[]={19.25,19.25,22.22,26.94,19.25,22.22,26.44};

  New values assigned in next two code lines
*****************************************************************/

  float aDisp[]={1.88,1.48,2.91,5.76,1.38,2.84,5.59};
  float wd[]={13.5,13.7,13.7,13.7,13.7,13.7,13.7};

  float minRatio=0.8,maxRatio=2.5;
  float m2cm=100.0,km2cm=1.0e5;
  float surfaceDisp,avgDisp,width,area;
  float smod=1.3e11;
  float magnitude;

  for (i=0; i<numModels; i++){
    surfaceDisp = sDisp[i];
    avgDisp = m2cm * aDisp[i];      /* Convert avg disp to cm */
    width = wd[i];
    printf("For %s\n",model[i]);

/* Surface-displacement formula */

    magnitude = 6.61 + 0.71*log10(surfaceDisp);
    printf("\tM = %.2f from surface-displacement formula\n",magnitude);

/* Rupture-width formula  */
```



```
    magnitude = 4.04 + 2.11*log10(width);
    printf("\tM = %.2f from rupture-width formula\n",magnitude);

/* Seismic-moment formula with min L/W ratio */

    area = minRatio*width*width*km2cm*km2cm;
    magnitude = (2./3.)*log10(smod*avgDisp*area) - 10.7;
    printf("\tM = %.2f from seismic-moment formula with L/W=%.1f\n",
      magnitude,minRatio);

/* Seismic-moment formula with max L/W ratio */

    area = maxRatio*width*width*km2cm*km2cm;
    magnitude = (2./3.)*log10(smod*avgDisp*area) - 10.7;
    printf("\tM = %.2f from seismic-moment formula with L/W=%.1f\n\n\n",
      magnitude,maxRatio);
  }
}
```

Output file from code MagEstimates.c (p. 45)

For SS20
M = 6.96 from surface-displacement formula
M = 6.43 from rupture-width formula
M = 6.33 from seismic-moment formula with L/W=0.8
M = 6.66 from seismic-moment formula with L/W=2.5

For IS05
M = 6.59 from surface-displacement formula
M = 6.44 from rupture-width formula
M = 6.27 from seismic-moment formula with L/W=0.8
M = 6.60 from seismic-moment formula with L/W=2.5

GW

For IS10
M = 6.83 from surface-displacement formula
M = 6.44 from rupture-width formula
M = 6.47 from seismic-moment formula with L/W=0.8
M = 6.80 from seismic-moment formula with L/W=2.5

For IS20
M = 7.05 from surface-displacement formula
M = 6.44 from rupture-width formula
M = 6.67 from seismic-moment formula with L/W=0.8
M = 7.00 from seismic-moment formula with L/W=2.5

For DS05
M = 6.40 from surface-displacement formula
M = 6.44 from rupture-width formula
M = 6.25 from seismic-moment formula with L/W=0.8
M = 6.58 from seismic-moment formula with L/W=2.5

For DS10
M = 6.66 from surface-displacement formula
M = 6.44 from rupture-width formula
M = 6.46 from seismic-moment formula with L/W=0.8
M = 6.79 from seismic-moment formula with L/W=2.5

For DS20
M = 6.88 from surface-displacement formula
M = 6.44 from rupture-width formula
M = 6.66 from seismic-moment formula with L/W=0.8
M = 6.99 from seismic-moment formula with L/W=2.5

June 25 1997

Acceleration Histories.

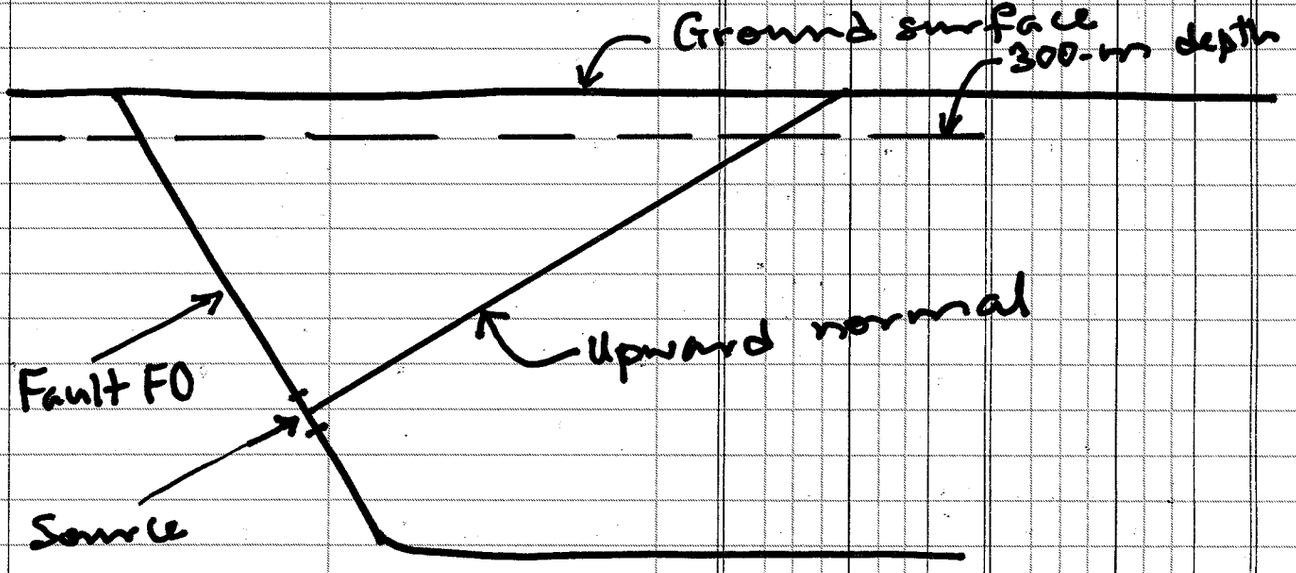
Acceleration histories were monitored at following points (nodes):

(1) Nodes on the ground surface (top of model);

(2) Nodes at depth of about 300 m (actually ~~300 m~~ 210 – 380 m) below the ground (GW) surface;

(3) Nodes along the main fault (F0) on the hangingwall surface; and

(4) Nodes on the upward normal to the fault through the source



The acceleration-history data are contained in the following files named jobName.fil, ~~LAT~~ where the value of jobName is defined in table on page 48 and .fil indicates that each file is an ABAQUS results file in binary format. The .fil files were read using ABAQUS/POST batch jobs described subsequently.

Table of .fil (ABAQUS result) files containing nodal acceleration histories for different analysis cases

Case ID	jobName	Ground surface	300-m depth	fault surface	Upward normal
SS20	sc13	Yes	Yes	No	No
IS05	sc11	Yes	Yes	No	No
IS10	sc10	Yes	Yes	No	No
IS20	sc02	Yes	Yes	No	No
IS20	sc29	Yes	Yes	Yes	Yes
DS05	sc08	Yes	Yes	No	No
DS05	sc28	Yes	Yes	Yes	Yes
DS10	sc07	Yes	Yes	No	No
DS10	sc27	Yes	Yes	Yes	Yes
DS20	sc06	Yes	Yes	No	No
DS20	sc26	Yes	Yes	Yes	Yes

Each of the files has a companion jobName.res file (also an ABAQUS binary file)

Accelerations histories stored in these files were extracted using ABAQUS/POST batch jobs, submitted in series through the C code reproduced on p. 49. One of the inputs required by the C-code (See p. 49) is a file name for a file that contains the following information:

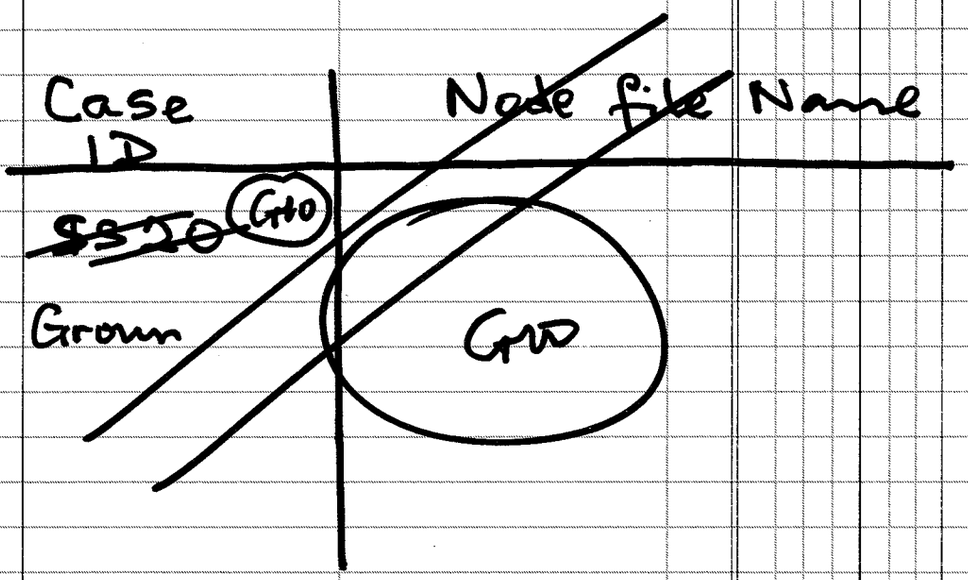
Node
X-coordinate
Y-coordinate
Distance: from source along normal,
from source along fault, or
from left boundary of model.

```
/*****************************************************************
AccHitories.c

This code extracts vertical and horizontal acceleration histories
for a series of nodes from ABAQUS results file. User
is prompted to supply name of file containing node definitions
(node number, x- and y-coordinates, and along-normal or up-dip distance
from source, as well as the names of ABAQUS restart and results files
(.res and .fil files). Name of output file for the acceleration
histories is calculated using source distance appended to a 3-character
string supplied by user.
*****************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void MakeCommandFile(int node, char* outFile, char* resultFile);
int GetFileName(char* name, char* prompt);
void DumpAndQuit(char* s);

main()
{
  char command[81],buf[81];
  char *restartFile="0123456789012345",*resultFile="0123456789012345";
  char *outFile="0123456789012345",*namePrefix="012345";
  char *nodeFile="0123456789012345";
  int node,nx,ny;
  float x,y,rd;
  FILE *Fnode;

/* Get file names for ABAQUS restart and results files */

  if (!GetFileName(restartFile,"Restart file, no extension (15 char max)") ||
    !GetFileName(resultFile,"Results file, no extension (15 char max)") ||
    !GetFileName(namePrefix,"Output file prefix (5 char max)") ||
    !GetFileName(nodeFile,"Node-definition file (15 char max)"))
      return;

  Fnode = fopen(nodeFile,"r");
  if (!Fnode)
    DumpAndQuit(strcat("Unable to open file ",nodeFile));

/***************************************
  For each node:
    Setup ABAQUS/POST command file;
    submit ABAQUS/POST job;
***************************************/

  while (fgets(buf,80,Fnode))
    if (sscanf(buf,"%d %f %f %f",&node,&x,&y,&rd) == 4){
      nx = rd;
      ny = (rd-nx)*1000;
      sprintf(outFile,"%s%02d.%03d",namePrefix,nx,ny);
      MakeCommandFile(node,outFile,resultFile);
      sprintf(command,"abaqus56 post restart=%s input=abapost.jnl",restartFile);
      system(command);
    }
}
```

or horizontal distance from left boundary of model.


```
void MakeCommandFile(int node, char *outFile, char *resultFile)
{
  FILE *Fjn;

  Fjn = fopen("abapost.jnl","w");
    if (!Fjn) DumpAndQuit("Unable to setup abapost.jnl file");

  fprintf(Fjn,"result,file=%s\n",resultFile);
  fprintf(Fjn,"set,reportfile=%s\n", outFile);
  fprintf(Fjn,"readcurve,name=xa,var=a1,node=%d\n",node);
  fprintf(Fjn,"readcurve,name=ya,var=a2,node=%d\n",node);
  fprintf(Fjn,"definecurve,name=xa2,operation=multiply\n");
  fprintf(Fjn,"xa\n");
  fprintf(Fjn,"xa\n");
  fprintf(Fjn,"definecurve,name=ya2,operation=multiply\n");
  fprintf(Fjn,"ya\n");
  fprintf(Fjn,"ya\n");
  fprintf(Fjn,"definecurve,name=am2,operation=add\n");
  fprintf(Fjn,"xa2\n");
  fprintf(Fjn,"ya2\n");
  fprintf(Fjn,"definecurve,name=accmag,operation=squareroot\n");
  fprintf(Fjn,"am2\n");
  fprintf(Fjn,"printcurve\n");
  fprintf(Fjn,"xa,ya,accmag\n\n");
  fclose(Fjn);

  return;
}

int GetFileName(char *name, char *prompt)
{
  printf("%s or return to quit: ",prompt);
  if (!gets(name) || strlen(name)==0)
    return(0);

  return(1);
}

void DumpAndQuit(char *s)
{
  printf("\n *** %s ***\n\n",s);
  exit(0);
}
```

Names of such node files are listed below



Node list	File Name
Ground-surface nodes	ndtop.xyd
300-m-depth nodes	nd300.xyd
(DS) DS05 upward normal	ds05Normal.xyd
(DS) DS05 along fault surface	ds05f0hw.xyd
(DS) DS20 upward normal } DS10 upward normal }	ds20Normal.xyd
(DS) DS20 along fault surface } DS10 along fault surface }	ds20f0hw.xyd
IS20 along fault surface	is20f0hw.xyd
IS20 upward normal	is20Normal.xyd

Extracts from ~~An example~~ two example node files are reproduced on page 51.

Extract from ndtop.xyd

Node	x (km)	y (km)	Distance (km)
14367	-5.0000	12.0000	0.0000
14385	-4.1960	12.0000	0.8040
14420	-3.3920	12.0000	1.6080
14438	-2.7810	12.0000	2.2190
14473	-2.1700	12.0000	2.8300
14491	-1.7060	12.0000	3.2940
14526	-1.2420	12.0000	3.7580
14544	-0.8890	12.0000	4.1110
14579	-0.5360	12.0000	4.4640
14597	-0.2680	12.0000	4.7320
6806	0.0000	12.0000	5.0000
6841	0.2800	12.0000	5.2800
6859	0.5600	12.0000	5.5600
6894	0.8400	12.0000	5.8400
6912	1.1210	12.0000	6.1210
6947	1.4010	12.0000	6.4010
6965	1.6810	12.0000	6.6810
7000	1.9610	12.0000	6.9610
7018	2.2410	12.0000	7.2410
7053	2.5210	12.0000	7.5210
7071	2.8010	12.0000	7.8010
7106	3.0820	12.0000	8.0820
7124	3.3620	12.0000	8.3620
7159	3.6420	12.0000	8.6420
7177	3.9220	12.0000	8.9220
7212	4.2020	12.0000	9.2020
7230	4.4820	12.0000	9.4820
7265	4.7630	12.0000	9.7630
7283	5.0430	12.0000	10.0430
7318	5.3230	12.0000	10.3230
5488	5.6030	12.0000	10.6030
5487	5.9480	12.0000	10.9480
5486	6.2940	12.0000	11.2940
5485	6.5560	12.0000	11.5560
5484	6.8180	12.0000	11.8180
5483	7.0180	12.0000	12.0180
5482	7.2170	12.0000	12.2170
5481	7.3680	12.0000	12.3680
5480	7.5200	12.0000	12.5200
5479	7.6350	12.0000	12.6350
266	7.7500	12.0000	12.7500
248	7.8000	12.0000	12.8000
213	7.8500	12.0000	12.8500
195	7.9000	12.0000	12.9000
160	7.9500	12.0000	12.9500
142	8.0000	12.0000	13.0000
107	8.0500	12.0000	13.0500
89	8.1000	12.0000	13.1000
54	8.1500	12.0000	13.1500
36	8.2000	12.0000	13.2000
1	8.2500	12.0000	13.2500
2083	8.4940	12.0000	13.4940
2082	8.7370	12.0000	13.7370
2081	9.0020	12.0000	14.0020
2080	9.2660	12.0000	14.2660
2079	9.5520	12.0000	14.5520
2078	9.8390	12.0000	14.8390
2077	10.1500	12.0000	15.1500
2076	10.4600	12.0000	15.4600

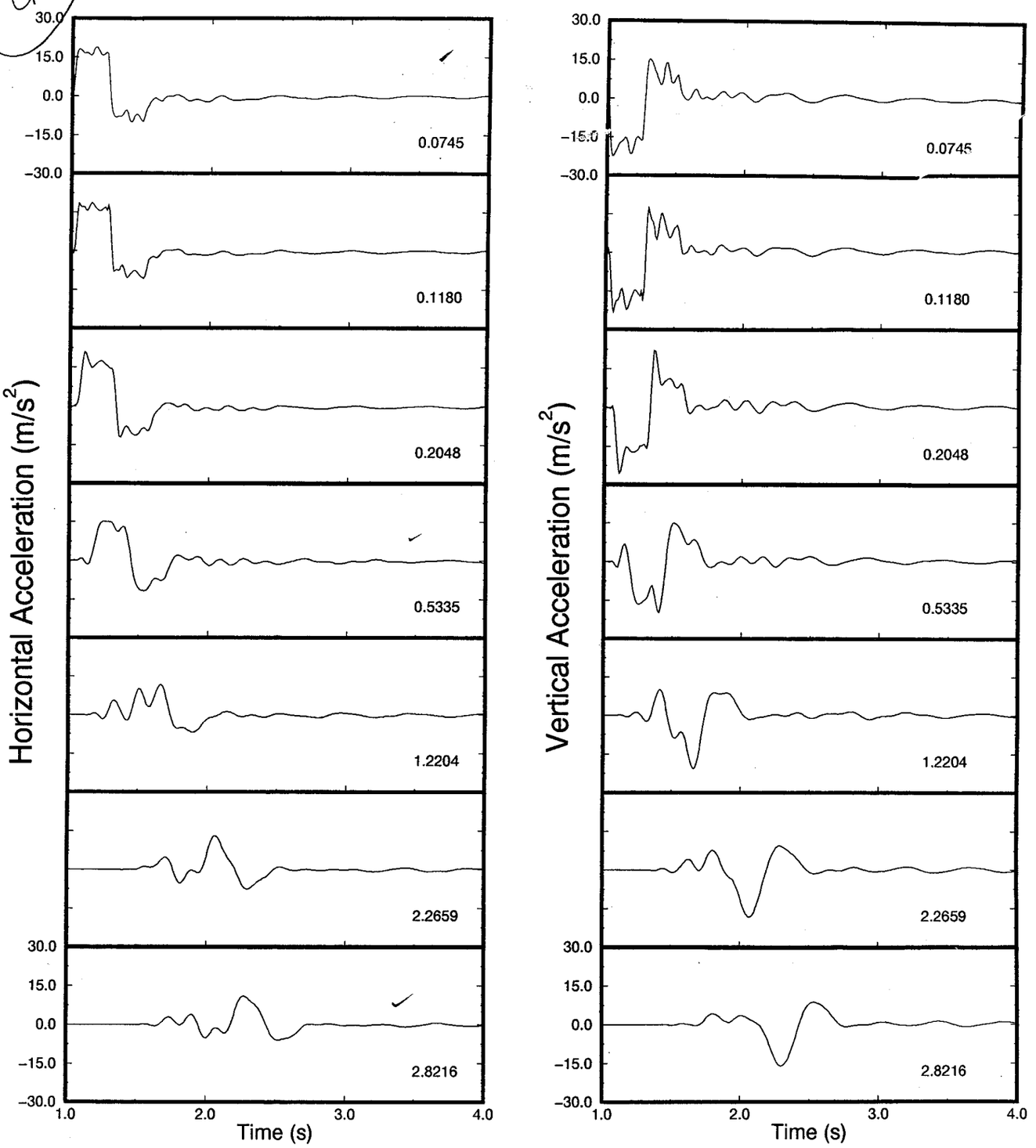
Continues on file

Extract from ds20Normal.xyd

Node	x (km)	y (km)	Distance (km)
507	13.8260	1.9720	0.00
485	13.8558	2.0065	0.0745
457	13.8859	2.0413	0.1180
435	13.9159	2.0761	0.1614
407	13.9459	2.1110	0.2048
10702	14.0750	2.2161	0.3692
10708	14.2041	2.3213	0.5335
10730	14.3271	2.4309	0.6949
10798	14.5726	2.3999	0.8920
10837	14.8032	2.6573	1.2204
10886	15.0345	2.6873	1.4357
10904	15.1411	2.8342	1.6015
10958	15.4755	3.0750	2.0115
10994	15.6144	3.0042	2.0963
11005	15.7063	3.1841	2.2659
11047	15.9470	3.3176	2.5411
11077	16.0959	3.2713	2.6469
11080	16.1824	3.4709	2.8216
11119	16.4255	3.6453	3.1193
11152	16.5820	3.6201	3.2423
11189	16.7449	3.6046	3.3756
11223	16.9890	3.8118	3.6906
11260	17.2370	4.0340	4.0165
11293	17.4038	4.0334	4.1606
11366	17.8205	4.2815	4.6455
11402	18.0674	4.5320	4.9846
11439	18.2388	4.5458	5.1400
11469	18.4119	4.5626	5.2983
11510	18.6552	4.8225	5.6389
11584	19.0691	5.1050	6.1386
11617	19.2414	5.1254	6.2980
1671	19.4137	5.1458	6.4575
1670	19.4792	5.3906	6.6366
1744	19.5792	5.3906	6.7232
1818	19.6792	5.3906	6.8098
1892	19.7792	5.3906	6.8964
1927	19.8948	5.6354	7.1189
1965	19.9448	5.6354	7.1622
2001	19.9948	5.6354	7.2055
2039	20.0448	5.6354	7.2488
4594	20.2657	5.6354	7.4401
4546	20.5492	5.8802	7.8080
4514	20.8297	6.1250	8.1733
4513	21.0476	6.1250	8.3621
4466	21.5392	6.3698	8.9102
4432	21.5949	6.6146	9.0808
4431	21.8100	6.6146	9.2671
4390	22.0763	6.8594	9.6201
4350	22.5528	7.1042	10.1552
4349	22.7665	7.1042	10.3403
4306	23.0242	7.3490	10.6858
4267	23.2814	7.5938	11.0310
4266	23.4945	7.5938	11.2155
4220	23.9651	7.8386	11.7455
4170	24.2233	8.0834	12.0915
4169	24.4412	8.0834	12.2802
4129	24.9269	8.3283	12.8233
4089	24.9695	8.5730	12.9825
4088	25.1983	8.5728	13.1806

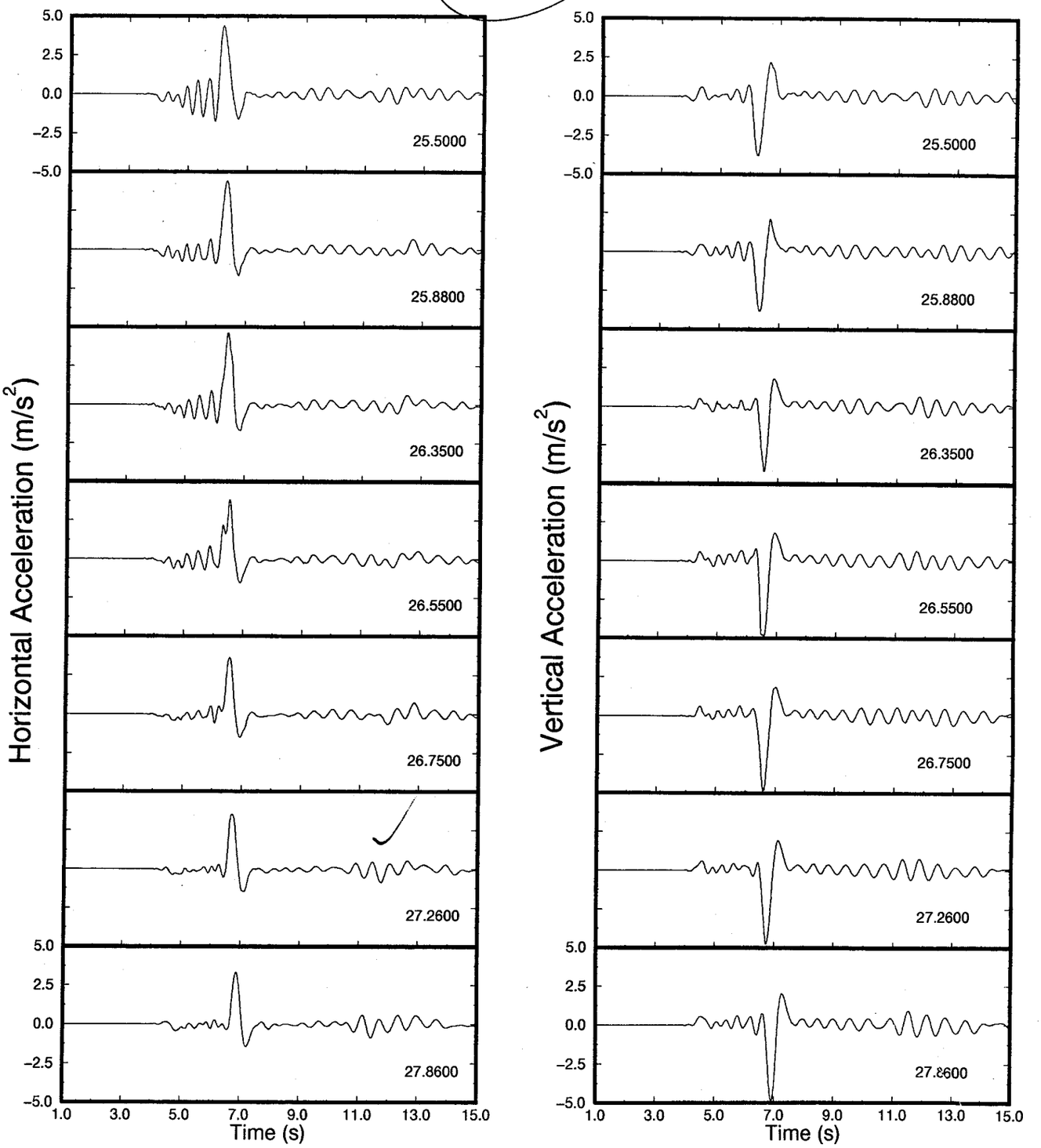
Continues on file

Typical acceleration histories from analysis case DS20 (2-km source at 10-km depth) for points on the upward normal. Number inside each plot box represents distance (km) from source along the upward normal.



Each acceleration history on this page is stored in a file named dsNkk.mmmm where kk is number of kilometers and mmmm is fraction of kilometers. For example the file for the top two plot boxes is dsN00.0745 and the bottom boxes dsN02.8216.

Typical acceleration histories from analysis case DS20 for points on the ground surface. Number inside each plot box represents distance (km) from left-boundary of model.



Each acceleration history on this page is stored in a file named topDskk.mmm. For example, the file for the top boxes is topDs25.500 and the file for the bottom boxes is topDs27.860.

July 7 1997

Profiles of Peak Acceleration

Profiles of peak horizontal acceleration, peak vertical acceleration, and peak acceleration magnitude were prepared for the following lines

(1) horizontal line ~~above~~ on ground surface

(2) horizontal line at 300-m depth

(3) line along hangingwall of fault F0, and

(4) line along upward normal ~~to~~

(See p. 47 for sketch illustrating these lines)

For each case, peak acceleration at a given point is the ~~magnitude of~~ maximum (positive or negative) value on the acceleration history for the point (e.g. p. 52 and 53).

Peak-acceleration values were extracted from the .fil files listed on p. 48 using ~~*~~ ABAQUS/POST batch jobs submitted in series (one node at a time) through the C codes GroundAccProfiles.c (for ground surface and 300-m depth lines) and FNormAccProfiles.C (for hangingwall and upward-normal lines). The two codes are similar except for the following differences:

GroundAccProfiles.c

Reads node numbers and corresponding x- and y- coordinates from a ~~user-supplied~~ (this) file containing output of ABAQUS/POST command "print node, original". User should have prepared the files (one for node set ND TOP and one for node set ND300DEP) in a previous ABAQUS/POST session.

FNormAccProfiles.c

Reads node numbers, x- and y-coordinates, and distance from source from the files listed on p. 50 for hangingwall surface and upward normal.

```
/*********************************************************************
GroundAccProfiles.c

This code prepares horizontal profiles of peak horizontal acceleration,
peak vertical acceleration, and peak acceleration magnitude using
acceleration histories stored in ABAQUS results file. User
is prompted to supply name of file containing node definitions
(node number and x- and y-coordinates) as well as the names of
ABAQUS restart and results files (.res and .fil files), and an
output file for the acceleration-profiles data.
*********************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Point{
  int node;
  float x,y,xamax,yamax,maxamag;
  struct Point *next;
};

float kmLeftBnd=-5.0;

void GetMaxAcc(struct Point* p);
void MakeCommandFile(int node, char* resultFile);
int DefineNodes(struct Point** first);
int GetFileName(char* name, char* prompt);
void StorePoint(struct Point* cur,struct Point** f,struct Point** l);
int IsBelow(struct Point* current,struct Point* p);
void PrintPoints(struct Point* first, char* outFile);
void DumpAndQuit(char* s);

main()
{
  char command[81];
  char *restartFile="0123456789012345",*resultFile="0123456789012345";
  char *outFile="0123456789012345";
  struct Point *FirstPoint,*p;

/* Get node definitions  */

  if (!DefineNodes(&FirstPoint))
    return;

/* Get file names for ABAQUS restart and results files */

  if (!GetFileName(restartFile,"Restart file, no extension") ||
    !GetFileName(resultFile,"Results file, no extension"))
      return;

  if (!GetFileName(outFile,"Acc profiles output file")) outFile = NULL;

/*****************************************
  For each node:
    Setup ABAQUS/POST command file;
    submit ABAQUS/POST job;
    get acceleration data from ABAQUS/POST output file

*****************************************/
```



```
  p = FirstPoint;
  while (p){
    MakeCommandFile(p->node,resultFile);
    sprintf(command,"abaqus56 post restart=%s input=abapost.jnl",restartFile);
    system(command);
    GetMaxAcc(p);
    p = p->next;
  }
  PrintPoints(FirstPoint,outFile);
}

void GetMaxAcc(struct Point *p)
{
  char buf[151];
  int found=0;
  float xamax,yamax,maxamag;
  FILE *Fout;

  Fout = fopen("acctmp.out","r");
  if (!Fout){
    sprintf(buf,"Unable to open acctmp.out for node %d",p->node);
    DumpAndQuit(buf);
  }
  while (fgets(buf,150,Fout))
    if (sscanf(buf,"%*f %f %f %f",&xamax,&yamax,&maxamag) == 3)
      found++;

  fclose(Fout);
  if (found){
    p->xamax = xamax;
    p->yamax = yamax;
    p->maxamag = maxamag;
  }
  else{
   sprintf(buf,"No acceleration data found for node %d",p->node);
   DumpAndQuit(buf);
  }

  return;
}

void MakeCommandFile(int node, char *resultFile)
{
  FILE *Fjn;

  Fjn = fopen("abapost.jnl","w");
    if (!Fjn) DumpAndQuit("Unable to setup abapost.jnl file");

  fprintf(Fjn,"result,file=%s\n",resultFile);
  fprintf(Fjn,"set,reportfile=acctmp.out\n");
  fprintf(Fjn,"readcurve,name=xa,var=a1,node=%d\n",node);
  fprintf(Fjn,"readcurve,name=ya,var=a2,node=%d\n",node);
  fclose(Fjn);

  system("cat acc.max.jnl >> abapost.jnl");
  return;
}

int DefineNodes(struct Point **FirstPoint)
```

GroundAccProfiles.c contd


```
{
  char buf[81],*fileName="0123456789012345";
  int choice,node;
  float x,y;
  struct Point *first,*last,*current;
  FILE *Fin;

  if (!GetFileName(fileName,"Node-definition file"))
    return(0);

  Fin = fopen(fileName,"r");
  if (!Fin)
    DumpAndQuit(strcat("Unable to open file ",fileName));

  first = last = NULL;
  while (fgets(buf,80,Fin))
    if (sscanf(buf,"%d %f %f %*f %*f",&node,&x,&y) == 3){
      current = (struct Point *)malloc(sizeof(struct Point));
      if (!current)
        DumpAndQuit("Memory allocation failure");
      current->node = node;
      current->x = x;
      current->y = y;
      StorePoint(current,&first,&last);
    }
  fclose(Fin);
  if (!first)
    DumpAndQuit(strcat("No valid input line found in file ",fileName));
  *FirstPoint = first;
  return(1);
}

int GetFileName(char *name, char *prompt)
{
  printf("%s (15 char max) or return to quit: ",prompt);
  if (!gets(name) || strlen(name)==0)
    return(0);

  return(1);
}


void StorePoint(struct Point *current,struct Point **first,struct Point **last)
{
  struct Point *old,*p;

  p = *first;

  if (!*last){          /* current is the very first item in the list */
    current->next = NULL;
    *last = current;
    *first = current;
    return;
  }

  old = NULL;
  while (p){
    if (IsBelow(current,p)){
      old = p;
      p = p->next;
```



```
    }
    else{
      if (old) {      /* insert current at this point in the list */
        old->next = current;
        current->next = p;
        return;
      }
      current->next = p;    /* current becomes new first element */
      *first = current;
      return;
    }
  }

  (*last)->next = current;  /* current becomes new last element */
  current->next = NULL;
  *last = current;
}

int IsBelow(struct Point *current,struct Point *p)
{
  int below,notBelow;
  float delta,xTol;

/*
   This code module determines whether first-named Point is
   to the right of second-named Point
*/

  below = 1;
  notBelow = 0;
  xTol = 0.05;

  delta = current->x - p->x;
  if (delta > xTol)
    return(below);

  return(notBelow);
}

void PrintPoints(struct Point *first, char *outFile)
{
  FILE *Fout;

  if (outFile){
    Fout = fopen(outFile,"w");
    if (!Fout)
      DumpAndQuit(strcat("Unable to open file ",outFile));
  }
  else Fout = stdout;

  if (!first)
    DumpAndQuit("Empty list: Nothing to print");

  fprintf(Fout,"%10s%12s%12s%15s%15s%15s\n",
    "Node","x (km)","y (km)","xAccMax (g)","yAccMax (g)","maxAccMag (g)");

  while (first){
    fprintf(Fout,"%10d%12.4f%12.4f%15.3f%15.3f%15.3f\n",
      first->node,
      (1.0e-3*first->x)-kmLeftBnd,
```

```
        1.0e-3*first->y,
        (first->xamax)/9.8,
        (first->yamax)/9.8,
        (first->maxamag)/9.8
      );
      first = first->next;
    }
  }

  void DumpAndQuit(char *s)
  {
    printf("\n *** %s ***\n\n",s);
    exit(0);
  }
```

Goto

```
/*********************************************************************
FNormAccProfiles.c

This code prepares profiles of peak horizontal acceleration,
peak vertical acceleration, and peak acceleration magnitude along
hangingwall or fault-normal starting from a source location, using
acceleration histories stored in ABAQUS results file. User
is prompted to supply name of file containing node definitions
(node number, x- and y-coordinates, and along-normal or up-dip distance
from source) as well as the names of ABAQUS restart and results files
(.res and .fil files), and output file for acceleration-profiles data.
*********************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Point{
  int node;
  float x,y,distance,xamax,yamax,maxamag;
  struct Point *next;
};

void GetMaxAcc(struct Point* p);
void MakeCommandFile(int node, char* resultFile);
int DefineNodes(struct Point** first);
int GetFileName(char* name, char* prompt);
void StorePoint(struct Point* cur,struct Point** f,struct Point** l);
void PrintPoints(struct Point* first, char* outFile);
void DumpAndQuit(char* s);

main()
{
  char command[81];
  char *restartFile="0123456789012345",*resultFile="0123456789012345";
  char *outFile="0123456789012345";
  struct Point *FirstPoint,*p;

/* Get node definitions  */

  if (!DefineNodes(&FirstPoint))
    return;

/* Get file names for ABAQUS restart and results files */

  if (!GetFileName(restartFile,"Restart file, no extension") ||
    !GetFileName(resultFile,"Results file, no extension"))
      return;

  if (!GetFileName(outFile,"Acc profiles output file")) outFile = NULL;

/*****************************************
  For each node:
    Setup ABAQUS/POST command file;
    submit ABAQUS/POST job;
    get acceleration data from ABAQUS/POST output file

*****************************************/

  p = FirstPoint;
```



```
  while (p){
    MakeCommandFile(p->node,resultFile);
    sprintf(command,"abaqus56 post restart=%s input=abapost.jnl",restartFile);
    system(command);
    GetMaxAcc(p);
    p = p->next;
  }
  PrintPoints(FirstPoint,outFile);
}

void GetMaxAcc(struct Point *p)
{
  char buf[151];
  int found=0;
  float xamax,yamax,maxamag;
  FILE *Fout;

  Fout = fopen("acctmp.out","r");
  if (!Fout){
    sprintf(buf,"Unable to open acctmp.out for node %d",p->node);
    DumpAndQuit(buf);
  }
  while (fgets(buf,150,Fout))
    if (sscanf(buf,"%*f %f %f %f",&xamax,&yamax,&maxamag) == 3)
      found++;

  fclose(Fout);
  if (found){
    p->xamax = xamax;
    p->yamax = yamax;
    p->maxamag = maxamag;
  }
  else{
   sprintf(buf,"No acceleration data found for node %d",p->node);
   DumpAndQuit(buf);
  }

  return;
}

void MakeCommandFile(int node, char *resultFile)
{
  FILE *Fjn;

  Fjn = fopen("abapost.jnl","w");
    if (!Fjn) DumpAndQuit("Unable to setup abapost.jnl file");

  fprintf(Fjn,"result,file=%s\n",resultFile);
  fprintf(Fjn,"set,reportfile=acctmp.out\n");
  fprintf(Fjn,"readcurve,name=xa,var=a1,node=%d\n",node);
  fprintf(Fjn,"readcurve,name=ya,var=a2,node=%d\n",node);
  fclose(Fjn);

  system("cat acc.max.jnl >> abapost.jnl");
  return;
}

int DefineNodes(struct Point **FirstPoint)
{
  char buf[81],*fileName="0123456789012345";
```

```
  int choice,node;
  float x,y,distance;
  struct Point *first,*last,*current;
  FILE *Fin;

  if (!GetFileName(fileName,"Node-definition file"))
    return(0);

  Fin = fopen(fileName,"r");
  if (!Fin)
    DumpAndQuit(strcat("Unable to open file ",fileName));

  first = last = NULL;
  while (fgets(buf,80,Fin))
    if (sscanf(buf,"%d %f %f %f",&node,&x,&y,&distance) == 4){
      current = (struct Point *)malloc(sizeof(struct Point));
      if (!current)
        DumpAndQuit("Memory allocation failure");
      current->node = node;
      current->x = x;
      current->y = y;
      current->distance = distance;
      StorePoint(current,&first,&last);
    }
  fclose(Fin);
  if (!first)
    DumpAndQuit(strcat("No valid input line found in file ",fileName));
  *FirstPoint = first;
  return(1);
}

int GetFileName(char *name, char *prompt)
{
  printf("%s (15 char max) or return to quit: ",prompt);
  if (!gets(name) || strlen(name)==0)
    return(0);

  return(1);
}


void StorePoint(struct Point *current,struct Point **first,struct Point **last)
{
  struct Point *old,*p;

  p = *first;

  if (!*last){          /* current is the very first item in the list */
    current->next = NULL;
    *last = current;
    *first = current;
    return;
  }

  old = NULL;
  while (p){
    if ((current->distance)>(p->distance)){
      old = p;
      p = p->next;
    }
```



```
    else{
      if (old) {      /* insert current at this point in the list */
        old->next = current;
        current->next = p;
        return;
      }
      current->next = p;     /* current becomes new first element */
      *first = current;
      return;
    }
  }

  (*last)->next = current;  /* current becomes new last element */
  current->next = NULL;
  *last = current;
}

void PrintPoints(struct Point *first, char *outFile)
{
  FILE *Fout;

  if (outFile){
    Fout = fopen(outFile,"w");
    if (!Fout)
      DumpAndQuit(strcat("Unable to open file ",outFile));
  }
  else Fout = stdout;

  if (!first)
    DumpAndQuit("Empty list: Nothing to print");

  fprintf(Fout,"%10s%12s%12s%15s%15s%15s%15s\n",
    "Node","x (km)","y (km)","frmSource (km)","xAccMax (g)",
    "yAccMax (g)","maxAccMag (g)");

  while (first){
    fprintf(Fout,"%10d%12.4f%12.4f%15.4f%15.3f%15.3f%15.3f\n",
      first->node,
      first->x,
      first->y,
      first->distance,
      (first->xamax)/9.8,
      (first->yamax)/9.8,
      (first->maxamag)/9.8
    );
    first = first->next;
  }
}

void DumpAndQuit(char *s)
{
  printf("\n *** %s ***\n\n",s);
  exit(0);
}
```

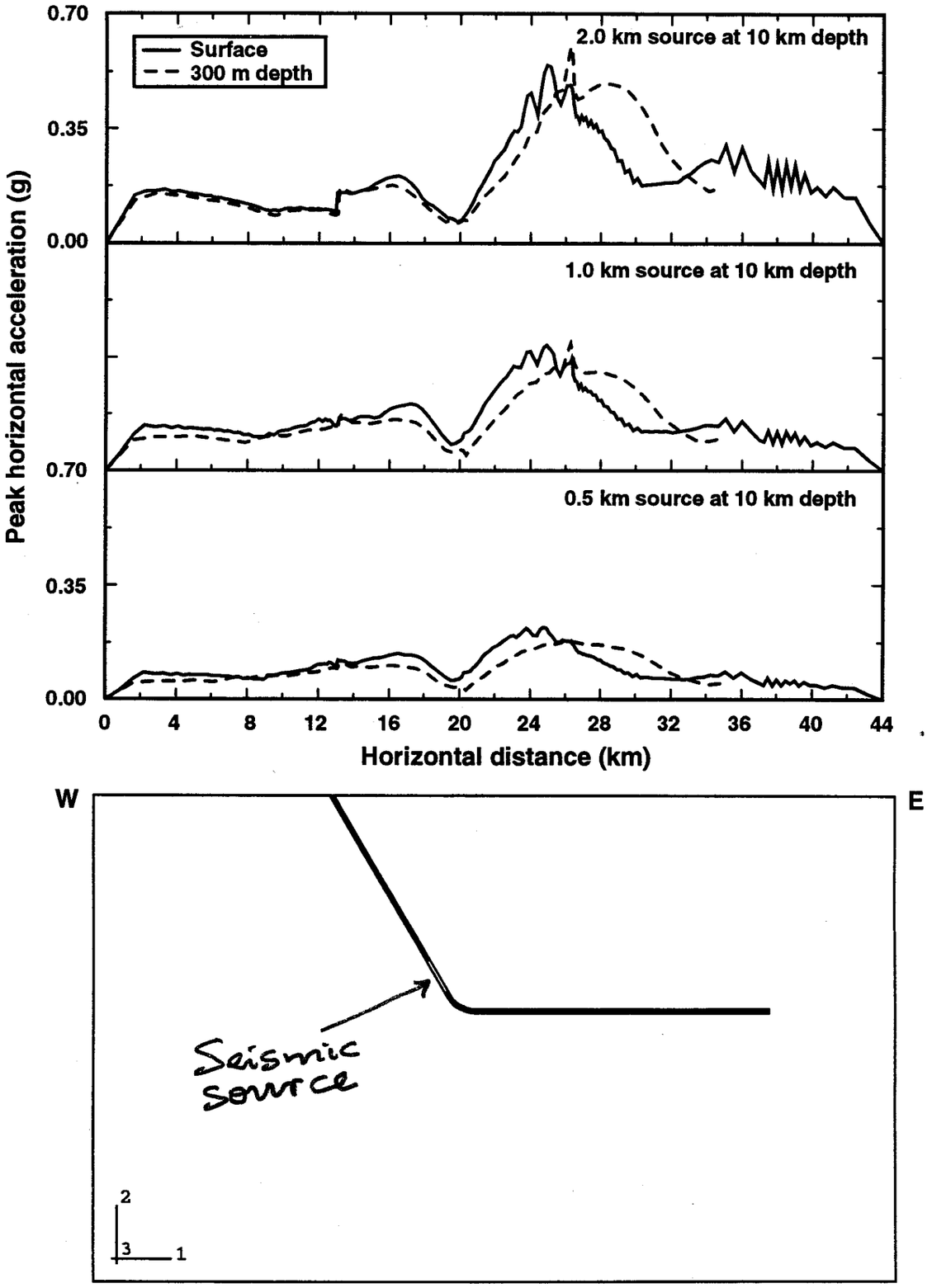
Each of the codes ~~is~~ reproduced on p. 55-58 appends a series of ABAQUS/POST commands to the top of file acc.max.jnl (reproduced below) to generate a command file that is then passed to ABAQUS/POST in batch mode to extract peak accelerations for a specific node from a specific .fil file.

file acc.max.jnl

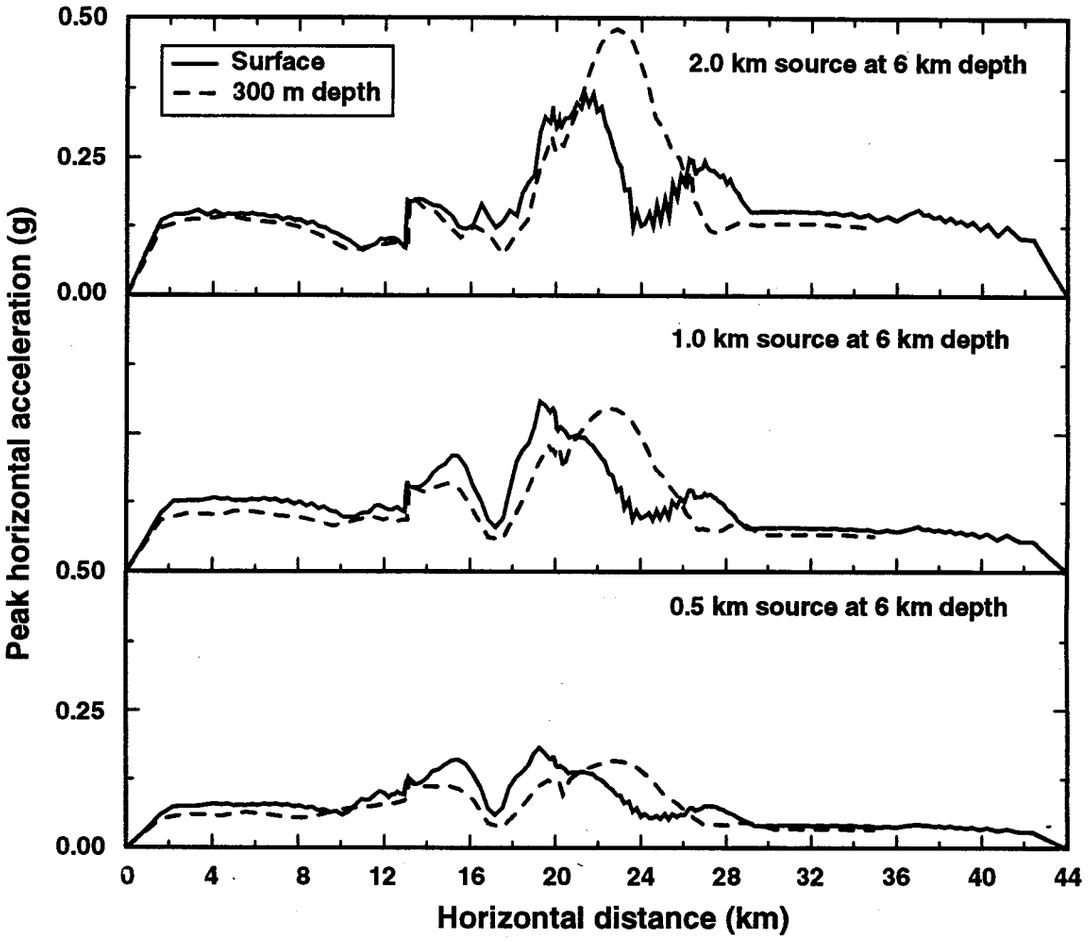
Listing for Goodluck Ofoegbu Wed Aug 13 11:29:35 1997 Page 1

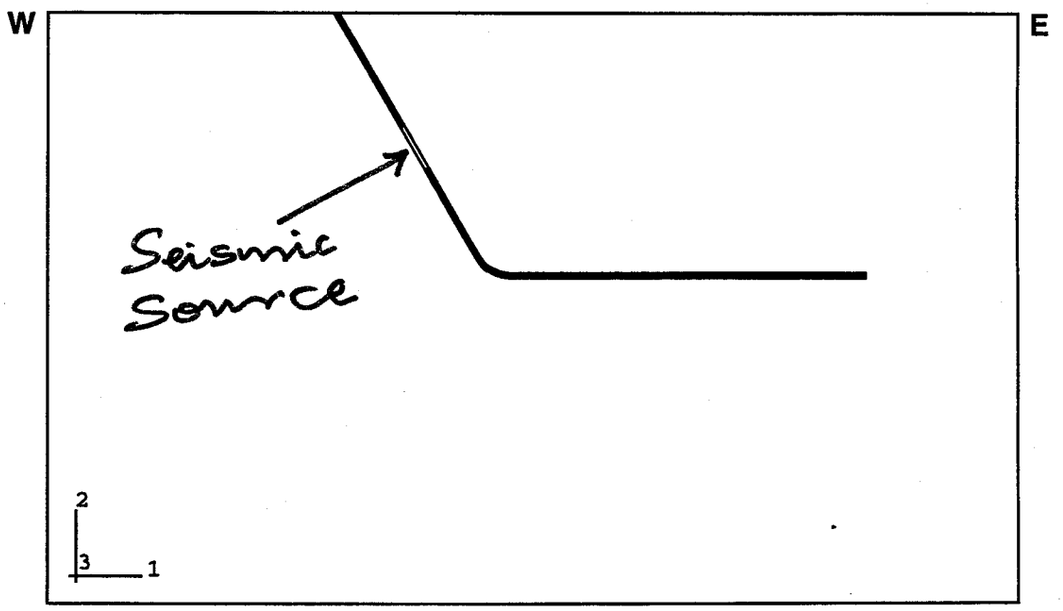
```
definecurve,name=xa2,operation=multiply
xa
xa
definecurve,name=ya2,operation=multiply
ya
ya
definecurve,name=am2,operation=add
xa2
ya2
definecurve,name=mag,operation=squareroot
am2
definecurve,name=maxmag,operation=maximum
mag
definecurve,name=xabs,operation=abs
xa
definecurve,name=yabs,operation=abs
ya
definecurve,name=xamax,operation=maximum
xabs
definecurve,name=yamax,operation=maximum
yabs
printcurve
xamax,yamax,maxmag

end
```

Examples of peak horizontal acceleration profiles obtained using the codes described on p. 54 through current page are shown in the following pages (p. 60-64).

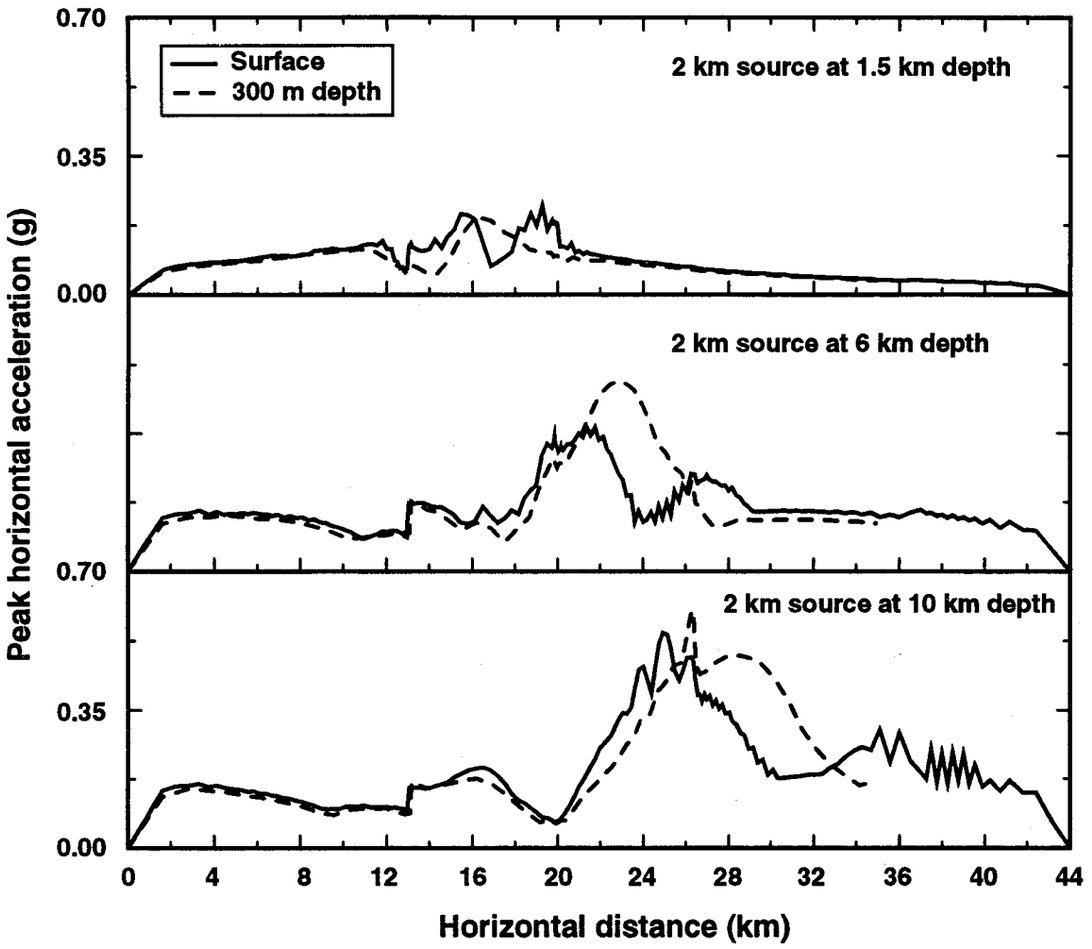


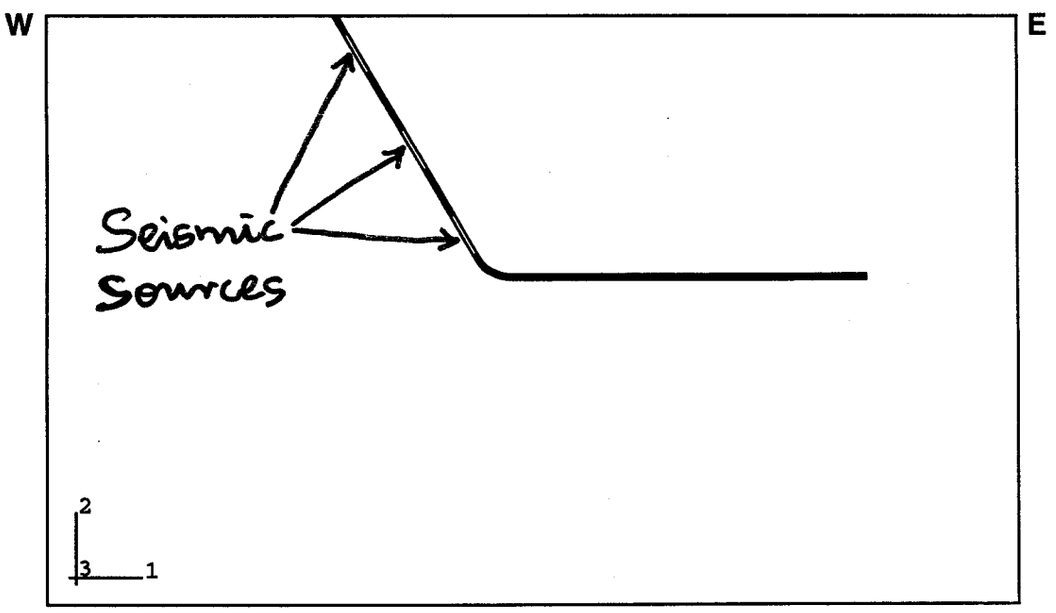
Profiles of peak horizontal acceleration at the ground surface and 300-m depth from analysis cases DS20, DS10 and DS05. Outline of the analysis domain shown at the bottom of the figure. Each profile extends west to east across the model domain.



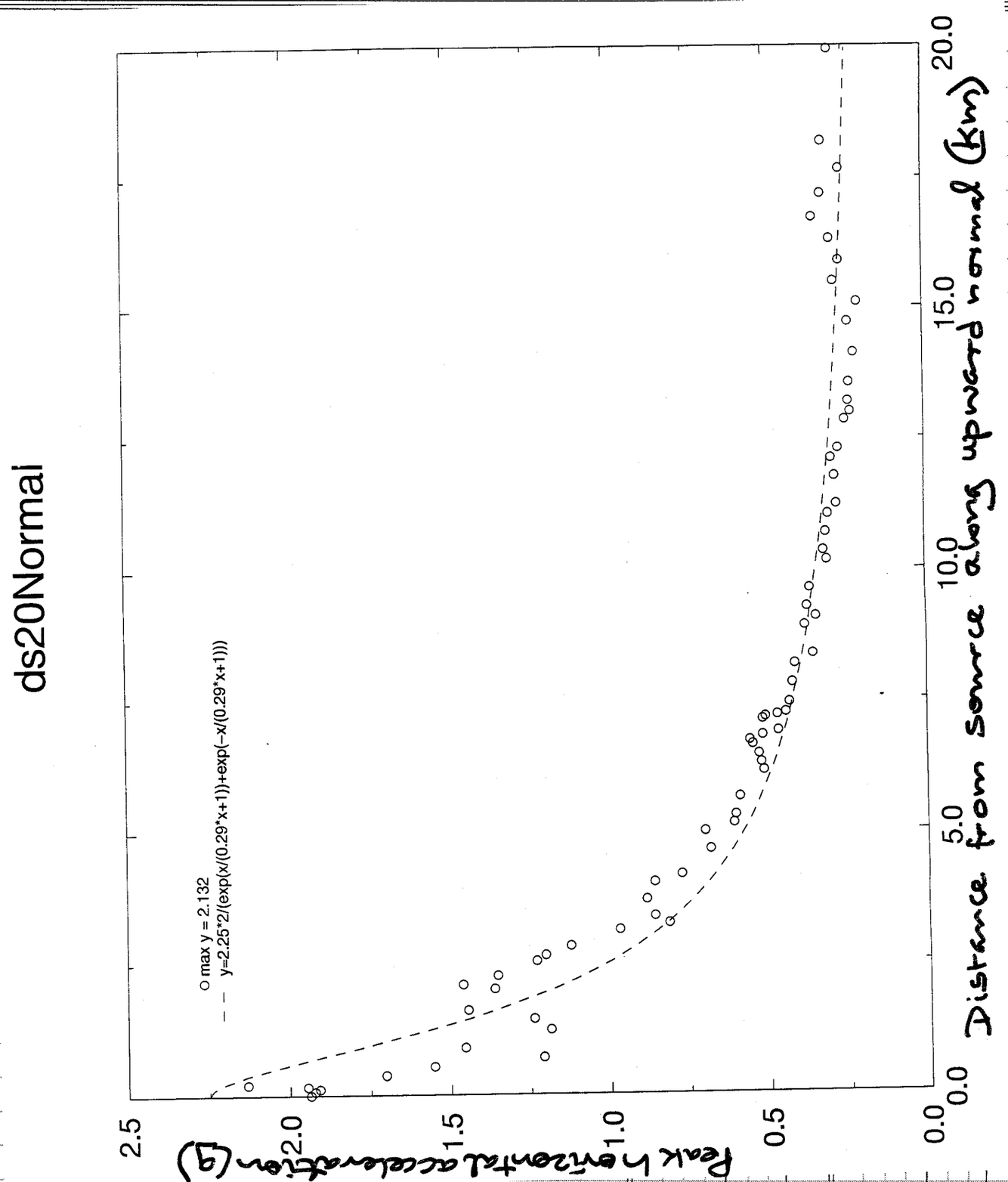


Profiles of peak horizontal acceleration from analysis cases IS20, IS10 and IS05. See previous page for additional description of the figure.

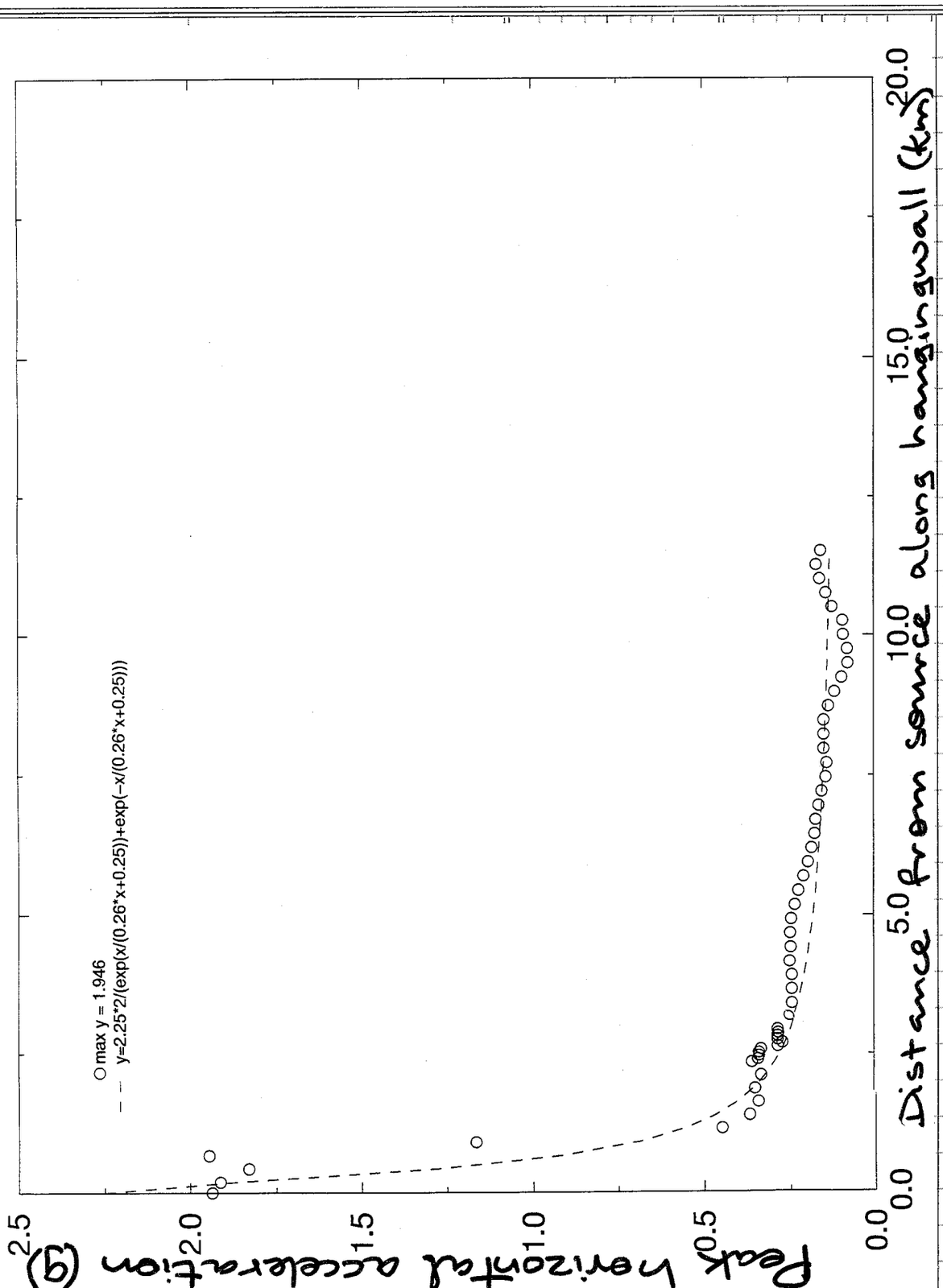




Profiles of peak horizontal acceleration from analysis cases ~~SS20~~ GM0 SS20, IS20 and DS20. See p. 60 for additional description of this figure.



Profile of peak horizontal acceleration along upward normal from analysis case DS20 (2-km source at 10-km depth). Profile shows decay of acceleration with increasing distance from source along upward normal.



Profile of peak horizontal acceleration along hangingwall of fault F0 from analysis case DS20. Profile shows decay of acceleration with increasing distance from source along hangingwall.

August 19 1997

Contours of Acceleration Resultant

Acceleration resultant A_r is defined as follows for the 2D model

$$A_r = \sqrt{A_1 A_1 + A_2 A_2}$$

where A_1 and A_2 are horizontal and vertical acceleration components. The components A_1 and A_2 were contoured using procedure described on page 34. ~~ABAQ~~ GW

Special procedure is required for developing A_r contours because ABAQUS does not give values of A_r.

A_r contours were prepared for cases DS20 and DS05 (2.0-km and 0.5-km sources, respectively, at 10-km depth) at times of 1.0, 3.0, and 5.0 s following start of seismic event. DS20 results were extracted from file sc03.res and DS05 results from file sc18.res.

A_r contour preparation followed the following steps:

(1) Extract A_1 and A_2 data from appropriate ABAQUS restart file using the following job file

Listing for Goodluck Ofoegbu

```
*HEADING
*PREPRINT,ECHO=NO,MODEL=NO,HISTORY=NO
*POST OUTPUT,STEP=4
50
*NODE PRINT, NSET=ALLNODES,SUMMARY=NO
A
*POST OUTPUT,STEP=5
200
*NODE PRINT, NSET=ALLNODES,SUMMARY=NO
A
*POST OUTPUT,STEP=6
200
*NODE PRINT, NSET=ALLNODES,SUMMARY=NO
A
```

Job step and increment number for time of 1 s

for time of 3 s

For time of 5 s

This file is submitted to ABAQUS following procedure for POST-OUTPUT jobs:

abaqus job=jobName oldjob=restartFileName

(2) Break the resulting output file (jobName.dat) using a text editor to obtain a separate A_1-and-A_2 file for 1.0, 3.0, and 5.0 s time. Each A_1-and-A_2 file contains several lines of data giving nodeNumber, A_1, and A_2 in each line.

(3) Calculate A_r from each A_1-and-A_2 file using C code Accxy2Mag.C, reproduced below.

```
/*********************************************************************
Accxy2Mag.c

Translate nodal-acceleration data (ABAQUS output) from x- and
y-components to acceleration magnitude.
*********************************************************************/

#include <stdio.h>
#include <math.h>

main()
{
  char buf[151];
  int node;
  float xacc,yacc;

  while (gets(buf))
    if (sscanf(buf,"%d %f %f",&node,&xacc,&yacc) == 3)
      printf("%10d,%12.4E\n",node,sqrt(xacc*xacc + yacc*yacc));
}
```

resultants

Output from the above code consists of lines of nodeNumber and A_r. This data will be input back into ABAQUS with the A_r introduced as user-defined field variables.

(4) Submit A_r values to ABAQUS to run ~~three~~ a dummy analysis step using the input file reproduced below. The three sets of A_r values (for time 1.0, 3.0, and 5.0 s) are introduced as three user-defined field variables FV1, FV2, and FV3 for A_r at time 1.0, 3.0, and 5.0 s, respectively.

```
*HEADING
Model to read in and plot magnitudes of acceleration resultant from
analysis cases based on m09 model. Magnitudes of acceleration
are read in as user-defined field variables, which can then
be plotted using ABAQUS/POST
*PREPRINT, MODEL=NO, HISTORY=NO, ECHO=NO
**
*RESTART, WRITE, OVERLAY
***
*** Read model from external file
*** Then read acceleration-magnitude files
***
*INCLUDE, INPUT=m09aModel.def
*INITIAL CONDITION,TYPE=FIELD,VARIABLE=1,INPUT=sc031.amag
*INITIAL CONDITION,TYPE=FIELD,VARIABLE=2,INPUT=sc033.amag
*INITIAL CONDITION,TYPE=FIELD,VARIABLE=3,INPUT=sc035.amag
***
*** Dummy Step
***
*STEP
*STATIC
1.0,1.0
*NODE PRINT, FREQ=0
*EL PRINT, FREQ=0
*END STEP
```

Contains model definition based on #m09 model.

A_r files for 1.0, 3.0, & 5.0 s.

Analysis step does nothing.

All that ABAQUS does in this analysis is place

the A_r values into its database as user-defined field variables.

The input files for Accxy2Mag.C are :

sc031.axy, sc033.axy, sc035.axy } A_1-and-A_2 files for DS20 case at time 1.0, 3.0, and 5.0 s

sc181.axy, sc183.axy, sc185.axy } A_1-and-A_2 files for DS05 case at time 1.0, 3.0, and 5.0 s

Output files from Accxy2Mag.C; i.e., A_r files, are:

sc031.amag, sc033.amag, sc035.amag } A_r files at 1.0, 3.0, and 5.0 s for DS20 case.

sc181.amag, sc183.amag, sc185.amag } A_r files at 1.0, 3.0, and 5.0 s for DS05 case.

The ABAQUS output files (restart files) containing the acceleration resultants as user-defined field variables are

ds20amag.res . from ds20amag.inp

ds05amag.res from ds05amag.inp.

(5) Process ABAQUS dummy-analysis results in ABAQUS/POST to obtain A_r contours. The commands used to accomplish this task, once ABAQUS/POST has been invoked for the appropriate restart file, are reproduced below:

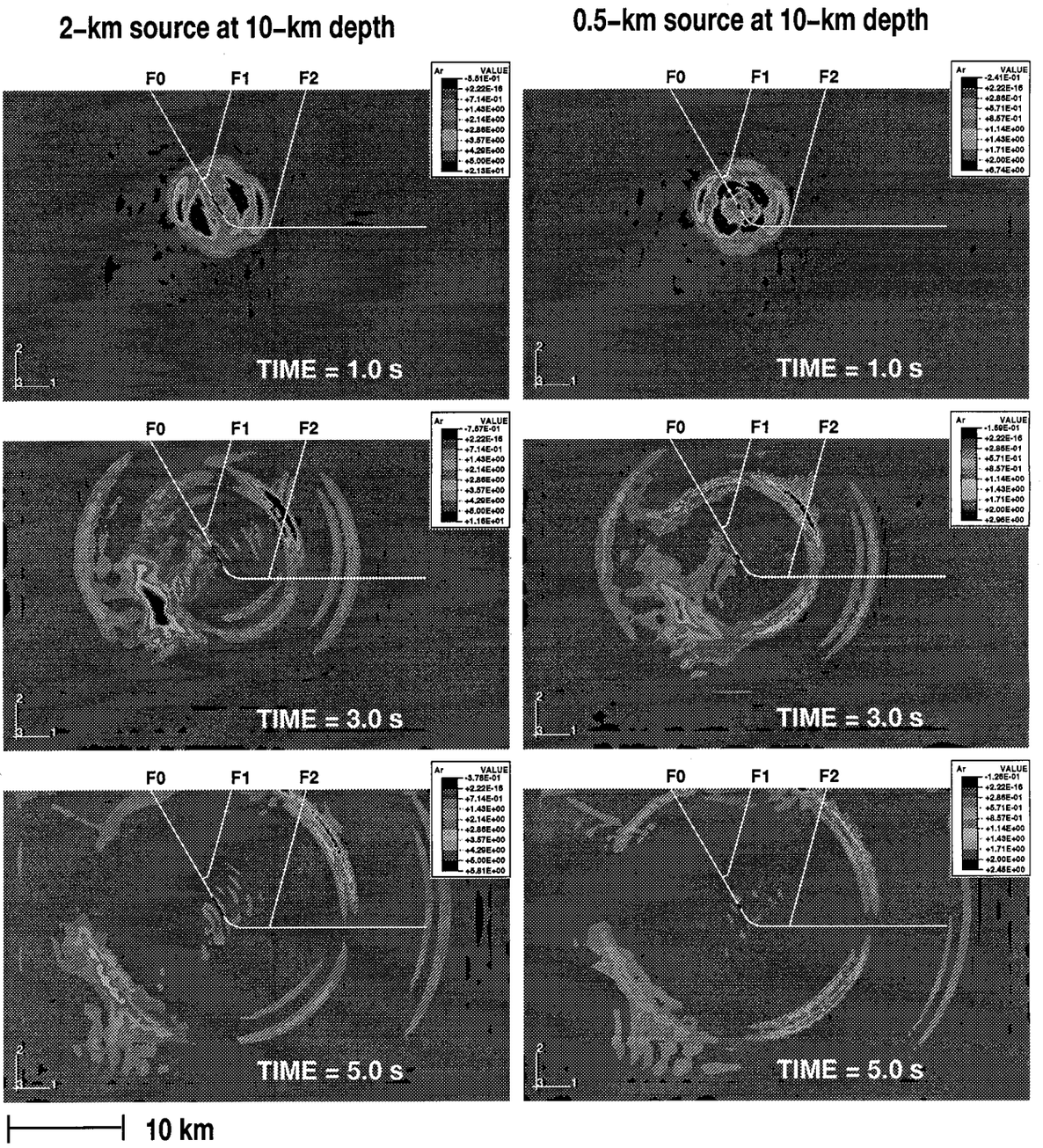
```
set, fill=on
set, c level=8
set, c max =5
set, c min =0
set, c legend size=0.25
set, c legend=(0.79, 0.99)
set, outline=off

cont,v=fv1      <-- Contours Ar for 1.0 s
pause
cont,v=fv2      <-- Contours Ar for 3.0 s
pause
cont,v=fv3      <-- Contours Ar for 5.0 s
```

The ABAQUS plots are stored in jobName.mpl files which were the processed through ABAQUS/PLOT to obtain the following postscript files:

ds20am1.ps ds20am3.ps ds20am5.ps	Ar contours at 1.0, 3.0, and 5.0 s for DS20 case.
ds05am1.ps ds05am3.ps ds05am5.ps	Ar contours at 1.0, 3.0, and 5.0 s for DS05 case.

Each of the files was processed using C code AddFaultGeom.C (p. 36) to superimpose fault geometry (from files geomDs20.ps and geomDs05.ps for DS20 and DS05 cases, respectively) on the contour plot. ~~The six ps~~ GMD

The 6 contour plots (with superimposed fault geometry showing seismic source) were combined on one page to produce the Ar contours mosaic. A black-and-white version of the mosaic is shown on p. 69.



Ar - contours mosaic for DS20 and DS05 cases.

August 25 1997

Re: Attenuation Relationships

Profiles of peak horizontal acceleration (a_{hp}) estimated using ground motion attenuation relationships from literature were compared with the profiles from the FE models (p. 54–62). Attenuation relationships were selected from a recent compilation of such relationships in a Special issue of Seismological Research Letters, the Table of Contents of which is reproduced below.

SEISMOLOGICAL RESEARCH LETTERS
Volume 68, Number 1 January/February 1997

ARTICLES

Relationships for shallow crustal earthquakes in active tectonic regions.

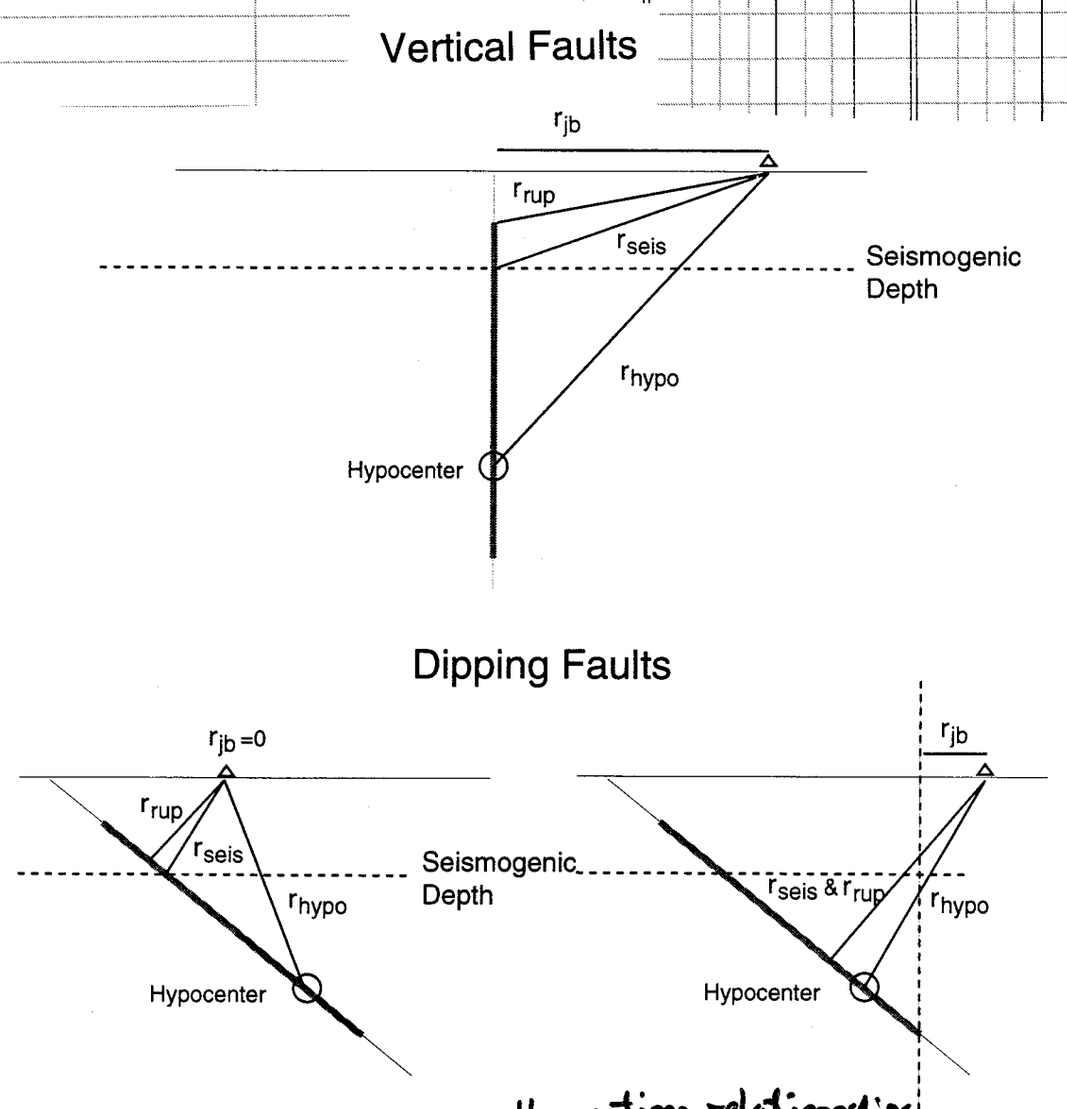
The three circled and checked (✓) were used.

Among these (two) Three relationships were selected from among the five for shallow crustal earthquakes in active tectonic regions. The three selected are:

Campbell (1997)
Sadigh et al. (1997) and
Spudich et al. (1997)

Abrahamson ~~and~~ and Silva (1997) give relationships for spectral acceleration only. Also Boore et al. (1997) indicated that their relationship is not applicable to normal-fault earthquakes.

The three relationships selected define site-to-source distance in different ways, as are illustrated in the following sketch:



attenuation relationships

▲ **Figure** Source-to-site distance measures for ground motion ~~attenuation~~. Seismogenic depth (long dashed line) is the depth of the top omogenic part of the crust. The distance measured by Campbell is the shortest distance to the rupture surface below the seismogenic depth.

Campbell (1997), Sadigh et al. (1997) and Spudich et al. (1997) use r_{seis}, r_{rup}, and r_{jb}, respectively.

Campbell (1997) Relationship

For a normal-fault earthquake ($F = 0.5$) and for a site on hard rock ($S_{HR} = 1$ and $S_{SR} = 0$).

$$\ln(a_{hp}) = -3.512 + 0.904M - 1.328\ln\sqrt{r_{seis}^2 + [0.149\exp(0.647M)]^2} + F[1.125 - 0.112\ln(r_{seis}) - 0.0957M] + S_{HR}[0.405 - 0.222\ln(r_{seis})]$$

The term for S_{SR} was omitted since $S_{SR} = 0$.

M = moment magnitude of earthquake.

The minimum depth to seismogenic zone d_{seis} was estimated based on Table 1 of Campbell (1997), which is reproduced below:

TABLE 1
Recommended Minimum Values for the Average Value of d_{SEIS}
(H_{TOP} = 3 km, H_{BOT} = 15 km)

Magnitude (M_W)	Rupture Width (W, km)	d_{SEIS} (km)		
		α = 30°	α = 45°	α = 90°
5.00	3.2	8.0	7.6	7.1
5.25	4.2	7.8	7.3	6.7
5.50	5.6	7.6	7.0	6.2
5.75	7.5	7.3	6.6	5.6
6.00	10.0	7.0	6.1	4.9
6.25	13.3	6.6	5.5	4.1
6.50	17.8	6.1	4.8	3.1
6.75	23.7	5.5	4.0	3.0
7.00	31.6	4.8	3.0	3.0

α = fault dip

Sadigh et al. (1997) Relationship*

Applicable to soft-rock sites and to both normal-fault and strike-slip earthquakes.
M = moment magnitude of earthquake

$$\ln(a_{hp}) = -0.624 + M - 2.1 \ln\left[r_{rup} + \exp(1.29649 + 0.25M)\right]$$

for $M \leq 6.5$

$$\ln(a_{hp}) = -1.274 + 1.1M - 2.1 \ln\left[r_{rup} + \exp(-0.48451 + 0.524M)\right]$$

for $M > 6.5$

Spudich et al. (1997) Relationship

Applicable to normal-fault earthquakes and for ~~sites~~ recording sites on rock.

$$\log_{10}(a_{hp}) = 0.156 + 0.229(M-6) - 0.945 \log_{10} R$$

$$R = \sqrt{r_{jb}^2 + (5.57)^2}$$

$5 \leq M \leq 7.7$

M = moment magnitude of earthquake.

Values of M for the simulated earthquakes range from 6.27 to 7.05 (p. 46). Therefore, the attenuation relationships were applied with M=6 and M=7 to obtain a range of accelerations.

The three attenuation relationships were applied to calculate two a_{hp} values (for M=6 and M=7) at each nodal point at the top of the finite element model (ground surface). Values of a_{hp} from finite element model are available at all such ground-surface points (see p. 54).

The calculation of a_{hp} using the attenuation relationships was implemented through the C codes PhaModels.c and that are reproduced on the following pages:

```
/*********************************************************************
PhaModels.c

Compute peak horizontal acceleration based on attenuation models of
Campbell (1997), Sadigh et al. (1997) and Spudich et al. (1997)
for comparison with values from finite element model. Accelerations
from attenuation models are computed for earthquakes of magnitude 6.0
and 7.0. Finite element results are from the current 2D model of
Bare Mountain Fault, for the case of a 2-km source at 10-km depth
(i.e., Case DS20). To use finite element results for other cases,
modify variables cName and srcDepth as follows:

srcDepth            source depth (km)
cName               finite-element-model case name
                    (used to compute name of file containing
                     finite element results)

      srcDepth    cName

      10.028      ds20
      10.028      ds10
      10.231      ds05
       6.15       is20
       6.15       is10
       6.37       is05

Case SS20 is not appropriate for the attenuation models

Author          G. I. Ofoegbu
Date            August 27 1997
System          ANSI C
*********************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

struct Line{
  float dYdX,yIntercept;
  float xFault,yFault;
};

float Degree2Radian(float angleInDegree);
struct Line FaultLine(float dipDegree, float xLoc, float yLoc);
float CampbellDistance(struct Line* fault, float eqMag, float x);
float SadighDistance(struct Line* fault, float x);
float JoynerBooreDistance(struct Line* fault, float x);
float AccByCampbell(struct Line* fault, float eqMag, float x);
float AccBySadigh(struct Line* fault, float eqMag, float x);
float AccBySpudich(struct Line* fault, float eqMag, float x);
float P2LDistance(float x, float y, float dYdX, float yIntercept);
void ComputeAndPrint(struct Line *bmF, float* eqMag, float srcDepth,
  FILE *Fin, FILE *Fout);
void DumpAndQuit(char* s);

main()
{
  char *cName="ds20";
  char *infile="----atop.dat",*outfile="----aMod.acc";
  float xFault=13.0,yGround=12.0,faultDip=60.0,srcDepth=10.028;
```

PhaModels.c Contd

```
  float eqMag[]={6.0,7.0};
  struct Line bmF;
  FILE *Fin,*Fout;

  bmF = FaultLine(faultDip,xFault,yGround);
  bmF.xFault = xFault;
  bmF.yFault = yGround;

  sprintf(infile,"%satop.dat",cName);
  sprintf(outfile,"%saMod.acc",cName);
  Fin = fopen(infile,"r");
  if (!Fin)
    DumpAndQuit(strcat("Unable to open file ",infile));
  Fout = fopen(outfile,"w");
  if (!Fout)
    DumpAndQuit(strcat("Unable to open output file ",outfile));

  ComputeAndPrint(&bmF,&eqMag[0],srcDepth,Fin,Fout);

  fclose(Fin);
  fclose(Fout);
}

void ComputeAndPrint(struct Line *bmF, float *mag, float srcDepth,
  FILE *Fin, FILE *Fout)
{
  char buf[151];
  int i;
  float xFault,yGround,xLoc;
  float feAcc,camAcc[2],sadAcc[2],spuAcc[2];
  float eqMag;

  fprintf(Fout,"Peak horizontal acceleration (g) ");
  fprintf(Fout,"from three attenuation models and FE model DS20\n");
  fprintf(Fout,"For Earthquake Hypocenter at depth %.1f km\n",srcDepth);
  fprintf(Fout,"%12s%12s%12s%12s%12s%12s%12s%12s\n","xPos km","FE_DS20",
      "Campbel_M6","Campbel_M7","Sadigh_M6","Sadigh_M7","Spudich_M6",
      "Spudich_M7");

  while (fgets(buf,150,Fin))
    if (sscanf(buf,"%*d %f %*f %f %*f %*f",&xLoc,&feAcc) == 2){

      for (i=0; i<2; i++){
        eqMag = mag[i];
        camAcc[i] = AccByCampbell(bmF,eqMag,xLoc);
        sadAcc[i] = AccBySadigh(bmF,eqMag,xLoc);
        spuAcc[i] = AccBySpudich(bmF,eqMag,xLoc);
      }

/* Print results */

      fprintf(Fout,"%12.2f%12.4f%12.4f%12.4f%12.4f%12.4f%12.4f%12.4f\n",
        xLoc,feAcc,camAcc[0],camAcc[1],sadAcc[0],sadAcc[1],
        spuAcc[0],spuAcc[1]);
    }
}

/*****************************************************************
AccByCampbell

This code returns magnitude of peak horizontal acceleration
using Campbell (1997) empirical attenuation function for a
normal-fault earthquake. Acceleration is computed for a hard-rock
site.

Author          G. I. Ofoegbu
Date            August 27 1997
System          ANSI C
*****************************************************************/

float AccByCampbell(struct Line* fLine, float eqMag, float x)
{
  float r,eTerm,fTypeTerm,hrTerm,rTerm,mTerm;
  float typeFactor=0.5;

  r = CampbellDistance(fLine,eqMag,x);
  eTerm = 0.149*exp(0.647*eqMag);
  mTerm = 0.904*eqMag - 3.512;
  rTerm = -1.328*log(sqrt(r*r + eTerm*eTerm));
  fTypeTerm = 1.125 - 0.112*log(r) - 0.0957*eqMag;
  fTypeTerm *= typeFactor;
  hrTerm = 0.405 - 0.222*log(r);
  return(
    exp(mTerm + rTerm + fTypeTerm + hrTerm)
  );
}

/*****************************************************************
AccBySadigh

This code returns magnitude of peak horizontal acceleration
using Sadigh et al (1997) empirical attenuation function. The
function is valid for both normal-fault and strike-slip earthquakes,
but only for acceleration at a soft-rock site.

Author          G. I. Ofoegbu
Date            August 27 1997
System          ANSI C
*****************************************************************/

float AccBySadigh(struct Line* fLine, float eqMag, float x)
{
  float r,mTerm,rTerm,eTerm;

  r = SadighDistance(fLine,x);

  if (eqMag <= 6.5){
    eTerm = exp(1.29649 + 0.25*eqMag);
    mTerm = eqMag - 0.624;
  }
  else{
    eTerm = exp(-0.48451 + 0.524*eqMag);
    mTerm = 1.1*eqMag - 1.274;
  }

  rTerm = -2.1*log(r + eTerm);
  return(
    exp(mTerm + rTerm)
  );
}
```

Campbell (1997)

Sadigh et al. (1997)

```
/*****************************************************************
AccBySpudich

This code returns magnitude of peak horizontal acceleration
using Spudich et al (1997) empirical attenuation function for
normal-fault earthquakes. Acceleration is computed for a rock site.
Function is valid for (eqMag >= 5) and (eqMag <= 7.7), where eqMag
is the earthquake magnitude (moment magnitude).

Author          G. I. Ofoegbu
Date            August 27 1997
System          ANSI C
*****************************************************************/

float AccBySpudich(struct Line* fLine, float eqMag, float x)
{
  float rjb,dTerm,rTerm,mTerm;

  rjb = JoynerBooreDistance(fLine,x);
  dTerm = sqrt(rjb*rjb + 5.57*5.57);
  rTerm = -0.945*log10(dTerm);
  mTerm = 0.156 + 0.229*(eqMag - 6.0);
  return(
    pow(10.0,(mTerm + rTerm))
  );
}

void DumpAndQuit(char *s)
{
  printf("\n *** %s ***\n\n",s);
  exit(0);
}
```

Spudich et al. (1997)

The following code modules (contained in file SitetoSource.c) were referenced by the code PhaModels.c.

```
#include <stdio.h>
#include <math.h>

struct Line{
  float dYdX,yIntercept;
  float xFault,yFault;
};

float Degree2Radian(float angleInDegree);
struct Line FaultLine(float dipDegree, float xLoc, float yLoc);
float CampbellDistance(struct Line* fault, float eqMag, float x);
float SadighDistance(struct Line* fault, float x);
float JoynerBooreDistance(struct Line* fault, float x);
float P2LDistance(float x, float y, float dYdX, float yIntercept);

/*****************************************************
Degree2Radian

Convert angle from degree to radian
*****************************************************/

float Degree2Radian(float angle)
{
  float pi;

  pi = 4.0*atan(1.0);
  return(angle*pi/180.);
}
/*************************************************************
FaultLine (2D Version)

Computes straight-line parameters for a normal fault defined
in terms of its dip and point of intersection (xLoc,yLoc)
with the ground surface. Fault dips in +x direction
*************************************************************/

struct Line FaultLine(float dip, float xLoc, float yLoc)
{
  struct Line line;
  float grad,angle;

  angle = Degree2Radian(dip);
  grad = -tan(angle);
  line.dYdX = grad;
  line.yIntercept = yLoc - grad*xLoc;
  return(line);
}

/*************************************************************
CampbellDistance

Compute distance parameter R_seis in Campbell(1997)
attenuation model. Code returns R_seis to calling program.

    d         Minimum seismogenic depth [from Campbell(1997)]
              (=3 for eqMag=7 or 5 for eqMag=6 for 60-degree fault)
    xf        x-coordinate of surface trace of fault
    x         x-coordinate of site
    eqMag     Moment magnitude of earthquake
    f         Line data structure that holds definition of
```

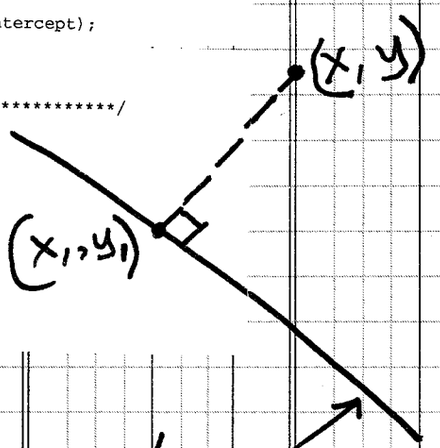
```
                Bare Mountain fault as a line in Y=aX+b format

   All points are on the ground surface; hence, y-coordinate of
   each point is the same as y-coordinate of surface trace of fault
   (f->yFault)

   Author        G. I. Ofoegbu
   Date          August 26 1997
   System        ANSI C
   ************************************************************/

   float CampbellDistance(struct Line* f, float eqMag, float x)
   {
     float d,del1X,del2X,xd,xmin;
     float dip=60.0,rad,xf;

     xf = f->xFault;
     if (eqMag > 6.9)
       d=3.0;
     else d=5.;           /* eqMag is either 7 or 6  in this code */
     rad = Degree2Radian(dip);
     del1X = d/tan(rad);
     rad = Degree2Radian(90.0-dip);
     del2X = d/tan(rad);
     xmin = xf + del1X + del2X;
     if (x > xmin)
       return(
         P2LDistance(x,f->yFault,f->dYdX,f->yIntercept)
       );
     else{
       xd = xmin - del2X;
       return(
         sqrt((x-xd)*(x-xd) + d*d)
       );
     }
   }

   /************************************************************
   JoynerBooreDistance

   Compute distance parameter R_jb in Spudichetal(1997)
   attenuation model. Code returns R_jb to calling program.

       xmin      Minimum x-coordinate on surface projection of fault
       xmax      Maximum x-coordiante on surface projection of fault
       x         x-coordinate of site
       f         Line data structure that holds definition of
                 Bare Mountain fault as a line in Y=aX+b format

   All points are on the ground surface, which is horizontal

   Author        G. I. Ofoegbu
   Date          August 26 1997
   System        ANSI C
   ************************************************************/

   float JoynerBooreDistance(struct Line* f, float x)
   {
     float xmin,xmax,rad;
     float dip=60.0,detachmentDepth=12.0;


     xmin = f->xFault;
     if (x < xmin)
       return(xmin-x);

     rad = Degree2Radian(dip);
     xmax = xmin + detachmentDepth/tan(rad);
     if (x > xmax)
       return(x-xmax);

     return(0.0);
   }

   /************************************************************
   SadighDistance

   Compute distance parameter R_rup in Sadighetal(1997)
   attenuation model. Code returns R_rup to calling program.

       xf        x-coordinate of surface trace of fault
       x         x-coordinate of site
       f         Line data structure that holds definition of
                 Bare Mountain fault as a line in Y=aX+b format

   All points are on the ground surface; hence, y-coordinate of
   each point is the same as y-coordinate of surface trace of fault
   (f->yFault)

   Author        G. I. Ofoegbu
   Date          August 26 1997
   System        ANSI C
   ************************************************************/

   float SadighDistance(struct Line *f, float x)
   {

     float xf;

     xf = f->xFault;

     if (x < xf)                        /* Point on footwall */
       return(xf-x);

     return(
       P2LDistance(x,f->yFault,f->dYdX,f->yIntercept)
     );
   }
```

r_{seis}

r_{jb}

r_{rup}

```
/*************************************************************************
C Function P2LDistance (2D Version)

Computes distance from a point to a line, both of which lie on the
same plane. The point is specified as (x,y). The line is specified
as (a,b) with reference to the equation (y = ax + b).

This is not a stand-alone function. It is designed to be used as a
module in a C program. The prototype definition for the function is

float P2LDistance(float x, float y, float dYdX, float yIntercept);

Author          G. I. Ofoegbu
Date            May 8 1997
System          ANSI C
*************************************************************************/

float P2LDistance(float x, float y, float a, float b)
{

  float x1,y1;

  x1 = (a*(y-b) + x)/(1.0 + a*a);
  y1 = a*x1 + b;
  return(
    sqrt((x-x1)*(x-x1) + (y-y1)*(y-y1))
  );
}
```



$$Y' = a + bX'$$

Code P2LDistance is based on the following:

The normal to line $y = ax + b$ has a gradient of $-1/a$. Using this fact, and the fact that the normal passes through both (X, Y) and (X_1, Y_1), and the fact that line passes through (X_1, Y_1) as well, it can be shown that both of the following equations apply:

$$Y_1 + \frac{X_1}{a} = Y + \frac{X}{a} \quad \text{- - - - (1)}$$

$$Y_1 = aX_1 + b \quad \text{- - - (2)}$$

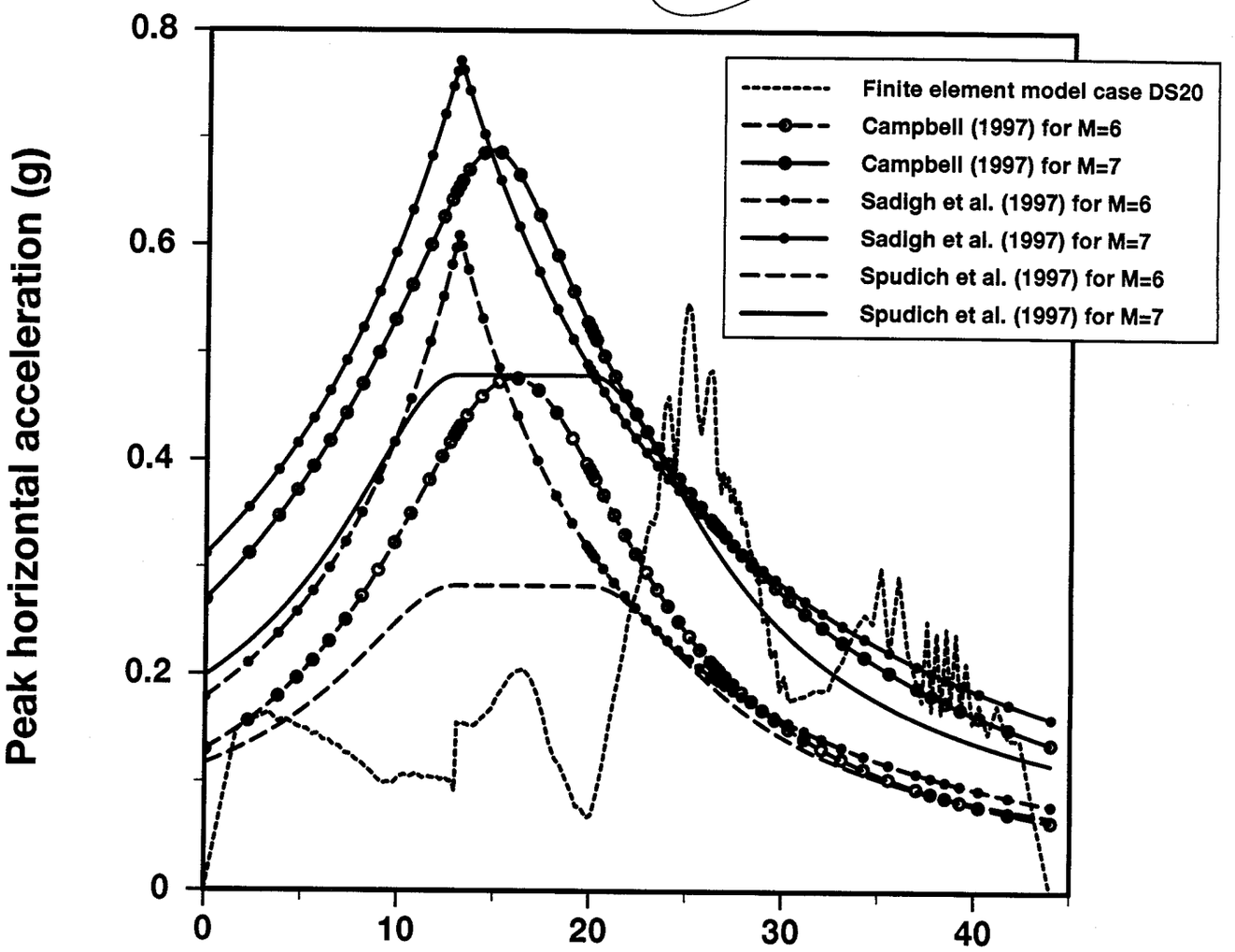
Substitute $aX_1 + b$ for Y_1 in equation (1) to obtain an equation that can then be solved to give

$$X_1 = \frac{a(Y - b) + X}{1 + a^2}$$

which can then be substituted into (2) to obtain Y_1.

Thereafter distance from (X, Y) to (X_1, Y_1) is calculated.

a_{hp} profiles calculated using the attenuation relationships are compared with the a_{hp} profile from case DS20 in the following figure.

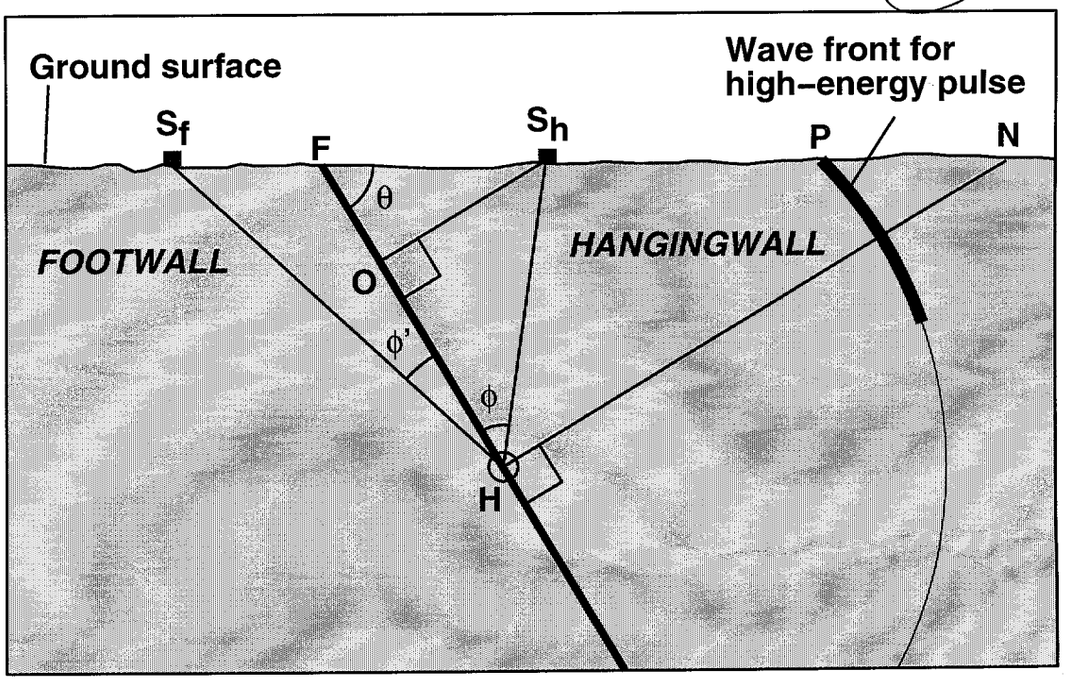


Verification of a_{hp} Profiles Computed Using Code on p. 74-77 (See Figure on p. 79)

The calculated results follow patterns that are consistent with the attenuation models. The sharp maximum on the Sadigh et al (1997) curves arises from the fact that $r_{rup} = 0$ at the surface trace of the fault (for a fault that ruptures to the surface). The Campbell curves have smooth maxima because r_{seis} has a nonzero minimum value due to the nonzero-minimum limit on d_{seis}. The Spudich et al curves are flat over the surface projection of the fault because $r_{jb} = 0$ (hence, $R = 5.57$) within this area.

Sept 9 1997

Attenuation Model Based on Fault Geometry and Site-to-Source Distance



Our results have shown that a normal-fault earthquake with hypocenter at H produces maximum ground motion at point P because of preferential propagation of seismic energy along the normal to the fault that passes through H.

Existing attenuation functions predict maximum motion in the area between E and F (where E is vertical projection of H to the ground surface).

Attenuation function for normal-fault earthquake should account for decrease of ground-motion magnitude

(i) as distance HS_h (or HS_f) increases, and

(ii) as path HS_h departs from HN, i.e., as angle Ø deviates from 90°.

The following ~~function is~~ model is proposed for ground-motion attenuation

relationships for normal-fault earthquakes

$$\gamma = \alpha_0 + \alpha_r f(r_s) + \alpha_\phi g(\phi) + \alpha_{r\phi} f(r_s) g(\phi) + h_w \beta \quad \text{---(1)}$$

$\gamma = \ln(a_{hp})$, i.e., natural log of peak horizontal acceleration.

r_s = Site-to-hypocenter distance, such as HS_h or HS_f.

$h_w = 1$ on hangingwall and $h_w = 0$ on footwall

At the surface trace of the fault Equation (1) gives

$$\ln(a_{hp}) = c_f \quad \text{on footwall side}$$

$$\ln(a_{hp}) = c_f + \beta \quad \text{on hangingwall side}$$

If $a_{hp} = a_{hwall}$ on hangingwall side and $a_{hp} = a_{fwall}$ on footwall side, then these two equations can be combined to give

$$\ln(a_{hwall}) = \ln(a_{fwall}) + \beta$$

which gives

$$\beta = \ln\left(\frac{a_{hwall}}{a_{fwall}}\right) \quad \text{---(2)}$$

Functions $f(r_s)$ and $g(\phi)$ are

$$f(r_s) = \ln(r_s) \quad \text{---(3)}$$

$$g(\phi) = 1 - r_{rup}/r_{hypo} \quad \text{---(4)}$$

r_{rup} = distance OS_h on the hangingwall or FS_f on the footwall

$r_{hypo} = HS_h$ or HS_f.

The ratio r_{rup}/r_{hypo} is equal to 1 on HN

[i.e., $g(\phi) = 0$ at N] but less than 1 at other points [i.e., $g(\phi) > 0$ and $g(\phi) < 1$ at other points].

Coefficients α_0, α_r, α_ϕ and $\alpha_{r\phi}$ can be evaluated through regression analysis and may be functions of earthquake magnitude.

The coefficients were evaluated using multiple-regression analysis in the computer code S-PLUS. The S-PLUS function used for the analyses is reproduced on the next page (using DS20 case as example).

Similar functions were run to obtain sets of α coefficients for cases DS20, DS10, DS05, IS20, IS10, and IS05.

The file assigned to variable fname (in the code on p. 83) contains the following data in each line:

n	Node number from finite element model
X	X-coordinate of node relative to west boundary of model (X is horizontal axis).
a	Nodal acceleration (peak horizontal acceleration) from finite element model
rn	Distance of site to upward fault normal (Not used).
rs	Distance r_{hypo}
rf	Distance r_{rup}
ratio	$1 - r_{rup}/r_{hypo}$

These files were generated from the output of code GroundAccProfiles.c (p. 55) using the code on p. 84–85.

The regression results are presented on p. 86.

```
function(fname = "GMotionModel/ds20top.rfsn", wmin = 1, wmax = 100, awmax =
        0.399)
{
        xfault <- 13
        xmin <- 0
        xmax <- 42.5
        a.hwall <- 0.155
        a.fwall <- 0.092
        hwfactor <- log(a.hwall/a.fwall)
        mat <- list(n = 0, x = 0, a = 0, rn = 0, rs = 0, rf = 0, ratio = 0)
        mat <- scan(fname, what = mat, skip = 1)
        x <- mat$x
        xrange <- (x > xmin & x < xmax)
        xx <- x[xrange]
        aa <- mat$a[xrange]
        a <- log(aa)
        rs <- mat$rs[xrange]
        ratio <- mat$ratio[xrange]
        vrs <- log(rs)
        w <- rep(wmin, length(xx))
        for(i in 1:length(w)) {
                if(aa[i] > awmax)
                        w[i] <- wmax
                if(xx[i] > xfault)
                        a[i] <- a[i] - hwfactor
        }
        a.fit <- lm(a ~ vrs + ratio + vrs:ratio, weights = w)
        k0 <- a.fit$coef[1]
        ks <- a.fit$coef[2]
        kn <- a.fit$coef[3]
        ksn <- a.fit$coef[4]
        xrange <- x > 0
        xx <- x[xrange]
        rs <- mat$rs[xrange]
        ratio <- mat$ratio[xrange]
        vrs <- log(rs)
        a <- mat$a[xrange]
        a.model <- k0 + ks * vrs + kn * ratio + ksn * vrs * ratio
        for(i in 1:length(xx)) {
                if(xx[i] > xfault)
                        a.model[i] <- a.model[i] + hwfactor
        }
        a.model <- exp(a.model)
        plot(xx, a, type = "p", pch = "*")
        lines(xx, a.model)
        a.fit$coef
}
```

$\beta = \ln(a_{hwall}/a_{fwall})$

Read file into data structure

Only data between $x>0$ and $x<42.5$ are used in regression analysis

log of FE acceleration values $[\ln(a)]$

r_{rup}

$1 - r_{rup}/r_{rupo}$

assign weight of 1 to every data point

Modify weight to 100 for all datapoints with $a > 0.399$

$\ln(a) - \beta$ calculated for ~~##~~ points on the hangingwall

Fit $\ln(a) - h_w\beta$ to $\alpha_0 + \alpha_r f(r_s) + \alpha_\phi g(\phi) + \alpha_{r\phi} f(r_s) g(\phi)$

with $f(r_s) = \ln(r_s)$

$g(\phi) = 1 - r_{rup}/r_{rupo}$

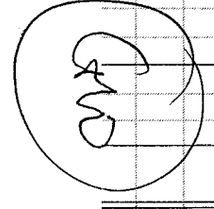
Extract $\alpha_0, \alpha_r, \alpha_\phi, \alpha_{r\phi}$

Calculate a from fitted model

Plot a from FE

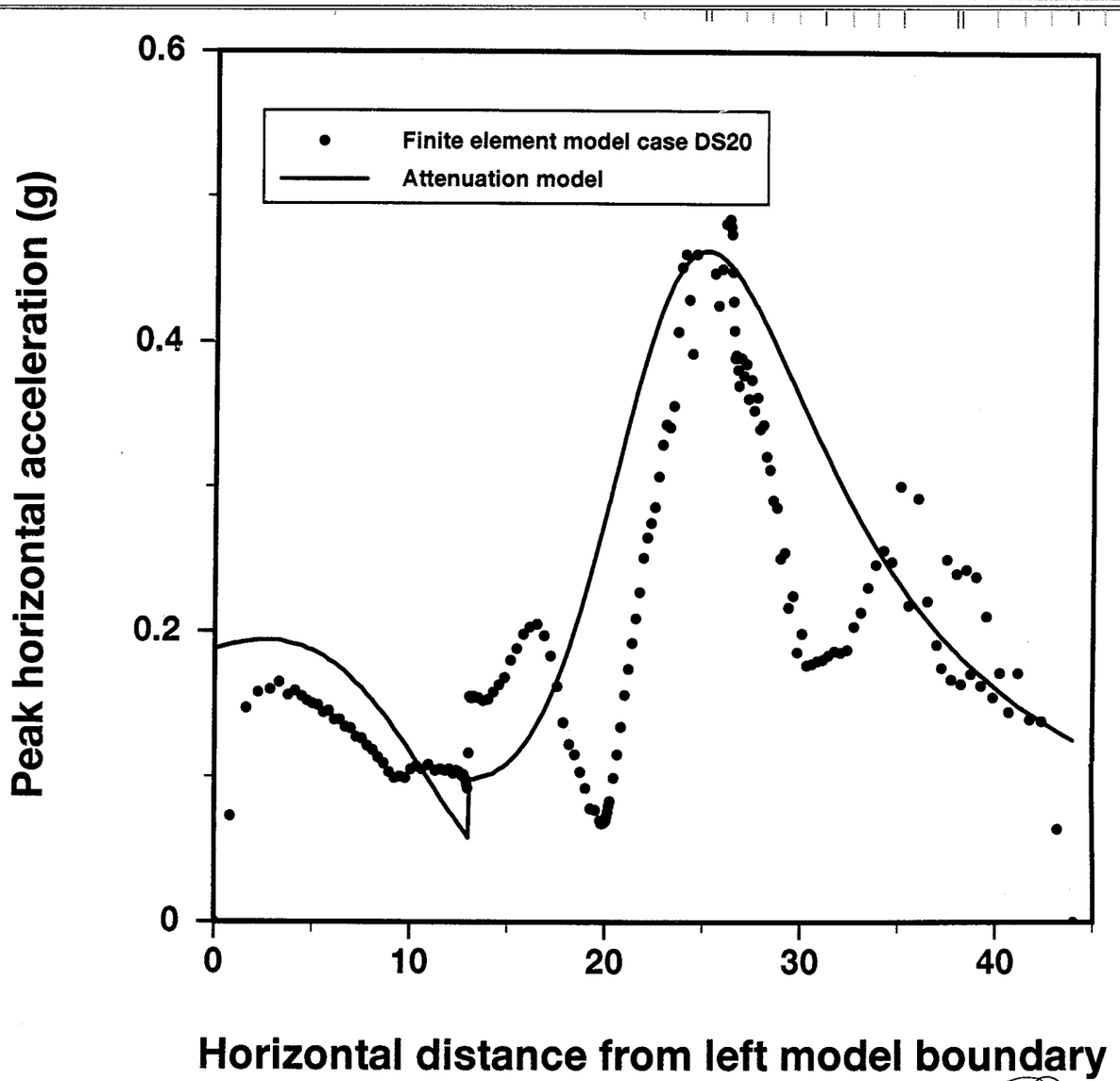
Plot a from model

Return vector of α's



```
/*********************************************************************
FromFSN.c

Compute site-to-source, site-to-normal, and site-to-fault distances
for sites on the ground surface, based on a specific 2D model of
Bare Mountain Fault.

Author          G. I. Ofoegbu
Date            August 28 1997
System          ANSI C
*********************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

struct Line{
  float dYdX,yIntercept;
};

void LineFromEnds(float x0, float y0, float x1, float y1, struct Line* line);
float P2LDistance(float x, float y, float dYdX, float yIntercept);
float Degree2Radian(float angleInDegree);
void BMFault2D(struct Line* fault,struct Line* normal);
void ComputeAndPrint(FILE *Fin, FILE *Fout);
void DumpAndQuit(char* s);

float xFaultTrace=13.0,yFaultTrace=12.0,faultDip=60.0,xNormalTrace;
float xSource,ySource,sourceDepth;

main()
{
  char buf[81];
  char *inFile="../AccProfiles/-s--atop.dat",*outFile="-s--top.rfsn";
  char *names[]={"ds20","ds10","ds05","is20","is10","is05","ss20"};
  int i;
  float depth[]={10.028,10.028,10.231,6.15,6.15,6.37,1.762};
  FILE *Fin,*Fout;

  for (i=0; i<7; i++){
    sourceDepth = depth[i];
    sprintf(inFile,"../AccProfiles/%satop.dat",names[i]);
    sprintf(outFile,"%stop.rfsn",names[i]);
    Fin = fopen(inFile,"r");
    if (!Fin)
      DumpAndQuit(strcat("Unable to open ",inFile));
    Fout = fopen(outFile,"w");
    if (!Fin)
      DumpAndQuit(strcat("Unable to open ",outFile));
    ComputeAndPrint(Fin,Fout);
    fclose(Fin);
    fclose(Fout);
  }
}

void ComputeAndPrint(FILE *Fin, FILE *Fout)
{
  char buf[121];
  int node;

  float depth,fromNormal,fromSource,fromFault,x,y;
  float feAcc,rfrsRatio;
  struct Line faultLine,normalLine;


/* Compute equations of fault line and normal line in form y=ax+b */

  BMFault2D(&faultLine,&normalLine);

  fprintf(Fout,"%10s%10s%15s%15s%15s%15s%15s\n","Node","x-km","feAcc-g",
    "fromNormal-km","fromSource-km","fromFault-km","1-rf/rs");

/*
  Obtain from file, for points on the ground surface:
    nodal-point numbers, x- and y- coordinates,
    and peak horizontal acceleration from finite element analyses.
*/

  while(fgets(buf,120,Fin))
    if (sscanf(buf,"%d %f %f %f %*f %*f",&node,&x,&y,&feAcc) == 4){

/*
   Distance from normal:
     Based on horizontal distance
       if point is beyond normal block (x>xNormalTrace);
     or on perpendicular distance from normal
*/

      if (x > xNormalTrace)
        fromNormal = sqrt((x-xNormalTrace)*(x-xNormalTrace));
      else
        fromNormal = P2LDistance(x,y,normalLine.dYdX,normalLine.yIntercept);

/*
   Distance from source:
     Based on radial distance from source point;
*/
      fromSource = sqrt(
        (x-xSource)*(x-xSource) + (y-ySource)*(y-ySource)
      );

/*
   Distance from fault:
     Based on horizontal distance if point is on footwall (x < xFaultTrace);
     or on perpendicular distance from fault if point is on hangingwall
*/
      if (x < xFaultTrace)
        fromFault = sqrt((x-xFaultTrace)*(x-xFaultTrace));
      else
        fromFault = P2LDistance(x,y,faultLine.dYdX,faultLine.yIntercept);

      rfrsRatio = fromFault/fromSource;
      if (rfrsRatio > 1.0) rfrsRatio=1.0;
      fprintf(Fout,"%10d%10.4f%15.3f%15.4f%15.4f%15.4f%15.4f\n",
        node,x,feAcc,fromNormal,fromSource,fromFault,1.0-rfrsRatio);
    }
}
```

```
/*********************************************************************
C Function BMFault2D

Determine equations of Bare-Mountain-Fault line and normal to the fault
through a source point on the fault. Each equation is defined as
y = ax + b and parameters (a,b) are returned through the Line data
structure defined above.

This is not a stand-alone function. It is designed to be used as a
module in a C program.

Author          G. I. Ofoegbu
Date            August 15 1997
System          ANSI C
*********************************************************************/

void BMFault2D(struct Line* faultLine,struct Line* normalLine)
{
  float angle,yNormalTrace,srcDepth;

  srcDepth = sourceDepth;
  angle = Degree2Radian(faultDip);
  ySource = yFaultTrace - srcDepth;
  xSource = xFaultTrace + srcDepth/tan(angle);
  xNormalTrace = xSource + srcDepth*tan(angle);
  yNormalTrace = yFaultTrace;

  LineFromEnds(xSource,ySource,xFaultTrace,yFaultTrace,faultLine);
  LineFromEnds(xSource,ySource,xNormalTrace,yNormalTrace,normalLine);
}

/*********************************************************
C function LineFromEnds  (2D Version)

Compute equation of line in terms of y = ax + b given end points.
Parameters (a,b) are returned through Line data structure provided
by calling code.

*********************************************************/

void LineFromEnds(float x0, float y0, float x1, float y1,
struct Line *line)
{
  float yIntercept,gradient;

  gradient = (y1 - y0)/(x1 - x0);
  yIntercept = y1 - gradient*x1;
  line->dYdX = gradient;
  line->yIntercept = yIntercept;
}
/*********************************************************
C function Degree2Radian

Convert angles from degree to radian

*********************************************************/

float Degree2Radian(float angle)
{
  float pi;

  pi = 4.0*atan(1.0);
  return(angle*pi/180.);
}

/*********************************************************************
C Function P2LDistance (2D Version)

Computes distance from a point to a line, both of which lie on the
same plane. The point is specified as (x,y). The line is specified
as (a,b) with reference to the equation (y = ax + b).

This is not a stand-alone function. It is designed to be used as a
module in a C program. The prototype definition for the function is

float P2LDistance(float x, float y, float dYdX, float yIntercept);

Author          G. I. Ofoegbu
Date            May 8 1997
System          ANSI C
*********************************************************************/

float P2LDistance(float x, float y, float a, float b)
{

  float x1,y1;

  x1 = (a*(y-b) + x)/(1.0 + a*a);
  y1 = a*x1 + b;
  return(
    sqrt((x-x1)*(x-x1) + (y-y1)*(y-y1))
  );
}

void DumpAndQuit(char *s)
{
  printf("\n *** %s ***\n\n",s);
  exit(0);
}
```



Attenuation model parameters for simulated earthquakes

Model ID	β	α_o	α_r	α_ϕ	$\alpha_{r\phi}$
DS20	0.5216	3.4022	-1.8215	-14.086	5.0240
DS10	0.0429	4.4719	-2.2125	-10.844	3.8215
DS05	0.0189	4.4624	-2.4192	-9.9868	3.5522
IS20	0.6816	0.2722	-0.9880	-7.5411	3.3181
IS10	0.3868	1.3105	-1.4957	-5.9546	2.7173
IS05	0.2211	0.7855	-1.4517	-2.7858	1.3364

Parameters β, α_o, α_r, α_ϕ, and $\alpha_{r\phi}$ are defined in Eq. (1).

I have reviewed this scientific notebook and find it in compliance with QAP-001. There is sufficient information regarding procedures used for conducting tests, aquiring and analyzing data so that another qualified individual could repeat the activity.

7-15-99