DEPARTMENT OF THE INTERIOR

U.S. GEOLOGICAL SURVEY

PROGRAM SEISRISK III ADAPTED TO PERSONAL COMPUTERS

by

E. P. Arnold[1]
U.S. Geological Survey

Open-File Report 89-557

OF89-557-A, Documentation, 31p., microfiche or paper copy;
OF89-557-B, OS/2 executable module, 3.5″ diskette;
OF89-557-C, OS/2 executable module, 5.25″ diskette;
OF89-557-D, OS/2/SAA source program, 3.5″ diskette;
OF89-557-E, OS/2/ANSI FORTRAN 77 source program, 3.5″ diskette;
OF89-557-F, OS/2/SAA source program, 5.25″ diskette;
OF89-557-G, OS/2/ANSI FORTRAN 77 source program, 5.25″ diskette;
OF89-557-H, DOS/SAA source program, 3.5″ diskette;
OF89-557-J, DOS/ANSI FORTRAN 77 source program, 3.5″ diskette;
OF89-557-K, DOS/SAA source program, 5.25″ diskette;
OF89-557-L, DOS/ANSI FORTRAN 77 source program, 5.25″ diskette;
OF89-557-M, DOS executable module, 3.5″ diskette;
OF89-557-N, DOS executable module, 2-5.25″ diskettes in BACKUP format.

All diskettes contain the described programs in ASCII format in a two level directory tree except for OF89-557-N. For it, use RESTORE with the /S option which creates a directory \SEIS3 on the desired drive. All versions are accompanied by a set of test data.

[1] U. S. Geological Survey, Box 25046, MS 966, Denver Federal Center, Denver, Colorado 80225.

1989

Abstract

SEISRISK III is a computer program designed to calculate maximum seismic ground-motion levels that have a specified probability of not being exceeded during fixed time periods at each of a set of sites uniformly spaced on a two-dimensional grid (Bender and Perkins, 1987). The program is here adapted for use on IBM PC and compatible microcomputers using either DOS or OS/2 operating systems. At least 512 KB of RAM is required for DOS operation. Instructions for operation are given at the beginning of the program listing in Appendix A.

Introduction

Ever since the publication of Bender and Perkins 1987 describing SEISRISK III, there have been increasing requests in the seismological and engineering communities for a version of the program adapted for use on microcomputers. Originally, this program was written for use on a Digital Equipment Corporation VAX 11/780 minicomputer with the VAX/VMS Operating System. The programming language selected was VAX/VMS FORTRAN. Since SEISRISK III was written for use by a small group of researchers, little attention was paid at first to the problems of portability to other types of computers. More specifically, both memory requirements and language aberations were ignored; who, after all, would worry about memory needs on a virtual memory machine? Likewise, it is natural to use every extension to a programming language that eases the programmer's job.

Unfortunately, these factors mitigate against being able to make a program portable, especially to machines of lesser computing power. In particular, personal computers based on the 8088 or 8086 processor operating under the Disk Operating System have severe restrictions placed on memory having only a 640 KB address space available.

The purpose of this project was to provide the general user with a version of SEISRISK III that will operate on almost any personal computer in the most efficient way while at the same time altering the original VAX program as little as possible.

This paper is meant to be a *supplement* to Bender and Perkins (1987) and will make references to it without notation. This implies that a copy of Bender and Perkins is necessary to the understanding of this report.

Operating Systems

Of the many operating systems available to users of personal computers, the two most popular are the IBM or Micro-Soft Disk Operating System (DOS) and the IBM or Micro-Soft Operating System/2 (OS/2). The versions given in this paper have been tested on both OS/2 and DOS but they ought to work equally well on any system with enough memory and a FORTRAN compiler. An example is any of the larger capacity Apple

computers. Since the author did not have one readily at hand, it was impossible to make the appropriate tests.

The version given in Appendix A, requires at least 640 KB of random access memory (RAM) and a mathematical co-processor for operation under DOS and at least 3 MB of RAM and a co-processor for operation under OS/2. Lesser quantities of RAM and the lack of a co-processor can be accomodated by some simple changes in source code described in the section on modifications. However, operation with much less than 512 KB RAM is quite impossible under "native" DOS, ie, DOS loaded alone as opposed to the DOS "window" in OS/2. The lack of a co-processor also increases the required memory as well as increasing running time.

Languages

The most important factor in determining the portability of any program is that of the language in which the source code is written. SEISRISK III was originally written in VAX/VMS FORTRAN as stated earlier. As a language, FORTRAN has been the most popular for decades with scientists and engineers who do programming. Because of this and because the earlier versions were written in FORTRAN, this author felt that some form of FORTRAN was the most portable way to share this program. The question was just what version of FORTRAN one should use.

The are at least 10 major versions of compilers readily available to the public all of which use a different set of standards. Fortunately, however, almost all use a ANSI FORTRAN-77 standard with various extensions. A few use a ANSI FORTRAN-66 standard also with various extensions. The latter compilers are both few in number and rapidly becoming obsolete. It was therefore decided that a version of SEISRISK III that adhered strictly to the ANSI X3.9–1978 FORTRAN-77 standard would be the most portable.

Unfortunately, FORTRAN-77 is so limited that it is difficult for the modern programmer to understand easily because it admits only six character names, uses only upper-case letters, &c. FORTRAN-77 also makes for some rather awkward constructions. It was therefore concluded that another version of SEISRISK III should be created employing a version of FORTRAN that has those extensions familiar to most scientists currently engaged in programming.

Finding such a standard is not easy if one attempts to avoid propriatary products that may have a rather limited distribution. After examining Lahey F77L, IBM Professional FORTRAN and FORTRAN/2, Ryan-McFarland FORTRAN, and various Micro-Soft compilers, the extensions to FORTRAN given in the IBM System Application Architecture, 1988 (SAA) guidelines are common to all. The SAA guidelines uses only features from the following list:

1. ANSI X3.9-1978 FORTRAN-77 standard.
2. INTEGER*2 types and type statements.

3. INTEGER*4 types and type statements.
4. LOGICAL*1 types and type statements.
5. LOGICAL*4 types and type statements.
6. REAL*4 types and type statements.
7. REAL*8 types and type statements.
8. COMPLEX*8 types and type statements.
9. COMPLEX*16 types and type statements.
10. Type statements with data initialization.
11. The IMPLICIT NONE statement.
12. Mixed expressions allowing double precision and complex values.
13. Format descriptor "Z" for hexadecimal.
14. Upper/lower case insensitivity.
15. ISA S61.1 Bit Manipulation Intrinsic functions.
16. COMPLEX*16 intrinsic functions.
17. The INCLUDE statement.
18. Names longer than six characters.
19. Non-leading underscore characters in names.
20. Association of character and non-character items in EQUIVALENCE and COMMON statements.
21. HFIX intrinsic functions.

Note that the DO WHILE and IMPLICIT statements other than IMPLICIT NONE are not admitted.

Two versions each for OS/2 and DOS source programs were made, one for the ANSI FORTRAN 77 standard the other for the SAA standard.

Modifications Made

Extensive alterations from the VAX/VMS version of the program were made. The most important of them are detailed below.

1. Interactive prompts and open statements were added to accomodate standard DOS and OS/2 file manipulation.

2. Format statements were altered to conform to the ANSI FORTRAN 77 and SAA standards. For the most part, this simply amounted to adding commas between specifier groups (not required in VAX/VMS FORTRAN) and changing the carriage control characters to provide double spacing and form feeds.

3. Calls to subprogram EXIT were replaced by a numbered STOP statements. Note that a *successful* program completion has been changed to STOP 7777.

4. The only change required to make SEISRISK III operate in the DOS environment is to change the dimension of the two ground-motion "accumulator" arrays, RAWBIN and SMOBIN. This was done with the introduction of a PARAMETER statement setting MAXAR to 701 rather than a value of 1601 for OS/2. This means that there are really only two versions of the source language which with changes in the parameter statement yield four different executable modules. All versions of the program were tested successfully on both the test data supplied with the program and with a much more complicated series of runs.

SEISRISK III, because it was designed to be used by only a few persons, is not very "user friendly". No attempt has been made in this distribution of the program to remedy this situation. It is planned that a future version of SEISRISK will be much easier to use.

DISTRIBUTION

Any scientist or engineer who is interested in obtaining a copy of SEISRISK III for a microcomputer should write the Books and Open- File Service Section, U.S. Geological Survey, M. S. 517, Denver Federal Center, Denver, CO 80225. Exceptionally, they may contact the author by either writing him at the address on page 1 or telephoning him at (303) 236–1579. The various versions of the program are available either as source code or as executable modules. DOS users should possess a machine having the full 640 KB of RAM and an appropriate mathematical co-processor. Although an executable module can be supplied for computers without a co-processor by special request, the run times are several times longer than those with one. This also applies to OS/2 users as well. With less than 640 KB of RAM the site areas must be so small that an inordinate number of runs are required. All versions are accompanied by a set of test data.

REFERENCES

American National Standards Institute, 1978. American National Standard Programming Language–FORTRAN: ANSI- X3.9-1978, New York, 398p.

Bender, B. and Perkins, D.M. 1987. SEISRISK III: A Computer Program for Seismic Hazard Estimation: U.S. Geological Survey Bulletin 1772, 48p.

International Business Machines Corporation, 1988. Systems Application Architecture, An Overview: Publication GC26–4341–3, San Jose, CA, 81pp.

APPENDIX A

SEISRISK III Source Program Listing for DOS using SAA Standards

The following is a source program for SEISRISK III following SAA standards for operation under DOS. It was compiled with the Lahey F77L compiler version 2.22 and the executable module is furnished with the Lahey runtime error routine. Other versions are similar.

```
      PROGRAM SEIS3
c                       SEISRISK III
c
c     computer program for determining ground motions
c         for use in seismic hazard mapping
c
c     ******Allows locations of future earthquakes in seismic
c        source zones to have a normal distribution*******
c
c     SEISRISK III is a revision of SEISRISK II, documented in Open
c        File Report 82-293.
c
c      See model description of SEISRISK III and additional
c         documentation in USGS Bulletin 1772
c
c     files:
c        file 15:  inputs
c        file 16:  outputs
c        file 02:  summary for plots--see text
c        file 03:  accelerations saved on file 03 for use in next
c                  run if next run is a continuation
c
c                          inputs
c
c  1. title-up to 80 characters   (free field)
c  2. isw, sigmax   (free field)
c       isw=0 new run (usual case)  isw=1 restart or continue
c             from previous run
c       sigmax=maximum standard deviation (km) for earthquake
c             location variability in seismic source zones for
c             this run.
c  3. prob,ntims, jtim(1),jtim(2)....jtim(ntims)
c    acceleration is sought for which there is probability prob of
c    not being exceeded in jtim(1), jtim(2), ...jtim(ntim)
c    years
c    prob = extreme probability in decimal
c    ntims = number of times for which calculation is done
c    jtim(i)  =  durations  (years) for which extreme motions are to be
c            calculated at the prob probability level, 1 <= i <= ntims
c  4. scale, dsw, sd, inos  (free field)
c       scale=scaling factor for ground motion boxes
c       scale=1 for motions .02 to 1.0--scale accordingly
c       dsw:  =1 if inputs are degrees and minutes
c             =0 if inputs are decimal degrees
c       sd=standard deviation in log acceleration
c            for acceleration variability around mean value
c       inos=1: divide magnitude interval in half and do calculations
c            at twice as many magnitudes (Assumes Gutenberg-Richter
c            b-value is same for all intervals.)
c         inos= any number other than 1: do calculations for original
c            magnitude intervals
c  5. x1,y1, x2,y2 (long,lat) in decimal degrees (free field)
c     transform great circle thru (x1,y1), (x2,y2), (0,0)
c     to equator for new coordiate system
c  6. fl1,ph1,fl2,ph2,phinc
c     (fl1,ph1) upper left    (fl2,ph2) lower right  (long, lat)
c     corners of seismic felt region for risk computation
c     rectangular region with sides parallel to arc
c     of great circle thru (x1,y1), (x2,y2) (0,0) --defined by 5.
c     other two sides perpendicular and thru given end points
c
c         + (fl1,ph1)
c        1    .
c     (x1,y1) 1     .
c    .             1     .
c        .              1     .
c           .              1     .
c              .              .     1
c           (fl2,ph2)+          (x2,y2)
c
c
c     area within dots represents felt region
c    '1' s represents line joining (x1,y1) (x2,y2)
c    phinc (long,lat) increment in degrees (in new coordinate
c       system) for which risk is to be computed
c
c  7. irow1, irow2, icol1, icol2   (free field)
c       starting and ending rows and columns for this run
c       accelerations computed for sites in these rows
c       and columns--may be subset of seismic felt region defined
c       in input 6.   Also irow1=irow2=icol1=icol2=0 permitted
c       if only individual sites on lines (input 8) are selected.
c  8. indv (free field)
c       number of line segments containing individual sites at
c       which acceleration is to be computed; zero if
c       only fixed grid is used.
c    If indv > 0 read next inputs, otherwise skip to input 9.
c     nvs (free field)
c       number of sites per line segment
c     xe1,ye1,xe2,ye2 (free field--indv lines, one pair per line)
c       end points (long, lat) of line segment:  sites will be
c       evenly spaced on line in new coordinate system
c    NOTE: NO SMOOTHING OF ACCELERATIONS AT SITES ON LINE FOR
c    ACCELERATIONS FROM EARTHQUAKES WITHIN AREAL SOURCE ZONES.
```

```
c    CALCULATIONS FOR UNIFORM SEISMICITY ONLY AT THESE SITES.
c  9. jent,mdis (free field)
c       jent=no of magnitudes for which acceleration is
c          tabulated as a function of distance   (maximum 8)
c       mdis=number of distances in attenuation table (maximum 20)
c  10. nam, tm (jent values of tm)
c       nam=identifier up to 10 characters-for example 'schn-seed'
c          identifies attenuation curves used
c       tm=magnitude for which table of distance versus acceleration
c          values follows  (free field)
c   *******magnitudes must be in descending order***
c  11. rtab(i), (atab(i,j),j=1,jent)   (mdis lines, i=1,mdis)
c     rtab(i)=ith tabular distance in kilometers from earthquake source
c     atab(i,j)=mean peak acceleration (or other ground motion parameter
c               at i th distance for j th magnitude.
c
c
c    source area inputs:
c
c    each source deck - -
c  1s. num, yrnoc, iprint, totl, dumid, als, bls, sigls
c     format(i2, f10.0, i2, f10.0, a4, 2f6.2, f5.2)
c     num=0 for seismic source areas if no averaging for earthquake
c           location uncertainty is to be done at end of ground
c           motion computations for this source zone
c      num=98 for seismic source zones if averaging for earthquake
c            location uncertainty is to be done at end of computations
c            for this source zone
c         =99 for 1st fault set
c         =0 for all fault sets after 1st
c         =99 at end of computation
c     yrnoc=number of years over which the earthquake occurrences
c              take place
c     iprint = -1  no statistics for this (intermediate) set of
c                occurrences
c            =1 statistical calculations and printout (on unit 16)
c            =2 same as=1 plus summary file for plot, etc (unit 2)
c            =3 omit printout; do summary file for plot (unit 2)
c    totl ignored here but used for faults
c    dumid = four character identifier for source zone or fault
c    als=earthquake location uncertainty standard deviation in km;
c        current value of als is ignored when num=0
c    bls, sigls ignored here but used for faults
c
c  2s. jseg,ifr,itot
c    jseg=num of pairs of quadrilatersl corner points in this source
c        set (jseg-1 quads).
c    seismicity to be apportioned by fractional area among itot sources.
c     ifr  = set number (ifr =1,2...itot in sequence)
c     itot=10 max;    total quad pairs for itot sources maximum 50
c  3s. jseg quadrilateral endpoint lines (two points per line):
c     quad corner points (left long, left lat), (right long, right lat),
c     in decimal degrees (if dsw=0)  degrees, minutes (if dsw=1)
c
c      x1,y1 _____________ x2,y2
c           1             1
c           1             1
c           1             1
c           1             1
c      x3,y3 _____________ x4,y4
c
c              1             1
c               1             1
c                1             1
c                 1             1
c                  1_____________1
c           x5,y5                 x6,y6
c
c     jseg=3 in this example:  quad endpoint pairs are
c                   (x1,y1)---(x2,y2)
c                   (x3,y3)---(x4,y4)
c                   (x5,y5)---(x6,y6)
c      subregions of a set are defined as shown
c      subregions of one set are separate from those of other sets
c      repeat 2s, 3s for itot sources
c  4s. noc(l)  l=1,lev  (lev=12f6.2)   (lev=12 maximum)
c      numbers of events expected in yrnoc years in each
c            magnitude interval for earthquake occurrences
c  5s. fm(l), l=1,lev  (12f6.2)
c      fm(l)=center of magnitude interval for which noc(l) occur
c
c       repeat 1s. thru 5s. for remaining sources
c
c          fault inputs:
c
c  1f.  num, yrnoc, iprint, totl, dumid, als, bls, sigls
c     parameters defined as in 1s. above
c   **note num=99 for first fault, =0 thereafter for successive faults
c     als,bls,sigls rupture length parameters
c     if als, bls non zero:
c          length=10**(als +bls*m +fr*sigl)
c          where m=magnitude;   sigl=standard deviation (in log length)
c     fr=(normally) 5 values in range (-2,+2)  if sigl non zero
c     fr=0 (1 value)--mean rupture length only if sigl=0
c     if als,bls,sigls=0 (or blank) previous values are used
c     if no previous values input, default values are used:
c          als=-1.085    bls=.389   sigls=.52
c     totl=distance between faults if this set of faults is a set
c          of "dummy" faults  used to approximate a uniform field
c          of faults.
c       totl=0 if individual fault or a number of well defined faults
c          are used.
c       totl determines whether ground motions for a site are to
c          be smoothed in distance or smoothed in magnitude
c          (smoothed in magnitude if totl=0)
c          (smoothed in distance (totl nonzero).
c  2f. jseg, ifr, itot  (free field)
c      jseg=num of fault segment end points to be connected
c          into single fault   (jseg-1 segments)
c      ifr, itot as defined in 2s
c  3f. (xl(i,i(fr),yl(i,i(fr), i=1,jseg)   (free field)
c         xl=long  yl=lat (degrees)  (ifr=fault number)
c       jseg=24 max          itot=26 max
c  4f. noc(l)   same as in 4s.
c  5f. fm(l) same as 5s.
c      repeat 2f, 3f for itot faults
c
c      repeat 1f. thru 5f. for remaining faults
c
c      end with num=99 (omit other inputs on line)
c * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
      parameter (maxar=701)
      character  idsw*16, nam*10,tytle*80,dumid*4
      real*4 fac(0:2500)
      dimension idsw(2),ibr(10),dtot(26)
      dimension xpt2(500),ypt2(500),xpt1(500),ypt1(500)
      dimension prls(10),rlls(10,25)
      dimension nrlsv(25),prlrl(10,25),rll(10,25)
      dimension disl(50),angl(50)
      common/ata/ata(20,25)
      common/inout/mdum(4),stsize,dum1
      common/slinds/sld1(4),perpa(4),sld2(4),sllq(4),
     & mina(4),perpaq(4),sldum2(8)
      common/extra/isid,shdis,distl(24),inear
      common /xlyl/xll(4),yl(4),xr(4),yr(4),xc(4),yc(4)
      common/itabs/dtab(105,25),maxa(25)
      common/magdis/jent,mdis
      dimension tmdif(10),atab(20,25),tm(10),atadif(20,25)
      common/rtb/rtab(20)
      common/xyarea/xsav(2,50),ysav(2,50),sarea(50)
c*******need maxdim=dimension of reg & of rawbin(maxdim,maxar)
      dimension regg(maxar)
      dimension reg(55),rawbin(55,maxar), smobin(55,maxar)
      common/rdis/rdis
      common/wds/max,maxp,maxsp
      common/ext/prob,ex,exl
      common/tims/ftim(20),jtim(20),ntims,nwt,iprint
c*** iscom = dimension of following sin,cos arrays
      common/sincos/sinx(500),cosx(500),siny(500),cosy(500)
      common/quad/maxq,ac(105),dumlog,aclog(105)
      dimension x(4),y(4),noc(24),fm(25)
      common/linps/dist(24,26),disq(24,26),jseg,jsegm,
     & xl(24,26),yp(24,26),coa(24,26),sia(24,26)
      common/lindis/dls(24),perp(24),min(24),persq(24)
      dimension jm(26), disrem(24)
      common/accel/gm(25),exdif(101),atah(20,25),aclim(2)
c arrays on previous line inserted for spreading out accelerations
      real noc
      integer dsw
      data iscom/500/
      data maxdim/55/
      data fac(0)/1./
      data imaxsq/400/
      data linset,line,nrls/0,0,0/
      data idsw/'decimal degrees ','degrees, minutes'/
      data r,pi2,pi/6378.,6.2831854,3.1415927/
      character*60 file15, file16, file02, file03
      print *, 'Enter input file name  '
      read(*,1002) file15
      print *, 'Enter output file name  '
      read(*,1002) file16
      print *, 'Enter auxilliary file name 02  '
      read(*,1002) file02
      print *, 'Enter auxilliary file name 03  '
      read(*,1002) file03
      open(15,file=file15,status='old')
      open(16,file=file16,status='unknown')
      open(2,file=file02,status='unknown',form='unformatted')
      open(3,file=file03,status='unknown',form='unformatted')
 1002 format(a60)
c     input on unit 15
      rewind 15
c     saved on unit 3 for later use or continuation
      rewind 3
      rad=pi/180.
      allerr=0.
      dlevsv=0.
      flevst=0.
      levno=0.
      tot5=0.
      read(15,20) tytle
   20 format(a80)
      write(16,30) tytle
   30 format(' ',a80)
c  isw tells whether to include data from a previous run, zero if no.
c  sigmax=maximum value of sigma (km, in earthquake location)
      read(15,*)isw,sigmax
      if(isw.eq.0) write(16,40)
   40 format(' isw=0: new run--no previous results included')
      if(isw.eq.1) write (16,50)
   50 format(' isw=1: run continuation; add to previous results')
      if(isw.eq.0) go to 70
      idrc=1
   60 read(3,end=70) (smobin(iqrp,idrc),iqrp=1,maxdim)
      idrc=idrc+1
      go to 60
   70 rewind 3
c    prob=extreme probability in decimal
c    ntims are number of 'jtims' for which calculation is done
c    jtims are the durations for which the extreme motions are
c    to be calculated at the prob extreme probability level
      read (15,*) prob, ntims, (jtim(jv),jv=1,ntims)
      if (prob.lt.1.) go to 90
      write(16,80) prob
   80 format(' prob =',f8.3,' must be decimal less than one')
      stop 5555
   90 continue
      write (16,100) prob,(jtim(jv),jv=1,ntims)
  100 format(' extreme probability ',f6.3
     &/'   for exposure times (years) ',2(10i5/))
      ex=-alog(prob)
      exl=alog(ex)
      do 110 jv=1,ntims
  110 ftim(jv)=jtim(jv)
c
      read(15,*) scale,dsw,sd,inos
      write(16,120) scale,idsw(dsw+1),idsw(dsw+1),sd
  120 format(' scale factor for ground motion "box" levels=',
     &f7.2/' coordinates input in ',a16,
     &/'    coordinates are printed in ',a16,
     &/' variability in attenuation, sigma=',f6.2)
c      scale=multiplicative factor for acceleration levels
c      dsw=0 if input is decimal degrees; =1 for degrees,minutes
c      sd=standard deviation for log acceleration
c   read (x1,y1), (x2,y2) =long,lat in dec. degrees of 2 points
c   to transform to equator
c      (x1,y1) to (0.,0)(x2,y2) to (dist. between points,0)
  130 read(15,*) x1,y1,x2,y2
      write (16,140) x1,y1,x2,y2
  140 format(' grid oriented parallel to great circle thru (',
     & f7.2,',',f6.2,'),(',f7.2,',',f6.2,')')
```

```
c  convert to decimal degrees unless dsw=0.
      if(dsw.ne.0) call cond(x1,x2,y1,y2)
      x1=x1*rad
      x2=x2*rad
      y1=y1*rad
      y2=y2*rad
c   transform to equator
      call inieqr(x2,y2,x1,y1)
c
      call toeqr(x1,y1,x1t,y1t)
      call toeqr(x2,y2,x2t,y2t)
c  read in upper left and lower right corners of affected rectangle
c  -- degrees, minutes ie 20.30=20 degrees,30 minutes (dsw=1)
c       (decimal degrees 20.50 = 20 1/2 degrees (dsw=0) )
c  opposite corners of gridded area (in latitude and longitude)
c  gridded area becomes rectangle surrounding new equator.
c  sites are at uniform increments in latitude and longitude
c  within this gridded area in new coordinate system.
c  in order to limit affected area you can specify
c   beginning and ending rows and columns as irow1 and 2
c   and icol1 and 2 (next input line)
c       ph1=phi=latitude
c       fl1=lambda=longitude
c
      read(15,*) fl1,ph1,fl2,ph2,phinc
      write (16,160) fl1,ph1,fl2,ph2
  160 format(' corners of gridded area-upper left=',f7.2,',',f7.2,
     &/23x,' lower right=',f7.2,',',f7.2)
c
c  convert to decimal degrees (unless dsw=0)
      if(dsw.eq.0) go to 170
      call cond(ph1,fl1,ph2,fl2)
      call condec(phinc)
  170 continue
c
c     determine number of subregions in map area and set up subscripting
c     irow=no of rows
c     icol=no of cols
      write(16,180) phinc,phinc
  180 format(' longitude increment=',f7.4,' (decimal degrees)',
     &/,' latitude increment =',f7.4,' (decimal degrees)')
      phinc=phinc*rad
      flinc=phinc
      dphinc=phinc*r
      fl1=fl1*rad
      ph1=ph1*rad
      fl2=fl2*rad
      ph2=ph2*rad
c    transform to equator and x-y rectangular coordinate system
c    convert to kilometers
      call toeqr(fl1,ph1,flam1,phi1)
      call to eqr(fl2,ph2,flam2,phi2)
      if(phi1.gt.0) go to 210
      if(phi2.gt.0) go to 220
c     apparent error in inputs or xformataion
  190 write(16,200)
  200 format(' probable input error--felt region upper left,',
     &' lower right'/' must be on opposite sides of new equator')
      stop 6666
  210  if(phi2.le.0) go to 230
      go to 190
  220 phs=phi1
      phi1=phi2
      phi2=phs
      fls=flam1
      flam1=flam2
      flam2=fls
c     have proper orientation--continue
  230 ph1=phi1
      if(flam1.lt.0) go to 240
      fl1=flam1
c    find total number of cols
      go to 260
  240 write(16,250)flam1,flam2
  250 format(' apparent error in xforming lat and long limits',
     & 'to eqr--',/,'  flam1=',f9.3,'  fl2=',f9.3)
      stop 4444
  260 continue
c   adjust to include area beyond edges for later smoothing
c     determine nbord=number of additional rows and columns at
c      which ground motions must be calculated to account for
c      earthquake location variability.
      nbord=2.*sigmax/dphinc+.01
      if(sigmax.eq.0.) go to 280
      if(nbord.ge.2) go to 280
      write (16,270) dphinc,sigmax,nbord
  270 format(' increments between sites =',e12.5,' too large for',
     & 'sigmax=',e12.5,' nbord=',i5)
      stop 3333
  280 ph1=ph1+nbord*phinc
      if (flam2.gt.fl1) flinc=-flinc
      fl1=fl1+nbord*flinc
      irow=(ph1-phi2)/phinc+1.01+nbord
      icol=(fl1-flam2)/flinc+1.01+nbord
      write(16,290)irow,icol,nbord
  290 format(' gridded region contains',i4,' rows, ',i4,' cols',
     &' including border ',i4,' rows and cols')
      read(15,*)irow1,irow2,icol1,icol2
c
c    limits rows and columns of felt points for which calculation is
c    to be made.
c
      write(16,300)irow1,irow2,icol1,icol2
  300 format(' for this run begin at row ',i3,' end row ',i3,
     & ', begin col ',i3,' end col ',i3)
c   for earthquake location variability, compute at more sites
c   calculate at additional rows and cols
      irow2=irow2+2*nbord
      icol2=icol2+2*nbord
      icols=icol2-icol1+1
      irows=irow2-irow1+1
      if(irow1.eq.0) irows=0
      if(icol1.eq.0) icols=0
      if(irow1.le.irow2) go to 320
      write(16,310) irow1
  310 format(' irow1 =',i4,'  too large--error stop')
      stop 8888
  320 if(icol1.le.icol2) go to 340
      write(16,330) icol1
  330 format(' icol1 =',i4,'   too large--error stop')
```

```
      stop 9999
  340 if(icols*irows.le.maxar) go to 370
  350 write(16,360)irows,icols,indvpt,maxdim
  360 format(' irows=',i4,' icols=',i4,' pts on line=',i4,
     &' exceeds maxdim of ',i4)
      stop 9999
  370 continue
c   read in number of line segments containing sites at which
c   calculation is to be done  (in addition to or instead of
c   computing for sites on a fixed grid)
c   zero if no line segments
      uply=ph1*r
      uplx=fl1*r
      boty=phi2*r
      botx=flam2*r
      write(16,380) uplx,botx,uply,boty
  380 format(' new coordinates (km) gridded area',
     &/,'       upper left=',2f9.2,'; lower right=',2f9.2)
      read(15,*)indv
      write(16,390) indv
  390 format(' sites are also located on ',i3,' line(s)')
      if (indv.eq.0) go to 460
c     calculate accelerations at nvs sites on each of indv lines
c     read in end point pairs (long,lat) xe1,ye1,xe2,ye2
c     for each of indv lines
c     nvs sites are evenly spaced on line*
c     nvs=2 gives accelerations at end points of line;
c     nvs=3 acceleration at 2 end points plus center of line,etc.
      read(15,*) nvs
      write(16,400) nvs
  400 format(i4,' sites per line')
      indvpt=0
      do 450 i=1,indv
      read(15,*) xe1,ye1,xe2,ye2
      write(16,410) i,xe1,ye1,xe2,ye2
  410 format(' line ',i2,' end points at ',f7.3,',',f7.3,' and ',f7.3,
     &',',f7.3)
      if(dsw.ne.0) call cond(xe1,xe2,ye1,ye2)
      xe1=xe1*rad
      xe2=xe2*rad
      ye1=ye1*rad
      ye2=ye2*rad
      call toeqr(xe1,ye1,xo1,yo1)
      call toeqr(xe2,ye2,xo2,yo2)
      fr=0.
      if (nvs.gt.1) frinc=1./(nvs-1)
      do 440 iq=1,nvs
      indvpt=indvpt+1
      if(indvpt.le.500) go to 430
      write(16,420)
  420 format(' more than 500 single points requested as sites',
     & 1x,'do 500')
      go to 460
  430 xl1=xo1+fr*(xo2-xo1)
      yp1=yo1+fr*(yo2-yo1)
      cosy(1)=cos(yp1)
      siny(1)=sin(yp1)
      sinx(1)=sin(xl1)
      cosx(1)=cos(xl1)
      call backw(1,1,xpt1(indvpt),ypt1(indvpt))
      xpt2(indvpt)=xl1*r
      ypt2(indvpt)=yp1*r
      fr=fr+frinc
  440 continue
  450 continue
  460 if(icols*irows+indvpt.gt.maxar) go to 350
      nrc=icols*irows
c
c     jent=number of magnitude entries in table of acceleration as a
c        function of magnitude and distance
c     mdis=number of distance entries
      read(15,*) jent,mdis
      if (jent.le.10) go to 480
      write (16,470) jent
  470 format(' jent=',i3,' too large--max 10 allowed')
      stop 1111
  480 if (mdis.le.20) go to 500
      write(16,490) mdis
  490 format(' mdis=',i4,'  too large--max 20 distances allowed')
      stop 0000
  500 continue
c     read identifier and magnitudes for which acceleration versus
c     distance table follows.  magnitudes must be in decreasing order
c
      read(15,*)nam,(tm(i),i=1,jent)
c    print magnitude headings for attenuation function.
      write (16,510) nam
  510 format(' attenuation function ',a10/30x,'magnitude')
      write(16,520) (tm(i),i=1,jent)
  520 format('   dist(km) ',f7.2,10f10.2)
c  read in table of acceleration vs distance values for set
c  of magnitudes (distance increasing in table)
      do 580 m=1,mdis
      read(15,*) rtab(m),(atab(m,j),j=1,jent)
      if(atab(1,1).lt.10) go to 540
      write (16,530) rtab(m),(atab(m,j),j=1,jent)
  530 format(1h ,f10.2,10f10.2)
      go to 560
  540 write(16,550) rtab(m),(atab(m,j),j=1,jent)
  550 format(1h ,f10.2,8f10.5)
c  take logs of tabular values since interp. is in log acc. etc.
  560 do 570 j=1,jent
  570 atab(m,j)=alog(atab(m,j))
  580 continue
c
c      define limits for acceleration boxes:   accelerations in the
c      range ac(j-1) < ac < ac(j) will be collected the jth box.
c      max=number of boxes;  acceleration values too large for
c      any of the box levels correspond to maxp=max+1
c      as here constituted boxes have limits ranging from
c     .02 to 1.0
c     For motions exceeding 1(g)., provide
c     appropriate distance vs ground table and select 'scale'
c     for multiplying boxes to get consistent range
c     or rewrite and recompile subroutine BOX
  600 call box(scale,sd)
c
c     basic step size for distances in source areas
      do 610 i=1,maxsp
  610 aclog(i)=alog(ac(i))
```

```
c     number of digits after decimal for print of
c     accelerations (subroutine out) determined by scale
c
c     identify rows and cols for file02
  620 ir2=irow2-2*nbord
      icl2=icol2-2*nbord
      write(2) irow1,ir2,icol1,icl2,indvpt,ntims,(jtim(jv),jv=1,ntims),
     &prob,sd
c     create table of sines and cosines for later back transformation
c     of felt points
      if(icol1.eq.0) go to 680
      yp1=(ph1-(irow1-1.)*phinc)
      xl1=(fl1-(icol1-1.)*flinc)
      irow2p=irow2+1
      icol2p=icol2+1
      if(irow2p.le.iscom) go to 650
  630 write(16,640) irow2p,icol2p,iscom
  640 format (i5,' rows,',i5,' cols needed in sincos arrays,',
     &' dimension=',i4,/' need to increase dimension')
      stop 2222
  650 if(icol2p.gt.iscom) go to 630
      do 660 ira=irow1,irow2p
      cosy(ira)=cos(yp1)
      siny(ira)=sin(yp1)
  660 yp1=yp1-phinc
      do 670 ii=icol1,icol2p
      cosx(ii)=cos(xl1)
      sinx(ii)=sin(xl1)
  670 xl1=xl1-flinc
  680 read(15,700)num,yrnoc,iprint,totl,dumid,als,bls,sigls
      numsav=num
  690 format(/'yrnoc= ',f6.0,' iprint=',i2,' for area ',a4)
  700 format(i2,f10.0,i2 ,f10.2,a4,2f6.2,f5.2)
c
      if(num.ne.99) go to 710
      if(line.eq.1) go to 2950
c  yrnoc=0 here if only source areas, no faults included
      if(yrnoc.eq.0.) go to 2950
      line=1
  710 write(16,690)yrnoc,iprint,dumid
  720 if(line.eq.1) go to 830
      if(num.eq.98) write (16,705) als
  705 format(' smooth for earthquake location uncertainty sigma=',
     1 f5.1)
      nbr=0
      ist=0
      iend=0
c  read identifiers for source area inputs
  730 read(15,*)jseg,ifr,itot
c     write(16,740) ifr,itot
  740 format(' source ',i2,' of ',i2)
c     jseg=number of pairs of quadrilateral end points in this
c        single source
c     itot=number of sources in this source area
c     ifr=identifies which source,  ifr=1,2,....itot
      if (itot.le.10) go to 760
      write(16,750) itot,jseg,ifr
  750 format(' itot =',i4,'  to large--max 10  jseg=',i4,
     & ' ifr=',i4)
      stop 3333
  760 ist=iend+1
      iend=ist+jseg-1
      if (iend.le.50) go to 780
      write(16,770)
  770 format('  too many quadrilaterals in source--max 50')
      stop 4444
c  read in boundaries for quadrilaterals for all subareas
c  in this source
c      read limits of seismic area x(i)=lambda(i),   y(i)=phi(i)
c      upper left(x,y) then  upper right(x,y) --1st quadrilateral
c      2nd line   lower left,lower right 1st quadrilateral=
c      2nd card   upper left,upper right   2nd quadrilateral,  etc.
c
  780 do 810 ii=ist,iend
      read(15,*)  (xsav(i,ii),ysav(i,ii), i=1,2)
  790 format(4f6.0)
  800 format(1h ,4f10.3)
      write(16,800)(xsav(i,ii),ysav(i,ii),i=1,2)
  810 continue
      if(ifr.eq.itot) go to 820
      nbr=nbr+1
      ibr(nbr)=iend
      go to 730
  820 num=iend
      go to 960
c     fault line read
  830 j=1
      write(16,840) totl
  840 format(' Distance between dummy faults=',f5.1)
      dsum=0.
c    fault inputs
  850 read(15,*) jseg ,ifr,itot
      write(16,860) ifr,itot
  860 format(' fault ',i2,' of ',i2)
c    jseg=number of end points of connected segments
c       for the current fault
c       jseg=one plus number of segments
c    itot=number of distinct faults in this zone
c    ifr identifies current fault, ifr=1,2,...itot
      if (jseg.le.24) go to 890
  870 write (16,880) jseg,itot
  880 format('  jseg=',i4,' (max 24) itot=',i4,' (max 26)-stop')
  890 if(itot.gt.26) go to 870
c    read in end points long,lat of each segment
      read(15,*)(xl(i,j),yp(i,j),i=1,jseg)
      write(16,900)(xl(i,j),yp(i,j),i=1,jseg)
  900 format(1h ,4(f10.2,f8.2,','))
      jm(j)=jseg
      do 920 i=1,jseg
c    convert to decimal degrees unless dsw=0
      if(dsw.eq.0) go to 910
      call condec(yp(i,j))
      call condec(xl(i,j))
  910 continue
c   transform to equator--rectangular coordinates
      yin=yp(i,j)*rad
      xin=xl(i,j)*rad
      call to eqr(xin,yin,xout,yout)
      xl(i,j)=xout*r
      yp(i,j)=yout*r
```

```
  920 continue
c   compute line parameters, lengths for fault segments
      jsegm=jseg-1
      dtot(j)=0.
      do 930 i=1,jsegm
      xdelta=xl(i+1,j)-xl(i,j)
      ydelta=yp(i+1,j)-yp(i,j)
      disq(i,j)=xdelta*xdelta+ydelta*ydelta
      dist(i,j)=sqrt(disq(i,j))
      coa(i,j)=(xl(i+1,j)-xl(i,j))/dist(i,j)
      sia(i,j)=(yp(i+1,j)-yp(i,j))/dist(i,j)
  930 dtot(j)=dtot(j)+dist(i,j)
      dsum=dsum+dtot(j)
      if(ifr.eq.itot) go to 950
  940 j=j+1
      go to 850
c     read in num of occurrences for each magnitude (12f6.2)
c     both area and fault input
  950 jtot=j
  960 read (15,1170) (noc(i),i=1,12)
      lev=12
  970 if(noc(lev).ne.0.0) go to 990
      lev=lev-1
      if(lev.gt.0) go to 970
      write(16,980)
  980 format(' apparent input err for no of occurrences at each level')
      go to 2990
  990 continue
      write(16,1000)lev
 1000 format(' nr of levels of seismicity = ',i2)
c
c  read in corresponding mag. interval center points. (lev of them)
c
 1010 format(' before normalizing to rate/year')
 1020 format( ' earthquake rate / year')
      read(15,1170)(fm(i),i=1,lev )
c  magnitudes must be evenly spaced and in increasing order.
c  if not entered in increasing order, program reverses order
c  of magnitudes and corresponding earthquake occurrences
c  at each magnitude.
c  Don't confuse this with the original table of ground
c   motions as a function of magnitude and distance.
c   The magnitudes in that table must be read in decreasing
c   (rather than increasing) order.
      if(fm(1).lt.fm(2)) go to 1040
      if(lev.eq.1) go to 1040
      levh=lev/2
      ll=lev
      do 1030 l=1,levh
      sav=fm(l)
      fm(l)=fm(ll)
      fm(ll)=sav
      sav=noc(l)
      noc(l)=noc(ll)
      noc(ll)=sav
 1030 ll=ll-1
 1040 continue
      if(lev.gt.1) go to 1050
c    we cannot calculate beta from numbers of earthquakes
c    if only one level--so give a default beta so we
c    can later spread out
      beta=-2.0
      fm(2)=fm(1)+.6
      delm=.6
      go to 1060
 1050 continue
      dlev=abs(fm(1)-fm(2))
      beta=alog( noc(2)/noc(1) )/dlev
 1060 write(16,1070)dumid, beta
 1070 format(1x,a4,' beta=',f8.4)
      if(yrnoc. eq. 1.) go to 1090
      write(16,1010)
      write(16,1100) (noc(ll),ll=1,lev)
c     this is a normalization to rate per year
      do 1080 lm=1,lev
      noc(lm)=noc(lm)/yrnoc
 1080 continue
 1090 write(16,1020)
      write(16,1100)(noc(l),l=1,lev)
 1100 format(' occurrences=',12f10.6)
      write(16,1110)(fm(l),l=1,lev)
 1110 format('  magnitudes=',12f10.2)
      summoc=0.
      do 1120 lm=1,lev
 1120 summoc=summoc+noc(lm)
c     evaluate at twice as many magnitudes as initially read
c     in if inos=1
      if(inos.ne.1)  go to 1140
      dlev=(fm(2)-fm(1))/2.
      zac=exp(abs(beta)*dlev)
      noc(1)=noc(1)*zac/(1.+zac)
      summoc2=noc(1)
      lev=2*lev
      fm(1)=fm(1)-dlev/2.
      do 1130 ll=2,lev
      fm(ll)=fm(ll-1)+dlev
      noc(ll)=noc(ll-1)/zac
 1130 summoc2=summoc2+noc(ll)
      write(16,1150) summoc,summoc2
      write(16,1100) (noc(l),l=1,lev)
      write(16,1110) (fm(l),l=1,lev)
 1140 continue
 1150 format(' total eqs=',2e12.5)
 1160 continue
 1170 format(12f6.2)
c
      if (line.eq.0) go to 1250
c    determine which rupture length magnitude relationship to use
      if(als.eq.0)   go to 1240
      al=als
      bl=bls
      sigl=sigls
c     compute break length
 1180 write(16,1190) al,bl,sigl
 1190 format(' fault rupture length parameters al=',f7.3,' bl=',f7.3,
     &'  sigl=',f6.2)
      linset=1
      if (sigl.gt.0) go to 1230
c     which single break default do we want
c     if bl=0, al non zero use old Algermissen-Perkins default values
```

```
c     use parameter values supplied for one break only
 1200 do 1210 i=1,lev
 1210 rlls(1,i)=10.**(al+bl*fm(i))
 1220 prls(1)=1.
      nrls=1
      go to 1350
c    set up rupture lengths and probabilities as fct of magnitude
c    nrls lengths per magnitude
c    al, bl, sigl non zero
 1230 call pbreak(fm,al,bl,sigl,rlls,prls,lev,nrls)
      go to 1350
 1240 if (nrls.eq.1) go to 1200
      if(linset.eq.1) go to 1230
c     compute default values
      al=-1.085
      bl=.389
      sigl=.52
      linset=1
      go to 1180
c  convert quad corner points to decimal degrees  (if dsw =1)
c
 1250 do 1280 ii=1,num
      do 1280 i=1,2
      if(dsw.eq.0) go to 1260
      call condec(xsav(i,ii))
      call condec(ysav(i,ii))
 1260 continue
      xin=xsav(i,ii)*rad
      yin=ysav(i,ii)*rad
      call to eqr(xin,yin,xout,yout)
      xsav(i,ii)=xout*r
      ysav(i,ii)=yout*r
c     write(16,1270) xsav(i,ii),ysav(i,ii),ii
 1270 format(' xsav,ysav=',2f9.3,' ii=',i3)
 1280 continue
c
 1290 nm=num-1
      stot=0.
      do 1330 ii=1,nm
      if(nbr.eq.0) go to 1310
      do 1300 iq=1,nbr
      if(ii.eq.ibr(iq)) go to 1330
 1300 continue
c    set line parameters for subsource
 1310 call setreg(ii)
      do 1320 i=1,2
      x(i)=xsav(i,ii)
      x(i+2)=xsav(i,ii+1)
      y(i)=ysav(i,ii)
 1320 y(i+2)=ysav(i,ii+1)
c    find area of subregion ii
      s1=abs(x(1)*(y(2)-y(3))+x(2)*(y(3)-y(1))+x(3)*(y(1)-y(2)))
      s2=abs(x(4)*(y(2)-y(3))+x(2)*(y(3)-y(4))+x(3)*(y(4)-y(2)))
      sarea(ii)= (s1+s2)/2.
      stot=stot+sarea(ii)
 1330 continue
      eua=noc(1)
c   find earthquake rate per unit area for lowest magnitude
c   rate will be printed but not used in program
      delm=(fm(2)-fm(1))/2.
      fm1=fm(1)-delm
      fm2=fm(1)+delm
      eua=eua/stot
      write(16,1340) dumid,stot,eua,fm1,fm2
 1340 format(1h ,a4,' area=',f9.0,' sq km, rate/sq km=',e12.5,
     & ' for mags ',f5.2,'-',f5.2)
 1350 dlev=fm(1)-fm(2)
c    create table of log accelerations for set of magnitudes
c    in (tm) for the original set of mdis distances as a function
c    of distance
c    next three lines test whether table has already
c     been computed--if so skip over
      if(lev.gt.levno) go to 1360
      if(fm(1).ne.flevst) go to 1360
      if(dlev.eq.dlevsv) go to 1520
 1360 do 1370 ill=2,jent
 1370 tmdif(ill)=tm(ill)-tm(ill-1)
      do 1420 ll=1,lev
      do 1380 il=2,jent
      if(fm(ll).gt.tm(il)) go to 1390
 1380 continue
      il=jent
 1390 ilm=il-1
      do 1400 j=1,mdis
 1400 ata(j,ll)=atab(j,ilm)+(fm(ll)-tm(ilm))/tmdif(il)
     & *(atab(j,il)-atab(j,ilm))
      do 1410 j=2,mdis
 1410 atadif(j,ll)=ata(j,ll)-ata(j-1,ll)
 1420 continue
c     find distance corresponding to each acceleration in
c     table ac (set up in subroutine box) for each magnitude
      do 1510 l=1,lev
      do 1500 j=1,maxp
      do 1430 i=2,mdis
      if(aclog(j).gt.ata(i,l)) go to 1450
 1430 continue
c     beyond table
 1440 dtab(j,l)=rtab(mdis)
      go to 1500
 1450 if(atadif(i,l).ne.0) go to 1490
c     no change in acceleration at this distance
 1460 if(i.eq.2) go to 1480
      write(16,1470) i,l,j
 1470 format(' apparent error indist-table i,l,j=',3i4)
      stop 5555
 1480 dtab(j,l)=0.
      go to 1500
 1490 dtab(j,l)=rtab(i)-(rtab(i)-rtab(i-1))*(ata(i,l)-aclog(j))
     &/atadif(i,l)
      if(dtab(j,l).lt.0) dtab(j,l)=0.
 1500 continue
 1510 continue
      dlevsv=dlev
      levno=lev
      flevst=fm(1)
 1520 continue
c    add half values of mag for later spreading out
      dmag=fm(2)-fm(1)
      bmag=fm(1)-dmag/2.
      levp=lev+1
```

```
      do 1530 i=1,levp
      gm(i)=bmag
 1530 bmag=bmag+dmag
      do 1570 ll=1,levp
      do 1540 il=2,jent
      if(gm(ll).gt.tm(il)) go to 1550
 1540 continue
      il=jent
 1550 do 1560 j=1,mdis
      atah(j,ll)=atab(j,il-1)+(gm(ll)-tm(il-1))/tmdif(il)
     & *(atab(j,il)-atab(j,il-1))
 1560 continue
 1570 continue
c     take exp differences for later accel spread out
c     the "exdif" values are used in subroutine "spread"
      if (lev.eq.1) betad=2.0
      if(lev.ne.1) betad=alog(noc(1)/noc(2))
      beta=betad/dmag
      fmul=1./(1.-exp(-beta*dmag))
      frak=exp(-beta*dmag/100.)
      frm=frak
      exdif(1)=0.
      do 1580 i=2,101
      exdif(i)=(1.-frm)*fmul
 1580 frm=frm*frak
      dtmax=dtab(1,1)
      if(dlev.lt.0) dtmax=dtab(1,lev)
c     find entry for each magnitude in distance-acceleration table
c     for zero distance (highest acceleration for that magnitude)
      do 1600 l=1,lev
      j=maxp
 1590 if(dtab(j,l).ne.0.) go to 1600
      j=j-1
      go to 1590
 1600 maxa(l)=j
c
      irow1s=irow1
      irow2s=irow2
      icol1s=icol1
      icol2s=icol2
      idrc=0
      do 2770 ispt=1,2
      if(ispt.eq.1) go to 1610
c     case of calculation done for single points on line
      if(indvpt.eq.0) go to 2770
      irow1s=1
      irow2s=1
      icol1s=1
      icol2s=indvpt
      go to 1620
 1610 if(irow1.eq.0) go to 2770
 1620 do 2760 ira=irow1s,irow2s
c
      ymid=(ph1-(ira-1.)*phinc)*r
      do 2750 ii=icol1s,icol2s
c
      idrc=idrc+1
      if(ispt.eq.2) go to 1630
      fii=ii
      xmid=(fl1-(fii-1.)*flinc)*r
      go to 1640
 1630 xmid=xpt2(ii)
      ymid=ypt2(ii)
 1640 if(isw.eq.0) go to 1660
c
c     after first time thru, read accelerations calculated
c     for previous sources at this site (saved in 2 dimensional
c     array rawbin) into reg (1 dimensional) array.
c
      do 1650 iqrp=1,maxdim
 1650 reg(iqrp)=rawbin(iqrp,idrc)
      go to 1680
c
c    first time thru clear reg array if not continuation run
c
 1660 do 1670 ll=1,maxp
 1670 reg(ll)=0.
      regg(idrc)=9999.999
c     set regg to min dist from fault
c
c     for fixed  affected area in row ira, col ii
c
c      source area computation
 1680 if(line.eq.1) go to 2120
      do 2110 k=1,nm
      if(nbr.eq.0) go to 1700
      do 1690 iq=1,nbr
      if(k.eq.ibr(iq)) go to 2110
 1690 continue
 1700 approx=0.
      do 1710 i=1,2
      xll(i)=xsav(i,k)
      xll(i+2)=xsav(i,k+1)
      yl(i)=ysav(i,k)
      yl(i+2)=ysav(i,k+1)
 1710 continue
      xr(3)=xll(1)
      yr(3)=yl(1)
      xr(1)=xll(2)
      yr(1)=yl(2)
      xr(2)=xll(4)
      yr(2)=yl(4)
      xr(4)=xll(3)
      yr(4)=yl(3)
      aread=sarea(k)/stot
c     determine whether site is inside or outside k th quadrilateral
c     subregion.  find distance rc to closest endpoint
c     and rf to farthest end point.  Set up table of distance versus
c     arc length for distances in the range rc to rf using
c     subroutines outsid or inside.
c      itst=no of radii at fixed increments between rc and rf
c           at which arc lengths are to be evaluated in table
c      itst=7   set in subroutine rrisk--recompile to change
      call rrisk(xmid,ymid,in,k,dis,rc,rf,itst)
      if(in.eq.1) go to 1740
      if(rc.lt.dtmax) go to 1730
      do 1720 ll=1,lev
 1720 reg(1)=reg(1)+aread*noc(ll)
      go to 2110
 1730 rcs=rc
```

```
      disl(1)=rc
      angl(1)=0.
      itst=itst+1
      ibeg=2
      go to 1750
 1740 disl(1)=0.
      angl(1)=0
      disl(2)=rc
      angl(2)=pi2*rc
      ibeg=3
      if(rc.eq.0) ibeg=2
      rcs=0.
      itst=itst+2
c   compute distance versus arc length table for even increments
c   in distance
 1750 do 1810 iii=ibeg,itst
      if(in.eq.0.) go to 1760
      call inside(xmid,ymid,k,dis,angle)
      go to 1780
 1760 continue
      call outsid(xmid,ymid,k,dis,angle)
 1770 format('err outside for ira=',i4,' ii=',i4)
 1780 continue
      disl(iii)=dis
      angl(iii)=angle*dis
 1810 dis=dis+stsize
      itst=itst+1
      disl(itst)=rf
      angl(itst)=0.
c   compute distance versus arc length table for distances
c   to selected vertices and perpendiculars to edges
      rcsq=rc*rc
      rfsq=rf*rf
      do 1940 i=1,4
      ire=0
      if(in.eq.0) ire=2
      if(sllq(i).le.rcsq+1.) go to 1930
      if(sllq(i).ge.rfsq-1.) go to 1930
      dis=sqrt(sllq(i))-.001
      if(dis.le.rc) goto 1930
c     add angle, distance corresponding to vertex
      if (in.eq.0) go to 1830
      call inside(xmid,ymid,k,dis,angle)
      go to 1860
 1820 if(mina(i).ne.3) go to 1940
c   closest side next
      dis=abs(perpa(i))
      if(dis.le.rc+.5) go to 1940
      if(dis.ge.rf-.5) go to 1940
      dis=dis-.001
      call inside(xmid,ymid,k,dis,angle)
      ire=1
      go to 1860
 1830 continue
      call outsid(xmid,ymid,k,dis,angle)
      go to 1860
 1840 if(mina(i).ne.3) go to 1940
      dis=abs(perpa(i))
      ire=1
      do 1850 iql=1,4
      if(abs(sllq(iql)-perpaq(i)).le. 2.) go to 1940
 1850 continue
      if(dis.le.rc+1.) go to 1940
      if(dis.ge.rf-1.) go to 1940
      dis=dis-.001
      call outsid(xmid,ymid,k,dis,angle)
 1860 continue
      do 1890 ll=1,itst
      if(dis-disl(ll)) 1900,1920,1890
 1890 continue
c   insert new distance, angle
 1900 jk=itst
      do 1910 mm=ll,itst
      angl(jk+1)=angl(jk)
      disl(jk+1)=disl(jk)
 1910 jk=jk-1
      disl(ll)=dis
      angl(ll)=angle*dis
      itst=itst+1
 1920 if(ire.eq.0) go to 1820
      if(ire.eq.2) go to 1840
      go to1940
 1930 if(in)1840,1840,1820
 1940 continue
      itst=itst+1
      disl(itst)=rf
      angl(itst)=0.
c   for each magnitude, source, determine fractional source area giving
c   acceleration a:  ac(i-1) < a < ac(i)  for each entry in acceleration
c   table ac.  equivalent to finding fractional source area at distance
c   from site between successive entries in acceleration-distance
c   table:  dtab(i,l) < d < dtab(i-1,l) for magnitude l.
c   interpolates for distance is disl vs angl table
c   area=integral(angle*dist*delta(dist))  from dtab(i-1,l) to dtab(i,l)
      do 2090 ll=1,lev
      stnoc=noc(ll)/stot
      iswd=0
      maxm=maxa(ll)
      if(maxm.le.0.) go to 1960
      approx=0.
      if(in.eq.0) go to 1950
      jlo=maxm
      go to 1980
 1950 if(rcs.lt.dtab(1,ll)) go to 1970
 1960 reg(1)=reg(1)+aread*noc(ll)
      go to 2080
 1970 jlo=indpt(dtab(1,ll),rcs,maxm)
 1980 j=jlo
      iqs=2
      do 2060 jk=1,jlo
      if(jk.gt.1) go to 1990
      dbot=rcs
      ddis=dtab(j,ll)
      if(dtab(j,ll).lt.rf) goto 2010
      reg(j+1)=reg(j+1)+sarea(k)*stnoc
      go to 2080
 1990 if(dtab(j,ll).lt.rf) go to 2000
      dbot=dtab(j+1,ll)
      ddis=rf
      iswd=1
```

```
      go to 2010
 2000 ddis=dtab(j,ll)
      dbot=dtab(j+1,ll)
 2010 anarea=0.
      do 2020 iq=iqs,itst
      if(dbot.le.disl(iq)) go to 2030
 2020 continue
      iq=itst
c    distance exceeds last value-error since table set up to handle this
 2030 dtop=disl(iq)
      if(disl(iq).gt.ddis) dtop=ddis
      dmid=(dtop+dbot)/2.
      delr=dtop-dbot
      fr=(dmid-disl(iq-1))/(disl(iq)-disl(iq-1))
      ang=angl(iq-1)+(angl(iq)-angl(iq-1))*fr
      anarea=anarea+ang*delr
      if(dtop.eq.ddis) go to 2050
      dbot=dtop
      iq=iq+1
      if(iq.le.itst) go to 2030
      write(16,2040)
 2040 format(' err in arc area computation')
      go to 2030
c    interpolate in angle , distance table
 2050 approx=approx+anarea
      reg(j+1)=reg(j+1)+anarea*stnoc
      j=j-1
      if(iswd.eq.0) go to 2060
      arerr=(approx-sarea(k))/sarea(k)
      allerr=allerr+1.
      if (abs(arerr).lt. .05) go to 2090
      tot5=tot5+1.
      if(abs(arerr).le. .10) go to 2090
      write(16,2100) arerr,ira,ii,rc,rf,k,ll,in,dumid
      go to 2090
 2060 iqs=iq
c     need to add remaining area if any to lowest acceleration
      dif=sarea(k)-approx
      if(dif.lt.0) go to 2080
 2070 reg(1)=reg(1)+dif*stnoc
 2080 approx=sarea(k)
 2090 continue
 2100 format(' int err=',f6.3,' row ',i2,' col ',i2,' rc=',
     & f7.2,' rf=',f7.2,' k=',i2,' ll=',i2,' in=',i1,1x,a4)
 2110 continue
      go to 2730
c     fault computation
 2120 do 2720 j=1,jtot
      jseg=jm(j)
      jsegm=jseg-1
      call cldis(xmid,ymid,j)
      do 2160 ll=1,lev
c     fill in rupture lengths--make  certain that rupture length
c     does not exceed total fault length
      nrl=nrls
      do 2130 irl=1,nrl
      rll(irl,ll)=rlls(irl,ll)
 2130 prlrl(irl,ll)=prls(irl)
      if(rll(1,ll).lt.dtot(j)) go to 2160
      rll(1,ll)=dtot(j)
      if(nrl.eq.1) go to 2160
 2140 if(rll(2,ll).lt.dtot(j)) go to 2160
      prlrl(1,ll)=prlrl(1,ll)+prlrl(2,ll)
      nrl=nrl-1
      if(nrl.eq.1) go to 2160
      do 2150 i=2,nrl
      prlrl(i,ll)=prlrl(i+1,ll)
 2150 rll(i,ll)=rll(i+1,ll)
      go to 2140
 2160 nrlsv(ll)=nrl
      pt1=sqrt(shdis)
      if(pt1.lt.regg(idrc)) regg(idrc)=pt1
      if (isid.eq.0) go to 2490
c    distance is monotonically increasing for all segments
c    (the decreasing case has been converted to increasing)
c    or single shortest distance is interior to fault
      do 2480 ll=1,lev
      nrl=nrlsv(ll)
      frcon=dtot(j)/dsum*noc(ll)
      maxm=maxa(ll)
      jlo=indpt(dtab(1,ll),pt1,maxm)
      if (jlo.gt.0) go to 2170
c   distance beyond table--lump acceleration in first box
      reg(1)=reg(1)+frcon
      go to 2480
 2170 continue
      if (isid.eq.1) go to 2190
 2180 ibeg=1
      ifin=jsegm
      ixt=1
      dnear=dtot(j)
      inear=jsegm
      go to 2310
 2190 ibeg=1
      ifin=inear
      ixt=2
c   dnear=distance along fault from one end to point on fault
c   closest to the site
c   dnear=total length of segments 1,2,...inear
c   inear defined sub cldis--in common/extra/
      dnear=0.
      do 2200 iq=1,inear
 2200 dnear=dnear+distl(iq)
      dfar=dtot(j)-dnear
      totac=0.
      do 2220 irl=1,nrl
      smd=rll(irl,ll)
      if(smd.lt.dtot(j)) go to 2210
c     break is entire segment
      totac=totac+frcon*prlrl(irl,ll)
      sumr=totac
      go to 2220
 2210 smd2=smd/2.
c    find fraction of fault at closest distance to site
c    for interior site
      arcn=smd2
      arcf=smd2
      if(dnear.lt.smd) arcn=dnear-smd2
      if(dfar.lt.smd) arcf=dfar-smd2
      arct=arcn+arcf
```

```
      if(arct.lt. .0001) go to 2220
      totac=totac+frcon/(dtot(j)-smd)*arct*prlrl(irl,ll)
 2220 continue
      sumr=totac
      totacs=totacs+totac
      jlop=jlo+1
      if(totl.gt.0) go to 2230
c     spread dummy faults over "totl" km distance if "totl" > 0
c      if "totl" = 0, spread over magnitude instead
      call spread(reg,jlop,ll,totac,pt1)
      go to 2310
 2230 width=totl/2.
      sumr=sumr/totl
      add=0
      if(abs(pt1).gt.width) go to 2280
      qtot=width+abs(pt1)
      dsvs=0.
      jlo2=maxm
      do 2270 iqk=1,2
      jrp=jlo2
      do 2250 jk=1,jlo2
      dl3 =dtab(jrp,ll)
      if(dl3.gt.qtot) go to 2260
      cary=dl3-dsvs
      reg(jrp+1)=reg(jrp+1)+sumr*cary
      add=add+sumr*cary
      jrp=jrp-1
 2240 format(' add=',e12.5)
 2250 dsvs=dl3
      jrp=0
 2260 cary=qtot-dsvs
      add=add+sumr*cary
      reg(jrp+1)=reg(jrp+1)+sumr*cary
      dsvs=0.
      qtot=width-abs(pt1)
 2270 jrp=jlo2
      go to 2310
 2280  dstart=abs(pt1)-width
      qtot=totl+dstart
      jlo2=indpt(dtab(1,ll),dstart,maxm)
      dsvs=dstart
      jrp=jlo2
      do 2290 jk=1,jlo2
      dl3=dtab(jrp,ll)
      if(dl3.gt.qtot) go to 2300
      cary=dl3-dsvs
      reg(jrp+1)=reg(jrp+1)+sumr*cary
      add=add+sumr*cary
      jrp=jrp-1
 2290 dsvs=dl3
      jrp=0
 2300 cary=qtot-dsvs
      reg(jrp+1)=reg(jrp+1)+cary*sumr
c   find frac of fault in distance range dtab(jr,ll) < d <dtab(jr-1,ll)
c   for each magnitude ll
 2310 do 2470 ithru=1,ixt
      jr=jlo
      ilo=1
      iii=ibeg
      casum=0.
      irl=1
      dsav=-dls(iii)
 2320 frac=0.
      jk=1
      smd=rll(irl,ll)
      if(smd.lt.dtot(j)) go to 2330
      if(ixt.eq.2)go to 2450
c     add in entire fault length
      totac=frcon*prlrl(irl,ll)
      jlop=jlo+1
      call spread(reg,jlop,ll,totac,pt1)
      go to 2450
 2330 cuml=dnear-smd
      if(cuml.le.0) go to 2450
      do 2340 iq=irl,nrl
 2340 frac=frac+prlrl(iq,ll)/(dtot(j)-rll(iq,ll))
      frac=frac*frcon
      dlen=0.
      do 2350 iq=ibeg,ifin
      disrem(iq)=distl(iq)
      dlen=dlen+disrem(iq)
      if (dlen.ge.cuml) go to 2370
 2350 continue
      write(16,2360)dlen,ibeg,ifin,cuml
 2360 format(' error for fault line dlen=',e12.5,' ibeg=',i3,
     &' ifin=',i3,' cuml=',e12.5)
      stop 6666
 2370 disrem(iq)=cuml-dlen+disrem(iq)
 2380 ifins=iq
 2390 dend=disrem(iii)-dls(iii)
c     dsav=starting distance along segment iii
 2400 continue
      do 2410 jk=ilo,jlo
      dl3=sqrt(dtab(jr,ll)**2-persq(iii))
      if(dl3.gt.dend) go to 2440
      carea=dl3-dsav
      casum=casum+carea
      reg(jr+1)=reg(jr+1)+carea*frac
      jr=jr-1
 2410 dsav=dl3
c    at top of table--remaining dist beyond table
 2420 do 2430 iq=irl,nrl
      cuml=dnear-rll(iq,ll)
 2430 reg(1)=reg(1)+(cuml-casum)*prlrl(iq,ll)*frcon
     & /(dtot(j)-rll(iq,ll))
      go to 2460
 2440 carea=dend-dsav
      casum=casum+carea
      reg(jr+1)=reg(jr+1)+carea*frac
      dsav=dend
c      at end of segment iii
      if(iii.eq.ifins)  go to 2450
      iii=iii+1
      dsav=-dls(iii)
      ilo=jk
      go to 2390
 2450 irl=irl+1
      ilo=jk
      if(irl.le.nrl) go to 2320
      if(ithru.eq.ixt) go to2480
```

```
 2460 irl=1
      ibeg=inear+1
      ifin=jsegm
      dnear=dtot(j)-dnear
 2470 continue
 2480 continue
      go to 2720
c     more than one turning point
c     brute force.  find contribution for single break
c     increment break centers by distance deltal along fault from
c     one end to other.
 2490 continue
      deltal=10.
      if(pt1.gt.100) deltal=20.
      if(pt1.lt.15.) deltal=5.
      do 2710 ll=1,lev
      nrl=nrlsv(ll)
      do 2710 irl=1,nrl
      smd=rll(irl,ll)
      prnoc=noc(ll)*prlrl(irl,ll)
      dj=dtot(j)-smd
      nd=dj/deltal+1.5
      if(nd.gt.1) go to 2500
      nd=2
      if(dj.gt.1.) go to 2500
c     break is entire fault length
      maxm=maxa(ll)
      jlo=indpt(dtab(1,ll),pt1,maxm)
      add=prnoc*dtot(j)/dsum
      jlop=jlo+1
      call spread(reg,jlop,ll,add,pt1)
      go to 2710
 2500 continue
      dmvp=prnoc*dtot(j)/(dsum*(nd-1))
      del=dj/(nd-1)
      do 2700 m=1,nd
      dst=(m-1)*del
      do 2510 i=1,jsegm
      if(dst.lt.dist(i,j)) go to 2530
      dst=dst-dist(i,j)
 2510 continue
      write(16,2520)dst,j
 2520 format(' apparent error fault dist--dst=',e12.5,'  j=',i4)
      go to 2990
 2530 ddsav=1.e8
      dfin=dst+smd
 2540 dover=0.
      if(dfin.le.dist(i,j)) go to 2550
      dover=dfin-dist(i,j)
      dfin=dist(i,j)
 2550 if(dls(i).lt.dst) go to 2560
      if(dls(i).gt.dfin) go to 2570
c      shortest distance to segment at point within segment
      dd=persq(i)
      go to 2580
c      shortest dist at start of segment
 2560 dd=(dls(i)-dst)**2+persq(i)
      go to 2580
c      shortest distance at end of segment
 2570 dd=(dls(i)-dfin)**2+persq(i)
 2580 continue
      if(dd.lt.ddsav) ddsav=dd
      if(abs(dover).lt.0.001) go to 2600
      dst=0
      i=i+1
      dfin=dover
      if(i.le.jsegm) go to 2540
 2590 format(' i too large i=',i2)
      write(16,2590) i
      stop 8888
 2600 dis=sqrt(ddsav)
c  have distance to subsegment of arc of fault
c
      if(m.gt.1) go to 2610
      jen2=indpt(dtab(1,ll),dis,maxa(ll))
      disav=dis
      go to 2700
 2610 disdif=dis-disav
      if(disdif.eq.0) go to 2620
      dfac=dmvp/disdif
c    spread out contributions into accel boxes
      if(disdif) 2630,2620,2660
 2620 jlop=jen2+1
      call spread(reg,jlop,ll,dmvp,dis)
      go to 2700
c     distance decreasing
 2630 jen2=jen2+1
 2640 distab=dis-dtab(jen2,ll)
      if (distab.lt.0) go to 2650
      reg(jen2)=reg(jen2)+(dis-disav)*dfac
      add=(dis-disav)*dfac
      if(add.lt.0) print *,2740
      jen2=jen2-1
      disav=dis
      go to 2700
 2650 reg(jen2)=reg(jen2)+(dtab(jen2,ll)-disav)*dfac
      add=(dtab(jen2,ll)-disav)*dfac
      if(add.lt.0) print*,2750
      disav=dtab(jen2,ll)
      jen2=jen2+1
      go to 2640
 2660 if(jen2.eq.0) go to 2680
 2670 distab=dis-dtab(jen2,ll)
      if(distab.gt.0) goto 2690
 2680 reg(jen2+1)=reg(jen2+1)+(dis-disav)*dfac
      add=(dis-disav)*dfac
      if(add.lt.0) print *, 2780
      disav=dis
      go to 2700
 2690 reg(jen2+1)=reg(jen2+1)+(dtab(jen2,ll)-disav)*dfac
      add=(dtab(jen2,ll)-disav)*dfac
      if(add.lt.0) print*, 2790
      disav=dtab(jen2,ll)
      jen2=jen2-1
      if (jen2.ge.1) go to 2670
      go to 2680
 2700 continue
 2710 continue
c    above ends loop for more than one turning point
 2720 continue
```

```
 2730 continue
      do 2740 iqr=1,maxdim
 2740 rawbin(iqr,idrc)=reg(iqr)
c     save accelerations for this source & site in rawbin array
c
c    ends loop for one site, one area or fault
c    repeat for all sites for this source before going on to next
c    source
 2750 continue
 2760 continue
c    one source completed for all sites
 2770 continue
c----smooth here if sigma for earthquake locations has changed
c----from previous value
      if(line.eq.1) go to 2850
c-----no smoothing for faults
      if(numsav.ne.98) go to 2850
c   if sigma=0, no smoothing
      if(als.eq.0) go to 2850
       if(nrc.eq.0) go to 2850
c-----
      sigd=(dphinc/als)**2
      do 2780 im=1,imaxsq
      fac(im)=exp((-im*sigd)/2.)
 2780 continue
      eqsum=0.
c    compute icon based on sigma and dphinc
c    probably enough to go out 2 sigma on either side of point
c    find how many rows or columns that corresponds to
      icon=2*als/dphinc+1.01
      if(icon.lt.5) icon=5
c   if icon<5, set icon=5
      irm=irow2-nbord
      icm=icol2-nbord
      isub1=irow1+nbord
      isub2=icol1+nbord
      do 2830 iia=isub2,icm
      j1=iia-icon
      j2=iia+icon
      if(j2.gt.icol2) j2=icol2
      if(j1.lt.icol1) j1=icol1
      do 2830 iraa=isub1,irm
      i1=iraa-icon
      i2=iraa+icon
      if(i1.lt.irow1) i1=irow1
      i2=iraa+icon
      if(i2.gt.irow2) i2=irow2
      do 2800 iq=1,maxp
      add=0.
      wtt=0.
      do 2790 i=i1,i2
      do 2790 j=j1,j2
      ksq=(iraa-i)**2+(iia-j)**2
      if(ksq.gt.imaxsq) go to 2790
c     we need to convert to proper subscript in rawbin
c      in rawbin, first all cols for one row
c      (i-1)*icol(s)+j=row and col with single subscript in rawbin
      isub=(i-irow1)*icols+j-icol1+1
      add=add+fac(ksq)*rawbin(iq,isub)
      wtt=wtt+fac(ksq)
 2790 continue
      reg(iq)=0
      if(wtt.ne.0) reg(iq)=add/wtt
      eqsum=eqsum+reg(iq)
 2800 continue
 2810 continue
      isub=(iraa-irow1)*icols+iia-icol1+1
      do 2820 iq=1,maxdim
 2820 smobin(iq,isub)=reg(iq)+smobin(iq,isub)
 2830 continue
c   now delete entries from rawbin array
      do 2840 isub=1,nrc
      do 2840 iq=1,maxdim
 2840 rawbin(iq,isub)=0.
 2850 continue
      if(iprint.lt.0) go to 2940
c  output all results to this point
      do 2930 ispt=1,2
      if (ispt.eq.1) go to 2860
      if(indvpt.eq.0) go to 2930
      irow1s=1
      irow2s=1
      icol1s=1
      icol2s=indvpt
      go to 2870
 2860 if(irow1.eq.0) go to 2930
      irow1s=irow1+nbord
      irow2s=irow2-nbord
      icol1s=icol1+nbord
      icol2s=icol2-nbord
 2870 do 2920 ira=irow1s,irow2s
      do 2920 ii=icol1s,icol2s
      if(ispt.eq.1) go to 2890
 2880 xouts=xpt1(ii)
      youts=ypt1(ii)
      isub=irows*icols+ii
      go to 2900
c     back transform point from equator to dec degrees
 2890 call backw(ii,ira,xouts,youts)
c     results calculated and printed in subroutine out
      isub=(ira-irow1)*icols+ii-icol1+1
 2900 do 2910 iqr=1,maxp
 2910 reg(iqr)=rawbin(iqr,isub)+smobin(iqr,isub)
      rdis=regg(isub)
      if(iprint.le.2) write(16,30)tytle
      call out(reg,naf,xouts,youts,sd)
 2920 continue
 2930 continue
 2940 continue
      isw=1
      go to 680
c     finished
 2950 nvals=icols*irows+indvpt
c     write to unit 3 for possible run continuation
c     may delete to save time and space if no run
c     continuations are desired.
      do 2970 i=1,nvals
      do 2960 iqr=1,maxdim
 2960 reg(iqr)=rawbin(iqr,i)+smobin(iqr,i)
      write(3) reg
```

```
 2970 continue
c    transfer to unit 3 for possible continuation or restart
 2980  continue
 2990 continue
c     write(16,2530) tot5,allerr
 3000 format(/' sources with error in area > .05=',f7.0,
     & ' out of ',f7.0,' sources ')
 3010 format(' number of brute force points=',i4)
      stop 7777
      end
      subroutine cond(x1,x2,y1,y2)
      call condec(x1)
      call condec(x2)
      call condec(y1)
      call condec(y2)
      return
      end
      subroutine condec(phi)
      i=phi
c     fr=phi-i
      phi=float(i)+(phi-float(i))/.60
      return
      end
c ****************************************************************
      subroutine lin(x1,y1,x2,y2,a,b,c,x3,y3)
c   coefficients a, b, c are determined so that points which are in the
c   same half plane as x3,y3 will be a positive distance from the line
c   joining x1,y1 and x2,y2
c       a x + b y + c=0
      if(x1.eq.x2) go to 10
      a=(y1-y2)/(x1-x2)
      b=-1
      c=y1-a*x1
      go to 20
   10 a=1.
      b=0
      c=-x1
   20 d=a*x3+b*y3+c
      if(d.gt.0.) return
      a=-a
      b=-b
      c=-c
      return
      end
c ****************************************************************
      subroutine linint(reg,t,sol)
c     calculates acceleration "sol" that has probability
c     "prob" of not being exceeded during "t" years at
c     site.
c     expected yearly accelerations in the range
c     ac(j-1) < a < ac(j) are in reg(j).
c     linint is called from subroutine out.
      common/tims/dummy(42),iprint
      common/quad/maxp,ac(105),dumlog,aclog(105)
      common/ext/prob,ex,exl
      dimension reg(106)
      ii=maxp-1
      sum=0.
   10 if(ii.le.2) go to 20
      if(reg(ii).gt.0.) go to 40
      ii=ii-1
      go to 10
   20 sol=0.
      if(iprint.le.2) write(16,30)
   30 format(' sol not obtained for time=',f7.0)
      return
   40 sum=sum+reg(ii)
      if(t*sum.gt.ex) go to 50
      ii=ii-1
      if(ii.ge.2) go to 40
      go to 20
   50 if(sum.eq.reg(ii)) go to 70
      y2=alog(t*(sum-reg(ii)))
      y1=alog(t*sum)
      exlt=exl
   60 a=(y1-y2)/(ac(ii-1)-ac(ii))
      b=y1-a*ac(ii-1)
      sol=(exlt-b)/a
      return
   70 y1=t*sum
      y2=0.
      exlt=ex
      if(iprint.le.2) write(16,80)
   80 format(' log interpolation fails--use linear')
      go to 60
      end
c ****************************************************************
      subroutine box(scale,sd)
c     determine boundaries ac(i) for acceleration levels
c     accelerations in the range ac(i-1) < a <ac(i) will be
c     accumulated in reg(i)
c     scale multiplies the basic levels computed
c     scale=1 for accelerations in the range .02 to 1.
      common /quad/maxq,ac(105),dumlog,aclog(105)
      common/wds/max,maxp,maxsp
      maxsp=104
      max=54
      maxp=55
      ac(5)=.02*scale
      dif=ac(5)
      do 10 i=6,55
      ac(i)=ac(i-1)+dif
   10 continue
c ----------------note important-------------
c     the following inserted to allow acceleration boxes for small
c     values below range of interest for sd computation
c     this should be changed later as desired
      j=4
      efac=exp(-sd/2.)
      do 20 i=1,4
      ac(j)=ac(j+1)*efac
   20 j=j-1
c    for later attenuation variability, expand acceleration range
c    used only in subroutine "out"
      y=alog(ac(max))
      w=exp(y+4.4*sd)
      w1=exp(y+2.0*sd)
c     dimension of ac, aclog is 105; we cannot exceed this dimension
      n=maxsp-max
      n2=n/2
```

```
      i2s=max+n2
      dif=(w1-ac(max))/n2
      do 30 i=maxp,i2s
      ac(i)=ac(i-1)+dif
   30 continue
      i2sp=i2s+1
      dif=(w-ac(i2s))/n2
      do 40 i=i2sp,maxsp
      ac(i)=ac(i-1)+dif
   40 continue
      return
      end
c ***************************************************************
      subroutine inieqr(x1,y1,x2,y2)
c    read xlong, ylat in radians for 1st point
c    read xlong, ylat in radians for 2nd point
c    points 1 and 2 to be transformed to equator
c    point 1 will be transformed to (0,0)
c    point 2 will be transformed to (d,0)
c     where d=great circle distance between (x1,y1) and (x2,y2)
c     output is transformation matrices f (forward) fi (backward)
c       generated and stored in common
      common/eqr/ r1(3,3),r2(3,3),r3(3,3),d(3,3),f(3,3),vec(3)
      common/inv/fi(3,3),res(3)
      cp1=cos(y1)
      cp2=cos(y2)
      cl1=cos(x1)
      sp1=sin(y1)
      sp2=sin(y2)
      sl1=sin(x1)
c  initialize matrices
      do 10 i=1,3
      do 10 j=1,3
      r1(i,j)=0.
      r2(i,j)=0.
   10 r3(i,j)=0.
      r1(1,1)=1.
      r2(2,2)=1.
      r3(3,3)=1.
      r2(1,1)=cp1
      r2(3,3)=cp1
      r2(1,3)=sp1
      r2(3,1)=-sp1
      r3(1,1)=cl1
      r3(2,2)=cl1
      r3(1,2)=sl1
      r3(2,1)=-sl1
      s1=cp1*sp2-sp1*cp2*cos(x2-x1)
      c1=cp2*sin(x2-x1)
      den=sqrt(c1*c1+s1*s1)
      sinth=s1/den
      costh=c1/den
      r1(2,2)=costh
      r1(3,3)=costh
      r1(2,3)=sinth
      r1(3,2)=-sinth
      call matmpy(r1,r2,d)
      call matmpy(d,r3,f)
c   transformation matrix in f! inverse matrix in fi
c   compute inverse
      do 20 i=1,3
      do 20 j=1,3
      r1(i,j)=0.
      r2(i,j)=0.
      r3(i,j)=0.
   20 continue
      r1(3,3)=1.
      r2(2,2)=1.
      r3(1,1)=1.
      r1(1,1)=cl1
      r1(2,2)=cl1
      r1(2,1)=sl1
      r1(1,2)=-sl1
      r2(1,1)=cp1
      r2(3,3)=cp1
      r2(1,3)=-sp1
      r2(3,1)=sp1
      r3(2,2)=costh
      r3(3,3)=costh
      r3(3,2)=sinth
      r3(2,3)=-sinth
      call matmpy(r1,r2,d)
      call matmpy(d,r3,fi)
      return
      end
c ***************************************************************
      subroutine toeqr(x3,y3,flam6,phi6)
c    transform point at (x3,y3)  (long,lat in radians) to new
c    equator using transformation defined in subroutine ineqr
      common/eqr/  r1(3,3),r2(3,3),r3(3,3),d(3,3),f(3,3),vec(3)
      common/inv/fi(3,3),res(3)
      cxmid=cos(x3)
      cymid=cos(y3)
      sxmid=sin(x3)
      symid=sin(y3)
      vec(1)=cxmid*cymid
      vec(2)=sxmid*cymid
      vec(3)=symid
      call vecmpy(f,vec,res)
      phi6=asin(res(3))
      cp6=cos(phi6)
      cl6=res(1)/cp6
      sl6=res(2)/cp6
      if(cl6.gt.1.) cl6=1.
      flam6=acos(cl6)
      if(sl6.lt.0.) flam6=-flam6
      return
      end
c ***************************************************************
      subroutine backw(ix,iy,flii,phii)
c     for printout and identification: point defined by
c     row ix and col iy in new coordinate system is transformed
c     to location (long, lat. in dec degrees) in original
c     coordinate system.
      common/inv/fi(3,3),res(3)
      dimension vec(3)
c     need cos, sins in common array
      common/sincos/sinx(500),cosx(500),siny(500),cosy(500)
      data rad/.0174533/
      sl6=sinx(ix)
```

```
      cl6=cosx(ix)
      cp6=cosy(iy)
      sp6=siny(iy)
      vec(1)=cl6*cp6
      vec(2)=sl6*cp6
      vec(3)=sp6
      call vecmpy(fi,vec,res)
      if(abs(res(3)).lt.1.0) go to 30
      if(res(3).ge.1.) go to 10
      phii=-90.
      go to 20
   10 phii=90.
   20 flii=0.
      return
   30 phii=asin(res(3))
      cpi=cos(phii)
      phii=phii/rad
      cli=res(1)/cpi
      sli=res(2)/cpi
      flii=acos(cli)
      if(sli.lt.0.) flii=-flii
      flii=flii/rad
      return
      end
c ***************************************************************
      subroutine matmpy(a,b,c)
c    multiply two 3 x 3 matrices A x B = C
      dimension a(3,3),b(3,3),c(3,3)
      do 10 i=1,3
      do 10 j=1,3
      c(i,j)=0.
      do 10 k=1,3
   10 c(i,j)=c(i,j)+a(i,k)*b(k,j)
      return
      end
c ***************************************************************
      subroutine vecmpy(f,vec,r)
c     vector multiplication used by subroutines backw, toeqr
      dimension f(3,3),vec(3),r(3)
      do 10 i=1,3
      r(i)=0.
      do 10 k=1,3
   10 r(i)=r(i)+f(i,k)*vec(k)
      return
      end
c ***************************************************************
      subroutine csqdis(x,y,xl,yl,xr,yr,isub)
c     for a quadrilateral source area, calculates
c     nearest distance from site at (x,y) to each side
c     and distance to each vertex.
c     called from subroutine rrisk
      dimension xl(4),yl(4),xr(4),yr(4)
      common/slinp/fsa(4,50),fsb(4,50),fsc(4,50),rta(4,50),dist(4,50)
     & ,disq(4,50)
      common /slinds/dls(4),perp(4),sl1(4),slsq(4),min(4),persq(4)
     &,srsq(4),dlp(4)
      do 10 i=1,4
   10 perp(i)=(fsa(i,isub)*x+fsb(i,isub)*y+fsc(i,isub))/rta(i,isub)
      do 20 i=1,4
      slsq(i)=(x-xl(i))**2+(y-yl(i))**2
      srsq(i)=(x-xr(i))**2+(y-yr(i))**2
   20 continue
      do 50 i=1,4
      persq(i)=perp(i)*perp(i)
      dl1=slsq(i)-persq(i)
      dl2=srsq(i)-persq(i)
      if(dl1.lt.0.) dl1=0.
      if(dl2.lt.0.) dl2=0.
      if(dl1+dl2.le.disq(i,isub)) go to 40
c      shortest distance external to segment
      if(dl1.lt.dl2) go to 30
      min(i)=1
      dlp(i)=sqrt(dl2)
      dls(i)=dist(i,isub)+dlp(i)
      go to 50
   30 min(i)=2
      dls(i)=-sqrt(dl1)
      go to 50
   40 min(i)=3
      dls(i)=sqrt(dl1)
      dlp(i)=0.
   50 continue
      return
      end
c ***************************************************************
      subroutine rrisk(xnot,ynot,in,isub,r,rc,rf,itst)
       common/inout/ic,ics,ifar,azimf,stsize,dum1
       common/xyarea/xsav(2,50),ysav(2,50),sarea(50)
       common/slinds/dls(4),perp(4),sl1(4),slsq(4),min(4),persq(4)
     &,srsq(4),dlp(4)
       common/slinp/aa(4,50),bb(4,50),cc(4,50),rta(4,50),dist(4,50),
     & disq(4,50)
       common/xlyl/xl(4),yl(4),xr(4),yr(4),xc(4),yc(4)
c         subroutine which loads temporary arrays with this
c           subsource's corners and determines if this site is
c           inside or outside the subrource.
      data pi2/6.2831853/
c   fixed number of radii at which distance-vs arc length
c   table is computed is set here   itst=7 ***************
      itst=7
      rc2=10000000.
      ics=0
      call csqdis(xnot,ynot,xl,yl,xr,yr,isub)
      rf2=0.
      do 20 ii=1,4
      if(slsq(ii).gt.rc2) go to 10
      rc2=slsq(ii)
   10 if(slsq(ii).lt.rf2) go to 20
      rf2=slsq(ii)
      ifar=ii
   20 continue
      do 30 ii=1,4
      if(min(ii).ne.3) go to  30
      if(persq(ii).gt.rc2) go to 30
      rc2=persq(ii)
      ics=ii
   30 continue
      rc=sqrt(rc2)
      rf=sqrt(rf2)
      stsize=(rf-rc)/itst
```

```
c        step thru source area.
      r=rc+(0.5)*stsize
c        determine if site is inside source area.
   40 do 50 ii=1,4
      d=aa(ii,isub)*xnot+bb(ii,isub)*ynot+cc(ii,isub)
      if(d.lt.0)go to 90
   50 continue
      in=1
c         determine azimuth of farthest point with respect to site
      azimf=acos((xl(ifar)-xnot)/rf)
      if (yl(ifar)-ynot) 60,70,70
   60 azimf=pi2 - azimf
   70 continue
c       rc is now distance from site to closest side.
c       rf is now distance from site to farthest corner.
c         pick step size based on fraction of area left
   80 return
c         if do loop finished, point lies within area.
   90 in=0
  100 return
      end
c ***************************************************************
      subroutine setreg(isub)
c    find equation of line thru each side of quadrilateral
c   find length of each side, etc.
      common/outs/xouts,youts
      dimension xl(4),yl(4),xr(4),yr(4)
      common/xyarea/xsav(2,50),ysav(2,50),sarea(50)
      common/slinp/aa(4,50),bb(4,50),cc(4,50),rta(4,50),dist(4,50),
     & disq(4,50)
      do 10 i=1,2
      xl(i)=xsav(i,isub)
      xl(i+2)=xsav(i,isub+1)
      yl(i)=ysav(i,isub)
      yl(i+2)=ysav(i,isub+1)
   10 continue
      xr(3)=xl(1)
      yr(3)=yl(1)
      xr(1)=xl(2)
      yr(1)=yl(2)
      xr(2)=xl(4)
      yr(2)=yl(4)
      xr(4)=xl(3)
      yr(4)=yl(3)
c         determine if any sides are vertical lines
      do 70 ii=1,4
      if(xl(ii).eq.xr(ii)) go to 20
      aa(ii,isub)=(yl(ii)-yr(ii))/(xl(ii)-xr(ii))
      bb(ii,isub)=-1.
      cc(ii,isub)=yl(ii)-aa(ii,isub)*xl(ii)
      go to 30
   20 aa(ii,isub)=1.
      bb(ii,isub)=0.
      cc(ii,isub)=-xl(ii)
   30 if(ii.gt.2) go to 40
      d=aa(ii,isub)*xl(3)+bb(ii,isub)*yl(3)+cc(ii,isub)
      go to 50
   40 d=aa(ii,isub)*xl(2)+bb(ii,isub)*yl(2)+cc(ii,isub)
   50 if (d.gt.0.) go to 60
      aa(ii,isub)=-aa(ii,isub)
      bb(ii,isub)=-bb(ii,isub)
      cc(ii,isub)=-cc(ii,isub)
   60 disq(ii,isub)=(xr(ii)-xl(ii))**2+(yr(ii)-yl(ii))**2
      dist(ii,isub)=sqrt(disq(ii,isub))
   70 rta(ii,isub)=sqrt(aa(ii,isub)*aa(ii,isub)+bb(ii,isub)*
     &bb(ii,isub))
      return
      end
      function indpt(y,yval,n)
c      points in descending order--non zero values from points
c      1 to n:  find first value in table exceeding yval
      dimension y(102)
      if(yval.lt.y(1)) go to 10
      indpt=0
      return
   10 if(yval.gt.y(n)) go to 20
      indpt=n
      return
   20 num=n
      idiv=2
   30 nh=num/idiv
      if(nh.le.1) go to 50
      if(yval.lt.y(nh)) go to 40
      idiv=2*idiv
      go to 30
c    yval lies approx between nh and 2*nh or 2*nh
   40 idiv=2*idiv
      nh2=nh+num/idiv
      if(nh.eq.nh2) goto 80
      if(yval.gt.y(nh2)) go to 40
c    yval further down on table than y(nh2)
      nh=nh2
      go to 40
   50 do 60 i=1,n
      if(y(i).lt.yval) go to 70
   60 continue
   70 indpt=i-1
      return
   80 continue
      do 90 i=nh,n
      if(y(i).lt.yval) go to70
   90 continue
      end
c ***************************************************************
      subroutine cldis(x,y,jl)
c     subroutine to compute shortest distance, etc to line
c     containing fault segment i of jl th fault (for each i)
c     determines whether closest point to fault is at
c     low end (min(i)=2)
c     interior  (min(i)=3)
c     high end  (min(i)=1)
c     set isid=0 if multiple turning points
c         isid=1 if single closest distance from site to fault
c         interior to fault
c         isid=2 if closest distance from site to fault at one end
c         of the fault
      dimension dlp(20)
      common/linps/distl(24,26),disq(24,26),jseg,jsegm,
     & xl(24,26),yp(24,26),coa(24,26),sia(24,26)
      common/extra/isid,shdis,dist(24),inear
```

```
      common /lindis/dls(24),perp(24),min(24),persq(24)
      shdis=1.e9
      do 10 ii=1,jsegm
   10 dist(ii)=distl(ii,jl)
      i1=0
      i2=0
      i3=0
      isid=0
      do 20 i=1,jseg
      slsq=(x-xl(i,jl))**2+(y-yp(i,jl))**2
      if(slsq.lt.shdis) shdis=slsq
   20 continue
      do 50 i=1,jsegm
      xd=x-xl(i,jl)
      yd=y-yp(i,jl)
      perp(i)=-xd*sia(i,jl)+yd*coa(i,jl)
      dls(i)=xd*coa(i,jl)+yd*sia(i,jl)
      persq(i)=perp(i)*perp(i)
      if(dls(i).lt.0) go to 30
      if(dls(i).le.dist(i)) go to 40
c      shortest distance external to segment
      min(i)=1
      i1=i1+1
      dlp(i)=dls(i)-dist(i)
      go to 50
   30 min(i)=2
      i2=i2+1
      go to 50
   40 min(i)=3
      i3=i3+1
      if(persq(i).lt.shdis) shdis=persq(i)
      dlp(i)=0.
   50 continue
      if(i1.ne.jsegm)  go to 90
      iup=jsegm
      isid=2
   60 iuh=iup/2
      if(2*iuh.ne.iup) dls(iuh+1)=-dlp(iuh+1)
      if(iup.eq.1)  go to 80
      j=iup
      do 70 i=1,iuh
      t=persq(i)
      persq(i)=persq(j)
      persq(j)=t
      dls(i)=-dlp(j)
      dls(j)=-dlp(i)
      t=dist(i)
      dist(i)=dist(j)
      min(i)=2
      min(j)=2
      dist(j)=t
   70 j=j-1
   80 return
   90 if(i2.ne.jsegm)  go to 100
      isid=2
      return
  100 if(i3.gt.1) return
      if(min(1).eq.2) return
      iturn=0
      do 110 k=1,jsegm
      if(min(k).ne.1) go to 120
      iturn=iturn+1
  110 continue
  120 if(iturn.ne.i1) return
      if(i3.eq.1 .and. min(k).eq.2) return
c   we have single turning point
      isid=1
      inear=k-1
      if(min(k).ne.3) go to 140
      inear=k
c      adjust to 2 sep segments
      j=jseg
      do 130 i=k,jsegm
      dls(j)=dls(j-1)
      persq(j)=persq(j-1)
      dist(j)=dist(j-1)
  130 j=j-1
      jsegm=jseg
      jseg=jseg+1
      dist(k+1)=dist(k+1)-dls(k)
      dist(k)=dls(k)
      dls(k+1)=0
  140 iup=inear
      go to 60
      end
c ***************************************************************
      subroutine pbreak(fm,al,bl,sigl,rlls,prls,lev,nrls)
      dimension fm(12),prls(12),rlls(10,12)
      dimension pmid(5),prang(5)
      data pmid/1.575,.74,0.,-.74,-1.575/
      data prang/.1151,.2295,.3108,.2295,.1151/
      data nrl/5/
c      evaluate al+bl*fm(l)+pmid(i)*sigl
c     where  1 <l <lev     and 1 < i < nrl
c     prang probability associated with jth break
c     for each magnitude; compute nrl rupture lengths
c      :::::::: must be ordered in descending order--largest break
c      ::::::::: length first for each magnitude
      do 10 j=1,lev
      con=al + bl*fm(j)
      do 10 i=1,nrl
      c=con+pmid(i)*sigl
   10 rlls(i,j)=10.**c
      do 20 i=1,nrl
   20 prls(i)=prang(i)
      nrls=nrl
      return
      end
c ***************************************************************
      subroutine bbgaus(x,gg)
c   computes normal probability integral gg from -(infinity) to xx
c    sets gg=1 if xx < -6;  gg=0  if xx > +6
c   uses approximation Handbook of Mathematical Functions-
c   NBS Applied Math Series 55--p299 7.1.26 for error function
      ax=abs(x)/1.4142136
      if(ax-4.2426408) 20,20,10
   10 gg=0.
      if(x.lt.0) gg=+1.
      return
   20 continue
```

```
      d=1.+(((((4.30638e-5*ax+2.765672e-4)*ax+1.520143e-4
     & )*ax+9.2705272e-3)*ax+4.2282012e-2)*ax+7.0523078e-2)*ax
      d=d*d
      d=d*d
      d=d*d
      gg=.5/(d*d)
      if(x) 30,30,40
   30 gg=1.-gg
   40 return
      end
c ************************************************************
      subroutine outsid(xnot,ynot,isub,r,angle)
      common/inout/ic,ics,ifar,azimf,stsize,dum1
      common/xysav/xsav(2,50),ysav(2,50),sarea(50)
       common/xlyl/xl(4),yl(4),xr(4),yr(4), xc(4),yc(4)
       common/slinp / aa(4,50),bb(4,50),cc(4,50),rta(4,50),dist(4,50)
     &,disq(4,50)
      common/slinds/dls(4),perp(4),sl1(4),slsq(4),min(4),persq(4)
     &,srsq(4),dlp(4)
      dimension jsub(3)
c     subroutine for calculating angle when site is outside
c          (quadrilateral) source area.
c      r is the radial distance for which angle is sought
c
c     (xnot,ynot)=point outside quadrilateral source region isub
c     r=radius of circle (or radius of annular ring of width dr)
c     centered at (xnot,ynot)
c     isub = index identifying quadrilateral source region
c
c    the area contained in the intersection of the region and annular
c     ring is determined by the fraction of the ring within the region.
c           area = angle x r x dr
c     where angle = the angle (in radians) subtended by the
c     part of the annular ring contained within the region.
c
c          (rc is closest distance between site and source.)
c          (rf is furthest distance between site and source)
c       subroutine csqdis has computed distance to vertices of subregion
c       and shortest distance to each side
c       adapted from subroutine by mcguire
c
      data jsub/2,3,1/
      npt=0
      angle=0.
      signal=1.
c        loop on each side
      rsq=r*r
      do 140 ii=1,4
      if(min(ii)-2) 10,40,70
c     closest point is xr(ii),yr(ii)
c       at most one intersection
   10 if(rsq.gt.slsq(ii)) go to 140
      if(rsq.lt.srsq(ii)) go to 140
      d=(sqrt(rsq-persq(ii))-dlp(ii))/dist(ii,isub)
   20 y1=(yl(ii)-yr(ii))*d+yr(ii)
      x1=(xl(ii)-xr(ii))*d+xr(ii)
c  intersects in range
c         does circle (radius r) intersect it?
c        store first point
   30 npt=npt+1
      xc(npt)=x1
      yc(npt)=y1
      go to 140
c   closest point is at xl(ii),yl(ii)
   40 if(rsq.lt.slsq(ii)) go to 140
      if(rsq.gt.srsq(ii)) go to 140
   50 d=(sqrt(rsq-persq(ii))+dls(ii))/dist(ii,isub)
   60 x1=xl(ii)+(xr(ii)-xl(ii))*d
      y1=yl(ii)+(yr(ii)-yl(ii))*d
      go to 30
   70 if(rsq.lt.persq(ii)) go to 140
      if(rsq.le.slsq(ii)) go to 80
      if(rsq.gt.srsq(ii)) go to 140
c    single intersection
c      on side nearer xr,yr
      go to 50
   80 if(rsq.gt.srsq(ii)) go to 90
c      two intersections on this side
c     can compute angle directly without determining coordinates
      arg=abs(perp(ii))/r
      as=acos(arg)
      go to 100
   90 d=(dls(ii)-sqrt(rsq-persq(ii)) )/dist(ii,isub)
      go to 60
c     single intersection
c         see if this side is closest to point, if so, treat specially.
  100 if (ii-ics) 110,120,110
c           both points are on boundary, calculate angle betwen them.
  110 sign=-1.
      go to 130
  120 sign=1.
      signal=-1.
  130 angle=sign*2.*as + angle
c         see if second point only is on boundary
  140 continue
      if(npt.gt.0) go to 150
      if(signal) 310,320,320
  150 go to (320,240,160,250),npt
c     this is an error unless radius is on a vertex and
c      was counted twice.
  160 do 170 i=1,3
      j=jsub(i)
      if(abs(xc(i)-xc(j)) .ge. .01) go to 170
      if(abs(yc(i)-yc(j)) .le. .01) go to 220
  170 continue
c   check for one of three points being a vertex
      do 210 i=1,4
      do 180 j=1,3
      if(abs(xc(j)-xl(i)).gt. .001) go to 180
      if(abs(yc(j)-yl(i)) .le. .001) go to 190
  180 continue
      go to 210
  190 if(j-2) 200,200,230
  200 xc(j)=xc(3)
      yc(j)=yc(3)
      go to 230
  210 continue
      go to 320
  220 if(i.ne.1) go to 230
      xc(2)=xc(3)
```

```
      yc(2)=yc(3)
  230 npt=2
      go to 240
  240 ad=sqrt((xc(1)-xc(2))*(xc(1)-xc(2))+(yc(1)-yc(2))*(yc(1)-yc(2)))
      angle=angle + signal*2.*asin(ad/(2.*r))
      go to 310
c          four intersection points (each on a different side).
c          determine angle by finding closest 2 intersections to
c          farthest corner, calculate angle between, and add angle
c          between other two intersections.
  250 dist1=10000000.
      i1=0
      i2=0
      i3=0
      i4=0
      do 300 jj=1,4
      distn=(xl(ifar)-xc(jj))*(xl(ifar)-xc(jj))
     &  +(yl(ifar)-yc(jj))*(yl(ifar)-yc(jj))
      if (distn-dist1) 260,260,270
  260 dist2=dist1
      dist1=distn
      i4=i3
      i3=i2
      i2=i1
      i1=jj
      go to 300
  270 if (distn-dist2) 280,280,290
  280 dist2=distn
      i4=i3
      i3=i2
      i2=jj
      go to 300
  290 i4=i3
      i3=jj
  300 continue
c           calculate angle between 2 closest points to farthest corner.
      ad=sqrt((xc(i1)-xc(i2))*(xc(i1)-xc(i2))
     &  + (yc(i1)-yc(i2))*(yc(i1)-yc(i2)))
      angle=2.*asin(ad/(2.*r))
c          calculate angle between 2 points farthest from
c          farthest corner and add to previous angle.
      ad=sqrt((xc(i3)-xc(i4))*(xc(i3)-xc(i4))
     &  +(yc(i3)-yc(i4))*(yc(i3)-yc(i4)))
      angle=2.*asin(ad/(2.*r)) + angle
  310 continue
c         compute rate of earthquakes in this annular source
      return
c         error printout
  320 write(16,330)isub,ic,npt,xnot,ynot,(xl(i),yl(i), i=1,4),r,angle,
     &(xc(i),yc(i),i=1,4)
  330 format(' ***** error in subroutine outsid. source no.',i3,
     &' debug values follow.....',/10x,2i5,5(/10x,4f14.6))
  340 stop 9999
      end
c ***************************************************************
      subroutine inside(xnot,ynot,isub,r,pangle)
      common/xyarea/xsav(2,50),ysav(2,50),sarea(50)
      common/inout/ic,ics,ifar,azimf,stsize,stepo
      common/slinds/dls(4),perp(4),sl1(4),slsq(4),min(4),persq(4)
     &,srsq(4),dlp(4)
      common/slinp/aa(4,50),bb(4,50),cc(4,50),rta(4,50),dist(4,50),
     & disq(4,50)
      common/xlyl/xl(4),yl(4),xr(4),yr(4),xc(4),yc(4)
      dimension jsub(3)
c
c     (xmid,ymid)=point inside quadrilateral source region ii
c     r=radius of circle (or radius of annular ring of width dr)
c     centered at (xmid,ymid)
c     ii = index identifying quadrilateral source region
c
c    the area contained in the intersection of the region and annular
c     ring is determined by the fraction of the ring within the region.
c           area = pangle x r x dr
c     where pangle = the angle (in radians) subtended by the
c     part of the annular ring contained within the region.
c     subroutine insid determines pangle.
c
c       adapted from subroutine by mcguire
      data pi,pi2/3.1415927,6.2831853/
      data jsub/2,3,1/
      rsq=r*r
      npt=0
      angle=0.
c         loop on each side
      do 110 ii=1,4
      if(min(ii)-2) 10,40,70
c     closest point is xr(ii),yr(ii)
c       at most one intersection
   10 if(rsq.gt.slsq(ii)) go to 110
      if(rsq.lt.srsq(ii)) go to 110
      d=(sqrt(rsq-persq(ii))-dlp(ii))/dist(ii,isub)
   20 y1=(yl(ii)-yr(ii))*d+yr(ii)
      x1=(xl(ii)-xr(ii))*d+xr(ii)
c  intersects in range
c          does circle (radius r) intersect it?
c        store first point
   30 npt=npt+1
      xc(npt)=x1
      yc(npt)=y1
      go to 110
c   closest point is at xl(ii),yl(ii)
   40 if(rsq.lt.slsq(ii)) go to 110
      if(rsq.gt.srsq(ii)) go to 110
   50 d=(sqrt(rsq-persq(ii))+dls(ii))/dist(ii,isub)
   60 x1=xl(ii)+(xr(ii)-xl(ii))*d
      y1=yl(ii)+(yr(ii)-yl(ii))*d
      go to 30
   70 if(rsq.lt.persq(ii)) go to 110
      if(rsq.le.slsq(ii)) go to 80
      if(rsq.gt.srsq(ii)) go to 110
c    single intersection
c     on side nearer xr,yr
      go to 50
   80 if(rsq.gt.srsq(ii)) go to 90
c     two intersections on this side
c    can compute angle directly without determining coordinates
      arg=abs(perp(ii))/r
      as=acos(arg)
      go to 100
   90 d=(dls(ii)-sqrt(rsq-persq(ii)))/dist(ii,isub)
```

```
      go to 60
c    single intersection
c          both points are on boundary, calculate angle betwen them.
  100 angle=2.*as + angle
c        see if second point only is on boundary
  110 continue
      if (npt) 570,120,140
  120 if (angle-0.001) 570,130,130
c        following is for case of no single intersection points;
c        angle is 2 * pi - angle calculated so far.
  130 pangle=pi2-angle
      go to 560
  140 go to (570,230,150,500),npt
c     this is an error unless radius is on a vertex and
c      was counted twice.
  150 do 160 i=1,3
      j=jsub(i)
      if(abs(xc(i)-xc(j)) .ge. .01) go to 160
      if(abs(yc(i)-yc(j)) .le. .01) go to 210
  160 continue
c   check for one of three points being a vertex
      do 200 i=1,4
      do 170 j=1,3
      if(abs(xc(j)-xl(i)).le. .001) go to 170
      if(abs(yc(j)-yl(i)) .gt. .001) go to 180
  170 continue
      go to 200
  180 if(j-2) 190,190,220
  190 xc(j)=xc(3)
      yc(j)=yc(3)
      go to 220
  200 continue
      go to 570
  210 if(i.ne.1) go to 220
      xc(2)=xc(3)
      yc(2)=yc(3)
  220 npt=2
      go to 230
c         2 intersection points; determine azimuths.
  230 if (xc(1)-xnot-r) 260,250,240
  240 if (xc(1)-xnot-r-0.001) 250,250,570
  250 azim1=0.0
      go to 300
  260 if (xc(1)-xnot+r) 270,280,290
  270 if (xc(1)-xnot+r+0.001) 570,280,280
  280 azim1=pi
      go to 320
  290 azim1=acos((xc(1)-xnot)/r)
  300 if (yc(1)-ynot) 310,320,320
  310 azim1=pi2  - azim1
  320 if (xc(2)-xnot-r) 350,340,330
  330 if (xc(2)-xnot-r-0.001) 340,340,570
  340 azim2=0.0
      go to 390
  350 if (xc(2)-xnot+r) 360,370,380
  360 if (xc(2)-xnot+r+0.001) 570,370,370
  370 azim2=pi
      go to 410
  380 azim2=acos((xc(2)-xnot)/r)
  390 if (yc(2)-ynot) 400,410,410
  400 azim2=pi2 -azim2
  410 pangle=azim2-azim1
      if (pangle) 420,570,460
  420 if (azim1-azimf) 430,570,440
  430 pangle=pi2 +pangle -angle
      go to 560
  440 if (azimf-azim2) 430,570,450
  450 pangle=-pangle-angle
      go to 560
  460 if (azim2-azimf) 470,570,480
  470 pangle=pi2 -pangle -angle
      go to 560
  480 if (azimf-azim1) 470,570,490
  490 pangle=pangle-angle
      go to 560
c        four intersection points (each on a different side).
c        determine angle by finding closest 2 intersections to
c        farthest corner, calculate angle betwen, and add angle
c        between other two intersections.
  500 dist1=10000000.
      i1=0
      i2=0
      i3=0
      i4=0
      do 550 jj=1,4
      distn=(xl(ifar)-xc(jj))*(xl(ifar)-xc(jj))
     &   +(yl(ifar)-yc(jj))*(yl(ifar)-yc(jj))
      if (distn-dist1) 510,510,520
  510 dist2=dist1
      dist1=distn
      i4=i3
      i3=i2
      i2=i1
      i1=jj
      go to 550
  520 if(distn-dist2) 530,530,540
  530 dist2=distn
      i4=i3
      i3=i2
      i2=jj
      go to 550
  540 i4=i3
      i3=jj
  550 continue
c        calculate angle between 2 closest points to farthest corner
      ad=sqrt((xc(i1)-xc(i2))*(xc(i1)-xc(i2))
     &  + (yc(i1)-yc(i2))*(yc(i1)-yc(i2)))
      pangle=2.*asin(ad/(2.*r))
c          calculate angle between 2 farthest points from
c          farthest corner and add to previous angle.
      ad=sqrt((xc(i3)-xc(i4))*(xc(i3)-xc(i4))
     & + (yc(i3)-yc(i4))*(yc(i3)-yc(i4)))
      pangle=2.*asin(ad/(2.*r))+pangle
c       angle for this radius is now known, calculate risk
  560 return
c     anarea=pangle*r*stsize
c          error printout
  570 write (16,580) isub,ifar,npt,xnot,ynot,(xl(i),yl(i), i=1,4),r,
     &pangle,(xc(i),yc(i),i=1,4)
  580 format (' ***** error in subroutine inside. source no.',
```

```
     & i3,' debug values follow.....',/10x,2i10,10(/10x,2f12.6))
  590 stop 0000
      end
c*************************************************************
      subroutine spread(reg,jlo,ll,totac,dmid)
c     magnitude smoother for fault ruptures:
c     totac acceleration occurrences calculated for ruptures of
c     magnitude fm(ll) at distance dmid from site
c     are "spread" to acceleration levels corresponding
c     to magnitudes in the range
c     fm(ll)-delta/2 < fm(ll) < fm(ll)+delta/2
c     [where fm(ll-1)=fm(ll)-delta].
c     earthquake occurrences follow a Gutenberg-Richter
c     magnitude-frequency relationship, calculated from number
c     of earthquakes input at 2 successive magnitudes.
      common/magdis/jent,mdis
      common/rtb/rtab(20)
      dimension reg(106)
      common/accel/gm(25),exdif(101),atah(20,25),aclim(2)
      common/wds/max,maxp,maxsp
      common/quad/maxq,ac(105),dumlog,aclog(105)
   10 continue
      if(dmid.eq.-1.) go to 60
      do 20 i=2,mdis
      if(dmid.le.rtab(i)) go to 40
   20 continue
c  distance exceeds table--we should not even be here
      write(16,30) dmid
   30 format(' in spread, dmid=',e12.5,' too large-should not be here')
      stop 1111
   40 fr=(rtab(i)-dmid)/(rtab(i)-rtab(i-1))
      ilim=1
      lp=ll+1
      do 50 lm=ll,lp
      aclim(ilim)=atah(i,lm)-fr*(atah(i,lm)-atah(i-1,lm))
   50 ilim=2
   60 acdif=aclim(2)-aclim(1)
      if(acdif.ne.0.) go to 90
      jbot=jlo+1
   70 if(aclog(jbot).le.aclim(1)) go to 80
      jbot=jbot-1
      if(jbot.ge.1) go to 70
   80 jup=jbot+1
      reg(jup)=reg(jup)+totac
      return
   90 add=0.
      jbot=jlo+1
  100 if(aclog(jbot).le.aclim(1)) go to 110
      jbot=jbot-1
      if(jbot.ge.1) go to 100
c    we are alower accel level--for lowest mag of range
  110 aclo=exdif(1)
      jup=jbot+1
  120 upl=aclog(jup)
      if(upl.gt.aclim(2)) upl=aclim(2)
      fr=(upl-aclim(1))/acdif
      iachi=100.*fr+1.00001
      quan=(+exdif(iachi)-aclo)*totac
      add=add+quan
      reg(jup)=reg(jup)+quan
      if(aclim(2).eq.upl) go to 130
      aclo=exdif(iachi)
      jup=jup+1
      if(jup.le.maxp) go to 120
      write(16,125)
  125 format(' spread routine needs higher acceleration level;',
     1/' rewrite BOX or increase scale factor')
      stop 2222
  130 continue
      return
      end
c*************************************************************
      subroutine out(reg,naf,xouts,youts,sd)
      common/tims/ftim(20),jtim(20),ntims,nwt,iprint
      common/rdis/rdis
      dimension qp(140),sol(20),sols(20)
      common/quad/maxq,ac(105),dumlog,aclog(105)
      dimension reg(106)
      common/ext/prob,ex,exl
      common/wds/max,maxp,maxsp
      dimension regs(106),cdfs(105),cdf(105)
      if(iprint.le.2) write(16,10) xouts,youts,rdis,sd
   10 format(' site at long ',f8.3,', lat ',f8.3,
     &/' shortest dist to fault=',f9.3,' km'/10x,
     &'zero attenuation variability',18x,'variability in atten, sigma=',
     & f5.2/2( '    g.m.   occ/yr    exc/yr  r(events) r(yrs) '))
      do 20 l=1,maxsp
   20 regs(l)=0
   30 sd6=4.4*sd
      kls=1
      do 70 l=1,max
      if(reg(l+1).eq.0) go to 70
      asav=(aclog(l)+aclog(l+1))/2.
      kl=kls
   40 if(aclog(kl)-asav.ge.-sd6) go to 50
      kl=kl+1
      if(kl.le.max) go to 40
      go to 70
   50 kls=kl
      w=(aclog(kl)-asav)/sd
      call bbgaus(w,g1)
      if(kl.eq.1) regs(1)=regs(1)+(1.-g1)*reg(l+1)
   60 kl=kl+1
      wh=(aclog(kl)-asav)/sd
      call bbgaus(wh,g2)
      regs(kl)=regs(kl)+(g1-g2)*reg(l+1)
      if(wh.ge.4.4) go to 70
      if(kl.ge.maxsp) go to 70
      g1=g2
      go to 60
   70 continue
c  we are really not interested in 1st 4 acceleration boxes but used
c  them for computation of variability in acceleration--correct so
c  box 5 accumulates all accelerations below ac(5)
c  ac(5) is the first value printed
   80 continue
      regs(1)=regs(1)+reg(1)
      regsav=reg(5)
      regssv=regs(5)
      do 90 i=1,4
```

```
      regs(5)=regs(5)+regs(i)
   90 reg(5)=reg(5)+reg(i)
      cdfs(maxsp)=0.
      cdf(maxp)=0.
      maxq=maxp
      k=max
      do 100 i=5,max
      cdf(k)=cdf(k+1)+reg(k+1)
  100 k=k-1
      mm=maxsp-1
      k=mm
      do 110 i=5,mm
      cdfs(k)=cdfs(k+1)+regs(k+1)
  110 k=k-1
      if((cdfs(5).ne.0.0).and.(cdf(5).ne.0.0)) go to 140
      if(iprint.ge.2) write(16,120)
  120 format(' no acceleration events accumulated. statistics calc.
     &bypassed')
      do 130 ier=1,ntims
      sol(ier)=0.0
  130 sols(ier)=0.
      go to 380
  140 continue
c
c  if accelerations do not exceed ac(5), the first acceleration
c  of interest, skip statistics calculation to avoid dividing by
c  zero or upsetting interpolation routines.
c
      cnums=cdfs(5)+regs(5)
      cnum=cdf(5)+reg(5)
      yrncnum=1./cnum
      nrep=0
      idone=0
      if(iprint.eq.3) go to 330
      do 300 i=5,maxsp
      if(i.gt.max) go to 150
      if(cdf(i).eq.0.0) go to 170
      if(reg(i).ne.0) go to 160
  150 if(regs(i).eq.0.) go to 190
c
c   compute return period in number of events and number of years
c   return period in events = average number of earthquakes needed
c   to produce an acceleration exceeding a
c      r(events)=(yearly earthquakes)/(yearly exceedances of a)
c   return period in years = average time between earthquakes
c   which cause acceleration a to be exceeded
c      r(yrs)=1/(yearly exceedances of a)
c
  160 if(i.le.max) retev=cnum/cdf(i)
      retyr=retev*yrncnum
c
c   compute acceleration which has probability prob of not being
c   exceeded during t years.  this is given by the value of a
c   for which
c      prob = exp(-t * number of exceedances of a per year)
c
      go to 180
  170 retev=99999.9
      retyr=99999.9
  180 if(cdfs(i).eq.0.) go to 190
      retevs=cnums/cdfs(i)
      retyrs=retevs/cnums
      go to 200
  190 retevs=99999.9
      retyrs=99999.9
      if(cdf(i).ne.0.) go to 200
      idone=1
  200 continue
      qp(nrep+1)=ac(i)
      if(i.le.max) go to 210
      qp(nrep+2)=0
      qp(nrep+3)=0
      retev=99999.9
      retyr=99999.9
      go to 220
  210 qp(nrep+2)=reg(i)
      qp(nrep+3)=cdf(i)
  220 if(retev.ge.100000.) retev=99999.9
      if(retyr.ge.100000.) retyr=99999.9
      qp(nrep+4)=retev
      qp(nrep+5)=retyr
      qp(nrep+6)=ac(i)
      qp(nrep+7)=regs(i)
      qp(nrep+8)=cdfs(i)
      if(retevs.ge.100000.) retevs=99999.9
      if(retyrs.ge.100000.) retyrs=99999.9
      qp(nrep+9)=retevs
      qp(nrep+10)=retyrs
      if(nrep.ne.130) go to 260
      if(naf.le.3) go to 230
      write(16,280) qp
      go to 240
  230 write (16,270) qp
  240 continue
      nrep=0
  250 format(1h1)
      if(idone.eq.1) go to 330
      go to 300
  260 nrep=nrep+10
  270 format(1h ,f8.2,2f9.5,2f8.1,f12.2,2f9.5,2f8.1)
  280 format(1h ,f8.4,2f9.5,2f8.1,f12.4,2f9.5,2f8.1)
      if(retev.eq.99999.9 .and. retevs.eq.99999.9) go to 310
  300 continue
  310 if(nrep.eq.0) go to  330
      if(naf.le.3) go to 320
      write(16,280) (qp(ll),ll=1,nrep)
      go to 330
  320 write(16,270) (qp(ll),ll=1,nrep)
  330 if (iprint.gt.2) go to 350
      write(16,340) cnum,cnums,sd
  340 format(2('   total yearly events ',f10.5,12x)
     &//10x,'zero attenuation variability',18x,
     & 'variability in atten, sigma=',f4.2)
c
c   compute acceleration which has probability prob of not being
c   exceeded during t years.  this is given by the value of a
c   for which
c      prob = exp(-t * number of exceedances of a per year)
c
  350 reg(5)=regsav
```

```
      regs(5)=regssv
      do 360 jv=1,ntims
      maxq=max
      call linint(reg,ftim(jv),sol(jv))
      maxq=maxsp
      call linint(regs,ftim(jv),sols(jv))
      if(iprint.le.2) write(16,370)prob,sol(jv),jtim(jv),sols(jv),
     &jtim(jv)
  360 continue
  370 format(f10.3,' ext prob =',f7.3,' for ',i5,' years',19x,
     & f7.3,' for ',i4,' years')
  380 rat=0.0
      if(sol(1).ne.0.0) rat=sol(ntims)/sol(1)
      ratsd=0
      if(sols(1).ne.0) ratsd=sols(ntims)/sols(1)
      if(iprint.le.2)write(16,390)jtim(ntims),prob,jtim(1),rat, ratsd
      if(iprint.ge.2) write(2) xouts,youts,(sol(jt),jt=1,ntims),
     &(sols(jt),jt=1,ntims)
  390 format('   ratio',i5,' yr',f6.3,' extreme value to',i5,' yr val
     &=',f7.2,7x,f7.2)
      return
      end
```

SEISRISK III is a computer program designed to calculate maximum seismic ground-motion levels that have a specified probability of not being exceeded during fixed time periods at each of a set of sites uniformly spaced on a two-dimensional grid (Bender and Perkins, 1987). To understand the theory on which the program is based one should have a copy of USGS Bulletin 1772. The program is here adapted for use on IBM PC and compatible microcomputers using either DOS or OS/2 operating systems. At least 640 KB of RAM is required for DOS operation along with an appropriate maths co-processor. Instructions for operation are given at the beginning of the program listing in Appendix A.

OF89–557–A, Documentation, 31p., microfiche or paper copy;
OF89–557–B, OS/2 executable module, 3.5″ diskette;
OF89–557–C, OS/2 executable module, 5.25″ diskette;
OF89–557–D, OS/2/SAA source program, 3.5″ diskette;
OF89–557–E, OS/2/ANSI FORTRAN 77 source program, 3.5″ diskette;
OF89–557–F, OS/2/SAA source program, 5.25″ diskette;
OF89–557–G, OS/2/ANSI FORTRAN 77 source program, 5.25″ diskette;
OF89–557–H, DOS/SAA source program, 3.5″ diskette;
OF89–557–J, DOS/ANSI FORTRAN 77 source program, 3.5″ diskette;
OF89–557–K, DOS/SAA source program, 5.25″ diskette;
OF89–557–L, DOS/ANSI FORTRAN 77 source program, 5.25″ diskette;
OF89–557–M, DOS executable module, 3.5″ diskette;
OF89–557–N, DOS executable module, 2–5.25″ diskettes in BACKUP format.

All diskettes contain the described programs in ASCII format in a two level directory tree except for OF89–557–N. For it, use RESTORE with the /S option which creates a directory \SEIS3 on the desired drive. All versions are accompanied by a set of test data.