308 ——— Q199902220001
Scientific Notebook # 221
(Documentation on tested
conduct for Modules of the

21
150
R

MARK JARZEMBA

CNWRA

The Boorum & Pease® Quality Guarantee

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

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

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

One Good Book Deserves Many Others.

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

Made in U.S.A.

5/16/97 MSJ

This notebook documents the testing and modification excercizes that I will conduct to Modules of the TPA code.

First task: DCAGS and DCAGW modules need to be modified to be able to use different dose conversion factors for a pluvial biosphere.

Two variables will need to be input to the modules in order to accomplish this: mean annual temperature (MAT) and mean annual precipitation (MAP) at the time of conversion.

The following pages show a printout of the modules in their current state.

```
c=======================================================================
      subroutine dcags( mxntime, mxnnucl,
     &        ntim, tim, nnucl, names,
     &        ciperm2gsatCP,
     &        remperyrpernuclgs  )
c=======================================================================
c dose through groundsurface pathway due to laydown of contaminated ashblanket
c compliance point which may be locate 25, 30, 35 km from repository
c
c mxntime = input, integer, size to dimension arrays
c mxnnucl = input, integer, size to dimension arrays
c ntim = input, integer, maximum number of times used to dimension arrays
c tim(ntim)  = input, double precision, array of times
c nnucl = input, integer, maximum number of nuclides used to dimension arrays
c names(nnucl) = input, character*6, names of nuclides to be tracked
c ciperm2gsatCP(mxntime,mxnnucl) = input, double precision, time-dependent
c               concentration [ci/m^2] of nuclides at ground surface
c               due to volcanic ashplume.  A separate module accounts for
c               the evolution of nuclides after the ash is layed down.
c               Hence, the errosion and leaching of nuclides is already
c               accounted for, and this module only converts a given
c               concentration [ci/m2] to annual effective dose equivalent
c               [rem/yr].
c remperyrpernuclgs[mxntime,nnucl] = output, double precision, array of
c             annual EDE (effective dose equivalent) per nuclide
c
c     This subroutine was re-written on 4/17/97 by Mark Jarzemba
c     in order to have a duality of how doses are calculated.
c     For compliance points closer to the repository than the cutoff pt,
c     the dcf's are based on direct exposure and inhalation only.
c     For points at a distance of greater than the cutoff pt. from the
c     repository, the dcf's are based on direct exposure, inhalation,
c     and ingestion of contaminated animal products and crops.
c
c
c
c
      implicit double precision (a-h,o-z)

      dimension tim(ntim)
      character*6 names(nnucl)
      character*60 name
      dimension ciperm2gsatCP(mxntime,mxnnucl)
      dimension remperyrpernuclgs(mxntime, mxnnucl)
      external ispquery
      external valuesp

      common / dcags1 / ikey
      common / dcags2 / dcf(43)

      if (ikey .ne. 73562 ) then
cc to avoid warning message, make trivial use of tim()
        ajunk = tim(1)

        call clearchar( 60, name )
        name = 'DistanceCutoffForDoseConversionDualityInDCAGS[km]'
        icutoffpt = ispquery( name )

        call clearchar( 60, name )
        name = 'DistanceToCriticalGroup[km][should be 5 or 30]'
        idist = ispquery( name )

        ikey = 73562

      endif
```



```
cutoffpt=valuesp( icutoffpt )
dist=valuesp( idist )

if (dist.ge.cutoffpt) then


c
c  These dcf's are for the Amargosa Desert farmer/rancher where
c  direct exposure, inhalation, and ingestion of animal products
c  and crops has been included

      dcf( 1) =  0.66D+04
      dcf( 2) =  0.72D+06
      dcf( 3) =  0.95D+04
      dcf( 4) =  0.67D+06
      dcf( 5) =  0.00D+00
      dcf( 6) =  0.57D+04
      dcf( 7) =  0.10D+06
      dcf( 8) =  0.24D+06
      dcf( 9) =  0.12D+07
      dcf(10) =  0.49D+06
      dcf(11) =  0.70D+06
      dcf(12) =  0.10D+05
      dcf(13) =  0.19D+05
      dcf(14) =  0.21D+07
      dcf(15) =  0.27D+07
      dcf(16) =  0.72D+06
      dcf(17) =  0.24D+03
      dcf(18) =  0.70D+06
      dcf(19) =  0.11D+07
      dcf(20) =  0.58D+04
      dcf(21) =  0.73D+06
      dcf(22) =  0.39D+06
      dcf(23) =  0.10D+05
      dcf(24) =  0.54D+04
      dcf(25) =  0.18D+05
      dcf(26) =  0.11D+03
      dcf(27) =  0.12D+06
      dcf(28) =  0.10D+05
      dcf(29) =  0.33D+06
      dcf(30) =  0.43D+05
      dcf(31) =  0.46D+04
      dcf(32) =  0.00D+00
      dcf(33) =  0.89D+02
      dcf(34) =  0.46D+04
      dcf(35) =  0.11D+04
      dcf(36) =  0.13D+06
      dcf(37) =  0.31D+03
      dcf(38) =  0.73D+05
      dcf(39) =  0.52D+04
      dcf(40) =  0.42D+03
      dcf(41) =  0.15D+03
      dcf(42) =  0.69D+05
      dcf(43) =  0.14D+01

else

c
c  These dcf's are for the closer in residential water user
c  and include only direct exposure and inhalation.

      dcf(1) = 0.24D+03
      dcf(2) = 0.79D+03
```

```
      dcf(3) = 0.52D+03
      dcf(4) = 0.93D+03
      dcf(5) = 0.48D+03
      dcf(6) = 0.28D+03
      dcf(7) = 0.48D+03
      dcf(8) = 0.57D+03
      dcf(9) = 0.24D+03
      dcf(10) = 0.11D+05
      dcf(11) = 0.53D+04
      dcf(12) = 0.51D+03
      dcf(13) = 0.13D+05
      dcf(14) = 0.49D+04
      dcf(15) = 0.21D+04
      dcf(16) = 0.83D+04
      dcf(17) = 0.80D+01
      dcf(18) = 0.53D+04
      dcf(19) = 0.36D+04
      dcf(20) = 0.27D+03
      dcf(21) = 0.10D+05
      dcf(22) = 0.47D+03
      dcf(23) = 0.55D+03
      dcf(24) = 0.26D+03
      dcf(25) = 0.11D+04
      dcf(26) = 0.49D+00
      dcf(27) = 0.48D+05
      dcf(28) = 0.30D+01
      dcf(29) = 0.22D+04
      dcf(30) = 0.48D+04
      dcf(31) = 0.43D+03
      dcf(32) = 0.00D+00
      dcf(33) = 0.22D-01
      dcf(34) = 0.69D+01
      dcf(35) = 0.46D+03
      dcf(36) = 0.13D+06
      dcf(37) = 0.13D+00
      dcf(38) = 0.24D+02
      dcf(39) = 0.18D+01
      dcf(40) = 0.35D-02
      dcf(41) = 0.14D-02
      dcf(42) = 0.59D+02
      dcf(43) = 0.14D+01


    end if

    do k = 1, nnucl
      ik = indexperiso( names(k) )
      do j=1,ntim
        remperyrpernuclgs(j,k)=dcf(ik)*ciperm2gsatCP(j,k)
      enddo
    enddo

    return
    end
```



```
c=======================================================================
      subroutine dcagw( mxntime, mxnnucl,
     &        ntim, tim, nnucl, names,
     &        qm3peryrallsafromsz,
     &        ciperyrallsafromsz,
     &        remperyrpernuclgw )
c=======================================================================
c calculates dose through groundwater pathway by extraction at well.
c calculates saturated zone flow from below repository to compliance
c point which may be located 5,25, 30, 35 km from repository
c
c if CP is less than 20 km, then use qm3peryrallsafromsz to calculate
c concentration of nuclides in GW.
c if CP is greater than 20 km, then use well pumping rate to calculate
c concentration of nuclides in GW, assuming all nuclides extracted
c from system by pumping
c
c mxntime = input, integer, size to dimension arrays
c mxnnucl = input, integer, size to dimension arrays
c ntim = input, integer, maximum number of times used to dimension arrays
c tim(ntim)  = input, double precision, array of times
c nnucl = input, integer, maximum number of nuclides used to dimension arrays
c names(nnucl) = input, character*6, names of nuclides to be tracked
c qm3peryrallsafromsz(mxntime) =  input, double precision, groundwater flow
c                   rate [m^3/yr] from all subareas from the saturated zone
c                   to the compliance point
c ciperyrallsafromsz(mxntime,mxnnucl) = input, double precision, nuclide
c                   release rate [Ci/yr] from all subareas from the
c                   saturated zone at the compliance point
c remperyrpernuclgw[mxntime,nnucl] = output, double precision, array of
c             annual EDE (effective dose equivalent) per nuclide
c
      implicit double precision (a-h,o-z)

      dimension tim(ntim)
      character*6 names(nnucl)
      dimension qm3peryrallsafromsz(mxntime)
      dimension ciperyrallsafromsz(mxntime,mxnnucl)
      dimension remperyrpernuclgw(mxntime, mxnnucl)

      character*60 name
      common / dcagw1 / ikey
      common / dcagw2 / dcf(43)
      common / dcagw3 / idist
      common / dcagw4 / dist

cc      common / dcagw5  / iownpeakdosev
cc      common / dcagw6  / iownpeakdoset
cc      common / dcagw7  / ipeakdosev
cc      common / dcagw8  / ipeakdoset
cc      common / dcagw9  / iownpeakdosenv(43)
cc      common / dcagw10 / iownpeakdosent(43)
cc      common / dcagw11 / ipeakdosenv(43)
cc      common / dcagw12 / ipeakdosent(43)
      common / dcagw13 / ipump

      external valuesp

cc      print *, ' dcagw: entered into dcagw '

      if( (nnucl .le. 0) .or. (nnucl .gt. 43) ) then
        print *, ' ***>>> Error in DCAGW <<<*** '
        print *, ' (nnucl .le. 0)  .or. (nnucl .gt. 43)'
        print *, ' nnucl = ', nnucl
```

```
        include 'stop.i'
      endif
      if( ntim .gt. mxntime ) then
        print *, ' ***>>> Error in DCAGW <<<*** '
        print *, ' ntim .gt. mxntime '
        print *, ' mxntime = ', mxntime
        print *, ' ntim    = ', ntim
        include 'stop.i'
      endif
      if( ntim .le. 0 ) then
        print *, ' ***>>> Error in DCAGW <<<*** '
        print *, ' ntim .le. 0 '
        print *, ' ntim = ', ntim
        include 'stop.i'
      endif

      if (ikey .ne. 39231) then
cc to avoid warning message, make trivial use of tim()
        ajunk = tim(1)

        call clearchar( 60, name )
        name = 'DistanceToCriticalGroup[km][should be 5 or 30]'
        idist = ispquery( name )
        dist  = valuesp( idist )

        call clearchar( 60, name )
        name = 'WellPumpingRateAtCriticalGroup[gal/day]'
        ipump = ispquery( name )

cc        call clearchar( 60, name )
cc        name = 'Peak Annual Dose [rem/yr]'
cc        iownpeakdosev = iaddconsmv( name )
cc        ipeakdosev = imvquery( name )

cc        call clearchar( 60, name )
cc        name = 'Time of Peak Annual Dose [yr]'
cc        iownpeakdoset = iaddconsmv( name )
cc        ipeakdoset = imvquery( name )

cc        do ii = 1, nnucl
cc         call clearchar( 60, name )
cc         name = 'Peak Annual Dose from ' // names(ii) // ' [rem/yr]'
cc         iownpeakdosenv(ii) = iaddconsmv( name )
cc         ipeakdosenv(ii) = imvquery( name )
cc         call clearchar( 60, name )
cc         name = 'Time of Peak Annual Dose from ' // names(ii) // ' [yr]'
cc         iownpeakdosent(ii) = iaddconsmv( name )
cc         ipeakdosent(ii) = imvquery( name )
cc        enddo

        if( dist .lt. 4.9999999d0 ) then
          print *, ' ***>>> Error in DCAGW <<<*** '
          print *, ' dist .lt. 5.0 [km] '
          print *, ' dist = ', dist
          include 'stop.i'
        elseif( dist .lt. 20.0d0 ) then
cc use drinking water dose conversion factors if the distance to the
cc critical group is less than 20 km away from the source.
cc
cc data from EPA, 1988, Limiting Values of Radionuclide INtake and Air
cc Concentration and Dose Conversion Factors for Inhalation,
cc Submersion and Ingestion. EPA-520/1-88-020, Washington, DC
cc Environmental Protection Agency
cc which is same data as DOE, 1988, Internal Dose Conversion Factors
cc for Calculation of Dose to the Public. DOE/EH-0071, Washington, DC,
cc U.S. Department of Energy
cc
          dcf( 1) = 1.86d+05
          dcf( 2) = 2.70d+06
          dcf( 3) = 2.45d+06
          dcf( 4) = 2.57d+06
          dcf( 5) = 2.34d+06
          dcf( 6) = 2.07d+05
          dcf( 7) = 4.00d+05
          dcf( 8) = 9.67d+05
          dcf( 9) = 3.91d+06
          dcf(10) = 1.83d+06
          dcf(11) = 2.64d+06
          dcf(12) = 2.58d+06
          dcf(13) = 1.94d+05
          dcf(14) = 7.72d+06
          dcf(15) = 1.03d+07
          dcf(16) = 2.73d+06
          dcf(17) = 5.00d+04
          dcf(18) = 2.66d+06
          dcf(19) = 3.24d+06
          dcf(20) = 2.11d+05
          dcf(21) = 2.58d+06
          dcf(22) = 1.47d+06
          dcf(23) = 2.58d+06
          dcf(24) = 1.96d+05
          dcf(25) = 9.56d+05
          dcf(26) = 2.84d+02
          dcf(27) = 3.65d+04
          dcf(28) = 5.16d+03
          dcf(29) = 2.01d+05
          dcf(30) = 1.42d+04
          dcf(31) = 1.13d+03
          dcf(32) = 5.56d+03
          dcf(33) = 1.09d+02
          dcf(34) = 1.07d+03
          dcf(35) = 9.83d+02
          dcf(36) = 5.21d+03
          dcf(37) = 1.21d+03
          dcf(38) = 1.04d+05
          dcf(39) = 6.35d+03
          dcf(40) = 4.21d+02
          dcf(41) = 1.53d+02
          dcf(42) = 2.21d+03
          dcf(43) = 1.52d+03

        else
cc
cc use mean groundwater pathway dose conversion factors
cc if the distance from source to critical group is more than
cc 20 km away. The DCFs are in units of [(rem/yr)/(pCi/liter)]
cc to convert to units of [(rem/yr)/(Ci/m^3)],
cc multiply by 10^9
cc
cc data from P.A. LaPlante, S.J. Maheras, and M.S. Jarzemba
cc Initial Analysis of Selected Site-Specific Dose Assessment
cc Parameters and Exposure Pathways Applicable to a Groundwater
cc Release Scenario at Yucca Mountain, CNWRA 95-018,
cc September, 1995, San Antonio, TX, Center for Nuclear Waste
cc Regulatory Analyses
cc
          dcf( 1) =  0.72d-04*1.0d9
```

```
      dcf( 2) =  0.81d-02*1.0d9
      dcf( 3) =  0.10d-03*1.0d9
      dcf( 4) =  0.76d-02*1.0d9
      dcf( 5) =  0.00d+00*1.0d9
      dcf( 6) =  0.60d-04*1.0d9
      dcf( 7) =  0.12d-02*1.0d9
      dcf( 8) =  0.28d-02*1.0d9
      dcf( 9) =  0.13d-01*1.0d9
      dcf(10) =  0.54d-02*1.0d9
      dcf(11) =  0.79d-02*1.0d9
      dcf(12) =  0.11d-03*1.0d9
      dcf(13) =  0.84d-04*1.0d9
      dcf(14) =  0.23d-01*1.0d9
      dcf(15) =  0.31d-01*1.0d9
      dcf(16) =  0.81d-02*1.0d9
      dcf(17) =  0.32d-05*1.0d9
      dcf(18) =  0.79d-02*1.0d9
      dcf(19) =  0.13d-01*1.0d9
      dcf(20) =  0.61d-04*1.0d9
      dcf(21) =  0.81d-02*1.0d9
      dcf(22) =  0.43d-02*1.0d9
      dcf(23) =  0.11d-03*1.0d9
      dcf(24) =  0.57d-04*1.0d9
      dcf(25) =  0.24d-03*1.0d9
      dcf(26) =  0.12d-05*1.0d9
      dcf(27) =  0.76d-03*1.0d9
      dcf(28) =  0.10d-03*1.0d9
      dcf(29) =  0.31d-02*1.0d9
      dcf(30) =  0.63d-03*1.0d9
      dcf(31) =  0.43d-04*1.0d9
      dcf(32) =  0.00d+00*1.0d9
      dcf(33) =  0.81d-06*1.0d9
      dcf(34) =  0.84d-05*1.0d9
      dcf(35) =  0.44d-05*1.0d9
      dcf(36) =  0.20d-03*1.0d9
      dcf(37) =  0.35d-05*1.0d9
      dcf(38) =  0.61d-03*1.0d9
      dcf(39) =  0.53d-04*1.0d9
      dcf(40) =  0.38d-05*1.0d9
      dcf(41) =  0.14d-05*1.0d9
      dcf(42) =  0.87d-04*1.0d9
      dcf(43) =  0.19d-04*1.0d9

    endif

    ikey=39231

  endif

  distnew = valuesp( idist )
cc      print *, ' dcagw: dist    = ', dist
cc      print *, ' dcagw: distnew = ', distnew
cc      print *, ' '

cc      print *, 'name ' , 'DCF ((rem/yr)/(Ci/m3))'
cc      do i = 1, nnucl
cc        ii = indexperiso( names(i) )
cc        print *, names(i), dcf(ii)
cc      enddo
cc      print *, ' '

  if( (distnew .lt. 20.0d0) .and. (dist .gt. 20.0d0) ) then
    print *, ' ***>>> Error in DCAGW <<<*** '
    print *, ' (distnew .lt. 20.0d0) .and. (dist .gt. 20.0d0) '
```



```
    print *, ' Distance to Critical Group must be either '
    print *, ' close in (drinking water pathway only) OR '
    print *, ' far away (groundwater pathway) '
    include 'stop.i'
  endif

  if( (distnew .gt. 20.0d0) .and. (dist .lt. 20.0d0) ) then
    print *, ' ***>>> Error in DCAGW <<<*** '
    print *, ' (distnew .gt. 20.0d0) .and. (dist .lt. 20.0d0) '
    print *, ' Distance to Critical Group must be either '
    print *, ' close in (drinking water pathway only) OR '
    print *, ' far away (groundwater pathway) '
    include 'stop.i'
  endif
  if( dist .lt. 20.0d0 ) then
    do k = 1, nnucl
      ik = indexperiso( names(k) )
      do j=1,ntim
        if( ciperyrallsafromsz(j,k) .lt. 0.0d0 ) then
          print *, ' ***>>> Error in DCAGW <<<*** '
          print *, ' ciperyrallsafromsz(j,k) .lt. 0.0d0 '
          print *, ' ciperyrallsafromsz(j,k) = ', ciperyrallsafromsz(j,k)
          print *, ' j = ', j
          print *, ' k = ', k
          print *, ' '
          include 'stop.i'
        endif
        remperyrpernuclgw(j,k)=dcf(ik)*ciperyrallsafromsz(j,k) /
&         qm3peryrallsafromsz(j)
      enddo
    enddo
  else
        pump = valuesp( ipump )
cc        print *, ' dcagw: pump = ', pump, ' [gal/day] '
cc
cc 10^6 Gallon = 3785.41 Meter^3
cc 1 Year = 365.25 Day
cc 1 Meter^3 = 264.172 Gallon
cc
cc (Ci/m^3)   = (Ci/yr)*(Gal/day)*(day/yr)*(m^3/Gallon)
cc pumprate = (m^3/yr) = (Gal/day)*(365day/yr)*(m^3/264Gallon)
        pumprate =  ( pump * 365.25d0 / 264.172d0 )
cc        print *, ' dcagw: pumprate = ', pumprate, ' [m^3/yr] '
        if( pumprate .lt. qm3peryrallsafromsz(ntim) ) then
          print *, ' ***>>> Error in DCAGW <<<*** '
          print *, ' pumprate .lt. qm3peryrallsafromsz '
          print *, ' pumprate   = ', pumprate
          print *, ' qm3peryrallsafromsz(ntim) = ',
&                  qm3peryrallsafromsz(ntim)
          print *, ' '
          include 'stop.i'
        endif
        do k = 1, nnucl
          ik = indexperiso( names(k) )
          do j=1,ntim
            if( ciperyrallsafromsz(j,k) .lt. 0.0d0 ) then
              print *, ' ***>>> Error in DCAGW <<<*** '
              print *, ' ciperyrallsafromsz(j,k) .lt. 0.0d0 '
              print *, ' ciperyrallsafromsz(j,k) = ', ciperyrallsafromsz(j,k)
              print *, ' j = ', j
              print *, ' k = ', k
              print *, ' '
              include 'stop.i'
            endif
```

```
     &        remperyrpernuclgw(j,k)=dcf(ik)*ciperyrallsafromsz(j,k) /
     &           pumprate
           enddo
         enddo
       endif

       return
       end
```

5/19/97 MSJ

- The "dividing line" between a pluvial and non-pluvial biosphere is based on the Köeppen-Geiger climate classification scheme. This initial solution to this problem is an

The boundary between arid and semi-arid in this scheme is given as

MAP ≤ 0.22 (MAT - 32) arid
MAP > 0.22 (MAT - 32) semi-arid

MAT = mean annual temperature (Fahrenheit)
MAP = mean annual precipitation (inches)

- the following is a new version of the code


```
c=======================================================================
      subroutine dcags( mxntime, mxnnucl,
     &         ntim, tim, nnucl, names,
     &         dMAT,dMAP,ciperm2gsatCP,
     &         remperyrpernuclgs  )
c=======================================================================
c dose through groundsurface pathway due to laydown of contaminated ashblanket
c compliance point which may be locate 25, 30, 35 km from repository
c
c mxntime = input, integer, size to dimension arrays
c mxnnucl = input, integer, size to dimension arrays
c ntim = input, integer, maximum number of times used to dimension arrays
c tim(ntim)  = input, double precision, array of times
c nnucl = input, integer, maximum number of nuclides used to dimension arrays
c names(nnucl) = input, character*6, names of nuclides to be tracked
c ciperm2gsatCP(mxntime,mxnnucl) = input, double precision, time-dependent
c                concentration [ci/m^2] of nuclides at ground surface
c                due to volcanic ashplume.  A separate module accounts for
c                the evolution of nuclides after the ash is layed down.
c                Hence, the errosion and leaching of nuclides is already
c                accounted for, and this module only converts a given
c                concentration [ci/m2] to annual effective dose equivalent
c                [rem/yr].
c remperyrpernuclgs[mxntime,nnucl] = output, double precision, array of
c              annual EDE (effective dose equivalent) per nuclide
c
c      This subroutine was re-written on 4/17/97 by Mark Jarzemba
c      in order to have a duality of how doses are calculated.
c      For compliance points closer to the repository than the cutoff pt,
c      the dcf's are based on direct exposure and inhalation only.
c      For points at a distance of greater than the cutoff pt. from the
c      repository, the dcf's are based on direct exposure, inhalation,
c      and ingestion of contaminated animal products and crops.
c
c      This subroutine was re-re-written on 5/19/97 by Mark Jarzemba
c      in order to have another duality of how doses are calculated.
c      For points in time where the mean annual temperature (dMAT) and
c      mean annual precipitation (dMAP) are such that the climate is
c      re-classified as a semi-arid climate according to the Koeppen-
c      Geiger climate classification scheme.
c
c      dMAT in degrees fahrenheit
c      dMAP in inches
c
       implicit double precision (a-h,o-z)

       dimension tim(ntim)
       character*6 names(nnucl)
       character*60 name
       dimension ciperm2gsatCP(mxntime,mxnnucl)
       dimension remperyrpernuclgs(mxntime, mxnnucl)
       external ispquery
       external valuesp

       common / dcags1 / ikey
       common / dcags2 / dcf(43)

       if (ikey .ne. 73562 ) then
cc to avoid warning message, make trivial use of tim()
         ajunk = tim(1)

         call clearchar( 60, name )
         name = 'DistanceCutoffForDoseConversionDualityInDCAGS[km]'
         icutoffpt = ispquery( name )
```

```
      call clearchar( 60, name )
      name = 'DistanceToCriticalGroup[km][should be 5 or 30]'
      idist = ispquery( name )

      ikey = 73562

    endif

    cutoffpt=valuesp( icutoffpt )
    dist=valuesp( idist )
    dKGboundary=0.22d0*(dMAT-32.0d0)
    if (dist.ge.cutoffpt) then


c     These dcf's are for the Amargosa Desert farmer/rancher where
c     direct exposure, inhalation, and ingestion of animal products
c     and crops has been included

      if(dMAP.lt.dKGboundary) then

c     These DCF's are for an ADFR receptor group in today's biosphere.
c     As of 5/19/97, there are no differences between the pluvial and
c     current biosphere DCF's.

      dcf( 1) =  0.66D+04
      dcf( 2) =  0.72D+06
      dcf( 3) =  0.95D+04
      dcf( 4) =  0.67D+06
      dcf( 5) =  0.00D+00
      dcf( 6) =  0.57D+04
      dcf( 7) =  0.10D+06
      dcf( 8) =  0.24D+06
      dcf( 9) =  0.12D+07
      dcf(10) =  0.49D+06
      dcf(11) =  0.70D+06
      dcf(12) =  0.10D+05
      dcf(13) =  0.19D+05
      dcf(14) =  0.21D+07
      dcf(15) =  0.27D+07
      dcf(16) =  0.72D+06
      dcf(17) =  0.24D+03
      dcf(18) =  0.70D+06
      dcf(19) =  0.11D+07
      dcf(20) =  0.58D+04
      dcf(21) =  0.73D+06
      dcf(22) =  0.39D+06
      dcf(23) =  0.10D+05
      dcf(24) =  0.54D+04
      dcf(25) =  0.18D+05
      dcf(26) =  0.11D+03
      dcf(27) =  0.12D+06
      dcf(28) =  0.10D+05
      dcf(29) =  0.33D+06
      dcf(30) =  0.43D+05
      dcf(31) =  0.46D+04
      dcf(32) =  0.00D+00
      dcf(33) =  0.89D+02
      dcf(34) =  0.46D+04
      dcf(35) =  0.11D+04
      dcf(36) =  0.13D+06
      dcf(37) =  0.31D+03
      dcf(38) =  0.73D+05
      dcf(39) =  0.52D+04
      dcf(40) =  0.42D+03
      dcf(41) =  0.15D+03
      dcf(42) =  0.69D+05
      dcf(43) =  0.14D+01
    else if(dMAP.ge.dKGboundary) then

c     These DCF's are for an ADFR receptor group in a  pluvial biosphere.
c     As of 5/19/97, there are no differences between the pluvial and
c     current biosphere DCF's.

      dcf( 1) =  0.66D+04
      dcf( 2) =  0.72D+06
      dcf( 3) =  0.95D+04
      dcf( 4) =  0.67D+06
      dcf( 5) =  0.00D+00
      dcf( 6) =  0.57D+04
      dcf( 7) =  0.10D+06
      dcf( 8) =  0.24D+06
      dcf( 9) =  0.12D+07
      dcf(10) =  0.49D+06
      dcf(11) =  0.70D+06
      dcf(12) =  0.10D+05
      dcf(13) =  0.19D+05
      dcf(14) =  0.21D+07
      dcf(15) =  0.27D+07
      dcf(16) =  0.72D+06
      dcf(17) =  0.24D+03
      dcf(18) =  0.70D+06
      dcf(19) =  0.11D+07
      dcf(20) =  0.58D+04
      dcf(21) =  0.73D+06
      dcf(22) =  0.39D+06
      dcf(23) =  0.10D+05
      dcf(24) =  0.54D+04
      dcf(25) =  0.18D+05
      dcf(26) =  0.11D+03
      dcf(27) =  0.12D+06
      dcf(28) =  0.10D+05
      dcf(29) =  0.33D+06
      dcf(30) =  0.43D+05
      dcf(31) =  0.46D+04
      dcf(32) =  0.00D+00
      dcf(33) =  0.89D+02
      dcf(34) =  0.46D+04
      dcf(35) =  0.11D+04
      dcf(36) =  0.13D+06
      dcf(37) =  0.31D+03
      dcf(38) =  0.73D+05
      dcf(39) =  0.52D+04
      dcf(40) =  0.42D+03
      dcf(41) =  0.15D+03
      dcf(42) =  0.69D+05
      dcf(43) =  0.14D+01
    end if

    else

c     These dcf's are for the closer in residential water user
c     and include only direct exposure and inhalation.

      if(dMAP.lt.dKGboundary) then

c     These DCF's are for an CIRWU receptor group in today's biosphere.
```

(rem/yr)/(Ci/m²)

(rem/yr)/(Ci/m²)

```
c     As of 5/19/97, there are no differences between the pluvial and
c     current biosphere DCF's.
c
      dcf(1)  = 0.24D+03
      dcf(2)  = 0.79D+03
      dcf(3)  = 0.52D+03
      dcf(4)  = 0.93D+03
      dcf(5)  = 0.48D+03
      dcf(6)  = 0.28D+03
      dcf(7)  = 0.48D+03
      dcf(8)  = 0.57D+03
      dcf(9)  = 0.24D+03
      dcf(10) = 0.11D+05
      dcf(11) = 0.53D+04
      dcf(12) = 0.51D+03
      dcf(13) = 0.13D+05
      dcf(14) = 0.49D+04
      dcf(15) = 0.21D+04
      dcf(16) = 0.83D+04
      dcf(17) = 0.80D+01
      dcf(18) = 0.53D+04
      dcf(19) = 0.36D+04
      dcf(20) = 0.27D+03
      dcf(21) = 0.10D+05
      dcf(22) = 0.47D+03
      dcf(23) = 0.55D+03
      dcf(24) = 0.26D+03
      dcf(25) = 0.11D+04
      dcf(26) = 0.49D+00
      dcf(27) = 0.48D+05
      dcf(28) = 0.30D+01
      dcf(29) = 0.22D+04
      dcf(30) = 0.48D+04
      dcf(31) = 0.43D+03
      dcf(32) = 0.00D+00
      dcf(33) = 0.22D-01
      dcf(34) = 0.69D+01
      dcf(35) = 0.46D+03
      dcf(36) = 0.13D+06
      dcf(37) = 0.13D+00
      dcf(38) = 0.24D+02
      dcf(39) = 0.18D+01
      dcf(40) = 0.35D-02
      dcf(41) = 0.14D-02
      dcf(42) = 0.59D+02
      dcf(43) = 0.14D+01

      else if(dMAP.ge.dKGboundary) then

c     These DCF's are for an CIRWU receptor group in a pluvial biosphere.
c     As of 5/19/97, there are no differences between the pluvial and
c     current biosphere DCF's.
c
      dcf(1)  = 0.24D+03
      dcf(2)  = 0.79D+03
      dcf(3)  = 0.52D+03
      dcf(4)  = 0.93D+03
      dcf(5)  = 0.48D+03
      dcf(6)  = 0.28D+03
      dcf(7)  = 0.48D+03
      dcf(8)  = 0.57D+03
      dcf(9)  = 0.24D+03
      dcf(10) = 0.11D+05
      dcf(11) = 0.53D+04
      dcf(12) = 0.51D+03
      dcf(13) = 0.13D+05
      dcf(14) = 0.49D+04
      dcf(15) = 0.21D+04
      dcf(16) = 0.83D+04
      dcf(17) = 0.80D+01
      dcf(18) = 0.53D+04
      dcf(19) = 0.36D+04
      dcf(20) = 0.27D+03
      dcf(21) = 0.10D+05
      dcf(22) = 0.47D+03
      dcf(23) = 0.55D+03
      dcf(24) = 0.26D+03
      dcf(25) = 0.11D+04
      dcf(26) = 0.49D+00
      dcf(27) = 0.48D+05
      dcf(28) = 0.30D+01
      dcf(29) = 0.22D+04
      dcf(30) = 0.48D+04
      dcf(31) = 0.43D+03
      dcf(32) = 0.00D+00
      dcf(33) = 0.22D-01
      dcf(34) = 0.69D+01
      dcf(35) = 0.46D+03
      dcf(36) = 0.13D+06
      dcf(37) = 0.13D+00
      dcf(38) = 0.24D+02
      dcf(39) = 0.18D+01
      dcf(40) = 0.35D-02
      dcf(41) = 0.14D-02
      dcf(42) = 0.59D+02
      dcf(43) = 0.14D+01
     end if
    end if

    do k = 1, nnucl
      ik = indexperiso( names(k) )
      do j=1,ntim
        remperyrpernuclgs(j,k)=dcf(ik)*ciperm2gsatCP(j,k)
      enddo
    enddo

    return
    end
```

(rem/yr)/(Ci/m²)

(rem/yr)/(Ci/m²)

rem/yr = (rem/yr)/(Ci/m²) · Ci/m²

Similar mod's were made to DCAGW.f and are shown in the following printout.

```
c=======================================================================
      subroutine dcagw( mxntime, mxnnucl,
     &        ntim, tim, nnucl, names,
     &        dMAT,dMAP,qm3peryrallsafromsz,
     &        ciperyrallsafromsz,
     &        remperyrpernuclgw )
c=======================================================================
c calculates dose through groundwater pathway by extraction at well.
c calculates saturated zone flow from below repository to compliance
c point which may be located 5,25, 30, 35 km from repository
c
c if CP is less than 20 km, then use qm3peryrallsafromsz to calculate
c concentration of nuclides in GW.
c if CP is greater than 20 km, then use well pumping rate to calculate
c concentration of nuclides in GW, assuming all nuclides extracted
c from system by pumping
c
c mxntime = input, integer, size to dimension arrays
c mxnnucl = input, integer, size to dimension arrays
c ntim = input, integer, maximum number of times used to dimension arrays
c tim(ntim)  = input, double precision, array of times
c nnucl = input, integer, maximum number of nuclides used to dimension arrays
c names(nnucl) = input, character*6, names of nuclides to be tracked
c qm3peryrallsafromsz(mxntime) =  input, double precision, groundwater flow
c                   rate [m^3/yr] from all subareas from the saturated zone
c                   to the compliance point
c ciperyrallsafromsz(mxntime,mxnnucl) = input, double precision, nuclide
c                   release rate [Ci/yr] from all subareas from the
c                   saturated zone at the compliance point
c remperyrpernuclgw[mxntime,nnucl] = output, double precision, array of
c             annual EDE (effective dose equivalent) per nuclide
c
      implicit double precision (a-h,o-z)
      dimension tim(ntim)
      character*6 names(nnucl)
      dimension qm3peryrallsafromsz(mxntime)
      dimension ciperyrallsafromsz(mxntime,mxnnucl)
      dimension remperyrpernuclgw(mxntime, mxnnucl)

      character*60 name
      common / dcagw1 / ikey
      common / dcagw2 / dcf(43)
      common / dcagw3 / idist
      common / dcagw4 / dist

cc      common / dcagw5  / iownpeakdosev
cc      common / dcagw6  / iownpeakdoset
cc      common / dcagw7  / ipeakdosev
cc      common / dcagw8  / ipeakdoset
cc      common / dcagw9  / iownpeakdosenv(43)
cc      common / dcagw10 / iownpeakdosent(43)
cc      common / dcagw11 / ipeakdosenv(43)
cc      common / dcagw12 / ipeakdosent(43)
      common / dcagw13 / ipump

      external valuesp

cc      print *, ' dcagw: entered into dcagw '

      if( (nnucl .le. 0) .or. (nnucl .gt. 43) ) then
        print *, ' ***>>> Error in DCAGW <<<*** '
        print *, ' (nnucl .le. 0)  .or. (nnucl .gt. 43)'
        print *, ' nnucl = ', nnucl
        include 'stop.i'
      endif
      if( ntim .gt. mxntime ) then
        print *, ' ***>>> Error in DCAGW <<<*** '
        print *, ' ntim .gt. mxntime '
        print *, ' mxntime = ', mxntime
        print *, ' ntim    = ', ntim
        include 'stop.i'
      endif
      if( ntim .le. 0 ) then
        print *, ' ***>>> Error in DCAGW <<<*** '
        print *, ' ntim .le. 0 '
        print *, ' ntim = ', ntim
        include 'stop.i'
      endif

      if (ikey .ne. 39231) then
cc to avoid warning message, make trivial use of tim()
        ajunk = tim(1)

        call clearchar( 60, name )
        name = 'DistanceToCriticalGroup[km][should be 5 or 30]'
        idist = ispquery( name )
        dist  = valuesp( idist )

        call clearchar( 60, name )
        name = 'WellPumpingRateAtCriticalGroup[gal/day]'
        ipump = ispquery( name )

        call clearchar( 60, name )
        name = 'DistanceCutoffForDoseConversionDualityInDCAGS[km]'
        icutoffpt = ispquery( name )
      cutoffpt = valuesp( icutoffpt)

cc        call clearchar( 60, name )
cc        name = 'Peak Annual Dose [rem/yr]'
cc        iownpeakdosev = iaddconsmv( name )
cc        ipeakdosev = imvquery( name )

cc        call clearchar( 60, name )
cc        name = 'Time of Peak Annual Dose [yr]'
cc        iownpeakdoset = iaddconsmv( name )
cc        ipeakdoset = imvquery( name )

cc        do ii = 1, nnucl
cc         call clearchar( 60, name )
cc         name = 'Peak Annual Dose from ' // names(ii) // ' [rem/yr]'
cc         iownpeakdosenv(ii) = iaddconsmv( name )
cc         ipeakdosenv(ii) = imvquery( name )
cc         call clearchar( 60, name )
cc         name = 'Time of Peak Annual Dose from ' // names(ii) // ' [yr]'
cc         iownpeakdosent(ii) = iaddconsmv( name )
cc         ipeakdosent(ii) = imvquery( name )
cc        enddo

      dKGcutoffpt=0.22d0*(dMAT-32.d0)

       if( dist .lt. cutoffpt ) then
cc use drinking water dose conversion factors if the distance to the
cc critical group is less than 20 km away from the source.
cc
cc data from EPA, 1988, Limiting Values of Radionuclide INtake and Air
cc Concentration and Dose Conversion Factors for Inhalation,
cc Submersion and Ingestion. EPA-520/1-88-020, Washington, DC
```

```
cc Environmental Protection Agency
cc
cc which is same data as DOE, 1988, Internal Dose Conversion Factors
cc for Calculation of Dose to the Public. DOE/EH-0071, Washington, DC,
cc U.S. Department of Energy
cc
c
c      The following modification has been added by Mark Jarzemba
c      on 5/20/97 to have a duality in how doses are calculated.
c      If the dMAT and dMAP are such that the climate is classified
c      as semi-arid, use pluvial DCF's.
c
c      As of 5/20/97, there is no difference between the pluvial and
c      non-pluvial DCF's.
c
       if (dMAP.lt.dKGcutoffpt) then
c
c      These DCF's are for the CIRWU in today's biosphere
c
          dcf( 1) = 1.86d+05
          dcf( 2) = 2.70d+06
          dcf( 3) = 2.45d+06
          dcf( 4) = 2.57d+06
          dcf( 5) = 2.34d+06
          dcf( 6) = 2.07d+05
          dcf( 7) = 4.00d+05
          dcf( 8) = 9.67d+05
          dcf( 9) = 3.91d+06
          dcf(10) = 1.83d+06
          dcf(11) = 2.64d+06
          dcf(12) = 2.58d+06
          dcf(13) = 1.94d+05
          dcf(14) = 7.72d+06
          dcf(15) = 1.03d+07
          dcf(16) = 2.73d+06
          dcf(17) = 5.00d+04
          dcf(18) = 2.66d+06
          dcf(19) = 3.24d+06
          dcf(20) = 2.11d+05
          dcf(21) = 2.58d+06
          dcf(22) = 1.47d+06
          dcf(23) = 2.58d+06
          dcf(24) = 1.96d+05
          dcf(25) = 9.56d+05
          dcf(26) = 2.84d+02
          dcf(27) = 3.65d+04
          dcf(28) = 5.16d+03
          dcf(29) = 2.01d+05
          dcf(30) = 1.42d+04
          dcf(31) = 1.13d+03
          dcf(32) = 5.56d+03
          dcf(33) = 1.09d+02
          dcf(34) = 1.07d+03
          dcf(35) = 9.83d+02
          dcf(36) = 5.21d+03
          dcf(37) = 1.21d+03
          dcf(38) = 1.04d+05
          dcf(39) = 6.35d+03
          dcf(40) = 4.21d+02
          dcf(41) = 1.53d+02
          dcf(42) = 2.21d+03
          dcf(43) = 1.52d+03
       else if (dMAP.ge.dKGcutoffpt) then
c
```

(rem/yr)/(Ci/m³)


```
c      These DCF's are for the CIRWU in a pluvial biosphere
c
          dcf( 1) = 1.86d+05
          dcf( 2) = 2.70d+06
          dcf( 3) = 2.45d+06
          dcf( 4) = 2.57d+06
          dcf( 5) = 2.34d+06
          dcf( 6) = 2.07d+05
          dcf( 7) = 4.00d+05
          dcf( 8) = 9.67d+05
          dcf( 9) = 3.91d+06
          dcf(10) = 1.83d+06
          dcf(11) = 2.64d+06
          dcf(12) = 2.58d+06
          dcf(13) = 1.94d+05
          dcf(14) = 7.72d+06
          dcf(15) = 1.03d+07
          dcf(16) = 2.73d+06
          dcf(17) = 5.00d+04
          dcf(18) = 2.66d+06
          dcf(19) = 3.24d+06
          dcf(20) = 2.11d+05
          dcf(21) = 2.58d+06
          dcf(22) = 1.47d+06
          dcf(23) = 2.58d+06
          dcf(24) = 1.96d+05
          dcf(25) = 9.56d+05
          dcf(26) = 2.84d+02
          dcf(27) = 3.65d+04
          dcf(28) = 5.16d+03
          dcf(29) = 2.01d+05
          dcf(30) = 1.42d+04
          dcf(31) = 1.13d+03
          dcf(32) = 5.56d+03
          dcf(33) = 1.09d+02
          dcf(34) = 1.07d+03
          dcf(35) = 9.83d+02
          dcf(36) = 5.21d+03
          dcf(37) = 1.21d+03
          dcf(38) = 1.04d+05
          dcf(39) = 6.35d+03
          dcf(40) = 4.21d+02
          dcf(41) = 1.53d+02
          dcf(42) = 2.21d+03
          dcf(43) = 1.52d+03
       end if
       else
cc
cc use mean groundwater pathway dose conversion factors
cc if the distance from source to critical group is more than
cc 20 km away. The DCFs are in units of [(rem/yr)/(pCi/liter)]
cc to convert to units of [(rem/yr)/(Ci/m^3)],
cc multiply by 10^9
cc
cc data from P.A. LaPlante, S.J. Maheras, and M.S. Jarzemba
cc Initial Analysis of Selected Site-Specific Dose Assessment
cc Parameters and Exposure Pathways Applicable to a Groundwater
cc Release Scenario at Yucca Mountain, CNWRA 95-018,
cc September, 1995, San Antonio, TX, Center for Nuclear Waste
cc Regulatory Analyses
cc
       if (dMAP.lt.dKGcutoffpt) then
c
c      These DCF's are for the ADFR in today's biosphere
```

(rem/yr)/(Ci/m³)

```
c
      dcf( 1) =  0.72d-04*1.0d9
      dcf( 2) =  0.81d-02*1.0d9
      dcf( 3) =  0.10d-03*1.0d9
      dcf( 4) =  0.76d-02*1.0d9
      dcf( 5) =  0.00d+00*1.0d9
      dcf( 6) =  0.60d-04*1.0d9
      dcf( 7) =  0.12d-02*1.0d9
      dcf( 8) =  0.28d-02*1.0d9
      dcf( 9) =  0.13d-01*1.0d9
      dcf(10) =  0.54d-02*1.0d9
      dcf(11) =  0.79d-02*1.0d9
      dcf(12) =  0.11d-03*1.0d9
      dcf(13) =  0.84d-04*1.0d9
      dcf(14) =  0.23d-01*1.0d9
      dcf(15) =  0.31d-01*1.0d9
      dcf(16) =  0.81d-02*1.0d9
      dcf(17) =  0.32d-05*1.0d9
      dcf(18) =  0.79d-02*1.0d9
      dcf(19) =  0.13d-01*1.0d9
      dcf(20) =  0.61d-04*1.0d9
      dcf(21) =  0.81d-02*1.0d9
      dcf(22) =  0.43d-02*1.0d9
      dcf(23) =  0.11d-03*1.0d9
      dcf(24) =  0.57d-04*1.0d9
      dcf(25) =  0.24d-03*1.0d9
      dcf(26) =  0.12d-05*1.0d9
      dcf(27) =  0.76d-03*1.0d9
      dcf(28) =  0.10d-03*1.0d9
      dcf(29) =  0.31d-02*1.0d9
      dcf(30) =  0.63d-03*1.0d9
      dcf(31) =  0.43d-04*1.0d9
      dcf(32) =  0.00d+00*1.0d9
      dcf(33) =  0.81d-06*1.0d9
      dcf(34) =  0.84d-05*1.0d9
      dcf(35) =  0.44d-05*1.0d9
      dcf(36) =  0.20d-03*1.0d9
      dcf(37) =  0.35d-05*1.0d9
      dcf(38) =  0.61d-03*1.0d9
      dcf(39) =  0.53d-04*1.0d9
      dcf(40) =  0.38d-05*1.0d9
      dcf(41) =  0.14d-05*1.0d9
      dcf(42) =  0.87d-04*1.0d9
      dcf(43) =  0.19d-04*1.0d9
    else if (dMAP.ge.dKGcutoffpt) then
c
c     These DCF's are for the ADFR in a pluvial biosphere
c
      dcf( 1) =  0.72d-04*1.0d9
      dcf( 2) =  0.81d-02*1.0d9
      dcf( 3) =  0.10d-03*1.0d9
      dcf( 4) =  0.76d-02*1.0d9
      dcf( 5) =  0.00d+00*1.0d9
      dcf( 6) =  0.60d-04*1.0d9
      dcf( 7) =  0.12d-02*1.0d9
      dcf( 8) =  0.28d-02*1.0d9
      dcf( 9) =  0.13d-01*1.0d9
      dcf(10) =  0.54d-02*1.0d9
      dcf(11) =  0.79d-02*1.0d9
      dcf(12) =  0.11d-03*1.0d9
      dcf(13) =  0.84d-04*1.0d9
      dcf(14) =  0.23d-01*1.0d9
      dcf(15) =  0.31d-01*1.0d9
      dcf(16) =  0.81d-02*1.0d9
      dcf(17) =  0.32d-05*1.0d9
      dcf(18) =  0.79d-02*1.0d9
      dcf(19) =  0.13d-01*1.0d9
      dcf(20) =  0.61d-04*1.0d9
      dcf(21) =  0.81d-02*1.0d9
      dcf(22) =  0.43d-02*1.0d9
      dcf(23) =  0.11d-03*1.0d9
      dcf(24) =  0.57d-04*1.0d9
      dcf(25) =  0.24d-03*1.0d9
      dcf(26) =  0.12d-05*1.0d9
      dcf(27) =  0.76d-03*1.0d9
      dcf(28) =  0.10d-03*1.0d9
      dcf(29) =  0.31d-02*1.0d9
      dcf(30) =  0.63d-03*1.0d9
      dcf(31) =  0.43d-04*1.0d9
      dcf(32) =  0.00d+00*1.0d9
      dcf(33) =  0.81d-06*1.0d9
      dcf(34) =  0.84d-05*1.0d9
      dcf(35) =  0.44d-05*1.0d9
      dcf(36) =  0.20d-03*1.0d9
      dcf(37) =  0.35d-05*1.0d9
      dcf(38) =  0.61d-03*1.0d9
      dcf(39) =  0.53d-04*1.0d9
      dcf(40) =  0.38d-05*1.0d9
      dcf(41) =  0.14d-05*1.0d9
      dcf(42) =  0.87d-04*1.0d9
      dcf(43) =  0.19d-04*1.0d9
    end if
  endif

  ikey=39231

endif

 distnew = valuesp( idist )
cc      print *, ' dcagw: dist    = ', dist
cc      print *, ' dcagw: distnew = ', distnew
cc      print *, ' '

cc      print *, 'name ' , 'DCF ((rem/yr)/(Ci/m3))'
cc      do i = 1, nnucl
cc        ii = indexperiso( names(i) )
cc        print *, names(i), dcf(ii)
cc      enddo
cc      print *, ' '

 if( (distnew .lt. 20.0d0) .and. (dist .gt. 20.0d0) ) then
   print *, ' ***>>> Error in DCAGW <<<*** '
   print *, ' (distnew .lt. 20.0d0) .and. (dist .gt. 20.0d0) '
   print *, ' Distance to Critical Group must be either '
   print *, ' close in (drinking water pathway only) OR '
   print *, ' far away (groundwater pathway) '
   include 'stop.i'
 endif

 if( (distnew .gt. 20.0d0) .and. (dist .lt. 20.0d0) ) then
   print *, ' ***>>> Error in DCAGW <<<*** '
   print *, ' (distnew .gt. 20.0d0) .and. (dist .lt. 20.0d0) '
   print *, ' Distance to Critical Group must be either '
   print *, ' close in (drinking water pathway only) OR '
   print *, ' far away (groundwater pathway) '
   include 'stop.i'
 endif
 if( dist .lt. 20.0d0 ) then
```

```
        do k = 1, nnucl
          ik = indexperiso( names(k) )
          do j=1,ntim
            if( ciperyrallsafromsz(j,k) .lt. 0.0d0 ) then
              print *, ' ***>>> Error in DCAGW <<<*** '
              print *, ' ciperyrallsafromsz(j,k) .lt. 0.0d0 '
              print *, ' ciperyrallsafromsz(j,k) = ', ciperyrallsafromsz(j,k)
              print *, ' j = ', j
              print *, ' k = ', k
              print *, ' '
              include 'stop.i'
            endif
            remperyrpernuclgw(j,k)=dcf(ik)*ciperyrallsafromsz(j,k) /
     &       qm3peryrallsafromsz(j)
          enddo
        enddo
      else
        pump = valuesp( ipump )
cc        print *, ' dcagw: pump = ', pump, ' [gal/day] '
cc
cc 10^6 Gallon = 3785.41 Meter^3
cc 1 Year = 365.25 Day
cc 1 Meter^3 = 264.172 Gallon
cc
cc (Ci/m^3)   = (Ci/yr)*(Gal/day)*(day/yr)*(m^3/Gallon)
cc pumprate = (m^3/yr) = (Gal/day)*(365day/yr)*(m^3/264Gallon)
        pumprate =  ( pump * 365.25d0 / 264.172d0 )
cc        print *, ' dcagw: pumprate = ', pumprate, ' [m^3/yr]
        if( pumprate .lt. qm3peryrallsafromsz(ntim) ) then
          print *, ' ***>>> Error in DCAGW <<<*** '
          print *, ' pumprate .lt. qm3peryrallsafromsz '
          print *, ' pumprate   = ', pumprate
          print *, ' qm3peryrallsafromsz(ntim) = ',
     &             qm3peryrallsafromsz(ntim)
          print *, ' '
          include 'stop.i'
        endif
        do k = 1, nnucl
          ik = indexperiso( names(k) )
          do j=1,ntim
            if( ciperyrallsafromsz(j,k) .lt. 0.0d0 ) then
              print *, ' ***>>> Error in DCAGW <<<*** '
              print *, ' ciperyrallsafromsz(j,k) .lt. 0.0d0 '
              print *, ' ciperyrallsafromsz(j,k) = ', ciperyrallsafromsz(j,k)
              print *, ' j = ', j
              print *, ' k = ', k
              print *, ' '
              include 'stop.i'
            endif
            remperyrpernuclgw(j,k)=dcf(ik)*ciperyrallsafromsz(j,k) /
     &         pumprate
          enddo
        enddo
      endif

      return
      end
```

rem/yr = (rem/yr)/(Ci/m³) · (Ci/yr)/(m³/yr)

YW 8/16/2000

pumprate = (pump * 365.25d0 / 264.172d0) ⇒ gal/day (m³/gallon) = m³/yr

YW 8/16/2000

(rem/yr) = (rem/yr)/(Ci/m³) · (Ci/yr)/(m³/yr)

YW 8/16/2000

5/20/97 MSJ — Now that these subroutines have been re-written, they must be re-tested. This will be accomplished by Mike Epley, a student hired on for the summer.

5/21/97 MSJ — Unfortunately, he won't start until after the required date. A short test routine will be used to verify the operation of the dose modules. Since the modules use the same dcf's for pluvial and non pluvial, a write statement is being added to show where the dcf's have come from.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      program testdcags
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      parameter (maxtimes=5,maxnucs=100,numtimes=1,numnucs=43)
      character*6 names1(maxnucs)
      dimension tim(numtimes)
      dimension ciperm2gsatCP(maxtimes,maxnucs)
      dimension remperyrpernuclgs(maxtimes,maxnucs)
      open (unit=10,file='testdcags.o',status='unknown')
      write(6,*)'please input the mean annual temperature (F)'
      read(5,*) dMAT
      write(6,*)'please input the mean annual precipitation (in.)'
      read(5,*) dMAP
      data (names1(j),j=1,43)
     &          /'  U238',' Cm246',' Pu242','Am242m',' Pu238',
     &           '  U234',' Th230',' Ra226',' Pb210',' Cm243',
     &           ' Am243',' Pu239','  U235',' Pa231',' Ac227',
     &           ' Cm245',' Pu241',' Am241',' Np237','  U233',
     &           ' Th229',' Cm244',' Pu240','  U236','  U232',
     &           ' Sm151',' Cs137',' Cs135','  I129',' Sn126',
     &           'Sn121m','Ag108m',' Pd107','  Tc99','  Mo93',
     &           '  Nb94','  Zr93','  Sr90','  Se79','  Ni63',
     &           '  Ni59','  Cl36','   C14'/
      do i=1,numnucs
       ciperm2gsatCP(1,i)=1.d0
      end do
      write(10,*) 'file: testdcags.o'
      write(10,100)'dMAT= ',dMAT,' degrees F'
      write(10,100)'dMAP= ',dMAP,' inches   '
100   format(a6,f10.2,a10)
      tim(1)=1.d0
      call dcags( maxtimes, maxnucs,
     &        numtimes, tim, numnucs, names1,
     &        dMAT,dMAP,ciperm2gsatCP,
     &        remperyrpernuclgs  )
      write(10,*)'nucl  ','dose (rem/y)'
      do k=1,numnucs
       write(10,110)names1(k),remperyrpernuclgs(1,k)
110    format(a6,e12.4)
      end do
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      subroutine clearchar(num,name)
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      character*60 name
      name=' '
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function valuesp(i)
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      if (i.eq.1) then
       write(6,*) 'input distance cutoff for separating CGs'
       read(5,*) valuesp
       write(10,200) 'cutoff distance= ', valuesp,' km'
200    format(a17,f10.4,a3)
      else if (i.eq.2) then
       write(6,*) 'input distance to CG'
       read(5,*) valuesp
       write(10,210) 'distance to CG= ', valuesp,' km'
210    format(a16,f10.4,a3)
      end if
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function ispquery(name)
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      character*60 name
      if (name.eq.
     &'DistanceCutoffForDoseConversionDualityInDCAGS[km]') then
       ispquery=1
      else if (name.eq.
     &'DistanceToCriticalGroup[km][should be 5 or 30]') then
       ispquery=2
      end if
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function indexperiso( name )
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      character*6 name,names2(43)
      data (names2(j),j=1,43)
     &          /'  U238',' Cm246',' Pu242','Am242m',' Pu238',
     &           '  U234',' Th230',' Ra226',' Pb210',' Cm243',
     &           ' Am243',' Pu239',' U235',' Pa231',' Ac227',
     &           ' Cm245',' Pu241',' Am241',' Np237',' U233',
     &           ' Th229',' Cm244',' Pu240',' U236',' U232',
     &           ' Sm151',' Cs137',' Cs135',' I129',' Sn126',
     &           'Sn121m','Ag108m',' Pd107',' Tc99',' Mo93',
     &           '  Nb94',' Zr93',' Sr90',' Se79',' Ni63',
     &           '  Ni59',' Cl36',' C14'/
      do i=1,43
       if (name.eq.names2(i)) then
        indexperiso=i
       end if
      end do
      return
      end
c
c     This ends the tester program and subroutines. Cut here with a
c     sharp knife for insertion of the following module into TPA.
c
c=======================================================================
      subroutine dcags( mxntime, mxnnucl,
     &        ntim, tim, nnucl, names,
     &        dMAT,dMAP,ciperm2gsatCP,
     &        remperyrpernuclgs  )
c=======================================================================
c dose through groundsurface pathway due to laydown of contaminated ashblanket
c compliance point which may be locate 25, 30, 35 km from repository
c
c mxntime = input, integer, size to dimension arrays
c mxnnucl = input, integer, size to dimension arrays
c ntim = input, integer, maximum number of times used to dimension arrays
c tim(ntim)  = input, double precision, array of times
c nnucl = input, integer, maximum number of nuclides used to dimension arrays
c names(nnucl) = input, character*6, names of nuclides to be tracked
c ciperm2gsatCP(mxntime,mxnnucl) = input, double precision, time-dependent
c               concentration [ci/m^2] of nuclides at ground surface
c               due to volcanic ashplume.  A separate module accounts for
c               the evolution of nuclides after the ash is layed down.
c               Hence, the errosion and leaching of nuclides is already
c               accounted for, and this module only converts a given
c               concentration [ci/m2] to annual effective dose equivalent
c               [rem/yr].
c remperyrpernuclgs[mxntime,nnucl] = output, double precision, array of
c             annual EDE (effective dose equivalent) per nuclide
c
c       This subroutine was re-written on 4/17/97 by Mark Jarzemba
c       in order to have a duality of how doses are calculated.
c       For compliance points closer to the repository than the cutoff pt,
c       the dcf's are based on direct exposure and inhalation only.
c       For points at a distance of greater than the cutoff pt. from the
c       repository, the dcf's are based on direct exposure, inhalation,
c       and ingestion of contaminated animal products and crops.
c
c       This subroutine was re-re-written on 5/19/97 by Mark Jarzemba
c       in order to have another duality of how doses are calculated.
c       For points in time where the mean annual temperature (dMAT) and
c       mean annual precipitation (dMAP) are such that the climate is
```

```
c       re-classified as a semi-arid climate according to the Koeppen-
c       Geiger climate classification scheme.
c
c       dMAT in degrees fahrenheit
c       dMAP in inches
c
      implicit double precision (a-h,o-z)

      dimension tim(ntim)
      character*6 names(nnucl)
      character*60 name
      dimension ciperm2gsatCP(mxntime,mxnnucl)
      dimension remperyrpernuclgs(mxntime, mxnnucl)
      external ispquery
      external valuesp

      common / dcags1 / ikey
      common / dcags2 / dcf(43)

      if (ikey .ne. 73562 ) then
cc to avoid warning message, make trivial use of tim()
        ajunk = tim(1)

        call clearchar( 60, name )
        name = 'DistanceCutoffForDoseConversionDualityInDCAGS[km]'
        icutoffpt = ispquery( name )

        call clearchar( 60, name )
        name = 'DistanceToCriticalGroup[km][should be 5 or 30]'
        idist = ispquery( name )

        ikey = 73562

      endif

      cutoffpt=valuesp( icutoffpt )
      dist=valuesp( idist )
      dKGboundary=0.22d0*(dMAT-32.0d0)
      if (dist.ge.cutoffpt) then


c
c       These dcf's are for the Amargosa Desert farmer/rancher where
c       direct exposure, inhalation, and ingestion of animal products
c       and crops has been included
c
       if(dMAP.lt.dKGboundary) then

c
c       These DCF's are for an ADFR receptor group in today's biosphere.
c       As of 5/19/97, there are no differences between the pluvial and
c       current biosphere DCF's, other than a small fraction was added
c       to the pluvial dcf's in order to be able to test the routine.

        dcf( 1) =  0.66D+04
        dcf( 2) =  0.72D+06
        dcf( 3) =  0.95D+04
        dcf( 4) =  0.67D+06
        dcf( 5) =  0.88D+04
        dcf( 6) =  0.57D+04
        dcf( 7) =  0.10D+06
        dcf( 8) =  0.24D+06
        dcf( 9) =  0.12D+07
        dcf(10) =  0.49D+06
        dcf(11) =  0.70D+06
        dcf(12) =  0.10D+05
        dcf(13) =  0.19D+05
        dcf(14) =  0.21D+07
        dcf(15) =  0.27D+07
        dcf(16) =  0.72D+06
        dcf(17) =  0.24D+03
        dcf(18) =  0.70D+06
        dcf(19) =  0.11D+07
        dcf(20) =  0.58D+04
        dcf(21) =  0.73D+06
        dcf(22) =  0.39D+06
        dcf(23) =  0.10D+05
        dcf(24) =  0.54D+04
        dcf(25) =  0.18D+05
        dcf(26) =  0.11D+03
        dcf(27) =  0.12D+06
        dcf(28) =  0.10D+05
        dcf(29) =  0.33D+06
        dcf(30) =  0.43D+05
        dcf(31) =  0.46D+04
        dcf(32) =  0.00D+00
        dcf(33) =  0.89D+02
        dcf(34) =  0.46D+04
        dcf(35) =  0.11D+04
        dcf(36) =  0.13D+06
        dcf(37) =  0.31D+03
        dcf(38) =  0.73D+05
        dcf(39) =  0.52D+04
        dcf(40) =  0.42D+03
        dcf(41) =  0.15D+03
        dcf(42) =  0.69D+05
        dcf(43) =  0.14D+01
      else if(dMAP.ge.dKGboundary) then
c
c       These DCF's are for an ADFR receptor group in a  pluvial biosphere.
c       As of 5/19/97, there are no differences between the pluvial and
c       current biosphere DCF's, other than a small fraction was added
c       to the pluvial dcf's in order to be able to test the routine.
c
        dcf( 1) =  0.6601D+04
        dcf( 2) =  0.7201D+06
        dcf( 3) =  0.9501D+04
        dcf( 4) =  0.6701D+06
        dcf( 5) =  0.8801D+04
        dcf( 6) =  0.5701D+04
        dcf( 7) =  0.1001D+06
        dcf( 8) =  0.2401D+06
        dcf( 9) =  0.1201D+07
        dcf(10) =  0.4901D+06
        dcf(11) =  0.7001D+06
        dcf(12) =  0.1001D+05
        dcf(13) =  0.1901D+05
        dcf(14) =  0.2101D+07
        dcf(15) =  0.2701D+07
        dcf(16) =  0.7201D+06
        dcf(17) =  0.2401D+03
        dcf(18) =  0.7001D+06
        dcf(19) =  0.1101D+07
        dcf(20) =  0.5801D+04
        dcf(21) =  0.7301D+06
        dcf(22) =  0.3901D+06
        dcf(23) =  0.1001D+05
        dcf(24) =  0.5401D+04
        dcf(25) =  0.1801D+05
        dcf(26) =  0.1101D+03
        dcf(27) =  0.1201D+06
        dcf(28) =  0.1001D+05
        dcf(29) =  0.3301D+06
        dcf(30) =  0.4301D+05
        dcf(31) =  0.4601D+04
        dcf(32) =  0.00D+00
        dcf(33) =  0.8901D+02
        dcf(34) =  0.4601D+04
        dcf(35) =  0.1101D+04
        dcf(36) =  0.1301D+06
        dcf(37) =  0.3101D+03
        dcf(38) =  0.7301D+05
        dcf(39) =  0.5201D+04
        dcf(40) =  0.4201D+03
        dcf(41) =  0.1501D+03
        dcf(42) =  0.6901D+05
        dcf(43) =  0.1401D+01
       end if

      else


c
c       These dcf's are for the closer in residential water user
c       and include only direct exposure and inhalation.
c
       if(dMAP.lt.dKGboundary) then

c
c       These DCF's are for an CIRWU receptor group in today's biosphere.
c       As of 5/19/97, there are no differences between the pluvial and
c       current biosphere DCF's, other than a small fraction was added
c       to the pluvial dcf's in order to be able to test the routine.
c
```

```
dcf(1) = 0.24D+03
dcf(2) = 0.79D+03
dcf(3) = 0.52D+03
dcf(4) = 0.93D+03
dcf(5) = 0.48D+03
dcf(6) = 0.28D+03
dcf(7) = 0.48D+03
dcf(8) = 0.57D+03
dcf(9) = 0.24D+03
dcf(10) = 0.11D+05
dcf(11) = 0.53D+04
dcf(12) = 0.51D+03
dcf(13) = 0.13D+05
dcf(14) = 0.49D+04
dcf(15) = 0.21D+04
dcf(16) = 0.83D+04
dcf(17) = 0.80D+01
dcf(18) = 0.53D+04
dcf(19) = 0.36D+04
dcf(20) = 0.27D+03
dcf(21) = 0.10D+05
dcf(22) = 0.47D+03
dcf(23) = 0.55D+03
dcf(24) = 0.26D+03
dcf(25) = 0.11D+04
dcf(26) = 0.49D+00
dcf(27) = 0.48D+05
dcf(28) = 0.30D+01
dcf(29) = 0.22D+04
dcf(30) = 0.48D+04
dcf(31) = 0.43D+03
dcf(32) = 0.00D+00
dcf(33) = 0.22D-01
dcf(34) = 0.69D+01
dcf(35) = 0.46D+03
dcf(36) = 0.13D+06
dcf(37) = 0.13D+00
dcf(38) = 0.24D+02
dcf(39) = 0.18D+01
dcf(40) = 0.35D-02
dcf(41) = 0.14D-02
dcf(42) = 0.59D+02
dcf(43) = 0.14D+01

else if(dMAP.ge.dKGboundary) then

c
c  These DCF's are for an CIRWU receptor group in a pluvial biosphere.
c  As of 5/19/97, there are no differences between the pluvial and
c  current biosphere DCF's, other than a small fraction was added
c  to the pluvial dcf's in order to be able to test the routine.
c
c

 dcf(1) = 0.2401D+03
 dcf(2) = 0.7901D+03
 dcf(3) = 0.5201D+03
 dcf(4) = 0.9301D+03
 dcf(5) = 0.4801D+03
 dcf(6) = 0.2801D+03
 dcf(7) = 0.4801D+03
 dcf(8) = 0.5701D+03
 dcf(9) = 0.2401D+03
 dcf(10) = 0.1101D+05
 dcf(11) = 0.5301D+04
 dcf(12) = 0.5101D+03
 dcf(13) = 0.1301D+05
 dcf(14) = 0.4901D+04
 dcf(15) = 0.2101D+04
 dcf(16) = 0.8301D+04
 dcf(17) = 0.8001D+01
 dcf(18) = 0.5301D+04
 dcf(19) = 0.3601D+04
 dcf(20) = 0.2701D+03
 dcf(21) = 0.1001D+05
 dcf(22) = 0.4701D+03
 dcf(23) = 0.5501D+03
 dcf(24) = 0.2601D+03
 dcf(25) = 0.1101D+04
 dcf(26) = 0.4901D+00
 dcf(27) = 0.4801D+05
 dcf(28) = 0.3001D+01
 dcf(29) = 0.2201D+04
 dcf(30) = 0.4801D+04
 dcf(31) = 0.4301D+03
 dcf(32) = 0.00D+00
 dcf(33) = 0.2201D-01
 dcf(34) = 0.6901D+01
 dcf(35) = 0.4601D+03
 dcf(36) = 0.1301D+06
 dcf(37) = 0.1301D+00
 dcf(38) = 0.2401D+02
 dcf(39) = 0.1801D+01
 dcf(40) = 0.3501D-02
 dcf(41) = 0.1401D-02
 dcf(42) = 0.5901D+02
 dcf(43) = 0.1401D+01
 end if
end if

do k = 1, nnucl
  ik = indexperiso( names(k) )
  do j=1,ntim
    remperyrpernuclgs(j,k)=dcf(ik)*ciperm2gsatCP(j,k)
  enddo
enddo

return
end
```

5/22/97
MSJ

file: testdcags.o
dMAT= 61.00 degrees F
dMAP= 5.90 inches
cutoff distance= 20.0000 km
distance to CG= 30.0000 km

nucl	dose (rem/y)
U238	0.6600E+04
Cm246	0.7200E+06
Pu242	0.9500E+04
Am242m	0.6700E+06
Pu238	0.8800E+04
U234	0.5700E+04
Th230	0.1000E+06
Ra226	0.2400E+06
Pb210	0.1200E+07
Cm243	0.4900E+06
Am243	0.7000E+06
Pu239	0.1000E+05
U235	0.1900E+05
Pa231	0.2100E+07
Ac227	0.2700E+07
Cm245	0.7200E+06
Pu241	0.2400E+03
Am241	0.7000E+06
Np237	0.1100E+07
U233	0.5800E+04
Th229	0.7300E+06
Cm244	0.3900E+06
Pu240	0.1000E+05
U236	0.5400E+04
U232	0.1800E+05
Sm151	0.1100E+03
Cs137	0.1200E+06
Cs135	0.1000E+05
I129	0.3300E+06
Sn126	0.4300E+05
Sn121m	0.4600E+04
Ag108m	0.0000E+00
Pd107	0.8900E+02
Tc99	0.4600E+04
Mo93	0.1100E+04
Nb94	0.1300E+06
Zr93	0.3100E+03
Sr90	0.7300E+05
Se79	0.5200E+04
Ni63	0.4200E+03
Ni59	0.1500E+03
Cl36	0.6900E+05
C14	0.1400E+01

note: ADFR
- today's
biosphere

5/22/97 - Run to verify that "today's biosphere" DCF's are
MSJ being utilized properly; MAT = 61°F MAP = 5.9"

file: testdcags.o
dMAT= 61.00 degrees F
dMAP= 5.90 inches
cutoff distance= 20.0000 km
distance to CG= 5.0000 km

nucl	dose (rem/y)
U238	0.2400E+03
Cm246	0.7900E+03
Pu242	0.5200E+03
Am242m	0.9300E+03
Pu238	0.4800E+03
U234	0.2800E+03
Th230	0.4800E+03
Ra226	0.5700E+03
Pb210	0.2400E+03
Cm243	0.1100E+05
Am243	0.5300E+04
Pu239	0.5100E+03
U235	0.1300E+05
Pa231	0.4900E+04
Ac227	0.2100E+04
Cm245	0.8300E+04
Pu241	0.8000E+01
Am241	0.5300E+04
Np237	0.3600E+04
U233	0.2700E+03
Th229	0.1000E+05
Cm244	0.4700E+03
Pu240	0.5500E+03
U236	0.2600E+03
U232	0.1100E+04
Sm151	0.4900E+00
Cs137	0.4800E+05
Cs135	0.3000E+01
I129	0.2200E+04
Sn126	0.4800E+04
Sn121m	0.4300E+03
Ag108m	0.0000E+00
Pd107	0.2200E-01
Tc99	0.6900E+01
Mo93	0.4600E+03
Nb94	0.1300E+06
Zr93	0.1300E+00
Sr90	0.2400E+02
Se79	0.1800E+01
Ni63	0.3500E-02
Ni59	0.1400E-02
Cl36	0.5900E+02
C14	0.1400E+01

Since the #'s
here do not
end in "01";
this is verification
that the numbers
are correct.

Note: for CIWU
CG

- Run to verify that pluvial DCF's are being utilized
MAT = 61°F ; MAP = 10"

file: testdcags.o
dMAT= 61.00 degrees F
dMAP= 10.00 inches
cutoff distance= 20.0000 km
distance to CG= 5.0000 km

nucl	dose (rem/y)
U238	0.2401E+03
Cm246	0.7901E+03
Pu242	0.5201E+03
Am242m	0.9301E+03
Pu238	0.4801E+03
U234	0.2801E+03
Th230	0.4801E+03
Ra226	0.5701E+03
Pb210	0.2401E+03
Cm243	0.1101E+05
Am243	0.5301E+04
Pu239	0.5101E+03
U235	0.1301E+05
Pa231	0.4901E+04
Ac227	0.2101E+04
Cm245	0.8301E+04
Pu241	0.8001E+01
Am241	0.5301E+04
Np237	0.3601E+04
U233	0.2701E+03
Th229	0.1001E+05
Cm244	0.4701E+03
Pu240	0.5501E+03
U236	0.2601E+03
U232	0.1101E+04
Sm151	0.4901E+00
Cs137	0.4801E+05
Cs135	0.3001E+01
I129	0.2201E+04
Sn126	0.4801E+04
Sn121m	0.4301E+03
Ag108m	0.0000E+00
Pd107	0.2201E-01
Tc99	0.6901E+01
Mo93	0.4601E+03
Nb94	0.1301E+06
Zr93	0.1301E+00
Sr90	0.2401E+02
Se79	0.1801E+01
Ni63	0.3501E-02
Ni59	0.1401E-02
Cl36	0.5901E+02
C14	0.1401E+01

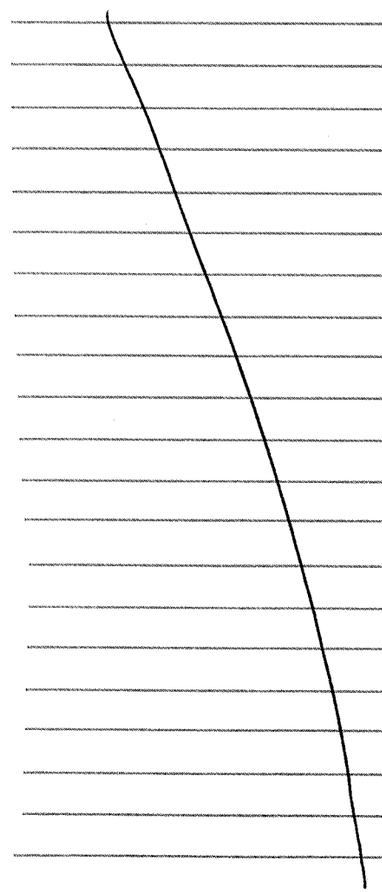
The fact that these no.'s end in an "01" is verification that the pluvial DCF's are being used.

note: CIWU

file: testdcags.o
dMAT= 61.00 degrees F
dMAP= 10.00 inches
cutoff distance= 20.0000 km
distance to CG= 30.0000 km

nucl	dose (rem/y)
U238	0.6601E+04
Cm246	0.7201E+06
Pu242	0.9501E+04
Am242m	0.6701E+06
Pu238	0.8801E+04
U234	0.5701E+04
Th230	0.1001E+06
Ra226	0.2401E+06
Pb210	0.1201E+07
Cm243	0.4901E+06
Am243	0.7001E+06
Pu239	0.1001E+05
U235	0.1901E+05
Pa231	0.2101E+07
Ac227	0.2701E+07
Cm245	0.7201E+06
Pu241	0.2401E+03
Am241	0.7001E+06
Np237	0.1101E+07
U233	0.5801E+04
Th229	0.7301E+06
Cm244	0.3901E+06
Pu240	0.1001E+05
U236	0.5401E+04
U232	0.1801E+05
Sm151	0.1101E+03
Cs137	0.1201E+06
Cs135	0.1001E+05
I129	0.3301E+06
Sn126	0.4301E+05
Sn121m	0.4601E+04
Ag108m	0.0000E+00
Pd107	0.8901E+02
Tc99	0.4601E+04
Mo93	0.1101E+04
Nb94	0.1301E+06
Zr93	0.3101E+03
Sr90	0.7301E+05
Se79	0.5201E+04
Ni63	0.4201E+03
Ni59	0.1501E+03
Cl36	0.6901E+05
C14	0.1401E+01

note: ADFR



```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      program testdcagw
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      parameter (maxtimes=5,maxnucs=100,numtimes=1,numnucs=43)
      character*6 names1(maxnucs)
      dimension tim(numtimes)
      dimension qm3peryrallsafromsz(maxtimes,maxnucs)
      dimension remperyrpernuclgw(maxtimes,maxnucs)
      dimension ciperyrallsafromsz(maxtimes,maxnucs)
      open (unit=10,file='testdcagw.o',status='unknown')
      write(6,*)'please input the mean annual temperature (F)'
      read(5,*) dMAT
      write(6,*)'please input the mean annual precipitation (in.)'
      read(5,*) dMAP
      data (names1(j),j=1,43)
     &          /'  U238',' Cm246',' Pu242','Am242m',' Pu238',
     &           '  U234',' Th230',' Ra226',' Pb210',' Cm243',
     &           ' Am243',' Pu239','  U235',' Pa231',' Ac227',
     &           ' Cm245',' Pu241',' Am241',' Np237',  ' U233',
     &           ' Th229',' Cm244',' Pu240','  U236','  U232',
     &           ' Sm151',' Cs137',' Cs135','  I129',' Sn126',
     &          'Sn121m','Ag108m',' Pd107','  Tc99',  ' Mo93',
     &           '  Nb94',  ' Zr93',  ' Sr90',  ' Se79',  ' Ni63',
     &           '  Ni59',  ' Cl36',  '  C14'/
      do i=1,numnucs
       qm3peryrallsafromsz(1,i)=1.d0
       ciperyrallsafromsz(1,i)=1.d0
      end do
      write(10,*) 'file: testdcagw.o'
      write(10,100)'dMAT= ',dMAT,' degrees F'
      write(10,100)'dMAP= ',dMAP,' inches   '
100   format(a6,f10.2,a10)
      tim(1)=1.d0
      call dcagw( maxtimes, maxnucs,
     &        numtimes, tim, numnucs, names1,
     &        dMAT,dMAP,qm3peryrallsafromsz,
     &        ciperyrallsafromsz,
     &        remperyrpernuclgw )
      write(10,*)'nucl  ','dose (rem/y)'
      do k=1,numnucs
       write(10,110)names1(k),remperyrpernuclgw(1,k)
110    format(a6,e12.4)
      end do
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      subroutine clearchar(num,name)
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      character*60 name
      name=' '
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function valuesp(i)
c                                                                       c
```

5/23/97
MSJ


```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      if (i.eq.1) then
       write(6,*) 'input distance cutoff for separating CGs'
       read(5,*) valuesp
       write(10,200) 'cutoff distance= ', valuesp,' km'
200    format(a17,f10.4,a3)
      else if (i.eq.2) then
       write(6,*) 'input distance to CG'
       read(5,*) valuesp
       write(10,210) 'distance to CG= ', valuesp,' km'
210    format(a16,f10.4,a3)
      else if (i.eq.3) then
        write(6,*) 'input pumping rate'
        read(5,*) valuesp
        write(10,220) 'pumping rate= ',valuesp,' gpd'
220     format(a14,f10.4,a4)
      end if
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function ispquery(name)
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      character*60 name
      if (name.eq.
     &'DistanceCutoffForDoseConversionDualityInDCAGS[km]') then
       ispquery=1
      else if (name.eq.
     &'DistanceToCriticalGroup[km][should be 5 or 30]') then
       ispquery=2
      else if (name.eq.
     &'WellPumpingRateAtCriticalGroup[gal/day]') then
       ispquery=3
      end if
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function indexperiso( name )
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      character*6 name,names2(43)
      data (names2(j),j=1,43)
     &          /'  U238',' Cm246',' Pu242','Am242m',' Pu238',
     &           '  U234',' Th230',' Ra226',' Pb210',' Cm243',
     &           ' Am243',' Pu239',' U235',' Pa231',' Ac227',
     &           ' Cm245',' Pu241',' Am241',' Np237',' U233',
     &           ' Th229',' Cm244',' Pu240',' U236',' U232',
     &           ' Sm151',' Cs137',' Cs135',' I129',' Sn126',
     &          'Sn121m','Ag108m',' Pd107',' Tc99',' Mo93',
     &           '  Nb94',' Zr93',' Sr90',' Se79',' Ni63',
     &           '  Ni59',' Cl36',' C14'/
      do i=1,43
       if (name.eq.names2(i)) then
        indexperiso=i
       end if
```

```
      end do
      return
      end
c
c     This ends the tester program and subroutines. Cut here with a
c     sharp knife for insertion of the following module into TPA.
c
c=======================================================================
      subroutine dcagw( mxntime, mxnnucl,
     &        ntim, tim, nnucl, names,
     &        dMAT,dMAP,qm3peryrallsafromsz,
     &        ciperyrallsafromsz,
     &        remperyrpernuclgw )
c=======================================================================
c calculates dose through groundwater pathway by extraction at well.
c calculates saturated zone flow from below repository to compliance
c point which may be located 5,25, 30, 35 km from repository
c
c if CP is less than 20 km, then use qm3peryrallsafromsz to calculate
c concentration of nuclides in GW.
c if CP is greater than 20 km, then use well pumping rate to calculate
c concentration of nuclides in GW, assuming all nuclides extracted
c from system by pumping
c
c mxntime = input, integer, size to dimension arrays
c mxnnucl = input, integer, size to dimension arrays
c ntim = input, integer, maximum number of times used to dimension arrays
c tim(ntim)  = input, double precision, array of times
c nnucl = input, integer, maximum number of nuclides used to dimension arrays
c names(nnucl) = input, character*6, names of nuclides to be tracked
c qm3peryrallsafromsz(mxntime) =  input, double precision, groundwater flow
c                   rate [m^3/yr] from all subareas from the saturated zone
c                   to the compliance point
c ciperyrallsafromsz(mxntime,mxnnucl) = input, double precision, nuclide
c                   release rate [Ci/yr] from all subareas from the
c                   saturated zone at the compliance point
c remperyrpernuclgw[mxntime,nnucl] = output, double precision, array of
c             annual EDE (effective dose equivalent) per nuclide
c
      implicit double precision (a-h,o-z)

      dimension tim(ntim)
      character*6 names(nnucl)
      dimension qm3peryrallsafromsz(mxntime)
      dimension ciperyrallsafromsz(mxntime,mxnnucl)
      dimension remperyrpernuclgw(mxntime, mxnnucl)

      character*60 name
      common / dcagw1 / ikey
      common / dcagw2 / dcf(43)
      common / dcagw3 / idist
      common / dcagw4 / dist

cc      common / dcagw5  / iownpeakdosev
cc      common / dcagw6  / iownpeakdoset
cc      common / dcagw7  / ipeakdosev
cc      common / dcagw8  / ipeakdoset
cc      common / dcagw9  / iownpeakdosenv(43)
cc      common / dcagw10 / iownpeakdosent(43)
cc      common / dcagw11 / ipeakdosenv(43)
cc      common / dcagw12 / ipeakdosent(43)
      common / dcagw13 / ipump

      external valuesp

cc      print *, ' dcagw: entered into dcagw '

      if( (nnucl .le. 0) .or. (nnucl .gt. 43) ) then
        print *, ' ***>>> Error in DCAGW <<<*** '
        print *, ' (nnucl .le. 0)  .or. (nnucl .gt. 43)'
        print *, ' nnucl = ', nnucl
        include 'stop.i'
      endif
      if( ntim .gt. mxntime ) then
        print *, ' ***>>> Error in DCAGW <<<*** '
        print *, ' ntim .gt. mxntime '
        print *, ' mxntime = ', mxntime
        print *, ' ntim    = ', ntim
        include 'stop.i'
      endif
      if( ntim .le. 0 ) then
        print *, ' ***>>> Error in DCAGW <<<*** '
        print *, ' ntim .le. 0 '
        print *, ' ntim = ', ntim
        include 'stop.i'
      endif

      if (ikey .ne. 39231) then
cc to avoid warning message, make trivial use of tim()
        ajunk = tim(1)

        call clearchar( 60, name )
        name = 'DistanceToCriticalGroup[km][should be 5 or 30]'
        idist = ispquery( name )
        dist  = valuesp( idist )

        call clearchar( 60, name )
        name = 'WellPumpingRateAtCriticalGroup[gal/day]'
        ipump = ispquery( name )

        call clearchar( 60, name )
        name = 'DistanceCutoffForDoseConversionDualityInDCAGS[km]'
        icutoffpt = ispquery( name )
        cutoffpt = valuesp( icutoffpt)

cc        call clearchar( 60, name )
cc        name = 'Peak Annual Dose [rem/yr]'
cc        iownpeakdosev = iaddconsmv( name )
cc        ipeakdosev = imvquery( name )

cc        call clearchar( 60, name )
cc        name = 'Time of Peak Annual Dose [yr]'
cc        iownpeakdoset = iaddconsmv( name )
cc        ipeakdoset = imvquery( name )

cc        do ii = 1, nnucl
cc         call clearchar( 60, name )
cc         name = 'Peak Annual Dose from ' // names(ii) // ' [rem/yr]'
cc         iownpeakdosenv(ii) = iaddconsmv( name )
cc         ipeakdosenv(ii) = imvquery( name )
cc         call clearchar( 60, name )
cc         name = 'Time of Peak Annual Dose from ' // names(ii) // ' [yr]'
cc         iownpeakdosent(ii) = iaddconsmv( name )
cc         ipeakdosent(ii) = imvquery( name )
cc        enddo
```

```
      dKGcutoffpt=0.22d0*(dMAT-32.d0)

      if( dist .lt. cutoffpt ) then
cc use drinking water dose conversion factors if the distance to the
cc critical group is less than 20 km away from the source.
cc
cc data from EPA, 1988, Limiting Values of Radionuclide INtake and Air
cc Concentration and Dose Conversion Factors for Inhalation,
cc Submersion and Ingestion. EPA-520/1-88-020, Washington, DC
cc Environmental Protection Agency
cc
cc which is same data as DOE, 1988, Internal Dose Conversion Factors
cc for Calculation of Dose to the Public. DOE/EH-0071, Washington, DC,
cc U.S. Department of Energy
cc
c
c       The following modification has been added by Mark Jarzemba
c       on 5/20/97 to have a duality in how doses are calculated.
c       If the dMAT and dMAP are such that the climate is classified
c       as semi-arid, use pluvial DCF's.
c
c       As of 5/20/97, there is no difference between the pluvial and
c       non-pluvial DCF's except for a small fraction added to the
c       pluvial DCF's for testing.
c
        if (dMAP.lt.dKGcutoffpt) then

c
c       These DCF's are for the CIRWU in today's biosphere
c

        dcf( 1) = 1.86d+05
        dcf( 2) = 2.70d+06
        dcf( 3) = 2.45d+06
        dcf( 4) = 2.57d+06
        dcf( 5) = 2.34d+06
        dcf( 6) = 2.07d+05
        dcf( 7) = 4.00d+05
        dcf( 8) = 9.67d+05
        dcf( 9) = 3.91d+06
        dcf(10) = 1.83d+06
        dcf(11) = 2.64d+06
        dcf(12) = 2.58d+06
        dcf(13) = 1.94d+05
        dcf(14) = 7.72d+06
        dcf(15) = 1.03d+07
        dcf(16) = 2.73d+06
        dcf(17) = 5.00d+04
        dcf(18) = 2.66d+06
        dcf(19) = 3.24d+06
        dcf(20) = 2.11d+05
        dcf(21) = 2.58d+06
        dcf(22) = 1.47d+06
        dcf(23) = 2.58d+06
        dcf(24) = 1.96d+05
        dcf(25) = 9.56d+05
        dcf(26) = 2.84d+02
        dcf(27) = 3.65d+04
        dcf(28) = 5.16d+03
        dcf(29) = 2.01d+05
        dcf(30) = 1.42d+04
        dcf(31) = 1.13d+03
        dcf(32) = 5.56d+03
        dcf(33) = 1.09d+02
        dcf(34) = 1.07d+03
        dcf(35) = 9.83d+02
        dcf(36) = 5.21d+03
        dcf(37) = 1.21d+03
        dcf(38) = 1.04d+05
        dcf(39) = 6.35d+03
        dcf(40) = 4.21d+02
        dcf(41) = 1.53d+02
        dcf(42) = 2.21d+03
        dcf(43) = 1.52d+03
       else if (dMAP.ge.dKGcutoffpt) then
c
c       These DCF's are for the CIRWU in a pluvial biosphere
c

        dcf( 1) = 1.861d+05
        dcf( 2) = 2.701d+06
        dcf( 3) = 2.451d+06
        dcf( 4) = 2.571d+06
        dcf( 5) = 2.341d+06
        dcf( 6) = 2.071d+05
        dcf( 7) = 4.001d+05
        dcf( 8) = 9.671d+05
        dcf( 9) = 3.911d+06
        dcf(10) = 1.831d+06
        dcf(11) = 2.641d+06
        dcf(12) = 2.581d+06
        dcf(13) = 1.941d+05
        dcf(14) = 7.721d+06
        dcf(15) = 1.031d+07
        dcf(16) = 2.731d+06
        dcf(17) = 5.001d+04
        dcf(18) = 2.661d+06
        dcf(19) = 3.241d+06
        dcf(20) = 2.111d+05
        dcf(21) = 2.581d+06
        dcf(22) = 1.471d+06
        dcf(23) = 2.581d+06
        dcf(24) = 1.961d+05
        dcf(25) = 9.561d+05
        dcf(26) = 2.841d+02
        dcf(27) = 3.651d+04
        dcf(28) = 5.161d+03
        dcf(29) = 2.011d+05
        dcf(30) = 1.421d+04
        dcf(31) = 1.131d+03
        dcf(32) = 5.561d+03
        dcf(33) = 1.091d+02
        dcf(34) = 1.071d+03
        dcf(35) = 9.831d+02
        dcf(36) = 5.211d+03
        dcf(37) = 1.211d+03
        dcf(38) = 1.041d+05
        dcf(39) = 6.351d+03
        dcf(40) = 4.211d+02
        dcf(41) = 1.531d+02
        dcf(42) = 2.211d+03
        dcf(43) = 1.521d+03
       end if
      else
cc
cc use mean groundwater pathway dose conversion factors
cc if the distance from source to critical group is more than
cc 20 km away. The DCFs are in units of [(rem/yr)/(pCi/liter)]
cc to convert to units of [(rem/yr)/(Ci/m^3)],
```

```
cc multiply by 10^9
cc
cc data from P.A. LaPlante, S.J. Maheras, and M.S. Jarzemba
cc Initial Analysis of Selected Site-Specific Dose Assessment
cc Parameters and Exposure Pathways Applicable to a Groundwater
cc Release Scenario at Yucca Mountain, CNWRA 95-018,
cc September, 1995, San Antonio, TX, Center for Nuclear Waste
cc Regulatory Analyses
cc
        if (dMAP.lt.dKGcutoffpt) then
c
c       These DCF's are for the ADFR in today's biosphere
c
          dcf( 1) =  0.72d+05
          dcf( 2) =  0.81d+07
          dcf( 3) =  0.10d+06
          dcf( 4) =  0.76d+07
          dcf( 5) =  0.95d+05
          dcf( 6) =  0.60d+05
          dcf( 7) =  0.12d+07
          dcf( 8) =  0.28d+07
          dcf( 9) =  0.13d+08
          dcf(10) =  0.54d+07
          dcf(11) =  0.79d+07
          dcf(12) =  0.11d+06
          dcf(13) =  0.84d+05
          dcf(14) =  0.23d+08
          dcf(15) =  0.31d+08
          dcf(16) =  0.81d+07
          dcf(17) =  0.32d+04
          dcf(18) =  0.79d+07
          dcf(19) =  0.13d+08
          dcf(20) =  0.61d+05
          dcf(21) =  0.81d+07
          dcf(22) =  0.43d+07
          dcf(23) =  0.11d+06
          dcf(24) =  0.57d+05
          dcf(25) =  0.24d+06
          dcf(26) =  0.12d+04
          dcf(27) =  0.76d+06
          dcf(28) =  0.10d+06
          dcf(29) =  0.31d+07
          dcf(30) =  0.63d+06
          dcf(31) =  0.43d+05
          dcf(32) =  0.00d+00
          dcf(33) =  0.81d+03
          dcf(34) =  0.84d+04
          dcf(35) =  0.44d+04
          dcf(36) =  0.20d+06
          dcf(37) =  0.35d+04
          dcf(38) =  0.61d+06
          dcf(39) =  0.53d+05
          dcf(40) =  0.38d+04
          dcf(41) =  0.14d+04
          dcf(42) =  0.87d+05
          dcf(43) =  0.19d+05
        else if (dMAP.ge.dKGcutoffpt) then
c
c       These DCF's are for the ADFR in a pluvial biosphere
c
          dcf( 1) =  0.7201d+05
          dcf( 2) =  0.8101d+07
          dcf( 3) =  0.1001d+06
```



```
          dcf( 4) =  0.7601d+07
          dcf( 5) =  0.9501d+05
          dcf( 6) =  0.6001d+05
          dcf( 7) =  0.1201d+07
          dcf( 8) =  0.2801d+07
          dcf( 9) =  0.1301d+08
          dcf(10) =  0.5401d+07
          dcf(11) =  0.7901d+07
          dcf(12) =  0.1101d+06
          dcf(13) =  0.8401d+05
          dcf(14) =  0.2301d+08
          dcf(15) =  0.3101d+08
          dcf(16) =  0.8101d+07
          dcf(17) =  0.3201d+04
          dcf(18) =  0.7901d+07
          dcf(19) =  0.1301d+08
          dcf(20) =  0.6101d+05
          dcf(21) =  0.8101d+07
          dcf(22) =  0.4301d+07
          dcf(23) =  0.1101d+06
          dcf(24) =  0.5701d+05
          dcf(25) =  0.2401d+06
          dcf(26) =  0.1201d+04
          dcf(27) =  0.7601d+06
          dcf(28) =  0.1001d+06
          dcf(29) =  0.3101d+07
          dcf(30) =  0.6301d+06
          dcf(31) =  0.4301d+05
          dcf(32) =  0.00d+00
          dcf(33) =  0.8101d+03
          dcf(34) =  0.8401d+04
          dcf(35) =  0.4401d+04
          dcf(36) =  0.2001d+06
          dcf(37) =  0.3501d+04
          dcf(38) =  0.6101d+06
          dcf(39) =  0.5301d+05
          dcf(40) =  0.3801d+04
          dcf(41) =  0.1401d+04
          dcf(42) =  0.8701d+05
          dcf(43) =  0.1901d+05
        end if
      endif

      ikey=39231

    endif
    if( dist .lt. cutoffpt ) then
      do k = 1, nnucl
        ik = indexperiso( names(k) )
        do j=1,ntim
          if( ciperyrallsafromsz(j,k) .lt. 0.0d0 ) then
            print *, ' ***>>> Error in DCAGW <<<*** '
            print *, ' ciperyrallsafromsz(j,k) .lt. 0.0d0 '
            print *, ' ciperyrallsafromsz(j,k) = ', ciperyrallsafromsz(j,k)
            print *, ' j = ', j
            print *, ' k = ', k
            print *, ' '
            include 'stop.i'
          endif
          remperyrpernuclgw(j,k)=dcf(ik)*ciperyrallsafromsz(j,k) /
     &       qm3peryrallsafromsz(j)
        enddo
      enddo
```

```
      else
        pump = valuesp( ipump )
cc        print *, ' dcagw: pump = ', pump, ' [gal/day] '
cc
cc 10^6 Gallon = 3785.41 Meter^3
cc 1 Year = 365.25 Day
cc 1 Meter^3 = 264.172 Gallon
cc
cc (Ci/m^3)   = (Ci/yr)*(Gal/day)*(day/yr)*(m^3/Gallon)
cc pumprate = (m^3/yr) = (Gal/day)*(365day/yr)*(m^3/264Gallon)
        pumprate =  ( pump * 365.25d0 / 264.172d0 )
cc        print *, ' dcagw: pumprate = ', pumprate, ' [m^3/yr] '
        if( pumprate .lt. qm3peryrallsafromsz(ntim) ) then
          print *, ' ***>>> Error in DCAGW <<<*** '
          print *, ' pumprate .lt. qm3peryrallsafromsz '
          print *, ' pumprate   = ', pumprate
          print *, ' qm3peryrallsafromsz(ntim) = ',
     &             qm3peryrallsafromsz(ntim)
          print *, ' '
          include 'stop.i'
        endif
        do k = 1, nnucl
          ik = indexperiso( names(k) )
          do j=1,ntim
            if( ciperyrallsafromsz(j,k) .lt. 0.0d0 ) then
              print *, ' ***>>> Error in DCAGW <<<*** '
              print *, ' ciperyrallsafromsz(j,k) .lt. 0.0d0 '
              print *, ' ciperyrallsafromsz(j,k) = ', ciperyrallsafromsz(j,k)
              print *, ' j = ', j
              print *, ' k = ', k
              print *, ' '
              include 'stop.i'
            endif
            remperyrpernuclgw(j,k)=dcf(ik)*ciperyrallsafromsz(j,k) /
&              pumprate
          enddo
        enddo
      endif

      return
      end
```


Amargosa Desert Farmer/Rancher

today's biosphere

```
file: testdcagw.o
dMAT=       61.00 degrees F
dMAP=        5.90 inches
distance to CG=     30.0000 km
cutoff distance=     20.0000 km
pumping rate=       .7233 gpd
nucl  dose (rem/y)
  U238  0.7200E+05
 Cm246  0.8100E+07
 Pu242  0.1000E+06
Am242m  0.7600E+07
 Pu238  0.9500E+05
  U234  0.6000E+05
 Th230  0.1200E+07
 Ra226  0.2800E+07
 Pb210  0.1300E+08
 Cm243  0.5400E+07
 Am243  0.7900E+07
 Pu239  0.1100E+06
  U235  0.8400E+05
 Pa231  0.2300E+08
 Ac227  0.3100E+08
 Cm245  0.8100E+07
 Pu241  0.3200E+04
 Am241  0.7900E+07
 Np237  0.1300E+08
  U233  0.6100E+05
 Th229  0.8100E+07
 Cm244  0.4300E+07
 Pu240  0.1100E+06
  U236  0.5700E+05
  U232  0.2400E+06
 Sm151  0.1200E+04
 Cs137  0.7600E+06
 Cs135  0.1000E+06
  I129  0.3100E+07
 Sn126  0.6300E+06
Sn121m  0.4300E+05
Ag108m  0.0000E+00
 Pd107  0.8100E+03
  Tc99  0.8400E+04
  Mo93  0.4400E+04
  Nb94  0.2000E+06
  Zr93  0.3500E+04
  Sr90  0.6100E+06
  Se79  0.5300E+05
  Ni63  0.3800E+04
  Ni59  0.1400E+04
  Cl36  0.8700E+05
   C14  0.1900E+05
```

pluvial biosphere

```
file: testdcagw.o
dMAT=       61.00 degrees F
dMAP=       10.00 inches
distance to CG=     30.0000 km
cutoff distance=     20.0000 km
pumping rate=       .7233 gpd
nucl  dose (rem/y)
  U238  0.7201E+05
 Cm246  0.8101E+07
 Pu242  0.1001E+06
Am242m  0.7601E+07
 Pu238  0.9501E+05
  U234  0.6001E+05
 Th230  0.1201E+07
 Ra226  0.2801E+07
 Pb210  0.1301E+08
 Cm243  0.5401E+07
 Am243  0.7901E+07
 Pu239  0.1101E+06
  U235  0.8401E+05
 Pa231  0.2301E+08
 Ac227  0.3101E+08
 Cm245  0.8101E+07
 Pu241  0.3201E+04
 Am241  0.7901E+07
 Np237  0.1301E+08
  U233  0.6101E+05
 Th229  0.8101E+07
 Cm244  0.4301E+07
 Pu240  0.1101E+06
  U236  0.5701E+05
  U232  0.2401E+06
 Sm151  0.1201E+04
 Cs137  0.7601E+06
 Cs135  0.1001E+06
  I129  0.3101E+07
 Sn126  0.6301E+06
Sn121m  0.4301E+05
Ag108m  0.0000E+00
 Pd107  0.8101E+03
  Tc99  0.8401E+04
  Mo93  0.4401E+04
  Nb94  0.2001E+06
  Zr93  0.3501E+04
  Sr90  0.6101E+06
  Se79  0.5301E+05
  Ni63  0.3801E+04
  Ni59  0.1401E+04
  Cl36  0.8701E+05
   C14  0.1901E+05
```

Closer In Residential Water User

today's biosphere

file: testdcagw.o
dMAT= 61.00 degrees F
dMAP= 5.90 inches
distance to CG= 5.0000 km
cutoff distance= 20.0000 km

nucl	dose (rem/y)
U238	0.1860E+06
Cm246	0.2700E+07
Pu242	0.2450E+07
Am242m	0.2570E+07
Pu238	0.2340E+07
U234	0.2070E+06
Th230	0.4000E+06
Ra226	0.9670E+06
Pb210	0.3910E+07
Cm243	0.1830E+07
Am243	0.2640E+07
Pu239	0.2580E+07
U235	0.1940E+06
Pa231	0.7720E+07
Ac227	0.1030E+08
Cm245	0.2730E+07
Pu241	0.5000E+05
Am241	0.2660E+07
Np237	0.3240E+07
U233	0.2110E+06
Th229	0.2580E+07
Cm244	0.1470E+07
Pu240	0.2580E+07
U236	0.1960E+06
U232	0.9560E+06
Sm151	0.2840E+03
Cs137	0.3650E+05
Cs135	0.5160E+04
I129	0.2010E+06
Sn126	0.1420E+05
Sn121m	0.1130E+04
Ag108m	0.5560E+04
Pd107	0.1090E+03
Tc99	0.1070E+04
Mo93	0.9830E+03
Nb94	0.5210E+04
Zr93	0.1210E+04
Sr90	0.1040E+06
Se79	0.6350E+04
Ni63	0.4210E+03
Ni59	0.1530E+03
Cl36	0.2210E+04
C14	0.1520E+04

pluvial biosphere

file: testdcagw.o
dMAT= 61.00 degrees F
dMAP= 10.00 inches
distance to CG= 5.0000 km
cutoff distance= 20.0000 km

nucl	dose (rem/y)
U238	0.1861E+06
Cm246	0.2701E+07
Pu242	0.2451E+07
Am242m	0.2571E+07
Pu238	0.2341E+07
U234	0.2071E+06
Th230	0.4001E+06
Ra226	0.9671E+06
Pb210	0.3911E+07
Cm243	0.1831E+07
Am243	0.2641E+07
Pu239	0.2581E+07
U235	0.1941E+06
Pa231	0.7721E+07
Ac227	0.1031E+08
Cm245	0.2731E+07
Pu241	0.5001E+05
Am241	0.2661E+07
Np237	0.3241E+07
U233	0.2111E+06
Th229	0.2581E+07
Cm244	0.1471E+07
Pu240	0.2581E+07
U236	0.1961E+06
U232	0.9561E+06
Sm151	0.2841E+03
Cs137	0.3651E+05
Cs135	0.5161E+04
I129	0.2011E+06
Sn126	0.1421E+05
Sn121m	0.1131E+04
Ag108m	0.5561E+04
Pd107	0.1091E+03
Tc99	0.1071E+04
Mo93	0.9831E+03
Nb94	0.5211E+04
Zr93	0.1211E+04
Sr90	0.1041E+06
Se79	0.6351E+04
Ni63	0.4211E+03
Ni59	0.1531E+03
Cl36	0.2211E+04
C14	0.1521E+04

5/20/97
MSJ

cc:Mail for: Mark Jarzemba

Subject: Importance of resuspension in volcanism
From: Mark Jarzemba 5/20/97 8:12 AM
To: tjm3 at pseudo
cc: Robert Baca at CNWRA-OS2
To: cam1 at pseudo
cc: Budhi Sagar at CNWRA-OS2
cc: Wesley Patrick at CNWRA-OS2
cc: Patrick LaPlante at CNWRA-OS2
cc: James Weldy at CNWRA-SUN
cc: Brittain Hill at CNWRA-OS2

Tim/Chris,

After yesterday's telecon, discussions on this end continued for several minutes. One of the ideas that was 'kicked around' was that we perform several sets of realizations using the TPA code to determine what the overall effect of using the new information on resuspension would be.

What was proposed was that we do three sets of realizations at 5km (residential water user- inhalation and direct exposure only) and at 20 km (Amargosa farmer- inhalation, direct exposure and ingestion):

Set 1. using the dose factors currently in the code

Set 2. using a new set of dose factors where the only change is in the inhalation pathway. These would use a resuspension factor of 10E-5 and assume 100% occupancy

Set 3. using the same as 2. above but accounting for 'dilution' due to blanket thickness (i.e. only the top 1 centimeter of ash is available for resuspension).

The problem with 3. above is that the calculations would have to be done with an unofficial version of the DCAGS module, however, I think it's necessary to account for this effect as it will more accurately predict doses for the high dose realizations and these realizations tend to dominate expected value calculations. Sets 1. and 2. above would be similar to sensitivity analyses that could be conducted with the current code.

This activity will not require much effort and after I'm done I could write up a 'one pager' to document the results for discussion.

What do you think?

Mark

The purpose of this entry is to document the sets of runs described on the previous e-mail.

Set 1: The following DCAGS module was used.

```
c=======================================================================
      subroutine dcags( mxntime, mxnnucl,
     &        ntim, tim, nnucl, names,
     &        ciperm2gsatCP,
     &        remperyrpernuclgs  )
c=======================================================================
c dose through groundsurface pathway due to laydown of contaminated ashblanket
c compliance point which may be locate 25, 30, 35 km from repository
c
c mxntime = input, integer, size to dimension arrays
c mxnnucl = input, integer, size to dimension arrays
c ntim = input, integer, maximum number of times used to dimension arrays
c tim(ntim)  = input, double precision, array of times
c nnucl = input, integer, maximum number of nuclides used to dimension arrays
c names(nnucl) = input, character*6, names of nuclides to be tracked
c ciperm2gsatCP(mxntime,mxnnucl) = input, double precision, time-dependent
c               concentration [ci/m^2] of nuclides at ground surface
c               due to volcanic ashplume.  A separate module accounts for
c               the evolution of nuclides after the ash is layed down.
c               Hence, the errosion and leaching of nuclides is already
c               accounted for, and this module only converts a given
c               concentration [ci/m2] to annual effective dose equivalent
c               [rem/yr].
c remperyrpernuclgs[mxntime,nnucl] = output, double precision, array of
c            annual EDE (effective dose equivalent) per nuclide
c
c       This subroutine was re-written on 4/17/97 by Mark Jarzemba
c       in order to have a duality of how doses are calculated.
c       For compliance points closer to the repository than the cutoff pt,
c       the dcf's are based on direct exposure and inhalation only.
c       For points at a distance of greater than the cutoff pt. from the
c       repository, the dcf's are based on direct exposure, inhalation,
c       and ingestion of contaminated animal products and crops.
c
c
c
      implicit double precision (a-h,o-z)

      dimension tim(ntim)
      character*6 names(nnucl)
      character*60 name
      dimension ciperm2gsatCP(mxntime,mxnnucl)
      dimension remperyrpernuclgs(mxntime, mxnnucl)
      external ispquery
      external valuesp

      common / dcags1 / ikey
      common / dcags2 / dcf(43)

      if (ikey .ne. 73562 ) then
cc to avoid warning message, make trivial use of tim()
        ajunk = tim(1)

        call clearchar( 60, name )
        name = 'DistanceCutoffForDoseConversionDualityInDCAGS[km]'
        icutoffpt = ispquery( name )

        call clearchar( 60, name )
        name = 'DistanceToCriticalGroup[km][should be 5 or 30]'
        idist = ispquery( name )

        ikey = 73562

      endif

      cutoffpt=valuesp( icutoffpt )
      dist=valuesp( idist )

      if (dist.ge.cutoffpt) then

c
c     These dcf's are for the Amargosa Desert farmer/rancher where
c     direct exposure, inhalation, and ingestion of animal products
c     and crops has been included
c
      dcf( 1) =  0.66D+04
      dcf( 2) =  0.72D+06
      dcf( 3) =  0.95D+04
      dcf( 4) =  0.67D+06
      dcf( 5) =  0.00D+00
      dcf( 6) =  0.57D+04
      dcf( 7) =  0.10D+06
      dcf( 8) =  0.24D+06
      dcf( 9) =  0.12D+07
      dcf(10) =  0.49D+06
      dcf(11) =  0.70D+06
      dcf(12) =  0.10D+05
      dcf(13) =  0.19D+05
      dcf(14) =  0.21D+07
      dcf(15) =  0.27D+07
      dcf(16) =  0.72D+06
      dcf(17) =  0.24D+03
      dcf(18) =  0.70D+06
      dcf(19) =  0.11D+07
      dcf(20) =  0.58D+04
      dcf(21) =  0.73D+06
      dcf(22) =  0.39D+06
      dcf(23) =  0.10D+05
      dcf(24) =  0.54D+04
      dcf(25) =  0.18D+05
      dcf(26) =  0.11D+03
      dcf(27) =  0.12D+06
      dcf(28) =  0.10D+05
      dcf(29) =  0.33D+06
      dcf(30) =  0.43D+05
      dcf(31) =  0.46D+04
      dcf(32) =  0.00D+00
      dcf(33) =  0.89D+02
      dcf(34) =  0.46D+04
      dcf(35) =  0.11D+04
      dcf(36) =  0.13D+06
      dcf(37) =  0.31D+03
      dcf(38) =  0.73D+05
      dcf(39) =  0.52D+04
      dcf(40) =  0.42D+03
      dcf(41) =  0.15D+03
      dcf(42) =  0.69D+05
      dcf(43) =  0.14D+01

      else

c
c     These dcf's are for the closer in residential water user
c     and include only direct exposure and inhalation.
c

      dcf(1) = 0.24D+03
      dcf(2) = 0.79D+03
      dcf(3) = 0.52D+03
      dcf(4) = 0.93D+03
      dcf(5) = 0.48D+03
      dcf(6) = 0.28D+03
      dcf(7) = 0.48D+03
      dcf(8) = 0.57D+03
      dcf(9) = 0.24D+03
      dcf(10) = 0.11D+05
      dcf(11) = 0.53D+04
      dcf(12) = 0.51D+03
      dcf(13) = 0.13D+05
      dcf(14) = 0.49D+04
      dcf(15) = 0.21D+04
      dcf(16) = 0.83D+04
      dcf(17) = 0.80D+01
      dcf(18) = 0.53D+04
      dcf(19) = 0.36D+04
      dcf(20) = 0.27D+03
      dcf(21) = 0.10D+05
      dcf(22) = 0.47D+03
      dcf(23) = 0.55D+03
      dcf(24) = 0.26D+03
      dcf(25) = 0.11D+04
      dcf(26) = 0.49D+00
      dcf(27) = 0.48D+05
      dcf(28) = 0.30D+01
      dcf(29) = 0.22D+04
      dcf(30) = 0.48D+04
      dcf(31) = 0.43D+03
      dcf(32) = 0.00D+00
      dcf(33) = 0.22D-01
      dcf(34) = 0.69D+01
      dcf(35) = 0.46D+03
      dcf(36) = 0.13D+06
      dcf(37) = 0.13D+00
      dcf(38) = 0.24D+02
      dcf(39) = 0.18D+01
      dcf(40) = 0.35D-02
      dcf(41) = 0.14D-02
      dcf(42) = 0.59D+02
      dcf(43) = 0.14D+01

      end if

      do k = 1, nnucl
        ik = indexperiso( names(k) )
        do j=1,ntim
          remperyrpernuclgs(j,k)=dcf(ik)*ciperm2gsatCP(j,k)
        enddo
      enddo

      return
      end
```

Results:

	peak doses ($\frac{mrem}{yr}$) 5 km	peak doses ($\frac{mrem}{yr}$) 20 km
1)	~~0.238~~ 0.238	~~20 km~~ ~~12.3~~ 12.3
2)	~~0.003~~ 0.641	~~37.0~~ 37.0
3)	~~0.232~~ 0.172	~~0~~ ~~5,490.0~~ ~~5,490.0~~
4)	~~0.963~~ 67.3	~~5,490.0~~ ~~2,040.0~~ 5,490.0
5)	~~4.090~~ 0.07	~~0~~ ~~671.0~~ 0
6)	~~0.884~~ 101	~~2,040~~ ~~0.2~~ 2,040
7)	~~3.400~~ 15.6	~~671.0~~ 671.0
8)	~~3.710~~ 52.7	~~0.2~~ 0.2
9)	~~0.006~~ 0.001	~~0~~ 0
10)	~~0.001~~ 25.6	~~0~~ 0
	= 26.3 ± 11.3	825 ± 557
	$E(\dot{D})$ = ~~1.35 ± 0.53~~ mrem/yr	~~825 ± 557~~ mrem/yr

$$E(\dot{D}) = \sum_{n=1}^{N} \frac{\dot{D}_n}{N} \qquad \sigma_E = \sum_{n=1}^{N} \sqrt{\frac{1}{N(N-1)} (\dot{D} - E(\dot{D}))^2}$$

- Event penetrates the repository in all realizations
- Wind direction = -90°
- In between the 5km and 20 km dose calcs, I turned of the calls to uzft and szft to save time meaning that the stochastic rnd. #'s changed for the realization #'s.

Using the same module but with inhalation doses calculated using a resuspension factor of $M = 10^{-5}$ $1/m$

	peak doses ($\frac{mrem}{yr}$) 5 km	peak doses ($\frac{mrem}{yr}$) 20 km
1)	1,410	338
2)	3,790	100
3)	1,020	0
4)	834,000	57,000
5)	416	0
6)	598,000	5,530
7)	92,500	1,860
8)	311,000	0.5
9)	3	0
10)	151,000	754
$E(\dot{D})$	199,000 ± 91,000 $\frac{mrem}{yr}$	6,580 ± 5,630

Same as above but accounting for blanket thickness

1)	1,410	33.8
2)	847	100
3)	161	0
4)	4,290	9,670
5)	416	0
6)	95,200	5,530
7)	6,240	1,860
8)	762	6
9)	3	0
10)	89,400	0
$E(\dot{D})$	19,900 ± 12,100 $\frac{mrem}{yr}$	1,720 ± 1,043 $(\frac{mrem}{yr})$

Using the re-written DCAGS module w/ $M = 2 \times 10^{-10}\ m^{-1}$

	peak doses $(\frac{mrem}{yr})$ 5 km	peak doses $(\frac{mrem}{yr})$ 20km
1)	0.264	12.3
2)	0.604	37.0
3)	0.161	0
4)	53	5,490.0
5)	0.080	0
6)	94.5	2,040.0
7)	14.5	671.0
8)	48.3	0.2
9)	0.001	0
10)	25.3	0
E(D) =	23.7 ± 10.2	825 ± 557

- This is done as a sanity check - #o.s should match set 1 (approximately)

Description	$E(D) \pm \sigma_E$ $(\frac{mrem}{yr})$ 5 km	20km
- Using current DCAGS module w/current DCF's for inhalation	26.3 ± 11.3	825 ± 557
- Using DCF's for inhalation that use $M = 10^{-5}$ 1/m; all contamination on surface	$1.99 \times 10^{5} \pm 9.1 \times 10^{4}$	$6.5 \times 10^{3} \pm 5.6 \times 10^{3}$
- Using modified DCAGS and accounting for blanket thickness ($M = 10^{-5}\ m^{-1}$)	$1.99 \times 10^{4} \pm 1.2 \times 10^{4}$	$1.72 \times 10^{3} \pm 1.04 \times 10^{3}$
- Using modified DCAGS not accounting for blanket thickness ($M = 2 \times 10^{-10}\ m^{-1}$)	~~23.7 ± 10.2~~ MS, 5/20/97	825 ± 557

This is a printout of the modified DCAGS.F subroutine.

```
C=======================================================================
      subroutine dcags( mxntime, mxnnucl,
     &         ntim, tim, nnucl, names,
     &         ciperm2gsatCP,gramsashpercm2,
     &         remperyrpernuclgs  )
C=======================================================================
c dose through groundsurface pathway due to laydown of contaminated ashblanket
c compliance point which may be locate 25, 30, 35 km from repository
c
c mxntime = input, integer, size to dimension arrays
c mxnnucl = input, integer, size to dimension arrays
c ntim = input, integer, maximum number of times used to dimension arrays
c tim(ntim)  = input, double precision, array of times
c nnucl = input, integer, maximum number of nuclides used to dimension arrays
c names(nnucl) = input, character*6, names of nuclides to be tracked
c ciperm2gsatCP(mxntime,mxnnucl) = input, double precision, time-dependent
c                concentration [ci/m^2] of nuclides at ground surface
c                due to volcanic ashplume.  A separate module accounts for
c                the evolution of nuclides after the ash is layed down.
c                Hence, the errosion and leaching of nuclides is already
c                accounted for, and this module only converts a given
c                concentration [ci/m2] to annual effective dose equivalent
c                [rem/yr].
c remperyrpernuclgs[mxntime,nnucl] = output, double precision, array of
c              annual EDE (effective dose equivalent) per nuclide
c
c       This subroutine was re-written on 4/17/97 by Mark Jarzemba
c       in order to have a duality of how doses are calculated.
c       For compliance points closer to the repository than the cutoff pt,
c       the dcf's are based on direct exposure and inhalation only.
c       For points at a distance of greater than the cutoff pt. from the
c       repository, the dcf's are based on direct exposure, inhalation,
c       and ingestion of contaminated animal products and crops.
c
c

      implicit double precision (a-h,o-z)

      dimension tim(ntim)
      character*6 names(nnucl)
      character*60 name
      dimension ciperm2gsatCP(mxntime,mxnnucl)
      dimension remperyrpernuclgs(mxntime, mxnnucl)
      external ispquery
      external valuesp

      common / dcags1 / ikey
      common / dcags2 / dcf(43)

      if (ikey .ne. 73562 ) then
cc to avoid warning message, make trivial use of tim()
        ajunk = tim(1)

        call clearchar( 60, name )
        name = 'DistanceCutoffForDoseConversionDualityInDCAGS[km]'
        icutoffpt = ispquery( name )

        call clearchar( 60, name )
        name = 'DistanceToCriticalGroup[km][should be 5 or 30]'
        idist = ispquery( name )

        ikey = 73562

      endif

      cutoffpt=valuesp( icutoffpt )
      dist=valuesp( idist )

      if (gramsashpercm2/1.65d0.lt.1.0d0) then
          T=1.d0
        else
          T=gramsashpercm2/1.65d0
      end if
      dM=2.00d-10
      if (dist.ge.cutoffpt) then

c
c       These dcf's are for the Amargosa Desert farmer/rancher where
c       direct exposure, inhalation, and ingestion of animal products
c       and crops has been included in that order. Units of
c        (rem/yr)/(Ci/m2)
c
```

```
dcf(1) = 4.80d01+dM*1.24d12/T+4.40d02+5.90d03
dcf(2) = 6.90d01+dM*4.73d12/T+2.30d03+7.20d05
dcf(3) = 5.90d01+dM*4.31d12/T+1.10d01+9.00d03
dcf(4) = 2.60d02+dM*4.46d12/T+3.00d03+6.70d05
dcf(5) = 5.47d01+dM*4.11d12/T+1.10d01+8.34d03
dcf(6) = 6.50d01+dM*1.39d12/T+1.10d01+5.00d03
dcf(7) = 6.50d01+dM*3.42d12/T+1.40d02+1.00d05
dcf(8) = 5.60d02+dM*9.00d10/T+5.00d04+1.90d05
dcf(9) = 2.20d02+dM*1.42d11/T+1.00d05+1.10d06
dcf(10) = 1.10d04+dM*3.22d12/T+1.50d03+4.80d05
```

```
dcf(11) = 4.60d03+dM*4.62d12/T+3.10d03+6.90d05
dcf(12) = 3.20d01+dM*4.50d12/T+1.20d01+9.50d03
dcf(13) = 1.30d04+dM*1.29d12/T+4.60d02+5.40d03
dcf(14) = 3.50d03+dM*1.35d13/T+3.60d03+2.10d06
dcf(15) = 1.40d01+dM*7.02d13/T+1.70d04+2.70d06
dcf(16) = 7.60d03+dM*4.77d12/T+2.30d03+7.10d05
dcf(17) = 1.70d03+dM*8.65d10/T+4.40d-1+2.30d02
dcf(18) = 4.60d03+dM*4.66d12/T+3.10d03+6.90d05
dcf(19) = 2.60d03+dM*5.67d12/T+1.10d05+1.00d06
dcf(20) = 6.30d01+dM*1.42d12/T+4.50d02+5.10d03
dcf(21) = 7.40d03+dM*2.25d13/T+5.30d03+7.10d05
dcf(22) = 7.80d01+dM*2.60d12/T+1.20d03+3.80d05
dcf(23) = 7.00d01+dM*4.50d12/T+1.20d01+9.60d03
dcf(24) = 5.70d01+dM*1.32d12/T+4.10d02+4.70d03
dcf(25) = 8.90d01+dM*6.91d12/T+1.30d03+1.60d04
dcf(26) = 4.40d-1+dM*3.14d08/T+3.90d01+7.30d01
dcf(27) = 4.80d04+dM*3.35d08/T+5.90d04+9.50d03
dcf(28) = 3.00d00+dM*4.77d07/T+8.50d03+1.40d03
dcf(29) = 2.20d03+dM*1.82d09/T+2.70d05+4.90d04
dcf(30) = 4.80d03+dM*1.05d09/T+3.40d04+4.00d03
dcf(31) = 4.30d02+dM*1.21d08/T+3.60d03+4.30d02
dcf(32) = 0.00d00+dM*2.97d09/T+0.00d00+0.00d00
dcf(33) = 0.00d00+dM*1.34d08/T+5.40d01+3.30d01
dcf(34) = 6.90d00+dM*8.73d07/T+1.50d02+4.40d03
dcf(35) = 4.60d02+dM*2.99d08/T+1.10d04+4.80d02
dcf(36) = 1.30d05+dM*4.35d09/T+1.30d-1+1.40d03
dcf(37) = 0.00d00+dM*3.38d09/T+6.00d-2+3.10d02
dcf(38) = 2.40d01+dM*1.36d10/T+3.60d04+3.70d04
dcf(39) = 1.80d00+dM*1.03d08/T+3.50d03+1.60d03
dcf(40) = 0.00d00+dM*6.60d07/T+3.10d02+1.10d02
dcf(41) = 0.00d00+dM*2.83d07/T+1.10d02+4.00d01
dcf(42) = 5.90d01+dM*2.30d08/T+2.70d04+4.10d04
dcf(43) = 1.40d00+dM*2.19d07/T+0.00d00+0.00d00
```

```
c       dcf( 1) =  0.66D+04
c       dcf( 2) =  0.72D+06
c       dcf( 3) =  0.95D+04
c       dcf( 4) =  0.67D+06
c       dcf( 5) =  0.00D+00
c       dcf( 6) =  0.57D+04
c       dcf( 7) =  0.10D+06
c       dcf( 8) =  0.24D+06
c       dcf( 9) =  0.12D+07
c       dcf(10) =  0.49D+06
c       dcf(11) =  0.70D+06
c       dcf(12) =  0.10D+05
c       dcf(13) =  0.19D+05
c       dcf(14) =  0.21D+07
c       dcf(15) =  0.27D+07
c       dcf(16) =  0.72D+06
c       dcf(17) =  0.24D+03
c       dcf(18) =  0.70D+06
c       dcf(19) =  0.11D+07
c       dcf(20) =  0.58D+04
c       dcf(21) =  0.73D+06
c       dcf(22) =  0.39D+06
c       dcf(23) =  0.10D+05
c       dcf(24) =  0.54D+04
c       dcf(25) =  0.18D+05
c       dcf(26) =  0.11D+03
c       dcf(27) =  0.12D+06
c       dcf(28) =  0.10D+05
c       dcf(29) =  0.33D+06
c       dcf(30) =  0.43D+05
c       dcf(31) =  0.46D+04
c       dcf(32) =  0.00D+00
c       dcf(33) =  0.89D+02
c       dcf(34) =  0.46D+04
c       dcf(35) =  0.11D+04
c       dcf(36) =  0.13D+06
c       dcf(37) =  0.31D+03
c       dcf(38) =  0.73D+05
c       dcf(39) =  0.52D+04
c       dcf(40) =  0.42D+03
c       dcf(41) =  0.15D+03
c       dcf(42) =  0.69D+05
c       dcf(43) =  0.14D+01
```



```
else

c
c   These dcf's are generated using the new modeling and are for
c   inhalation only.
c
c
c
c   These dcf's are for the closer in residential water user
c   and include only direct exposure and inhalation. Units of
c   (rem/y)/(Ci/m2). M is the resuspension factor in 1/m.
c
c
c
c
c
c
c
      dcf(1) = 4.80d01+dM*1.24d12/T
      dcf(2) = 6.90d01+dM*4.73d12/T
```

```
dcf(3) = 5.90d01+dM*4.31d12/T
dcf(4) = 2.60d02+dM*4.46d12/T
dcf(5) = 5.47d01+dM*4.11d12/T
dcf(6) = 6.50d01+dM*1.39d12/T
dcf(7) = 6.50d01+dM*3.42d12/T
dcf(8) = 5.60d02+dM*9.00d10/T
dcf(9) = 2.20d02+dM*1.42d11/T
dcf(10) = 1.10d04+dM*3.22d12/T
dcf(11) = 4.60d03+dM*4.62d12/T
dcf(12) = 3.20d01+dM*4.50d12/T
dcf(13) = 1.30d04+dM*1.29d12/T
dcf(14) = 3.50d03+dM*1.35d13/T
dcf(15) = 1.40d01+dM*7.02d13/T
dcf(16) = 7.60d03+dM*4.77d12/T
dcf(17) = 1.70d03+dM*8.65d10/T
dcf(18) = 4.60d03+dM*4.66d12/T
dcf(19) = 2.60d03+dM*5.67d12/T
dcf(20) = 6.30d01+dM*1.42d12/T
dcf(21) = 7.40d03+dM*2.25d13/T
dcf(22) = 7.80d01+dM*2.60d12/T
dcf(23) = 7.00d01+dM*4.50d12/T
dcf(24) = 5.70d01+dM*1.32d12/T
dcf(25) = 8.90d01+dM*6.91d12/T
dcf(26) = 4.40d-1+dM*3.14d8/T
dcf(27) = 4.80d04+dM*3.35d8/T
dcf(28) = 3.00d00+dM*4.77d7/T
dcf(29) = 2.20d03+dM*1.82d9/T
dcf(30) = 4.80d03+dM*1.05d9/T
dcf(31) = 4.30d02+dM*1.21d8/T
dcf(32) = 0.00d00+dM*2.97d9/T
dcf(33) = 0.00d00+dM*1.34d8/T
dcf(34) = 6.90d00+dM*8.73d7/T
dcf(35) = 4.60d02+dM*2.99d8/T
dcf(36) = 1.30d05+dM*4.35d9/T
dcf(37) = 0.00d00+dM*3.38d9/T
dcf(38) = 2.40d01+dM*1.36d10/T
dcf(39) = 1.80d00+dM*1.03d8/T
dcf(40) = 0.00d00+dM*6.60d7/T
dcf(41) = 0.00d00+dM*2.83d7/T
dcf(42) = 5.90d01+dM*2.30d8/T
dcf(43) = 1.40d00+dM*2.19d7/T
```

```
c       dcf(1) = 0.24D+03
c       dcf(2) = 0.79D+03
c       dcf(3) = 0.52D+03
c       dcf(4) = 0.93D+03
c       dcf(5) = 0.48D+03
c       dcf(6) = 0.28D+03
c       dcf(7) = 0.48D+03
c       dcf(8) = 0.57D+03
c       dcf(9) = 0.24D+03
c       dcf(10) = 0.11D+05
c       dcf(11) = 0.53D+04
c       dcf(12) = 0.51D+03
c       dcf(13) = 0.13D+05
c       dcf(14) = 0.49D+04
c       dcf(15) = 0.21D+04
c       dcf(16) = 0.83D+04
c       dcf(17) = 0.80D+01
c       dcf(18) = 0.53D+04
c       dcf(19) = 0.36D+04
c       dcf(20) = 0.27D+03
c       dcf(21) = 0.10D+05
c       dcf(22) = 0.47D+03
c       dcf(23) = 0.55D+03
c       dcf(24) = 0.26D+03
c       dcf(25) = 0.11D+04
c       dcf(26) = 0.49D+00
c       dcf(27) = 0.48D+05
c       dcf(28) = 0.30D+01
c       dcf(29) = 0.22D+04
c       dcf(30) = 0.48D+04
c       dcf(31) = 0.43D+03
c       dcf(32) = 0.00D+00
c       dcf(33) = 0.22D-01
c       dcf(34) = 0.69D+01
c       dcf(35) = 0.46D+03
c       dcf(36) = 0.13D+06
c       dcf(37) = 0.13D+00
c       dcf(38) = 0.24D+02
c       dcf(39) = 0.18D+01
c       dcf(40) = 0.35D-02
c       dcf(41) = 0.14D-02
c       dcf(42) = 0.59D+02
c       dcf(43) = 0.14D+01

    end if

    do k = 1, nnucl
      ik = indexperiso( names(k) )
      do j=1,ntim
```

```
        remperyrpernuclgs(j,k)=dcf(ik)*ciperm2gsatCP(j,k)
      enddo
    enddo

    return
    end
```

5/23/97
MSJ

As part of our checks of the TPA code, we are going through the code and performing units checks. The following pages document my units checks of the ASHPLUME.P code.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c                                                                      c
c
        Program ashplume                                               c
c                                                                      c
c       This program calculates the ash and fuel areal mass density    c
c       at the x,y points on the earth's surface. The user is given    c
c       the option of recording particulate size distribution          c
c       information if so desired                                      c
c                                                                      c
c       Written by: Mark S. Jarzemba, 1/22/97 (this version)           c
c                                                                      c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c       maxd = maximum dimension of isopach points                     c
c       nmy  = maximum number of volcanoes                             c
c                                                                      c
      parameter (maxd=500)
      parameter (nmy=1000)
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      character*60 version,title
      common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
      common /two/ h,werupt0,airden,airvis,c,u
      common /three/ fdmin,fdmax,fdmean,hmin,hmax,xmin,xmax
      common /three1/ dmin,dmax,rhomin,rhomax,rhomean
      common /four/ ymin,ymax,acutoff
      common /five/ x,y,udir,frhomin,frhomax,frhomean,drho,cmax
      common /six/ numptsx,numptsy
      common /seven/ Uran
      common /eight/ rhocut
      common /nine/ v,icount
      common /eleven/ ashrholow,ashrhohi
      common /twelve/ power,tdur
      common /thirteen/ numapts
      common /fifteen/ ipchar
      common /sixteen/ title
      dimension v(10000),xash(maxd,maxd),xfuel(maxd,maxd)
      open(unit=10,file='ashplume.out',status='unknown')
      version='ASHPLUME version 1.0'

c                                                                      c
c       Uran    = amount of waste extruded (1 container ~ 10MTU)       c
c       icount  = a counter for the string of random numbers           c
c                 stored in v                                          c
c       iseed   = the random number seed                               c
c       max     = the number of random numbers to get- 8 random nos.   c
c                 per realization                                      c
c       numapts = the number of particle sizes to use in the histogrm  c
c                 of particle size at the dose pt.                     c
c                                                                      c

      icount=1
      iseed=10
      numapts=30
      write (10,*)'iseed= ',iseed
      write (10,*) version
      max=10000
      pi=dacos(-1.d0)
      call rand(iseed,v,max)
```



```
      ichoice=0
c                                                                      c
c        ask the user for their choice of either a number of           c
c        stochastically sampled volcanos or a specific input parameter c
c        set for one volcano                                           c
c                                                                      c
      do while (ichoice.eq.0)

       write (6,31) '************************************************
     &**********************'
       write (6,30) '*',' ','*'
        write (6,30) '*',' Do you want stochastic sampling of numerou
     &s volcanoes (ENTER 1)   ','*'
       write (6,30) '*',' ','*'
       write (6,30) '*','                                  OR
     &                ','*'
             write (6,30) '*',' ','*'
        write (6,30) '*','   one volcano for  a  specific input  param
     &eter  set   (ENTER 2)?   ','*'
       write (6,30) '*',' ','*'
       write (6,30) '*','   For the stochastic sampling routine,the c
     &ode will take input     ','*'
       write (6,30) '*','    data from the file -ashplume.in-; else you
     & will be asked to      ','*'
       write (6,30) '*','      input some of the parameters from the
     & keyboard              ','*'
       write (6,30) '*',' ','*'
        write (6,31) '************************************************
     &**********************'
30      format (a1,a68,a1)
31      format (a70)
       read (5,*) ichoice
        if (ichoice.ne.1.and.ichoice.ne.2) then
          write (6,*)
          write (6,*)'*** TRY AGAIN- must  enter a 1 or a 2 ***'
          write (6,*)
          ichoice=0
        end if
      end do
      if (ichoice.eq.1) then
        write (6,*) 'Number of volcanoes to evaluate? (max=1000)'
       read (5,*) num
      else
        num=1
      end if
      write (6,*)
      write (6,*)'Would you like particulate size information at
     & the dose points?'
       write (6,*)
11        write (6,*)'Enter 1 for yes or 2 for no'
       read (5,*) ipchar
       if (ipchar.ne.1.and.ipchar.ne.2) then
         write (6,*)' you must enter a 1 or 2'
        go to 11
       end if
c                                                                      c
c        time to calculate xash, xfuel in g/cm**2                      c
c                                                                      c
```

```
      do k=1,num
c
c     inputdata selects the parameters for the realization
c
         call inputdata
         if (k.eq.1) then
           write(10,333) title
333           format(a60)
           write(10,*)
         end if
         if (ichoice.eq.2) then
           call userinput
         end if
         call outheader (k)
         write (6,*)
         write (10,*)
         write (6,50)'x (km)','y (km)','xash (g/cm^2)',
     &               'xfuel (g/cm^2)'
              write (10,50)'x (km)','y (km)','xash (g/cm^2)',
     &               'xfuel (g/cm^2)'
50         format(2a12,2a18)
         write (6,*)
         write (10,*)
c
c     determine the grid spacing
c
         if (numptsx.ne.1) then
           deltax=(xmax-xmin)/(numptsx-1)   km
         end if
         if (numptsy.ne.1) then
           deltay=(ymax-ymin)/(numptsy-1)   km
         end if
         x=xmin
         itemp1=0
c                                                              c
c     start scrolling through x starting at xmin               c
c                                                              c
         do i=1,numptsx
           y=ymin
           temp2=0.d0
           blahold=0.d0
c                                                              c
c     start scrolling through y starting at ymin               c
c                                                              c
           do j=1,numptsy
c                                                              c
c     don't calculate if x=y=0, i.e. invalid at volcano vent.  c
c     temp variables to determine when the blanket has         c
c     been 'scrolled through' for this realization. Do this    c
c     check because you don't want to calculate a bunch of zeros c
c                                                              c
             if (x.ne.0.d0.or.y.ne.0.d0) then
               call ashcalca(xash(i,j),k)
               call ashcalcf(xfuel(i,j))
c                                                              c
c     If so selected, now calculate the particle size histograms c
c     at the dose points                                       c
c                                                              c
```

```
c
c
c
           if (ipchar.eq.1) then
            call histf(rhomin,cmax,x,y,k,frhomin,frhomax,
     &                 frhomean)
           end if
           blah=xash(i,j)                                  g-ash/cm²
           if (blah.lt.acutoff) then
             blah=0.d0
            xash(i,j)=0.d0                                 g-ash/cm²
            xfuel(i,j)=0.d0                                g-fuel/cm²
           end if
           temp2=temp2+blah
           if (temp2.ne.0.d0) then
             itemp1=1
           end if
          write (6,100) x,y,xash(i,j),xfuel(i,j)
          write (10,100) x,y,xash(i,j),xfuel(i,j)
100           format (2f12.3,2e18.4)
           end if
         if (blahold.gt.0.d0.and.blah.eq.0.d0) then
           go to 20
         end if
         blahold=blah                                      g-ash/cm²
                                                             km
         y=y+deltay
       end do
20       continue
       if (temp2.eq.0.d0.and.itemp1.eq.1) then
         write(6,*)
         write(10,*)
         go to 10
       end if
       x=x+deltax
       write (6,*)
       write (10,*)
     end do
10     continue
   end do
   close(unit=10)
   end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
      subroutine outheader (ivol)
c                                                                      c
c      This subroutine outputs the header blocks                       c
c                                                                      c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      parameter (nmx=200, nmy=200)
      parameter (maxd= 200)
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
      common /two/ h,werupt0,airden,airvis,c,u
      common /three/ fdmin,fdmax,fdmean,hmin,hmax,xmin,xmax
      common /threel/ dmin,dmax,rhomin,rhomax,rhomean
      common /four/ ymin,ymax,acutoff
      common /five/ x,y,udir,frhomin,frhomax,frhomean,drho,cmax
      common /six/ numptsx,numptsy
      common /seven/ Uran
      common /eight/ rhocut
```

```
      common /twelve/ power,tdur
c
c       write the volcanic parameters to the screen
c
      write (6,120) '************************************
     &*********************************************'
      write (6,130) '*',' ','*'
       write (6,140) '*          ','realization number',ivol,'*'
       write (6,100) '*          ','wind speed (cm/s)',u,'*'
       write (6,100) '*          ','wind direction (deg)',udir,'*'
       write (6,100) '*          ','mean particle diameter (cm)',
     & dmean,'*'
       write (6,100) '*          ','log- std dev',dsigma,'*'
       write (6,100) '*          ','column ht (km)',hmax,'*'
       write (6,110) '*          ','event duration (s)',tdur,'*'
       write (6,110) '*          ','ash mass (g)',q,'*'
       write (6,110) '*          ','event power (W)',power,'*'
       write (6,100) '*          ','beta',beta,'*'
       write (6,100) '*          ','vent exit velocity (cm/s)',
     & werupt0,'*'
       write (6,100) '*          ','particle shape parameter',fshape,
     & '*'
       write (6,110) '*          ','air density (g/cc)',airden,'*'
       write (6,110) '*          ','air viscosity (g/cm-s)',
     & airvis,'    *'
       write (6,100) '*          ','eddy diff. constant (cm2/s5/2)',
     & c,'*'
       write (6,100) '*          ','size cutoff (cm)',dmax,'*'
      write (6,100) '*          ','incorporation ratio',rhocut,'*'
      write (6,100) '*          ','fuel particle minimum log-diam',
     &                          frhomin,'*'
      write (6,100) '*          ','fuel particle median log-diam',
     &                  frhomean,'*'
      write (6,100) '*          ','fuel particle maximum log-diam',
     &                  frhomax,'*'
      write (6,110) '*          ','total fuel mass available (g)',
     &                  Uran,'*'
      write (6,130) '*',' ','*'
       write (6,120) '************************************
     &*********************************************'
c
c       write the volcanic parameters to the output file
c
      write (10,120) '************************************
     &*********************************************'
      write (10,130) '*',' ','*'
       write (10,140) '*          ','realization number',ivol,'*'
      write (10,100) '*          ','wind speed (cm/s)',u,'*'
       write (10,100) '*          ','wind direction (deg)',udir,'*'
       write (10,100) '*          ','mean particle diameter (cm)',
     & dmean,'*'
       write (10,100) '*          ','log- std dev',dsigma,'*'
       write (10,100) '*          ','column ht (km)',hmax,'*'
       write (10,110) '*          ','event duration (s)',tdur,'*'
       write (10,110) '*          ','ash mass (g)',q,'    *'
       write (10,110) '*          ','event power (W)',power,'*'
       write (10,100) '*          ','beta',beta,'*'
       write (10,100) '*          ','vent exit velocity (cm/s)',
```



```
     & werupt0,'*'
       write (10,100) '*          ','particle shape parameter',
     & fshape,'    *'
       write (10,110) '*          ','air density (g/cc)',airden,'*'
       write (10,110) '*          ','air viscosity (g/cm-s)',
     & airvis,'*'
       write (10,100) '*          ','eddy diff. constant (cm2/s5/2)',
     & c,'*'
       write (10,100) '*          ','size cutoff (cm)',dmax,'*'
      write (10,100) '*          ','incorporation ratio',rhocut,'*'
      write (10,100) '*          ','fuel particle minimum log-diam',
     &                          frhomin,'*'
      write (10,100) '*          ','fuel particle median log-diam',
     &                          frhomean,'*'
      write (10,100) '*          ','fuel particle maximum log-diam',
     &                          frhomax,'*'
      write (10,110) '*          ','total fuel mass available (g)',
     &                          Uran,'*'
      write (10,130) '*',' ','*'
       write (10,120) '************************************
     &*********************************************'
100       format(a10,a30,f20.4,a10)
110       format(a10,a30,e20.4,a10)
120       format(a70)
130       format(a1,a68,a1)
140       format(a1,a39,i20,a10)
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
      subroutine userinput
c
c       This subroutine asks the user for the specific parameters      c
c       for a given volcano if that option has been chosen             c
c                                                                      c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
      common /two/ h,werupt0,airden,airvis,c,u
      common /three/ fdmin,fdmax,fdmean,hmin,hmax,xmin,xmax
      common /three1/ dmin,dmax,rhomin,rhomax,rhomean
      common /four/ ymin,ymax,acutoff
      common /five/ x,y,udir,frhomin,frhomax,frhomean,drho,cmax
      common /six/ numptsx,numptsy
      common /seven/ Uran
      common /eight/ rhocut
      common /twelve/ power,tdur
      write (6,*) 'NOTE: the folowing parameters are still input'
      write (6,*) 'from the input data file -ashplume.in-'
      write (6,*)
      write (6,*) '  grid parameteres'
      write (6,*) '  air properties (viscocity,density)'
      write (6,*) '  fuel particle size characteristics:'
      write (6,*) '       (min,med,max diameters)'
      write (6,*) '  ash particle shape parameter'
      write (6,*) '  eddy difusivity constant'
      write (6,*) '  max ash particle diameter for transport'
```

```
      write (6,*) '  cutoff ash blanket density'
      write (6,*)
      write (6,*) 'please input the following parameters'
      write (6,*)

      write (6,*) 'the event duration (in sec)'
      read (5,*) tdur
      write (6,*) 'the column height (in km)'
      read (5,*) hmax
      write (6,*) 'the magical constant beta'
      read (5,*) beta
      write (6,*) 'the mean ash particle diameter (in cm)'
      read (5,*) dmean
      write (6,*) 'sigma for the ash lognormal dist. (unitless)'
      read (5,*) dsigma
      write (6,*) 'the incorporation ratio'
      read (5,*) rhocut
      write (6,*) 'the mass of fuel to incorporate'
      read (5,*) Uran
      write (6,*) 'the wind direction- relation to due east (deg)'
      read (5,*) udir
      write (6,*) 'the wind speed (cm/s)'
      read (5,*) u
      write (6,*) 'the initial eruption velocity (cm/s)'
      read (5,*) werupt0

      rhomean=dlog10(dmean)
      rhomin=rhomean-5.d0*dsigma
      rhomax=rhomean+5.d0*dsigma
      power=(hmax/0.0082d0)**4
      qdot=(hmax/0.24)**4.d0
      q=qdot*tdur*1.0d3
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                     c
      subroutine ashcalca(ash,ivol)
c                                                                     c
c       This subroutine calculates the ash densities at the dose pts. c
c                                                                     c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      external dinnerf,dinnera
      common /three1/ dmin,dmax,rhomin,rhomax,rhomean
      common /five/ x,y,udir,frhomin,frhomax,frhomean,drho,cmax
      common /thirteen/ numapts
c                                                                     c
c       do integration up to rhomax or a max corresponding to the     c
c       limit in the input file depending on which is greater         c
c                                                                     c
c

      cmax=rhomax
      blah=dlog10(dmax)
      if (cmax.ge.blah) then
        cmax=blah
      end if
      call qromb1(dinnera,rhomin,cmax,ash,x,y,ivol,numapts)
```

empirical relationship from Wilson,(1978) W
empirical relationship from Wilson,(1978) kg/s
g


```
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                     c
      subroutine ashcalcf(fuel)
c                                                                     c
c       This subroutine calculates the fuel densities at the dose pts.c
c                                                                     c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      external dinnerf,dinnera
      common /three1/ dmin,dmax,rhomin,rhomax,rhomean
      common /five/ x,y,udir,frhomin,frhomax,frhomean,drho,cmax
      common /thirteen/ numapts
c                                                                     c
c       do integration up to rhomax or a max corresponding to the userc
c       input depending on which is greater                           c
c                                                                     c

      cmax=rhomax
      blah=dlog10(dmax)
      if (cmax.ge.blah) then
        cmax=blah
      end if
      call qromb2(dinnerf,rhomin,cmax,fuel)
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                     c
      function dinnerf(rho)
c                                                                     c
c       This function does the inner integration of equation (1) of   c
c       Jarzemba 1996 while accounting for fuel mass/ash mass to      c
c       calculate the fuel depostion                                  c
c                                                                     c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /three/ fdmin,fdmax,fdmean,hmin,hmax,xmin,xmax
      external dintegrandf
c                                                                     c
c       dintegrandf is the integrand of Eq 1 of Jarzemba (1996)       c
c       including the fuel fraction                                   c
c                                                                     c
c       call qromb(dintegrandf,hmin,hmax,s,rho)
      call qgaus(dintegrandf,hmin,hmax,s,rho)
      dinnerf=s
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                     c
      function dinnera(rho)
c                                                                     c
c       This function does the inner integration of equation (1) of   c
c       Jarzemba 1996 to calculate the ash deposition                 c
c                                                                     c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
```

```
      implicit integer (i-n)
      common /three/ fdmin,fdmax,fdmean,hmin,hmax,xmin,xmax
      external dintegranda
c                                                                     c
c       dintegranda is the integrand of Eq 1 of Jarzemba (1996)       c
c                                                                     c
c       call qromb(dintegranda,hmin,hmax,s,rho)
      call qgaus(dintegranda,hmin,hmax,s,rho)
      dinnera=s
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                     c
      function dintegrandf(z,rho)
c                                                                     c
c       This function is the integrand for fuel deposition-           c
c       the coordinate system has to be adjusted to match the main    c
c       coordinate system                                             c
c                                                                     c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
      common /two/ h,werupt0,airden,airvis,c,u
      common /five/ x,y,udir,frhomin,frhomax,frhomean,drho,cmax
      pi=dacos(-1.d0)
c                                                                     c
c       must perfrom a coordinate transformation from the main system c
c       [x,y] to the plume calculation system [xp,yp] that has x      c
c       aligned with the wind                                         c
c                                                                     c
      if (x.ne.0.d0) then
        theta=datan(y/x)                                   radians
      else if (x.eq.0.d0 .and. y.lt.0.d0) then
        theta=-pi/2.d0                                     radians
      else if (x.eq.0.d0 .and. y.gt.0.d0) then
        theta=pi/2.d0                                      radians
      end if
      theta=theta*180.d0/pi                                radians - degrees
      if (y.lt.0.d0 .and. x.lt.0.d0) then
        theta=theta-180.d0
      else if (y.gt.0.d0 .and. x.lt.0.d0) then
        theta=theta+180.d0
      else if (y.eq.0.d0 .and. x.lt.0.d0) then
        theta=-180.d0
      else if (y.eq.0.d0 .and. x.gt.0.d0) then
        theta=0.d0
      end if
      thetap=theta-udir                                    degrees
      if (thetap.ge.-360.d0 .and. thetap.le.-180.d0) then
        thetap=thetap+360.d0
      else if (thetap.gt.180.d0 .and. thetap.le.360.d0) then
        thetap=thetap-360.d0
      end if
      thetapr=thetap*pi/180.d0                             radians
      if (thetap.gt.0.d0 .and. thetap.lt.90.d0) then
```



```
        xp=dsqrt((x**2+y**2)/(1+(dtan(thetapr))**2))
        yp=dsqrt((x**2+y**2)/(1+(1/(dtan(thetapr))**2)))
      else if (thetap.gt.90.d0 .and. thetap.lt.180.d0) then
        xp=-dsqrt((x**2+y**2)/(1+(dtan(thetapr))**2))
        yp=dsqrt((x**2+y**2)/(1+(1/(dtan(thetapr))**2)))
      else if (thetap.gt.-90.d0 .and. thetap.lt.0.d0) then
        xp=dsqrt((x**2+y**2)/(1+(dtan(thetapr))**2))
        yp=-dsqrt((x**2+y**2)/(1+(1/(dtan(thetapr))**2)))
      else if (thetap.gt.-180.d0 .and. thetap.lt.-90.d0) then
        xp=-dsqrt((x**2+y**2)/(1+(dtan(thetapr))**2))
        yp=-dsqrt((x**2+y**2)/(1+(1/(dtan(thetapr))**2)))
      else if (thetap.eq.0.d0) then                          Km
        xp=dsqrt(x**2+y**2)
        yp=0.d0
      else if (thetap.eq.180.d0.or.thetap.eq.-180.d0) then
        xp=-dsqrt(x**2+y**2)
        yp=0.d0
      else if (thetap.eq.90.d0) then
        xp=0.d0
        yp=dsqrt(x**2+y**2)
      else if (thetap.eq.-90.d0) then
        xp=0.d0
        yp=-dsqrt(x**2+y**2)
      end if
      xpcm=1.0d5*xp                                          cm
      ypcm=1.0d5*yp (cm - cm/s · s)^2   cm^2                 cm
      arg=-5.d0*((xpcm-u*tf(z,rho))**2+ypcm**2)/
     &                (8.d0*c*(tf(z,rho)+ts(z))**(2.5d0))  cm^2/s^2 · s^2/s^2   unitless
      dintegrandf=FF(rho)*(5.d0*P(z,rho)*q*f(rho))/
     &                (8.d0*pi*c*(tf(z,rho)+ts(z))**(2.5d0))*   g/cm^2/s^2 · s^2/km   gfuel/cm^2/km
     &                      dmyexp(arg)
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                     c
      function dintegranda(z,rho)
c                                                                     c
c       This function is the integrand for ash deposition-            c
c       the coordinate system has to be adjusted to match the main    c
c       coordinate system                                             c
c                                                                     c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
      common /two/ h,werupt0,airden,airvis,c,u
      common /five/ x,y,udir,frhomin,frhomax,frhomean,drho,cmax
      pi=dacos(-1.d0)
c                                                                     c
c       must perfrom a coordinate transformation from the main system c
c       [x,y] to the plume calculation system [xp,yp] that has x      c
c       direction aligned with the wind                               c
c                                                                     c
      if (x.ne.0.d0) then
        theta=datan(y/x)                                   radians
      else if (x.eq.0.d0 .and. y.lt.0.d0) then
        theta=-pi/2.d0                                     radians
      else if (x.eq.0.d0 .and. y.gt.0.d0) then
```

```
  theta=pi/2.d0
end if
theta=theta*180.d0/pi
if (y.lt.0.d0 .and. x.lt.0.d0) then
  theta=theta-180.d0
else if (y.gt.0.d0 .and. x.lt.0.d0) then
  theta=theta+180.d0
else if (y.eq.0.d0 .and. x.lt.0.d0) then
  theta=-180.d0
else if (y.eq.0.d0 .and. x.gt.0.d0) then
  theta=0.d0
end if
thetap=theta-udir
if (thetap.ge.-360.d0 .and. thetap.le.-180.d0) then
  thetap=thetap+360.d0
else if (thetap.gt.180.d0 .and. thetap.le.360.d0) then
  thetap=thetap-360.d0
end if
thetapr=thetap*pi/180.d0
if (thetap.gt.0.d0 .and. thetap.lt.90.d0) then
  xp=dsqrt((x**2+y**2)/(1+(dtan(thetapr))**2))
  yp=dsqrt((x**2+y**2)/(1+(1/(dtan(thetapr))**2)))
else if (thetap.gt.90.d0 .and.thetap.lt.180.d0) then
  xp=-dsqrt((x**2+y**2)/(1+(dtan(thetapr))**2))
  yp=dsqrt((x**2+y**2)/(1+(1/(dtan(thetapr))**2)))
else if (thetap.gt.-90.d0 .and. thetap.lt.0.d0) then
  xp=dsqrt((x**2+y**2)/(1+(dtan(thetapr))**2))
  yp=-dsqrt((x**2+y**2)/(1+(1/(dtan(thetapr))**2)))
else if (thetap.gt.-180.d0 .and. thetap.lt.-90.d0) then
  xp=-dsqrt((x**2+y**2)/(1+(dtan(thetapr))**2))
  yp=-dsqrt((x**2+y**2)/(1+(1/(dtan(thetapr))**2)))
else if (thetap.eq.0.d0) then
   xp=dsqrt(x**2+y**2)
  yp=0.d0
else if (thetap.eq.180.d0.or.thetap.eq.-180.d0) then
  xp=-dsqrt(x**2+y**2)
 yp=0.d0
else if (thetap.eq.90.d0) then
  xp=0.d0
  yp=dsqrt(x**2+y**2)
else if (thetap.eq.-90.d0) then
 xp=0.d0
  yp=-dsqrt(x**2+y**2)
end if
xpcm=1.0d5*xp
ypcm=1.0d5*yp
arg=-5.d0*((xpcm-u*tf(z,rho))**2+ypcm**2)/
&                  (8.d0*c*(tf(z,rho)+ts(z))**(2.5d0))
dintegranda=(5.d0*P(z,rho)*q*f(rho))/
&                  (8.d0*pi*c*(tf(z,rho)+ts(z))**(2.5d0))*
&                        dmyexp(arg)
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function dmyexp(x)
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

radians

radians → degrees

degrees

degrees

degrees

degrees

degrees

degrees

radians

km

cm

cm

$(cm - cm/s \cdot s)^2$ cm^2

$cm^2/s^{5/2}\ (s)^{5/2} = cm^2/cm^2$ ✓

$g/cm^2/s^{5/2} \cdot s^{5/2}/km = g/cm^2\text{-}km$ ✓

MSJ


```
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      if (x.lt.-230.d0) then
        dmyexp=0.d0
      else if (x.gt.230.d0) then
        dmyexp=1.d99
        write(6,*) 'large exponential has been truncated to 1.d99'
      else
        dmyexp=dexp(x)
      end if
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function tf(z,rho)
c                                                                       c
c      This function subroutine calculates the particle fall time      c
c      in s as a function of height (z) and rho                         c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      tf=0.752d06*((1-dmyexp(-0.0625d0*z))/v0(rho))**0.926d0
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function ts(z)
c
c      This function subroutine calculates the particle diffusion      c
c      time in the eruption column in s                                 c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /two/ h,werupt0,airden,airvis,c,u
      zcm=z*1.0d5
      ts=((5.0d0*zcm**2)/(288.d0*c))**(0.4d0)
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function f(rho)
c                                                                       c
c      This function in the pdf of ash mass with the log-particle      c
c      diameter                                                         c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
      common /three1/ dmin,dmax,rhomin,rhomax,rhomean
      pi=dacos(-1.d0)
      arg=-((rho-rhomean)**2)/(2*dsigma**2)
      f=1/(dsqrt(2*pi)*dsigma)*dmyexp(arg)
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

empirical function from Suzuki tf in s for Vo in cm/s ✓

empirical function from Suzuki; ts in s for c in $cm^2/s^{5/2}$, z in cm ✓

unitless since dsigma is unitless

MSJ

```
c                                                                       c
      function dm(rho)
c                                                                       c
c       This function is the pdf for fuel mass as a function           c
c       of the log-particle diameter                                    c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /five/ x,y,udir,frhomin,frhomax,frhomean,drho,cmax
      dk1=2.d0/((frhomax-frhomin)*(frhomean-frhomin))    all 'rho' are unitless
      dk2=-2.d0/((frhomax-frhomin)*(frhomax-frhomean))   all 'rho' are unitless
      if (rho.le.frhomin.or.rho.ge.frhomax) then
        dm=0.d0
      end if
      if (rho.gt.frhomin.and.rho.lt.frhomean) then
        dm=dk1*(rho-frhomin)                              unitless
      end if
      if (rho.ge.frhomean.and.rho.lt.frhomax) then
        dm=dk2*(rho-frhomean)+dk1*(frhomean-frhomin)      unitless
      end if
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function FF(rho)
c                                                                       c
c       This function calculates the fuel fraction as a function       c
c       of log-diameter                                                 c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      external Fcumm,FFintegrand
      common /five/ x,y,udir,frhomin,frhomax,frhomean,drho,cmax
      common /seven/ Uran
      common /eight/ rhocut
      cmin=frhomin+rhocut
      call qromb3(FFintegrand,cmin,rho,s)
      FF=s                                                 unitless
c                                                                       c
c       If FF is greater than 1 then it is truncated to zero in        c
c       order to remove the contamination that these particles carry   c
c       from the scenario, as they would be too dense to transport by  c
c       this mechanism                                                  c
c                                                                       c
      if (FF.ge.1.d0) then
        FF=0.d0
      end if
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function FFintegrand(rho)
c                                                                       c
c       This function is the integrand of Equation 2-7 from            c
c       IM 5708-771-610                                                 c
c                                                                       c
```



```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      external Fcumm
      common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
      common /seven/ Uran
      common /eight/ rhocut
      FFintegrand=Uran/q*dm(rho-rhocut)/(1.d0-Fcumm(rho))   unitless= g/g
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      function Fcumm(rho)
c                                                                       c
c       This subroutine uses a table lookup to yield the CDF           c
c       from thestandard normal distribution                           c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
      common /three1/ dmin,dmax,rhomin,rhomax,rhomean
      dimension fc(400)
c                                                                       c
c       This subroutine will accurately sample from the                c
c       cdf of the standard normal distribution out to                 c
c       at least the fourth decimal place                              c
c                                                                       c
c                                                                       c
      data (fc(i),i=1,95)
     & /0.0000d0,0.0040d0,0.0080d0,0.0120d0,0.0160d0,
     &  0.0199d0,0.0239d0,0.0279d0,0.0319d0,0.0359d0,
     &  0.0398d0,0.0438d0,0.0478d0,0.0517d0,0.0557d0,
     &  0.0596d0,0.0636d0,0.0675d0,0.0714d0,0.0754d0,
     &  0.0793d0,0.0832d0,0.0871d0,0.0910d0,0.0948d0,
     &  0.0987d0,0.1026d0,0.1064d0,0.1103d0,0.1141d0,
     &  0.1179d0,0.1217d0,0.1255d0,0.1293d0,0.1331d0,
     &  0.1368d0,0.1406d0,0.1443d0,0.1480d0,0.1517d0,
     &  0.1554d0,0.1591d0,0.1628d0,0.1664d0,0.1700d0,
     &  0.1736d0,0.1772d0,0.1808d0,0.1844d0,0.1879d0,
     &  0.1915d0,0.1950d0,0.1985d0,0.2019d0,0.2054d0,
     &  0.2088d0,0.2123d0,0.2157d0,0.2190d0,0.2224d0,
     &  0.2258d0,0.2291d0,0.2324d0,0.2357d0,0.2389d0,
     &  0.2422d0,0.2454d0,0.2486d0,0.2518d0,0.2549d0,
     &  0.2580d0,0.2612d0,0.2642d0,0.2673d0,0.2704d0,
     &  0.2734d0,0.2764d0,0.2794d0,0.2823d0,0.2852d0,
     &  0.2881d0,0.2910d0,0.2939d0,0.2967d0,0.2996d0,
     &  0.3023d0,0.3051d0,0.3078d0,0.3106d0,0.3133d0,
     &  0.3159d0,0.3186d0,0.3212d0,0.3238d0,0.3264d0/
      data (fc(i),i=96,190)
     & /0.3289d0,0.3315d0,0.3340d0,0.3365d0,0.3389d0,
     &  0.3413d0,0.3438d0,0.3461d0,0.3485d0,0.3508d0,
     &  0.3531d0,0.3554d0,0.3577d0,0.3599d0,0.3621d0,
     &  0.3643d0,0.3665d0,0.3686d0,0.3708d0,0.3729d0,
     &  0.3749d0,0.3770d0,0.3790d0,0.3810d0,0.3830d0,
     &  0.3849d0,0.3869d0,0.3888d0,0.3907d0,0.3925d0,
     &  0.3944d0,0.3962d0,0.3980d0,0.3997d0,0.4015d0,
     &  0.4032d0,0.4049d0,0.4066d0,0.4082d0,0.4099d0,
```

```
&   0.4115d0,0.4131d0,0.4147d0,0.4162d0,0.4177d0,
&   0.4192d0,0.4207d0,0.4222d0,0.4236d0,0.4251d0,
&   0.4265d0,0.4279d0,0.4292d0,0.4306d0,0.4319d0,
&   0.4332d0,0.4345d0,0.4357d0,0.4370d0,0.4382d0,
&   0.4394d0,0.4406d0,0.4418d0,0.4429d0,0.4441d0,
&   0.4452d0,0.4463d0,0.4474d0,0.4484d0,0.4495d0,
&   0.4505d0,0.4515d0,0.4525d0,0.4535d0,0.4545d0,
&   0.4554d0,0.4564d0,0.4573d0,0.4582d0,0.4591d0,
&   0.4599d0,0.4608d0,0.4616d0,0.4625d0,0.4633d0,
&   0.4641d0,0.4649d0,0.4656d0,0.4664d0,0.4671d0,
&   0.4678d0,0.4686d0,0.4693d0,0.4699d0,0.4706d0/
    data (fc(i),i=191,285)
& /0.4713d0,0.4719d0,0.4726d0,0.4732d0,0.4738d0,
&   0.4744d0,0.4750d0,0.4756d0,0.4761d0,0.4767d0,
&   0.4772d0,0.4778d0,0.4783d0,0.4788d0,0.4793d0,
&   0.4798d0,0.4803d0,0.4808d0,0.4812d0,0.4817d0,
&   0.4821d0,0.4826d0,0.4830d0,0.4834d0,0.4838d0,
&   0.4842d0,0.4846d0,0.4850d0,0.4854d0,0.4857d0,
&   0.4861d0,0.4864d0,0.4868d0,0.4871d0,0.4875d0,
&   0.4878d0,0.4881d0,0.4884d0,0.4887d0,0.4890d0,
&   0.4893d0,0.4896d0,0.4898d0,0.4901d0,0.4904d0,
&   0.4906d0,0.4909d0,0.4911d0,0.4913d0,0.4916d0,
&   0.4918d0,0.4920d0,0.4922d0,0.4925d0,0.4927d0,
&   0.4929d0,0.4931d0,0.4932d0,0.4934d0,0.4936d0,
&   0.4938d0,0.4940d0,0.4941d0,0.4943d0,0.4945d0,
&   0.4946d0,0.4948d0,0.4949d0,0.4951d0,0.4952d0,
&   0.4953d0,0.4955d0,0.4956d0,0.4957d0,0.4959d0,
&   0.4960d0,0.4961d0,0.4962d0,0.4963d0,0.4964d0,
&   0.4965d0,0.4966d0,0.4967d0,0.4968d0,0.4969d0,
&   0.4970d0,0.4971d0,0.4972d0,0.4973d0,0.4974d0,
&   0.4974d0,0.4975d0,0.4976d0,0.4977d0,0.4977d0/
    data (fc(i),i=286,380)
& /0.4978d0,0.4979d0,0.4979d0,0.4980d0,0.4981d0,
&   0.4981d0,0.4982d0,0.4982d0,0.4983d0,0.4984d0,
&   0.4984d0,0.4985d0,0.4985d0,0.4986d0,0.4986d0,
&   0.498650d0,0.498694d0,0.498736d0,0.498777d0,0.498817d0,
&   0.498856d0,0.498893d0,0.498930d0,0.498965d0,0.498999d0,
&   0.499032d0,0.499065d0,0.499096d0,0.499126d0,0.499155d0,
&   0.499184d0,0.499211d0,0.499238d0,0.499264d0,0.499289d0,
&   0.499313d0,0.499336d0,0.499359d0,0.499381d0,0.499402d0,
&   0.499423d0,0.499443d0,0.499462d0,0.499481d0,0.499499d0,
&   0.499517d0,0.499534d0,0.499550d0,0.499566d0,0.499581d0,
&   0.499596d0,0.499610d0,0.499624d0,0.499638d0,0.499651d0,
&   0.499663d0,0.499675d0,0.499687d0,0.499698d0,0.499709d0,
&   0.499720d0,0.499730d0,0.499740d0,0.499749d0,0.499758d0,
&   0.499767d0,0.499776d0,0.499784d0,0.499792d0,0.499800d0,
&   0.499807d0,0.499815d0,0.499822d0,0.499828d0,0.499835d0,
&   0.499841d0,0.499847d0,0.499853d0,0.499858d0,0.499864d0,
&   0.499869d0,0.499874d0,0.499879d0,0.499883d0,0.499888d0,
&   0.499892d0,0.499896d0,0.499900d0,0.499904d0,0.499908d0,
&   0.499912d0,0.499915d0,0.499918d0,0.499922d0,0.499925d0/
    data (fc(i),i=381,400)
& /0.499928d0,0.499931d0,0.499933d0,0.499936d0,0.499938d0,
&   0.499941d0,0.499943d0,0.499946d0,0.499948d0,0.499950d0,
&   0.499952d0,0.499954d0,0.499956d0,0.499958d0,0.499959d0,
&   0.499961d0,0.499963d0,0.499964d0,0.499966d0,0.499967d0/
    deltax=0.01d0
     rhonorm=(rho-rhomean)/dsigma
```



```
      if (rhonorm.ge.4.d0) then
        Fcumm=0.99999d0
        return
      else if (rhonorm.le.-4.d0) then
        Fcumm=0.00001d0
        return
      end if
      av=dabs(rhonorm)
      ilow=idnint(dint(100*av))+1
      ihi=ilow+1
      if (rhonorm.lt.0.d0) then
        Fcumm=0.5d0-fc(ihi)+(av-((ihi-1)*deltax))*
     &                (fc(ilow)-fc(ihi))/(deltax)
      else if (rhonorm.gt.0.d0) then
        Fcumm=0.5d0+fc(ilow)+(rhonorm-((ilow-1)*deltax))*
     &                (fc(ihi)-fc(ilow))/(deltax)
      else if (rhonorm.eq.0.d0) then
        Fcumm=0.5d0
      end if
       return
       end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
       function P(z,rho)                                               c
c                                                                      c
c       This function subroutine calculates the probability            c
c       density function as a function of height for                   c
c       a given particle size                                          c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
      common /two/ h,werupt0,airden,airvis,c,u
      common /three/ fdmin,fdmax,fdmean,hmin,hmax,xmin,xmax

c
c       The formulation of P(z) in Suzuki (1983) is incorrect. The
c       correct formulation should have Y(z)=Beta*W(z)/V0. This
c       subroutine has been corrected.
c       MSJ 5/7/97
c
       wz=werupt0*(1-z/hmax)
c       yz=beta*(wz-v0(rho))/v0(rho)
      yz=beta*wz/v0(rho)
c       y0=beta*(werupt0-v0(rho))/v0(rho0
      y0=beta*werupt0/v0(rho)
       anum= beta*werupt0*yz*dmyexp(-yz)
       denom= v0(rho)*hmax*(1.d0-(1.d0+y0)*dmyexp(-y0))
       P=anum/denom
c                                                                      c
c       The formula above, directly from Suzuki (1983) has the         c
c       possiblity of being negative for certain values of z.          c
c       Must set to zero if this is ever so.                           c
c                                                                      c
c       With the correction added above, this comment is no longer     c
c       accurate MSJ 5/7/97                                            c
c                                                                      c
       if (P.le.0.d0) then
```

```
          P=0.d0
        end if
            return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
        function v0(rho)
c                                                                       c
c        The argument d is the diameter of the particle in cm           c
c        v0 is the terminal fall velocity of particles at sea-level     c
c        given in Suzuki, 1983.                                         c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c        g : gravitational accelleration in cm/s**2
c
        implicit double precision (a-h,o-z)
        implicit integer (i-n)
        common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
        common /two/ h,werupt0,airden,airvis,c,u
        g=980.d0
        d=10.d0**rho
        term1=9.d0*airvis*fshape**(-0.32d0)
        term2=81.d0*airvis**2*fshape**(-.64d0)
        term3=1.5d0*airden*ashden(rho)*d**3*g*dsqrt(1.07d0-fshape)
        anum = ashden(rho)*g*d**2
        denom = term1+dsqrt(term2 + term3)
        v0 = anum/denom
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
        function ashden(rho)
c                                                                       c
c        This function calculates the contaminated ash density          c
c        as a function of rho                                           c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        implicit double precision (a-h,o-z)
        implicit integer (i-n)
        external f,FF
        common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
        common /eleven/ ashrholow,ashrhohi
        if (rho.ge.ashrhohi) then
          ashden=ashdenmin
        else if (rho.le.ashrholow) then
          ashden=ashdenmax
        else
          slope=(ashdenmin-ashdenmax)/(ashrhohi-ashrholow)
          yint=ashdenmax
          ashden=slope*(rho-ashrholow)+yint
        end if
c                                                                       c
c        Need to correct for any fuel that has been absorbed            c
c        by the ash                                                     c
c                                                                       c
        ashden=ashden*(1.d0+FF(rho))
        return
```



```
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
        subroutine inputdata
c                                                                       c
c        This subroutine reads the input data and also stochastically   c
c        calculates the important parameters                            c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        parameter (nmx=200, nmy=200)
        parameter (maxd= 200)
        implicit double precision (a-h,o-z)
        implicit integer (i-n)
        character*60 title
        common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
        common /two/ h,werupt0,airden,airvis,c,u
        common /three/ fdmin,fdmax,fdmean,hmin,hmax,xmin,xmax
        common /three1/ dmin,dmax,rhomin,rhomax,rhomean
        common /four/ ymin,ymax,acutoff
        common /five/ x,y,udir,frhomin,frhomax,frhomean,drho,cmax
        common /six/ numptsx,numptsy
        common /seven/ Uran
        common /eight/ rhocut
        common /nine/ v,icount
        common /eleven/ ashrholow,ashrhohi
        common /twelve/ power,tdur
        common /sixteen/ title
        dimension v(10000)
c                                                                       c
c        get input parameters from data file                            c
c                                                                       c
      open (unit=8,file='ashplume.in',status='old')
        read (8,111)          title
111        format(a60)
        read (8,*)            xmin,xmax
        read (8,*)            ymin,ymax
        read (8,*)            numptsx
        read (8,*)            numptsy

        read (8,*)            tlogmin,tlogmax
        read (8,*)            powlogmin,powlogmax
        read (8,*)            betalogmin,betalogmax
        read (8,*)            dmeanmin,dmeanmed,dmeanmax
        read (8,*)            dsigmamin,dsigmamax

        read (8,*)            ashdenmin,ashdenmax
        read (8,*)            ashrholow,ashrhohi
        read (8,*)            fshape
        read (8,*)            airden,airvis
        read (8,*)            c
        read (8,*)            dmax

        read (8,*)            fdmin, fdmean, fdmax
        read (8,*)            hmin
        read (8,*)            acutoff
        read (8,*)            rhocut
        read (8,*)            Uran
```

```
          frhomin=dlog10(fdmin)
          frhomax=dlog10(fdmax)
          frhomean=dlog10(fdmean)
        close (unit=8)
        s=1100.d0
c
c        Distributions
c
c        Power: loguniform
c        Duration:loguniform
c        beta: loguniform
c        vrad: deterministic
c        dmean: logtriangular
c        dsigma: loguniform
c        u: exponential with directionally varying lambda
c        udir: quasi-uniform
c
        rnd2=v(icount)
        icount=icount+1
        tlog=tlogmin+(tlogmax-tlogmin)*rnd2
        tdur=10.d0**tlog
c
c         tdur in sec
c
       rnd3=v(icount)
        icount=icount+1
        plog=powlogmin+(powlogmax-powlogmin)*rnd3
        power=10.d0**plog
        if (power.gt.7.d13) then
          power=7.d13
        end if
        hmax=(8.2d0*power**0.25)/1000.d0
        qdot=(hmax/0.24)**4.d0
        q=qdot*tdur*1.0d3
c
c        wind direction-quasi-uniform using SCP 5,000 ft data
c
        call winddirect(udir)
c
c        wind speed-exponential with direction dependent lambda
c
        call windspeed (udir,u)
c
c        dmean-logtriangular
c        dsigma-loguniform
c
        rnd5=v(icount)
        icount=icount+1
        call triangular(rhomean,dmeanmin,dmeanmed,dmeanmax)
             dmean=10.d0**rhomean
c
c      dsigma is sampled uniformly between min and max
c
             dsigma= dsigmamin+rnd5*(dsigmamax-dsigmamin)
c
c        beta-loguniform
c
             rnd6=v(icount)
```



```
             icount=icount+1
             betalog=betalogmin+(betalogmax-betalogmin)*rnd6
             beta=10.d0**betalog
             rhomin=rhomean-5.d0*dsigma
             rhomax=rhomean+5.d0*dsigma
             rlog=-2.069d0+2.d0*dlog10(ashden(rhomean))+0.27435d0*
          &          dlog10(qdot*1.d3)
             vrad=10.d0**rlog
             werupt0=qdot*1.d3/(ashden(rhomean)*3.141592d0*(vrad*1.d2)**2)
             return
             end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
             subroutine histf(rhomin,cmax,x,y,ivol,frhomin,frhomax,frhomean)
c
c             This subroutine is "in charge" of calculating the fuel
c             particulate size distribution at the dose point
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
             implicit double precision (a-h,o-z)
             implicit integer (i-n)
             character*19 out
             character*4 ixx,iyy,ivv
             external FFintegrandhist
             common /eight/ rhocut
             common /thirteen/ numapts
             ixloc=int(x)
             iyloc=int(y)
             write(ixx,300) ixloc
             write(iyy,300) iyloc
             write(ivv,300) ivol
300             format(i4)
             call filename('fuel',x,y,ivol,out)
             open(unit=22,file=out,status='unknown')
             write(22,301)'rho-f','fuel mass (g)'
301             format(a12,a15)
             do ii=1,numapts+1
               rhof=frhomin+dfloat(ii-1)/(numapts-1)*(frhomax-frhomin)
               dmina=rhof+rhocut
               if (dmina.lt.rhomin) then
                 dmina=rhomin
               end if
               call qromb4(FFintegrandhist,dmina,cmax,ans,rhof,x,y,ivol)
               blah=rhof-rhocut
               write(22,302) blah,ans
302               format(f12.4,e15.5)
             end do
             close (unit=22)
             return
             end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
             function FFintegrandhist(rhoa,rhof,x,y,ivol)
c
c             This function is the integrand for calculating particle size
c             distributions at the dose points
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /one/ beta,q,ashdenmin,ashdenmax,dmean,dsigma,fshape
      common /seven/ Uran
      common /eight/ rhocut
      external dma
      if (rhoa.ge.rhof+rhocut) then
        FFintegrandhist=Uran/q*dm(rhof)/
     &          (1.d0-Fcumm(rhof+rhocut))*dma(rhoa,x,y,ivol)
      else
        FFintegrandhist=0.d0
      end if
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
      function dma(rhoa,x,y,ivol)
c                                                                      c
c      This function reads in the ash deposition at the dose point     c
c      currently being examined                                        c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      character*19 infile
      character*12 blah1
      character*15 blah2
      character*4 ix,iy,iv
      dimension dr(31),dd(31)
      common /thirteen/ numapts
      ixloc=int(x)
      iyloc=int(y)
      write(ix,300) ixloc
      write(iy,300) iyloc
      write(iv,300) ivol
300      format(i4)
      call filename('ashh',x,y,ivol,infile)
      open (unit=30,file=infile,status='unknown')
      read (30,299) blah1,blah2
299      format (a12,a15)
      do i=1,numapts
        read(30,301) dr(i),dd(i)
301        format(f12.4,e15.5)
      end do
      do j=1,numapts
        if (rhoa.ge.dr(j).and.rhoa.le.dr(j+1)) then
          dma=dd(j)+(rhoa-dr(j))/(dr(j+1)-dr(j))*
     &              (dd(j+1)-dd(j))
          close (unit=30)
          return
        end if
      end do
      close (unit=30)
      dma=0.d0
      return
      end

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```



```
c                                                                      c
      subroutine triangular(x,xl,xm,xh)
c                                                                      c
c       This subroutine returns as x a variable sampled from a         c
c       logtriangular distribution with lower bound, mode and          c
c       upperbound of xl,xm,xh respectively                            c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /nine/ v,icount
      dimension v(10000)
      rnd4=v(icount)
      icount=icount+1
      zetah=(xm-xl)/(xh-xl)
      dk1=2.d0/(xh-xl)/(xm-xl)
      dk2=-2.d0/(xh-xl)/(xh-xm)
      if (rnd4.le.zetah) then
        x=xl+dsqrt(2.d0*rnd4/dk1)
      else if (rnd4.gt.zetah) then
        a=0.5d0*dk2
        b=-dk2*xm+dk1*(xm-xl)
        c=0.5d0*dk2*xm**2+dk1*xm*(xl-xm)-rnd4+0.5d0*
     &                          dk1*(xm-xl)**2
        x=(-b+dsqrt(b**2-4.d0*a*c))/(2.d0*a)
      end if
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
      subroutine winddirect(udir)
c                                                                      c
c       This subroutine selects the wind direction as                  c
c       described in Jarzemba (1996)                                   c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      common /nine/ v,icount
      dimension cutoff(17),angle(17),v(10000)
      data cutoff/0.017804d0,0.053412d0,0.136498d0,0.267058d0,
     &            0.376848d0,0.430260d0,0.460923d0,0.482684d0,
     &            0.496532d0,0.523238d0,0.563792d0,0.635998d0,
     &            0.776448d0,0.887228d0,0.953499d0,0.985151d0,
     &            1.000000/
          data angle/0.d0,22.5d0,45.d0,67.5d0,90.d0,112.5d0,135.d0,
     &            157.5d0,180.d0,-22.5d0,-45.d0,-67.5d0,-90.d0,
     &            -112.5d0,-135.d0,-157.5d0,1000.d0/
      rnd=v(icount)
      icount=icount+1
      do i=1,17
              if (rnd.le.cutoff(i)) then
                      udir=angle(i)
                      go to 10
              end if
      end do
c                                                                      c
c       This statement sets the wind direction to -90 degrees          c
```

```
c          for all realizations                                          c
c                                                                        c
10       continue
         return
          end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                        c
         subroutine windspeed(udir,u)
c                                                                        c
c          This subroutine selects the wind speed u given a              c
c          wind direction udir as described                              c
c          in Jarzemba (1996)                                            c
c                                                                        c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c
         implicit double precision (a-h,o-z)
         implicit integer (i-n)
         common /nine/ v,icount
         dimension angle(17),ula(17),v(10000)
         data angle/0.d0,22.5d0,45.d0,67.5d0,90.d0,112.5d0,135.d0,
     &                   157.5d0,180.d0,-22.5d0,-45.d0,-67.5d0,-90.d0,
     &                   -112.5d0,-135.d0,-157.5d0,1000.d0/
              data ula/0.313d0,0.222d0,0.149d0,0.139d0,0.156d0,
     &                   0.217d0,0.323d0,0.4d0,0.417d0,0.244d0,
     &                   0.213d0,0.189d0,0.172d0,0.185d0,0.208d0,
     &                   0.294d0,99999.d0/
         do i=1,17
                  if (udir.eq.angle(i)) then
                           ulambda=ula(i)
                  end if
         end do
         if (udir.eq.1000.d0) then
                  udir=0.d0
                  u=0.d0
         end if
         rnd=v(icount)
         icount=icount+1
         u=-dlog(1-rnd)/ulambda
         u=u*100.d0
         return
         end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                        c
         subroutine qromb4(func,a,b,s,rhof,x,y,ivol)
c                                                                        c
c          This subroutine is used to calculate the fuel particulate     c
c          distribution with size at the dose point.                     c
c          It is origially from Numerical Recipes.                       c
c                                                                        c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         implicit double precision (a-h,o-z)
         implicit integer (i-n)
         parameter (eps=30.d-4,jmax=22,jmaxp=jmax+1,k=5,km=k-1)
         external func
         dimension ss(jmaxp),h(jmaxp)
         h(1)=1.d0
```



```
         it=0
         do j=1,jmax
           call trapzd4(func,a,b,ss(j),j,it,rhof,x,y,ivol)
           if (j.ge.k) then
             call polint4(h(j-km),ss(j-km),k,0.d0,s,dss)
             if (dabs(dss).le.eps*dabs(s)) then
               return
               end if
           end if
           ss(j+1)=ss(j)
           h(j+1)=0.25d0*h(j)
         end do
         write(6,*) 'integration surpassed iteration limit'
         write(6,*) 'for calculating fuel particluate'
         write(6,*) 'distribution at the dose point.'
         write(6,*) 'Accuracy of fuel distribution'
         write(6,*) 'cannot be gauranteed.'
         return
         end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                        c
         subroutine trapzd4(func,a,b,s,n,it,rhof,x,y,ivol)
c                                                                        c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         implicit double precision (a-h,o-z)
         implicit integer (i-n)
         external func
         if (n.eq.1) then
                  s=0.5d0*(b-a)*(func(a,rhof,x,y,ivol)+
     &                           func(b,rhof,x,y,ivol))
                  it=1
         else
                  itnm=it
                  del=(b-a)/itnm
                  xx=a+0.5d0*del
                  sum=0.d0
                  do j=1,it
                           sum=sum+func(xx,rhof,x,y,ivol)
                           xx=xx+del
                  end do
                  s=0.5d0*(s+(b-a)*sum/itnm)
                  it=2*it
         end if
         return
         end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                        c
         subroutine polint4(xa,ya,n,x,y,dy)
c                                                                        c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         implicit double precision (a-h,o-z)
         implicit integer (i-n)
         parameter (nmax=10)
         dimension xa(n),ya(n),c(nmax),d(nmax)
         ns=1
         dif=dabs(x-xa(1))
         do i=1,n
           dift=dabs(x-xa(i))
```

```
      if (dift.lt.dif)then
        ns=i
        dif=dift
      end if
      c(i)=ya(i)
      d(i)=ya(i)
    end do
    y=ya(ns)
    ns=ns-1
    do m=1,n-1
      do i=1,n-m
        ho=xa(i)-x
        hp=xa(i+m)-x
        w=c(i+1)-d(i)
        den=ho-hp
        if (den.eq.0.d0) then
          write(6,*) 'error in polint4'
          return
        end if
        den=w/den
        d(i)=hp*den
        c(i)=ho*den
      end do
      if (2*ns.lt.n-m) then
        dy=c(ns+1)
      else
        dy=d(ns)
        ns=ns-1
      end if
      y=y+dy
    end do
    return
    end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      subroutine qromb3(func,a,b,s)
c                                                                       c
c       This subroutine is used to calculate the fuel fraction         c
c       of the ash as a function of rho                                 c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      parameter (eps=10.d-4,jmax=20,jmaxp=jmax+1,k=5,km=k-1)
      external func
      dimension ss(jmaxp),h(jmaxp)
      h(1)=1.d0
      it=0
      do j=1,jmax
        call trapzd3(func,a,b,ss(j),j,it)
        if (j.ge.k) then
          call polint3(h(j-km),ss(j-km),k,0.d0,s,dss)
          if (dabs(dss).le.eps*dabs(s)) then
            return
            end if
        end if
        ss(j+1)=ss(j)
        h(j+1)=0.25d0*h(j)
      end do
```



```
      write(6,*) 'integration surpassed iteration limit'
      write(6,*) 'for calculating fuel fraction function.'
      write(6,*) 'Accuracy cannot be gauranteed.'
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      subroutine trapzd3(func,a,b,s,n,it)
c                                                                       c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      external func
      if (n.eq.1) then
        s=0.5d0*(b-a)*(func(a)+func(b))
        it=1
      else
        itnm=it
        del=(b-a)/itnm
        xx=a+0.5d0*del
        sum=0.d0
        do j=1,it
          sum=sum+func(xx)
          xx=xx+del
        end do
        s=0.5d0*(s+(b-a)*sum/itnm)
        it=2*it
      end if
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      subroutine polint3(xa,ya,n,x,y,dy)
c                                                                       c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      parameter (nmax=10)
      dimension xa(n),ya(n),c(nmax),d(nmax)
      ns=1
      dif=dabs(x-xa(1))
      do i=1,n
        dift=dabs(x-xa(i))
        if (dift.lt.dif)then
          ns=i
          dif=dift
        end if
        c(i)=ya(i)
        d(i)=ya(i)
      end do
      y=ya(ns)
      ns=ns-1
      do m=1,n-1
        do i=1,n-m
          ho=xa(i)-x
          hp=xa(i+m)-x
          w=c(i+1)-d(i)
          den=ho-hp
```

```
            if (den.eq.0.d0) then
              write(6,*) 'error in polint3'
              return
            end if
            den=w/den
            d(i)=hp*den
            c(i)=ho*den
          end do
          if (2*ns.lt.n-m) then
            dy=c(ns+1)
          else
            dy=d(ns)
            ns=ns-1
          end if
          y=y+dy
        end do
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
        subroutine qromb2(func,a,b,s)
c                                                                       c
c       This subroutine is used to calculate the outer integral for     c
c       the fuel distribution at the point of interest                  c
c                                                                       c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        implicit double precision (a-h,o-z)
        implicit integer (i-n)
c                                                                       c
c          Recommend using eps=1.d-4 MSJ 5/7/97                         c
c                                                                       c
        parameter (eps=1.d-4,jmax=22,jmaxp=jmax+1,k=5,km=k-1)
        external func
        dimension ss(jmaxp),h(jmaxp)
        h(1)=1.d0
        it=0
        do j=1,jmax
          call trapzd2(func,a,b,ss(j),j,it)
          if (j.ge.k) then
            call polint2(h(j-km),ss(j-km),k,0.d0,s,dss)
            if (dabs(dss).le.eps*dabs(s)) then
              return
            end if
          end if
          ss(j+1)=ss(j)
          h(j+1)=0.25d0*h(j)
        end do
         write(6,*) 'Integration surpassed iteration limit'
        write(6,*) 'for calculating outer integral for fuel'
        write(6,*) 'areal density at the dose point.'
        write(6,*) 'Accuracy cannot be gauranteed.'
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
        subroutine trapzd2(func,a,b,s,n,it)
c                                                                       c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```



```
        implicit double precision (a-h,o-z)
        implicit integer (i-n)
        external func
        if (n.eq.1) then
                s=0.5d0*(b-a)*(func(a)+func(b))
                it=1
        else
                itnm=it
                del=(b-a)/itnm
                xx=a+0.5d0*del
                sum=0.d0
                do j=1,it
                        sum=sum+func(xx)
                        xx=xx+del
                end do
                s=0.5d0*(s+(b-a)*sum/itnm)
                it=2*it
        end if
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
        subroutine polint2(xa,ya,n,x,y,dy)
c                                                                       c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        implicit double precision (a-h,o-z)
        implicit integer (i-n)
        parameter (nmax=10)
        dimension xa(n),ya(n),c(nmax),d(nmax)
        ns=1
        dif=dabs(x-xa(1))
        do i=1,n
          dift=dabs(x-xa(i))
          if (dift.lt.dif)then
            ns=i
            dif=dift
          end if
          c(i)=ya(i)
          d(i)=ya(i)
        end do
        y=ya(ns)
        ns=ns-1
        do m=1,n-1
          do i=1,n-m
            ho=xa(i)-x
            hp=xa(i+m)-x
            w=c(i+1)-d(i)
            den=ho-hp
            if (den.eq.0.d0) then
              write(6,*) 'error in polint2'
              return
            end if
            den=w/den
            d(i)=hp*den
            c(i)=ho*den
          end do
          if (2*ns.lt.n-m) then
            dy=c(ns+1)
```

```
          else
            dy=d(ns)
            ns=ns-1
          end if
          y=y+dy
        end do
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
        subroutine qromb1(func,a,b,s,xloc,yloc,ivol,numapts)
c                                                                       c
c       This subroutine is used to calculate the outer integral of      c
c       the ash distribution at the point of interest.                  c
c       It has additions in it to allow                                 c
c       for outputting the incremental the results for subsequent use   c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        implicit double precision (a-h,o-z)
        implicit integer (i-n)
        character*19 outfile
c                                                                       c
c         eps=1.d-3 is generally gaurantees maximal accuracy            c
c                                                                       c
c                                                                       c
c          Recommend using eps=1.d-4 MSJ 5/7/97                         c
c                                                                       c
        parameter (eps=1.d-4,jmax=22,jmaxp=jmax+1,k=5,km=k-1)
        external func
         common /four/ ymin,ymax,acutoff
        common /fifteen/ ipchar
        dimension ss(jmaxp),h(jmaxp)
        call filename('ashh',xloc,yloc,ivol,outfile)
        if (ipchar.eq.1) then
          open (unit=20,file=outfile,status='unknown')
          write (20,200)' rho ',' ash mass (g) '
200       format (a12,a15)
          do nn=1,numapts
            arho=a+dfloat(nn-1)/(numapts-1)*(b-a)
             if (func(arho).gt.acutoff) then
              write (20,201) arho,func(arho)
             else
               write (20,201) arho,0.d0
             end if
201          format(f12.5,e15.5)
          end do
          write(20,*)'EOF'
           close(unit=20)
        end if
        h(1)=1.d0
        it=0
        do j=1,jmax
          call trapzd1(func,a,b,ss(j),j,it)
          if (j.ge.k) then
            call polint1(h(j-km),ss(j-km),k,0.d0,s,dss)
            if (dabs(dss).le.eps*dabs(s)) then
             return
            end if
```



```
          end if
          ss(j+1)=ss(j)
          h(j+1)=0.25d0*h(j)
        end do
         write(6,*) 'Integration surpassed iteration limit'
        write(6,*) 'for calculating ash areal'
        write(6,*) 'density at the dose point.'
        write(6,*) 'Accuracy cannot be gauranteed'
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
        subroutine trapzd1(func,a,b,s,n,it)
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        implicit double precision (a-h,o-z)
        implicit integer (i-n)
        external func
        if (n.eq.1) then
                s=0.5d0*(b-a)*(func(a)+func(b))
                it=1
        else
                itnm=it
                del=(b-a)/itnm
                xx=a+0.5d0*del
                sum=0.d0
                do j=1,it
                        sum=sum+func(xx)
                        xx=xx+del
                end do
                s=0.5d0*(s+(b-a)*sum/itnm)
                it=2*it
        end if
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
        subroutine polint1(xa,ya,n,x,y,dy)
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        implicit double precision (a-h,o-z)
        implicit integer (i-n)
        parameter (nmax=10)
        dimension xa(n),ya(n),c(nmax),d(nmax)
        ns=1
        dif=dabs(x-xa(1))
        do i=1,n
          dift=dabs(x-xa(i))
          if (dift.lt.dif)then
            ns=i
            dif=dift
          end if
          c(i)=ya(i)
          d(i)=ya(i)
        end do
        y=ya(ns)
        ns=ns-1
        do m=1,n-1
```

```
            do i=1,n-m
              ho=xa(i)-x
              hp=xa(i+m)-x
              w=c(i+1)-d(i)
              den=ho-hp
              if (den.eq.0.d0) then
                write(6,*) 'error in polint1'
                return
              end if
              den=w/den
              d(i)=hp*den
              c(i)=ho*den
            end do
            if (2*ns.lt.n-m) then
              dy=c(ns+1)
            else
              dy=d(ns)
              ns=ns-1
            end if
            y=y+dy
          end do
          return
          end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
          subroutine filename(type,x,y,ivol,file)
c                                                                       c
c         This subroutine "calculates" output very specific filenames   c
c         for storing particle size distribution information at the     c
c         dose points.                                                  c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
          implicit double precision (a-h,o-z)
          implicit integer (i-n)
          character*19 file
          character*4 type,ix,iy,iv
          ixloc=int(x)
          iyloc=int(y)
c                                                                       c
c         Calculate x location character string                         c
c                                                                       c
          if(iabs(ixloc).ge.0.and.iabs(ixloc).lt.10.and.
     &                                                 ixloc.lt.0) then
            write(ix,300)'m','00',iabs(ixloc)
300            format(a1,a2,i1)
          else if(iabs(ixloc).ge.0.and.iabs(ixloc).lt.10.and.
     &                                                 ixloc.ge.0) then
            write(ix,301)'p','00',ixloc
301            format(a1,a2,i1)
          else if(iabs(ixloc).ge.10.and.iabs(ixloc).lt.100.and.
     &                                                 ixloc.lt.0) then
            write(ix,310)'m','0',iabs(ixloc)
310            format(a1,a1,i2)
          else if(iabs(ixloc).ge.10.and.iabs(ixloc).lt.100.and.
     &                                                 ixloc.ge.0) then
            write(ix,311)'p','0',ixloc
311            format(a1,a1,i2)
          else if(iabs(ixloc).ge.100.and.iabs(ixloc).lt.1000.and.
```



```
     &                                                 ixloc.lt.0) then
            write(ix,320)'m',iabs(ixloc)
320            format(a1,i3)
          else if(iabs(ixloc).ge.100.and.iabs(ixloc).lt.1000.and.
     &                                                 ixloc.ge.0) then
            write(ix,321)'p','0',ixloc
321            format(a1,i3)
          end if
c                                                                       c
c         Calculate y location character string                         c
c                                                                       c
          if(iabs(iyloc).ge.0.and.iabs(iyloc).lt.10.and.
     &                                                 iyloc.lt.0) then
            write(iy,330)'m','00',iabs(iyloc)
330            format(a1,a2,i1)
          else if(iabs(iyloc).ge.0.and.iabs(iyloc).lt.10.and.
     &                                                 iyloc.ge.0) then
            write(iy,331)'p','00',iyloc
331            format(a1,a2,i1)
          else if(iabs(iyloc).ge.10.and.iabs(iyloc).lt.100.and.
     &                                                 iyloc.lt.0) then
            write(iy,340)'m','0',iabs(iyloc)
340            format(a1,a1,i2)
          else if(iabs(iyloc).ge.10.and.iabs(iyloc).lt.100.and.
     &                                                 iyloc.ge.0) then
            write(iy,341)'p','0',iyloc
341            format(a1,a1,i2)
          else if(iabs(iyloc).ge.100.and.iabs(iyloc).lt.1000.and.
     &                                                 iyloc.lt.0) then
            write(iy,350)'m',iabs(iyloc)
350            format(a1,i3)
          else if(iabs(iyloc).ge.100.and.iabs(iyloc).lt.1000.and.
     &                                                 iyloc.ge.0) then
            write(iy,351)'p','0',iyloc
351            format(a1,i3)
          end if
c                                                                       c
c         Calculate volcano number character string                     c
c                                                                       c
          if(iabs(ivol).ge.0.and.iabs(ivol).lt.10) then
            write(iv,360)'000',iabs(ivol)
360            format(a3,i1)
          else if(iabs(ivol).ge.10.and.iabs(ivol).lt.100) then
            write(iv,370)'00',iabs(ivol)
370            format(a2,i2)
          else if(iabs(ivol).ge.100.and.iabs(ivol).lt.1000) then
            write(iv,380)'0',iabs(ivol)
380            format(a1,i3)
          else if(iabs(ivol).ge.1000.and.iabs(ivol).lt.10000) then
            write(iv,390) ivol
390            format(i4)
          else
            write(6,*)'ivol=',ivol
            write(6,*) 'error in filename'
            pause
            stop
          end if
          file=type//'x'//ix//'y'//iy//'v'//iv
```

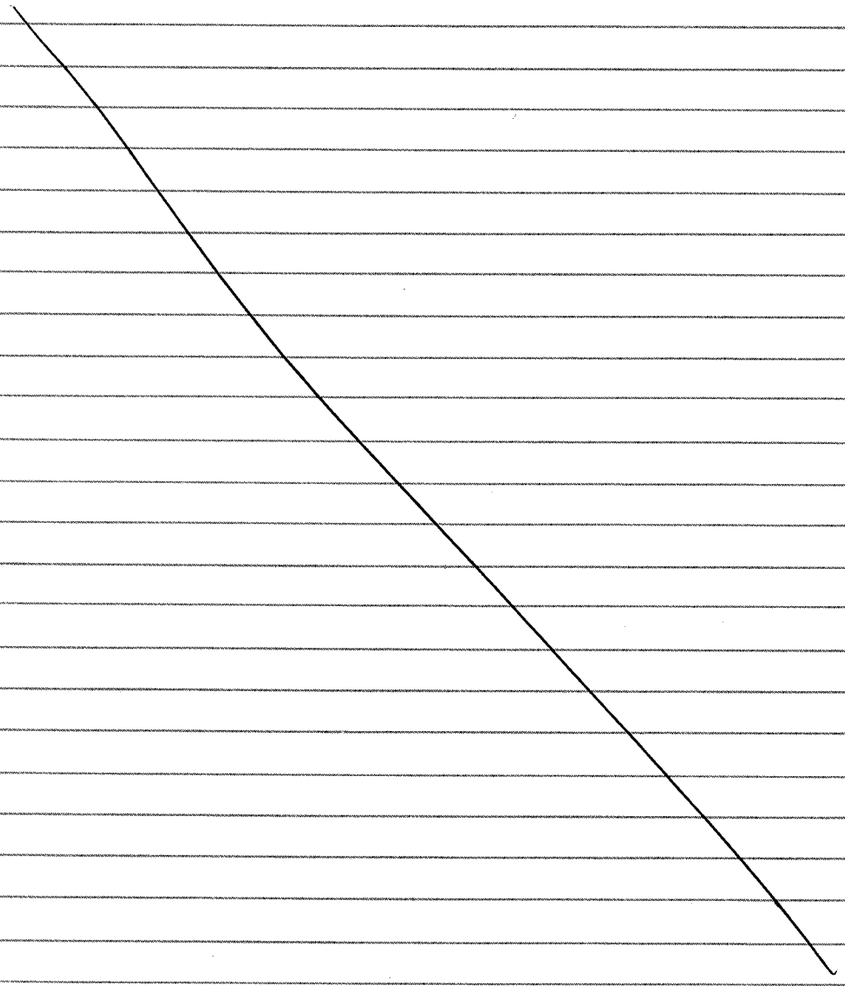
```
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine qgaus(func,a,b,ss,rho)
c
c       This subroutine uses n-point gauss-legendre integration
c       to do the inner integral for both ash and fuel depositions.
c       It uses gauleg to calculate the abscissas and weights
c       for the integration.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       implicit double precision (a-h,o-z)
      implicit integer (i-n)
c
c       n=250 generally gaurantees @1% accuracy
c       n=400 generally gaurantees maximum accuracy
c
c       Recommend using n=20 MSJ 5/7/97
c
       parameter (n=20)
       external func
       dimension x(n),w(n)
      call gauleg(0.d0,1.d0,x,w,n)
      xm=0.5d0*(b+a)
      xr=0.5d0*(b-a)
      ss=0.d0
      do 11 j=1,n
         dx=xr*x(j)
         ss=ss+w(j)*(func(xm+dx,rho)+func(xm-dx,rho))
11    continue
      ss=xr*ss
       return
       end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine gauleg(x1,x2,x,w,n)
c
c       This subroutine is called by gauleg to calculate the
c       abscissas and weights of for the gauss-legendre integration.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       implicit double precision (a-h,o-z)
      implicit integer (i-n)
       parameter (eps=3.d-14)
       dimension x(n),w(n)
       m=(n+1)/2
       xm=0.5d0*(x1+x2)
       xl=0.5d0*(x2-x1)
       do 12 i=1,m
         z=cos(dacos(-1.d0)*(i-0.25d0)/(n+0.5d0))
1        continue
         p1=1.d0
         p2=0.d0
         do 11 j=1,n
           p3=p2
           p2=p1
           p1=((2.d0*j-1.d0)*z*p2-(j-1.d0)*p3)/j
```



```
11          continue
          pp=n*(z*p1-p2)/(z*z-1.d0)
         z1=z
          z=z1-p1/pp
         if (dabs(z-z1).gt.eps) then
           go to 1
          end if
          x(i)=xm-xl*z
          x(n+1-i)=xm+xl*z
          w(i)=2.d0*xl/((1.d0-z*z)*pp*pp)
          w(n+1-i)=w(i)
12        continue
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
      subroutine rand(iseed,v,max)
c                                                                       c
c       This subroutine returns a string of uniformly distributed       c
c       homogenously distributed random numbers between 0 and 1.        c
c       The string is of length max and is held in the array v.         c
c       This routine was given to me by Randy Manteufel                 c
c                                                                       c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      dimension v(max)
      data ia / 1021      /
      data m  / 1048576   /
      data am / 1048576.d0 /
      do 100 i = 1, max
        iseed = mod( iseed * ia, m )
        v(i) = dble( iseed ) / am
100   continue
      return
      end
```

5/23/97 MSJ Similar units checks are required for DCAGS and DCAGW. Rather than re-printout these modules, these checks are being performed on the printouts on p. 11 (DCAGS) and p. 16 (DCAGW) in red ink.

5/27/97 MSJ Similar units check required for ASHRMOVO.f module is performed on the following printout.




```
c=======================================================================
      subroutine ashrmovo(gramsfpercm2,
     &                    nt,time,
     &                    ciperm2)
c=======================================================================
c by M.S. Jarzemba and R.D. Manteufel, March 1, 1997
c
c Re-checked by MSJ on 5/27/97. Units labels on Kd's were changed to cm3/g.
c This is just a mis-label, not an error.
c
c       Explanation of variables
c
c inputs:
c
c       gramsfpercm2    Array of g/cm2 of fuel deposition at dose points
c       nt              The number of time points
c       time            Array of times for dose calculation
c
c outputs:
c
c       ciperm2 Array of nuclide surface concentration in Ci/m^2
c                       as a function of time and nuclide
c
c       toe             The time of event in yr
c       dlbr            Blanket removal rate from erosion (1/yr)
c       sol             Array of nuclide solubilities (moles/m3)
c       dkd             Array of soil kd's in cm3/g
c       fpe     fraction of precipitation lost to evapotranspiration
c       fpsat   Fraction of year soil is saturated due to precipitation
c       fie     Fraction of irrigation lost to evapotranspiration
c       fisat   Fraction of year soil is saturated due to irrigation
c       precip  Annual precipitation in m/yr
c       dirr    Annual irrigation in m/yr
c       depthsoil  Depth of the rooting zone in m
c       rhosoil Bulk density of the soil in g/cm3
c       theta   Saturation fraction of soil

      implicit double precision (a-h,o-z)
      implicit integer (i-n)
      character*60 name
      include 'maxntime.i'
      parameter(maxnnucsvol = 43 )
      dimension sol(maxnnucsvol),dkd(maxnnucsvol)
      dimension dlt(maxnnucsvol),dlr(maxnnucsvol),dll(maxnnucsvol)
      dimension dn(maxntime,maxnnucsvol)
      dimension tempdnt1(maxnnucsvol),tempdn(maxnnucsvol)
      dimension ciperm2(maxntime,maxnnucsvol)
      dimension time(maxntime)
      external ispquery
      external valuesp
      external halflifeperiso

      save ikey
      save itime
      save idlbr
      save ifpe
      save ifie
      save ifpsat
```

```
save ifisat
save iprecip
save idirr
save irhosoil
save itheta
save idepthsoil
save ikduranium
save ikdcurium
save ikdplutonium
save ikdamericium
save ikdthorium
save ikdradium
save ikdlead
save ikdprotactinium
save ikdactinium
save ikdneptunium
save ikdsamarium
save ikdcesium
save ikdiodine
save ikdtin
save ikdsilver
save ikdpaladium
save ikdtechnetium
save ikdmolybdenum
save ikdniobium
save ikdzirconium
save ikdstrontium
save ikdselenium
save ikdnickel
save ikdchlorine
save ikdcarbon
save isoluranium
save isolcurium
save isolplutonium
save isolamericium
save isolthorium
save isolradium
save isollead
save isolprotactinium
save isolactinium
save isolneptunium
save isolsamarium
save isolcesium
save isoliodine
save isoltin
save isolsilver
save isolpaladium
save isoltechnetium
save isolmolybdenum
save isolniobium
save isolzirconium
save isolstrontium
save isolselenium
save isolnickel
save isolchlorine
save isolcarbon

if( ikey .ne. 23567 ) then
```



```
call clearchar( 60, name )
name = 'TimeOfNextVolcanicEventinRegionOfInterest[yr]'
itime = ispquery( name )

call clearchar( 60, name )
name = 'RelativeRateOfBlanketRemoval[1/yr]'
idlbr = ispquery( name )

call clearchar( 60, name )
name = 'FractionOfPrecipitationLostToEvapotranspiration'
ifpe = ispquery( name )

call clearchar( 60, name )
name = 'FractionOfIrrigationLostToEvapotranspiration'
ifie = ispquery( name )

call clearchar( 60, name )
name = 'FractionOfYearSoilIsSaturatedDueToPrecipitation'
ifpsat = ispquery( name )

call clearchar( 60, name )
name = 'FractionOfYearSoilIsSaturatedDueToIrrigation'
ifisat = ispquery( name )

call clearchar( 60, name )
name = 'AnnualPrecipitation[m/yr]'
iprecip = ispquery( name )

call clearchar( 60, name )
name = 'AnnualIrrigation[m/yr]'
idirr = ispquery( name )

call clearchar( 60, name )
name = 'AshBulkDensity[g/cm3]'
irhosoil = ispquery( name )

call clearchar( 60, name )
name = 'AshVolumetricMoistureFractionAtSaturation'
itheta = ispquery( name )

call clearchar( 60, name )
name = 'DepthOfTheRootingZone[m]'
idepthsoil = ispquery( name )

call clearchar( 60, name )
name = 'KdOfUraniumInVolcanicAsh[cm3/g]'
ikduranium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfCuriumInVolcanicAsh[cm3/g]'
ikdcurium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfPlutoniumInVolcanicAsh[cm3/g]'
ikdplutonium = ispquery( name )

call clearchar( 60, name )
```

```
name = 'KdOfAmericiumInVolcanicAsh[cm3/g]'
ikdamericium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfThoriumInVolcanicAsh[cm3/g]'
ikdthorium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfRadiumInVolcanicAsh[cm3/g]'
ikdradium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfLeadInVolcanicAsh[cm3/g]'
ikdlead = ispquery( name )

call clearchar( 60, name )
name = 'KdOfProtactiniumInVolcanicAsh[cm3/g]'
ikdprotactinium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfActiniumInVolcanicAsh[cm3/g]'
ikdactinium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfNeptuniumInVolcanicAsh[cm3/g]'
ikdneptunium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfSamariumInVolcanicAsh[cm3/g]'
ikdsamarium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfCesiumInVolcanicAsh[cm3/g]'
ikdcesium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfIodineInVolcanicAsh[cm3/g]'
ikdiodine = ispquery( name )

call clearchar( 60, name )
name = 'KdOfTinInVolcanicAsh[cm3/g]'
ikdtin = ispquery( name )

call clearchar( 60, name )
name = 'KdOfSilverInVolcanicAsh[cm3/g]'
ikdsilver = ispquery( name )

call clearchar( 60, name )
name = 'KdOfPaladiumInVolcanicAsh[cm3/g]'
ikdpaladium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfTechnetiumInVolcanicAsh[cm3/g]'
ikdtechnetium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfMolybdenumInVolcanicAsh[cm3/g]'
ikdmolybdenum = ispquery( name )
```



```
call clearchar( 60, name )
name = 'KdOfNiobiumInVolcanicAsh[cm3/g]'
ikdniobium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfZirconiumInVolcanicAsh[cm3/g]'
ikdzirconium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfStrontiumInVolcanicAsh[cm3/g]'
ikdstrontium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfSeleniumInVolcanicAsh[cm3/g]'
ikdselenium = ispquery( name )

call clearchar( 60, name )
name = 'KdOfNickelInVolcanicAsh[cm3/g]'
ikdnickel = ispquery( name )

call clearchar( 60, name )
name = 'KdOfChlorineInVolcanicAsh[cm3/g]'
ikdchlorine = ispquery( name )

call clearchar( 60, name )
name = 'KdOfCarbonInVolcanicAsh[cm3/g]'
ikdcarbon = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfUraniumInVolcanicAsh[moles/liter]'
isoluranium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfCuriumInVolcanicAsh[moles/liter]'
isolcurium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfPlutoniumInVolcanicAsh[moles/liter]'
isolplutonium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfAmericiumInVolcanicAsh[moles/liter]'
isolamericium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfThoriumInVolcanicAsh[moles/liter]'
isolthorium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfRadiumInVolcanicAsh[moles/liter]'
isolradium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfLeadInVolcanicAsh[moles/liter]'
isollead = ispquery( name )

call clearchar( 60, name )
```

```
name = 'SolubilityOfProtactiniumInVolcanicAsh[moles/liter]'
isolprotactinium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfActiniumInVolcanicAsh[moles/liter]'
isolactinium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfNeptuniumInVolcanicAsh[moles/liter]'
isolneptunium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfSamariumInVolcanicAsh[moles/liter]'
isolsamarium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfCesiumInVolcanicAsh[moles/liter]'
isolcesium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfIodineInVolcanicAsh[moles/liter]'
isoliodine = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfTinInVolcanicAsh[moles/liter]'
isoltin = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfSilverInVolcanicAsh[moles/liter]'
isolsilver = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfPaladiumInVolcanicAsh[moles/liter]'
isolpaladium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfTechnetiumInVolcanicAsh[moles/liter]'
isoltechnetium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfMolybdenumInVolcanicAsh[moles/liter]'
isolmolybdenum = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfNiobiumInVolcanicAsh[moles/liter]'
isolniobium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfZirconiumInVolcanicAsh[moles/liter]'
isolzirconium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfStrontiumInVolcanicAsh[moles/liter]'
isolstrontium = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfSeleniumInVolcanicAsh[moles/liter]'
isolselenium = ispquery( name )
```



```
call clearchar( 60, name )
name = 'SolubilityOfNickelInVolcanicAsh[moles/liter]'
isolnickel = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfChlorineInVolcanicAsh[moles/liter]'
isolchlorine = ispquery( name )

call clearchar( 60, name )
name = 'SolubilityOfCarbonInVolcanicAsh[moles/liter]'
isolcarbon = ispquery( name )

ikey=23567

end if

toe = valuesp( itime )
dlbr = valuesp( idlbr )
fpe = valuesp( ifpe )
fie = valuesp( ifie )
fpsat = valuesp( ifpsat )
fisat = valuesp( ifisat )
precip = valuesp( iprecip )
dirr = valuesp( idirr )
rhosoil = valuesp( irhosoil )
theta = valuesp( itheta )
depthsoil = valuesp( idepthsoil )


dkd(1) = valuesp( ikduranium )
dkd(2) = valuesp( ikdcurium )
dkd(3) = valuesp( ikdplutonium )
dkd(4) = valuesp( ikdamericium )
dkd(5) = valuesp( ikdplutonium )
dkd(6) = valuesp( ikduranium )
dkd(7) = valuesp( ikdthorium )
dkd(8) = valuesp( ikdradium )
dkd(9) = valuesp( ikdlead )
dkd(10) = valuesp( ikdcurium )
dkd(11) = valuesp( ikdamericium )
dkd(12) = valuesp( ikdplutonium )
dkd(13) = valuesp( ikduranium )
dkd(14) = valuesp( ikdprotactinium )
dkd(15) = valuesp( ikdactinium )
dkd(16) = valuesp( ikdcurium )
dkd(17) = valuesp( ikdplutonium )
dkd(18) = valuesp( ikdamericium )
dkd(19) = valuesp( ikdneptunium )
dkd(20) = valuesp( ikduranium )
dkd(21) = valuesp( ikdthorium )
dkd(22) = valuesp( ikdcurium )
dkd(23) = valuesp( ikdplutonium )
dkd(24) = valuesp( ikduranium )
dkd(25) = valuesp( ikduranium )
dkd(26) = valuesp( ikdsamarium )
dkd(27) = valuesp( ikdcesium )
dkd(28) = valuesp( ikdcesium )
```

```
dkd(29) = valuesp( ikdiodine )
dkd(30) = valuesp( ikdtin )
dkd(31) = valuesp( ikdtin )
dkd(32) = valuesp( ikdsilver )
dkd(33) = valuesp( ikdpaladium )
dkd(34) = valuesp( ikdtechnetium )
dkd(35) = valuesp( ikdmolybdenum )
dkd(36) = valuesp( ikdniobium )
dkd(37) = valuesp( ikdzirconium )
dkd(38) = valuesp( ikdstrontium )
dkd(39) = valuesp( ikdselenium )
dkd(40) = valuesp( ikdnickel )
dkd(41) = valuesp( ikdnickel )
dkd(42) = valuesp( ikdchlorine )
dkd(43) = valuesp( ikdcarbon )

sol(1) = valuesp( isoluranium )
sol(2) = valuesp( isolcurium )
sol(3) = valuesp( isolplutonium )
sol(4) = valuesp( isolamericium )
sol(5) = valuesp( isolplutonium )
sol(6) = valuesp( isoluranium )
sol(7) = valuesp( isolthorium )
sol(8) = valuesp( isolradium )
sol(9) = valuesp( isollead )
sol(10) = valuesp( isolcurium )
sol(11) = valuesp( isolamericium )
sol(12) = valuesp( isolplutonium )
sol(13) = valuesp( isoluranium )
sol(14) = valuesp( isolprotactinium )
sol(15) = valuesp( isolactinium )
sol(16) = valuesp( isolcurium )
sol(17) = valuesp( isolplutonium )
sol(18) = valuesp( isolamericium )
sol(19) = valuesp( isolneptunium )
sol(20) = valuesp( isoluranium )
sol(21) = valuesp( isolthorium )
sol(22) = valuesp( isolcurium )
sol(23) = valuesp( isolplutonium )
sol(24) = valuesp( isoluranium )
sol(25) = valuesp( isoluranium )
sol(26) = valuesp( isolsamarium )
sol(27) = valuesp( isolcesium )
sol(28) = valuesp( isolcesium )
sol(29) = valuesp( isoliodine )
sol(30) = valuesp( isoltin )
sol(31) = valuesp( isoltin )
sol(32) = valuesp( isolsilver )
sol(33) = valuesp( isolpaladium )
sol(34) = valuesp( isoltechnetium )
sol(35) = valuesp( isolmolybdenum )
sol(36) = valuesp( isolniobium )
sol(37) = valuesp( isolzirconium )
sol(38) = valuesp( isolstrontium )
sol(39) = valuesp( isolselenium )
sol(40) = valuesp( isolnickel )
sol(41) = valuesp( isolnickel )
sol(42) = valuesp( isolchlorine )
```



```
sol(43) = valuesp( isolcarbon )

cc    Must convert solubilities in moles/liter to moles/m3
      do i=1,43
        sol(i)=sol(i)*1000.d0        moles/m3 = moles/e · e/m3
      end do

      call newinventdb()
      call leachrate(dkd,fpe,fpsat,fie,fisat,precip,dirr,
     &               depthsoil,rhosoil,theta,dll)
      do i=1,43
        dlr(i)=0.6931472d0/halflifeperiso(i)        1/yr
      end do

cc    Since the toe could occur between time points, must trancate the
cc    toe to the previous time point to catch the peak

      iflag=0
      it=1
      do 10 while (iflag.eq.0)
       if (toe.ge.time(it).and.toe.lt.time(it+1)) then
        iflag=1
        itoe=it
       end if
       it=it+1
10    continue

cc    Assign all of the areal concentrations before the event
cc    to zero

      do j=1,43
        do k=1,itoe-1
          dn(k,j)=0.d0
        end do
      end do

cc    Must do a call to decayremove43mol to get the radionuclide inventory
cc    of the spent fuel at the time of the event.

      dt=time(itoe)-time(1)
      call decay43mol( dt, tempdnt1(1) )
      call decayremove43mol( dt, dlr, dnt1, tempdnt1 )
      do n=1,43
        dn(itoe,n)=tempdnt1(n)        moles/MTU
      end do

cc    Now must calculate the radionuclide concentrations for the remaining
cc    time points in the TPI

      tempnt=nt-itoe
      do i=1,tempnt
        dt=time(itoe+i)-time(itoe+(i-1))
        do m=1,43
          decision=sol(m)*(precip*(1.d0-fpe)*fpsat+    moles/m3 · m/yr = moles/m2-yr
     &             dirr*(1.d0-fie)*fisat)

cc    The following line has been changed from the original because
```

```
cc      an error was found. The first part of the if statement needs
cc      to be multiplied by the number of MTU/m2. MSJ 5/27/97
cc
cc
cc           if(dll(m)*tempdnt1(m).le.decision) then
                                                  1/yr  moles/MTU  g/cm2 · 10^4 cm2/m2 · 10^-6 MTU/g = moles/m2 yr
             if(dll(m)*tempdnt1(m)*gramsfpercm2/100.d0.le.decision) then
               dlt(m)=dlr(m)+dlbr+dll(m)
             else
cc
cc      The following line was also changed on 5/27/97 due to a units
cc      error. MSJ.
cc
cc             dlt(m)=dlr(m)+dlbr+decision/tempdnt1(m)
                                  1/yr  1/yr  moles/m2.yr / moles/m2 = 1/yr
               dlt(m)=dlr(m)+dlbr+decision/(tempdnt1(m)*gramsfpercm2/100.d0)
             end if
           end do
           call decayremove43mol( dt, dlt, tempdnt1, tempdn )
           do j=1,43
             dn(itoe+i,j)=tempdn(j)
             if (dn(itoe+i,j).lt.0.99d-99) then
               dn(itoe+i,j)=0.d0
               tempdn(j)=0.d0
             end if
             tempdnt1(j)=tempdn(j)
           end do
         end do
cc
cc       dn numbers are in moles/MTU, must convert them to number/g of sf
cc
           do i=1,nt
             do j=1,43
               dn(i,j)=dn(i,j)*6.023d23/1.d6    moles/MTU · 6.023E23 #/mole · MTU/10^6 g
             end do
           end do
cc
cc       Must tranform grams of sf per cm2 to Ci of radionuclide per m^2
cc
           do it=1,nt
             do inuc=1,43
cc
cc     The following two lines replace the commented out lines
cc     because the 1.0d-04 should be a 1.0d+04 to convert from
cc     Ci/cm2 to Ci/m2
cc     MSJ 5/14/97
cc                                  {1/yr · yr/d · d/h · h/s · #/S} = Bq/g · Ci/Bq · g/cm2 · 10^4 cm2/m2 = Ci/m2
           ciperm2(it,inuc)=(dlr(inuc)/365.d0/24.d0/3600.d0)*
     &          dn(it,inuc)/3.7d10*gramsfpercm2*(1.0d+4)
cc           ciperm2(it,inuc)=(dlr(inuc)/365.d0/24.d0/3600.d0)*
cc     &          dn(it,inuc)/3.7d10*gramsfpercm2*(1.0d-4)
             if (ciperm2(it,inuc).lt.0.99d-99) then
               ciperm2(it,inuc)=0.d0
             end if
           end do
         end do
         return
         end
c=======================================================================
```



```
==
      subroutine leachrate(dkd,fpe,fpsat,fie,fisat,precip,dirr,
     &           depthsoil,rhosoil,theta,dll)
c=======================================================================
==
c by M.S. Jarzemba, March 1, 1997
c       parameters
c
c       inputs:
c
c       dkd     Array of soil kd values in cm3/g
c       fpe     fraction of precipitation lost to evapotranspiration
c       fpsat   Fraction of year soil is saturated due to precipitation
c       fie     Fraction of irrigation lost to evapotranspiration
c       fisat   Fraction of year soil is saturated due to irrigation
c       precip  Annual precipitation in m/yr
c       dirr    Annual irrigation in m/yr
c       depthsoil  Depth of the rooting zone in m
c       rhosoil Bulk density of the soil in g/cm3
c       theta   Saturation fraction of soil
c
c       outputs:
c
c       dll     Array of leach rates for nuclides in the blanket (1/yr)
c
        implicit double precision (a-h,o-z)
        implicit integer (i-n)
        parameter(maxn=50)
        dimension dkd(maxn),dll(maxn)
        do i=1,43                                  m/yr + m/yr / [m · (unitless)] = 1/yr
          dll(i)=(precip*(1.d0-fpe)*fpsat+dirr*(1.d0-fie)*fisat)/
     &           (depthsoil*(1.d0+rhosoil/theta*dkd(i)))
        end do
        return
        end
```

5/27/97 MSJ

This section documents the tests performed to verify the operation of the ASHRMOVO.f module. Since the FORTRAN 77 file used to test this module is quite lengthy; it is not printed out, but is included on the diskette at the end of this notebook. The file is testashrmovo.f.

The following is the output file for the test runs. All outputs are in units of Ci/m²

```
testashrmovo.o
                    TimeOfNextVolcanicEventinRegionOfInterest[yr] = 0.1000E+04
                                    RelativeRateOfBlanketRemoval[1/yr] = 0.1000E-02
                  FractionOfPrecipitationLostToEvapotranspiration = 0.5000E+00
                     FractionOfIrrigationLostToEvapotranspiration = 0.5000E+00
                  FractionOfYearSoilIsSaturatedDueToPrecipitation = 0.5400E-01
                     FractionOfYearSoilIsSaturatedDueToIrrigation = 0.2000E+00
                                          AnnualPrecipitation[m/yr] = 0.1500E+00
                                          AnnualIrrigation[m/yr] = 0.1520E+01
                                             AshBulkDensity[g/cm3] = 0.2000E+01
                      AshVolumetricMoistureFractionAtSaturation = 0.4000E+00
                                       DepthOfTheRootingZone[m] = 0.1500E+00
                                  KdOfUraniumInVolcanicAsh[cm3/g] = 0.3500E+02
                                   KdOfCuriumInVolcanicAsh[cm3/g] = 0.4000E+04
                                KdOfPlutoniumInVolcanicAsh[cm3/g] = 0.5500E+03
                               KdOfAmericiumInVolcanicAsh[cm3/g] = 0.1900E+04
                                  KdOfThoriumInVolcanicAsh[cm3/g] = 0.3200E+04
                                   KdOfRadiumInVolcanicAsh[cm3/g] = 0.5000E+03
                                    KdOfLeadInVolcanicAsh[cm3/g] = 0.2700E+03
                              KdOfProtactiniumInVolcanicAsh[cm3/g] = 0.5500E+03
                                KdOfActiniumInVolcanicAsh[cm3/g] = 0.4500E+03
                               KdOfNeptuniumInVolcanicAsh[cm3/g] = 0.5000E+01
                                KdOfSamariumInVolcanicAsh[cm3/g] = 0.2450E+03
                                  KdOfCesiumInVolcanicAsh[cm3/g] = 0.2800E+03
                                  KdOfIodineInVolcanicAsh[cm3/g] = 0.1000E+01
                                     KdOfTinInVolcanicAsh[cm3/g] = 0.1300E+03
                                  KdOfSilverInVolcanicAsh[cm3/g] = 0.5500E+02
                                 KdOfPaladiumInVolcanicAsh[cm3/g] = 0.5500E+02
                              KdOfTechnetiumInVolcanicAsh[cm3/g] = 0.1000E+00
                             KdOfMolybdenumInVolcanicAsh[cm3/g] = 0.1000E+02
                                 KdOfNiobiumInVolcanicAsh[cm3/g] = 0.1600E+03
                               KdOfZirconiumInVolcanicAsh[cm3/g] = 0.6000E+03
                               KdOfStrontiumInVolcanicAsh[cm3/g] = 0.1500E+02
                                KdOfSeleniumInVolcanicAsh[cm3/g] = 0.1500E+03
                                  KdOfNickelInVolcanicAsh[cm3/g] = 0.4000E+03
                                KdOfChlorineInVolcanicAsh[cm3/g] = 0.0000E+00
                                  KdOfCarbonInVolcanicAsh[cm3/g] = 0.5000E+01
                SolubilityOfUraniumInVolcanicAsh[moles/liter] = 0.4500E-04
                  SolubilityOfCuriumInVolcanicAsh[moles/liter] = 0.1000E-05
```



```
              SolubilityOfPlutoniumInVolcanicAsh[moles/liter] = 0.5000E-05
                SolubilityOfAmericiumInVolcanicAsh[moles/liter] = 0.5000E-05
                  SolubilityOfThoriumInVolcanicAsh[moles/liter] = 0.3200E-08
                    SolubilityOfRadiumInVolcanicAsh[moles/liter] = 0.1000E-06
                      SolubilityOfLeadInVolcanicAsh[moles/liter] = 0.3200E-06
                SolubilityOfProtactiniumInVolcanicAsh[moles/liter] = 0.3200E-07
                  SolubilityOfActiniumInVolcanicAsh[moles/liter] = 0.5000E-05
                 SolubilityOfNeptuniumInVolcanicAsh[moles/liter] = 0.1600E-03
                  SolubilityOfSamariumInVolcanicAsh[moles/liter] = 0.5000E-05
                    SolubilityOfCesiumInVolcanicAsh[moles/liter] = 0.1000E+01
                    SolubilityOfIodineInVolcanicAsh[moles/liter] = 0.5000E-07
                       SolubilityOfTinInVolcanicAsh[moles/liter] = 0.1000E+01
                    SolubilityOfSilverInVolcanicAsh[moles/liter] = 0.1000E+01
                   SolubilityOfPaladiumInVolcanicAsh[moles/liter] = 0.9500E-03
                SolubilityOfTechnetiumInVolcanicAsh[moles/liter] = 0.1000E+01
               SolubilityOfMolybdenumInVolcanicAsh[moles/liter] = 0.1000E+01
                  SolubilityOfNiobiumInVolcanicAsh[moles/liter] = 0.1000E-07
                SolubilityOfZirconiumInVolcanicAsh[moles/liter] = 0.3200E-09
                SolubilityOfStrontiumInVolcanicAsh[moles/liter] = 0.1300E-03
                  SolubilityOfSeleniumInVolcanicAsh[moles/liter] = 0.1000E+00
                    SolubilityOfNickelInVolcanicAsh[moles/liter] = 0.2000E+01
                  SolubilityOfChlorineInVolcanicAsh[moles/liter] = 0.1000E+01
                    SolubilityOfCarbonInVolcanicAsh[moles/liter] = 0.1000E+01
     1.0000 g-sf/cm2

Chain #1

  time(yr)     Cm-246     Pu-242      U-238      U-234      Th-230      Ra-226      Pb-210
        .0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       1.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       1.3 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       1.5 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       1.8 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       2.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       2.5 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       3.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       3.5 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       4.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       5.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       6.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       7.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       8.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
       9.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      10.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      12.5 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      15.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      17.5 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      20.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      25.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      30.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      35.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      40.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      50.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      60.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      70.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      80.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
      90.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
     100.0 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00 0.000E+00
```

125.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
150.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
175.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
200.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
250.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
300.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
350.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
400.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
500.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
600.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
700.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
800.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
900.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1000.0	0.221E-03	0.160E-01	0.319E-02	0.189E-01	0.165E-03	0.306E-04	0.288E-04
1250.0	0.158E-03	0.113E-01	0.238E-02	0.335E-02	0.143E-03	0.301E-04	0.147E-04
1500.0	0.113E-03	0.800E-02	0.175E-02	0.595E-03	0.112E-03	0.274E-04	0.134E-04
1750.0	0.804E-04	0.567E-02	0.125E-02	0.106E-03	0.858E-04	0.235E-04	0.115E-04
2000.0	0.575E-04	0.401E-02	0.873E-03	0.192E-04	0.656E-04	0.194E-04	0.955E-05
2500.0	0.293E-04	0.201E-02	0.384E-03	0.795E-06	0.382E-04	0.124E-04	0.613E-05
3000.0	0.150E-04	0.101E-02	0.112E-03	0.886E-07	0.222E-04	0.758E-05	0.375E-05
3500.0	0.764E-05	0.505E-03	0.549E-05	0.907E-08	0.129E-04	0.453E-05	0.224E-05
4000.0	0.390E-05	0.253E-03	0.173E-06	0.531E-09	0.752E-05	0.267E-05	0.132E-05
5000.0	0.102E-05	0.635E-04	0.175E-09	0.102E-11	0.255E-05	0.915E-06	0.453E-06
6000.0	0.265E-06	0.160E-04	0.620E-12	0.172E-14	0.862E-06	0.311E-06	0.154E-06
7000.0	0.690E-07	0.401E-05	0.113E-12	0.596E-16	0.292E-06	0.105E-06	0.521E-07
8000.0	0.180E-07	0.101E-05	0.282E-13	0.145E-16	0.988E-07	0.357E-07	0.176E-07
9000.0	0.468E-08	0.253E-06	0.709E-14	0.363E-17	0.335E-07	0.121E-07	0.598E-08
10000.0	0.122E-08	0.635E-07	0.178E-14	0.912E-18	0.113E-07	0.409E-08	0.202E-08

Subchain #1a parents

time(yr)	Am-242m	Pu-238
.0	0.000E+00	0.000E+00
1.0	0.000E+00	0.000E+00
1.3	0.000E+00	0.000E+00
1.5	0.000E+00	0.000E+00
1.8	0.000E+00	0.000E+00
2.0	0.000E+00	0.000E+00
2.5	0.000E+00	0.000E+00
3.0	0.000E+00	0.000E+00
3.5	0.000E+00	0.000E+00
4.0	0.000E+00	0.000E+00
5.0	0.000E+00	0.000E+00
6.0	0.000E+00	0.000E+00
7.0	0.000E+00	0.000E+00
8.0	0.000E+00	0.000E+00
9.0	0.000E+00	0.000E+00
10.0	0.000E+00	0.000E+00
12.5	0.000E+00	0.000E+00
15.0	0.000E+00	0.000E+00
17.5	0.000E+00	0.000E+00
20.0	0.000E+00	0.000E+00
25.0	0.000E+00	0.000E+00
30.0	0.000E+00	0.000E+00
35.0	0.000E+00	0.000E+00
40.0	0.000E+00	0.000E+00
50.0	0.000E+00	0.000E+00
60.0	0.000E+00	0.000E+00

70.0	0.000E+00	0.000E+00
80.0	0.000E+00	0.000E+00
90.0	0.000E+00	0.000E+00
100.0	0.000E+00	0.000E+00
125.0	0.000E+00	0.000E+00
150.0	0.000E+00	0.000E+00
175.0	0.000E+00	0.000E+00
200.0	0.000E+00	0.000E+00
250.0	0.000E+00	0.000E+00
300.0	0.000E+00	0.000E+00
350.0	0.000E+00	0.000E+00
400.0	0.000E+00	0.000E+00
500.0	0.000E+00	0.000E+00
600.0	0.000E+00	0.000E+00
700.0	0.000E+00	0.000E+00
800.0	0.000E+00	0.000E+00
900.0	0.000E+00	0.000E+00
1000.0	0.728E-03	0.856E-02
1250.0	0.564E-04	0.170E-03
1500.0	0.437E-05	0.641E-05
1750.0	0.339E-06	0.405E-06
2000.0	0.263E-07	0.301E-07
2500.0	0.158E-09	0.179E-09
3000.0	0.948E-12	0.108E-11
3500.0	0.569E-14	0.647E-14
4000.0	0.342E-16	0.389E-16
5000.0	0.123E-20	0.140E-20
6000.0	0.445E-25	0.506E-25
7000.0	0.161E-29	0.183E-29
8000.0	0.579E-34	0.659E-34
9000.0	0.209E-38	0.238E-38
10000.0	0.754E-43	0.857E-43

Chain #2

time(yr)	Cm-243	Pu-239	U-235	Pa-231	Ac-227
.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.3	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.8	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
2.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
2.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
3.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
3.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
4.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
5.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
6.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
7.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
8.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
9.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
10.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
12.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
15.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
17.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
20.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
25.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
30.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00

35.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
40.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
50.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
60.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
70.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
80.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
90.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
100.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
125.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
150.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
175.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
200.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
250.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
300.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
350.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
400.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
500.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
600.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
700.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
800.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
900.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1000.0	0.284E-11	0.300E+01	0.172E-03	0.382E-05	0.370E-05
1250.0	0.114E-16	0.209E+01	0.308E-04	0.302E-05	0.151E-05
1500.0	0.460E-22	0.146E+01	0.568E-05	0.218E-05	0.109E-05
1750.0	0.185E-27	0.102E+01	0.115E-05	0.154E-05	0.770E-06
2000.0	0.744E-33	0.714E+00	0.299E-06	0.108E-05	0.541E-06
2500.0	0.120E-43	0.348E+00	0.681E-07	0.534E-06	0.267E-06
3000.0	0.195E-54	0.170E+00	0.308E-07	0.263E-06	0.131E-06
3500.0	0.315E-65	0.831E-01	0.149E-07	0.129E-06	0.646E-07
4000.0	0.510E-76	0.405E-01	0.729E-08	0.636E-07	0.318E-07
5000.0	0.000E+00	0.967E-02	0.174E-08	0.154E-07	0.770E-08
6000.0	0.000E+00	0.231E-02	0.415E-09	0.373E-08	0.186E-08
7000.0	0.000E+00	0.550E-03	0.989E-10	0.903E-09	0.451E-09
8000.0	0.000E+00	0.131E-03	0.236E-10	0.219E-09	0.109E-09
9000.0	0.000E+00	0.314E-04	0.564E-11	0.530E-10	0.265E-10
10000.0	0.000E+00	0.750E-05	0.135E-11	0.128E-10	0.641E-11

Subchain #2a parent

time(yr)	Am-243
.0	0.000E+00
1.0	0.000E+00
1.3	0.000E+00
1.5	0.000E+00
1.8	0.000E+00
2.0	0.000E+00
2.5	0.000E+00
3.0	0.000E+00
3.5	0.000E+00
4.0	0.000E+00
5.0	0.000E+00
6.0	0.000E+00
7.0	0.000E+00
8.0	0.000E+00
9.0	0.000E+00
10.0	0.000E+00
12.5	0.000E+00
15.0	0.000E+00

17.5	0.000E+00
20.0	0.000E+00
25.0	0.000E+00
30.0	0.000E+00
35.0	0.000E+00
40.0	0.000E+00
50.0	0.000E+00
60.0	0.000E+00
70.0	0.000E+00
80.0	0.000E+00
90.0	0.000E+00
100.0	0.000E+00
125.0	0.000E+00
150.0	0.000E+00
175.0	0.000E+00
200.0	0.000E+00
250.0	0.000E+00
300.0	0.000E+00
350.0	0.000E+00
400.0	0.000E+00
500.0	0.000E+00
600.0	0.000E+00
700.0	0.000E+00
800.0	0.000E+00
900.0	0.000E+00
1000.0	0.140E+00
1250.0	0.101E+00
1500.0	0.732E-01
1750.0	0.529E-01
2000.0	0.383E-01
2500.0	0.200E-01
3000.0	0.105E-01
3500.0	0.546E-02
4000.0	0.286E-02
5000.0	0.780E-03
6000.0	0.213E-03
7000.0	0.583E-04
8000.0	0.159E-04
9000.0	0.435E-05
10000.0	0.119E-05

Chain #3

time(yr)	Cm-245	Pu-241	Am-241	Np-237	U-233	Th-229
.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.3	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.8	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
2.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
2.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
3.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
3.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
4.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
5.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
6.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
7.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
8.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00

9.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
10.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
12.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
15.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
17.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
20.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
25.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
30.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
35.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
40.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
50.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
60.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
70.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
80.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
90.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
100.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
125.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
150.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
175.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
200.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
250.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
300.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
350.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
400.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
500.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
600.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
700.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
800.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
900.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1000.0	0.115E-02	0.115E-02	0.824E+01	0.952E-02	0.313E-04	0.125E-05
1250.0	0.849E-03	0.424E-03	0.280E+01	0.251E-04	0.578E-05	0.121E-05
1500.0	0.627E-03	0.313E-03	0.952E+00	0.840E-05	0.103E-05	0.938E-06
1750.0	0.463E-03	0.231E-03	0.324E+00	0.286E-05	0.185E-06	0.695E-06
2000.0	0.341E-03	0.170E-03	0.110E+00	0.971E-06	0.336E-07	0.510E-06
2500.0	0.186E-03	0.928E-04	0.127E-01	0.112E-06	0.119E-08	0.273E-06
3000.0	0.101E-03	0.505E-04	0.149E-02	0.132E-07	0.535E-10	0.146E-06
3500.0	0.552E-04	0.275E-04	0.184E-03	0.161E-08	0.359E-11	0.778E-07
4000.0	0.300E-04	0.150E-04	0.273E-04	0.236E-09	0.370E-12	0.416E-07
5000.0	0.891E-05	0.445E-05	0.256E-05	0.210E-10	0.183E-13	0.119E-07
6000.0	0.264E-05	0.132E-05	0.686E-06	0.558E-11	0.431E-14	0.338E-08
7000.0	0.784E-06	0.392E-06	0.203E-06	0.165E-11	0.126E-14	0.966E-09
8000.0	0.233E-06	0.116E-06	0.601E-07	0.489E-12	0.375E-15	0.276E-09
9000.0	0.690E-07	0.345E-07	0.178E-07	0.145E-12	0.111E-15	0.787E-10
10000.0	0.205E-07	0.102E-07	0.529E-08	0.430E-13	0.330E-16	0.225E-10

Chain #4

time(yr)	Cm-244	Pu-240	U-236
.0	0.000E+00	0.000E+00	0.000E+00
1.0	0.000E+00	0.000E+00	0.000E+00
1.3	0.000E+00	0.000E+00	0.000E+00
1.5	0.000E+00	0.000E+00	0.000E+00
1.8	0.000E+00	0.000E+00	0.000E+00
2.0	0.000E+00	0.000E+00	0.000E+00
2.5	0.000E+00	0.000E+00	0.000E+00
3.0	0.000E+00	0.000E+00	0.000E+00
3.5	0.000E+00	0.000E+00	0.000E+00
4.0	0.000E+00	0.000E+00	0.000E+00

5.0	0.000E+00	0.000E+00	0.000E+00
6.0	0.000E+00	0.000E+00	0.000E+00
7.0	0.000E+00	0.000E+00	0.000E+00
8.0	0.000E+00	0.000E+00	0.000E+00
9.0	0.000E+00	0.000E+00	0.000E+00
10.0	0.000E+00	0.000E+00	0.000E+00
12.5	0.000E+00	0.000E+00	0.000E+00
15.0	0.000E+00	0.000E+00	0.000E+00
17.5	0.000E+00	0.000E+00	0.000E+00
20.0	0.000E+00	0.000E+00	0.000E+00
25.0	0.000E+00	0.000E+00	0.000E+00
30.0	0.000E+00	0.000E+00	0.000E+00
35.0	0.000E+00	0.000E+00	0.000E+00
40.0	0.000E+00	0.000E+00	0.000E+00
50.0	0.000E+00	0.000E+00	0.000E+00
60.0	0.000E+00	0.000E+00	0.000E+00
70.0	0.000E+00	0.000E+00	0.000E+00
80.0	0.000E+00	0.000E+00	0.000E+00
90.0	0.000E+00	0.000E+00	0.000E+00
100.0	0.000E+00	0.000E+00	0.000E+00
125.0	0.000E+00	0.000E+00	0.000E+00
150.0	0.000E+00	0.000E+00	0.000E+00
175.0	0.000E+00	0.000E+00	0.000E+00
200.0	0.000E+00	0.000E+00	0.000E+00
250.0	0.000E+00	0.000E+00	0.000E+00
300.0	0.000E+00	0.000E+00	0.000E+00
350.0	0.000E+00	0.000E+00	0.000E+00
400.0	0.000E+00	0.000E+00	0.000E+00
500.0	0.000E+00	0.000E+00	0.000E+00
600.0	0.000E+00	0.000E+00	0.000E+00
700.0	0.000E+00	0.000E+00	0.000E+00
800.0	0.000E+00	0.000E+00	0.000E+00
900.0	0.000E+00	0.000E+00	0.000E+00
1000.0	0.149E-15	0.459E+01	0.255E-02
1250.0	0.559E-24	0.308E+01	0.465E-03
1500.0	0.210E-32	0.207E+01	0.911E-04
1750.0	0.788E-41	0.139E+01	0.219E-04
2000.0	0.296E-49	0.936E+00	0.772E-05
2500.0	0.417E-66	0.423E+00	0.243E-05
3000.0	0.588E-83	0.191E+00	0.106E-05
3500.0	0.000E+00	0.861E-01	0.479E-06
4000.0	0.000E+00	0.389E-01	0.216E-06
5000.0	0.000E+00	0.793E-02	0.441E-07
6000.0	0.000E+00	0.162E-02	0.900E-08
7000.0	0.000E+00	0.330E-03	0.183E-08
8000.0	0.000E+00	0.672E-04	0.374E-09
9000.0	0.000E+00	0.137E-04	0.762E-10
10000.0	0.000E+00	0.279E-05	0.155E-10

U-232 and the Fission Products

time(yr)	U-232	Sm-151	Cs-137	Cs-135	I-129	Sn-126	Sn-121m
.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.3	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.8	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
2.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00

2.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
3.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
3.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
4.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
5.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
6.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
7.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
8.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
9.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
10.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
12.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
15.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
17.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
20.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
25.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
30.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
35.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
40.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
50.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
60.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
70.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
80.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
90.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
100.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
125.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
150.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
175.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
200.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
250.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
300.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
350.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
400.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
500.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
600.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
700.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
800.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
900.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1000.0	0.140E-07	0.127E-02	0.488E-07	0.351E-02	0.295E-03	0.711E-02	0.604E-08
1250.0	0.202E-10	0.170E-04	0.303E-12	0.227E-02	0.198E-03	0.370E-02	0.307E-11
1500.0	0.292E-13	0.228E-06	0.189E-17	0.147E-02	0.123E-03	0.193E-02	0.156E-14
1750.0	0.421E-16	0.305E-08	0.117E-22	0.949E-03	0.666E-04	0.100E-02	0.791E-18
2000.0	0.607E-19	0.408E-10	0.729E-28	0.614E-03	0.266E-04	0.522E-03	0.402E-21
2500.0	0.126E-24	0.731E-14	0.282E-38	0.257E-03	0.574E-06	0.141E-03	0.104E-27
3000.0	0.263E-30	0.131E-17	0.109E-48	0.107E-03	0.778E-44	0.383E-04	0.268E-34
3500.0	0.547E-36	0.235E-21	0.421E-59	0.449E-04	0.105E-81	0.104E-04	0.690E-41
4000.0	0.114E-41	0.421E-25	0.163E-69	0.188E-04	0.000E+00	0.281E-05	0.178E-47
5000.0	0.493E-53	0.135E-32	0.243E-90	0.329E-05	0.000E+00	0.206E-06	0.119E-60
6000.0	0.214E-64	0.435E-40	0.000E+00	0.575E-06	0.000E+00	0.151E-07	0.789E-74
7000.0	0.926E-76	0.140E-47	0.000E+00	0.101E-06	0.000E+00	0.111E-08	0.525E-87
8000.0	0.401E-87	0.449E-55	0.000E+00	0.176E-07	0.000E+00	0.816E-10	0.000E+00
9000.0	0.000E+00	0.144E-62	0.000E+00	0.308E-08	0.000E+00	0.599E-11	0.000E+00
10000.0	0.000E+00	0.463E-70	0.000E+00	0.539E-09	0.000E+00	0.439E-12	0.000E+00

time(yr)	Ag-108m	Pd-107	Tc-99	Mo-93	Nb-94	Zr-93	Sr-90
.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.3	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00

1.8	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
2.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
2.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
3.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
3.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
4.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
5.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
6.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
7.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
8.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
9.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
10.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
12.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
15.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
17.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
20.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
25.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
30.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
35.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
40.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
50.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
60.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
70.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
80.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
90.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
100.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
125.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
150.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
175.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
200.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
250.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
300.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
350.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
400.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
500.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
600.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
700.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
800.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
900.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1000.0	0.469E-06	0.105E-02	0.123E+00	0.826E-04	0.487E-02	0.185E-01	0.167E-07
1250.0	0.929E-08	0.319E-03	0.476E-76	0.355E-06	0.269E-02	0.144E-01	0.287E-14
1500.0	0.184E-09	0.967E-04	0.000E+00	0.153E-08	0.149E-02	0.112E-01	0.495E-21
1750.0	0.365E-11	0.293E-04	0.000E+00	0.657E-11	0.825E-03	0.872E-02	0.854E-28
2000.0	0.723E-13	0.891E-05	0.000E+00	0.283E-13	0.456E-03	0.679E-02	0.147E-34
2500.0	0.284E-16	0.820E-06	0.000E+00	0.523E-18	0.140E-03	0.411E-02	0.437E-48
3000.0	0.111E-19	0.756E-07	0.000E+00	0.968E-23	0.428E-04	0.249E-02	0.130E-61
3500.0	0.438E-23	0.696E-08	0.000E+00	0.179E-27	0.131E-04	0.151E-02	0.386E-75
4000.0	0.172E-26	0.641E-09	0.000E+00	0.332E-32	0.401E-05	0.909E-03	0.115E-88
5000.0	0.265E-33	0.544E-11	0.000E+00	0.114E-41	0.376E-06	0.330E-03	0.000E+00
6000.0	0.409E-40	0.462E-13	0.000E+00	0.389E-51	0.353E-07	0.117E-03	0.000E+00
7000.0	0.630E-47	0.392E-15	0.000E+00	0.133E-60	0.331E-08	0.389E-04	0.000E+00
8000.0	0.972E-54	0.332E-17	0.000E+00	0.456E-70	0.310E-09	0.106E-04	0.000E+00
9000.0	0.150E-60	0.282E-19	0.000E+00	0.156E-79	0.291E-10	0.276E-05	0.000E+00
10000.0	0.231E-67	0.239E-21	0.000E+00	0.534E-89	0.272E-11	0.716E-06	0.000E+00

time(yr)	Se-79	Ni-63	Ni-59	Cl-36	C-14
.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00

1.3	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1.8	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
2.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
2.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
3.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
3.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
4.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
5.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
6.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
7.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
8.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
9.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
10.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
12.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
15.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
17.5	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
20.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
25.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
30.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
35.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
40.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
50.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
60.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
70.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
80.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
90.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
100.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
125.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
150.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
175.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
200.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
250.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
300.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
350.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
400.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
500.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
600.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
700.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
800.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
900.0	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
1000.0	0.376E-02	0.145E-02	0.244E-01	0.117E-03	0.118E-01
1250.0	0.206E-02	0.228E-04	0.166E-01	0.000E+00	0.390E-06
1500.0	0.113E-02	0.361E-06	0.113E-01	0.000E+00	0.129E-10
1750.0	0.618E-03	0.571E-08	0.770E-02	0.000E+00	0.429E-15
2000.0	0.339E-03	0.903E-10	0.524E-02	0.000E+00	0.142E-19
2500.0	0.102E-03	0.226E-13	0.243E-02	0.000E+00	0.157E-28
3000.0	0.305E-04	0.564E-17	0.113E-02	0.000E+00	0.173E-37
3500.0	0.917E-05	0.141E-20	0.522E-03	0.000E+00	0.190E-46
4000.0	0.275E-05	0.353E-24	0.242E-03	0.000E+00	0.209E-55
5000.0	0.248E-06	0.220E-31	0.521E-04	0.000E+00	0.253E-73
6000.0	0.223E-07	0.138E-38	0.112E-04	0.000E+00	0.307E-91
7000.0	0.201E-08	0.860E-46	0.241E-05	0.000E+00	0.000E+00
8000.0	0.181E-09	0.537E-53	0.517E-06	0.000E+00	0.000E+00
9000.0	0.164E-10	0.336E-60	0.111E-06	0.000E+00	0.000E+00
10000.0	0.147E-11	0.210E-67	0.239E-07	0.000E+00	0.000E+00

The following 'hand calculations' all use data for U-238 to verify ASHRMOVO.f operations

Calculation of the 'transition pt.' (Ci/m^2) for transitioning from a release to a solubility limited regime (Jarzemba & Manteufel, 1997, Health Physics. In Press). The t_p will be the point on a Log(ci per m^2) v. time plot where the slope has changed (to be shown later).

t_p = max rate of release (ie solubility limit) [$Ci/m^2.yr$] / leach rate ($1/yr$)

= numerator / denominator

Using data from the previous testashrmovo.o file

numerator = sol (precip. rt $(1-f_{pe}) \times f_{psat}$ + irr rt. $(1-F_{ie}) \times f_{isat}$)

sol = solubility = 4.5×10^{-2} moles/m^3

precip rt = precipitation rt = 0.15 m/yr

f_{pe} = fraction of precip lost to evap. = 0.5

f_{psat} = fraction of year soil is sat. due to precip = 0.054

irr rt = irrigation rt. = 1.52 m/yr

F_{ie} = fraction of irrigation lost to evap = 0.5

f_{isat} = fraction of year soil is sat. due to irrigation = 0.2

numerator = $4.5 \times 10^{-2} (0.15(1-0.5)0.054 + 1.52(1-0.5)0.2)$

= 7.02×10^{-3} moles/$m^2.yr$ * 8×10^{-5} Ci/mole

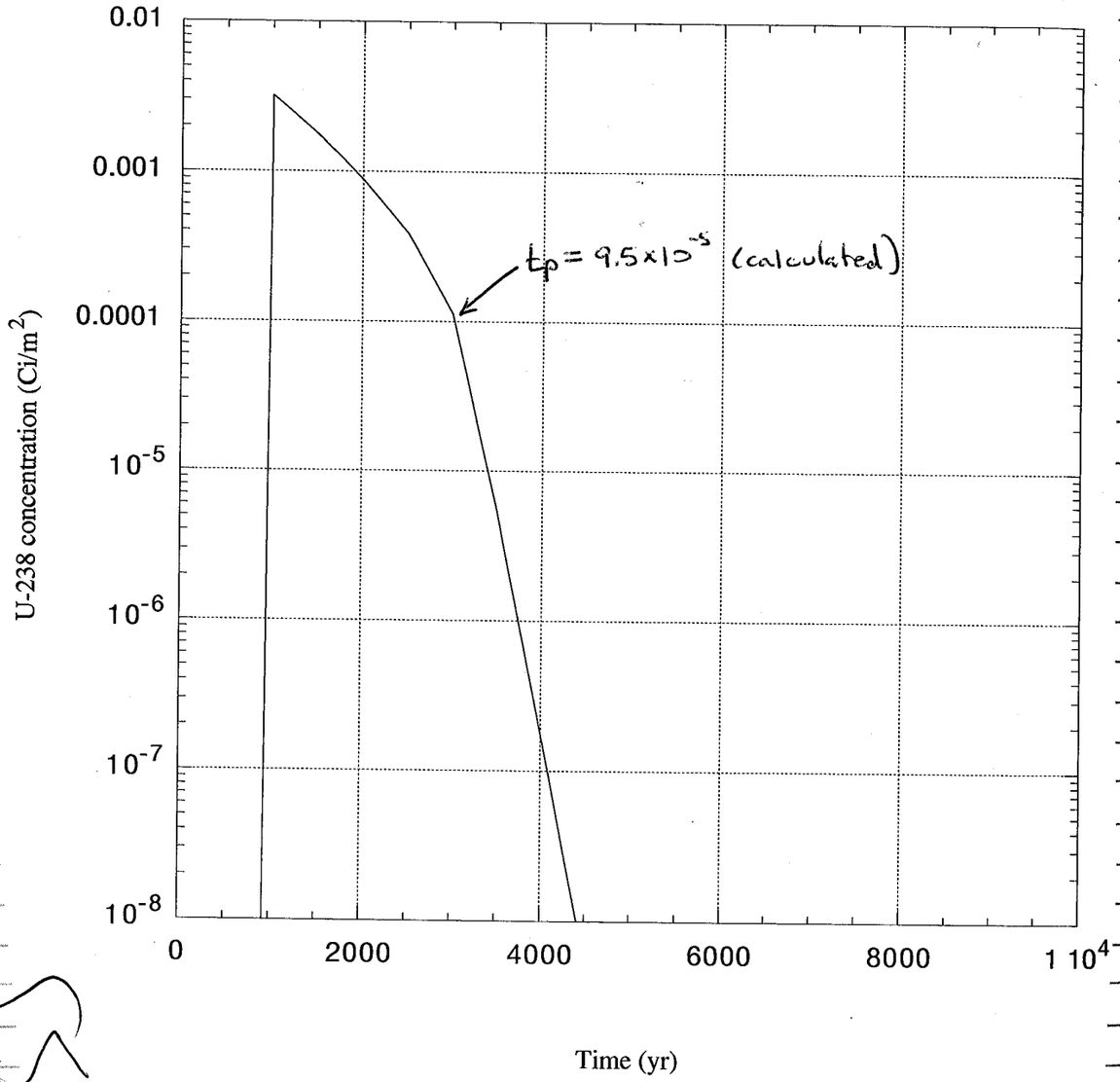
= 5.619×10^{-7} Ci/$m^2.yr$

denominator = DESCRIBED IN SUBROUTINE ON p. 101

$= [0.15(1-0.5) \times 0.054 + 1.52(1-0.5)0.2] / [0.15 + (1 + \frac{2}{0.4}35)]$

= 5.91×10^{-3} $1/yr$

$t_p = 5.619 \times 10^{-7} / 5.91 \times 10^{-3} = 9.5 \times 10^{-5}$ Ci/m^2



Data from testashrmovo.o file

This serves as verification that the solubility/release rt. limit portion of the code is working as advertised.

The 1,000 yr inventories should just be the Ci of activity per 10,000 g of spent fuel. The following is a Table to verify this pt.

Nuclide	testashrmovo.o (Ci/m^2)	INVENT rpt. (Ci/0.01MTU)
Cm-246	0.221E-3	0.22E-03
Am-241	8.24	8
Ac-227	0.37E-5	0.4E-05

From these tests, it appears as if the ashrmovo.f module is functioning as advertised.

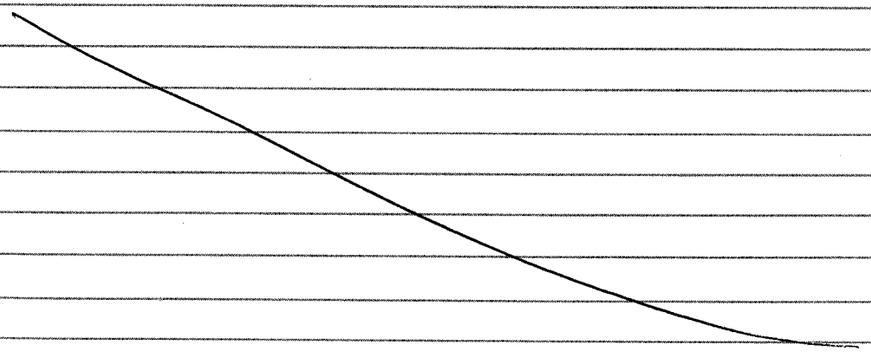
5/29/97 msj

One last check:

The slope after t_p on the previous page should be due to $\lambda^T = \lambda^{Leach} + \lambda^{EROSION} + \lambda^{RAD.\ DECAY}$

$\lambda^T = 5.91 \times 10^{-3}\,(1/yr) + 1 \times 10^{-3}\,(1/yr) + 1.55 \times 10^{-10}\,(1/yr)$
$= 6.91 \times 10^{-3}\,(1/yr)$

In 1,000 yrs, the concentration should decrease by a factor of $e^{-6.91} \cong 1 \times 10^{-3}$ which it does.



2/19/99 With the incorporation of the herein described code modules, ~~th~~ mgj into TPA, this project phase is closed

GW

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

Gordon Withmeyer 2/25/99

(Element Manager signature and date above line,
Name of Element beneath line)"

Gordon Withmeyer

ADDITIONAL INFORMATION FOR SCIENTIFIC NOTEBOOK #: 221

Document Date:	05/16/1997
Availability:	Southwest Research Institute® Center for Nuclear Waste Regulatory Analyses 6220 Culebra Road San Antonio, Texas 78228
Contact:	Southwest Research Institute® Center for Nuclear Waste Regulatory Analyses 6220 Culebra Road San Antonio, TX 78228-5166 Attn.: Director of Administration 210.522.5054
Data Sensitivity:	■"Non-Sensitive" □ Sensitive □"Non-Sensitive - Copyright" □ Sensitive - Copyright
Date Generated:	09/14/1999
Operating System: (including version number)	SUN OS, but readable using NT 4.0
Application Used: (including version number)	Fortran
Media Type: (CDs, 3 1/2, 5 1/4 disks, etc.)	1 - 3 ½ disk, 1 CD
File Types: (.exe, .bat, .zip, etc.)	UNIX executable (.e); object files (.o); input/output files from Fortran Code
Remarks: (computer runs, etc.)	Media contains: Fortran program; executable and input/output files.