```
!   homogeneous shield.f90
!   ************************************************************************
!   Monte Carlo Simulation of Multigroup Angular Neutron Transport in homogeneous
!   shield
!   Program logic based on the demon demonstration shielding code
!   M. Ragheb, Univ. of Illinois
!   ************************************************************************
    program homogeneous shield
!   Geometry:       Semi-infinite homogeneous slab
!   Composition:    Single light element as neutron scatterer
!   Source:         Point monoenergetic neutron source on shield's face

!       Computed Parameters:
!
!   Leakage Energy
!   Transmitted current
!   Transmitted dose
!   Reflected current
!   Reflected dose
!   Neutrons absorption as a function of energy and position
!
!       Simulation Logic:
!
!   Source particles sampling
!       Select polar angle for incident particle
!       Select azimuthal angle for incident particle
!       Calculate direction cosines for incident particle
!   Increment number of simulations counter
!   Test for completion of single particle history
!   Test for completion of overall simulation
!   Store results for future visualization
!   Sampling the Transport Kernel
!       Sample a path length
!       Calculate particle's position within the shield
!       Check for particle transmission or reflection
!       In case of reflection or transmission accumulate scores, and terminate history
!   Sampling the Collision Kernel
!       Determine whether particle scatters or gets absorbed
!       For scattering, calculate the cosine of the scattering angle
!       Calculate the outgoing neutron energy after collision
!       Calculate the polar angle and the azimuthal angles of the scattred neutron
!       Calculate new direction cosines of scattered particle
!   Analysis of reflected and transmitted radiation
!       Calculate dose rates
!       Calculate absorption rates

!   Declare variables and reserve memory storage
    dimension xnit(144)
    dimension ftd(8),ft(8,6),dt(8,6),fr(8,6),dr(8,6),cda(8),cdb(8)

!   Declare neutrons transmission, reflection, and absorption arrays
    integer neneg,nang,nhist,ndivt
    real eo,tt,coa,aw
    data pi /3.142593/
    common n1,nn1,ndt,iop,eas,ang(6),bng(6),t1(21),e(8), &
          & t(8,6),r(8,6),a(8,20),z,g
    equivalence (ft(1,1),xnit(1)),(dt(1,1),xnit(49)),(dr(1,1),xnit(97))

!   Define tab character for plotting routines
    character*1 tab
    tab=char(9)

    do i=1,144
        xnit(i)=0.0
    end do

!   t1(i):  boundary points of shield regions
!   ang(i): angular intervals less than ninety degrees
!   bng(i): angular intervals larger than ninety degrees
!   ftd(i): flux to dose conversion factors
!   e(i):   energy intervals
!   t(i,j): neutrons transmissions array for energy i and angle j
!   r(i,j): neutrons reflections array for energy i and angle j
!   a(i,k): neutrons absorptions array for energy i at position k

!   Initialize variables
```

```
    ktest=10
!   neneg:  number of output energy groups
    neneg=6
!   nang:   number of angular intervals
    nang=5
!   nhist:  number of particle histories in simulation
    nhist=100
    write(*,*) 'Number of Simulations=',nhist
!   ndivt:  number of shield regions
    ndivt=5
!   eo:     source particle energy in MeV
    eo=3.0
!   tt:     shield thickness in cms
    tt=2.0
!   coa:    small angle value below which the polar angle theta is considered to be
!           zero, cut-off angle
    coa=1.0e-05
!   aw:     scattering material mass number A, in amus (A=28 amus, Aluminum)
    aw=28.0

!   Subdivide angular intervals in degrees
!   Assign angles less than ninety degrees
    nn1=nang+1
    ang(1)= 0.0
    ang(2)= 15.
    ang(3)= 30.
    ang(4)= 45.
    ang(5)= 60.
    ang(6)= 90.
!   Calculate angles larger than ninety degrees
    do i=1,nn1
        bng(i)=180.0-ang(i)
    end do

!   Convert angles from degrees to radians
!   Convert cutoff angle
    coa=coa*.0174533
!   Convert other angles
    do i=1,nn1
        ang(i)=ang(i)*.0174533
        bng(i)=bng(i)*.0174533
    end do

!   Define flux to dose conversion factors
    ftd(1)=2.00e-09
    ftd(2)=3.20e-09
    ftd(3)=3.90e-09
    ftd(4)=4.10e-09
    ftd(5)=4.20e-09
    ftd(6)=4.42e-09

!   Subdivide shield into "ndivt" regions
    ndt=ndivt+1
    tt1=tt/ndivt
    t1(1)=0.0
    do i=2,ndt
        t1(i)=t1(i-1)+tt1
    end do

!   Subdivide energy range (MeV)
    n1=neneg+1
    e(1)=0.0
    e(2)=0.5
    e(3)=1.0
    e(4)=1.5
    e(5)=2.0
    e(6)=2.5
    e(7)=3.0

!   Initialize data storage arrays
!   i= energy variable
!   j= angular variable
!   k= position variable
    do i=1,neneg
        do j=1,nang
            t(i,j)=0.0
```

```
            r(i,j)=0.0
        end do
        do k=1,ndivt
            a(i,k)=0.0
        end do
    end do

!   Initialize history counter
    nh=0

!   Generate particle history
!
!   Sampling source parameters
1   call source(alfa,b,g,eo,eas,z)

!   Initialize scattering counter
    s=0.0

!   Increment history counter
    nh=nh+1

!   Check for histories completion, if simulation is complete switch to output
    if (nhist-nh) 5,2,2

!   Sample Transport Kernel
!   Get total cross section correponding to particle's energy eas
2   call crosec(tcs,scs,eas,ktest)
!   Sample neutron path length, pl
    call random(rr)
    pl=-log(rr)/tcs

!   Determine particle position after its transport by one path length
    z=z+g*pl

!   Determine whether particle has been transmitted,reflected, or absorbed
!   iop=1 Particle transmitted
!   iop=2 Particle reflected
!   iop=3 Particle absorbed
    if(tt-z) 7,7,8

!   Particle Transmitted
7   iop=1
!   Store transmitted particle
    call score
!   Start new particle history
    goto 1

8   if(z) 9,9,3
!   Particle Reflected
9   iop=2
!   Store reflected particle
    call score
!   Start new particle history
    goto 1

!   Sample Collision Kernel

!   Sample Absorption or Scattering if particle is neither reflected
!   nor transmitted
3   call random(rr)
    if(rr-scs/tcs)10,10,11

!   Particle Absorbed
11  iop=3
!   Store Absorbed Particle
    call score
!   Start new particle history
    goto 1

!   Particle scattered
!
!   Increment scattering counter
10  s=s+1.0
!   Sample new scattering angle in center of mass coordinates
!   Isotropic scattering in center of mass system
    call random(rr)
```

```
!   Cosine of the scattering angle
    csa=2.0*rr - 1.0
!   Calculate new particle's energy after scattering
    dum=1.0+aw
    dum=dum*dum
    e1=eas*(1.0+2.0*aw*csa+aw*aw)/dum
    eas=e1
!   Calculate cosine of scattering angle in laboratory system
    cthes=(1.0+aw*csa)/sqrt(1.0+aw*aw+2.0*aw*csa)

!   Sample new azimuthal angle
    call random(rr)
    phis=pi*(2.0*rr-1.0)

!   Calculate new direction cosines
!   Rotate axes

    aa=cthes
    bb=sqrt(1.0-aa*aa)
    c=cos(phis)
!   Check for value of azimuthal angle
    if( phis) 12,12,13
12  d= -sqrt(1.0-c*c)
    goto 14
13  d=  sqrt(1.0-c*c)
!   Check for cut off angle "coa"
14  if((1.0-abs(g))-coa) 15,15,16
15  alfap=bb*c
    bp=bb*d
    gp=aa*g
    goto 17
16  alfap=((bb*c*g*alfa-bb*d*b)/sqrt(1.0-g*g)) + aa*alfa
    bp=((bb*c*g*b + bb*d*alfa)/sqrt(1.0-g*g)) + aa*b
    gp= -(bb*c*sqrt(1.0-g*g)) + aa*g
17  alfa=alfap
    b=bp
    g=gp

!   Go back to a new sampling of the transport kernel
    goto 2

!   Analysis of reflected and transmitted angular radiation
!   Results are calculated per source particle
!   Calculate dose rates
5   do i=1,nang
        cda(i)=0.5*(cos(ang(i))+cos(ang(i+1)))
        cdb(i)=0.5*(cos(bng(i+1))-cos(bng(i)))
    end do
    do i=1,neneg
        do j=1,nang
!   Angular Transmissions
            ft(i,j)=t(i,j)/(nhist*cda(j))
            dt(i,j)=ft(i,j)*ftd(i)
            t(i,j)=t(i,j)/nhist
!   Angular Reflections
            fr(i,j)=r(i,j)/(nhist*cdb(j))
            dr(i,j)=fr(i,j)*ftd(i)
            r(i,j)=r(i,j)/nhist
        end do
    end do
!   Calculate absorption rates
    do i=1,neneg
        do j=1,ndivt
            a(i,j)=a(i,j)/nhist
        end do
    end do

!   Store data and output results

!   Open output file
    open(12,file='output_data')
!   Transmission Results
    write(*,*) 'Transmitted flux, dose and current per source particle'
    write(*,*) 'Flux per source particle (energy,angle)'
    do i=1,neneg
        write(*,60) (ft(i,j),j=1,nang)
```

```
        write(12,60) (ft(i,j),j=1,nang)
60  format(5e14.8)
    end do
    pause
    write(*,*) 'Dose per source particle (energy,angle)'
    do i=1,neneg
        write(*,60) (dt(i,j),j=1,nang)
        write(12,60) (dt(i,j),j=1,nang)
    end do
    pause
    write(*,*) 'Current per source particle (energy,angle)'
    do i=1,neneg
        write(*,60) (t(i,j),j=1,nang)
        write(12,60) (t(i,j),j=1,nang)
    end do
    pause
!   Reflection Results
    write(*,*) 'Reflected flux, dose and current per source particle'
    write(*,*) 'Flux per source particle (energy,angle)'
    do i=1,neneg
        write(*,60) (fr(i,j),j=1,nang)
        write(12,60) (fr(i,j),j=1,nang)
    end do
    pause
    write(*,*) 'Dose per source particle (energy,angle)'
    do i=1,neneg
        write(*,60) (dr(i,j),j=1,nang)
        write(12,60) (dr(i,j),j=1,nang)
    end do
    pause
    write(*,*) 'Current per source particle (energy,angle)'
    do i=1,neneg
        write(*,60) (r(i,j),j=1,nang)
        write(12,60) (r(i,j),j=1,nang)
    end do
    pause
!   Absorption Results
    write(*,*) 'Particles absorbed per source particle'
    write(*,*) 'Absorptions per source particle (energy,position)'
    do i=1,neneg
        write(*,60) (a(i,j),j=1,ndivt)
        write(12,60) (a(i,j),j=1,ndivt)
    end do

    write(*,*) neneg, ndivt
    pause

!   Generate output for plotting routines

!   Generate file for plotting in Excel
    open (unit=111,file='plot_excel.xls',status='unknown')
    do  i=1,neneg
        write(111,333)(a(i,j),tab,j=1,ndivt)
    end do
!   Generate file for plotting in Array visualiser
    open (unit=222, file='plot_array_visualiser.agl',status='unknown')
    do i=1,neneg
        write(222,333)(a(i,j),tab,j=1,ndivt)
    end do
333 format(5(e14.8,a1)
)
    pause

    call exit
    stop
    end

!   Cross sections handling subroutine
    subroutine crosec(tcs,scs,eas,ktest)
!   tcs:    Returned value of total macroscopic cross section (cm-1)
!   scs:    Returned value of macroscopic scattering cross section (cm-1)
!   eas:    Neutron energy at which cross sections are to be calculated (MeV)
    dimension ee(40),tot(40),sca(40)
!   tot(i): Total macroscopic cross section in energy group i  (cm-1)
!   sca(i): Scattering macroscopic cross section in energy group i   (cm-1)
!   ee(i):  Energy groups for cross sections (MeV)
```

```
!   If ktest is not equal to 1, this is the first call to the subroutine
!   When first called, subroutine reads total and scattering cross sections as
!   a function of energy
    if (ktest-1) 10,40,10
10  open(11,file='input data')
!   After first call set ktest equal to unity
    ktest=1
    read(11,20)ncse
20  format(i10)
!   ncse: number of energy groups for cross section data
!   Read energy intervals
    read(11,30)(ee(i), i=1,ncse)
!   Read total group cross sections
    read(11,30)(tot(i), i=1,ncse)
!   Read scattering group cross sections
    read(11,30)(sca(i), i=1,ncse)
30  format(6e10.30)

!   When called after the first time, subroutine uses linear interpolation
!   to return a value of the total cross section "tcs"  and a value of the
!   scattering cross section "scs" at the current neutron energy "eas".
40  if(eas-ee(1)) 50,60,70
50  i=2
    goto 100
60  i=1
    goto 200
70  do 80 i=2,ncse
        if(eas-ee(i)) 100,200,80
80  continue
    i=ncse
100 factor= (eas-ee(i-1))/(ee(i)-ee(i-1))
    scs=sca(i-1)+(sca(i)-sca(i-1))*factor
    goto 300
200 tcs=tot(i)
    scs=sca(i)
!   write(*,*) tcs,scs
!   pause
300 return
    end

!   Scoring Subroutine
    subroutine score
!   This subroutine scores particle transmissions, reflections and absorptions
    common n1,nn1,ndt,iop,eas,ang(6),bng(6),t1(21),e(8), &
          & t(8,6),r(8,6),a(8,20),z,g
!   Determine energy group
    do 6 i=2,n1
        if (eas-e(i))77,77,6
77      ii=i-1
        goto 7
6   continue
7   i=ii
!   Determine whether particle has been transmitted, reflected, or absorbed
    goto (4,5,10),iop
!   iop=1 Particle transmitted
!   iop=2 Particle reflected
!   iop=3 Particle absorbed

!   Particle transmitted
!   Calculate angular bin
4   do 2 j=2,nn1
        if (cos(ang(j))-g)33,33,2
33      jj=j-1
        goto 3
2   continue
3   j=jj
!   Store in transmission angular bins
    t(i,j)=t(i,j)+1.0
    goto 100

!   Particle reflected
!   Calculate angular bin
5   do 12 j=2,nn1
        if (g-cos(bng(j)))73,73,12
73      jj=j-1
```

```
    goto 13
12  continue
13  j=jj
!   Store in reflection angular bins
    r(i,j)=r(i,j)+1.0
    goto 100

!   Particle Absorbed
!   Calculate region bin
10  do 15 k=2,ndt
        if(z-t1(k)) 74,74,15
74      kk=k-1
    goto 14
15  continue
14  k=kk
    a(i,k)=a(i,k)+1.0

100 return
    end

!   Source Sampling Routine
    subroutine source(alfa,beta,gamma,eo,eas,z)
    data pi /3.142593/
!   Sample source's polar angle from a cosine angle between zero and unity
    call random(rr)
    cthe=sqrt(rr)
!   Sample source's azimuthal angle between -pi to +pi
    call random(rr)
    phi=pi*(2.0*rr-1.0)
!   Calculate sine of polar angle
    sthea=sqrt(1.0-cthe*cthe)
!   Calculate direction cosines
    alfa=sthea*cos(phi)
    beta=sthea*sin(phi)
    gamma=cthe
!   Initial source energy
    eas=eo
!   Initial source position
    z=0.0
    return
    end
```