NSEG 5994 – Monte Carlo Methods for Particle Transport

Homework – 6

Nagendra Krishnamurthy

26th October, 2011

Problem 1:

The probabilities obtained for the three angles of particle emission from the source are provided here for the three processes. It is seen that for this ratio of scattering to total cross-section, most of the particles are absorbed. In the next three columns, the obtained FOM values are provided. These will be used for comparison in the subsequent paragraphs.

Case 1						
Initial angle	**Probability**			**FOM**		
	Leaked	**Reflected**	**Absorbed**	**Leaked**	**Reflected**	**Absorbed**
0	5.81E-05	3.51E-02	9.65E-01	8.98E+03	5.64E+06	4.24E+09
30	1.58E-05	3.84E-02	9.62E-01	2.40E+03	6.08E+06	3.81E+09
60	1.50E-06	5.08E-02	9.49E-01	2.45E+02	8.76E+06	3.06E+09
Case 2						
Initial angle	**Probability**			**FOM**		
	Leaked	**Reflected**	**Absorbed**	**Leaked**	**Reflected**	**Absorbed**
0	1.26E-03	2.84E-01	7.14E-01	6.91E+04	2.17E+07	1.36E+08
30	8.39E-04	2.99E-01	7.00E-01	4.66E+04	2.37E+07	1.29E+08
60	4.45E-04	3.68E-01	6.32E-01	2.62E+04	3.43E+07	1.01E+08

Precision:

The table below presents the relative errors and variances calculated for the three different initial angles provided. Firstly, it is noted that the relative error is the maximum for the number of particles leaked (or rather the probability) and is the limiting constraint for stopping the simulations. It is also seen that for the first two angles (= 0 and 30), the relative error for probabilities of all the three types of interactions are less than 0.1 threshold. However, for the third case (angle = 60), the relative error for probability of leaking is not below 0.1 and hence, the maximum limit for number of particle histories became the limiting criteria in this case. Since the values of the probability is high mainly for the absorption case, looking at the relative error and variance for that case, we can say that calculations are precise (variance is around 5% of the value and the relative error is very less as well).

Another point to note is the increase in the variance values for the reflected probability for the case 2, in which the number of reflections is considerably higher and is of the same order as the number of absorptions.

Case 1						
Initial angle	Variance			Relative error		
	Leaked	Reflected	Absorbed	Leaked	Reflected	Absorbed
0	5.81E-05	3.39E-02	3.40E-02	1.00E-01	3.59E-03	1.31E-04
30	1.58E-05	3.69E-02	3.70E-02	1.00E-01	2.09E-03	8.34E-05
60	1.50E-06	4.82E-02	4.82E-02	2.29E-01	1.37E-03	7.31E-05
Case 2						
Initial angle	Variance			Relative error		
	Leaked	Reflected	Absorbed	Leaked	Reflected	Absorbed
0	1.26E-03	2.04E-01	2.04E-01	9.99E-02	5.64E-03	2.25E-03
30	8.38E-04	2.10E-01	2.10E-01	1.00E-01	4.43E-03	1.90E-03
60	4.45E-04	2.33E-01	2.33E-01	1.00E-01	2.76E-03	1.61E-03

Accuracy and FOM:

To prove that the simulations are accurate, we would like to show that the central limit theorem is valid for the number of particle histories considered. In order to prove this, we will compute the FOM for the three different angles, for each of the three processes. We will show that the FOM does not vary when the number of particle histories is increased, as required if the central limit theorem is valid. For this purpose, we removed the constraint that the simulation would stop for maximum relative error going below 0.1. Instead, we ran each of the 3 cases for 10 million and 20 million particle histories and compare the obtained values to the FOM values obtained for the original calculations in which the constraint also included the maximum relative error not exceeding 0.1. From the corresponding FOM values being roughly the same between the simulations, it can be concluded that the central limit theorem is applicable and hence, the number of histories are large enough for the calculations to be accurate.

Case 1						
Initial angle	FOM, 10 million histories			FOM, 20 million histories		
	Leaked	Reflected	Absorbed	Leaked	Reflected	Absorbed
0	8.99E+03	5.60E+06	4.20E+09	9.18E+03	5.60E+06	4.19E+09
30	2.70E+03	6.07E+06	3.79E+09	2.59E+03	6.05E+06	3.80E+09
60	2.59E+02	8.16E+06	2.84E+09	2.47E+02	8.29E+06	2.88E+09
Case 2						
Initial angle	FOM, 10 million histories			FOM, 20 million histories		
	Leaked	Reflected	Absorbed	Leaked	Reflected	Absorbed
0	6.37E+04	2.21E+07	1.38E+08	6.28E+04	2.20E+07	1.38E+08
30	4.67E+04	2.40E+07	1.27E+08	4.62E+04	2.41E+07	1.27E+08
60	2.71E+04	3.44E+07	1.01E+08	2.68E+04	3.45E+07	1.02E+08

Source code:

```fortran
MODULE allData

  TYPE shield
    REAL :: sigmaTotal, sigmaAbs, xMin, xMax, absRatio, thickness
  END TYPE shield

  TYPE (shield), ALLOCATABLE :: shld(:)

  INTEGER :: nParticles, nShields, parStatus, nParAbs, nParRef,      &
      nParLeak, currentRegion, collStatus, freeFlightFlag
  REAL :: xInit, scatterAngleInit(3), xLocation, scatterAngle
  REAL :: probAbs(3), probRef(3), probLeak(3),                       &
      relativeErrorLeak(3), relativeErrorRef(3),                     &
      relativeErrorAbs(3), FOMLeak(3), FOMRef(3), FOMAbs(3),         &
      varianceLeak(3), varianceRef(3), varianceAbs(3)

END MODULE allData


PROGRAM multiRegionShield

  USE allData

  IMPLICIT NONE

  INTEGER :: iParticle, iShield, unitProbFile, iScatterAngle
  REAL :: pi, maxRelativeError, timeIn, timeOut, totalTime
  CHARACTER(80) :: nameProbFile

  ! Initialization
  xInit = 0.0
  pi = DACOS(-1.0)

  CALL RANDOM_SEED()

  nShields = 1
  ALLOCATE(shld(nShields))

  shld(1)%thickness = 1.0
  shld(1)%xMin = xInit
  shld(1)%xMax = shld(1)%xMin + shld(1)%thickness

  shld(1)%sigmaTotal = 10.0
! shld(1)%sigmaAbs = 8.0 ! for ratio Es/Et = 0.2
  shld(1)%sigmaAbs = 2.0 ! for ratio Es/Et = 0.8
  shld(1)%absRatio = shld(1)%sigmaAbs/shld(1)%sigmaTotal

  scatterAngleInit(1) = DCOS(0.0)
  scatterAngleInit(2) = DCOS(pi/6.0)
  scatterAngleInit(3) = DCOS(pi/3.0)
  PRINT*, scatterAngleInit(:)

  nParticles = 20000000

  ! Loop over the three scattering angles
  DO iScatterAngle = 1, 3

    nParLeak = 0
    nParAbs = 0
    nParRef = 0
    maxRelativeError = 100.0

    iParticle = 0
```

```
    CALL CPU_TIME(timeIn)

    DO WHILE ((iParticle .LT. nParticles) .AND.                     &
!       (maxRelativeError .GT. 0.1))
        (maxRelativeError .GT. 0.0))

      iParticle = iParticle + 1
      IF (MOD(iParticle,1000000) .EQ. 0)                              &
          PRINT*, 'particle number:', iParticle

      ! Initial x and mu values
      xLocation = xInit
      scatterAngle = scatterAngleInit(iScatterAngle)
      parStatus = 0
      currentRegion = 1
      collStatus = 0

      DO WHILE (parStatus .EQ. 0)

        CALL freeFlightPL

        IF (parStatus .EQ. 0 .AND. collStatus .EQ. 1)               &
            CALL getInteraction

        IF (parStatus .EQ. 0 .AND. collStatus .EQ. 1)               &
            CALL getScatterAngle

      END DO

      IF ((nParLeak .GT. 0) .AND. (nParRef .GT. 0) .AND.            &
          (nParAbs .GT. 0)) THEN

      probLeak(iScatterAngle) = 1.0*nParLeak/iParticle
      probRef(iScatterAngle) = 1.0*nParRef/iParticle
      probAbs(iScatterAngle) = 1.0*nParAbs/iParticle

      varianceLeak(iScatterAngle) = probLeak(iScatterAngle)*        &
          (1.0-probLeak(iScatterAngle))
      varianceRef(iScatterAngle) = probRef(iScatterAngle)*          &
          (1.0-probRef(iScatterAngle))
      varianceAbs(iScatterAngle) = probAbs(iScatterAngle)*          &
          (1.0-probAbs(iScatterAngle))

      relativeErrorLeak(iScatterAngle) =                            &
          DSQRT(1.0/nParLeak - 1.0/iParticle)
      relativeErrorRef(iScatterAngle) =                             &
          DSQRT(1.0/nParRef - 1.0/iParticle)
      relativeErrorAbs(iScatterAngle) =                             &
          DSQRT(1.0/nParAbs - 1.0/iParticle)

      maxRelativeError = MAX(relativeErrorLeak(iScatterAngle),      &
          relativeErrorRef(iScatterAngle),                          &
          relativeErrorAbs(iScatterAngle))

      END IF

    END DO

    CALL CPU_TIME(timeOut)
    totalTime = timeOut - timeIn
    totalTime = totalTime/60.0 ! Conversion to minutes
```

```
    FOMLeak(iScatterAngle) = 1.0                                      &
        /relativeErrorLeak(iScatterAngle)**2./totalTime
    FOMRef(iScatterAngle) = 1.0                                       &
        /relativeErrorRef(iScatterAngle)**2./totalTime
    FOMAbs(iScatterAngle) = 1.0                                       &
        /relativeErrorAbs(iScatterAngle)**2./totalTime

  END DO

  unitProbFile = 101
  nameProbFile = 'problem_1.dat'

  OPEN (UNIT = unitProbFile, FILE = nameProbFile,                     &
      POSITION = 'append', FORM = 'formatted', ACTION = 'write')
  DO iScatterAngle = 1, 3
    WRITE(unitProbFile,501) iScatterAngle, probLeak(iScatterAngle), &
        probRef(iScatterAngle), probAbs(iScatterAngle),             &
        FOMLeak(iScatterAngle), FOMRef(iScatterAngle),              &
        FOMAbs(iScatterAngle), relativeErrorLeak(iScatterAngle),    &
        relativeErrorRef(iScatterAngle),                            &
        relativeErrorAbs(iScatterAngle),                            &
        varianceLeak(iScatterAngle), varianceRef(iScatterAngle),    &
        varianceAbs(iScatterAngle)
  END DO
  CLOSE(unitProbFile)

501 FORMAT (1(i1.1, 1X), 12(e12.5, 1X))

END PROGRAM multiRegionShield

SUBROUTINE freeFlightPL

  USE allData

  IMPLICIT NONE

  REAL :: randNum, pathLength

  CALL RANDOM_NUMBER(randNum)

  pathLength = -LOG(randNum)/shld(currentRegion)%sigmaTotal
  xLocation = xLocation + pathLength*scatterAngle

  IF ((xLocation .GT. shld(currentRegion)%xMax)) THEN
    xLocation = shld(currentRegion)%xMax
    currentRegion = currentRegion + 1
    IF (currentRegion .GT. nShields) THEN
      parStatus = 1
      nParLeak = nParLeak + 1
    END IF
  ELSE IF (xLocation .LT. shld(currentRegion)%xMin) THEN
    xLocation = shld(currentRegion)%xMin
    currentRegion = currentRegion - 1
    IF (currentRegion .LT. 1) THEN
      parStatus = 2
      nParRef = nParRef + 1
    END IF
  ELSE
    collStatus = 1
  END IF

END SUBROUTINE freeFlightPL
```

```
SUBROUTINE getInteraction

  USE allData

  IMPLICIT NONE

  REAL :: randNum

  CALL RANDOM_NUMBER(randNum)

  IF (randNum .LT. shld(currentRegion)%absRatio) THEN
    parStatus = 3
    nParAbs = nParAbs + 1
  END IF

END SUBROUTINE getInteraction

SUBROUTINE getScatterAngle

  USE allData

  IMPLICIT NONE

  REAL :: randNum, pi, mu_0, phi_0, mu_prime

  pi = DACOS(-1.0)

  CALL RANDOM_NUMBER(randNum)
  mu_0 = 2.0*randNum - 1.0

  CALL RANDOM_NUMBER(randNum)
  phi_0 = 2.0*pi*randNum

  mu_prime = scatterAngle*mu_0                                           &
      + DSQRT(1.0-scatterAngle**2.0)                                     &
       *DSQRT(1.0-mu_0**2.0)                                             &
       *DCOS(phi_0)

  scatterAngle = mu_prime

  collStatus = 0

END SUBROUTINE getScatterAngle
```

Problem 2:

Tables presented here are in the same order as in problem 1 for easier comparison.

Firstly, the total probabilities are checked to be sure that we are effectively obtaining the same results. Next, we will analyze the three features – precision, accuracy and FOM.

Case 1						
Initial angle	**Probability**			**FOM**		
	Leaked	**Reflected**	**Absorbed**	**Leaked**	**Reflected**	**Absorbed**
0	6.59E-05	3.53E-02	9.65E-01	2.47E+03	7.54E+06	5.59E+09
30	1.67E-05	3.83E-02	9.62E-01	8.06E+02	8.47E+06	5.33E+09
60	1.68E-06	5.09E-02	9.49E-01	4.81E+02	1.37E+07	4.75E+09

Case 2						
Initial angle	Probability			FOM		
	Leaked	Reflected	Absorbed	Leaked	Reflected	Absorbed
0	1.11E-03	2.88E-01	7.11E-01	5.31E+04	8.95E+06	5.50E+07
30	7.84E-04	3.01E-01	6.98E-01	6.19E+04	1.11E+07	5.99E+07
60	5.15E-04	3.66E-01	6.34E-01	6.32E+04	1.84E+07	5.53E+07

Precision:

With regards to the relative error, it is found that for the same relative error in the probability of leaked particles (which obviously becomes the constraint), the relative errors (and variances) are lesser for the other two processes in Case 1, while in Case 2 they are higher when implicit capturing is used. It is noted that Case 1 is absorption dominant while in Case 2, both absorption and reflection are of the same order indicating that implicit capture scheme does help in reducing the variance and hence improving the precision in absorption dominant processes!

Case 1						
Initial angle	Variance			Relative error		
	Leaked	Reflected	Absorbed	Leaked	Reflected	Absorbed
0	5.39E-05	5.06E-03	5.11E-03	9.96E-02	1.80E-03	6.62E-05
30	1.08E-05	5.40E-03	5.41E-03	9.97E-02	9.73E-04	3.88E-05
60	2.07E-07	6.64E-03	6.64E-03	9.99E-02	5.93E-04	3.18E-05
Case 2						
Initial angle	Variance			Relative error		
	Leaked	Reflected	Absorbed	Leaked	Reflected	Absorbed
0	2.36E-04	9.37E-02	9.33E-02	1.00E-01	7.70E-03	3.11E-03
30	1.15E-04	9.48E-02	9.44E-02	1.00E-01	7.48E-03	3.21E-03
60	5.83E-05	1.01E-01	1.01E-01	1.00E-01	5.86E-03	3.38E-03

Accuracy and FOM:

With respect to the FOM values, an almost constant value of FOM is obtained just like in the problem 1 indicating the sufficiency of the number of particle histories considered.

Case 1						
Initial angle	FOM, 10 million histories			FOM, 20 million histories		
	Leaked	Reflected	Absorbed	Leaked	Reflected	Absorbed
0	2.27E+03	7.55E+06	5.62E+09	2.21E+03	7.56E+06	5.62E+09
30	8.26E+02	8.52E+06	5.36E+09	7.45E+02	8.53E+06	5.35E+09
60	4.45E+02	1.37E+07	4.77E+09	4.81E+02	1.37E+07	4.77E+09
Case 2						

Initial angle	FOM, 10 million histories			FOM, 20 million histories		
	Leaked	Reflected	Absorbed	Leaked	Reflected	Absorbed
0	5.37E+04	9.58E+06	6.01E+07	5.46E+04	9.57E+06	6.02E+07
30	5.45E+04	1.11E+07	5.85E+07	5.43E+04	1.11E+07	5.85E+07
60	4.95E+04	1.87E+07	5.54E+07	5.00E+04	1.87E+07	5.53E+07

Source code:

```
MODULE allData

  TYPE shield
    REAL :: sigmaTotal, sigmaAbs, xMin, xMax, absRatio, thickness
  END TYPE shield

  TYPE (shield), ALLOCATABLE :: shld(:)

  INTEGER :: nParticles, nShields, parStatus, nParAbs, nParRef,      &
      nParLeak, currentRegion, collStatus, freeFlightFlag
  REAL :: xInit, scatterAngleInit(3), xLocation, scatterAngle,      &
      parRef, parLeak, parAbs, weight, weightCutoff
  REAL :: probAbs(3), probRef(3), probLeak(3),                      &
      relativeErrorLeak(3), relativeErrorRef(3),                    &
      relativeErrorAbs(3), FOMLeak(3), FOMRef(3), FOMAbs(3),        &
      varianceLeak(3), varianceRef(3), varianceAbs(3)

END MODULE allData


PROGRAM implicitCapture

  USE allData

  IMPLICIT NONE

  INTEGER :: iParticle, iShield, unitProbFile, iScatterAngle,       &
      rrd, flag
  REAL :: pi, probLeakSq, probRefSq, probAbsSq, timeIn, timeOut,    &
      totalTime, maxRelativeError
  CHARACTER(80) :: nameProbFile

  ! Initialization
  xInit = 0.0
  pi = DACOS(-1.0)

  CALL RANDOM_SEED()

  nShields = 1
  ALLOCATE(shld(nShields))

  shld(1)%thickness = 1.0
  shld(1)%xMin = xInit
  shld(1)%xMax = shld(1)%xMin + shld(1)%thickness

  shld(1)%sigmaTotal = 10.0
! shld(1)%sigmaAbs = 8.0 ! for ratio Es/Et = 0.2
  shld(1)%sigmaAbs = 2.0 ! for ratio Es/Et = 0.8
  shld(1)%absRatio = shld(1)%sigmaAbs/shld(1)%sigmaTotal

  scatterAngleInit(1) = DCOS(0.0)
```

```
  scatterAngleInit(2) = DCOS(pi/6.0)
  scatterAngleInit(3) = DCOS(pi/3.0)

  rrd = 5
  weightCutoff = 1.0E-7
  nParticles = 20000000

  ! Loop over the three scattering angles
  DO iScatterAngle = 1, 3

    nParLeak = 0
    nParAbs = 0
    nParRef = 0
    parLeak = 0.0
    parRef = 0.0
    parAbs = 0.0
    probLeakSq = 0.0; probRefSq = 0.0; probAbsSq = 0.0
    maxRelativeError = 100.0

    iParticle = 0

    CALL CPU_TIME(timeIn)

    DO WHILE ((iParticle .LT. nParticles) .AND.                     &
!        (maxRelativeError .GT. 0.1))
         (maxRelativeError .GT. 0.0))

      iParticle = iParticle + 1
      IF (MOD(iParticle,1000000) .EQ. 0)                             &
          PRINT*, 'particle number:', iParticle

      xLocation = xInit
      scatterAngle = scatterAngleInit(iScatterAngle)
      parStatus = 0
      currentRegion = 1
      collStatus = 0
      weight = 1.0

      DO WHILE (parStatus .EQ. 0)

        CALL freeFlightPL

        IF (parStatus .EQ. 0 .AND. collStatus .EQ. 1)               &
            CALL getInteraction

        IF (weight .lt. weightCutoff) THEN
          CALL russianRoulette(rrd,flag)
          IF (flag .eq. 0) THEN
            parStatus = 3
          ELSE
            weight = weight*rrd
          END IF
        END IF

        IF (parStatus .EQ. 0 .AND. collStatus .EQ. 1)               &
            CALL getScatterAngle

      END DO

      IF (parStatus .EQ. 1) THEN
        probLeakSq = probLeakSq + weight**2.0
        relativeErrorLeak(iScatterAngle) = DSQRT(                   &
            probLeakSq/parLeak**2.0 - 1.0/iParticle)
```

```fortran
      ELSE IF (parStatus .EQ. 2) THEN
        probRefSq = probRefSq + weight**2.0
        relativeErrorRef(iScatterAngle) = DSQRT(                    &
            probRefSq/parRef**2.0 - 1.0/iParticle)
      END IF
      probAbsSq = probAbsSq + (1.0-weight)**2.0
      parAbs = parAbs + (1.0-weight)
      relativeErrorAbs(iScatterAngle) = DSQRT(                      &
          probAbsSq/parAbs**2.0 - 1.0/iParticle)

      IF ((parLeak .GT. 0.00001) .AND. (parRef .GT. 0.00001) .AND.  &
          (parAbs .GT. 0.00001)) THEN
      maxRelativeError = MAX(relativeErrorLeak(iScatterAngle),      &
          relativeErrorRef(iScatterAngle),                          &
          relativeErrorAbs(iScatterAngle))
      END IF

    END DO

    CALL CPU_TIME(timeOut)
    totalTime = timeOut - timeIn
    totalTime = totalTime/60.0 ! Conversion to minutes

    probLeak(iScatterAngle) = 1.0*parLeak/iParticle
    probRef(iScatterAngle) = 1.0*parRef/iParticle
    probAbs(iScatterAngle) = 1.0-probLeak(iScatterAngle)            &
        -probRef(iScatterAngle)

    varianceLeak(iScatterAngle) = probLeakSq/iParticle              &
        - probLeak(iScatterAngle)**2.0
    varianceRef(iScatterAngle) = probRefSq/iParticle                &
        - probRef(iScatterAngle)**2.0
    varianceAbs(iScatterAngle) = probAbsSq/iParticle                &
        - probAbs(iScatterAngle)**2.0

    FOMLeak(iScatterAngle) = 1.0                                    &
        /relativeErrorLeak(iScatterAngle)**2./totalTime
    FOMRef(iScatterAngle) = 1.0                                     &
        /relativeErrorRef(iScatterAngle)**2./totalTime
    FOMAbs(iScatterAngle) = 1.0                                     &
        /relativeErrorAbs(iScatterAngle)**2./totalTime

  END DO

  unitProbFile = 101
  nameProbFile = 'problem_2.dat'

  OPEN (UNIT = unitProbFile, FILE = nameProbFile,                   &
      POSITION = 'append', FORM = 'formatted', ACTION = 'write')
  DO iScatterAngle = 1, 3
    WRITE(unitProbFile,501) iScatterAngle, probLeak(iScatterAngle), &
        probRef(iScatterAngle), probAbs(iScatterAngle),             &
        FOMLeak(iScatterAngle), FOMRef(iScatterAngle),              &
        FOMAbs(iScatterAngle), relativeErrorLeak(iScatterAngle),    &
        relativeErrorRef(iScatterAngle),                            &
        relativeErrorAbs(iScatterAngle),                            &
        varianceLeak(iScatterAngle), varianceRef(iScatterAngle),    &
        varianceAbs(iScatterAngle)
  END DO
  CLOSE(unitProbFile)

501 FORMAT (1(i1.1, 1X), 12(e12.5, 1X))
```

```
END PROGRAM implicitCapture

SUBROUTINE freeFlightPL

  USE allData

  IMPLICIT NONE

  REAL :: randNum, pathLength

  CALL RANDOM_NUMBER(randNum)

  pathLength = -LOG(randNum)/shld(currentRegion)%sigmaTotal
  xLocation = xLocation + pathLength*scatterAngle

  IF ((xLocation .GT. shld(currentRegion)%xMax)) THEN
    xLocation = shld(currentRegion)%xMax
    currentRegion = currentRegion + 1
    IF (currentRegion .GT. nShields) THEN
      parStatus = 1
      nParLeak = nParLeak + 1
      parLeak = parLeak + weight
    END IF
  ELSE IF (xLocation .LT. shld(currentRegion)%xMin) THEN
    xLocation = shld(currentRegion)%xMin
    currentRegion = currentRegion - 1
    IF (currentRegion .LT. 1) THEN
      parStatus = 2
      nParRef = nParRef + 1
      parRef = parRef + weight
    END IF
  ELSE
    collStatus = 1
  END IF

END SUBROUTINE freeFlightPL


SUBROUTINE getInteraction

  USE allData

  IMPLICIT NONE

  REAL :: randNum

  weight = weight                                                 &
      *(1.0-(shld(currentRegion)%sigmaAbs                         &
      /shld(currentRegion)%sigmaTotal))

END SUBROUTINE getInteraction

SUBROUTINE getScatterAngle

  USE allData

  IMPLICIT NONE

  REAL :: randNum, pi, mu_0, phi_0, mu_prime

  pi = DACOS(-1.0)

  CALL RANDOM_NUMBER(randNum)
```

```
  mu_0 = 2.0*randNum - 1.0

  CALL RANDOM_NUMBER(randNum)
  phi_0 = 2.0*pi*randNum

  mu_prime = scatterAngle*mu_0                                           &
      + DSQRT(1.0-scatterAngle**2.0)                                     &
       *DSQRT(1.0-mu_0**2.0)                                             &
       *DCOS(phi_0)

  scatterAngle = mu_prime

  collStatus = 0

END SUBROUTINE getScatterAngle

SUBROUTINE russianRoulette(rrd,flag)

  USE allData

  IMPLICIT NONE

  INTEGER, INTENT(IN) :: rrd
  INTEGER, INTENT(OUT) :: flag
  REAL :: randNum

  CALL RANDOM_NUMBER(randNum)
  IF (randNum .GT. 1.0/rrd) THEN
    flag = 0
  ELSE
    flag = 1
  END IF

END SUBROUTINE russianRoulette
```