

線源の作り方実習  
形状 (cg の円筒形状)  
(September 9, 2005, Draft)

平山 英夫、波戸 芳仁

〒 305-0801 茨城県つくば市大穂 1 - 1  
高エネルギー加速器研究機構

## Contents

1. 基にするユーザーコード <code>ucsource.f</code> の概要	1
2. 実習課題 1 : 線源を Ir-192 の $\gamma$ 線源に変更する	1
2.1. <code>if</code> 文を使用する方法	1
2.2. <code>data</code> 文を使用する方法	2
3. 実習課題 2 : $^{90}\text{Sr}$ - $\beta$ 線源	2
4. 実習課題 3 : 面線源 ( $1.5\text{cm} \leq r \leq 4.0\text{cm}$ )	5
4.1. 直接サンプリング	5
4.2. Rejection 法	6
5. 実習課題 4 : 等方線源 ( $2\pi$ )	7
5.1. 直接サンプリング	7
5.2. Rejection 法	8

## 1. 基にするユーザーコード `ucsource.f` の概要

形状としては、`ucnaicgv.f` で使用している `cg` を用いた円筒形状である。各種の線源のテストを行うことを目的にしているので、物質は全て真空 (0) に設定している。単一エネルギー (1.253MeV) の光子が、原点にビーム状に入力する様に設定されている。

## 2. 実習課題 1 : 線源を Ir-192 の $\gamma$ 線源に変更する

Ir-192 から放出される  $\gamma$  線のエネルギーと崩壊当たりの放出率は、以下である。

Energy	Emission Rate	PDF	CDF
0.296	28.7	0.1385	0.1385
0.308	30	0.1448	0.2833
0.317	82.7	0.3991	0.6824
0.468	47.8	0.2307	0.9131
0.589	4.5	0.0217	0.9348
0.604	8.2	0.0396	0.9744
0.612	5.3	0.0256	1.0000

放出率から、確率密度関数 (PDF) 及び累積分布関数 (CDF) を計算すると、上記の表の 3 及び 4 カラムの値となる。

### 2.1. `if` 文を使用する方法

1. `cp ucsource.f ucsource1.f`
2. `ucsource1.f` の変更

- 372 行の

```
ekin = ekein
```

を以下の様に変更する。

```
call RANDOMSET(rnnow)
if (rnnow.lt.0.1385) then
  ekin=0.296
elseif (rnnow.lt.0.2833) then
  ekin=0.308
else if (rnnow.lt.0.6824) then
  ekin=0.317
else if (rnnow.lt.0.9131) then
  ekin=0.468
else if (rnnow.lt.0.9348) then
  ekin=0.589
else if (rnnow.lt.0.9744) then
  ekin=0.604
else
  ekin=0.612
end if
```

3. `cp ucsource.data ucsource1.data`
4. `cp ucsource.inp ucsource1.inp`
5. `ucsource1.f` を `egs5run` で実行する。  
ユニット 4 のファイル名及びユニット 25 のファイル名は、何も入力しないでリターンする。

6. モードの選択で、1の計算モードを選ぶ。
7. ヒストリー数として、10000を入力する。
8. egs5job.outの光子のスペクトルが、上記の表の3カラムに対応する率になっていることを確認する。

## 2.2. data文を使用する方法

1. cp ucsource1.f ucsource2.f
2. ucsource2.fの変更

- 122行の後に、以下を挿入する。

```
real*8  esbin(7),escdf(7)
data esbin/0.296,0.308,0.317,0.468,0.589,0.604,0.612/
data escdf/0.1385,0.2833,0.6824, 0.9131, 0.9348, 0.9744,1.0/
```

- 265行の

```
ekein=1.333      ! Kinetic energy
```

を

```
ekein=esbin(7)   ! Kinetic energy
```

に変更する。

- 376-391行を以下の様に修正する。

```
call randomset(rnnow)
do ie=1,7
  if(rnnow.le.escdf(ie)) go to 1000
end do
1000  ekin=esbin(ie)
```

3. cp ucsource1.data ucsource2.data
4. cp ucsource1.inp ucsource2.inp
5. ucsource2.fをegs5runで実行する。  
ユニット4のファイル名及びユニット25のファイル名は、何も入力しないでリターンする。
6. モードの選択で、1の計算モードを選ぶ。
7. ヒストリー数として、10000を入力する。
8. egs5job.outの光子のスペクトルが、先の表の3カラムに対応する率になっていることを確認する。

## 3. 実習課題2：<sup>90</sup>Sr-β線源

β線源は、γ線源と異なり、スペクトルは連続である。連続型の過程のサンプリングでは、一般には直接サンプリングは難しい。近似的な方法であるが、スペクトルの形が与えられている場合にどのような場合にも適用できる方法は、横軸(この場合は、エネルギー)を等間隔に区分し、その区間の積分値の全領域の積分値に対する割合を確率密度関数とし、乱数により対応するエネルギー区間をサンプリングし、エネルギー区間内では、一様分布として直線内挿によりエネルギーを決定する方法である。積分が困難な場合には、区間内の変化が直線であると仮定して台形公式を使用する。この場合、精度を上げるには、分点数を多くすると共に、対応する値を理論値等からできるだけ精度良く求める必要がある。

この方法を理解するために、 $^{90}\text{Sr}$  の  $\beta$  線を例にしてサンプリングルーチンを作成する。ICRU Report 56 には、 $^{90}\text{Sr}$  の  $\beta$  線スペクトルが、(エネルギー/最大エネルギー)を 41 等分した各区分当たりの崩壊当たりの  $\beta$  線数で与られている。(次表及び図)このデータを使用して  $\beta$  線のエネルギーを決定するルーチンを作成する。

$E/E_{max}$	$\beta$ per dis. per bin	$E/E_{max}$	$\beta$ per dis. per bin
0.00	1.597	0.025	1.538
0.05	1.532	0.075	1.526
0.10	1.518	0.125	1.509
0.15	1.500	0.175	1.490
0.20	1.479	0.225	1.466
0.25	1.453	0.275	1.439
0.30	1.422	0.325	1.404
0.35	1.384	0.375	1.361
0.40	1.335	0.425	1.306
0.45	1.274	0.475	1.238
0.50	1.198	0.525	1.154
0.55	1.106	0.575	1.053
0.60	0.997	0.625	0.935
0.65	0.870	0.675	0.801
0.70	0.729	0.725	0.654
0.75	0.577	0.775	0.498
0.80	0.420	0.825	0.343
0.85	0.268	0.875	0.198
0.90	0.135	0.925	0.081
0.95	0.038	0.975	0.010
1.00	0.000		

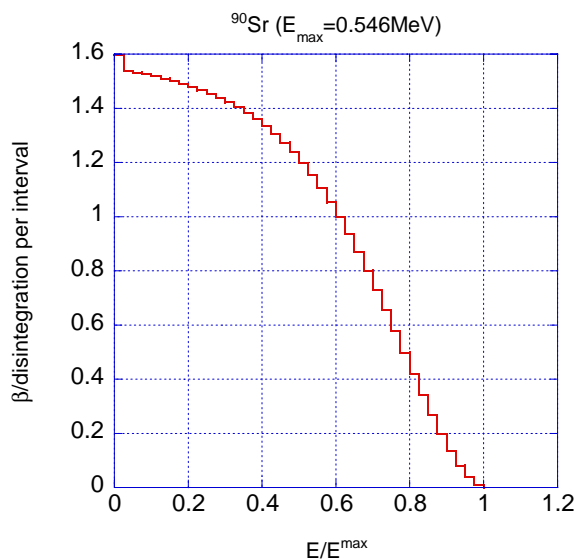


Figure 1:  $\beta$ -ray spectrum.

1. cp ucsource2.f ucsource3.f

## 2. ucsource3.fの変更

- 121 行に, deltaes, emax, tnum を追加する。
- 123-125 行を以下のように変更する。Fortran では、6 カラムの文字があると継続行の意味になる。以下の\*は、この継続行を示すために使用しているので、必ず 6 カラムに書く。

```
real*8  esbin(41), espdf(41), escdf(41)
data espdf/1.597,1.538 ,1.532,1.526 ,1.518,1.509 ,1.500,1.490 ,
*         1.479,1.466 ,1.453,1.439 ,1.422,1.404 ,1.384,1.361 ,
*         1.335,1.306 ,1.274,1.238 ,1.198,1.154 ,1.106,1.053 ,
*         0.997,0.935 ,0.870,0.801 ,0.729,0.654 ,0.577,0.498 ,
*         0.420,0.343 ,0.268,0.198 ,0.135,0.081 ,0.038,0.010 ,
*         0.000/
```

- 267 行の後に、以下を挿入する。

```
emax=0.546
deltaes=0.025
do i=1,41
  esbin(i)=(i-1)*deltaes*emax
end do
```

- 273,274 行の

```
iqin=0           ! Incident charge - photons
ekein=esbin(7)   ! Kinetic energy
```

を

```
iqin=-1          ! Incident charge - electrons
ekein=esbin(41)  ! Kinetic energy
```

に変更する。

- 283 行の後に

```
!-----
! Calculate CDF from pdf
!-----
tnum=0.D0
do ie=1,41
  tnum=tnum+espdf(ie)
end do

do ie=1,41
  if(ie.eq.1) then
    escdf(ie)=0.0
  else
    escdf(ie)=escdf(ie-1)+espdf(ie-1)/tnum
  end if
end do
```

を挿入する。

- 370 行の

```
deltae=ebin(13)/50
```

を

```
deltae=emax/50
```

に変更する。

- 402-405 行の

```

do ie=1,7
  if(rnnow.le.escdf(ie)) go to 1000
end do
1000  ekin=esbin(ie)

```

を以下の様に修正する。

```

do ie=2,41
  if(rnnow.le.escdf(ie)) go to 1000
end do
1000  ekin=esbin(ie-1)+(rnnow-escdf(ie-1))*(esbin(ie)-esbin(ie-1))/
*      (escdf(ie)-escdf(ie-1))

```

に変更する。

3. cp ucsource2.data ucsource3.data

4. cp ucsource2.inp ucsource3.inp

5. ucsource3.f を egs5run で実行する。

ユニット 4 のファイル名及びユニット 25 のファイル名は、何も入力しないでリターンする。

6. モードの選択で、1 の計算モードを選ぶ。

7. ヒストリー数として、10000 を入力する。

8. egs5job.out の電子のスペクトルが、連続分布になっていることを確認する。

#### 4. 実習課題 3 : 面線源 ( $1.5\text{cm} \leq r \leq 4.0\text{cm}$ )

半径 1.5cm から 4cm の領域に一様に分布している面線源の場合に、線源位置をサンプリングするルーチンを作成する。

##### 4.1. 直接サンプリング

半径  $R_0$  から  $R_1$  の領域に一様に分布している面線源の場合の半径の分布に関する確率密度関数(pdf) は、次のようになる。

$$f(r)dr = c \times 2\pi r dr$$

$$\int_{R_0}^{R_1} f(\xi)d\xi = c\pi[\xi^2] = c\pi[R_1^2 - R_0^2] = 1$$

$$c = \frac{1}{\pi(R_1^2 - R_0^2)} \rightarrow f(r)dr = \frac{2rdr}{R_1^2 - R_0^2}$$

線源位置の半径 ( $r$ ) は、以下の式を解くことにより決定する。

$$\eta = \int_{R_0}^r f(\xi)d\xi = \frac{r^2 - R_0^2}{R_1^2 - R_0^2}$$

$$r = \sqrt{R_0^2 + \eta(R_1^2 - R_0^2)}$$

$x$  と  $y$  の位置は、 $\phi$  を、0 から  $2\pi$  の一様分布から決定し、

$$x = r \cos \phi, \quad y = r \sin \phi$$

により決定する。具体的なプログラムは、以下のようにする。

1. cp ucsource3.f ucsource4.f

2. ucsource4.f の変更

- 121 行に, r02, r12, phai を追加する。
- 283 行の後に以下を挿入する。

```
r02=1.5*1.5
r12=4.0*4.0
```

- 422 行の後に以下を挿入する。

```
call RANDOMSET(rnnow)
rr0=sqrt(r02+rnnow*(r12-r02))
call RANDOMSET(rnnow)
phai=PI*(2.0*rnnow-1.0)
xin=rr0*cos(phai)
yin=rr0*sin(phai)
```

3. cp ucsource3.data ucsource4.data

4. cp ucsource3.inp ucsource4.inp

5. ucsource4.f を egs5run で実行する。

ユニット 4 のファイル名及びユニット 25 のファイル名は、何も入力しないでリターンする。

6. モードの選択で、0 の飛跡表示モードを選ぶ。

7. ヒストリー数として、500 を入力する。

8. CGView で、座標軸を X-Y にし、軸を若干傾け、半径 1.5-4.0 の領域から電子が出ていることを確認する。

#### 4.2. Rejection 法

Rejection 法では、x 及び y をそれぞれ -1 から 1 の範囲の正方形内で一様にサンプリングし、(x,y) が線源領域内であれば、それを線源位置とし、領域外であれば、サンプリングをやり直す (新たな乱数を用いてサンプリングすることにより、位置を決定する。具体的なプログラムは、以下のようにする。

1. cp ucsource4.f ucsource5.f

2. ucsource5.f の変更

- 121 行の r02, r12 を r0, r1 に変更する。
- 284-285 行の

```
r02=1.5*1.5
r12=4.0*4.0
```

を以下のように変更する。

```
r0=1.5
r1=4.0
```

- 422-427 行を以下の様に変更する。

```
1100 call randomset(rnnow)
      xi0=2.0*rnnow-1.0
      call randomset(rnnow)
      yi0=2.0*rnnow-1.0
      rr0=sqrt(xi0*xi0+yi0*yi0)
      if (rr0.gt.1.0.or.rr0.lt.r0/r1) go to 1100
      xin =r1*xi0
      yin =r1*yi0
```



3. cp ucsource4.data ucsource5.data
4. cp ucsource4.inp ucsource5.inp
5. ucsource5.f を egs5run で実行する。  
ユニット 4 のファイル名及びユニット 25 のファイル名は、何も入力しないでリターンする。
6. モードの選択で、0 の飛跡表示モードを選ぶ。
7. ヒストリー数として、500 を入力する。
8. CGView で、直接サンプリングと同じように半径 1.5-4cm の領域から電子が出ていることを確認する。

## 5. 実習課題 4 : 等方線源 ( $2\pi$ )

### 5.1. 直接サンプリング

等方線源の場合、Z 方向の方向余弦である  $w$  の確率密度関数は、以下の様になる。

$$f(\theta)d\theta = c \times 2\pi \sin \theta d\theta \quad (0 \leq \theta \leq \pi)$$

$$w = \cos \theta$$

とすると

$$\frac{dw}{d\theta} = -\sin \theta \rightarrow g(w) = -c \times 2\pi dw$$

$$\int_1^{-1} g(w)dw = -c \times 2\pi \times (-2) = 1$$

から

$$c = \frac{1}{4\pi} \rightarrow g(w)dw = -\frac{1}{2}dw$$

となる。 $w$  は、以下の式を解くことにより決定することができる。

$$\eta = \int_1^w g(w)dw = \frac{1}{2}(1-w) \rightarrow w = 1 - 2\eta$$

$1 - 2\eta$  と  $2\eta - 1$  は、等価なので、どちらを使用しても良い。この問題のように  $\cos \theta$  が正の領域にのみ等方線源の場合は、

$$\int_1^0 g(w)dw = -c \times 2\pi \times (-1) = 1$$

から

$$c = \frac{1}{2\pi} \rightarrow g(w)dw = -dw$$

となるので、

$$\eta = \int_1^w g(w)dw = w \rightarrow w = 1 - \eta$$

$1 - \eta$  と  $\eta$  は、等価なので、 $w = \eta$  とする。となる。実際のプログラムは、以下の様にする。

1. cp ucsource.f ucsource6.f
2. ucsource6.f の変更
  - 121 行に ,phai を追加する。
  - 382 行の後に以下を挿入する。

```

call randomset(rnnow)
win=rnnow
write(6,*) 'win=',win
call randomset(rnnow)
phai=PI*(2.0*rnnow-1.0)
uin=dsqrt(1.0-win*win)*cos(phai)
vin=dsqrt(1.0-win*win)*sin(phai)

```

3. cp ucsource.data ucsource6.data
4. cp ucsource.inp ucsource6.inp
5. ucsource6.f を egs5run で実行する。  
ユニット 4 のファイル名及びユニット 25 のファイル名は、何も入力しないでリターンする。
6. モードの選択で、0 の飛跡表示モードを選ぶ。
7. ヒストリー数として、500 を入力する。
8. CGView で、光子が  $2\pi$  方向に等方的に発生していることを確認する。

## 5.2. Rejection 法

Rejection 法では、 $x,y$  及び  $z$  をそれぞれ -1 から 1 の立方体中で一様にサンプリングし、サンプリングされた位置が半径 1 の球の内側の場合は、サンプリングされた位置の半径に対する  $x,y$  及び  $z$  の比をそれぞれの方向余弦とする。球の外側の場合は、サンプリングをやり直す。実際のプログラムは、以下のようにする。

1. cp ucsource6.f ucsource7.f

2. ucsource7.f の変更

- 383-389 行を以下の様に修正入する。

```

1300    call randomset(rnnow)
        zi0=rnnow
        call randomset(rnnow)
        xi0=2.0*rnnow-1.0
        call randomset(rnnow)
        yi0=2.0*rnnow-1.0
        rr0=dsqrt(xi0*xi0+yi0*yi0+zi0*zi0)
        if(rr0.gt.1.0) go to 1300
        win = zi0/rr0
        uin = xi0/rr0
        vin = yi0/rr0

```

3. cp ucsource6.data ucsource7.data
4. cp ucsource6.inp ucsource7.inp
5. ucsource7.f を egs5run で実行する。  
ユニット 4 のファイル名及びユニット 25 のファイル名は、何も入力しないでリターンする。
6. モードの選択で、0 の飛跡表示モードを選ぶ。
7. ヒストリー数として、500 を入力する。
8. CGView で、光子が直接サンプリングの場合と同じように  $2\pi$  方向に等方的に発生していることを確認する。



```

include 'include/egs5_misc.f'
include 'include/egs5_thresh.f'
include 'include/egs5_uphiot.f'
include 'include/egs5_useful.f'
include 'include/egs5_usersc.f'
include 'include/egs5_userxt.f'
include 'include/randomm.f'

!
! -----
! Auxiliary-code COMMONs
! -----
include 'auxcommons/aux_h.f'      ! Auxiliary-code "header" file

include 'auxcommons/edata.f'
include 'auxcommons/etaly1.f'
include 'auxcommons/instuf.f'
include 'auxcommons/lines.f'
include 'auxcommons/nfac.f'
include 'auxcommons/watch.f'

include 'auxcommons/etaly2.f'      ! Added SJW for energy balance

!
! -----
! cg related COMMONs
! -----
include 'auxcommons/geom_common.f' ! geom-common file
integer irinn

common/totals/
* deltae,spg(50),spe(50),imode      ! Variables to score
real*8 deltae,spg,spe,spp
integer imode

!**** real*8                                ! Arguments
real*8 totke
real*8 rnnow,etot
real*8 esumt

real*8                                ! Local variables
* availke,avspe,avspg,avsp,desci2,ekin,ekinm,
* rr0,sigspg,sigspe,sigspp,tef,wtin,wtsm,
* xi0,yi0,zi0

real                                ! Local variables
* elow,eup,rdet,rtcov,rtgap,tcov,tdet,tgap

real
* tarray(2),tt,tt0,tt1,cputime

integer
* i,icases,idin,ie,ifti,ifto,ii,iiz,imed,ireg,isam,
* isot,izn,nlist,j,k,n,ndet,nd,nbatch,ncaspb,
* ner,nofbat

character*24 medarr(1)

!
! -----
! Open files
! -----
! -----
! Units 7-26 are used in pegs and closed. It is better not
! to use as output file. If they are used, they must be opened
! after getcg etc. Unit for pict must be 39.
! -----

open(1,FILE='egs5job.out',STATUS='unknown')
open(UNIT= 4,FILE='egs5job.inp',STATUS='old')
open(39,FILE='egs5job.pic',STATUS='unknown')

!
! =====
! call counters_out(0)
! =====

! -----
! Step 2: pegs5-call
! -----
! =====
! call block_set                                ! Initialize some general variables
! =====

```

```

! -----
! Define media before calling PEGS5
! -----

nmed=1
medarr(1)='NAI'

do j=1,nmed
  do i=1,24
    media(i,j)=medarr(j)(i:i)
  end do
end do

chard(1) = 3.81d0      ! optional, but recommended to invoke
                    ! automatic step-size control

write(1,*) 'chard =',(chard(j),j=1,1)

! -----
! Run KEK PEGS5 before calling HATCH
! -----
write(1,100)
100  FORMAT(' PEGS5-call comes next'//)

! =====
! call pegs5
! =====

! -----
! Step 3: Pre-hatch-call-initialization
! -----

! Initialize cg related parameter
! -----
npreci=2      ! PICT data mode for CGView

itbody=0
irppin=0
isphin=0
irccin=0
itorin=0
itrcin=0
izonin=0
itverr=0
igmmax=0
ifti = 4      ! Input unit number for cg-data
ifto = 39     ! Output unit number for PICT

write(39,110)
110  FORMAT('CSTA')
call geomgt(ifti,ifto)
write(39,120)
120  FORMAT('CEND')

! -----
! Get nreg from cg input data
! -----
nreg=izonin
if (nreg.gt.mxreg) then
  write(1,130) nreg,mxreg
130  FORMAT(' NREG(=',I12,') must be less than MXREG(=',I12,')' /
* ' You must change MXREG in include/egs5_h.f.')
  stop
end if
! Set medium index for each region
! Vacuum region

do i=1,nreg
  med(i)=0    ! Set all region to vacuum
end do

! -----
! Set parameter estepe and estepe2
! -----
estepe=0.10
estepe2=0.20
write(1,140) estepe, estepe2
140  FORMAT(1X,'ESTEPE at EKMAX: ',F10.5,' (estepe)',
*      /,1X,'ESTEPE at ECUT: ',F10.5,' (estepe2)')

```

```

! -----
! Random number seeds. Must be defined before call hatch
! or defaults will be used. inseed (1- 2^31)
! -----
luxlev = 1
inseed=1
write(1,150) inseed
150 FORMAT(/,' inseed=',I12,5X,
* (seed for generating unique sequences of Ranlux)')

! =====
! call rlxinit ! Initialize the Ranlux random-number generator
! =====

! -----
! Step 4: Determination-of-incident-particle-parameters
! -----
! Define initial variables for incident particle normally incident
! on the slab
iqin=0 ! Incident charge - photons
ekein=1.253 ! Kinetic energy
xin=0.0 ! Incident at origin
yin=0.0
zin=0.0
uin=0.0 ! Moving along z axis
vin=0.0
win=1.0
irin=1 ! Starts in region 2, could be 1
wtin=1.0 ! Weight = 1 since no variance reduction used

! -----
! Step 5: hatch-call
! -----
! Maximum total energy of an electron for this problem must be
! defined before hatch call
emaxe = ekein + RM ! photon

write(1,160)
160 format(/,' Start ucsource '//
*' Call hatch to get cross-section data')

! -----
! Open files (before HATCH call)
! -----
open(UNIT=KMPI,FILE='pgs5job.pegs5dat',STATUS='old')
open(UNIT=KMPO,FILE='egs5job.dummy',STATUS='unknown')

write(1,170)
170 FORMAT(/,' HATCH-call comes next',/)

! =====
! call hatch
! =====

! -----
! Close files (after HATCH call)
! -----
close(UNIT=KMPI)
close(UNIT=KMPO)

write(39,180)
180 FORMAT('MSTA')
write(39,190) nreg
190 FORMAT(I4)
write(39,200) (med(i),i=1,nreg)
200 FORMAT(15I4)
write(39,210)
210 FORMAT('MEND')

! -----
! Selection mode from Keyboard.
! -----
write(6,220)
220 FORMAT(' Key in mode. 0:trajectory display, 1:dose calculation')
read(5,*) imode

! -----
! Step 6: Initialization-for-howfar
! -----

```

```

! Step 7: Initialization-for-ausgab
-----
      ncount = 0
      ilines = 0
      nwrite = 10
      nlines = 10
      idin = -1
      totke = 0.
      wtsum = 0.

!
=====
      call ecnsv1(0,nreg,totke)
      call ntally(0,nreg)
=====
!
      Energy bin width
      deltae=ekein/ 50

!
      Zero the variables
      do j=1,50
         spg(j)=0.D0
         spe(j)=0.D0
      end do

!
      Set histories, number of batch and histories per batch
      write(6,*) (' Key in number of cases. ')
      read(5,*) ncases

      tt=etime(tarray)
      tt0=tarray(1)

-----
! Step 8: Shower-call
-----
230      nofbat=1
      write(39,230) nofbat
      format('0',I5)

      do i=1,ncases
! -----
! Start of batch -loop
! -----

!
! Select incident energy
! -----

      wtin = 1.0

      wtsum = wtsum + wtin
! Keep running sum of weights

!
      Determine source energy
      ekin = ekein

      if(i.eq.1) then
         ekinm=ekin
      else
         if(ekin.gt.ekinm) ekinm=ekin
      end if
      etot = ekin + iabs(iqin)*RM
! Incident total energy (MeV)
      availke = etot + iqin*RM
! Available K.E. (MeV) in system
      totke = totke + availke
! Keep running sum of KE

!
      Sample direction

!
      Sample position

! -----
! Get source region from cg input data
! -----

      if(irin.le.0.or.irin.gt.nreg) then
         call srzone(xin,yin,zin,iqin+2,0,irinn)
         call rstnxt(iqin+2,0,irinn)
      else
         irinn=irin
      end if

!
=====
      call shower (iqin,etot,xin,yin,zin,uin,vin,win,irinn,wtin)
=====
!

```

```

        ncount = ncount + 1          ! Count total number of actual cases

        call plotxyz(99,0,0,0.D0,0.D0,0.D0,0.D0,0,0,0.D0)

240      write(39,240)                ! Set end of batch for CG View
        FORMAT('9')

        end do                        ! -----
                                     ! End of batch loop
                                     ! -----

        tt=etime(tarray)
        tt1=tarray(1)
        cputime=tt1-tt0
        write(1,250) cputime
250     format(' Elapsed Time (sec)=',G15.5)

!-----
! Step 9:  Output-of-results
!-----

! -----
! Source spectrum.  Incident particle spectrum to detector.
! -----
260     write(1,260)
        FORMAT(/' Particle spectrum reach to region 3'/
*         30X,'particles/source'/
*         ' Upper energy',11X,' Gamma',18X,' Electron')

        do ie=1,50
            elow=deltae*(ie-1)
            eup=deltae*ie
            if (elow .gt. ekinm ) go to 280

! -----
! Gamma spectrum per source
! -----
            spg(ie)=spg(ie)/ncount
! -----
! Electron spectrum per source
! -----
            spe(ie)=spe(ie)/ncount

        write(1,270) eup,spg(ie),spe(ie)
270     FORMAT(G10.5,' MeV--',8X,G12.5,8X,G12.5)
        end do

280     continue

! =====
! call counters_out(1)
! =====

        stop

        end

!-----last line of main code-----

!-----ausgab.f-----
! Version:  030831-1300
! Reference: SLAC-265 (p.19-20, Appendix 2)
!-----
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
! -----
! Required subroutine for use with the EGS5 Code System
! -----
! A AUSGAB to:
! 1) Score energy deposition
! 2) Score particle information enter to detector from outside
! 3) Print out particle transport information
! 4) call plotxyz if imode=0
! -----

        subroutine ausgab(iarg)
        implicit none

```



```

include 'include/egs5_h.f'           ! Main EGS "header" file
include 'include/egs5_epcont.f'     ! COMMONs required by EGS5 code
include 'include/egs5_misc.f'
include 'include/egs5_stack.f'
include 'include/egs5_useful.f'

include 'auxcommons/aux_h.f'       ! Auxiliary-code "header" file
include 'auxcommons/etaly1.f'      ! Auxiliary-code COMMONs
include 'auxcommons/lines.f'
include 'auxcommons/ntaly1.f'
include 'auxcommons/watch.f'

include 'auxcommons/etaly2.f'     ! Added SJW for energy balance

common/totals/
* deltae,spg(50),spe(50),imode
real*8 deltae,spg,spe,spp
integer imode

integer                               ! Arguments
* iarg

real*8                                 ! Local variables
* edepwt

integer
* ie,iql,irl
!
! -----
! Set some local variables
! -----
irl = ir(np)
iql = iq(np)
edepwt = edep*wt(np)
!
! -----
! Score energy deposition inside NaI detector
! -----
if (irl .eq. 3) then
!
! -----
! Score particle information if it enters from outside
! -----
if (irl .ne. irold .and. iarg .eq. 0) then
  if (iql .eq. 0) then                ! photon
    ie = e(np)/deltae +1
    if(ie .gt. 50) ie = 50
    spg(ie) = spg(ie) + wt(np)
  else                                ! electron
    ie = (e(np) - RM)/deltae +1
    if(ie .gt. 50) ie = 50
    spe(ie) = spe(ie) + wt(np)
  end if
end if
end if
!
! -----
! Output particle information for plot
! -----
if (imode.eq.0) then
  call plotxyz(iarg,np,iq(np),x(np),y(np),z(np),e(np),ir(np),
* wt(np))
end if

return
end

!-----last line of ausgab.f-----
!-----howfar.f-----
! Version: 050716-1300
! Reference: T. Torii and T. Sugita, "Development of PRESTA-CG
! Incorporating Combinatorial Geometry in EGS4/PRESTA", JNC TN1410 2002-201,
! Japan Nuclear Cycle Development Institute (2002).

```

```
! Improved version is provided by T. Sugita. 7/27/2004
!-----
!23456789|123456789|123456789|123456789|123456789|123456789|123456789|12
```

```
!-----
! Required (geometry) subroutine for use with the EGS5 Code System
!-----
! This is a CG-HOWFAR.
!-----
```

```

      subroutine howfar
      implicit none
c
      include 'include/egs5_h.f'          ! Main EGS "header" file
      include 'include/egs5_epcont.f'    ! COMMONs required by EGS5 code
      include 'include/egs5_stack.f'
      include 'auxcommons/geom_common.f' ! geom-common file
c
c
      integer i,j,jjj,ir_np,nozone,jty,kno
      integer irnear,irnext,irlold,irlfg,itvlf,ihitcg
      double precision xidd,yidd,zidd,x_np,y_np,z_np,u_np,v_np,w_np
      double precision tval,tval0,tval00,tval10,tvalmn,delhow
      double precision atvalttmp
      integer iq_np
c
      ir_np = ir(np)
      iq_np = iq(np) + 2
c
      if(ir_np.le.0) then
        write(6,*) 'Stopped in howfar with ir(np) <=0'
        stop
      end if
c
      if(ir_np.gt.izonin) then
        write(6,*) 'Stopped in howfar with ir(np) > izonin'
        stop
      end if
c
      if(ir_np.EQ.izonin) then
        idisc=1
        return
      end if
c
      tval=1.d+30
      itvalm=0
c
c
      body check
      u_np=u(np)
      v_np=v(np)
      w_np=w(np)
      x_np=x(np)
      y_np=y(np)
      z_np=z(np)
c
      do i=1,nbbody(ir_np)
        nozone=ABS(nbzone(i,ir_np))
        jty=itblty(nozone)
        kno=itblno(nozone)
c
      rpp check
        if(jty.eq.ityknd(1)) then
          if(kno.le.0.or.kno.gt.irppin) go to 190
          call rppcg1(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c
      sph check
        elseif(jty.eq.ityknd(2)) then
          if(kno.le.0.or.kno.gt.isphin) go to 190
          call sphcg1(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c
      rcc check
        elseif(jty.eq.ityknd(3)) then
          if(kno.le.0.or.kno.gt.irccin) go to 190
          call rcccg1(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c
      trc check
        elseif(jty.eq.ityknd(4)) then
          if(kno.le.0.or.kno.gt.itrcin) go to 190
          call trccg1(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c
      tor check
        elseif(jty.eq.ityknd(5)) then

```

```

        if(kno.le.0.or.kno.gt.itorin) go to 190
        call torcgl(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c
c**** add new geometry in here
c
        end if
190    continue
        end do
c
        irnear=ir_np
        if(itvalm.eq.0) then
            tval0=cgeps1
            xidd=x_np+tval0*u_np
            yidd=y_np+tval0*v_np
            zidd=z_np+tval0*w_np
310    continue
            if(x_np.ne.xidd.or.y_np.ne.yidd.or.z_np.ne.zidd) goto 320
            tval0=tval0*10.d0
            xidd=x_np+tval0*u_np
            yidd=y_np+tval0*v_np
            zidd=z_np+tval0*w_np
            go to 310
320    continue
        write(*,*) 'srzone:1'
        call srzone(xidd,yidd,zidd,iq_np,ir_np,irnext)
c
        if(irnext.ne.ir_np) then
            tval=0.0d0
            irnear=irnext
        else
            tval00=0.0d0
            tval10=10.0d0*tval0
            irlold=ir_np
            irlfg=0
330    continue
            if(irlfg.eq.1) go to 340
            tval00=tval00+tval10
            if(tval00.gt.1.0d+06) then
                write(6,9000) iq(np),ir(np),x(np),y(np),z(np),
&                               u(np),v(np),w(np),tval00
9000 format(' TVAL00 ERROR : iq,ir,x,y,z,u,v,w,tval=',
&           2I3,1P7E12.5)
                stop
            end if
            xidd=x_np+tval00*u_np
            yidd=y_np+tval00*v_np
            zidd=z_np+tval00*w_np
            call srzold(xidd,yidd,zidd,irlold,irlfg)
            go to 330
340    continue
c
        tval=tval00
        do j=1,10
            xidd=x_np+tval00*u_np
            yidd=y_np+tval00*v_np
            zidd=z_np+tval00*w_np
c
            write(*,*) 'srzone:2'
            call srzone(xidd,yidd,zidd,iq_np,irlold,irnext)
            if(irnext.ne.irlold) then
                tval=tval00
                irnear=irnext
            end if
            tval00=tval00-tval0
        end do
        if(ir_np.eq.irnear) then
            write(0,*) 'ir(np),tval=',ir_np,tval
        end if
    end if
else
    do j=1,itvalm-1
        do i=j+1,itvalm
            if(atval(i).lt.atval(j)) then
                atvaltmp=atval(i)
                atval(i)=atval(j)
                atval(j)=atvaltmp
            endif
        enddo
    enddo
    itvlf=0

```

```

tvalmn=tval
do jjj=1,itvalm
  if(tvalmn.gt.atval(jjj)) then
    tvalmn=atval(jjj)
  end if
  delhow=cgeps2
  tval0=atval(jjj)+delhow
  xidd=x_np+tval0*u_np
  yidd=y_np+tval0*v_np
  zidd=z_np+tval0*w_np
410  continue
  if(x_np.ne.xidd.or.y_np.ne.yidd.or.z_np.ne.zidd) go to 420
    delhow=delhow*10.d0
    tval0=atval(jjj)+delhow
    xidd=x_np+tval0*u_np
    yidd=y_np+tval0*v_np
    zidd=z_np+tval0*w_np
  go to 410
420  continue
c  write(*,*) 'srzone:3'
  call srzone(xidd,yidd,zidd,iq_np,ir_np,irnext)
  if((irnext.ne.ir_np.or.atval(jjj).ge.1.).and.
&    tval.gt.atval(jjj)) THEN
    tval=atval(jjj)
    irnear=irnext
    itvlfq=1
    goto 425
  end if
end do
425  continue
  if(itvlfq.eq.0) then
    tval0=cgmnst
    xidd=x_np+tval0*u_np
    yidd=y_np+tval0*v_np
    zidd=z_np+tval0*w_np
430  continue
  if(x_np.ne.xidd.or.y_np.ne.yidd.or.z_np.ne.zidd) go to 440
    tval0=tval0*10.d0
    xidd=x_np+tval0*u_np
    yidd=y_np+tval0*v_np
    zidd=z_np+tval0*w_np
  go to 430
440  continue
  if(tvalmn.gt.tval0) then
    tval=tvalmn
  else
    tval=tval0
  end if
end if
end if
ihitcg=0
if(tval.le.ustep) then
  ustep=tval
  ihitcg=1
end if
if(ihitcg.eq.1) THEN
  if(irnear.eq.0) THEN
    write(6,9200) iq(np),ir(np),x(np),y(np),z(np),
&    u(np),v(np),w(np),tval
9200 format(' TVAL ERROR : iq,ir,x,y,z,u,v,w,tval=',2I3,1P7E12.5)
    idisc=1
    itverr=itverr+1
    if(itverr.ge.100) then
      stop
    end if
    return
  end if
  irnew=irnear
  if(irnew.ne.ir_np) then
    call rstnxt(iq_np,ir_np,irnew)
  endif
end if
return
end
!-----last line of subroutine howfar-----

```