

線源の作り方実習
(July 04, 2007, 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 : 線源エネルギー	1
2.1. Co-60 の γ -線源	1
2.1.1. if 文の使用する方法:	1
2.1.2. data 文の使用する方法:	2
2.1.3. data ファイルを使用する方法:	4
2.2. Ir-192 の γ 線源	5
2.3. Sr-90- β 線源	6
3. 実習課題 2 : 線源位置	11
3.1. 直接サンプリング	11
3.2. Rejection 法	12
4. 実習課題 4 : 線源方向 (2π)	13
4.1. 直接サンプリング	13
4.2. Rejection 法	14

1. 基にするユーザーコード `ucsource.f` の概要

形状としては、Fig. 1 に示すように `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、Z-軸上-5cm の位置からビーム状に入力するように設定されている。

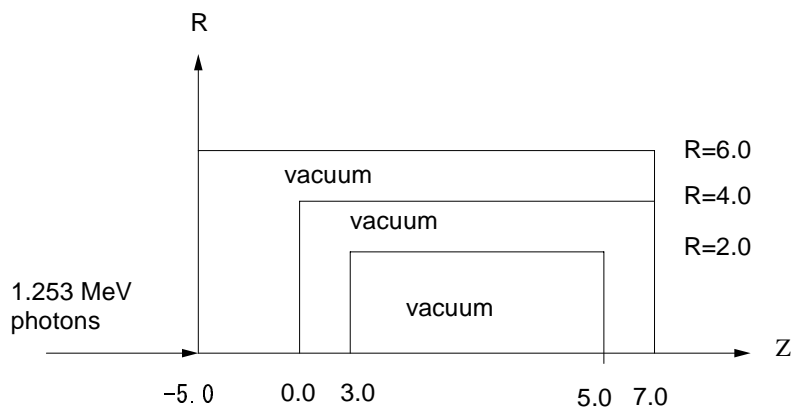


Figure 1: Geometry of `ucsource.f`

2. 実習課題 1 : 線源エネルギー

2.1. Co-60 の γ -線源

線源を 1.173MeV と 1.333MeV の 線が同じ確率で発生する Co-60 の γ -線源に変更する。if 文を使用する方法、data 文を使用する方法とデータをファイルから読み込む方法がある。

2.1.1. if 文の使用する方法:

1. `cp ucsource.f ucsource1_0.f`

2. `ucsource1_0.f` の変更

- 線源エネルギーに関する配列を増やす。

```
* esbin(1),spg(1),spe(1)
```

を

```
* esbin(2),spg(2),spe(2)
```

に変更。

```
nsebin=1
```

を

```
nsebin=2
```

に変更。

- 運動エネルギーの最大値を変更する。

```
ekein=1.253          ! Kinetic energy
```

を

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

に変更。

```
esbin(1)=ekein
```

を

```
esbin(1)=1.173
esbin(2)=1.333
```

に変更。

- エネルギーのサンプリング部分を変更する。

```
ekin = ekein
spg(1)=spg(1)+1.0
```

を

```
call randomset(rnnow)
if(rnnow.le.0.5) then
  ekin = 1.173
  spg(1)=spg(1)+1.0
else
  ekin = 1.333
  spg(2)=spg(2)+1.0
end if
```

に変更。

3. ucsourcel_0.f を egs5run で実行する。

ユニット 4 のファイル名に ucsourcel を入力し、ユニット 25 には "return" を入力する。

4. "Does this user code read from the terminal?" に対して 1 を入力する。

5. egs5job.out の光子の線源スペクトルが、1 対 1 に近い比率になっていることを確認する。

Sampled source spectrum		particles/source		
Upper energy		Gamma	Electron	pdf
1.1730	MeV--	0.49710	0.0000	
1.3330	MeV--	0.50290	0.0000	

2.1.2. data 文の使用する方法:

1. cp ucsourcel_0.f ucsourcel_1.f

2. ucsourcel_1.f の変更

- real*8 宣言に espdf(2), escdf(2) を追加する。

```
real*8
* availke,ekin,tnum,wtin,wtsum,xi0,yi0,zi0,
* esbin(2),spg(2),spe(2)
```

! Local variables

を

```
real*8
* availke,ekin,tnum,wtin,wtsum,xi0,yi0,zi0,
* esbin(2),spg(2),spe(2),espdf(2),escdf(2)
```

! Local variables

に変更。

- integer の宣言後に data 文を定義する。

```

integer
* i, icases, idin, ie, ifti, ifto, ii,
* j, k, n, nd, ner, nsebin

```

を

```

integer
* i, icases, idin, ie, ifti, ifto, ii,
* j, k, n, nd, ner, nsebin

data esbin/1.173,1.333/
data espdf/0.5,0.5/

```

に変更。

- nsebin=2 の後に、cdf を計算する文を追加する。

```
nsebin=2
```

を

```

nsebin=2
!-----
! Calculate cdf from pdf
!-----
tnum=0.D0
do ie=1,nsebin
  tnum=tnum+espdf(ie)
end do

escdf(1)=espdf(1)/tnum
do ie=2,nsebin
  escdf(ie)=escdf(ie-1)+espdf(ie)/tnum
end do

```

に変更。

- 運動エネルギーの最大値の表現を変更する。

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

を

```
ekein=esbin(nsebin) ! Maximum kinetic energy
```

に変更。

- 不要な文を削除する。

```
esbin(1)=1.173
esbin(2)=1.333
```

を削除する。

- 線源エネルギーのサンプリング部を変更する。

```

!-----
! Determine source energy
!-----
call randomset(rnnow)
if(rnnow.le.0.5) then
  ekin = 1.173
  spg(1)=spg(1)+1.0
else
  ekin = 1.333
  spg(2)=spg(2)+1.0
end if

```

を

```

! -----
! Determine source energy
! -----
call randomset(rnnow)
do ie=1,nsebin
  if(rnnow.le.escdf(ie)) go to 1000
end do
1000 ekin=esbin(ie)
      if(iqin.eq.0) then
          spg(ie)=spg(ie)+1.0
      else
          spe(ie)=spe(ie)+1.0
      end if

```

に変更。

3. ucsource1_1.f を egs5run で実行する。
 ユニット 4 のファイル名に ucsource を入力し、ユニット 25 には”return”を入力する。
4. ”Does this user code read from the terminal?”に対して 1 を入力する。
5. egs5job.out の光子の線源スペクトルが、1 対 1 に近い比率になっていることを確認する。

```

Sampled source spectrum
Upper energy      particles/source
1.1730 MeV--      Gamma
1.3330 MeV--      Electron      pdf
                  0.49710      0.0000
                  0.50290      0.0000

```

2.1.3. data ファイルを使用する方法:

1. cp ucsource1_1.f ucsource1_2.f
2. ucsource1_2.f の変更

- local variable を変更する。

```

      real*8                                ! Local variables
      * availke,ekin,tnum,wtin,wtsum,xi0,yi0,zi0,
      * esbin(2),spg(2),spe(2),espdf(2),escdf(2)

```

を

```

      real*8                                ! Local variables
      * availke,ekin,tnum,wtin,wtsum,xi0,yi0,zi0,esbin(MXEBIN),
      * spg(MXEBIN),spe(MXEBIN),espdf(MXEBIN),escdf(MXEBIN)

```

に変更する。

- data 文

```

      data esbin/1.173,1.333/
      data espdf/0.5,0.5/

```

を削除する。

- open 文を追加する。

```

      open(6,FILE='egs5job.out',STATUS='unknown')

```

を

```

      open(6,FILE='egs5job.out',STATUS='unknown')
      open(2,file='co60.inp',status='unknown')

```

に変更。

- co60.inp は、線源のエネルギーとその確率密度関数で以下の内容のファイルであり、配布ファイルに含まれている。

```
1.173,1.333
0.5,0.5
```

- nsebin=2 の後に、以下を挿入する。

```
read(2,*) (esbin(i),i=1,nsebin)
read(2,*) (espdf(i),i=1,nsebin)
```

- 結果の出力部を変更する。

```
write(6,170) esbin(ie),spg(ie),spe(ie)
170 FORMAT(G10.5,' MeV--',8X,G12.5,8X,G12.5)
```

を

```
write(6,170) esbin(ie),spg(ie),spe(ie),espdf(ie)/tnum
170 FORMAT(G10.5,' MeV--',8X,G12.5,8X,G12.5,8X,G12.5)
```

に変更する。

3. ucsource1_2.f を egs5run で実行する。

ユニット 4 のファイル名に ucsource を入力し、ユニット 25 には "return" を入力する。

4. "Does this user code read from the terminal?" に対して 1 を入力する。

5. egs5job.out の光子の線源スペクトルが、1 対 1 に近い比率になっていることを確認する。

```
Sampled source spectrum
Upper energy      Gamma      Electron      pdf
1.1730 MeV--      0.49710      0.0000      0.50000
1.3330 MeV--      0.50290      0.0000      0.50000
```

2.2. Ir-192 の γ 線源

Ir-192 から放出される γ 線のエネルギーと崩壊当たりの放出率は、以下の通りである。(アイソトープ手帳第 10 版)

Energy	Emission Rate
0.296	28.7
0.308	30.0
0.317	82.7
0.468	47.8
0.589	4.5
0.604	8.2
0.612	5.3

1. cp ucsource1_2.f ucsource2.f

2. ucsoure2.f の変更

- 線源データファイルに関する open 文を変更する。

```
open(2,file='co60.inp',status='unknown')
```

を

```
open(2,file='ir192.inp',status='unknown')
```

に変更。

- ir192.inp は、線源のエネルギーとその確率密度関数で以下の内容のファイルで、配布ファイルに含まれている。

```
0.296,0.308,0.317,0.468,0.589,0.604,0.612  
0.287,0.300,0.827,0.478,0.045,0.082,0.053
```

- 線源のデータ数を変更する。¹

```
nsebin=2
```

を

```
nsebin=7
```

に変更。

3. ucsource2.f を egs5run で実行する。
ユニット 4 のファイル名に ucsource を入力し、ユニット 25 には "return" を入力する。
4. "Does this user code read from the terminal?" に対して 1 を入力する。
5. egs5job.out の光子の線源スペクトルが、設定した比率に近いものになっていることを確認する。

```
Sampled source spectrum  
particles/source  
Upper energy      Gamma      Electron      pdf  
.29600    MeV--      0.14040      0.0000      0.13851  
.30800    MeV--      0.14210      0.0000      0.14479  
.31700    MeV--      0.40050      0.0000      0.39913  
.46800    MeV--      0.22930      0.0000      0.23069  
.58900    MeV--      0.22000E-01  0.0000      0.21718E-01  
.60400    MeV--      0.39300E-01  0.0000      0.39575E-01  
.61200    MeV--      0.26400E-01  0.0000      0.25579E-01
```

2.3. Sr-90- 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、Sr-90 の β 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、Sr-90 の β 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 β 線数で与えられている。(次表及び図) このデータを使用して β 線のエネルギーを決定するルーチンを作成する。

¹この問題のように、配列の引数となる変数の値を変更する場合には、まずデバッガー機能を含めてコンパイル、実行を行い、配列範囲外アクセスが起きないことを確認するべきである。方法としては、"egs5run" と入力するところで "egs5run db" と入力する。これにより、デバッガー機能を含めたコンパイルが行われる。つぎに "egs5job.exe" と入力して、計算を実行する。配列範囲外アクセスが起きなければ計算は通常通り終了する。(追加的なメッセージはなにも表示されない) 配列範囲外アクセスが起きた場合には、ソースのどの行で、どの配列の何番目の要素に不正なアクセスが行われたかが表示されるので、ソースの当該部分を修正する。なお、デバッガーを含めてコンパイルした場合実行速度が低下するので、デバッガーの使用はプログラム変更の場合のみとする方がよい。

Table 1 β -ray spectrum from Sr-90(ICRU Report 56).

E/E_{max}	β per dis. per bin	E/E_{max}	β per dis. per bin
0.00	1.597	0.025	1.538
0.05	1.532	0.075	1.526
0.01	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		

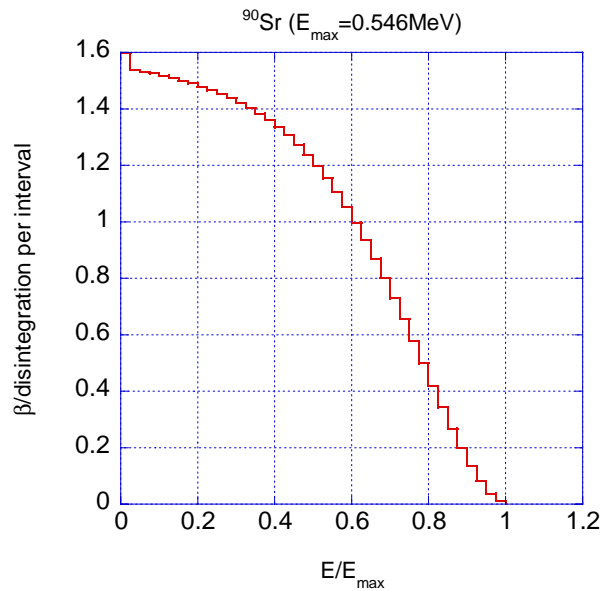


Figure 2: β -ray spectrum from Sr-90(ICRU Report 56).

1. cp ucsourc2.f ucsourc3.f

2. ucsource3.fの変更

- local variable に, deltaes, emax を追加する。

```
real*8                                ! Local variables
* availke,ekin,tnum,wtin,wtsum,xi0,yi0,zi0,esbin(MXEBIN),
* spg(MXEBIN),spe(MXEBIN),espdf(MXEBIN),escdf(MXEBIN)
```

を

```
real*8                                ! Local variables
* availke,ekin,tnum,wtin,wtsum,xi0,yi0,zi0,esbin(MXEBIN),
* spg(MXEBIN),spe(MXEBIN),espdf(MXEBIN),escdf(MXEBIN),
* deltaes,emax
```

に変更。

- 線源に関する open 文を変更する。

```
open(2,file='ir192.inp',status='unknown')
```

を

```
open(2,file='sr90beta.inp',status='unknown')
```

に変更。

- sr90beta.inp は、上記の崩壊当たり各区分エネルギー当たりの β 線の放出率であり、以下の内容のファイルで、配布ファイルに含まれている。放出率から累積分布関数 (cdf) を求めてサンプリングに使用する。

```
0.546
41
0.025
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
```

0.546 は β 線の最大エネルギー (E_{max} , MeV)、41 は分点数、0.025 は、 E/E_{max} の区分幅である。

- 線源データファイルから nsebin を読み込み部を変更する。

```
nsebin=7                                ! Number of source energy bins
read(2,*) (esbin(i),i=1,nsebin)
read(2,*) (espdf(i),i=1,nsebin)
```

を

```
read(2,*) emax                          ! Maximum beta-ray energy
read(2,*) nsebin                         ! Number of source energy bins
read(2,*) deltaes                        ! Source energy bin width in MeV
read(2,*) (espdf(i),i=1,nsebin)
```

に変更。

- エネルギービンの値を計算する文を追加する。

```
do ie=1,nsebin
  tnum=tnum+espdf(ie)
end do
```

を

```
do ie=1,nsebin
  esbin(ie)=(ie-1)*deltaes*emax
  tnum=tnum+espdf(ie)
end do
```

に変更。

- cdf 作成の部分を変更する。

```
escdf(1)=espdf(1)/tnum
do ie=2,nsebin
  escdf(ie)=escdf(ie-1)+espdf(ie)/tnum
end do
```

を

```
escdf(1)=0.0
do ie=2,nsebin
  escdf(ie)=escdf(ie-1)+espdf(ie-1)/tnum
end do
```

に変更。

- 線源粒子の種類を変更する。

```
iqin=0          ! Incident charge - photons
```

を

```
iqin=-1        ! Incident charge - electrons
```

に変更。

- ヒストリー数を増やす。

```
ncases=10000
```

を

```
ncases=100000
```

に変更。

- 線源エネルギーのサンプリング部を変更する。

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

を。

```
do ie=2,nsebin
  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))
```

に修正。

- 結果の出力部分を変更する。

```
do ie=1,nsebin
! -----
! Gamma spectrum per source
! -----
!   spg(ie)=spg(ie)/ncount
! -----
! Electron spectrum per source
! -----
!   spe(ie)=spe(ie)/ncount
! -----
write(6,170) esbin(ie),spg(ie),spe(ie),espdf(ie)/tnum
170  FORMAT(G10.5,' MeV--',8X,G12.5,8X,G12.5,8X,G12.5)
end do
```

```

を
do ie=2,nsebin
! -----
! Gamma spectrum per source
! -----
! spg(ie)=spg(ie)/ncount
! -----
! Electron spectrum per source
! -----
! spe(ie)=spe(ie)/ncount
! -----
write(6,170) esbin(ie),spg(ie),spe(ie),espdf(ie-1)/tnum
170 FORMAT(G10.5,' MeV--',8X,G12.5,8X,G12.5,8X,G12.5)
end do

```

に修正。

3. ucsourse3.f を egs5run で実行する。
 ユニット 4 のファイル名に ucsourse を入力し、ユニット 25 には”return”を入力する。
4. ”Does this user code read from the terminal?”に対して 1 を入力する。
5. egs5job.out の電子の線源スペクトルが、設定した比率に近いものになっていることを確認する。

```

Sampled source spectrum
                                particles/source
Upper energy      Gamma      Electron      pdf
.13650E-01 MeV--      0.0000      0.38200E-01      0.37663E-01
.27300E-01 MeV--      0.0000      0.37300E-01      0.37516E-01
.40950E-01 MeV--      0.0000      0.36000E-01      0.37369E-01
.54600E-01 MeV--      0.0000      0.34800E-01      0.37173E-01
.68250E-01 MeV--      0.0000      0.39400E-01      0.36953E-01
.81900E-01 MeV--      0.0000      0.34600E-01      0.36732E-01
.95550E-01 MeV--      0.0000      0.37600E-01      0.36487E-01
.10920      MeV--      0.0000      0.35500E-01      0.36218E-01
.12285      MeV--      0.0000      0.36000E-01      0.35900E-01
.13650      MeV--      0.0000      0.34600E-01      0.35581E-01
.15015      MeV--      0.0000      0.32200E-01      0.35239E-01
.16380      MeV--      0.0000      0.35000E-01      0.34822E-01
.17745      MeV--      0.0000      0.34900E-01      0.34381E-01
.19110      MeV--      0.0000      0.32400E-01      0.33892E-01
.20475      MeV--      0.0000      0.31000E-01      0.33328E-01
.21840      MeV--      0.0000      0.33000E-01      0.32692E-01
.23205      MeV--      0.0000      0.33800E-01      0.31982E-01
.24570      MeV--      0.0000      0.31000E-01      0.31198E-01
.25935      MeV--      0.0000      0.32500E-01      0.30316E-01
.27300      MeV--      0.0000      0.30500E-01      0.29337E-01
.28665      MeV--      0.0000      0.26400E-01      0.28259E-01
.30030      MeV--      0.0000      0.30700E-01      0.27084E-01
.31395      MeV--      0.0000      0.23900E-01      0.25786E-01
.32760      MeV--      0.0000      0.26300E-01      0.24415E-01
.34125      MeV--      0.0000      0.24700E-01      0.22896E-01
.35490      MeV--      0.0000      0.21400E-01      0.21305E-01
.36855      MeV--      0.0000      0.19200E-01      0.19615E-01
.38220      MeV--      0.0000      0.16300E-01      0.17852E-01
.39585      MeV--      0.0000      0.15500E-01      0.16015E-01
.40950      MeV--      0.0000      0.13800E-01      0.14130E-01
.42315      MeV--      0.0000      0.11900E-01      0.12195E-01
.43680      MeV--      0.0000      0.96000E-02      0.10285E-01
.45045      MeV--      0.0000      0.98000E-02      0.83995E-02
.46410      MeV--      0.0000      0.66000E-02      0.65628E-02
.47775      MeV--      0.0000      0.43000E-02      0.48487E-02
.49140      MeV--      0.0000      0.27000E-02      0.33059E-02
.50505      MeV--      0.0000      0.26000E-02      0.19835E-02
.51870      MeV--      0.0000      0.12000E-02      0.93055E-03
.53235      MeV--      0.0000      0.10000E-03      0.24488E-03
.54600      MeV--      0.0000      0.0000      0.0000

```

3. 実習課題2：線源位置

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。

3.1. 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。

$$f(r)dr = c \times 2\pi r dr$$
$$\int_{R_0}^{R_1} f(\xi)d\xi = c\pi[\xi^2]_{R_0}^{R_1} = c\pi[R_1^2 - R_0^2] = 1$$
$$c = \frac{1}{\pi(R_1^2 - R_0^2)} \rightarrow f(r)dr = \frac{2rdr}{R_1^2 - R_0^2}$$

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、 ϕ を、0 から 2π 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 ucsource.f ucsource4.f

2. ucsource4.f の変更

- local variable に r02,r12,phai を追加する。

```
* esbin(1),spg(1),spe(1)
```

を

```
* esbin(1),spg(1),spe(1),r02,r12,phai,rr0
```

に変更する。

- 線源の weight 設定後に、r01,r12 の設定文を挿入する。

```
wtin=1.0          ! Weight = 1 since no variance reduction used
```

を

```
wtin=1.0          ! Weight = 1 since no variance reduction used
```

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

に変更。

- 線源位置のサンプリングを挿入する。

```
! -----  
! Determine source position  
! -----
```

を

```

! -----
! Determine source position
! -----
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)

```

に変更。

3. `ucsource4.f` を `egs5run` で実行する。
 ユニット 4 のファイル名に `ucsource` を入力し、ユニット 25 には "return" を入力する。
4. "Does this user code read from the terminal?" に対して 1 を入力する。
5. CGView で、座標軸を X-Y にし、軸を若干傾け、半径 1.5-4.0 の領域から光子が出ていることを確認する。

3.2. 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、 $R_0/R_1 \leq r = \sqrt{x^2+y^2} \leq 1.0$ の場合には $x * R_1$ 及び $y * R_1$ を線源位置とし、それ以外の場合は、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

- `local variable` を変更する。

```
* esbin(1),spg(1),spe(1),r02,r12,phai,rr0
```

を

```
* esbin(1),spg(1),spe(1),r0,r1,rr0
```

に変更。

- `r02,r12` の設定を `r0,r1` の設定に変更する。

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

を

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

に変更。

- 線源位置のサンプリング方法を変更する。

```

! -----
! Determine source position
! -----
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)

```

を

```
! -----  
! Determine source position  
! -----  
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. ucsourse5.f を egs5run で実行する。

ユニット 4 のファイル名に ucsourse を入力し、ユニット 25 には "return" を入力する。

4. "Does this user code read from the terminal?" に対して 1 を入力する。

5. CGView で、直接サンプリングと同じように半径 1.5-4cm の領域から光子が出ていることを確認する。

4. 実習課題 4 : 線源方向 (2π)

4.1. 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 $\cos \theta$ が正の領域のみに限られる等方線源の場合は、

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

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 η は、等価なので、 $w = \eta$ とする。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

- local variable に,phai,rr0 を追加する。

```
real*8
* availke,ekin,tnum,wtin,wtsum,xi0,yi0,zi0,
* esbin(1),spg(1),spe(1)
```

! Local variables

を

```
real*8
* availke,ekin,tnum,wtin,wtsum,xi0,yi0,zi0,
* esbin(1),spg(1),spe(1),phai,rr0
```

! Local variables

に変更。

- 線源の方向をサンプリングする文を挿入する。

```
! -----
! Determine source direction
! -----
```

を

```
! -----
! Determine source direction
! -----
call randomset(rnnow)
win=rnnow
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. ucsource6.f を egs5run で実行する。

ユニット 4 のファイル名に ucsource を入力し、ユニット 25 には "return" を入力する。

4. "Does this user code read from the terminal?" に対して 1 を入力する。

5. CGView で、光子が 2π 方向に等方的に発生していることを確認する。

4.2. 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、原点からサンプリングされた点に向かう方向を方向余弦とする。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

- 線源の方向をサンプリングする部分を修正する。

```
! -----
! Determine source direction
! -----
call randomset(rnnow)
win=rnnow
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)
```

を


```

! -----
! Determine source direction
! -----
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. `ucsource7.f` を `egs5run` で実行する。
 ユニット 4 のファイル名に `ucsource` を入力し、ユニット 25 には `"return"` を入力する。
4. "Does this user code read from the terminal?" に対して 1 を入力する。
5. CGView で、光子が直接サンプリングの場合と同じように 2π 方向に等方的に発生していることを確認する。


```

! -----
! include 'include/egs5_h.f' ! Main EGS "header" file
!
! include 'include/egs5_bounds.f'
! include 'include/egs5_brempr.f'
! include 'include/egs5_edge.f'
! include 'include/egs5_media.f'
! 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/ ! Variables to score
! * maxpict
! integer maxpict
!
! **** real*8 ! Arguments
! real*8 totke
! real*8 rnow,etot
!
! real*8 ! Local variables
! * availke,ekin,tnum,wtin,wtsum,xi0,yi0,zi0,
! * esbin(1),spg(1),spe(1)
!
! real
! * tarray(2),tt,tt0,tt1,cputime
!
! integer
! * i,icases,idin,ie,ifti,ifto,ii,
! * j,k,n,nd,ner,nsebin
!
! 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 call pegs5. Unit for pict must be 39.
! -----
!
! open(6,FILE='egs5job.out',STATUS='unknown')
! open(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) = 1.0d0      ! optional, but recommended to invoke
                    ! automatic step-size control

write(6,*) 'chard =',(chard(j),j=1,1)

! -----
! Run KEK PEGS5 before calling HATCH
! -----
100 write(6,100)
   FORMAT(' PEGS5-call comes next'/)

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

!-----
! Step 3: Pre-hatch-call-initialization
!-----
!-----
! Initialize cg related parameter
!-----

npreci=3      ! PICT data mode for CGView in free format

ifti = 4      ! Input unit number for cg-data
ifto = 39     ! Output unit number for PICT

write(6,fmt="( ' CG data' )")
call geomgt(ifti,6) ! Read in CG data
write(6,fmt="( ' End of CG data',/ )")

if(npreci.eq.3) write(ifto,fmt="( 'CSTA-FREE' )")
if(npreci.eq.2) write(ifto,fmt="( 'CSTA' )")

rewind ifti
call geomgt(ifti,ifto)! Dummy call to write geom info for ifto
110 write(ifto,110)
   FORMAT('CEND')

!-----
! Get nreg from cg input data
!-----

nreg=izonin

! Read material for each region from egs5job.data
read(4,*) (med(i),i=1,nreg)

! -----
! Random number seeds. Must be defined before call hatch
! or defaults will be used. inseed (1- 2^31)
! -----

luxlev = 1
inseed=1
write(6,120) inseed
120 FORMAT(/, ' inseed=',I12,5X,
* (seed for generating unique sequences of Ranlux)')

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

!-----
! Step 4: Determination-of-incident-particle-parameters
!-----
! Define initial variables for incident particle normally incident
! on the slab
nsebin=1
iqin=0      ! Incident charge - photons

```

```

ekein=1.253      ! Kinetic energy
xin=0.0         ! Source position
yin=0.0
zin=-5.0
uin=0.0         ! Moving along z axis
vin=0.0
win=1.0
irin=1          ! Starts in region 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

130  write(6,130)
      format(/,' 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')

140  write(6,140)
      FORMAT(/,' HATCH-call comes next',/)

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

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

      write(6,fmt="( ' CG data' )")

      write(39,fmt="( 'MSTA' )")
      write(39,fmt="(i4)") nreg
      write(39,fmt="(15i4)") (med(i),i=1,nreg)
      write(39,fmt="( 'MEND' )")
-----
! 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)
! =====

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

! Set histories and histories to write trajectories
! ncases=10000
! Set maximum number for pict
! maxpict=500

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

```

```

-----
! Step 8: Shower-call
-----
!
  Write batch number
  write(39,fmt="( '0 1' )")
  do i=1,ncases
    ! -----
    ! Start of batch -loop
    ! -----

    wtin = 1.0

    wtsum = wtsum + wtin          ! Keep running sum of weights

    ! -----
    ! Determine source energy
    ! -----
    ekin = ekein
    spg(1)=spg(1)+1.0

    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

    ! -----
    ! Determine source direction
    ! -----

    ! -----
    ! Determine source 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

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

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

  write(39,fmt="( '9' )")      ! Set end of batch for CG View

  tt=etime(tarray)
  tt1=tarray(1)
  cputime=tt1-tt0
  write(6,150) cputime
150  format(' Elapsed Time (sec)=',G15.5)

-----
! Step 9: Output-of-results
-----
!
! -----
! Source spectrum. Incident particle spectrum to detector.
! -----
160  write(6,160)
  FORMAT(/' Sampled source spectrum'/
*      30X,'particles/source'/
*      ' Upper energy',11X,' Gamma',14X,' Electron',
*      11X,' pdf')

  do ie=1,nsebin

  ! -----
  ! Gamma spectrum per source
  ! -----

```

```

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

      write(6,170) esbin(ie),spg(ie),spe(ie)
170  FORMAT(G10.5,' MeV--',8X,G12.5,8X,G12.5)
      end do

!=====
! 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: produce trajectory data for 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/lines.f'       ! Auxiliary-code COMMONs

      common/totals/                     ! Variables to score
      * maxpict
      integer maxpict

      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)

!-----
! Output particle information for plot
!-----
      if (ncount.le.maxpict) 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: 060620-1400
! 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,tval10,tvalmn,delhow
double precision atvaltmp
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
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,nbody(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 torcg1(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c      rec check
        elseif(jty.eq.ityknd(6)) then
            if(kno.le.0.or.kno.gt.irecin) go to 190
            call reccg1(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c      ell check
        elseif(jty.eq.ityknd(7)) then
            if(kno.le.0.or.kno.gt.iellin) go to 190
            call ellcg1(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c      wed check
        elseif(jty.eq.ityknd(8)) then
            if(kno.le.0.or.kno.gt.iwedin) go to 190
            call wedcg1(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c      box check
        elseif(jty.eq.ityknd(9)) then
            if(kno.le.0.or.kno.gt.iboxin) go to 190
            call boxcg1(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c      arb check
        elseif(jty.eq.ityknd(10)) then
            if(kno.le.0.or.kno.gt.iarbin) go to 190
            call arbcg1(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c      hex check
        elseif(jty.eq.ityknd(11)) then
            if(kno.le.0.or.kno.gt.ihexin) go to 190
            call hexcg1(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c      haf check
        elseif(jty.eq.ityknd(12)) then
            if(kno.le.0.or.kno.gt.ihafin) go to 190
            call hafcg1(kno,x_np,y_np,z_np,u_np,v_np,w_np)
c      tec check
        elseif(jty.eq.ityknd(13)) then
            if(kno.le.0.or.kno.gt.itecin) go to 190
            call teccg1(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=cgepsi
            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
&              itvlf=1
&              goto 425
&          end if
&          end do
425          continue
&          if(itvlf.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-----

```