

線源ルーチンの書き方

平山 英夫、波戸 芳仁

KEK, 高エネルギー加速器研究機構

2020-08-03

線源ルーチン

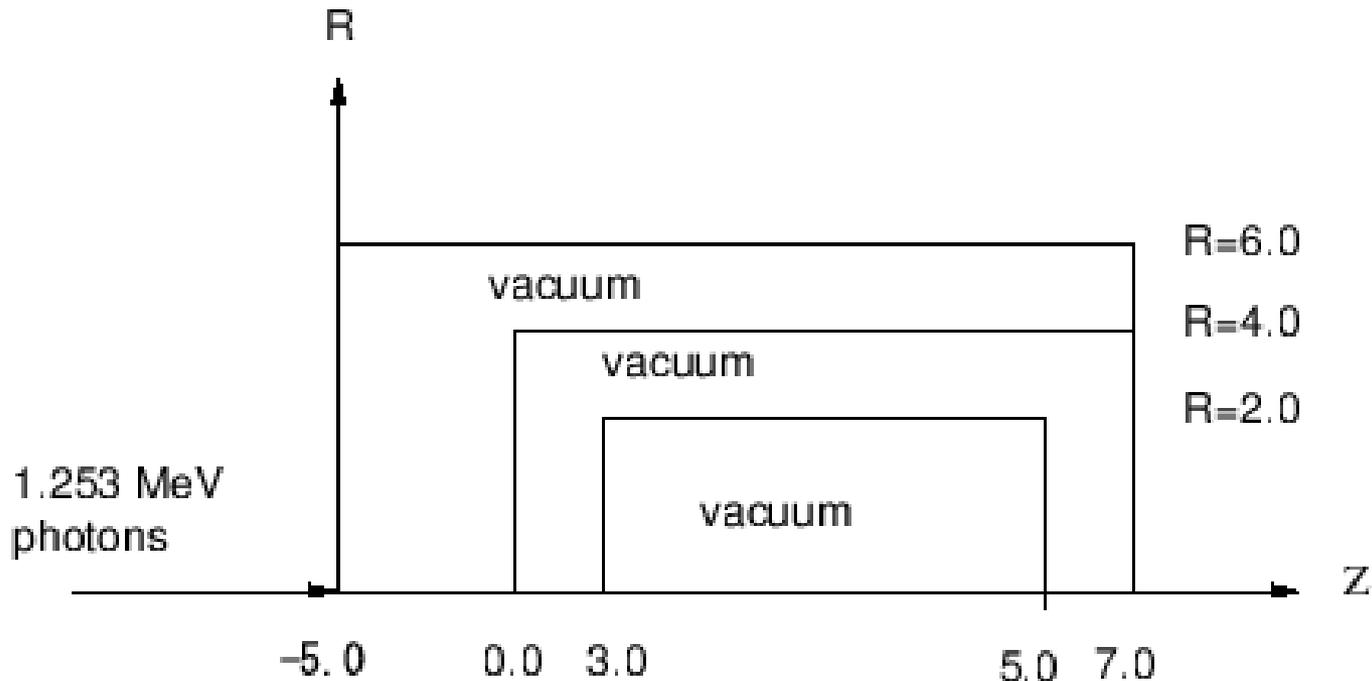
- 線源ルーチンは、線源粒子のエネルギー、位置、方向を決定するルーチン
- これらのパラメータが、個々の線源で異なる場合には、線源ルーチンを”Shower call loop”内で、call shower の前に書く必要がある。
- これらのパラメータは、subroutine shower の引数を介してegs5に引き渡される。
- ユーザーコードucsource.fを基に線源ルーチンの書き方を理解する実習

ucsource.f

“線源ルーチンの書き方”を理解するための簡単なユーザーコード

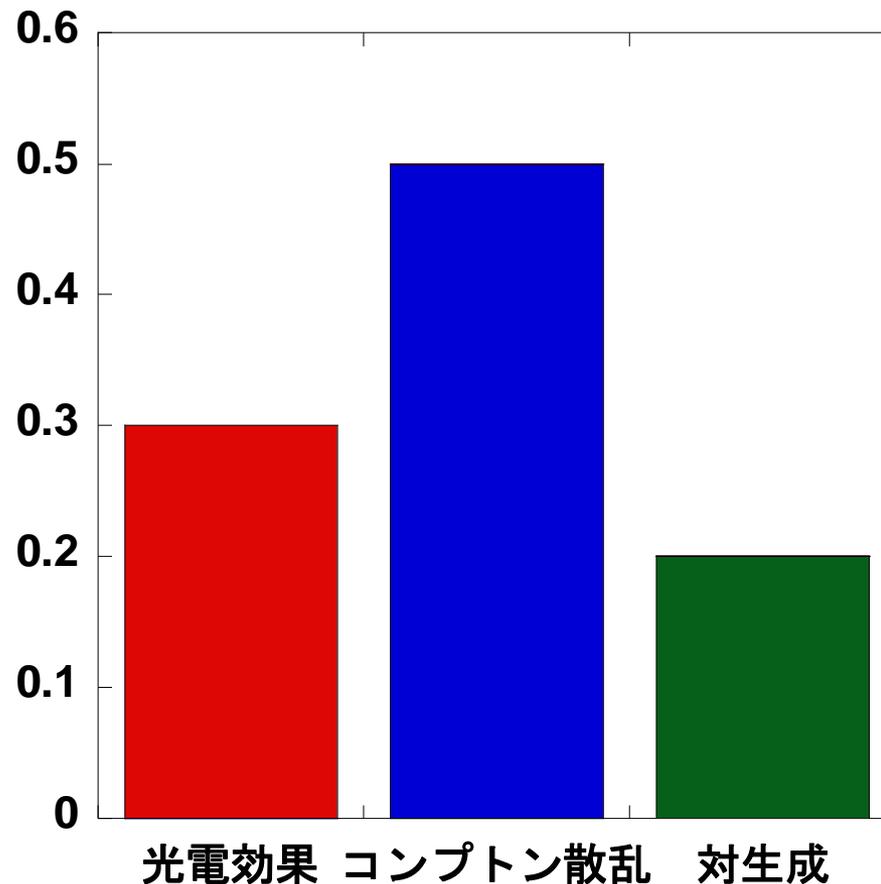
すべてのリージョンは真空 (0)

1.253 MeV の光子ビームが、円筒の中心にZ軸上で、 $Z=-5.0\text{cm}$ の位置から垂直に入射する。

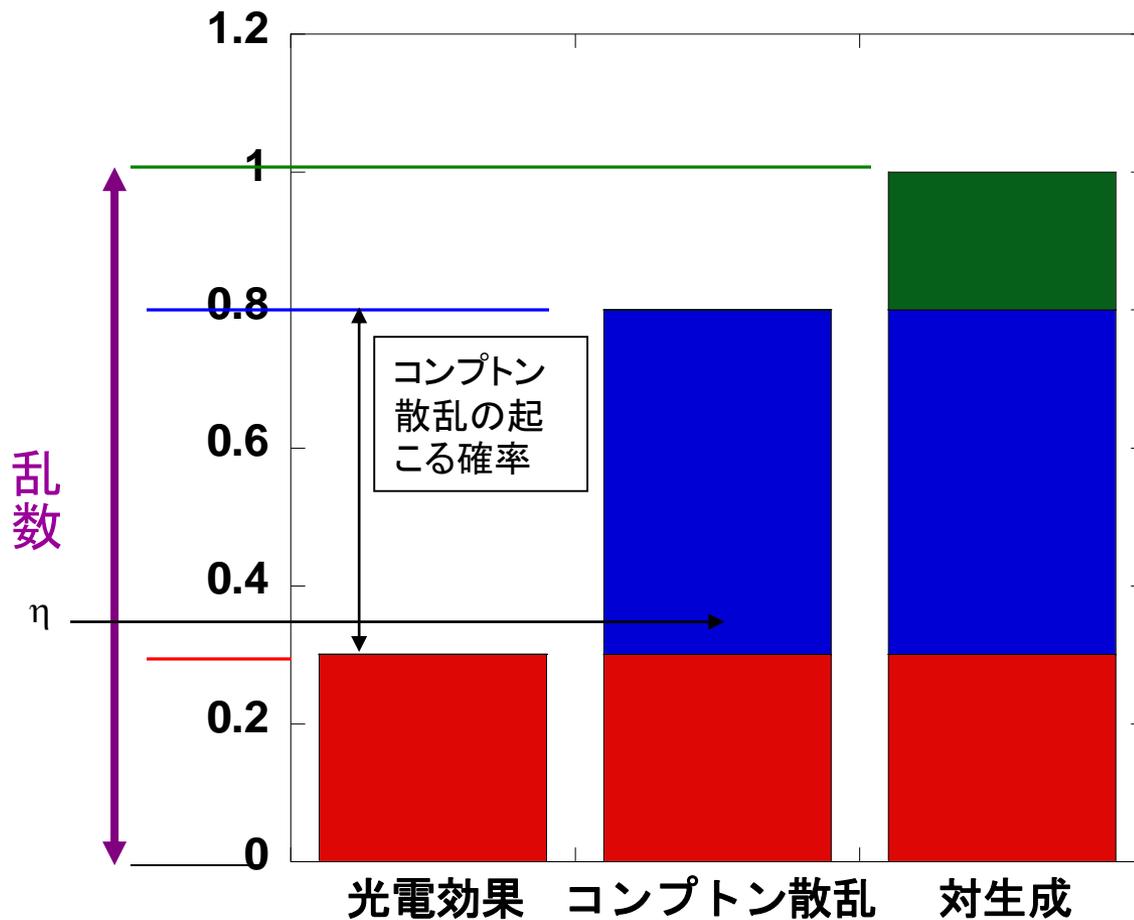


離散型サンプリングの導入(1)

例題) 光電効果:30%、コンプトン散乱:50%、対生成:20%を
乱数をもちいてサンプリングせよ。



離散型サンプリングの導入(2)



「累積分布関数」または「積み上げ計算」

離散型確率過程のサンプリング

- x_1, x_2, \dots, x_n を確率 p_1, p_2, \dots, p_n を持つ n 個の独立背反物理事象とする。(例えば、光子の反応における、光電吸収、コンプトン散乱、電子対生成とそれぞれの発生確率)

$$F(x_n) = \sum_{i=1}^n p_i = 1 \quad \text{累積分布関数}$$

- η を 0 から 1 の間の一様な乱数とすると、以下の条件の時、事象 x_i が起きることになる。

$$F(x_{i-1}) = \sum_{j=1}^{i-1} p_j \leq \eta < F(x_i) = \sum_{j=1}^i p_j$$

サンプリング法 (連続型確率過程)

- ある物理現象が x と $x+dx$ で発生する確率を $f(x)dx$ とする。 $[a \leq x \leq b]$ $f(x)$ を確率密度関数(PDF)という。

$$\left(\int_a^b f(x)dx = 1\right)$$

- 累積分布関数(CDF: $F(x)$)

$$F(x) = \int_a^x f(x_i)dx_i$$

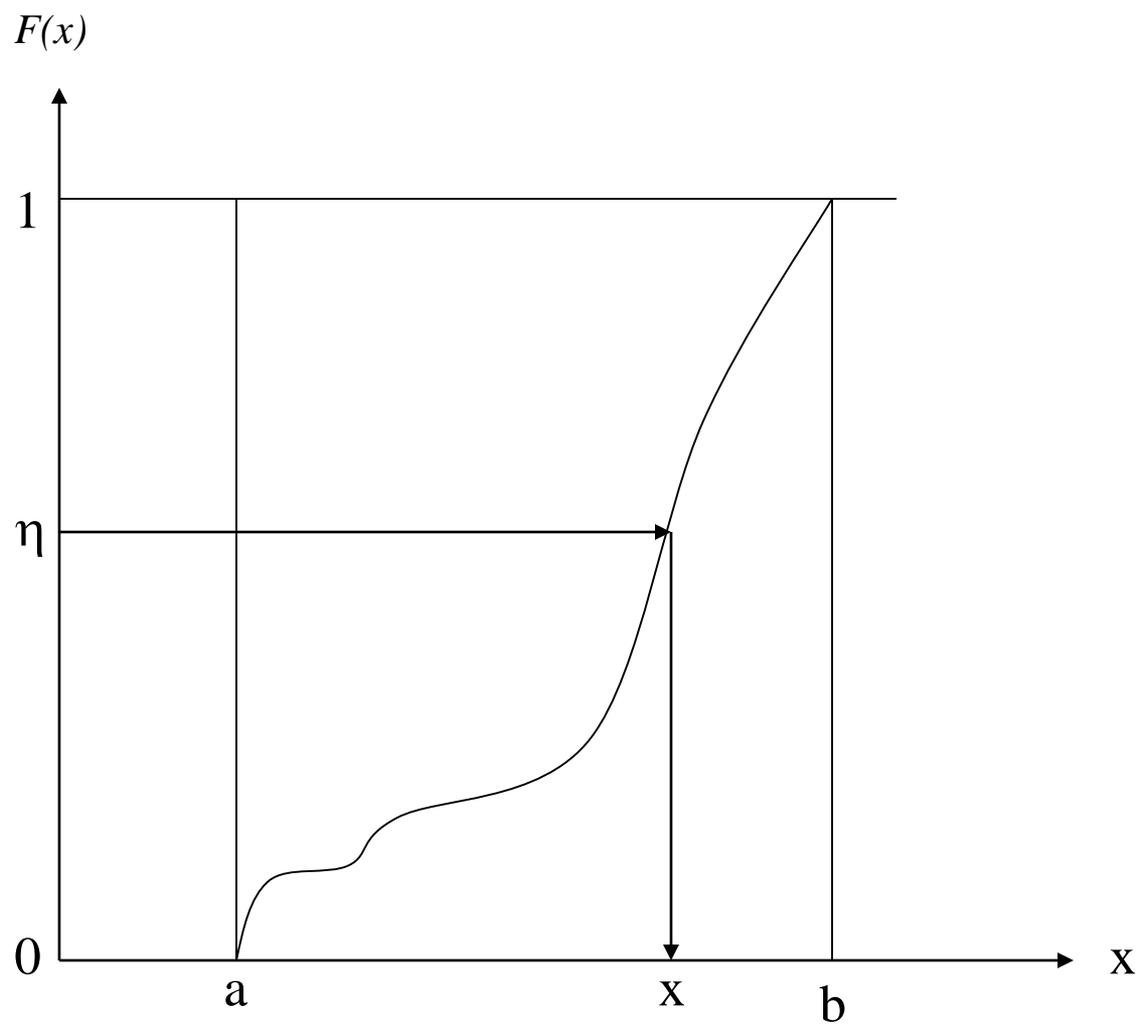
- η を0から1の間の一様な乱数とすると、

$$\eta = F(x) = \int_a^x f(x_i)dx_i$$

と関係づける事ができる。上式から x は

$$x = F^{-1}(\eta)$$

- この式を解析的に解くことが可能な場合には、 x を直接計算して決めが可能であり、“**直接サンプリング法**”と呼ぶ。



直接サンプリング法の例-飛行距離の決定

- 1個の入射粒子が単位距離あたりに反応する確率を Σ_t とする時、 l と $l+dl$ の間で最初に反応が起きる確率

$$p(l)dl = e^{-\Sigma_t l} \Sigma_t dl$$

$$\eta = P(l) = \int_0^l p(l_1) dl_1 = 1 - e^{-\Sigma_t l}$$

$$l = -\frac{1}{\Sigma_t} \ln(1 - \eta) = -\lambda \ln(1 - \eta)$$

l : 飛行距離

$\lambda=1/\Sigma_t$: 平均自由行程(mean free path)

η : 乱数 ($1-\eta$ と η は等価)

$$l = -\lambda \ln(\eta)$$

放射性同位元素からの γ -線

- γ -線のエネルギーは、**離散的確率変数**
- γ -線のエネルギー E_i は、0-1の乱数, η , を用いて決定することができる。

$$F(E_{i-1}) = \sum_{j=1}^{i-1} p_j \leq \eta < F(E_i) = \sum_{j=1}^i p_j$$

p_j : E_j を放出する確率密度関数

$F(E_i)$: E_i の累積分布関数

γ -線のエネルギーをサンプリングするには、3つの方法がある。

- (1) “if statement”を使用する方法
- (2) “data statement”を使用する方法
- (3) “data file”を使用する方法

Co-60からの γ -線 : "if statement"を使用する方法

cp ucsource.f ucsource1_0.f : Cygwin/Linuxの場合

COPY ucsource.f ucsource1_0.f : Windowsの場合

ucsource1_0.f を以下のように変更する。

- nsebin=1 → nsebin=2
- esbin(1), spg(1), spe(1) → * esbin(2), spg(2), spe(2)
- ekein=1.253 → ekein=1.333

esbin(1)=ekein

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

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

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

ucsource1_0.f をegs5runで実行

- Cygwin/Linuxの場合
 - ユーザーコードとして ucsource1_0 を入力
 - ユニット4のファイルとして ucsource を入力
 - ユニット25には、何も入力せずリターンする。
 - “Does this user code read from the terminal?”に対して1を入力する
- Windowsの場合
 - egs5run ucsource1_0 ucsource
- 結果の出力 egs5job.out でサンプリングされた光子スペクトルを調べる。
 - 1.173 MeV と1.333 MeVの γ 線の割合は？

結果の例

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

Co-60からの γ -線 : ”data statement”を使う方法

cp ucsource1_0.f ucsource1_1.f

ucsource1_1.f を以下のように変更する:

esbin(2),spg(2),spe(2) → esbin(2),spg(2),spe(2) ,**espdf(2),escdf(2)**

* j,k,n,nd,ner,nsebinの後に以下のデータ文を挿入する。

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

```
nsebin=2
```



```
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
```

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

次の2行を削除

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

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



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

ucsource1_1.f をegs5runで実行

- Cygwin/Linuxの場合
 - ユーザーコードとして ucsource1_1 を入力する
 - ユニット4のファイルとして ucsource を入力する
 - ユニット25のファイルには、何も入力せずリターン.
 - “Does this user code read from the terminal?”に対して1を入力する
- Windowsの場合
 - egs5run ucsource1_1 ucsource
- 結果の出力 egs5job.out 中のサンプリングされた光子スペクトルを調べる
 - 1.173 MeVと 1.333 MeVの γ 線の割合は？

Co-60からの γ 線 : "data file"を使用する方法

```
cp ucsource1_1.f ucsource1_2.f
```

ucsource1_2.f を以下の様に変更する:

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



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

右記の2行を削除する

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

線源のデータファイルに関するオープン文を追加

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



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

co60.inp は γ 線のエネルギーとその確率密度関数(pdf)のデータファイルであり、配布ファイル中に含まれている

1.173,1.333	← エネルギー
0.5,0.5	← 確率密度関数

nsebin=2



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

結果の出力部を変更

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



```
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)
```

ucsource1_2.f を egs5run で実行

- Cygwin/Linuxの場合
 - ユーザーコードとしてucsource1_2を入力する
 - ユニット4のファイル名として ucsource を入力する.
 - ユニット25のファイル名には何も入力せずリターン
 - “Does this user code read from the terminal?”に対して1を入力
- Windowsの場合
 - egs5run ucsource1_2 ucsource
- 結果の出力egs5job.out中で、サンプリングされた γ 線スペクトルを調べる。
 - 1.173 MeVと1.333 MeVの γ 線の割合は？

^{192}Ir からの γ 線エネルギーの決定

- Ir-192からの γ -線のエネルギーと崩壊あたりの放出確率は、以下の表に示されている

i	Energy(MeV)	Emission rate(%)
1	0.296	28.7
2	0.308	30.0
3	0.317	82.7
4	0.468	47.8
5	0.589	4.5
6	0.604	8.2
7	0.612	5.3

Ir-192からの γ -線

cp ucsource1_2.f ucsource2.f

ucsource2.f を以下の様に変更:

線源データファイルの open 文を変更する

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



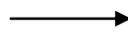
```
open(2,file='ir192.inp',status='unknown')
```

ir192.inp は、線源の γ 線エネルギーとその放出確率に関するデータファイルで、配布ファイル中に含まれている

```
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
```

← エネルギー
← 確率密度関数

```
nsebin=2
```



```
nsebin=7
```

ucsource2.f をegs5runで実行

- Cygwin/Linuxの場合
 - ユーザーコードとしてucsource2を入力
 - ユニット4のファイルとして ucsource を入力
 - ユニット25のファイルとしては何も入力せずリターン
 - “Does this user code read from the terminal?”に対して1を入力
- Windowsの場合
 - egs5run ucsource2 ucsource
- 結果の出力ファイルegs5job.out中で、サンプリングされた γ 線のスペクトルを調べる。
 - spg と espdf を比較する。

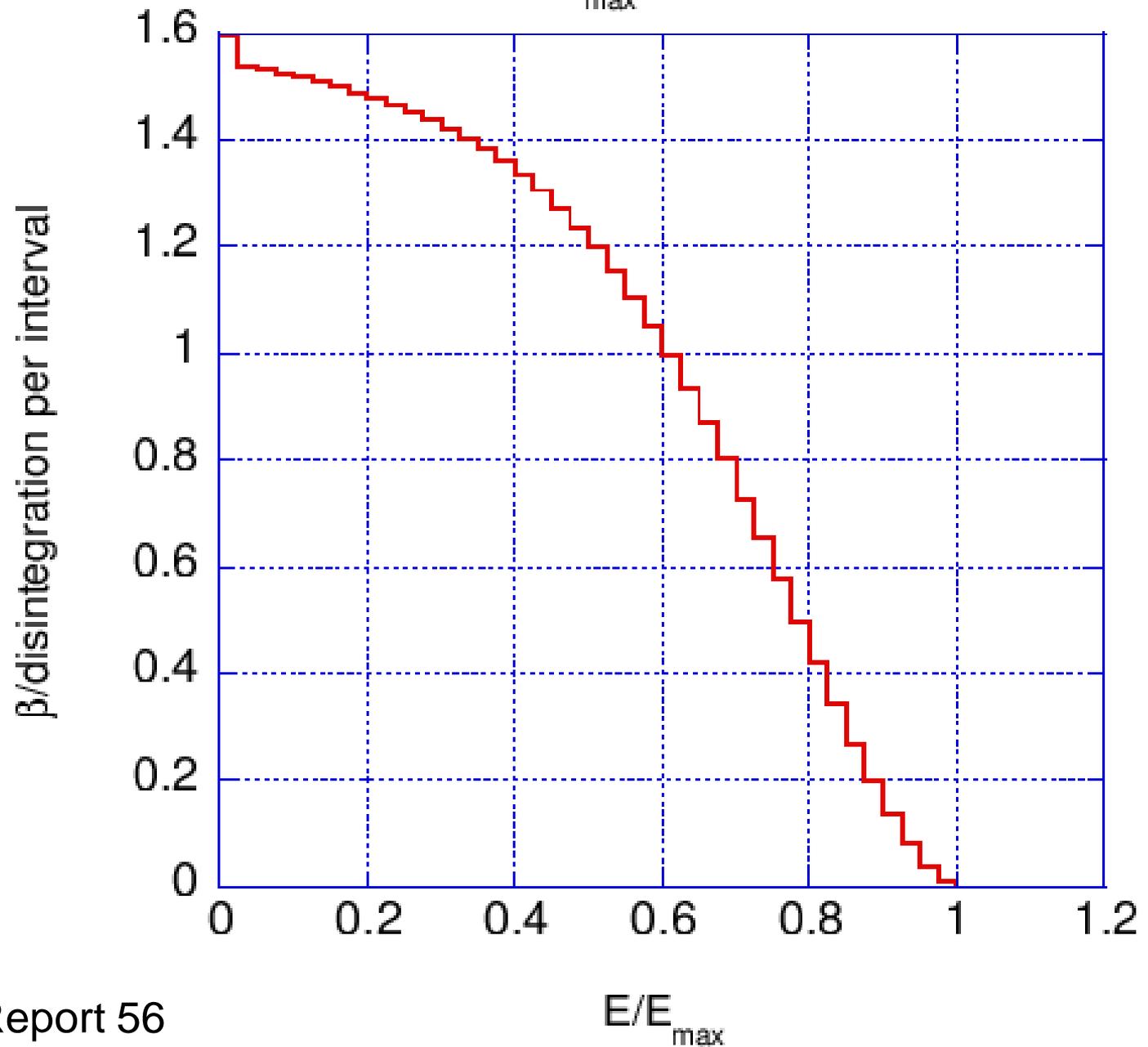
結果の例

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

^{90}Sr からの β 線のエネルギー決定

- β 線のエネルギーは、連続であり、この点が γ 線の場合と異なる。
- 連続分布の場合は、一般的に直接法によるサンプリングが難しい。
- 近似的な方法として、以下の方法がある：
 - β 線のスペクトルを等間隔の n 個のビンに分割する。
 - 全領域の積分値に対する個々のビン間の積分値の割合を確率密度関数とする。
 - 確率密度関数から作成した累積分布関数を用いてビン番号を決定する。
 - 個々のビンでは、エネルギーは一様分布をしていると仮定して、 β 線のエネルギーを決定する。

^{90}Sr ($E_{\text{max}}=0.546\text{MeV}$)



Sr-90からのβ線エネルギーのサンプリング

cp ucsource2.f ucsource3.f

ucsource3.f を以下のように変更:

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

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

線源のデータファイルに関するopen文を変更する。

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

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

「*」などの記号を6カラム目に書くと、その行は前の行からの継続行となる。

sr90.inp は、 β 線の最大エネルギー、ビン数、 β 線の最大エネルギーに対する比で表したビン幅及びエネルギービンあたりの β 線の放出率に関するデータファイルで、配布ファイルに含まれている。

0.546	←	β -線の運動エネルギーの最大値
41	←	エネルギービン数
0.025	←	エネルギービン幅 (E/E_{\max})
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		

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

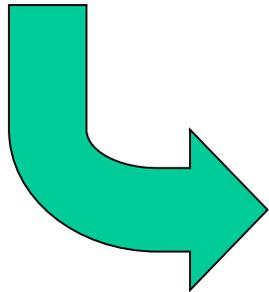


```
read(2,*) emax      ! Maximum beta-ray energy
read(2,*) nsebin    ! Number of source energy bins
read(2,*) deltaes   ! Source energy bin width in MeV
read(2,*) (espdf(i),i=1,nsebin)
```

```
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
```

iqin=0 ! Incident charge - photons



```
tnum=0.D0
do ie=1,nsebin
  esbin(ie)=(ie-1)*deltaes*emax
  tnum=tnum+espdf(ie)
end do
```

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

iqin=-1 ! Incident charge - electrons

ヒストリー数を増やす。

```
ncases=10000
```



```
ncases=100000
```

線源のエネルギーのサンプリング部分を変更する。

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



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

```
do ie=1,nsebin
```

```
→ 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)/tnum  
170  FORMAT(G10.5,' MeV--',8X,G12.5,8X,G12.5,8X,G12.5)  
    end do
```



```
    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
```

ucsource3.f を egs5run で実行

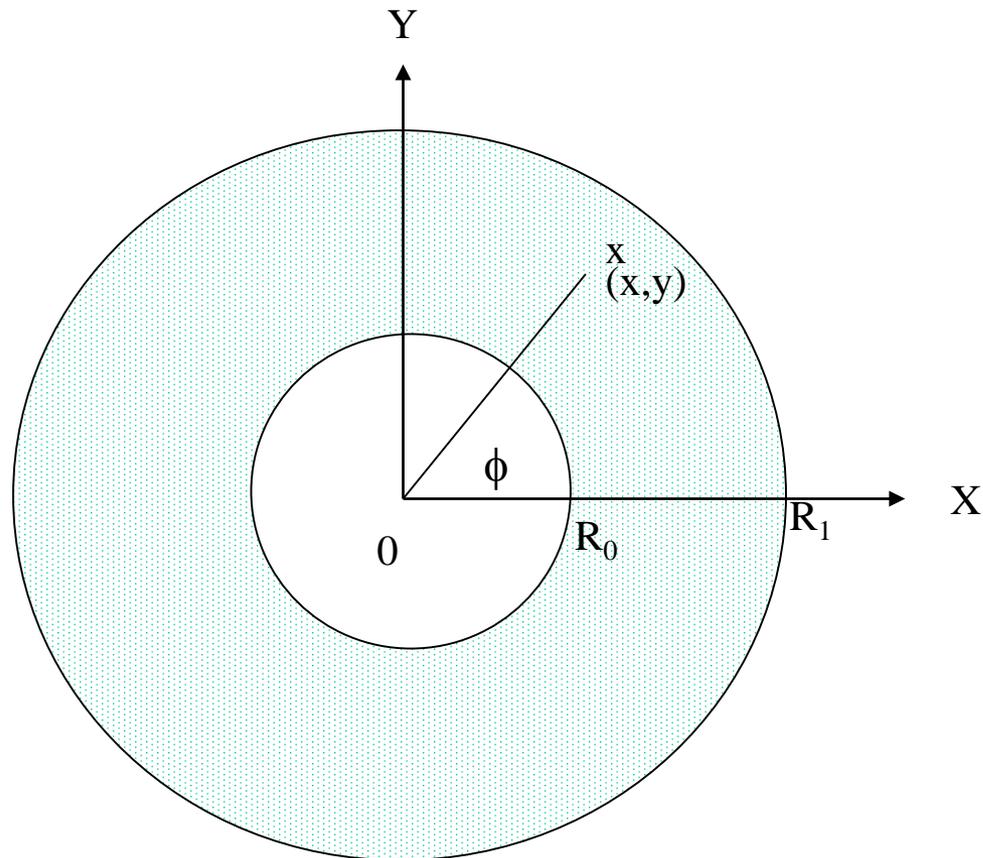
- Cygwin/Linuxの場合
 - ユーザーコードとして ucsource3 を入力
 - ユニット4のファイル名として ucsource を入力
 - ユニット25のファイル名として何も入力せずリターン
 - “Does this user code read from the terminal?”に対して 1を入力
- Windowsの場合
 - egs5run ucsource3 ucsource
- 結果の出力ファイルegs5job.outで、サンプリングされた β 線のスペクトルを調べる。
 - spe(ie) と espdf(ie-1)/tnumを比較する。

結果の例

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

$R_0 < R < R_1$ 領域に一様に分布した面線源

- X-Y平面で、半径 R_0 と R_1 の間に一様に線源が分布している面線源



- この場合、半径に関する確率密度関数(PDF)は以下のようになる。

$$f(r)dr = 2\pi dr / \pi(R_1^2 - R_0^2) = 2rdr / (R_1^2 - R_0^2); \int_{R_0}^{R_1} f(\xi)d\xi = 1.$$

- 半径(r)は、以下の式を解くことにより求めることができる。

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

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

- x と y は、以下の式から決定する。

$$x = r \cos(\phi)$$

$$y = r \sin(\phi)$$

線源の位置:直接法

cp ucsource.f ucsource4.f

ucsource4.f を以下の様に変更:

Local variable を追加する。

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



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

r02及びr12を規定する文を挿入する。

```
wtin=1.0
```



```
wtin=1.0  
r02=1.5*1.5  
r12=4.0*4.0
```

線源の位置をサンプリングする文を挿入する。

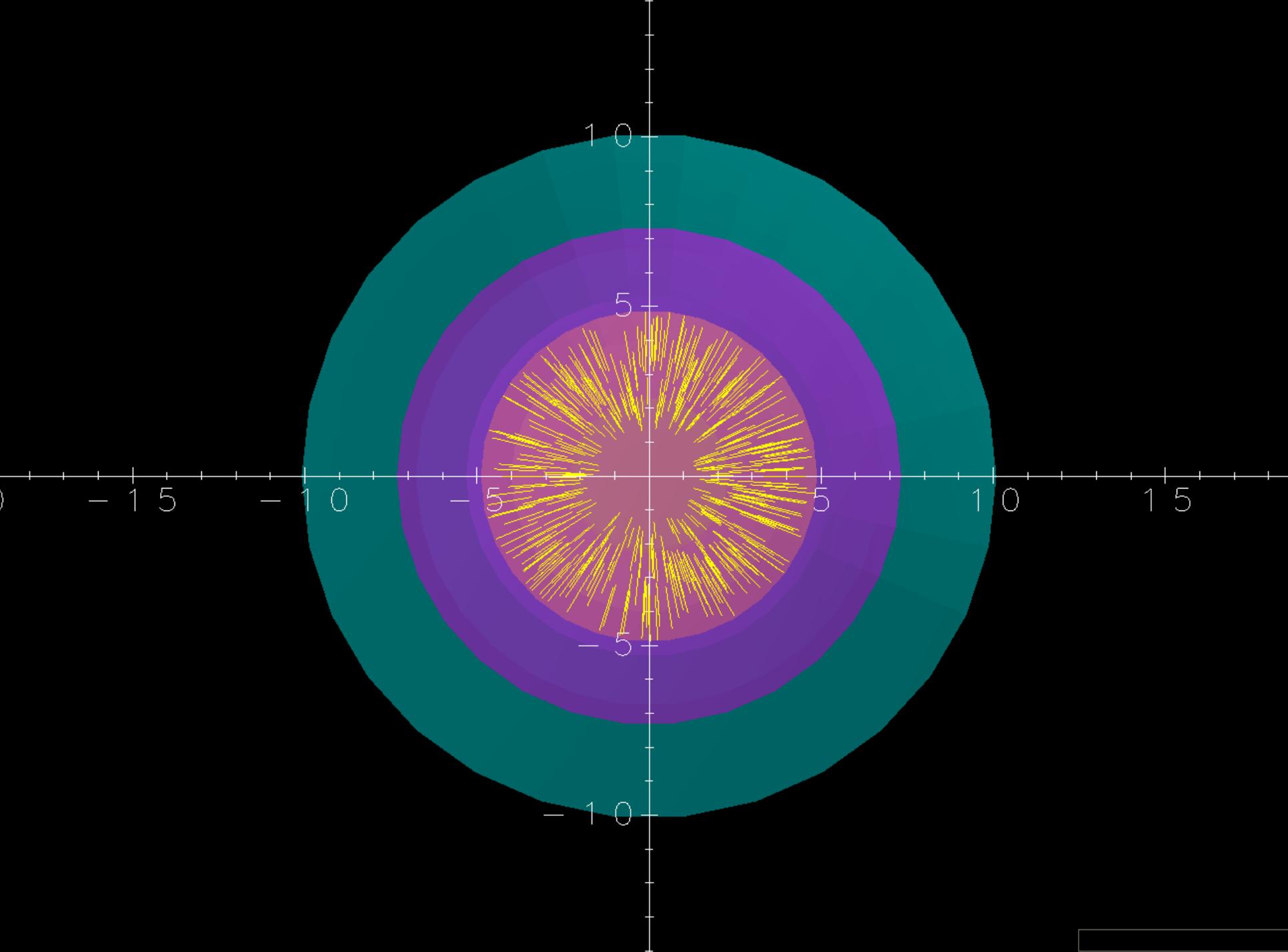
```
! -----  
! Determine source position  
! -----
```



```
! -----  
! 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)
```

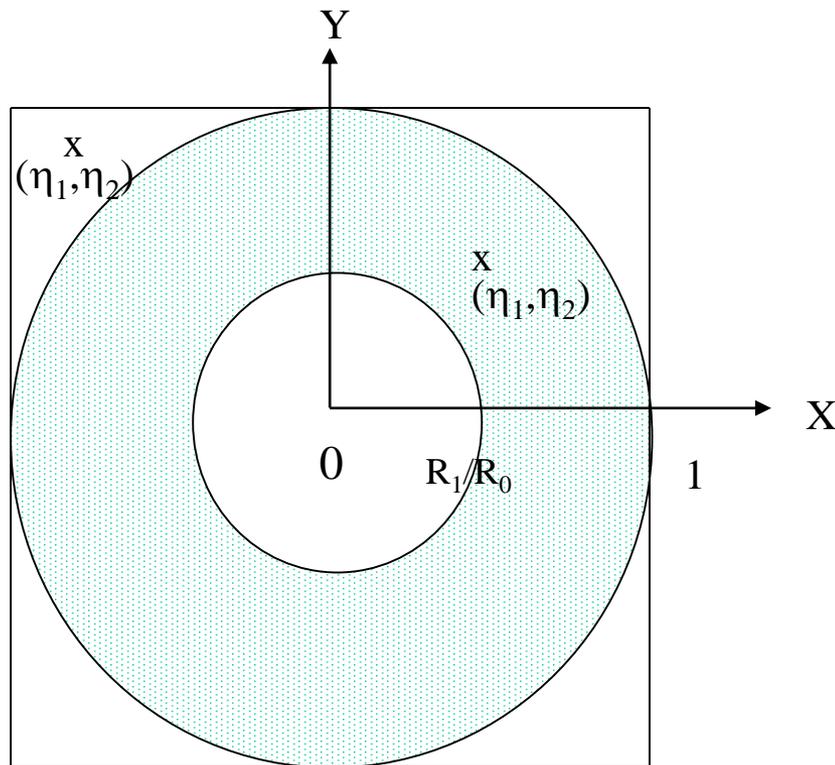
ucsource4.f を egs5run で実行

- Cygwin/Linuxの場合
 - ユーザーコードとして ucsource4 を入力
 - ユニット4のファイル名として ucsource を入力
 - ユニット25のファイル名として何も入力せずリターン
 - “Does this user code read from the terminal?”に対して 1を入力
- Windowsの場合
 - egs5run ucsource4 ucsource
- 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。



線源位置:Rejection 法

- 線源位置 (x,y) は、“rejection” 法により簡単に決定することができる。
- $-1 \leq x \leq 1$; $-1 \leq y \leq 1$ の正方形の領域に一様に分布した点をサンプリングする。
- もし、 (x,y) が $R_0/R_1 < R < 1$ の領域内の点であれば、 x 及び y を R_1 倍した点を線源位置とする。領域外であれば、サンプリングをやり直す。



線源位置:Rejection 法

cp ucsource4.f ucsource5.f

ucsource5.f を以下のように変更:

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



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

半径を自乗の値を定義した文を半径の値を定義する変更する。

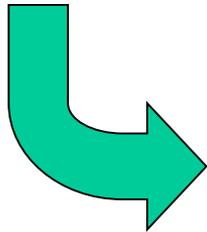
```
r02=1.5*1.5  
r12=4.0*4.0
```



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

位置のサンプリング部分を変更する。

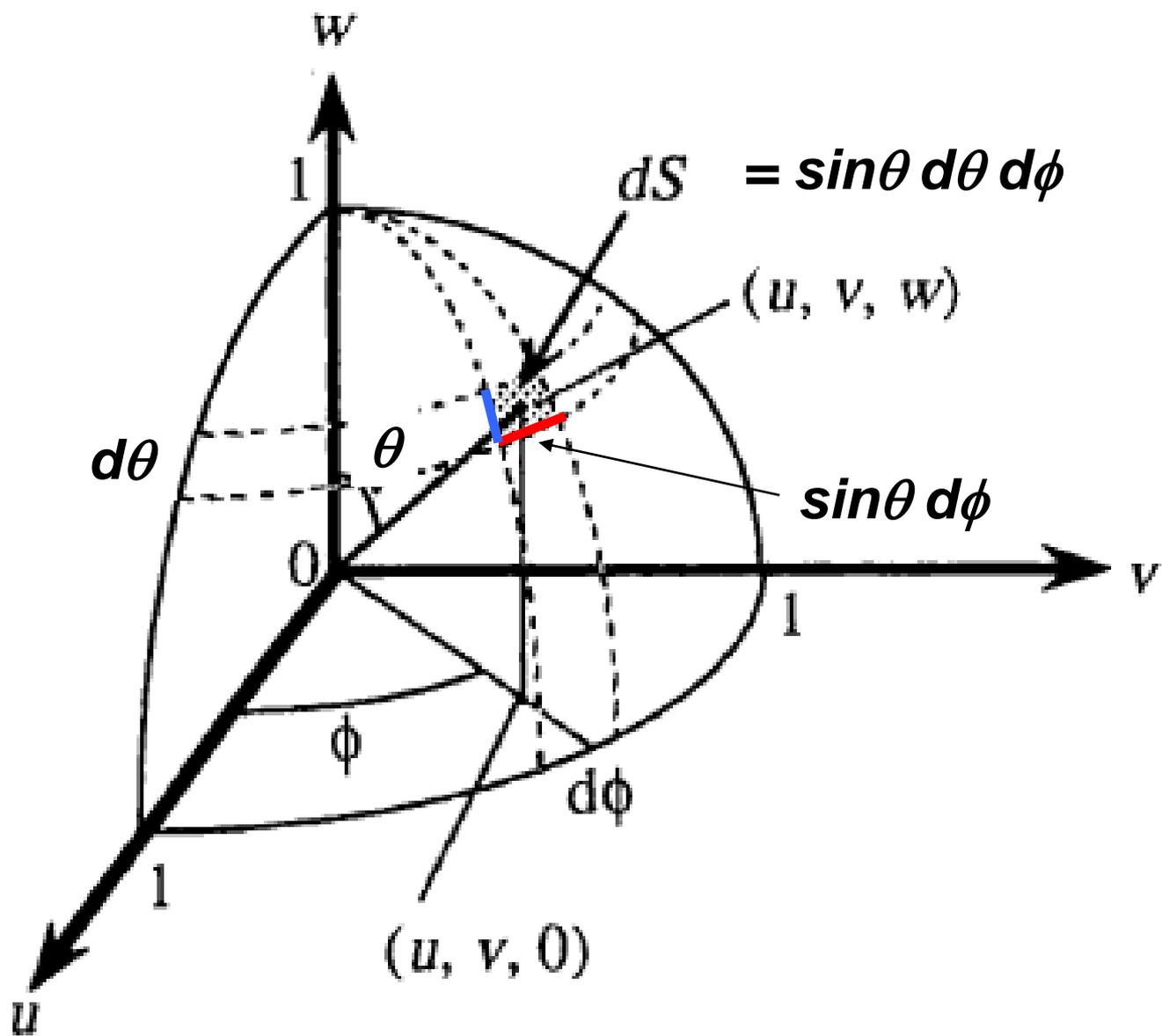
```
! -----  
! Determine source position  
! -----  
call randomset(rnnow)  
rr0=sqrt(r02+rnnow*(r12-r02))  
call randomset(rnnow)  
phai=PI*(2.0*rnnow-1.0)  
xin=rr0*cos(phai)  
yin=rr0*sin(phai)
```



```
! -----  
! 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
```

ucsource5.f を egs5run で実行

- Cygwin/Linuxの場合
 - ユーザーコードとして ucsource5 を入力
 - ユニット4のファイル名として ucsource を入力
 - ユニット25のファイル名として何も入力せずリターン
 - “Does this user code read from the terminal?”に対して 1を入力
- Windowsの場合
 - egs5run ucsource5 ucsource
- 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。



球面座標系

点等方線源からの方向の決定：直接法

等方線源のZ方向の方向余弦は、以下の様にしてサンプリングすることができる。

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

$$w = \cos \theta$$

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

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

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

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

線源が 2π 領域にのみ放出される線源の場合には、 w は以下のようにしてサンプリングすることができる。

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

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

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

線源の方向 (2π):直接法

cp ucsource.f ucsource6.f

ucsource6.f を以下のように変更する。

phai と rr0 を local variable に追加する。

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



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

方向のサンプリングルーチンを挿入する。

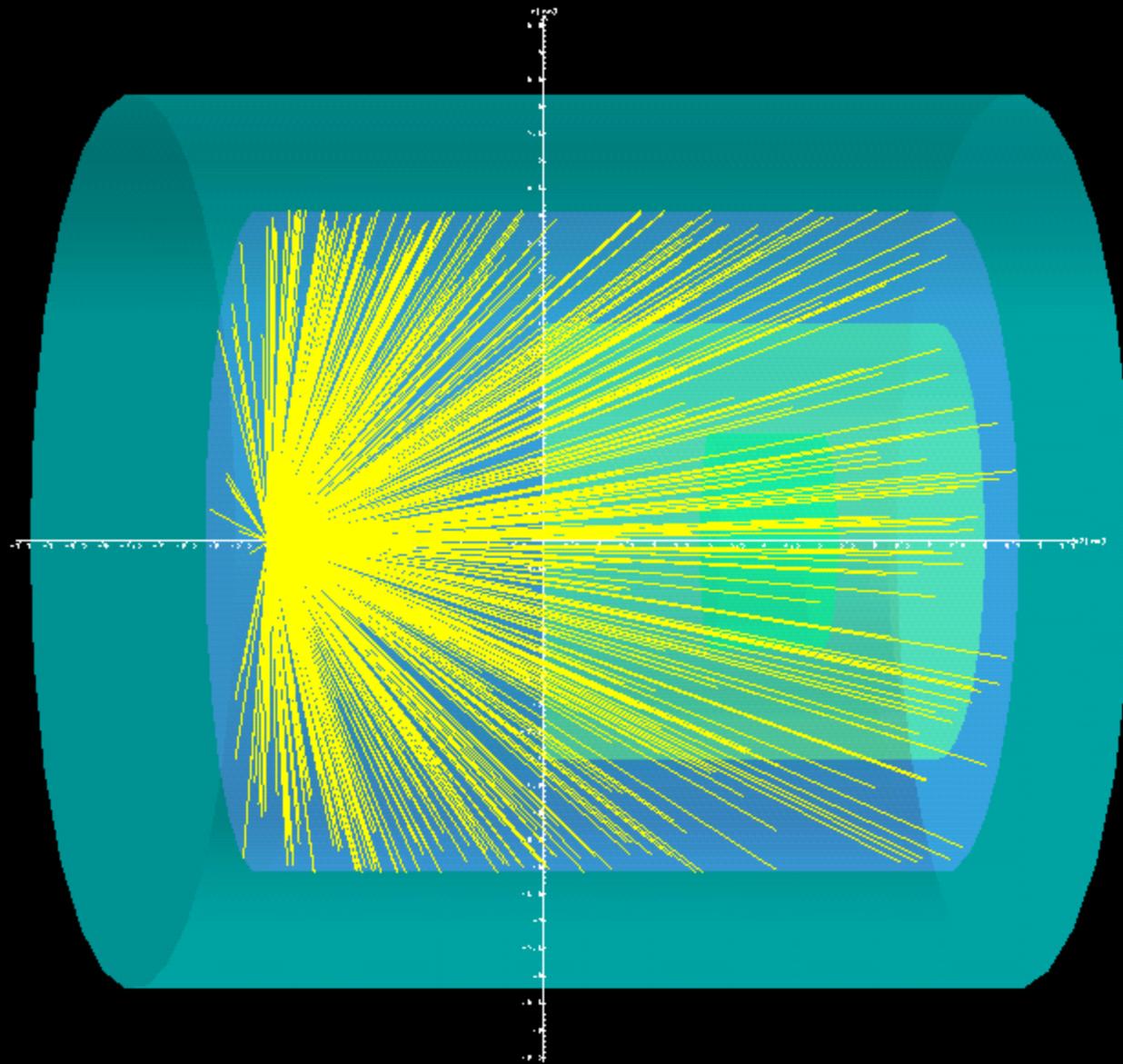
```
! -----  
! Determine source direction  
! -----
```



```
! -----  
! 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)
```

ucsource6.f をegs5runで実行

- Cygwin/Linuxの場合
 - ユーザーコードとして ucsource6 を入力
 - ユニット4のファイル名として ucsource を入力.
 - ユニット25のファイル名には、何も入力せずにリターン
 - “Does this user code read from the terminal?”に対して 1を入力
- Windowsの場合
 - egs5run ucsource6 ucsource
- Cgviewで飛跡を調べる。
 - Z-Y 面表示にする。
 - 光子が 2π の領域に等方的に放出していることを確認する。



等方線源からの方向の決定 : Rejection 法

- この問題では、Rejection 法がより効果的である。
- $-1 \leq x \leq 1; -1 \leq y \leq 1; -1 \leq z \leq 1$ の直方体内に一様に分布した点 (x, y, z) をサンプリングする。
- もし、この点が半径1の球内の点であれば、

$$R = \sqrt{x_1^2 + y_1^2 + z_1^2} \leq 1,$$

以下の式から u, v, w を決定する。

$$u = x_1 / R; v = y_1 / R; w = z_1 / R.$$

- 球外の点の場合は、サンプリングをやり直す。

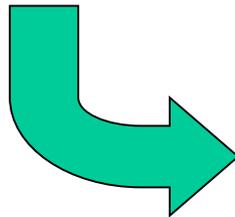
線源の方向 (2π):Rejection 法

cp ucsource6.f ucsource7.f

ucsource7.f を変更

方向のサンプリング部分を変更する。

```
! -----  
! 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)
```



```
! -----  
! 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.  
      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
```

ucsource7.f をegs5runで実行

- Cygwin/Linuxの場合
 - ユーザーコードとして ucsource7 を入力
 - ユニット4のファイル名として ucsource を入力.
 - ユニット25のファイル名には、何も入力せずにリターン
 - “Does this user code read from the terminal?”に対して 1を入力
- Windowsの場合
 - egs5run ucsource7 ucsource
- Cgviewで飛跡を調べる。
 - Z-Y 面表示にする。
 - 光子が 2π の領域に等方的に放出していることを確認する。

変更記録

- 2008-07-25
 - Determine position → Determine source position
 - Typo級の修正のためファイル名、日付は変更せず。
- 2008-9-30
 - 「*などの記号を6カラム目に書くと、その行は前の行からの継続行となる」という記述を21頁に追加
- 2009-3-6
 - ekinを削除
 - 関連したreal*8部分の不整合を修正(テキストに対応する様に)
- 2009-6-24
 - DOSの場合の入力を追加
- 2013-08-12
 - Ucsource1_0 → ucsource1_1 (Page 10)
- 2018-08-05
 - DOS -> Windows