

サンプルユーザーコード ucnaicgv

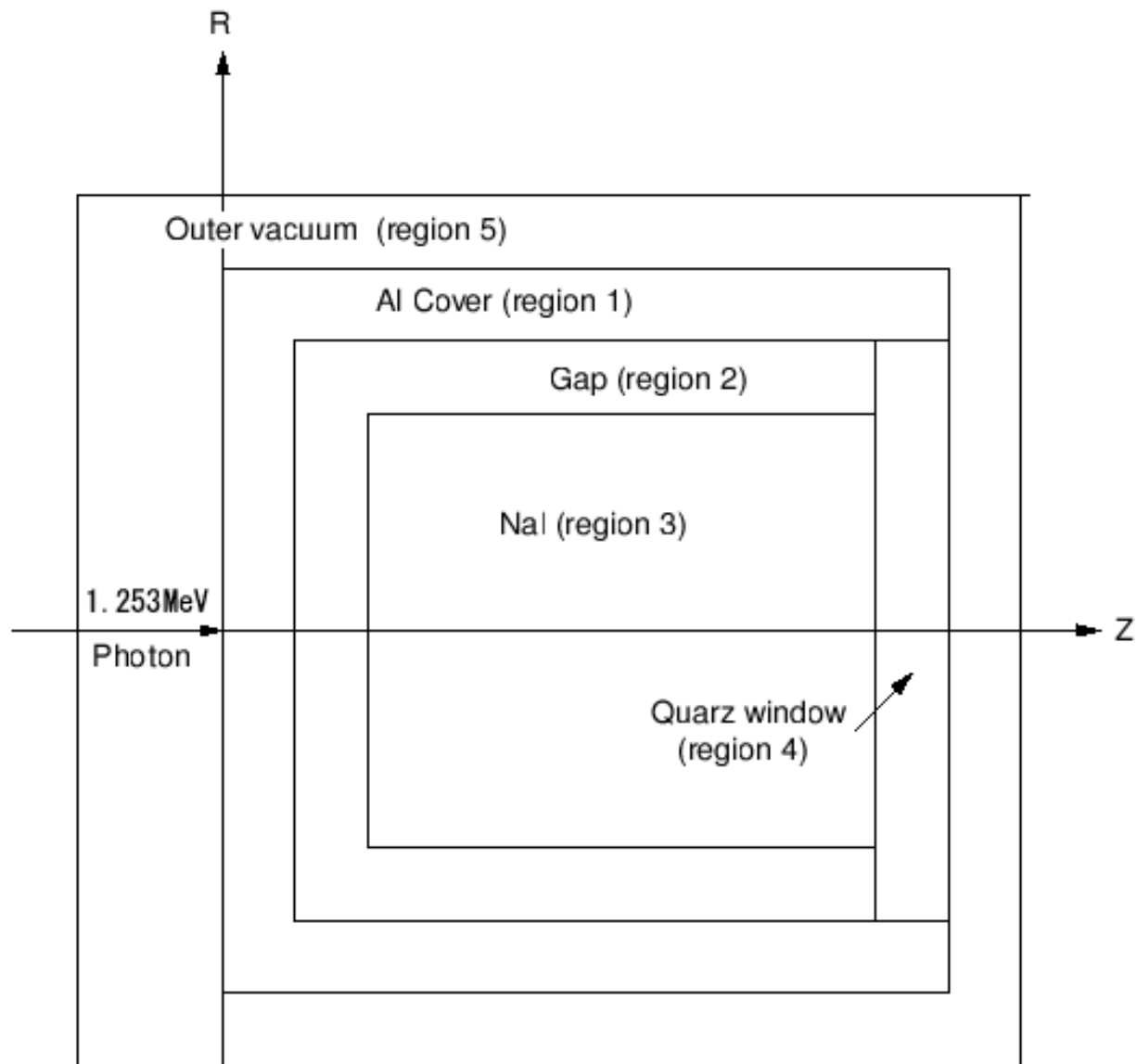
平山 英夫、波戸 芳仁
KEK, 高エネルギー加速器研究機構

テキスト: naicgv.pdf,
egs5_user_manual.pdf

ユーザーコードで利用可能な変数、
オプションについては
`egs5_user_manual`を参照

ucnaicgv.f

- 計算課題: NaI検出器のレスポンス計算
- 形状: CG形状(RCC:円筒)
- 1.253MeV γ 線のペンシルビーム
- 出力
 - 飛跡 (CGView): egs5job.pic
 - 計算結果: egs5job.out



Step 1: Initialization

- egs5及びpegs5で使われているcommonは、それぞれincludeディレクトリー及びpegscommonsディレクトリーのファイルを ”include”文で取り込む
- 著者から提供されたジオメトリー関係などのユーザーコードのみで使用されるcommonは、auxcommonsディレクトリーのファイルをinclude文で取り込む

配列の大きさの指定

- commonで使用されている変数の配列の大きさは、parameter文で指定
 - egs5で使用されているcommonの変数は、
include/egs5_h.f
 - ユーザーコードでのみ使用されるcommonの変数は、
auxcommns/aux_h.f
- commonと同じようにinclude文により取り込まれる。
- 配列の大きさを変更する場合は、parameter文の変数を変更する

```
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'
```

egs5 common に含まれる変数をメインプログラム等のプログラム単位で使用する場合は、include文で当該commonを指定

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'**

ジオメトリー関係等ユーザーコード
のみで使用されるcommon

CG関係のcommonで、CGを使用する場合には常に必要(変更無し)

**include 'auxcommons/geom_common.f' ! geom-common file
integer irinn**

include/egs5_h.f 内

! Maximum number of different media (excluding vacuum)

integer MXMED

parameter (MXMED = 4)

物質の数を増やしたい場合には、この数値を変更する。

include/egs5_misc.f 内

common/MISC/

! Miscellaneous COMMON

*** rhor(MXREG), dunit,**

*** med(MXREG), iraylr(MXREG), lpolar(MXREG), incohr(MXREG),**

*** iprofr(MXREG), impacr(MXREG),**

*** kmpi, kmpo, noscat**

real*8

*** rhor, dunit**

integer

*** med, iraylr, lpolar, incohr, iprofr, impacr, kmpi, kmpo, noscat**

common/totals/

! Variables to score

このユーザーコード固有の
common

*** depe,deltae,spec(3,50),maxpict**

real*8 depe,deltae,spec

integer maxpict

main programで使用
する倍精度の実数

real*8

! Local variables

*** availke,avpe,avph,avspe,avspg,avspp,avte,desci2,pefs,pef2s,**

*** rr0,sigpe,sigte,sigph,sigspg,sigspe,sigspp,tefs,tef2s,wtin,wtsum,**

*** xi0,yi0,zi0**

real*8

*** phs(50),ph2s(50),specs(3,50),spec2s(3,50)**

real **! Local variables**

* **elow,eup,rdet,rtcov,rtgap,tcov,tdet,tgap**

main programで使用する単
精度の実数

real

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

integer

* **i,icases,idin,ie,ifti,ifto,ii,iiz,imed,ireg,isam,**

* **izn,nlist,j,k,n,ner,ntype**

main programで使用する整数

Open文

- ユーザーコードから、pegsを実行するのに伴い、ユニット7-26は、pegsで close されることから、メインプログラムで open していても、pegs実行後に、再度 open することが必要となる。そのため、ユニット7-26の使用を避ける方が良い。
- 飛跡情報を出力するplotxyz.fのユニットは、9から39に変更

```
open(6,FILE='egs5job.out',STATUS='unknown')  
open(4,FILE='egs5job.inp',STATUS='old')  
open(39,FILE='egs5job.pic',STATUS='unknown')
```

Step 2:pegs5-call

- 物質データ及び各物質のcharacteristic distanceを設定した後で、pegs5をcallする。

```
nmed=4
```

```
medarr(1)='NAI
```

```
medarr(2)='AL
```

```
medarr(3)='QUARTZ
```

```
medarr(4)='AIR-AT-NTP
```

- pegs5で作成する物質データの名前。pegs5の入力データ(ユニット24から読み込み)と対応

```
do j=1,nmed
```

```
do i=1,24
```

```
media(i,j)=medarr(j)(i:i)
```

```
end do
```

```
end do
```

各物質のcharacteristic distance
当該物質のリージョンで中、最も小さいサイズを指定

```
chard(1) = 7.62d0
```

```
chard(2) = 0.1d0
```

```
chard(3) = 0.5d0
```

```
chard(4) = 5.0d0
```

! optional, but recommended to invoke

! automatic step-size control

Step 3:Pre-hatch-call-initialization

```
write(6,*) 'Read cg-related data'
```

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

```
!   Initialize CG related parameters
```

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

```
  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-TIME')")
```

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

```
  rewind ifti
```

```
  call geomgt(ifti,ifto)! Dummy call to write geom info for ifto
```

```
  write(ifto,110)
```

```
110  FORMAT('CEND')
```

RCC	1	0.00	0.0	0.0	0.00	0.0	NaIの領域(円筒)を定義するためのbody
		7.62	3.81				
RCC	2	0.00	0.0	-0.5	0.00	0.0	空隙に内包される領域(円筒)を定義するためのbody
		8.12	4.31				
RCC	3	0.00	0.0	-0.6	0.00	0.0	Alカバーに内包される領域(円筒)を定義するためのbody
		8.72	4.41				
RCC	4	0.00	0.0	7.62	0.00	0.0	フォトマルの窓ガラス領域(円筒)を定義するためのbody
		0.5	4.31				
RCC	5	0.00	0.0	-5.6	0.00	0.0	空気の領域(円筒)を定義するためのbody
		18.72	9.41				
RCC	6	0.0	0.0	-10.0	0.0	0.0	体系全体を覆うbody
		30.0	12.0				

END

Z1	+1				NaI
Z2	+2	-1			空隙
Z3	+3	-2	-4		アルミニウム
Z4	+4				窓ガラス
Z5	+5	-3			空気
Z6	+6	-5			計算終了の領域

END

1	0	2	3	4	0	各リージョンの物質指定
---	---	---	---	---	---	-------------

nreg=izonin

! Read material for each region from egs5job.data

read(4,*) (med(i),i=1,nreg) 各リージョンへの物質割り当てデータ読み込み

! Set option except vacuum region

オプションの設定

do i=1,nreg-1

if(med(i).ne.0) then

iphtr(i) = 1 ! Switches for PE-angle sampling

iedgfl(i) = 1 ! K & L-edge fluorescence

iauger(i) = 0 ! K & L-Auger

iraylr(i) = 0 ! Rayleigh scattering

lpolar(i) = 0 ! Linearly-polarized photon scattering

incohr(i) = 0 ! S/Z rejection

iprofr(i) = 0 ! Doppler broadening

impacr(i) = 0 ! Electron impact ionization

end if

end do

リージョン毎に設定できるオプション

ecut, pcut	カットオフエネルギー (全エネルギー)
iphtr	光電子の角度分布のサンプリング
iedgfl	K & L-特性X線の発生
iauger	K & L-オージェ電子の発生
iraylr	レイリー散乱
lpolar	光子散乱での直線偏光
incohr	S/Z rejection
iprofr	ドップラー広がり
impacr	電子衝突電離

乱数(ranlux乱数)

```
! -----  
! Random number seeds. Must be defined before call hatch  
! or defaults will be used. inseed (1- 2^31)  
! -----  
    luxlev = 1  
    inseed=1  
    write(1,120) inseed  
120  FORMAT(/,' inseed=',I12,5X,  
    *      '(seed for generating unique sequences of Ranlux)')  
  
!      =====  
    call rluxinit ! Initialize the Ranlux random-number generator  
!      =====
```

異なったinseed毎に、重複しない乱数を発生することが可能
並列計算の場合に有効

Step 4: 入射粒子のパラメーター設定

iqin=0	! Incident charge - photons	粒子の種類(電荷)
ekein=1.253	! Kinetic energy	粒子の運動エネルギー
xin=0.0	! Incident at origin	位置
yin=0.0		位置
zin=0.0		方向
uin=0.0	! Moving along z axis	入射粒子のリージョン
vin=0.0		
win=1.0		0に設定するとCGでは、線源位置から調べる。
irin=0	! Starting region (0: Automatic search in CG)	
wtin=1.0	! Weight = 1 since no variance reduction used	
deltae=0.05	! Energy bin of response	

線源領域の決定

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

線源リージョンのサーチ

```
if(irin.le.0.or.irin.gt.nreg) then  
  call srzone(xin,yin,zin,iqin+2,0,irin)  
  if(irin.le.0.or.irin.ge.nreg) then  
    write(6,fmt="(' Stopped in MAIN. irin = ',i5)")irin  
    stop  
  end if  
  call rstnxt(iqin+2,0,irin)  
end if
```

Step 5: hatch-call

- 電子・陽電子の全エネルギーの最大値をemaxeを0.d0に設定し、hatch を call する。(hatchで、emaxeを計算する。)
- 読み込んだ情報を確認するために、物質データ及び各リージョンの情報を入力する

```
emaxe = 0.D0 ! dummy value to extract min(UE,UP+RM)
```

```
write(6,130)
```

```
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')
```

```
write(6,140)
```

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

```
! =====
```

```
call hatch
```

```
! =====
```

Step 6: Initialization-for-howfar

- ユーザーコードで使用する形状データを設定する
 - 平板、円筒、球などに関するデータ
- CGを使用しているこのユーザーコードでは、形状に関するデータは、cg入力データとしてstep 6以前に処理しているので、このstepで設定することはない

Step 7: Initialization-for-ausgab

- 計算で求める量の初期化、レスポンスのエネルギービン幅の設定等
- 計算するヒストリー数(ncases)と飛跡表示データを記録するヒストリー数(maxpict)を設定する

! **Set histories**
ncases=10000

! **Set maximum number for pict**
maxpict=50

Step 8: Shower-call

- ncases 回数 call shower を繰り返す。
- 各ヒストリー毎に、線源情報が異なる場合には、call shower の前に、線源情報(粒子の種類、エネルギー、位置、方向)を設定する。
- ヒストリー終了毎に、検出器中の吸収エネルギー等の分析を行う。

! -----
! **Select incident energy**
! -----

wtin = 1.0

wtsum = wtsum + wtin	! Keep running sum of weights
etot = ekein + 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

このユーザーコードでは、単一エネルギーの光子(iqin=0)なので、各ヒストリーで初期設定した同じ値を使用しているが、ヒストリー毎にエネルギーが異なる場合(分布している場合、複数のγ線を放出する線源)には、ekeinを決定するサンプリングルーチンが必要

egs5で使用するエネルギー(showerに引き渡すエネルギー)は、全エネルギーなので、etotを設定する。(電子・陽電子の場合は、運動エネルギーに電子の静止質量を加える。

ヒストリー毎に線源の方向が異なる場合には、ここに、線源の方向を決定するルーチンを挿入する。

```
! -----  
! Select incident angle  
! -----  
  
! -----  
! Print first NWRITE or NLINES, whichever comes first  
! -----  
if (ncount .le. nwrite .and. ilines .le. nlines) then  
  ilines = ilines + 1  
  write(6,280) etot,xin,yin,zin,uin,vin,win,iqin,irin,idin  
280  FORMAT(7G15.7,3I5)  
end if  
  
! -----  
! Compare maximum energy of material data and incident energy  
! -----  
if(etot+(1-iabs(iqin))*RM.gt.emaxe) then  
  write(6,fmt="(' Stopped in MAIN.',  
1  ' (Incident kinetic energy + RM) > min(UE,UP+RM).)')"  
  stop  
end if  
  
! -----  
! Verify the normalization of source direction cosines  
! -----  
if(abs(uin*uin+vin*vin+win*win-1.0).gt.1.e-6) then  
  write(6,fmt="(' Following source direction cosines are not',  
1  ' normarized.',3e12.5)")uin,vin,win  
  stop  
end if
```

iqin, etot, xin, yin, zin, uin, vin, win, irinn, 及び wtin という条件で、showerをスタートする。

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

```
if (depe .gt. 0.D0) then  
  ie=depe/deltae + 1  
  if (ie .gt. 50) ie = 50  
  ph(ie)=ph(ie)+wtin  
  tef=tef + wtin  
  if(depe .ge. ekein*0.999) pef=pef +wtin  
  depe = 0.D0  
end if
```

ヒストリー毎の情報を処理する。この処理が必要かどうかは、問題に依存する。

このユーザーコードでは、検出器の効率とレスポンスを計算することを目的としているので、吸収エネルギーが0でない場合は、当該ヒストリーでの検出器中の吸収エネルギーからエネルギー番号を決め、その値を+1する。吸収エネルギーの値から、全検出効率を+1し、吸収エネルギーが入射粒子の運動エネルギーと見なされる場合にはピーク検出効率を+1する。

この計算では、エネルギー吸収をあると全て検出されるとして全検出効率に加えているが、あるエネルギー以上のみを測定する結果との比較の場合には、ピーク検出効率と同じ様な判定が必要になる

Coincidence及びanti-coincidence

- 検出器間でcoincidenceやanti-coincidenceの計算を行う場合の、この例と同様にヒストリー終了毎に、処理を行う
 - Coincidenceの場合は、coincidenceをとる検出器の両方にエネルギー吸収があった場合にのみ主検出器の当該エネルギービンの値を+1増やす
 - Anti-coincidenceの場合は、逆に、主検出器以外の検出器にエネルギー吸収がない場合にのみ主検出器の当該エネルギービンの値を+1増やす

統計的な誤差評価

- x をモンテカルロ計算によって求める量とする。
- MCNPで使用している誤差を評価する方法
 - 計算は N 個の“入射”粒子について行われ、 x_i は、 i -番目のヒストリーの結果であるとする

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad x_i \text{ の平均値}$$

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2 \approx \overline{x^2} - (\bar{x})^2; (\overline{x^2} = \frac{1}{N} \sum_{i=1}^N x_i^2) \quad x_i \text{ の分散}$$

$$s_{\bar{x}}^2 = \frac{1}{N} s^2 \approx \frac{1}{N} [\overline{x^2} - \bar{x}^2] \quad \bar{x} \text{ の分散}$$

$$s_{\bar{x}} \approx \left[\frac{1}{N} (\overline{x^2} - \bar{x}^2) \right]^{1/2} \quad \text{標準偏差}$$

!
! If some energy is deposited inside detector add pulse-height and efficiency.

```
if (depe .gt. 0.D0) then
  ie=depe/deltae + 1
  if (ie .gt. 50) ie = 50
  phs(ie)=phs(ie)+wtin
  ph2s(ie)=ph2s(ie)+wtin*wtin
  tefs=tefs + wtin
  tef2s=tef2s + wtin*wtin
  if(depe .ge. ekein*0.999) then
    pefs=pefs +wtin
    pef2s=pef2s +wtin*wtin
  end if
  depe = 0.D0
end if
do ntype=1,3
  do ie=1,50
    specs(ntype,ie)=specs(ntype,ie)+spec(ntype,ie)
    spec2s(ntype,ie)=spec2s(ntype,ie)+
*     spec(ntype,ie)*spec(ntype,ie)
    spec(ntype,ie)=0.D0
  end do
end do
```

MCNPの方法で誤差を評価するために、
ヒストリー毎の計算すべき量とその自乗
の和を求める。

Step 9: Output-of-results

- 線源条件や、形状等の情報の出力
 - どの様な計算であるかを示すために出力
 - cgの場合は、形状をデータから直接示すことが容易でないので、必要な情報を設定して出力する
- ヒストリー毎に得られた求めたい量の和とその自乗和から、求めたい量の平均値と統計的な誤差を計算し、出力する

ピーク検出効率

```
! -----  
! Peak efficiency  
! -----  
avpe = pefs/ncount  
pef2s=pef2s/ncount  
sigpe=dsqrt((pef2s-avpe*avpe)/ncount)  
avpe = avpe*100.0  
sigpe = sigpe*100.0  
write(6,350) avpe,sigpe  
350 FORMAT(' Peak efficiency =',G11.4,'+-',G9.2,' %')
```


ausgab の機能

- ausgab は、ユーザーが得たい情報を記録するサブルーチンである
- NaI検出器中での沈着エネルギーの記録

```
! -----  
! Score energy deposition inside NaI detector  
! -----  
if (med(irl). eq. 1) then  
    depe = depe + edepwt
```

当該リージョンの物質番号(`med(irl)`)が、1 (NaI)の時、
検出器中のエネルギー付与を加算

ausgab の機能

- 検出器外部から、検出器に入射した各粒子のエネルギー情報の記録

```
! -----  
! Score particle information if it enters from outside  
! -----
```

```
if (irl .ne. irold .and. iarg .eq. 0) then ← 粒子の移動に伴い、リージョンが変わる  
  if (iql .eq. 0) then          ! photon      =検出器の外から入射  
    ie = e(np)/deltae +1  
    if(ie .gt. 50) ie = 50  
    spg(1,ie) = spg(1,ie) + wt(np)  
  elseif (iql .eq. -1) then     ! electron  
    ie = (e(np) - RM)/deltae +1  
    if(ie .gt. 50) ie = 50  
    spe(1,ie) = spe(1,ie) + wt(np)  
  else                          ! positron  
    ie = (e(np) - RM)/deltae +1  
    if(ie .gt. 50) ie = 50  
    spp(1,ie) = spp(1,ie) + wt(np)  
  end if  
end if  
end if  
end if
```

howfarの役割

- howfar は、egs にジオメトリーに関する情報を伝えるサブルーチン
- howfar は、ustep の途中で、リージョン境界があるかどうかを調べる。ある場合には、
 - ustep を境界までの距離に置き換える
 - irnew を粒子が入っていくリージョン番号に設定する
- 粒子が、ユーザーが追跡を止めたい領域(例: 体系外)に達したばあいには、idiscard フラグを1に設定する
- 使用するジオメトリルーティン毎に異なったhowfarとなる
 - cgを使用している場合は、このユーザーコードのhowfarを使用する

実習課題

- 実習課題1: NaI検出器の計算
 - 次のように変更して、ピーク検出効率及び全検出効率の変化を調べよ。
 - 線源を、Cs-137の単一エネルギー光子(0.662MeV)に変える。
 - 線源を、Co-60に変え、1.173MeVと1.333MeV光子を同じ確率で発生させる。
 - 1.253MeV線源について、一方向(Z-方向)のみに放出している線源光子を、等方線源に変更する。
 - 1.253MeV線源で、検出器の有感領域の厚さを2倍する。
- 実習課題2: Ge検出器の計算
 - 検出器を、Geに変更して、同じ大きさのNaIと、1.253MeV線源に対するピーク及び全検出効率と比較せよ。
- 実習課題3: 空気電離箱の計算
 - 検出器を、摂氏20°C、1気圧の空気に変え、1.253MeV線源に対して、吸収エネルギーを求めよ。検出器の途中のギャップを除き、3インチ直径で3インチ長さの空気の領域の周辺に厚さ、5mmのAlがある形状とする。
 - 空気のW値(33.97 eV/pair)を用いて、入射光子1個当たりのこの電離箱の出力(Coulomb/source)を求めよ。電荷素量を、 $1.602 \times 10^{-19} \text{ C/e}$ とする。

変更記録

- 2009-6-24
 - ucnaicgv.fとの整合性をとる
- 2011-6-23 (ucnaicgv.fとの整合性をとる)