２００７年前期
A1課題演習

対消滅からの２γ、３γの比と、
ortho-Ps寿命の測定

池田 達哉　許 金祥　下澤 雅明　田中 慎一郎　福田 泰嵩

平成19年10月15日

目 次

1 Introduction

Positronium(Ps) は、電子と陽電子からなり、共に質量が軽いので、電磁相互作用のみによって束縛されている。そのため、QED（量子電磁気学）で記述でき、Ps の実験は QED の検証によく用いられる。

Ps は水素原子と似ているが、電子、陽電子の質量が等しいので、共通重心を中心としてお互いを回る。換算質量が水素原子のほぼ 1/2 倍であることにより、Bohr 半径は水素原子の 2 倍の 106pm であり、イオン化エネルギーは半分で 6.8eV である。

Ps の基底状態は、１重項状態（$^1S_0$,para-Ps,$S = 0$）と 3 重項状態（$^3S_1$,ortho-Ps,$S = 1$）が存在する（図 1a、1b 参照）。

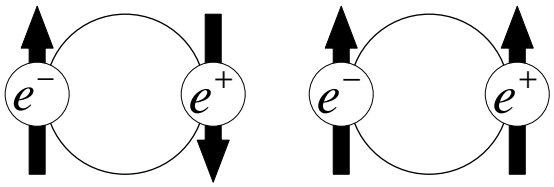

図1a:para-Ps　　図1b:ortho-Ps

para-Ps と ortho-Ps のエネルギー差は $8.4 \times 10^{-4} eV$ であり、para-Ps の方が低いが、Ps の形成は通常このエネルギー間隔が無視できるほど高いエネルギーで起こるので、生成確率は状態数 2S+1 に支配されることになり、para-Ps と ortho-Ps の生成の比は真空中では 1:3 となる。一方、空気中では、pick-off 消滅や、Spin 交換反応により、その比率が変わる。

また、物質中の Ps は通常、対消滅して γ 線を放出する。真空中での寿命はそれぞれ 140psec、123nsec となるが、空気中では、先に述べた比率の変化から寿命も変わってくる。

本実験は空気中での para-Ps と ortho-Ps の生成の比と、ortho-Ps の寿命について計り、その妥当性について考察をする。

2 Theory

2.1 Selection Rule

para-Ps は 2γ、$4\gamma \ldots$ を、ortho-Ps は 3γ、$5\gamma \ldots$ を放出するのだが、この section ではその機構について説明する。（以下では自然単位系、$c = \hbar = 1$ を用いる。）

current vector に荷電共役演算子 C を作用させると、

$$(\hat{j^0}, \hat{\boldsymbol{j}}) \longrightarrow (-\hat{j^0}, -\hat{\boldsymbol{j}}) \tag{1}$$

となる。一方、荷電粒子と電磁場との相互作用が $\hat{A}$ と $\hat{j}$ の積で与えられるので、

$$\Delta H = \int \frac{d^3\boldsymbol{p}}{2\pi^3} eA^\mu j_\mu \tag{2}$$

荷電共役変換の下で、電磁場の相互作用の不変性を考えると、

$$(\hat{\Phi}, \hat{\boldsymbol{A}}) \longrightarrow (-\hat{\Phi}, -\hat{\boldsymbol{A}}) \tag{3}$$

よって、n-photons では

$$C|n\gamma >= (-1)^n |n\gamma > \tag{4}$$

次に、全軌道角運動量 L、全スピン S の Ps を考えると、荷電共役変換の下で、

$$Spin \longrightarrow (-1)^{S+1} \tag{5}$$
$$Y_L^m \longrightarrow (-1)^L \tag{6}$$
$$anti - commutation \longrightarrow (-1)^1 \tag{7}$$

以上から

$$C|Ps >= (-1)^{L+S} |Ps > \tag{8}$$

となる。荷電共役変換の下で、不変に保たれると仮定すると、

$$(-1)^n = (-1)^(L + S)|Ps > \tag{9}$$

$L = 0, S = 0$、para-Ps のときは、偶数個、$L = 0, S = 1$、ortho-Ps のときは、奇数個の photon を放出することがわかる。実際の実験では、検出される光子の数によって para-Ps が対消滅したのか、ortho-Ps が対消滅したのかがわかる。

2.2 Branching Ratio

本実験は空気中で行うが、その比較として真空中での $para - Ps$ と $ortho - Ps$ の比、寿命を求める。
電子、陽電子の Spin は total で見れば、偏極してないはずだから、$para - Ps$ と $ortho - Ps$ の比は、Spin に関して平均をとると

$$para - Ps : ortho - Ps = 1 : 3 \tag{10}$$

となる。

2.3 Lifetime

寿命は、$Lifetime$ = (崩壊確率)$^{-1}$ = (散乱断面積 × FluxDensity)$^{-1}$ から求まる。そこで、まず散乱断面積を求める。2γ への崩壊の Feynman Diagram（図 2.1）から、M 行列は

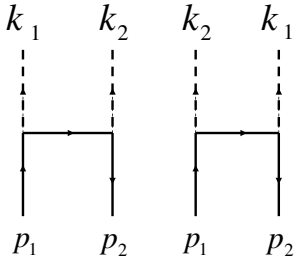

図 2.1:Feynman Diagram for 2 γ

$$
\begin{aligned}
i\mathcal{M} &= \epsilon^*_\mu(-ie\gamma^\mu)\bar{v}(p_2)\frac{-i(\not{p}_1 - \not{k}_1 + m)}{(p_1-k_1)^2-m^2}\epsilon_\nu(-ie\gamma^\nu)u(p_1) \\
&\quad +\epsilon_\nu(-ie\gamma^\nu)\bar{v}(p_2)\frac{-i(\not{p}_1 - \not{k}_2 + m)}{(p_1-k_2)^2-m^2}\epsilon^*_\mu(-ie\gamma^\mu)u(p_1) \\
&= ie^2\epsilon^*_\mu\epsilon_\nu\bar{v}(p_2)\left(\frac{\gamma^\nu \not{k}_2\gamma^\mu - 2\gamma^\mu \not{p}^\nu}{2p_1k_1} + \frac{\gamma^\nu \not{k}_2\gamma^\mu - 2\gamma^\mu \not{p}^\nu}{2p_1k_2}\right)u(p_1) \qquad (11)
\end{aligned}
$$

M 行列の絶対値の 2 乗について Spin 平均、Hericity の和をとり、偏極のない断面積を求める。

$$
\begin{aligned}
&\frac{1}{4}\sum_{e^-e^+}\sum_{polarizations}|\mathcal{M}|^2 \\
= &\frac{e^2}{4}\sum_{e^-e^+}\sum_{polarizations}\epsilon^*_\mu\epsilon_\nu\epsilon_\rho\epsilon^*_\sigma\bar{v}(p_2)\left(\frac{\gamma^\nu \not{k}_2\gamma^\mu - 2\gamma^\mu p^\nu}{2p_1k_1} + \frac{\gamma^\nu \not{k}_2\gamma^\mu - 2\gamma^\mu p^\nu}{2p_1k_2}\right)u(p_1) \\
&\bar{u}(p_1)\left(\frac{\gamma^\sigma \not{k}_2\gamma^\rho - 2\gamma^\rho p^\sigma}{2p_1k_1} + \frac{\gamma^\rho \not{k}_2\gamma^\sigma - 2\gamma^\sigma p^\rho}{2p_1k_2}\right)v(p_2) \qquad (12)
\end{aligned}
$$

$$
\sum_{polarization}\epsilon^*_\mu\epsilon_\rho \to g_{\mu\rho} \qquad (13)
$$

という置き換えをして、

$$
= \frac{e^2}{4}g_{\mu\rho}g_{\nu\sigma}\sum_{p_1}\sum_{p_1}\bar{v}(p_2)\left(\frac{\gamma^\nu \not{k}_2\gamma^\mu - 2\gamma^\mu p^\nu}{2p_1k_1} + \frac{\gamma^\nu \not{k}_2\gamma^\mu - 2\gamma^\mu p^\nu}{2p_1k_2}\right)u(p_1) \qquad (14)
$$

また、

$$\sum_s u(p,s)u(\bar{p},s) = \not{p} + m \tag{15}$$

$$\sum_s v(p,s)v(\bar{p},s) = \not{p} - m \tag{16}$$

という関係から、

$$\begin{aligned}
& \bar{u}(p_1)(\frac{\gamma^\sigma \not{k}_2\gamma^\rho - 2\gamma^\rho p^\sigma}{2p_1k_1} + \frac{\gamma^\rho \not{k}_2\gamma^\sigma - 2\gamma^\sigma p^\rho}{2p_1k_2})v(p_2) \\
= & \frac{e^2}{4}g_{\mu\rho}g_{\nu\sigma}(\not{p}_2 - m)_{da}(\frac{\gamma^\nu \not{k}_1\gamma^\mu - 2\gamma^\mu p^\nu}{2p_1k_1} + \frac{\gamma^\nu \not{k}_2\gamma^\mu - 2\gamma^\nu p^\mu}{2p_1k_2})_{ab} \\
& (\not{p}_1 + m)_{bc}(\frac{\gamma^\sigma \not{k}_1\gamma^\rho - 2\gamma^\rho p^\sigma}{2p_1k_1} + \frac{\gamma^\rho \not{k}_2\gamma^\sigma - 2\gamma^\sigma p^\rho}{2p_1k_2})_{cd} \\
= & \frac{e^2}{4}g_{\mu\rho}g_{\nu\sigma}tr[(\not{p}_2 - m)(\frac{\gamma^\nu \not{k}_1\gamma^\mu - 2\gamma^\mu p^\nu}{2p_1k_1} + \frac{\gamma^\nu \not{k}_2\gamma^\mu - 2\gamma^\nu p^\mu}{2p_1k_2})(\not{p}_1 + m) \\
& (\frac{\gamma^\sigma \not{k}_1\gamma^\rho - 2\gamma^\rho p^\sigma}{2p_1k_1} + \frac{\gamma^\rho \not{k}_2\gamma^\sigma - 2\gamma^\sigma p^\rho}{2p_1k_2})] \\
= & \frac{e^2}{4}\left[\frac{(A)}{4(p_1k_1)^2} + \frac{(B)}{4(p_1k_1)(p_1k_2)} + \frac{(C)}{4(p_1k_2)(p_1k_1)} + \frac{(D)}{4(p_1k_2)^2}\right]
\end{aligned} \tag{17}$$

ここで、対称性から (A) = (C)、(B) = (D) がわかる。

$$\begin{aligned}
= & \ tr\left[(\not{p}_2 - m)(\gamma^\nu \not{k}_1\gamma^\mu - 2\gamma^\mu p^\nu)(\not{p}_1 + m)(\gamma_\nu \not{k}_1\gamma_\mu - 2\gamma_\mu p_\nu)\right] \\
= & \ tr[4\ \not{p}_2\gamma^\mu p_1^\nu\ \not{p}_1\gamma_\mu p^\nu - 2\ \not{p}_2\gamma^\mu p_1^\nu\ \not{p}_1\gamma_\nu\ \not{k}_1\gamma_\nu - 2\ \not{p}_2\gamma^\mu\ \not{k}_1\gamma^\nu\ \not{p}_1\gamma_\mu p_\nu \\
& + \not{p}_2\gamma^\mu\ \not{k}_1\gamma^\nu\ \not{p}_1\gamma_\nu\ \not{k}_1\gamma_\mu - m^2(4\gamma^\mu p_1^\nu\gamma_\mu p_{1\nu} - 2\gamma^\mu p_1^\nu\gamma_\nu\ \not{k}_1\gamma^\mu - 2\gamma^\mu\ \not{k}_1\gamma^\nu\gamma_\mu p_\nu \\
& + \gamma^\mu\ \not{k}_1\gamma^\nu\gamma_\nu\ \not{k}_1\gamma_\mu)]
\end{aligned} \tag{18}$$

また、

$$\begin{aligned}
= & \ tr\left[(\not{p}_2 - m)(\gamma^\mu \not{k}_1\gamma^\nu - 2\gamma^\mu p_1^\nu)(\not{p}_1 + m)(\gamma_\mu \not{k}_2\gamma_\nu - 2\gamma_\nu p_{1\nu})\right] \\
= & \ tr[4\ \not{p}_2\gamma^\mu p_1^\nu\ \not{p}_1\gamma_\nu p_{1\mu} - 2\ \not{p}_2\gamma^\mu p_1^\nu\ \not{p}_1\gamma_\mu\ \not{k}_2\gamma_\nu - 2\ \not{p}_2\gamma^\mu\ \not{k}_1\gamma^\nu\ \not{p}_1\gamma_\nu p_{1\mu} \\
& + \not{p}_2\gamma^\mu\ \not{k}_1\gamma^\nu\ \not{p}_1\gamma_\mu\ \not{k}_2\gamma_\nu - m^2(4\gamma^\mu p_1^\nu\gamma_\nu p_{1\mu} - 2\gamma^\mu p_1^\nu\gamma_\mu\ \not{k}_2\gamma^\nu - 2\gamma^\mu\ \not{k}_1\gamma^\nu\gamma_\nu p_{1\mu} \\
& + \gamma^\mu\ \not{k}_1\gamma^\nu\gamma_\mu\ \not{k}_2\gamma_\nu)]
\end{aligned} \tag{19}$$

ここで、Mandelstam 変数

$$\begin{aligned} s &= (p_1+p_2)^2 = (k_1+k_2)^2 = 2m^2 + 2p_1\cdot p_2 = 2k_1\cdot k \\ t &= (p_1-k_1)^2 = (p_2-k_2)^2 = m^2 - 2p_1\cdot k_1 = m^2 - 2p_2\cdot k_2 \\ u &= (p_1-k_2)^2 = (p_2-k_1)^2 = m^2 - 2p_1\cdot k_2 = m^2 - 2p_2\cdot k_1 \end{aligned} \tag{20}$$

を導入すると、

$$= 16\left[-2m^4 + m^2(m^2-t) + \frac{1}{2}(m^2-t)(m^2-u)\right] \tag{21}$$

$$= 8\left[-4m^4 + m^2(m^2-t) + m^2(m^2-u)\right] = \tag{22}$$

$$= 16\left[-2m^4 + m^2(m^2-u) + \frac{1}{2}(m^2-u)(m^2-t)\right] \tag{23}$$

これより、

$$\frac{1}{4}\sum_{e^-e^+}\sum_{polarizations}|\mathcal{M}|^2 = 2e^2\left[\frac{p_1k_2}{p_1k_2} + \frac{p_1k_1}{p_1k_2} + 2m^2(\frac{1}{p_1k_1}+\frac{1}{p_1k_2}) - m^4(\frac{1}{p_1k_1}+\frac{1}{p_1k_2})^2\right] \tag{24}$$

次に、重心系（CM）での微分散乱断面積、散乱断面積を求め、これを実験室系（LAB）に変換する。

$$\left(\frac{d\sigma}{d\Omega}\right)_{CM} = \frac{1}{2E_A 2E_B|v_A - v_B|}\frac{|\boldsymbol{p}_1|}{(2\pi)^2 4E_{CM}}|\mathcal{M}(p_A,p_B\to p_1,p_2)|^2 \tag{25}$$

これに、

$$\begin{aligned} p_1 &= (E,0,0,p\hat{z}) && (26)\\ p_2 &= (E,0,0,-p\hat{z}) && (27)\\ k_1 &= (E,E\sin\theta,0,E\cos\theta) && (28)\\ k_2 &= (E,-E\sin\theta,0,-E\cos\theta) && (29) \end{aligned}$$

と設定し、適用すると、

$$\left(\frac{d\sigma}{d\cos\theta}\right)_{CM} = \frac{2\pi\alpha^2}{s}(\frac{E}{p})\left[\frac{E^2+p^2\cos^2\theta}{m^2+p^2\sin^2\theta} + \frac{2m^2}{m^2+p^2\sin^2\theta} - \frac{2m^4}{(m^2+p^2\sin^2\theta)^2}\right] \tag{30}$$

$$\sigma_{CM} = \int_0^1 d(\cos\theta)\frac{d\sigma}{d\cos\theta}$$

$$= \frac{2\pi\alpha^2}{s}\left[-\frac{E^2+m^2}{E^2} + \frac{1}{2}\left(\frac{E}{p} + \frac{E^2+2m^2}{Ep} + \frac{m^4}{E^3p}\right)\log\frac{E+p}{E-p}\right] \tag{31}$$

これを、e^- が静止している LAB 系（今回の実験と同じ状況）へ変換する。

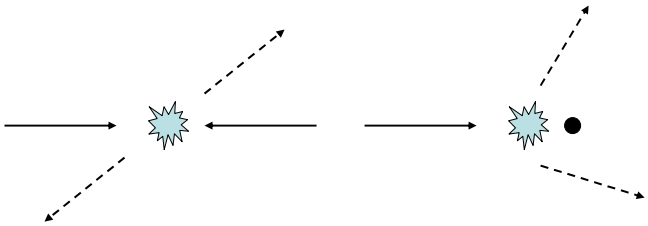

図 2.2:CM 系　　図 2.3:LAB 系

図からわかるように、

$$p_{e^+} = (E, \boldsymbol{p}) \tag{32}$$
$$p_{e^-} = (E, \boldsymbol{0}) \tag{33}$$

となり、LAB 系での断面積は、

$$\sigma_{LAB} = \pi\left(\frac{\alpha}{m}\right)^2 \frac{1}{\gamma+1}\left[-\frac{\gamma+3}{\sqrt{\gamma^2-1}} + \frac{\gamma^2+4\gamma+1}{\gamma^2-1}\log\left(\gamma+\sqrt{\gamma^2-1}\right)\right] \tag{34}$$

ここで $\gamma = \sqrt{1-(\frac{v}{c})^2}$。シリカパウダー中では e^+ はエネルギーを失い、十分低速になるので、$\gamma \sim 1$ として、

$$\sigma_{2\gamma} = \sigma_{LAB} \sim \frac{\pi}{v}\left(\frac{\alpha}{m}\right)^2 \tag{35}$$

となる。3γ への対消滅も上と同様の計算から散乱断面積を求めることができる。ここでは、tree-level の Feynman Diagram、M 行列、散乱断面積を与えておく。3γ の場合

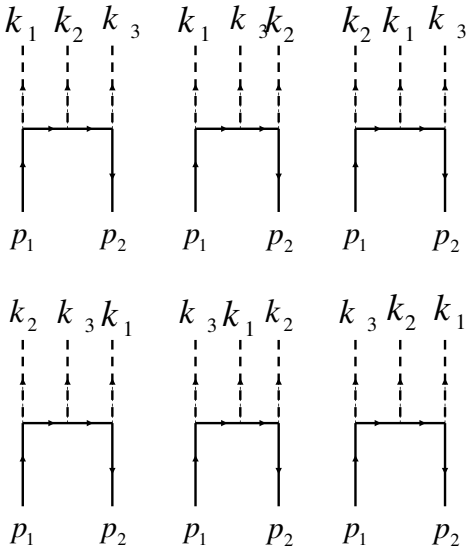

図 2.4:Feynman Diagram for 3γ

$$
\begin{aligned}
i\mathcal{M} &= \bar{v}(p_2)\epsilon^*_\mu(-ie\gamma^\mu)\frac{-i(\not{p}_1 - \not{k}_1 + m)}{(p_1-k_1)^2-m^2}\epsilon^*_\nu(-ie\gamma^\nu)\frac{-i(\not{p}_2 - \not{k}_3 + m)}{(p_2-k_3)^2-m^2}\epsilon^*_\xi(-ie\gamma^\xi)u(p_1) \\
&+ \bar{v}(p_2)\epsilon^*_\mu(-ie\gamma^\mu)\frac{-i(\not{p}_1 - \not{k}_1 + m)}{(p_1-k_1)^2-m^2}\epsilon^*_\nu(-ie\gamma^\nu)\frac{-i(\not{p}_2 - \not{k}_2 + m)}{(p_2-k_2)^2-m^2}\epsilon^*_\xi(-ie\gamma^\xi)u(p_1) \\
&+ \bar{v}(p_2)\epsilon^*_\mu(-ie\gamma^\mu)\frac{-i(\not{p}_1 - \not{k}_2 + m)}{(p_1-k_2)^2-m^2}\epsilon^*_\nu(-ie\gamma^\nu)\frac{-i(\not{p}_2 - \not{k}_1 + m)}{(p_2-k_1)^2-m^2}\epsilon^*_\xi(-ie\gamma^\xi)u(p_1) \\
&+ \bar{v}(p_2)\epsilon^*_\mu(-ie\gamma^\mu)\frac{-i(\not{p}_1 - \not{k}_3 + m)}{(p_1-k_3)^2-m^2}\epsilon^*_\nu(-ie\gamma^\nu)\frac{-i(\not{p}_2 - \not{k}_1 + m)}{(p_2-k_1)^2-m^2}\epsilon^*_\xi(-ie\gamma^\xi)u(p_1) \\
&+ \bar{v}(p_2)\epsilon^*_\mu(-ie\gamma^\mu)\frac{-i(\not{p}_1 - \not{k}_3 + m)}{(p_1-k_3)^2-m^2}\epsilon^*_\nu(-ie\gamma^\nu)\frac{-i(\not{p}_2 - \not{k}_2 + m)}{(p_2-k_2)^2-m^2}\epsilon^*_\xi(-ie\gamma^\xi)u(p_1) \\
&+ \bar{v}(p_2)\epsilon^*_\mu(-ie\gamma^\mu)\frac{-i(\not{p}_1 - \not{k}_2 + m)}{(p_1-k_2)^2-m^2}\epsilon^*_\nu(-ie\gamma^\nu)\frac{-i(\not{p}_2 - \not{k}_1 + m)}{(p_2-k_1)^2-m^2}\epsilon^*_\xi(-ie\gamma^\xi)u(p_1)
\end{aligned}
\tag{36}
$$

$$
\sigma_{3\gamma} = \frac{4(\pi^2-9)}{3v}\alpha\left(\frac{\alpha}{m}\right)^2 \tag{37}
$$

4γ、5γ、それ以上の場合も同様に計算できるが、Feynman Diagram の vertex が１つ増える毎に 2γ、3γ に対して散乱断面積が order として α^2 小さくなる。そこで、今回はこれらの値は無視する。

Flux Density は、Ps の基底状態での波動関数は、$a = \frac{2}{m\alpha^2} = 2a_0$（上で求めた、Bohr 半径の 2 倍）として、

$$
\Psi(r) = \frac{1}{\sqrt{\pi a^3}}\exp^{-\frac{r}{a}} \tag{38}
$$

と表せ、$para-Ps$ と $ortho-Ps$ の比は 1 : 3 となるので、

$$
\Gamma_{para-Ps} \approx 4\Gamma_{2\gamma} = 4\sigma_{2\gamma}v|\Psi(0)|^2 = \frac{1}{2}m\alpha^5 = 0.805\times10^{10}\left[sec^{-1}\right] \tag{39}
$$

となる。よって、

$$
\tau_{para-Ps} = 1.23\times10^{-10}\,[sec] \tag{40}
$$

同様に、

$$
\Gamma_{ortho-Ps} \approx \frac{4}{3}\Gamma_{3\gamma}\frac{2(\pi^2-9)}{9\pi}m\alpha^6 = 0.723\times10^{7}\left[sec^{-1}\right] \tag{41}
$$

$$
\tau_{ortho-Ps} = 1.4\times10^{-7}\,[sec] \tag{42}
$$

となる。

空気中であると上でもふれたように、Spin 交換反応などにより $ortho-Ps$ から $para-Ps$ へと移るために、その比がかわり、寿命も変わってくる。

3 Methods

3.1 positroniumの2γ、3γ崩壊比の測定

3.1.1 実験概容

本実験では、放射線β^+をシリカパウダーに当て、それにより発生したpositoroniumが崩壊するときの2γ、3γ崩壊の比率を測定した。放射線源としては、^{22}Naを使用した。

γ線を観測する装置としてNaI scintillatorを5つ用い、この5つのうちのどれか3つのscintillatorが反応した場合の信号を3γ崩壊の候補となる信号として、5つのうちどれか2つが反応した場合の信号を2γ崩壊の候補となる信号として観測した。もちろん、本実験のセットアップでは実験装置の幾何学的配置などから観測しきれない2γ、3γ崩壊のイベントが存在するため、コンピュータシミュレーションを用いてその分を補正した。

3.1.2 実験装置

以下に本実験で用いた実験装置とそれに関する説明を列挙する。

^{22}Na　β^+の放射線源として用いた。

NaI scintillator　γ線を観測するのに用いた。NaIの結晶部分と光電子増倍管（PMT）からなる(図)。実験では5つを使用した。それぞれのサイズは表に示してある。

シリカパウダー　仕事函数が小さく電子が簡単に自由になる物質で、β^+と反応してpositoroniumを作りやすくするのに用いた。

真空容器　図のように真空容器にはドーナツ状の鉛と放射線源、ビニール袋に入れたシリカパウダーを入れた。空気中ではortho-positroniumの寿命が短くなるので、その影響を軽減するために計画段階では真空中で実験を行う予定であったが容器の問題で今回は空気中で行った。

negative high voltage　NaI scintillatorを作動させるための電源として用いた。（以後、HVと記す。）

amp　scintilatorからのアナログ信号を増幅するのに用いた。（以後、AMPと記す。）

discriminator　NaI scintillatorからの観測信号をデジタル信号に変換するのに用いた。（以後、Discと記す。）

coincidence　複数個の信号が同時に観測されたかどうかを判定するのに用いた。（以後、Coin と記す。）

divider　複数個の信号を加え合わせるのに用いた。（以後、Div と記す。）

fixed delay　信号の到着時間を遅らせるのに用いた。（以後、Delay と記す。）

analog to digital converter　NaI scintillator が受け取った放射線のエネルギーを測定するのに用いた。（以後、ADC と記す。）

time to digital converter　NaI scintillator が受け取った信号の時間差を測定するのに用いた。（以後、TDC と記す。）

gate generator　デジタル信号の幅を広げるのに用いた。（以後、Gate と記す。）

attenuator　送られてくる信号の最大電圧の大きさを調節するのに用いた。（以後、Atten と記す。）

scaler　観測している信号がどれぐらいの頻度で来ているのかを調べるのに用いた。

oscilloscope　配線中を流れている信号の確認に用いた。

鉛ブロック　放射線を遮蔽するのに用いた。

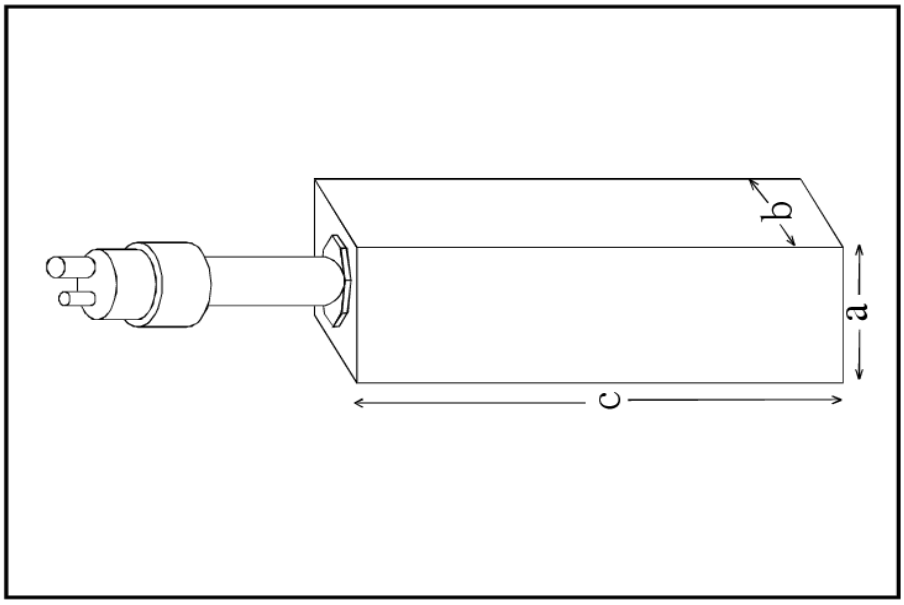

図 3.1. NaI scintillator の模式図

	a [cm]	b [cm]	c [cm]
scintillator 1	5.60	5.55	16.93
scintillator 2	5.60	5.55	16.89
scintillator 3	5.58	5.55	16.94
scintillator 4	5.59	5.55	16.95
scintillator 5	5.55	5.49	16.94

表 3.1. 5 つの NaI scintillator のサイズ. a, b, c などは上の図に示した部分の寸法である.

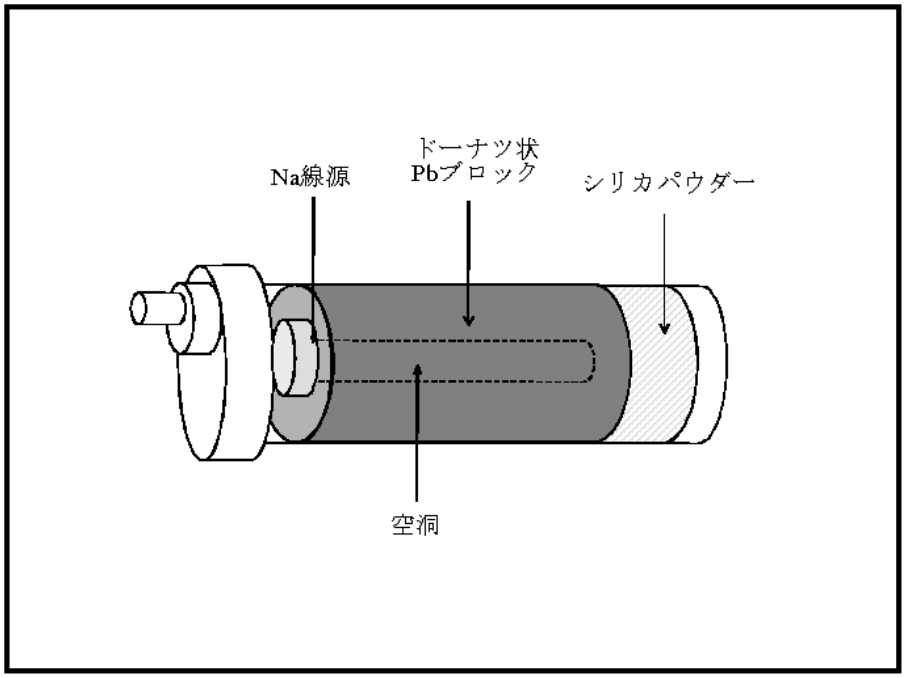

図 3.2. 真空容器とその内部の模式図. 線源 ^{22}Na はテープで鉛ブロックに固定した.

3.1.3 測定の原理

5 つの scintillator のうち任意の 3 つ以上が同時に反応したときのイベントのみを記録する原理について説明する。

5 つの scintillator からの信号を Disc を通してデジタル信号に変換する。この信号を Div を用いて全て足し合わせる。すると Div から出てくる信号は、5 つのうちいくつの scintillator が反応しているかにしたがって、図のように高さが離散的に 5 段階異なるものになっている。この信号を再び Disc に入れ、適当な閾値を課して、ある電圧以上の信号のみを取り出すようにすれば先に述べた目的を達成することができる。同様にこの閾値を変えることによって、任意の個数以上の scintillator が同時に反応している信号のみを取り出すことも可能である。

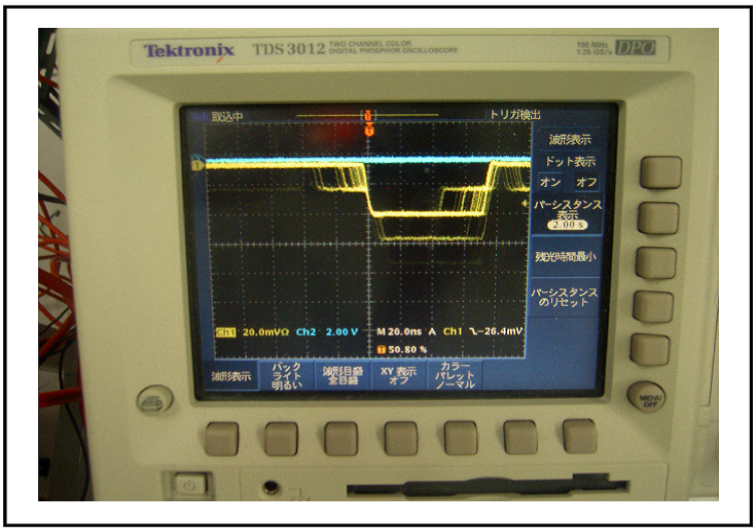

図 3.3.　osciloscope で見た離散的な階段状の信号.

図に本実験での装置の配線図を示す。Div の部分については、5 つの信号を正しく足し合わせるために図のようにした。

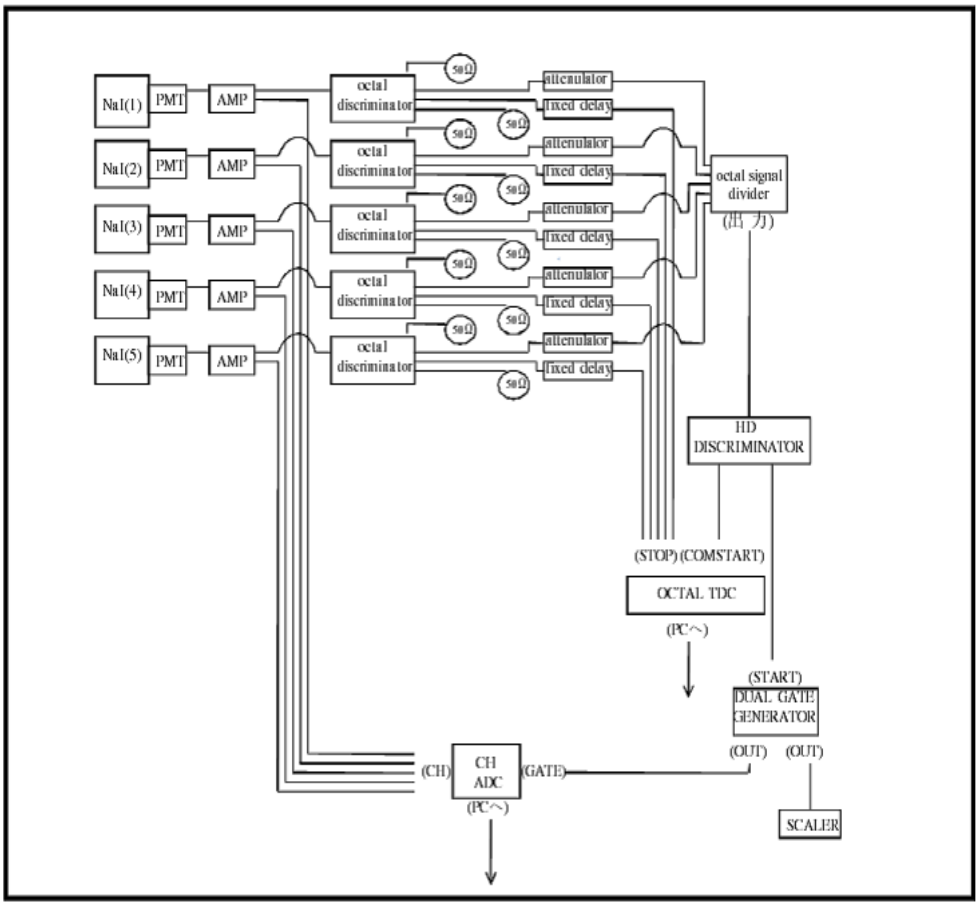

図 3.4.　3γ、2γ 崩壊比測定の回路図. Atten は Div に入る個々の信号を小さくして、うまく次の Disc で閾値を課するために入れた.

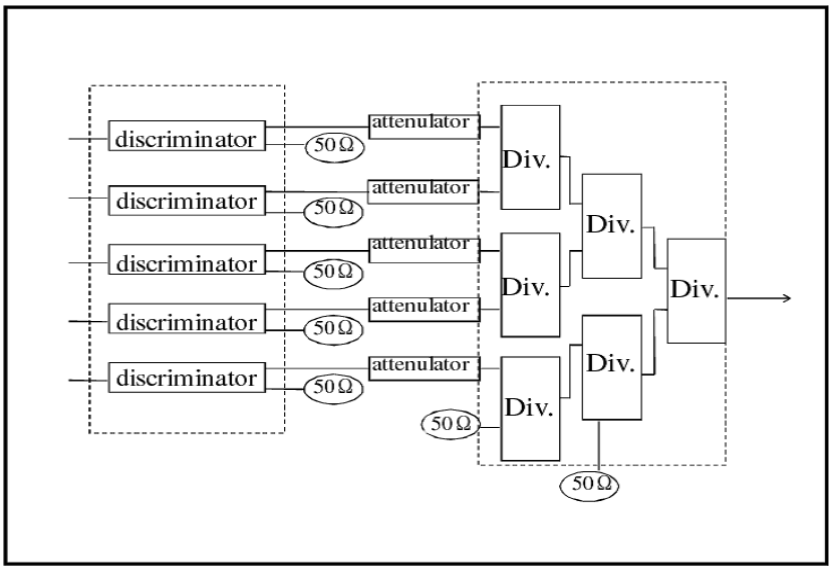

図 3.5.　Div を使い信号を足し合わせるための回路図. 50 Ωの抵抗を使い、
どの信号の扱いも同じになるようにしてうまく足し合わせられるようにしてある。

3.1.4　実験のセットアップ

本実験において、各種装置の値の設定をどのように行ったかを説明する。以下では Disc1 の閾値、デジタル信号の幅を threshould1、width1 などと記す。

HV については、^{22}Na から放射される 0.511 MeV と 1.275 MeV の γ 線のうち、本実験に必要な 0.511 MeV の γ 線が ADC の観測可能領域に入るように設定した。

Disc1 の閾値、threshold1 は scintillator が受け取った γ 線のうち 0.511 MeV のものが観測でき、かつ低エネルギーのその他の放射線があまり観測にかからないようにるように oscilloscope で scintillator からの生の信号を見て決定した。デジタル信号の幅、width1 は複数の scintilator に同時に入った信号を Coin で正しく同時と判定できるように幅を調整した。

Atten は Disc2 の閾値を調整して信号を選び分ける際に、Disc2 の閾値が設定できる範囲に信号の高さがくるように値を設定した。

Disc2 の閾値、threshold2 は 3γ、2γ 崩壊のどちらを観測するかによって、階段状の信号の 2 段目と 3 段目の間、あるいは 1 段目と 2 段目の間の値になるように設定した。width2 に関しては先の width1 と同じように決定した。

Gate の幅は ADC での測定時にエネルギーを測定する時間を決定するものなので、oscilloscope を見ながらデジタル信号のなっている時間帯にアナログ信号が十分含まれるように設定した。

以下に本実験で用いた装置の各値の設定値を示す。なお、HV ch1 などは scintillator 1 の作動用の電源電圧である。また Disc1 の閾値、デジタル信号の幅を threshould1、width1 などと記す。gata は Gata Generator によって広げられた信号の幅である。

HV ch1	-923 V
HV ch2	-859 V
HV ch3	-1030 V
HV ch4	-891 V
HV ch5	-989 V
threshould1	12.3 mV
width1	142 ns
threshould2	39.8 mV (3γ 崩壊測定時)、23.5 mV (2γ 崩壊測定時)
width2	123 ns
gate	500 ns
atten	15 dB

表 3.2　2γ、3γ 崩壊比測定実験における各種装置の設定値.
5 つの Atten の設定値は全て同じにした.

3.2　positronium の寿命の測定

3.2.1　実験概容

本実験ではシリカパウダーに放射線、β^+ が入射した時間と、その β^+ によりできた positronium が 3γ 崩壊をする時間の差を測定することによって、ortho-positronium の寿命を測定する。放射線源としては引き続き ^{22}Na を使用した。γ 線を測定する装置には NaI scintillator を 4 つ用いて、β^+ がシリカパウダーに入射した時間を測定するのには、plastic scintilator と光電子増倍管を用いた。

4 つの scintillator のうちの 2 つ以上の scintillator が反応したイベントのデータを集めた。これは para-positronium の 2γ 崩壊の信号も同時に計測するためである。

3.2.2　実験装置

以下に本実験で用いた実験装置とそれに関する説明を列挙する。ほとんどの装置は 2γ、3γ 崩壊比の測定で使用したものと同じなので、先の実験と異なるものだけを記す。

NaI scintillator　光電子増倍管を挿入するために、幾何学的配置上実験では 4 つを使用した。それぞれのサイズは図に示してある。

plastic scintillator　放射線が入射すると発光する。β^+ が入射した時間を測定するために用いた。

真空容器　図のように真空容器にはドーナツ状の鉛と放射線源、ビニール袋に入れたシリカパウダーを入れた。シリカパウダーに放射線 β^+ が入射した時間を測定するためにドーナツ状の鉛とシリカパウダーの間に plastic scintillator を挟んだ

光電子増倍管　plastic scintillator の反応により発する光を観測するのに用いた。

暗箱　光電子増倍管に入る光が plastic scintillator からのものだけになるように、装置全体を囲うのに用いた。

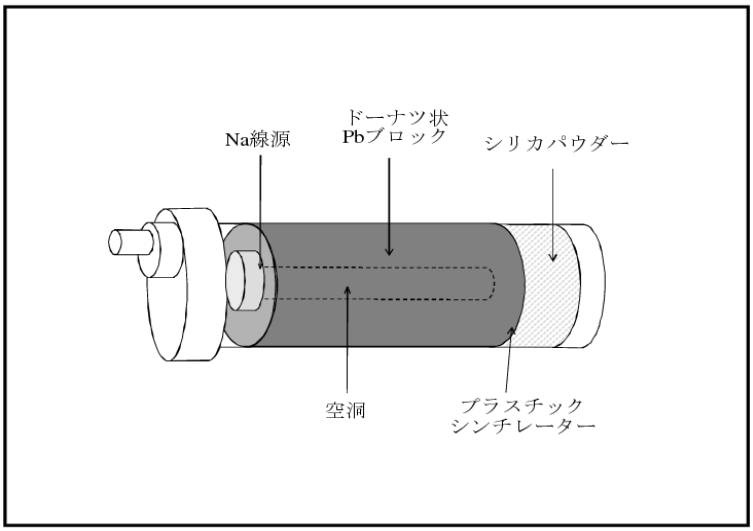

図 3.6.　真空容器の図.　鉛とシリカパウダーの間に、新たに plastic sintillator を挟んだ。

3.2.3　測定の原理

2γ 崩壊は ortho-positronium ではなく para-positronium の崩壊現象であるが、この 2γ 崩壊を測定することによってなぜ ortho-positronium の寿命が決定できるのかを説明する。

先に述べたように ortho-positronium の寿命は para-positronium の寿命よりも長く、その多くが pick off や spin conversion などによって para-positronium に変わってしまい、その para-positronium は ortho-positronium の寿命に比べてすぐに 2γ 崩壊を起こしてしまう。para-positronium の寿命は ortho-positronium の寿命に比べ十分短いので、ortho-positronium から para-positronium に変化して 2γ 崩壊をするまでの時間は、ほぼ ortho-positronium の寿命に等しいと考えられる。よって β^+ が入射してから、2γ 崩壊が起こるまでの時間を測れば、多くの場合は ortho-positronium の寿命を測っていることとなる。

本実験では 4 つの scintillator を用いて、そのうち 2 つ以上が同時に反応したイベントを 2γ 崩壊のイベントの候補として観測した。2 つ以上の scintillator に同時に入った信号を選び出す方法は 4.1.3 節で述べた方法と同じである。

図に本実験での装置の配線図を示す。

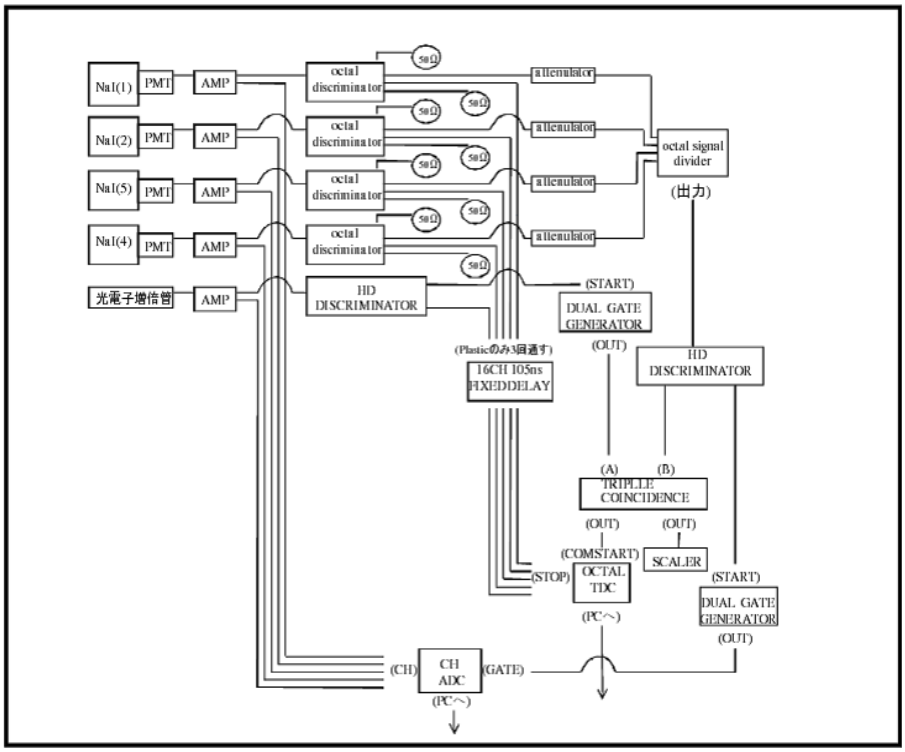

図 3.7　ortho-positronium の寿命測定の回路図

3.2.4　実験のセットアップ

本実験において、各種装置の値の設定をどのように行ったかを説明する。

HV ch1	-914 V
HV ch2	-859 V
HV ch4	-891 V
HV ch5	-989 V
HV plas	-1000 V
threshould1	12.3 mV
width1	142 ns
threshould2	49.9 mV
width2	123 ns
gate	500 ns
atten	15 dB

表 3.3.　orthoo-positronium の寿命測定実験における各種装置の設定値

4 Results

4.1 e+e-→ 2γ と e+e-→ 3γ の比

以下に 4 の図にある略語について説明をする。ただし、4.3 では e1,e2,e3 と e4 は単にシンチレーターの番号を意味している。

e1 二つの信号が入ったときにプログラムで取り出したエネルギーの一個目

e2 二つの信号が入ったときにプログラムで取り出したエネルギーの二個目

e3 二つの信号が入ったときにプログラムで取り出したエネルギーの三個目

t 各チャンネルの到達時間（相対）

Entries イベントの総数

4.1.1 *file*_0913.dat(2γ 以上のイベント) についての結果

2γ のイベントを取り出す前に、軽くカット (400keV< 各エネルギー <600keV) を入れたエネルギーの分布を図 1 に示す。このときのデータ数は、40201 である。なお、本来の数は 100000 である。

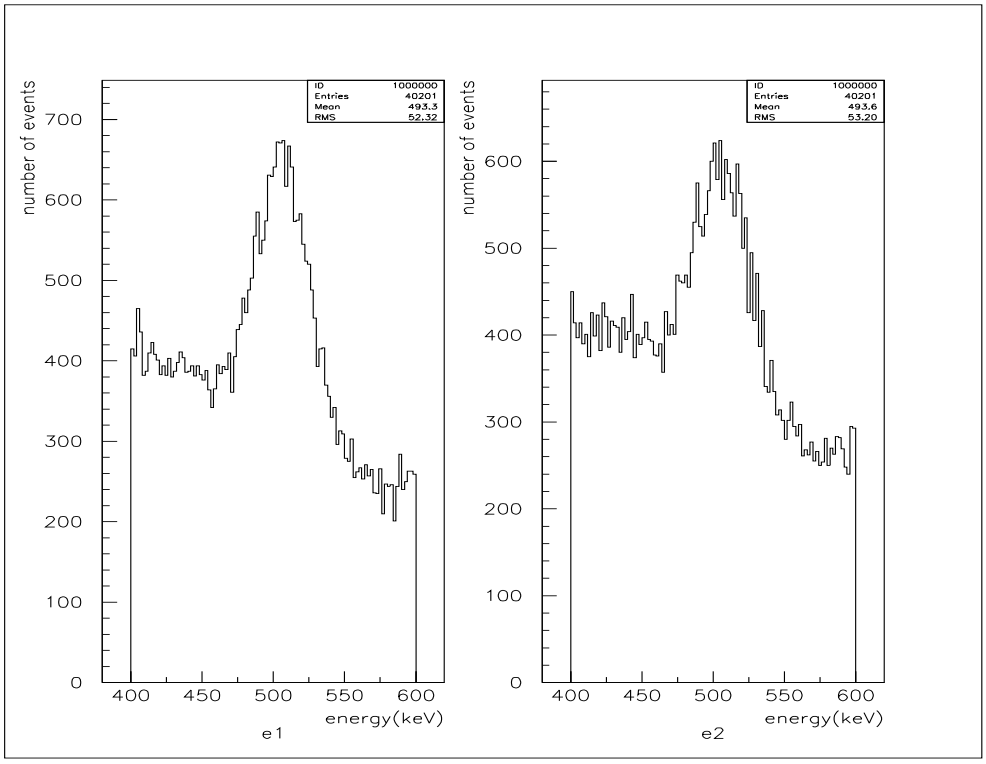

図 4.1：2γ のイベント

2γのイベントを取り出すためのカット(476keV<e1<534keVと473keV<e2<535keV)を入れたエネルギーの分布を図2に示す。このときのデータ数は7524である。総エネルギーが1022keV付近であるかどうかを確認するために、e1+e2のグラフをプロットした。また、同時性を確認するためにt2-t1のグラフもプロットした。

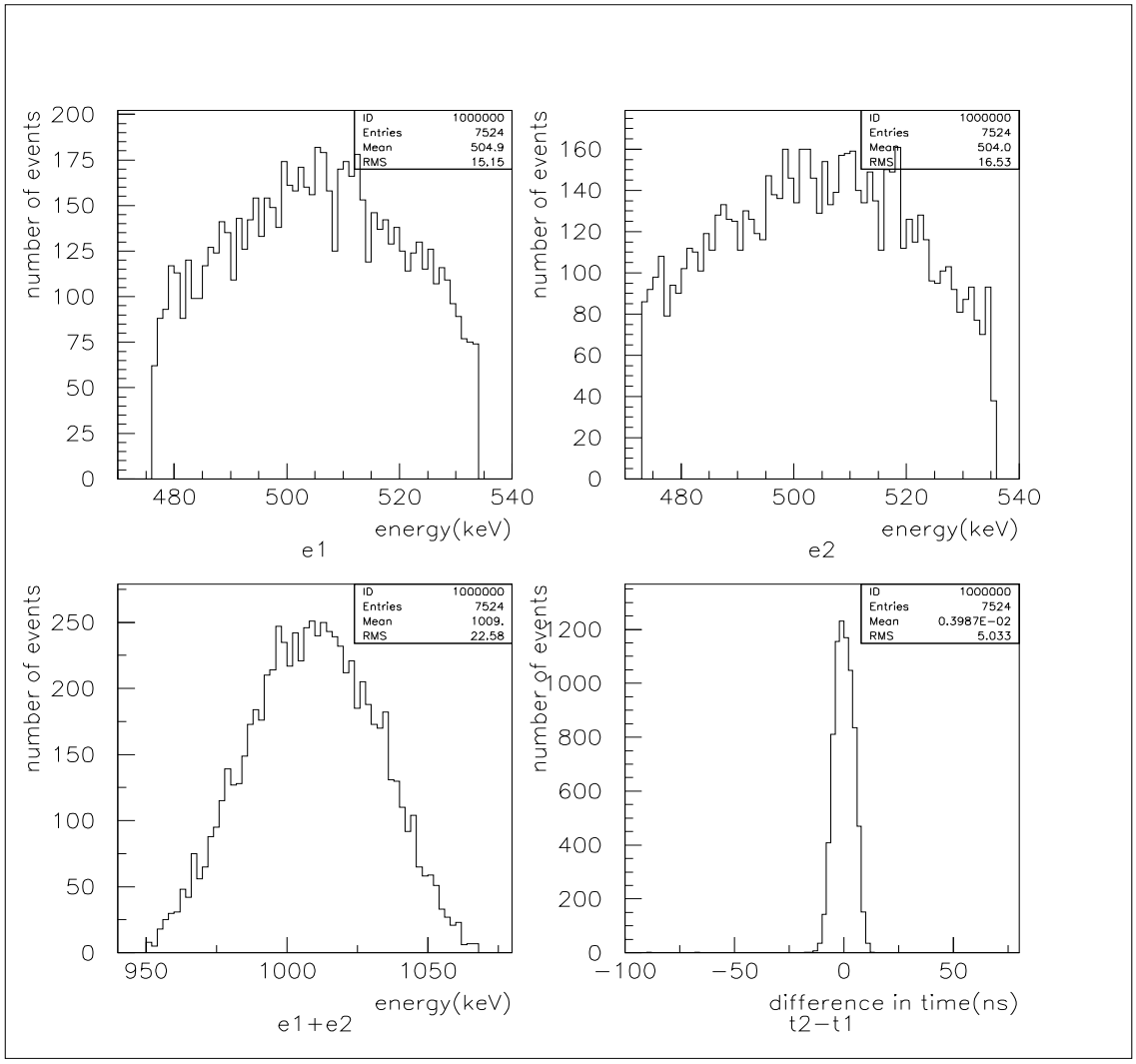

図4.2：2γのイベント(476keV<e1<534keVと473keV<e2<535keVと962keV<総エネルギー<1062keVのカット)

3γ のイベントを取り出す前のエネルギー分布は図 3 に示す。このときのデータ 75847 である。

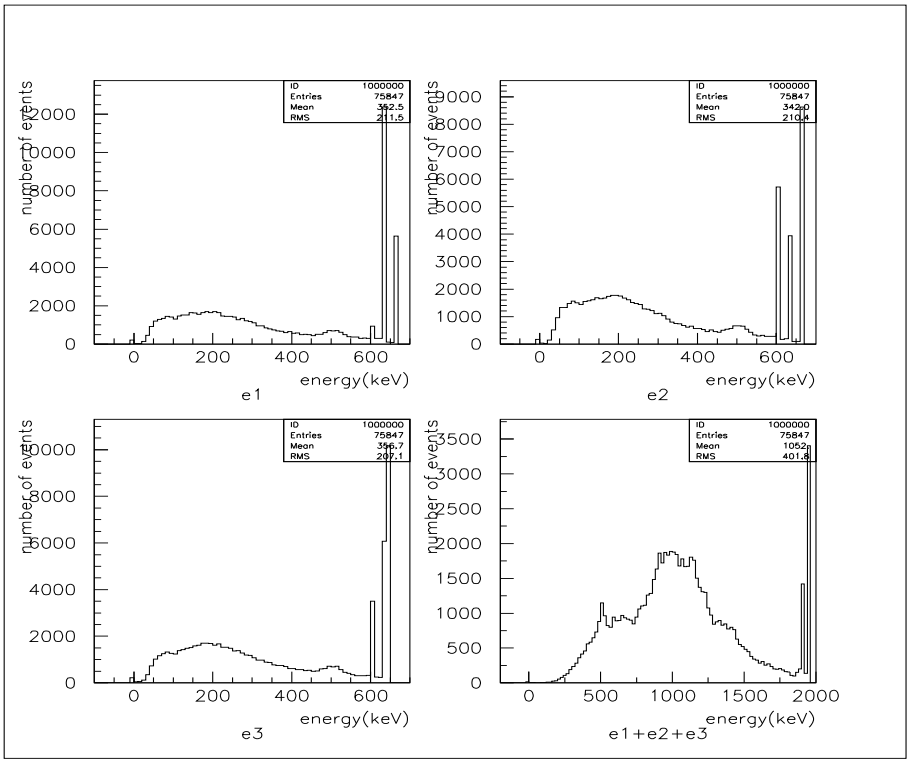

図 4.3：3γ のイベント (ノーカット)

このまま 3γ のイベントについて厳しいカットを入れると、イベント数が急に減ってしまうから、新たに 1,500,000 個の 3γ のイベントを取り、3γ のイベント (ノーカット) と 3γ のイベント (カット) の比を求める。

4.1.2 *file*_0918**.dat.dat(3γ 以上のイベント) についての結果**

新たな 1,500,000 個のイベントに各エネルギー <475keV と ei+ej>500keV のカットを入れると、e1,e2,e3,e1+e2+e3(e0) のグラフは図 4 のようになる。

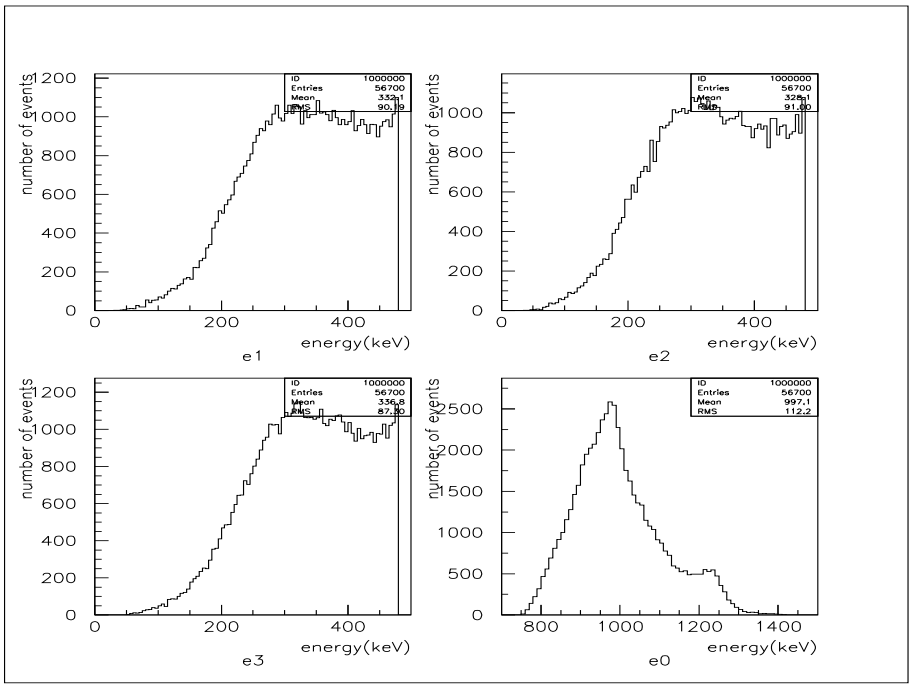

図 4.4：3γ のイベント (各エネルギー <475keV と ei+ej>500keV のカット)

e0 のグラフについて、1022keV 付近の積分をとると、3γ のイベント (カット) 数が得られた。それは 14,230 個である。

4.2　分岐比の計算した結果

2γ 以上の threshold より、1,500,000 イベントのうち、
2γ のイベント数は 7,524 であった。
3γ のイベント数 (ノーカット) は 75,847 であった。

3γ 以上の threshold より、1,500,000 イベントのうち、
3γ のイベント数 (ノーカット) は 1,412,464 であった。
3γ のイベント数 (922keV< 総エネルギー <1122keV のカット) は 14,230 であった。

以上より、BR(e+e-→ 2γ / e+e-→ 3γ)=$\left(\frac{7524}{75847}\right)\left(\frac{1412464}{14230}\right)$=9.85 シミュレーションの結果より、今回のような配置では e+e-→ 2γ の取れる率は 33489/100000 であり、e+e-→ 3γ の取れる率は 7669/100000 であるため、真の BR(e+e-→ 2γ / e+e-→ 3γ)=$\left(\frac{7524}{75847}\right)\left(\frac{1412464}{14230}\right)\left(\frac{7669}{33489}\right)$=2.26 となる。
理論では、真空中に、BR(e+e-→ 2γ / e+e-→ 3γ)=$\frac{1}{3}$ であるが、今回の演習では真空ポンプが得られなかったため、空気中について比を測定した。Ortho-Ps は他の粒子との相互作用でパラ Ps に変換しつつあるため、e+e-→ 2γ の方が多くなった。従って、今回のこのような数値が得られたと考えられる。

4.3　オルソポジトロニウムの寿命

各シンチレーターで得られたデータを図 4.6〜図 4.11 に示す。時間とエネルギーの相関を見るために、図 4.5 をプロットした。threshold をかけると、エネルギーによって、信号が入ってくる時間も違ってくる (図 4.6) ので、図 4.5 に基づいて補正をしなければならない。

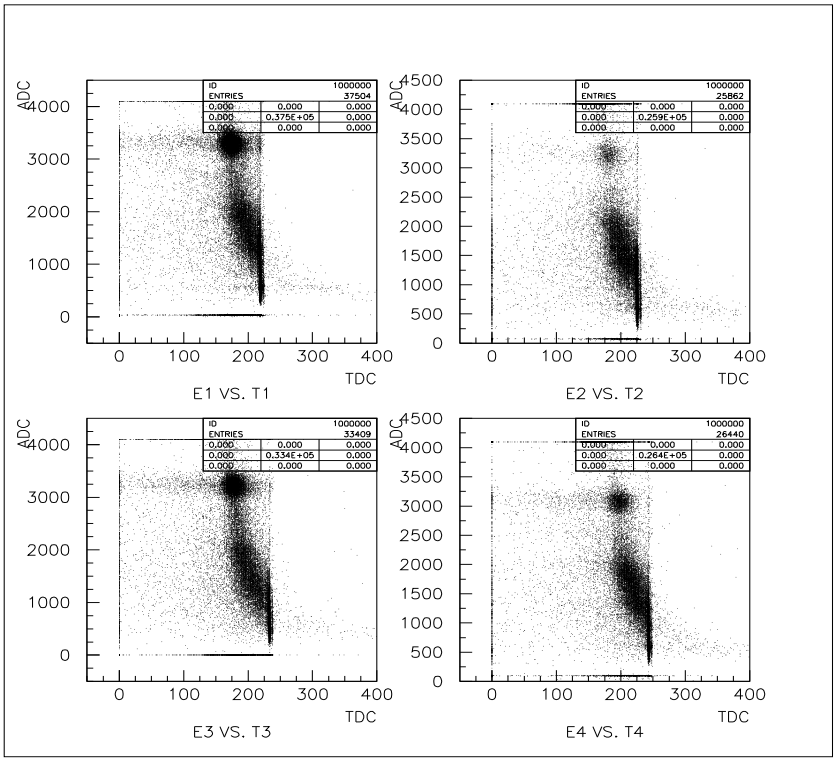

図 4.5：時間とエネルギーの相関 (生データ)

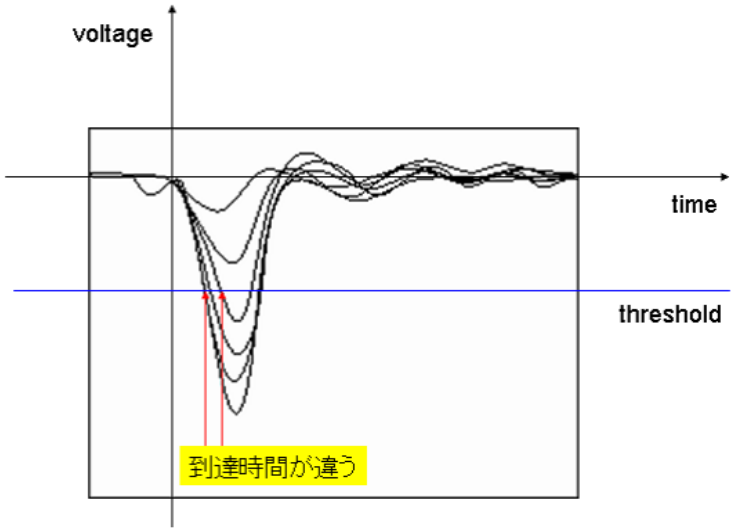

図 4.6：信号の入り方

補正したものは図 4.7 に示してる。補正というのは T を補正した時間、t を元の時間、e をエネルギーとすると、

$$T = t + \sqrt{((e - 152.069)/2.382)} - 12.826 \tag{43}$$

である。補正したものは一直線になり、同時のはずの信号はエネルギーに関係なく、同時に入ることが確認できた。

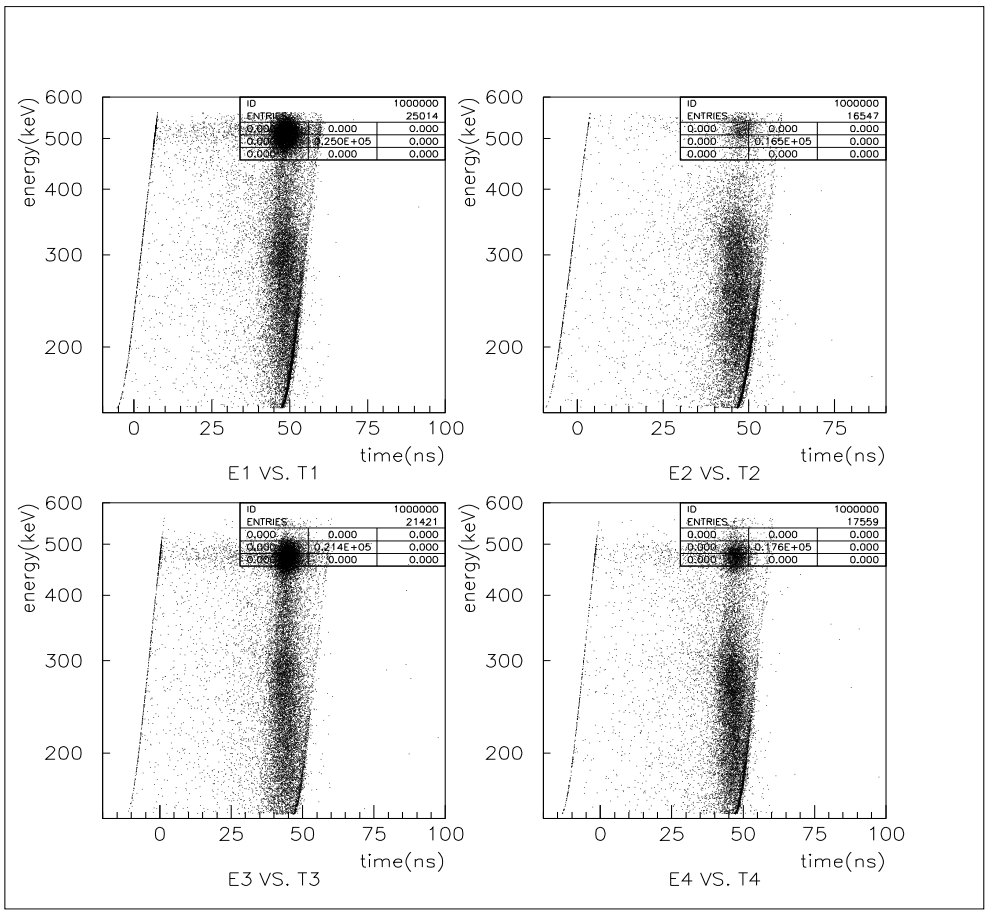

図 4.7：時間とエネルギーの相関（補正したもの

各チャンネルのエネルギー分布は図 4.8 に示されている。カットは入れていないため、511keV 付近のピークはおそらく 2γ のイベントまたはその弱散乱のイベントに由来するものである。寿命を求めるとき、エネルギーについてのカットを 150keV< エネルギー <460keV の範囲にした。

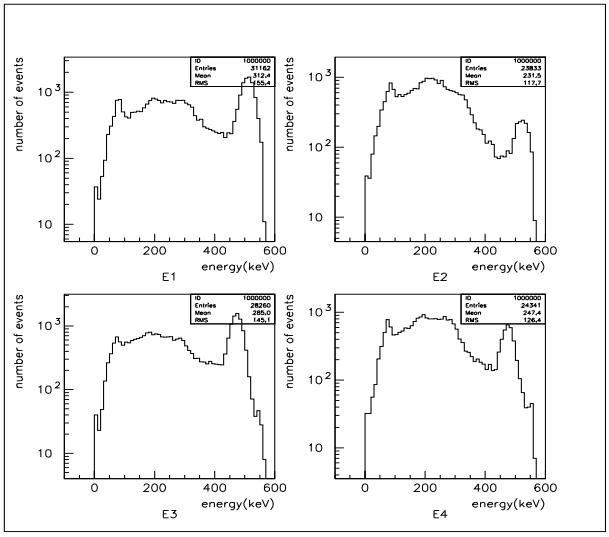

図 4.8：四つのチャンネルのエネルギー分布

各チャンネルの到達時間分布は図 4.9 に示されている。500ns 付近以上のものを overflow と見なし、カットを入れたものは図 4.10 に示す。

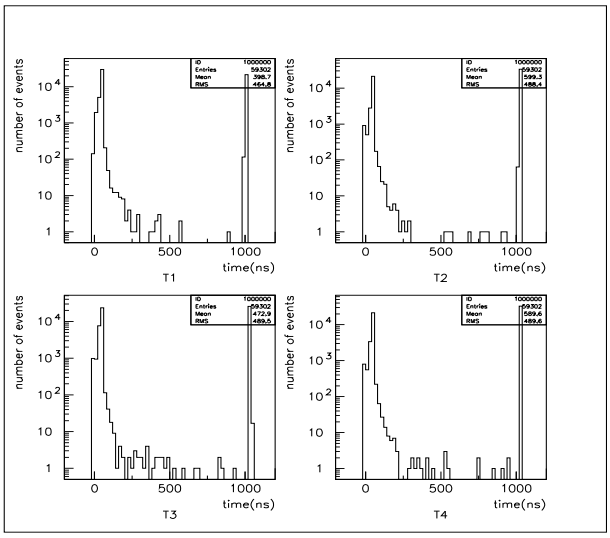

図 4.9：四つのチャンネル到達時間分布

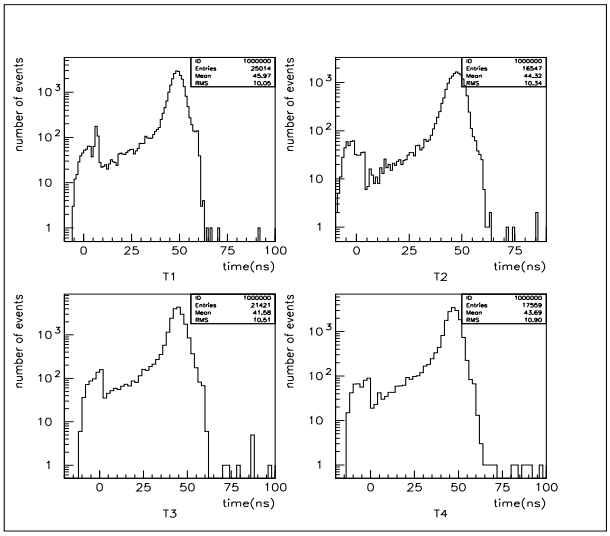

図 4.10：四つのチャンネル到達時間分布（時間カットしたもの

ortho-Ps の崩壊幅を求めるために、各シンチレーターに信号が入った時間 (t1-t4) を Plastic シンチレーターに入った時間 (t5) から引き、縦軸をイベント数とし、横軸を時間としてグラフを描いた。

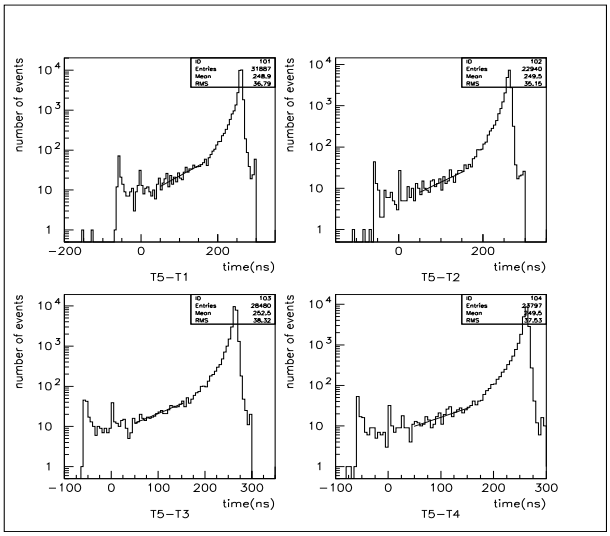

図 4.11：四つのチャンネルのそれぞれの寿命

」ピークのところはシンチレーターの分解能と見なし、100ns 付近で、傾きが一定であるところの Fitting をした。それが崩壊幅であり、逆数が寿命を表している。

4.4　寿命の計算した結果

CH no.	End time (ns)	傾き 1/(0.1ns) (傾き＝崩壊幅)	Start time (ns)			
			25	50	75	100
CH 1	175	崩壊幅 1/(0.1ns)	0.122 ±0.0106	0.116 ±0.0146	0.121 ±0.0263	0.121 ±0.0438
CH 2	175	崩壊幅 1/(0.1ns)	0.123 ±0.0187	0.131 ±0.0484	0.142 ±0.0520	0.172 ±0.0530
CH 3	175	崩壊幅 1/(0.1ns)	0.112 ±0.0110	0.108 ±0.0147	0.114 ±0.0264	0.121 ±0.0464
CH 4	175	崩壊幅 1/(0.1ns)	0.110 ±0.0152	0.110 ±0.0219	0.123 ±0.0511	0.129 ±0.0525

図 4.12：四つのチャンネルのそれぞれの崩壊幅

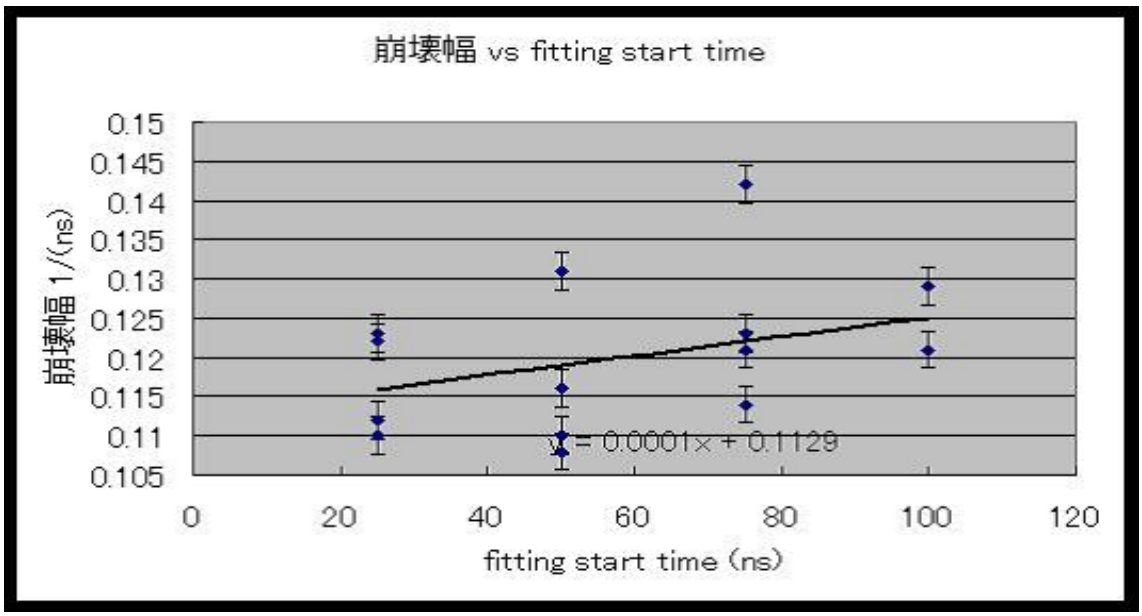

図 4.13：寿命の Fitting

Fitting start time の平均 (62.5ns) のところでは崩壊幅=0.011915/(ns) △ は誤差で △ = 0.000375/(ns)。崩壊幅、λ = [0.011475 ± 0.000375]/(ns) なので、平均寿命、τ = 87.15 ± 0.28(ns) となる。

理論では、真空中に、τ = 140[ns] であるが、空気中では 87[ns] まで下がるのである。これも Ortho-Ps が他の粒子との相互作用でパラ Ps に変換しつつあるため、3γ の数が減少し、寿命が短く見えてくる。従って、今回のこのような数値が得られたと考えられる。

5 Discussion

5.1 空気中の粒子からの効果

Ortho-Ps の数を N_o とすると、真空中では、

$$\frac{dN_{o,v}}{dt} = - \left(\frac{1}{\tau}\right) N_{o,v}$$

という式が成立し、空気中では、

$$\frac{dN_{o,a}}{dt} = - \left(\frac{1}{\tau}\right) N_{o,a} - \lambda N_{o,a}$$

という式が成立する。これらを $N_{o,v}$、$N_{o,a}$ について解くと、

$$N_{o,v} = N_{o,v,initial} \exp\left(-\left(\frac{t}{\tau}\right)\right) \text{ と}$$
$$N_{o,a} = N_{o,a,initial} \exp\left[-\left(\frac{1}{\tau} + \lambda\right)t\right]$$

が得られる。真空中の寿命 (140ns) と結果から出した寿命 (87.15ns) を $\left(\frac{1}{\tau}\right)$ と $\left(\frac{1}{\tau} + \lambda\right)$ に関係付けると、

$$\frac{1}{\tau} = \frac{1}{140}$$
$$\frac{1}{\tau} + \lambda = \frac{1}{87.15}$$

であることがわかる。$\lambda = \frac{1}{87.15} - \frac{1}{140} = 0.0043/[ns]$ が得られる。

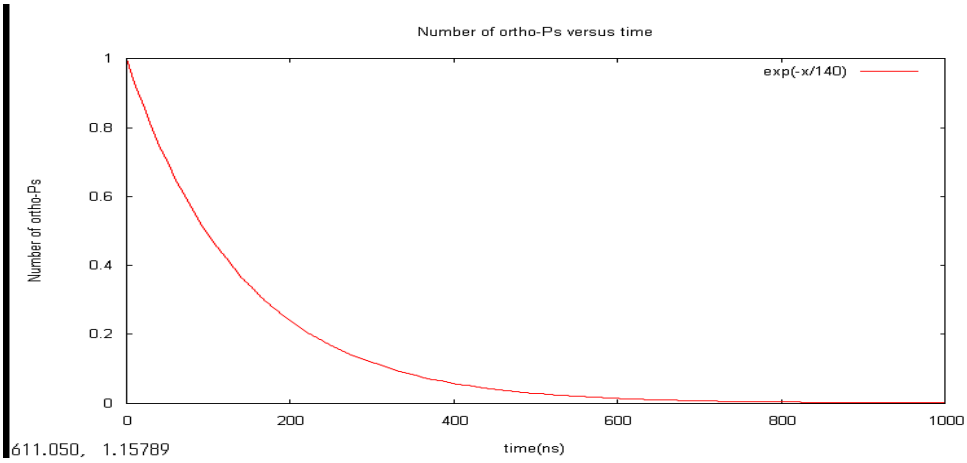

図 5.1：真空中の減衰

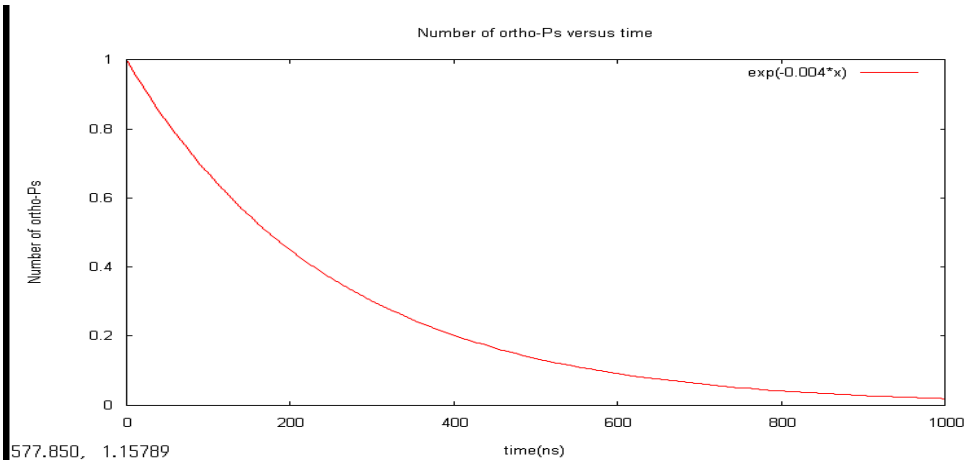

図 5.2：Para-Ps への変換などの減衰

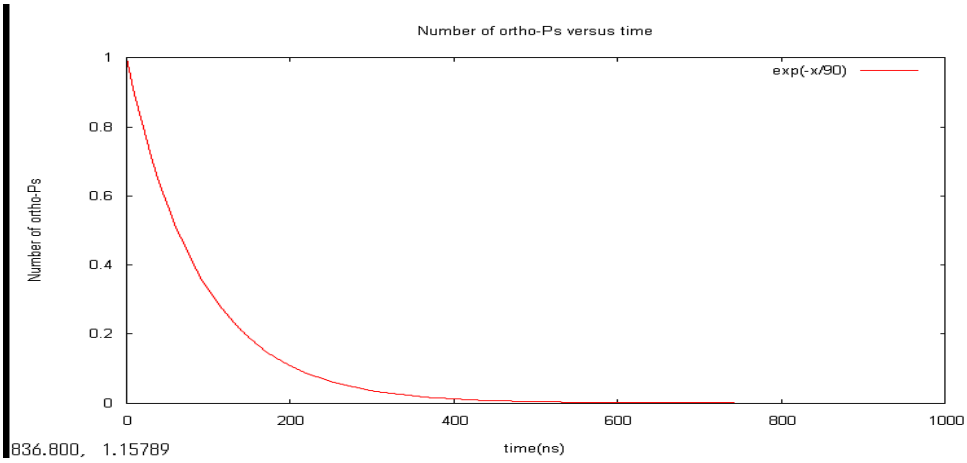

図 5.3：空気中の減衰

6 Conclusion

この課題演習で得られた結果を簡潔にまとめる。
β^+ の線源とした ^{22}Na を使い、シリカパウダーを静止電子源とし、
1. 2γ と 3γ の比, **Ratio(e+e-→ 2γ / e+e-→ 3γ)=2.26** だった。
2. ortho-Ps の寿命を測定した。**平均寿命、 τ = 87.15 ± 0.28(ns)** となった。

7 Appendix

7.1 Notation

$$x^\mu = (x^0, \boldsymbol{x})$$
$$\not{p} = \gamma^\mu p_\mu$$
$$\alpha = \frac{e^2}{4\pi} = \frac{e^2}{4\pi\hbar c} \approx \frac{1}{137}$$

7.2 寿命の求め方

ある放射線物が単位時間内に崩壊する確率は崩壊定数,λ と呼ばれている。崩壊定数 λ を持つ物質が N(t) 個存在した時、その減衰率は N(t) に比例するので、次の式が成立する。

$$\frac{dN(t)}{dt} = -\lambda N(t) \tag{44}$$

この式の解は、

$$N(t) = N(0)\exp(-\lambda t) \tag{45}$$

となる。ここで、$N(0)$ は時刻 t=0 における物質の個数である。放射線物の平均寿命は個数が $\frac{1}{e}$ になるまでの時間なので、以上の二つの式より、平均寿命、τ は

$$\tau = \frac{1}{\lambda} \tag{46}$$

となる。式 (2) の両辺に log をとると、

$$logN(t) = -\lambda t + logN(0) \tag{47}$$

となる。このことより、崩壊する粒子数とその時間をプロットし、粒子数を対数で表示すると、そのときの傾きの逆数が平均寿命になる。

8 Simulation

8.1 シミュレーションの概要

本実験の 5 つの scintillator の配置では、たとえ同じ数の 2γ 崩壊と 3γ 崩壊が起きたとしても、検出できる数が異なってくる。これを補正するためにコンピュータシミュレーションを用いた。このシミュレーションについて説明する。

シミュレーションではまず空間にデカルト座標を設定し、positronium がこの座標系の原点で崩壊するとした。次に放出される γ 線の 4 元運動量ベクトルを決定する。これには 4 元運動量の保存から得られる式を用い、空間成分の運動方向の決定のためには乱数を用いた。乱数の生成には Mersenu Twister 法を用いた。

次にこれにより得られた各 γ 線の軌道が各 scintillator にヒットするかを判定する。ヒットすればそこで各 γ 線のエネルギーに応じて減衰係数が決定できるのでこれを用いて NaI 中の透過距離を求めた。この透過距離の分だけ進行した後も依然として NaI 中に γ 線が含まれていれば、NaI 中で何らかの反応を起こしたものとみなした。どの反応が起きたかについては、各反応（Compton 効果、光電吸収、Rayleigh 散乱）の断面積より発生確率が決定できるので乱数を用いて決定した。光電吸収が起きたなら γ 線はずべてのエネルギーを scintillator に落とし、Compton 散乱なら、さらに散乱角を乱数を用いて決定しそれから scinrillator に落とすエネルギーを決定した。Rayleigh 散乱ではエネルギーは落とさないとした。なお各断面積などの値は参考文献 [2] を参照した。

このように各 γ 線に対して、ヒットした scintillator と落としたエネルギーが決定できる。これから本実験の測定方法で検出できるという条件を課す。つまり、3γ 崩壊ならヒットした scintillator の数は 3 つであるか（2γ 崩壊なら 2 つであるか）という条件と、一つめの Discriminator でかけている閾値に対応したエネルギー以上を、γ 線が各 scintillator で落としているかという条件である。

このようにすると、初めに発生させた 3γ（あるいは 2γ）崩壊のイベント数に対して、本実験のセットアップで検出されるイベント数の比を求めることが出来る。実験ではこの比の値を用いて解析を行った。

8.2 プログラム

以下に 2γ、3γ 崩壊の検出率を求めるのに用いたプログラムを示す。

8.2.1 2γ 崩壊

```
/*************************************************************************
 * 2007 年度前期 A1 最終実験シミュレーションプログラム 2
*************************************************************************
```

```
 *
 * オルソポジトロニウムの 2 ガンマ崩壊のイベントを、
 * 実験のセットアップで全イベントのうちどれぐらいを
 * 観測できているかをシミュレーションする。
 *
*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "mt19937ar.h"

#define E_elec 5.11e-1 /* 電子の静止エネルギー (MeV) */
#define NaI_density 3.67 /* NaI の密度 (g/立方 cm) */
#define N 100000 /* 3 ガンマ崩壊の全イベント数 */
#define M 128 /* 1.275(MeV) までのカラムの数 */
#define dX 1.0e-2 /* カラムの幅 */

#define dL 1.0e-1
#define R_LIM 2.0e1

#define CUT   2.0e-1 /* Discriminator の閾値によるカット */

/*
#define DEBUG /**/
/*
#define DEBUG1/**/
/*
#define DEBUG2/**/
/*
#define DEBUG3/**/
/*
#define DEBUG4/**/

double phot_ab(double E); /* NaI シンチレーターの光電吸収による質量減衰
係数 */
double comp_scat(double E); /* NaI シンチレーターのコンプトン効果による
質量減衰係数 */
```

```
double Rayleigh_scat(double E); /* Rayleigh散乱による質量減衰係数 */
double resolve(double E); /* シンチレーターのエネルギー分解能 */

void scintilation(double *x0, double *p, double E,
 double CE, double *drop_E);
void histgram(double x, int *num); /* シンチレーターに入った光子のエネ
ルギー分布を作る函数 */

double L[3], L1[3], L2[3], L3[3], L4[3], L5[3];

int main(void)
{
FILE *fp1, *fp2, *fp3, *fp4, *fp5;
double phi, ran;
double p1[3], p2[3], gamma1[3], gamma2[3];
double p[2];
double E1, E2, drop_E1[5], drop_E2[5];
double e[2], drop_E[5];
double t, x, y, z;
double op, mu, tau, E, rata, bra, sangle;
double sigma1, sigma2, sigma;
int scint[5], sum, count[5], count1[5], count2[5];
int hist[5][M], e1hist[M], e2hist[M], e12hist[M], k;
unsigned int experi, num;
register unsigned int i, j;

/*********************************************************************
* 書き込みファイルのオープン
*********************************************************************/

if((fp1=fopen("07_a1_1.txt", "w"))==NULL) {
printf("file open error!!\n");
exit(1);
}

if((fp2=fopen("07_a1_2.txt", "w"))==NULL) {
printf("file open error!!\n");
exit(1);
}
```

```
if((fp4=fopen("07_a1_4.txt", "w"))==NULL) {
printf("file open error!!\n");
exit(1);
}

if((fp5=fopen("07_a1_5.txt", "w"))==NULL) {
printf("file open error!!\n");
exit(1);
}

/*********************************************************************
* 書き込み内容の書き込み
*********************************************************************/

fprintf(fp1, "# energy1(Mev) energy2(MeV) energy3(MeV) energy4(NeV)
         energy5(MeV) scinti1 scinti2 scinti3 scinti4 scinti5\n");

/*********************************************************************
* Mersenne Twister（乱数生成器）の初期化
*********************************************************************/

int utime;
long ltime;

ltime = time(NULL);
utime = (unsigned int) ltime/2;
srand(utime);

unsigned long init[4]={rand(), rand(), rand(), rand()}, length=4;
init_by_array(init, length);

/*********************************************************************
* 各シンチレーターの配置に関するデータの設定
*********************************************************************/

/********************* シンチレーターの中心 ************************/

L[0] = 3.25e0;
```

```
L[1] = 2.44e0;
L[2] = 3.25e0;

/******************** シンチレーター1の寸法 ***********************/

L1[0] = 5.6e0;
L1[1] = 1.693e1;
L1[2] = 5.55e0;

/******************** シンチレーター2の寸法 ***********************/

L2[0] = 5.6e0;
L2[1] = 1.689e1;
L2[2] = 5.55e0;

/******************** シンチレーター3の寸法 ***********************/

L3[0] = 5.58e0;
L3[1] = 1.694e1;
L3[2] = 5.55e0;

/******************** シンチレーター4の寸法 ***********************/

L4[0] = 5.59e0;
L4[1] = 1.695e1;
L4[2] = 5.55e0;

/******************** シンチレーター5の寸法 ***********************/

L5[0] = 5.55e0;
L5[1] = 1.694e1;
L5[2] = 5.49e0;

/*************************************************************************
* N事象のうち、測定にかかる回数の初期化
*************************************************************************/

experi = 0;
for(i=0; i<5; i++) {
```

```
scint[i] = 0;
for(j=0; j<M; j++) {
hist[i][j] = 0;
}
}
for(i=0; i<M; i++) {
e1hist[i] = 0;
e2hist[i] = 0;
e12hist[i] = 0;
}

/*****************************************************************
* N 事象のシミュレーションの開始
*****************************************************************/

for(i=0; i<N; i++) {

/*****************************************************************
* ガンマ線の発生位置の決定
*****************************************************************/

gamma1[0] = 0.0e0;
gamma1[1] = 0.0e0;
gamma1[2] = 0.0e0;

gamma2[0] = 0.0e0;
gamma2[1] = 0.0e0;
gamma2[2] = 0.0e0;

/*****************************************************************
* 2 本のガンマ線の運動量ベクトルの決定
*****************************************************************/

phi = 2.0e0 * M_PI * genrand_real2();
ran = 2.0e0 * genrand_real1() - 1.0e0;

E1 = E_elec;
E2 = E_elec;
```

```
p1[0] = E1 * sqrt(1.0e0-ran*ran) * cos(phi);
p1[1] = E1 * sqrt(1.0e0-ran*ran) * sin(phi);
p1[2] = E1 * ran;

p2[0] = - p1[0];
p2[1] = - p1[1];
p2[2] = - p1[2];

p[0] = sqrt(pow(p1[0], 2.0e0) + pow(p1[1], 2.0e0) + pow(p1[2], 2.0e0));
p[1] = sqrt(pow(p2[0], 2.0e0) + pow(p2[1], 2.0e0) + pow(p2[2], 2.0e0));

/*********************************************************************
* 各シンチレーターに対する当たり判定
*********************************************************************/


/*********** 各シンチレーターのガンマ線検出数の初期化 **************/

for(j=0; j<5; j++) {
count1[j] = 0;
count2[j] = 0;
count[j] = 0;
}


/*********** シンチレーターに落とすガンマ線エネルギーの初期化 **************/

for(j=0; j<5; j++) {
drop_E1[j] = 0.0e0;
drop_E2[j] = 0.0e0;
}

/***************************** ガンマ線1 ***************************/

scintilation(gamma1, p1, E1, 0.0e0, drop_E1);

/***************************** ガンマ線2 ***************************/

scintilation(gamma2, p2, E2, 0.0e0, drop_E2);
```

```
/*********************************************************************
* N事象のうち、測定にかかる回数と各シンチレーターのカウント
*********************************************************************/

for(j=0; j<5; j++) {
drop_E[j] = drop_E1[j] + drop_E2[j];

/************** Discriminatorの閾値によるカット ****************/

if(drop_E[j]<CUT) {
drop_E[j] = 0.0e0;
}

/*********************************************************************/

histgram(drop_E[j], hist[j]);
if(drop_E[j]>0.0e0) {
count[j]++;
}
}
sum = count[0] + count[1] + count[2] + count[3] + count[4];

if(sum==2) {
k = 0;
for(j=0; j<5; j++) {
scint[j] = scint[j] + count[j];
if(count[j]) {
e[k] = drop_E[j];
k++;
}
}
experi++;
histgram(e[0], e1hist);
histgram(e[1], e2hist);
histgram(e[0]+e[1], e12hist);

sigma1 = resolve(E_elec) * (E_elec) / sqrt(2.0e0*log(2.0e0));
sigma2 = resolve(E_elec) * (E_elec) / sqrt(2.0e0*log(2.0e0));
```

```
sigma = resolve(2.0e0*E_elec) * (2.0e0*E_elec) / sqrt(2.0e0*log(2.0e0));
if(e[0]+e[1]>2.0e0*E_elec-sigma && e[0]+e[1]<2.0e0*E_elec+sigma) {
if(e[0]>E_elec-sigma1 && e[0]<E_elec+sigma1) {
if(e[1]>E_elec-sigma2 && e[1]<E_elec+sigma2) {
num++;
}
}
}
}

/*************************************************************************
* データのファイルへの書き出し
*************************************************************************/

fprintf(fp1, "%e %e %e %e %e %d %d %d %d %d\n", drop_E[0], drop_E[1],
                drop_E[2], drop_E[3], drop_E[4], count[0],
                 count[1], count[2], count[3], count[4]);

}

/*************************************************************************
* データのファイルへの書き出し
*************************************************************************/

fprintf(fp2, "# イベント数 観測数 1sigma の個数 (cut あり)
         scinti1 scinti2 scinti3 scinti4 scinti5\n");
fprintf(fp2, "%d %d %d %d %d %d %d %d", N, experi,
         num, scint[0], scint[1], scint[2], scint[3], scint[4]);

for(i=0; i<5; i++) {
fprintf(fp4, "# scintillator %d\n", i+1);
for(j=0; j<M; j++) {
E = dX / 2.0e0 + dX * ((double)j);
fprintf(fp4, "%e %d\n", E, hist[i][j]);
}
fprintf(fp4, "\n\n");
}

fprintf(fp5, "# E1\n");
```

```
for(i=0; i<M; i++) {
E = dX / 2.0e0 + dX * ((double)i);
fprintf(fp5, "%e %d\n", E, e1hist[i]);
}
fprintf(fp5, "\n\n");
fprintf(fp5, "# E2\n");
for(i=0; i<M; i++) {
E = dX / 2.0e0 + dX * ((double)i);
fprintf(fp5, "%e %d\n", E, e2hist[i]);
}
fprintf(fp5, "\n\n");
fprintf(fp5, "# E1+E2\n");
for(i=0; i<M; i++) {
E = dX / 2.0e0 + dX * ((double)i);
fprintf(fp5, "%e %d\n", E, e12hist[i]);
}

/*************************************************************************
* 書き込みファイルのクローズ
*************************************************************************/

if(fclose(fp1)==EOF) {
printf("file close error!!\n");
exit(1);
}

if(fclose(fp2)==EOF) {
printf("file close error!!\n");
exit(1);
}

if(fclose(fp4)==EOF) {
printf("file close error!!\n");
exit(1);
}

if(fclose(fp5)==EOF) {
printf("file close error!!\n");
exit(1);
```

```
}

return 0;
}


/*********************************************************************
* シンチレーターの当たり判定をする函数
*********************************************************************/

void scintilation(double *x0, double *p,
 double E, double CE, double *drop_E)
{
double x[3], y[3], phi,z, Comp_E, d;
double sangle, new_p[3], temp[3], R[3], new_E, r;
double mu, tau, pht_rata, cmp_rata, bra;
double Res, s;
double x1, x2, r1, r2, op;
int first, scint;
int i;

/*********************************************************************
* シンチレーターについての当たり判定
*********************************************************************

first = 1;
Comp_E = CE;
scint = 0;

x[0] = (*(x0));
x[1] = (*(x0+1));
x[2] = (*(x0+2));

d = sqrt(x[0]*x[0] + x[1]*x[1] + x[2]*x[2]);
r = sqrt(pow((*(p)), 2.0e0) + pow((*(p+1)), 2.0e0)
         + pow((*(p+2)), 2.0e0));


while(d<R_LIM) {
```

```
#ifdef DEBUG
if((*(p))>0.0e0) {
printf("%e %e %e\n", x[0], x[1], x[2]);
}
#endif

/************************ シンチレーター1 ************************/

if(scint-1) {
if(x[0]<L1[0]/2.0e0 && x[0]>-L1[0]/2.0e0) {
if(x[1]<L1[1]/2.0e0 && x[1]>-L1[1]/2.0e0) {
if(x[2]<-L[2] && x[2]>-L[2]-L1[2]) {

/******************** NaI中の透過距離の決定 ************************/

mu = (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)) * NaI_density; /* NaI
の線減衰係数の決定 */
pht_rata = phot_ab(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* 光
電吸収が起きる割合 */
cmp_rata = comp_scat(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* コ
ンプトン散乱が起きる割合 */
tau = - log(1.0e0 - genrand_real2()) / mu; /* 透過距離の決定 */

y[0] = x[0] + tau * (*(p)) / r;
y[1] = x[1] + tau * (*(p+1)) / r;
y[2] = x[2] + tau * (*(p+2)) / r;

#ifdef DEBUG
printf("%e %e %e\n", y[0], y[1], y[2]);
#endif

if(y[0]<L1[0]/2.0e0 && y[0]>-L1[0]/2.0e0) {
if(y[1]<L1[1]/2.0e0 && y[1]>-L1[1]/2.0e0) {
if(y[2]<-L[2] && y[2]>-L[2]-L1[2]) {

bra = genrand_real2(); /* 光電吸収、Compton効果、Rayleigh散乱のうちど
れが起こるかの選択 */
```

```
if(bra<pht_rata) {
(*(drop_E)) = (*(drop_E)) + E + Comp_E;
Res = resolve((*(drop_E)));
s = Res * (*(drop_E)) / sqrt(2.0e0*log(2.0e0));
r1 = -genrand_real2() + 1.0e0;
r2 = -genrand_real2() + 1.0e0;
x1 = s * sqrt(-2.0e0*log(r1)) * cos(2.0e0*M_PI*r2);
x2 = s * sqrt(-2.0e0*log(r1)) * sin(2.0e0*M_PI*r2);

op = genrand_real2();
if(op<5.0e-1) {
(*(drop_E)) = (*(drop_E)) + x1;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E)), x1, Res, 1);
#endif

}
else {
(*(drop_E)) = (*(drop_E)) + x2;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E)), x2, Res, 1);
#endif

}
}
else {
phi = 2.0e0 * M_PI * genrand_real2();
z = 2.0e0 * genrand_real1() - 1.0e0;

new_p[0] = sqrt(1.0e0-z*z) * cos(phi);
new_p[1] = sqrt(1.0e0-z*z) * sin(phi);
new_p[2] = z;

#ifdef DEBUG3
printf("%e %e %e\n", new_p[0], new_p[1], new_p[2]);
#endif
```

```
if(bra<pht_rata+cmp_rata) {
temp[0] = (*(p)) - new_p[0];
temp[1] = (*(p+1)) - new_p[1];
temp[2] = (*(p+2)) - new_p[2];

R[0] = sqrt(pow((*(p)), 2.0e0) + pow((*(p+1)), 2.0e0) + pow((*(p+2)), 2.0e0));
R[1] = sqrt(pow(new_p[0], 2.0e0) + pow(new_p[1], 2.0e0) + pow(new_p[2], 2.0e0));
R[2] = sqrt(pow(temp[0], 2.0e0) + pow(temp[1], 2.0e0) + pow(temp[2], 2.0e0));

sangle = acos((R[0] * R[0] + R[1] * R[1] - R[2] * R[2]) / (2.0e0 * R[1] * R[0]));

#ifdef DEBUG1
int i;
for(i=0;i<36; i++) {
if(sangle<M_PI/3.6e1*((double)i + 1.0e0) && sangle>M_PI/3.6e1*((double)i)) {
Ahist[i]++;
}
}
#endif

new_E = 1.0e0 / ((1.0e0 - cos(sangle)) / E_elec + 1.0e0 / E);
Comp_E = Comp_E + (E - new_E);

scintilation(y, new_p, new_E, Comp_E, drop_E);
}
else {
scintilation(y, new_p, E, Comp_E, drop_E);
}
}

break;

}
}
}

scint = 1;

}
```

```
}
}
}

/************************ シンチレーター2 ************************/

if(scint-2) {
if(x[0]<L2[0]/2.0e0 && x[0]>-L2[0]/2.0e0) {
if(x[1]>L[1] && x[1]<L[1]+L2[1]) {
if(x[2]<L2[2]/2.0e0 && x[2]>-L2[2]/2.0e0) {

/******************** NaI中の透過距離の決定 ************************/

mu = (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)) * NaI_density; /* NaI
の線減衰係数の決定 */
pht_rata = phot_ab(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* 光
電吸収が起きる割合 */
cmp_rata = comp_scat(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* コ
ンプトン散乱が起きる割合 */
tau = - log(1.0e0 - genrand_real2()) / mu; /* 透過距離の決定 */

y[0] = x[0] + tau * (*(p)) / r;
y[1] = x[1] + tau * (*(p+1)) / r;
y[2] = x[2] + tau * (*(p+2)) / r;

#ifdef DEBUG
printf("%e %e %e\n", y[0], y[1], y[2]);
#endif

if(y[0]<L2[0]/2.0e0 && y[0]>-L2[0]/2.0e0) {
if(y[1]>L[1] && y[1]<L[1]+L2[1]) {
if(y[2]<L2[2]/2.0e0 && y[2]>-L2[2]/2.0e0) {

bra = genrand_real2(); /* 光電吸収、Compton効果、Rayleigh散乱のうちど
れが起こるかの選択 */

if(bra<pht_rata) {
(*(drop_E+1)) = (*(drop_E+1)) + E + Comp_E;
Res = resolve((*(drop_E+1)));
```

```
s = Res * (*(drop_E+1)) / sqrt(2.0e0*log(2.0e0));
r1 = -genrand_real2() + 1.0e0;
r2 = -genrand_real2() + 1.0e0;
x1 = s * sqrt(-2.0e0*log(r1)) * cos(2.0e0*M_PI*r2);
x2 = s * sqrt(-2.0e0*log(r1)) * sin(2.0e0*M_PI*r2);

op = genrand_real2();
if(op<5.0e-1) {
(*(drop_E+1)) = (*(drop_E+1)) + x1;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+1)), x1, Res, 2);
#endif

}
else {
(*(drop_E+1)) = (*(drop_E+1)) + x2;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+1)), x2, Res, 2);
#endif

}
}
else {
phi = 2.0e0 * M_PI * genrand_real2();
z = 2.0e0 * genrand_real1() - 1.0e0;

new_p[0] = sqrt(1.0e0-z*z) * cos(phi);
new_p[1] = sqrt(1.0e0-z*z) * sin(phi);
new_p[2] = z;

#ifdef DEBUG3
printf("%e %e %e\n", new_p[0], new_p[1], new_p[2]);
#endif

if(bra<pht_rata+cmp_rata) {
temp[0] = (*(p)) - new_p[0];
temp[1] = (*(p+1)) - new_p[1];
```

```
temp[2] = (*(p+2)) - new_p[2];

R[0] = sqrt(pow((*(p)), 2.0e0) + pow((*(p+1)), 2.0e0) + pow((*(p+2)), 2.0e0));
R[1] = sqrt(pow(new_p[0], 2.0e0) + pow(new_p[1], 2.0e0) + pow(new_p[2], 2.0e0));
R[2] = sqrt(pow(temp[0], 2.0e0) + pow(temp[1], 2.0e0) + pow(temp[2], 2.0e0));

sangle = acos((R[0] * R[0] + R[1] * R[1] - R[2] * R[2]) / (2.0e0 * R[1] * R[0]));

#ifdef DEBUG1
int i;
for(i=0;i<36; i++) {
if(sangle<M_PI/3.6e1*((double)i + 1.0e0) && sangle>M_PI/3.6e1*((double)i)) {
Ahist[i]++;
}
}
#endif

new_E = 1.0e0 / ((1.0e0 - cos(sangle)) / E_elec + 1.0e0 / E);
Comp_E = Comp_E + (E - new_E);

scintilation(y, new_p, new_E, Comp_E, drop_E);
}
else {
scintilation(y, new_p, E, Comp_E, drop_E);
}
}

break;

}
}
}

scint = 2;

}
}
}
}
```

```
/************************ シンチレーター3 ************************/

if(scint-3) {
if(x[0]>L[0] && x[0]<L[0]+L3[0]) {
if(x[1]<L3[1]/2.0e0 && x[1]>-L3[1]/2.0e0) {
if(x[2]<L3[2]/2.0e0 && x[2]>-L3[2]/2.0e0) {

/******************** NaI中の透過距離の決定 ************************/

mu = (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)) * NaI_density; /* NaI
の線減衰係数の決定 */
pht_rata = phot_ab(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* 光
電吸収が起きる割合 */
cmp_rata = comp_scat(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* コ
ンプトン散乱が起きる割合 */
tau = - log(1.0e0 - genrand_real2()) / mu; /* 透過距離の決定 */

y[0] = x[0] + tau * (*(p)) / r;
y[1] = x[1] + tau * (*(p+1)) / r;
y[2] = x[2] + tau * (*(p+2)) / r;

#ifdef DEBUG
printf("%e %e %e\n", y[0], y[1], y[2]);
#endif

if(y[0]>L[0] && y[0]<L[0]+L3[0]) {
if(y[1]<L3[1]/2.0e0 && y[1]>-L3[1]/2.0e0) {
if(y[2]<L3[2]/2.0e0 && y[2]>-L3[2]/2.0e0) {

bra = genrand_real2(); /* 光電吸収、Compton効果、Rayleigh散乱のうちど
れが起こるかの選択 */

if(bra<pht_rata) {
(*(drop_E+2)) = (*(drop_E+2)) + E + Comp_E;
Res = resolve((*(drop_E+2)));
s = Res * (*(drop_E+2)) / sqrt(2.0e0*log(2.0e0));
r1 = -genrand_real2() + 1.0e0;
r2 = -genrand_real2() + 1.0e0;
```

```
x1 = s * sqrt(-2.0e0*log(r1)) * cos(2.0e0*M_PI*r2);
x2 = s * sqrt(-2.0e0*log(r1)) * sin(2.0e0*M_PI*r2);

op = genrand_real2();
if(op<5.0e-1) {
(*(drop_E+2)) = (*(drop_E+2)) + x1;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+2)), x1, Res, 3);
#endif

}
else {
(*(drop_E+2)) = (*(drop_E+2)) + x2;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+2)), x2, Res, 3);
#endif

}
}
else {
phi = 2.0e0 * M_PI * genrand_real2();
z = 2.0e0 * genrand_real1() - 1.0e0;

new_p[0] = sqrt(1.0e0-z*z) * cos(phi);
new_p[1] = sqrt(1.0e0-z*z) * sin(phi);
new_p[2] = z;

#ifdef DEBUG3
printf("%e %e %e\n", new_p[0], new_p[1], new_p[2]);
#endif

if(bra<pht_rata+cmp_rata) {
temp[0] = (*(p)) - new_p[0];
temp[1] = (*(p+1)) - new_p[1];
temp[2] = (*(p+2)) - new_p[2];

R[0] = sqrt(pow((*(p)), 2.0e0) + pow((*(p+1)), 2.0e0) + pow((*(p+2)), 2.0e0));
```

```
R[1] = sqrt(pow(new_p[0], 2.0e0) + pow(new_p[1], 2.0e0) + pow(new_p[2], 2.0e0));
R[2] = sqrt(pow(temp[0], 2.0e0) + pow(temp[1], 2.0e0) + pow(temp[2], 2.0e0));

sangle = acos((R[0] * R[0] + R[1] * R[1] - R[2] * R[2]) / (2.0e0 * R[1] * R[0]));

#ifdef DEBUG1
int i;
for(i=0;i<36; i++) {
if(sangle<M_PI/3.6e1*((double)i + 1.0e0) && sangle>M_PI/3.6e1*((double)i)) {
Ahist[i]++;
}
}
#endif

new_E = 1.0e0 / ((1.0e0 - cos(sangle)) / E_elec + 1.0e0 / E);
Comp_E = Comp_E + (E - new_E);

scintilation(y, new_p, new_E, Comp_E, drop_E);
}
else {
scintilation(y, new_p, E, Comp_E, drop_E);
}
}

break;

}
}
}

scint = 3;

}
}
}
}

/************************ シンチレーター4 ************************/
```

```
if(scint-4) {
if(x[0]<-L[0] && x[0]>-L[0]-L4[0]) {
if(x[1]<L4[1]/2.0e0 && x[1]>-L4[1]/2.0e0) {
if(x[2]<L4[2]/2.0e0 && x[2]>-L4[2]/2.0e0) {

/******************** NaI中の透過距離の決定 ************************/

mu = (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)) * NaI_density; /* NaI
の線減衰係数の決定 */
pht_rata = phot_ab(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* 光
電吸収が起きる割合 */
cmp_rata = comp_scat(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* コ
ンプトン散乱が起きる割合 */
tau = - log(1.0e0 - genrand_real2()) / mu; /* 透過距離の決定 */

y[0] = x[0] + tau * (*(p)) / r;
y[1] = x[1] + tau * (*(p+1)) / r;
y[2] = x[2] + tau * (*(p+2)) / r;

#ifdef DEBUG
printf("%e %e %e\n", y[0], y[1], y[2]);
#endif

if(y[0]<-L[0] && y[0]>-L[0]-L4[0]) {
if(y[1]<L4[1]/2.0e0 && y[1]>-L4[1]/2.0e0) {
if(y[2]<L4[2]/2.0e0 && y[2]>-L4[2]/2.0e0) {

bra = genrand_real2(); /* 光電吸収、Compton効果、Rayleigh散乱のうちど
れが起こるかの選択 */

if(bra<pht_rata) {
(*(drop_E+3)) = (*(drop_E+3)) + E + Comp_E;
Res = resolve((*(drop_E+3)));
s = Res * (*(drop_E+3)) / sqrt(2.0e0*log(2.0e0));
r1 = -genrand_real2() + 1.0e0;
r2 = -genrand_real2() + 1.0e0;
x1 = s * sqrt(-2.0e0*log(r1)) * cos(2.0e0*M_PI*r2);
x2 = s * sqrt(-2.0e0*log(r1)) * sin(2.0e0*M_PI*r2);
```

```
op = genrand_real2();
if(op<5.0e-1) {
(*(drop_E+3)) = (*(drop_E+3)) + x1;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+3)), x1, Res, 4);
#endif

}
else {
(*(drop_E+3)) = (*(drop_E+3)) + x2;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+3)), x2, Res, 4);
#endif

}
}
else {
phi = 2.0e0 * M_PI * genrand_real2();
z = 2.0e0 * genrand_real1() - 1.0e0;

new_p[0] = sqrt(1.0e0-z*z) * cos(phi);
new_p[1] = sqrt(1.0e0-z*z) * sin(phi);
new_p[2] = z;

#ifdef DEBUG3
printf("%e %e %e\n", new_p[0], new_p[1], new_p[2]);
#endif

if(bra<pht_rata+cmp_rata) {
temp[0] = (*(p)) - new_p[0];
temp[1] = (*(p+1)) - new_p[1];
temp[2] = (*(p+2)) - new_p[2];

R[0] = sqrt(pow((*(p)), 2.0e0) + pow((*(p+1)), 2.0e0) + pow((*(p+2)), 2.0e0));
R[1] = sqrt(pow(new_p[0], 2.0e0) + pow(new_p[1], 2.0e0) + pow(new_p[2], 2.0e0));
R[2] = sqrt(pow(temp[0], 2.0e0) + pow(temp[1], 2.0e0) + pow(temp[2], 2.0e0));
```

```
sangle = acos((R[0] * R[0] + R[1] * R[1] - R[2] * R[2]) / (2.0e0 * R[1] * R[0]));

#ifdef DEBUG1
int i;
for(i=0;i<36; i++) {
if(sangle<M_PI/3.6e1*((double)i + 1.0e0) && sangle>M_PI/3.6e1*((double)i)) {
Ahist[i]++;
}
}
#endif

new_E = 1.0e0 / ((1.0e0 - cos(sangle)) / E_elec + 1.0e0 / E);
Comp_E = Comp_E + (E - new_E);

scintilation(y, new_p, new_E, Comp_E, drop_E);
}
else {
scintilation(y, new_p, E, Comp_E, drop_E);
}
}

break;

}
}
}

scint = 4;

}
}
}
}

/************************ シンチレーター5 ************************/

if(scint-5) {
if(x[0]<L5[0]/2.0e0 && x[0]>-L5[0]/2.0e0) {
if(x[1]<L5[1]/2.0e0 && x[1]>-L5[1]/2.0e0) {
```

```
if(x[2]>L[2] && x[2]<L[2]+L5[2]) {

/******************** NaI中の透過距離の決定 ************************/

mu = (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)) * NaI_density; /* NaI
の線減衰係数の決定 */
pht_rata = phot_ab(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* 光
電吸収が起きる割合 */
cmp_rata = comp_scat(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* コ
ンプトン散乱が起きる割合 */
tau = - log(1.0e0 - genrand_real2()) / mu; /* 透過距離の決定 */

y[0] = x[0] + tau * (*(p)) / r;
y[1] = x[1] + tau * (*(p+1)) / r;
y[2] = x[2] + tau * (*(p+2)) / r;

#ifdef DEBUG
printf("%e %e %e\n", y[0], y[1], y[2]);
#endif

if(y[0]<L5[0]/2.0e0 && y[0]>-L5[0]/2.0e0) {
if(y[1]<L5[1]/2.0e0 && y[1]>-L5[1]/2.0e0) {
if(y[2]>L[2] && y[2]<L[2]+L5[2]) {

bra = genrand_real2(); /* 光電吸収、Compton効果、Rayleigh散乱のうちど
れが起こるかの選択 */

if(bra<pht_rata) {
(*(drop_E+4)) = (*(drop_E+4)) + E + Comp_E;
Res = resolve((*(drop_E+4)));
s = Res * (*(drop_E+4)) / sqrt(2.0e0*log(2.0e0));
r1 = -genrand_real2() + 1.0e0;
r2 = -genrand_real2() + 1.0e0;
x1 = s * sqrt(-2.0e0*log(r1)) * cos(2.0e0*M_PI*r2);
x2 = s * sqrt(-2.0e0*log(r1)) * sin(2.0e0*M_PI*r2);

op = genrand_real2();
if(op<5.0e-1) {
(*(drop_E+4)) = (*(drop_E+4)) + x1;
```

```
#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+4)), x1, Res, 5);
#endif

}
else {
(*(drop_E+4)) = (*(drop_E+4)) + x2;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+4)), x2, Res, 5);
#endif

}
}
else {
phi = 2.0e0 * M_PI * genrand_real2();
z = 2.0e0 * genrand_real1() - 1.0e0;

new_p[0] = sqrt(1.0e0-z*z) * cos(phi);
new_p[1] = sqrt(1.0e0-z*z) * sin(phi);
new_p[2] = z;

#ifdef DEBUG3
printf("%e %e %e\n", new_p[0], new_p[1], new_p[2]);
#endif

if(bra<pht_rata+cmp_rata) {
temp[0] = (*(p)) - new_p[0];
temp[1] = (*(p+1)) - new_p[1];
temp[2] = (*(p+2)) - new_p[2];

R[0] = sqrt(pow((*(p)), 2.0e0) + pow((*(p+1)), 2.0e0) + pow((*(p+2)), 2.0e0));
R[1] = sqrt(pow(new_p[0], 2.0e0) + pow(new_p[1], 2.0e0) + pow(new_p[2], 2.0e0));
R[2] = sqrt(pow(temp[0], 2.0e0) + pow(temp[1], 2.0e0) + pow(temp[2], 2.0e0));

sangle = acos((R[0] * R[0] + R[1] * R[1] - R[2] * R[2]) / (2.0e0 * R[1] * R[0]));

#ifdef DEBUG1
```

```
int i;
for(i=0;i<36; i++) {
if(sangle<M_PI/3.6e1*((double)i + 1.0e0) && sangle>M_PI/3.6e1*((double)i)) {
Ahist[i]++;
}
}
#endif

new_E = 1.0e0 / ((1.0e0 - cos(sangle)) / E_elec + 1.0e0 / E);
Comp_E = Comp_E + (E - new_E);

scintilation(y, new_p, new_E, Comp_E, drop_E);
}
else {
scintilation(y, new_p, E, Comp_E, drop_E);
}
}

break;

}
}
}

scint = 5;

}
}
}
}

if(first) {
if(Comp_E>0.0e0) {
for(i=0; i<5; i++) {
if(scint==i+1) {
(*(drop_E+i)) = (*(drop_E+i)) + Comp_E;
Res = resolve((*(drop_E+i)));
s = Res * (*(drop_E+i)) / sqrt(2.0e0*log(2.0e0));
r1 = -genrand_real2() + 1.0e0;
```

```
r2 = -genrand_real2() + 1.0e0;
x1 = s * sqrt(-2.0e0*log(r1)) * cos(2.0e0*M_PI*r2);
x2 = s * sqrt(-2.0e0*log(r1)) * sin(2.0e0*M_PI*r2);
op = genrand_real2();
if(op<5.0e-1) {
(*(drop_E+i)) = (*(drop_E+i)) + x1;

#ifdef DEBUG4
printf("%e 1%d\n", (*(drop_E+i)), i+1);
#endif

}
else {
(*(drop_E+i)) = (*(drop_E+i)) + x2;

#ifdef DEBUG4
printf("%e 1%d\n", (*(drop_E+i)), i+1);
#endif

}
Comp_E = 0.0e0;
}
}
}
}

x[0] = x[0] + dL * (*(p)) / r;
x[1] = x[1] + dL * (*(p+1)) / r;
x[2] = x[2] + dL * (*(p+2)) / r;
d = sqrt(x[0]*x[0] + x[1]*x[1] + x[2]*x[2]);

first = 0;
}

#ifdef DEBUG
if((*(p))>0.0e0) {
printf("\n\n\n");
}
#endif
```

```
}

/*************************************************************************
* ヒストグラムを作る函数
*************************************************************************/

void histgram(double x, int *num)
{
double min, max;
int n;
register int i;

n = M;

for(i=0; i<n; i++) {
min = dX * ((double)i);
max = dX * (((double)i) + 1.0e0);
if(min<x && x<max) {
(*(num+i))++;
}
}
}

/*************************************************************************
* 光電効果による NaI シンチレーターの質量減衰係数
*************************************************************************/

double phot_ab(double E)
{
double c;

if(E<1.0e-2) {
c = 1.1e2;
}
else if(E<1.5e-2) {
c = 6.0e1;
}
else if(E<2.0e-2) {
```

```
c = 2.7e1;
}
else if(E<2.5e-2) {
c = 1.3e1;
}
else if(E<3.0e-2) {
c = 7.0e0;
}
else if(E<3.4e-2) {
c = 4.8e0;
}
else if(E<4.0e-2) {
c = 2.2e1;
}
else if(E<5.0e-2) {
c = 1.4e1;
}
else if(E<6.0e-2) {
c = 8.0e0;
}
else if(E<7.0e-2) {
c = 4.8e0;
}
else if(E<8.0e-2) {
c = 3.0e0;
}
else if(E<9.0e-2) {
c = 2.3e0;
}
else if(E<1.0e-1) {
c = 1.6e0;
}
else if(E<1.5e-1) {
c = 8.0e-1;
}
else if(E<2.0e-1) {
c = 2.8e-1;
}
else if(E<2.5e-1) {
```

```
c = 1.3e-1;
}
else if(E<3.0e-1) {
c = 7.5e-2;
}
else if(E<4.0e-1) {
c = 3.6e-2;
}
else if(E<5.0e-1) {
c = 2.0e-2;
}
else if(E<6.0e-1){
c = 1.25e-2;
}
else if(E<7.0e-1) {
c = 8.5e-3;
}
else if(E<8.0e-1) {
c = 5.9e-3;
}
else if(E<9.0e-1) {
c = 4.3e-3;
}
else if(E<1.0e0) {
c = 3.5e-3;
}
else {
c = 2.2e-3;
}

return c;
}


/*****************************************************************
* コンプトン効果による NaI シンチレーターの質量減衰係数
*****************************************************************/

double comp_scat(double E)
```

```
{
double c;

if(E<1.0e-2) {
c = 1.65e-1;
}
else if(E<2.0e-2) {
c = 1.6e-1;
}
else if(E<3.0e-2) {
c = 1.55e-2;
}
else if(E<4.0e-2) {
c = 1.5e-1;
}
else if(E<5.0e-2) {
c = 1.45e-1;
}
else if(E<7.0e-2) {
c = 1.4e-1;
}
else if(E<9.0e-2) {
c = 1.35e-1;
}
else if(E<1.0e-1) {
c = 1.3e-1;
}
else if(E<1.5e-1) {
c = 1.25e-1;
}
else if(E<2.0e-1) {
c = 1.1e-1;
}
else if(E<2.5e-1) {
c = 1.0e-1;
}
else if(E<3.0e-1) {
c = 9.25e-2;
}
```

```
else if(E<4.0e-1) {
c = 8.5e-2;
}
else if(E<5.0e-1) {
c = 7.7e-2;
}
else if(E<6.0e-1){
c = 7.5e-2;
}
else if(E<7.0e-1) {
c = 6.7e-2;
}
else if(E<8.0e-1) {
c = 6.3e-2;
}
else if(E<9.0e-1) {
c = 5.8e-2;
}
else if(E<1.0e0) {
c = 5.6e-2;
}
else {
c = 5.0e-2;
}

return c;
}

/*****************************************************************
* Rayleigh 散乱による NaI シンチレーターの質量減衰係数
*****************************************************************/

double Rayleigh_scat(double E)
{
double c;

if(E<1.0e-2) {
c = 2.2e0;
}
```

```
else if(E<1.5e-2) {
c = 1.7e0;
}
else if(E<2.0e-2) {
c = 1.2e0;
}
else if(E<2.5e-2) {
c = 8.0e-1;
}
else if(E<3.0e-2) {
c = 5.7e-1;
}
else if(E<4.0e-2) {
c = 4.1e-1;
}
else if(E<5.0e-2) {
c = 2.7e-1;
}
else if(E<6.0e-2) {
c = 1.8e-1;
}
else if(E<7.0e-2) {
c = 1.4e-1;
}
else if(E<8.0e-2) {
c = 1.1e-1;
}
else if(E<9.0e-2) {
c = 8.8e-2;
}
else if(E<1.0e-1) {
c = 7.3e-2;
}
else if(E<1.5e-1) {
c = 5.0e-2;
}
else if(E<2.0e-1) {
c = 2.6e-2;
}
```

```
else if(E<2.5e-1) {
c = 1.6e-2;
}
else if(E<3.0e-1) {
c = 1.1e-2;
}
else if(E<4.0e-1) {
c = 6.5e-3;
}
else if(E<5.0e-1) {
c = 4.3e-3;
}
else if(E<6.0e-1){
c = 2.9e-3;
}
else if(E<7.0e-1) {
c = 2.1e-3;
}
else if(E<8.0e-1) {
c = 1.6e-3;
}
else if(E<9.0e-1) {
c = 1.25e-3;
}
else if(E<1.0e0) {
c = 1.0e-3;
}
else {
c = 0.0e0;
}

return c;
}

/*************************************************************************
* NaI シンチレーターのエネルギー分解能
*************************************************************************/

double resolve(double E)
```

```
{
double R, K;

K = 9.674e-2;

R = K / sqrt(E);

return R;
}
```

8.2.2 3γ 崩壊

```
/*************************************************************************
 * 2007 年度前期 A1 最終実験シミュレーションプログラム 1
*************************************************************************
 *
 * オルソポジトロニウムの 3 ガンマ崩壊のイベントを、
 * 実験のセットアップで全イベントのうちどれぐらいを
 * 観測できているかをシミュレーションする。
 *
*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "mt19937ar.h"

#define E_elec 5.11e-1 /* 電子の静止エネルギー (MeV) */
#define NaI_density 3.67 /* NaI の密度 (g/立方 cm) */
#define N 100000 /* 3 ガンマ崩壊の全イベント数 */
#define M 128 /* 1.275(MeV) までのカラムの数 */
#define dX 1.0e-2 /* カラムの幅 */

#define dL 1.0e-1
#define R_LIM 2.0e1

#define CUT 2.0e-1 /* Discriminator の閾値によるカット */
```

```
/*
#define DEBUG /**/
/*
#define DEBUG1/**/
/*
#define DEBUG2/**/
/*
#define DEBUG3/**/
/*
#define DEBUG4/**/

double phot_ab(double E); /* NaIシンチレーターの光電吸収による質量減衰
係数 */
double comp_scat(double E); /* NaIシンチレーターのコンプトン効果による
質量減衰係数 */
double Rayleigh_scat(double E); /* Rayleigh散乱による質量減衰係数 */
double resolve(double E); /* シンチレーターのエネルギー分解能 */

void scintilation(double *x0, double *p, double E, double CE, double *drop_E);
void histgram(double x, int *num); /* シンチレーターに入った光子のエネ
ルギー分布を作る函数 */

double L[3], L1[3], L2[3], L3[3], L4[3], L5[3];

int main(void)
{
FILE *fp1, *fp2, *fp3, *fp4, *fp5;
double theta[2], r;
double p1[3], p2[3], p3[3], gamma1[3], gamma2[3], gamma3[3], temp1[3], temp2[3];
double phi1, phi2, ran;
double p[3], m12[4], m23[4], m31[4], msq[3];
double E1, E2, E3, drop_E1[5], drop_E2[5], drop_E3[5], drop_E[5], e[3];
double t, x, y, z;
double op, mu, tau, E, rata, bra, sangle;
double sigma1, sigma2, sigma3, sigma;
int scint[5], sum, count[5], count1[5], count2[5], count3[5], check;
int hist[5][M], e1hist[M], e2hist[M], e3hist[M];
int e12hist[M], e23hist[M], e31hist[M], e123hist[M], k;
```

```
unsigned int experi, num;
register unsigned int i, j;

/*************************************************************************
* 書き込みファイルのオープン
*************************************************************************/

if((fp1=fopen("07_a1_1.txt", "w"))==NULL) {
printf("file open error!!\n");
exit(1);
}

if((fp2=fopen("07_a1_2.txt", "w"))==NULL) {
printf("file open error!!\n");
exit(1);
}

if((fp3=fopen("07_a1_3.txt", "w"))==NULL) {
printf("file open error!!\n");
exit(1);
}

if((fp4=fopen("07_a1_4.txt", "w"))==NULL) {
printf("file open error!!\n");
exit(1);
}

if((fp5=fopen("07_a1_5.txt", "w"))==NULL) {
printf("file open error!!\n");
exit(1);
}

/*************************************************************************
* 書き込み内容の書き込み
*************************************************************************/

fprintf(fp1, "# energy1(Mev) energy2(MeV) energy3(MeV) energy4(NeV)
         energy5(MeV) scinti1 scinti2 scinti3 scinti4 scinti5\n");
```

```
fprintf(fp3, "# m12 m23 m31\n");

/*************************************************************************
* Mersenne Twister（乱数生成器）の初期化
*************************************************************************/

int utime;
long ltime;

ltime = time(NULL);
utime = (unsigned int) ltime/2;
srand(utime);

unsigned long init[4]={rand(), rand(), rand(), rand()}, length=4;
init_by_array(init, length);

/*************************************************************************
* 各シンチレーターの配置に関するデータの設定
*************************************************************************/

/********************* シンチレーターの中心 ************************/

L[0] = 3.25e0;
L[1] = 2.44e0;
L[2] = 3.25e0;

/********************* シンチレーター1の寸法 ************************/

L1[0] = 5.6e0;
L1[1] = 1.693e1;
L1[2] = 5.55e0;

/********************* シンチレーター2の寸法 ************************/

L2[0] = 5.6e0;
L2[1] = 1.689e1;
L2[2] = 5.55e0;

/********************* シンチレーター3の寸法 ************************/
```

```
L3[0] = 5.58e0;
L3[1] = 1.694e1;
L3[2] = 5.55e0;

/********************* シンチレーター4の寸法 ***********************/

L4[0] = 5.59e0;
L4[1] = 1.695e1;
L4[2] = 5.55e0;

/********************* シンチレーター5の寸法 ***********************/

L5[0] = 5.55e0;
L5[1] = 1.694e1;
L5[2] = 5.49e0;

/*************************************************************************
* N事象のうち、測定にかかる回数の初期化
*************************************************************************/

experi = 0;
num = 0;
for(i=0; i<5; i++) {
scint[i] = 0;
for(j=0; j<M; j++) {
hist[i][j] = 0;
}
}
for(i=0; i<M; i++) {
e1hist[j] = 0;
e2hist[j] = 0;
e3hist[j] = 0;
e12hist[i] = 0;
e23hist[i] = 0;
e31hist[i] = 0;
e123hist[i] = 0;
}
```

```
/*****************************************************************
* N 事象のシミュレーションの開始
*****************************************************************/

for(i=0; i<N; i++) {

/*****************************************************************
* ガンマ線の発生位置の決定
*****************************************************************/

gamma1[0] = 0.0e0;
gamma1[1] = 0.0e0;
gamma1[2] = 0.0e0;

gamma2[0] = 0.0e0;
gamma2[1] = 0.0e0;
gamma2[2] = 0.0e0;

gamma3[0] = 0.0e0;
gamma3[1] = 0.0e0;
gamma3[2] = 0.0e0;

/*****************************************************************
* 3 本のガンマ線の運動量ベクトルの決定
*****************************************************************/


#ifdef TRY

theta[0] = M_PI / 3.0e0;
theta[1] = M_PI / 3.0e0;

E1 = 2.0e0 * E_elec / 3.0e0;
E2 = 2.0e0 * E_elec / 3.0e0;
E3 = 2.0e0 * E_elec / 3.0e0;

#else

for(;;) {
```

```
for(;;) {
theta[0] = M_PI * genrand_real1();
theta[1] = M_PI * genrand_real1();

if(theta[0]!=M_PI || theta[1]!=M_PI) {
break;
}
}


if(theta[0]==0.0e0 && theta[1]==0.0e0) {
E1 = E_elec;
E2 = E_elec * genrand_real2();
E3 = E_elec - E2;
}
else if(theta[0]==0.0e0 && theta[1]==M_PI) {
E2 = E_elec;
E3 = E_elec * genrand_real2();
E1 = E_elec - E3;
}
else if(theta[0]==M_PI && theta[1]==0.0e0) {
E3 = E_elec;
E1 = E_elec * genrand_real2();
E2 = E_elec - E1;
}
else {

op = 3.0e0 * genrand_real2();

if(op<1.0e0) {
E2 = 2.0e0 * E_elec * sin(theta[1]) / ((1.0e0 + cos(theta[0]))
                * sin(theta[1]) + (1.0e0 + cos(theta[1])) * sin(theta[0]));
E3 = 2.0e0 * E_elec * sin(theta[0]) / ((1.0e0 + cos(theta[0]))
                * sin(theta[1]) + (1.0e0 + cos(theta[1])) * sin(theta[0]));
E1 = 2.0e0 * E_elec - (E2 + E3);
}
else if(op<2.0e0) {
E3 = 2.0e0 * E_elec * sin(theta[1]) / ((1.0e0 + cos(theta[0]))
                * sin(theta[1]) + (1.0e0 + cos(theta[1])) * sin(theta[0]));
```

```
E1 = 2.0e0 * E_elec * sin(theta[0]) / ((1.0e0 + cos(theta[0]))
                * sin(theta[1]) + (1.0e0 + cos(theta[1])) * sin(theta[0]));
E2 = 2.0e0 * E_elec -(E3 + E1);
}
else {
E1 = 2.0e0 * E_elec * sin(theta[1]) / ((1.0e0 + cos(theta[0]))
                * sin(theta[1]) + (1.0e0 + cos(theta[1])) * sin(theta[0]));
E2 = 2.0e0 * E_elec * sin(theta[0]) / ((1.0e0 + cos(theta[0]))
                * sin(theta[1]) + (1.0e0 + cos(theta[1])) * sin(theta[0]));
E3 = 2.0e0 * E_elec - (E1 + E2);
}
}

if(E1>0.0e0 && E2>0.0e0 && E3>0.0e0) {
break;
}
}

#endif

phi1 = 2.0e0 * M_PI * genrand_real2();

/***************** ガンマ線1の運動量ベクトルの決定 ******************/

phi2 = 2.0e0 * M_PI * genrand_real2();
ran = 2.0e0 * genrand_real1() - 1.0e0;

p1[0] = E1 * sqrt(1.0e0-ran*ran) * cos(phi2);
p1[1] = E1 * sqrt(1.0e0-ran*ran) * sin(phi2);
p1[2] = E1 * ran;

/***************** ガンマ線2の運動量ベクトルの決定 ******************/

temp1[0] = 0.0e0;
temp1[1] = E2 * sin(theta[0]);
temp1[2] = - E2 * cos(theta[0]);

temp2[0] = temp1[0] * cos(phi1) - temp1[1] * sin(phi1);
temp2[1] = temp1[0] * sin(phi1) + temp1[1] * cos(phi1);
```

```
temp2[2] = temp1[2];

temp1[0] = temp2[0] * ran + temp2[2] * sqrt(1.0e0-ran*ran);
temp1[1] = temp2[1];
temp1[2] = -temp2[0] * sqrt(1.0e0-ran*ran) + temp2[2] * ran;

p2[0] = temp1[0] * cos(phi2) - temp1[1] * sin(phi2);
p2[1] = temp1[0] * sin(phi2) + temp1[1] * cos(phi2);
p2[2] = temp1[2];

/***************** ガンマ線3の運動量ベクトルの決定 ******************/

temp1[0] = 0.0e0;
temp1[1] = - E3 * sin(theta[1]);
temp1[2] = - E3 * cos(theta[1]);

temp2[0] = temp1[0] * cos(phi1) - temp1[1] * sin(phi1);
temp2[1] = temp1[0] * sin(phi1) + temp1[1] * cos(phi1);
temp2[2] = temp1[2];

temp1[0] = temp2[0] * ran + temp2[2] * sqrt(1.0e0-ran*ran);
temp1[1] = temp2[1];
temp1[2] = -temp2[0] * sqrt(1.0e0-ran*ran) + temp2[2] * ran;

p3[0] = temp1[0] * cos(phi2) - temp1[1] * sin(phi2);
p3[1] = temp1[0] * sin(phi2) + temp1[1] * cos(phi2);
p3[2] = temp1[2];

/****************** Dalitz plot のための準備 ***********************/

p[0] = sqrt(pow(p1[0], 2.0e0) + pow(p1[1], 2.0e0) + pow(p1[2], 2.0e0));
p[1] = sqrt(pow(p2[0], 2.0e0) + pow(p2[1], 2.0e0) + pow(p2[2], 2.0e0));
p[2] = sqrt(pow(p3[0], 2.0e0) + pow(p3[1], 2.0e0) + pow(p3[2], 2.0e0));

m12[0] = E1 + E2;
m23[0] = E2 + E3;
m31[0] = E3 + E1;
for(j=1; j<4; j++) {
m12[j] = p1[j-1] + p2[j-1];
```

```
m23[j] = p2[j-1] + p3[j-1];
m31[j] = p3[j-1] + p1[j-1];
}

msq[0] = pow(m12[0], 2.0e0) - pow(m12[1], 2.0e0)
                  - pow(m12[2], 2.0e0) - pow(m12[3], 2.0e0);
msq[1] = pow(m23[0], 2.0e0) - pow(m23[1], 2.0e0)
                  - pow(m23[2], 2.0e0) - pow(m23[3], 2.0e0);
msq[2] = pow(m31[0], 2.0e0) - pow(m31[1], 2.0e0)
                 - pow(m31[2], 2.0e0) - pow(m31[3], 2.0e0);

/*********************************************************************
* 各シンチレーターに対する当たり判定
*********************************************************************/


/*********** 各シンチレーターのガンマ線検出数の初期化 **************/

for(j=0; j<5; j++) {
count1[j] = 0;
count2[j] = 0;
count3[j] = 0;
count[j] = 0;
}
check = 0;

/*********** シンチレーターに落とすガンマ線エネルギーの初期化 **************/

for(j=0; j<5; j++) {
drop_E1[j] = 0.0e0;
drop_E2[j] = 0.0e0;
drop_E3[j] = 0.0e0;
}

/***************************** ガンマ線1 ***************************/

scintilation(gamma1, p1, E1, 0.0e0, drop_E1);

/***************************** ガンマ線2 ***************************/
```

```
scintilation(gamma2, p2, E2, 0.0e0, drop_E2);

/*************************** ガンマ線3 **************************/

scintilation(gamma3, p3, E3, 0.0e0, drop_E3);

/*********************************************************************
* N事象のうち、測定にかかる回数と各シンチレーターのカウント
*********************************************************************/

for(j=0; j<5; j++) {
drop_E[j] = drop_E1[j] + drop_E2[j] + drop_E3[j];

/************** Discriminatorの閾値によるカット *****************/

if(drop_E[j]<CUT) {
drop_E[j] = 0.0e0;
}

/*********************************************************************/

histgram(drop_E[j], hist[j]);
if(drop_E[j]>0.0e0) {
count[j]++;
}
}

sum = count[0] + count[1] + count[2] + count[3] + count[4];

if(sum==3) {
k = 0;
for(j=0; j<5; j++) {
scint[j] = scint[j] + count[j];
if(count[j]) {
e[k] = drop_E[j];
k++;
}
}
```

```
experi++;
histgram(e[0], e1hist);
histgram(e[1], e2hist);
histgram(e[2], e3hist);
histgram(e[0]+e[1], e12hist);
histgram(e[1]+e[2], e23hist);
histgram(e[2]+e[0], e31hist);
histgram(e[0]+e[1]+e[2], e123hist);

sigma1 = resolve(E_elec) * (E_elec) / sqrt(2.0e0*log(2.0e0));
sigma2 = resolve(E_elec) * (E_elec) / sqrt(2.0e0*log(2.0e0));
sigma3 = resolve(E_elec) * (E_elec) / sqrt(2.0e0*log(2.0e0));
sigma = resolve(2.0e0*E_elec) * (2.0e0*E_elec) / sqrt(2.0e0*log(2.0e0));
if((e[0]+e[1]+e[2]>2.0e0*E_elec-sigma) && (e[0]+e[1]+e[2]<2.0e0*E_elec+sigma)) {
if(e[0]+e[1]>E_elec-sigma1) {
if(e[1]+e[2]>E_elec-sigma2) {
if(e[2]+e[0]>E_elec-sigma3) {
num++;
}
}
}
}
}

/*********************************************************************
* データのファイルへの書き出し
*********************************************************************/

fprintf(fp1, "%e %e %e %e %e %d %d %d %d %d\n", drop_E[0],
                 drop_E[1], drop_E[2], drop_E[3], drop_E[4], count[0],
                  count[1], count[2], count[3], count[4]);

fprintf(fp3, "%e %e %e\n", msq[0], msq[1], msq[2]);
}

/*********************************************************************
* データのファイルへの書き出し
*********************************************************************/
```

```
fprintf(fp2, "# イベント数 観測数 1sigma の個数 scinti1
         scinti2 scinti3 scinti4 scinti5\n");
fprintf(fp2, "%d %d %d %d %d %d %d %d", N, experi, num,
         scint[0], scint[1], scint[2], scint[3], scint[4]);

for(i=0; i<5; i++) {
fprintf(fp4, "# scintillator %d\n", i+1);
for(j=0; j<M; j++) {
E = dX / 2.0e0 + dX * ((double)j);
fprintf(fp4, "%e %d\n", E, hist[i][j]);
}
fprintf(fp4, "\n\n");
}
fprintf(fp5, "# E1\n");
for(i=0; i<M; i++) {
E = dX / 2.0e0 + dX * ((double)i);
fprintf(fp5, "%e %d\n", E, e1hist[i]);
}
fprintf(fp5, "\n\n");
fprintf(fp5, "# E2\n");
for(i=0; i<M; i++) {
E = dX / 2.0e0 + dX * ((double)i);
fprintf(fp5, "%e %d\n", E, e2hist[i]);
}
fprintf(fp5, "\n\n");
fprintf(fp5, "# E3\n");
for(i=0; i<M; i++) {
E = dX / 2.0e0 + dX * ((double)i);
fprintf(fp5, "%e %d\n", E, e3hist[i]);
}
fprintf(fp5, "\n\n");
fprintf(fp5, "# E1+E2\n");
for(i=0; i<M; i++) {
E = dX / 2.0e0 + dX * ((double)i);
fprintf(fp5, "%e %d\n", E, e12hist[i]);
}
fprintf(fp5, "\n\n");
fprintf(fp5, "# E2+E3\n");
for(i=0; i<M; i++) {
```

```
E = dX / 2.0e0 + dX * ((double)i);
fprintf(fp5, "%e %d\n", E, e23hist[i]);
}
fprintf(fp5, "\n\n");
fprintf(fp5, "# E3+E1\n");
for(i=0; i<M; i++) {
E = dX / 2.0e0 + dX * ((double)i);
fprintf(fp5, "%e %d\n", E, e31hist[i]);
}
fprintf(fp5, "\n\n");
fprintf(fp5, "# E1+E2+E3\n");
for(i=0; i<M; i++) {
E = dX / 2.0e0 + dX * ((double)i);
fprintf(fp5, "%e %d\n", E, e123hist[i]);
}

/*************************************************************************
* 書き込みファイルのクローズ
*************************************************************************/

if(fclose(fp1)==EOF) {
printf("file close error!!\n");
exit(1);
}

if(fclose(fp2)==EOF) {
printf("file close error!!\n");
exit(1);
}

if(fclose(fp3)==EOF) {
printf("file close error!!\n");
exit(1);
}

if(fclose(fp4)==EOF) {
printf("file close error!!\n");
exit(1);
}
```

```
if(fclose(fp5)==EOF) {
printf("file close error!!\n");
exit(1);
}

return 0;
}


/*****************************************************************
* シンチレーターの当たり判定をする函数
*****************************************************************/

void scintilation(double *x0, double *p, double E, double CE, double *drop_E)
{
double x[3], y[3], Comp_E, d;
double phi, z;
double sangle, new_p[3], temp[3], R[3], new_E, r;
double mu, tau, pht_rata, cmp_rata, bra;
double Res, s;
double x1, x2, r1, r2, op;
int first, scint;
int i;

/*****************************************************************************
* シンチレーターについての当たり判定
*****************************************************************************

first = 1;
Comp_E = CE;
scint = 0;

x[0] = (*(x0));
x[1] = (*(x0+1));
x[2] = (*(x0+2));

d = sqrt(x[0]*x[0] + x[1]*x[1] + x[2]*x[2]);
r = sqrt(pow((*(p)), 2.0e0) + pow((*(p+1)), 2.0e0) + pow((*(p+2)), 2.0e0));
```

```
while(d<R_LIM) {

#ifdef DEBUG
if((*(p))>0.0e0) {
printf("%e %e %e\n", x[0], x[1], x[2]);
}
#endif

/************************ シンチレーター1 ************************/

if(scint-1) {
if(x[0]<L1[0]/2.0e0 && x[0]>-L1[0]/2.0e0) {
if(x[1]<L1[1]/2.0e0 && x[1]>-L1[1]/2.0e0) {
if(x[2]<-L[2] && x[2]>-L[2]-L1[2]) {

/******************** NaI中の透過距離の決定 ************************/

mu = (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)) * NaI_density; /* NaI
の線減衰係数の決定 */
pht_rata = phot_ab(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* 光
電吸収が起きる割合 */
cmp_rata = comp_scat(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* コ
ンプトン散乱が起きる割合 */
tau = - log(1.0e0 - genrand_real2()) / mu; /* 透過距離の決定 */

y[0] = x[0] + tau * (*(p)) / r;
y[1] = x[1] + tau * (*(p+1)) / r;
y[2] = x[2] + tau * (*(p+2)) / r;

#ifdef DEBUG
printf("%e %e %e\n", y[0], y[1], y[2]);
#endif

if(y[0]<L1[0]/2.0e0 && y[0]>-L1[0]/2.0e0) {
if(y[1]<L1[1]/2.0e0 && y[1]>-L1[1]/2.0e0) {
if(y[2]<-L[2] && y[2]>-L[2]-L1[2]) {
```

```
bra = genrand_real2(); /* 光電吸収、Compton効果、Rayleigh散乱のうちどれが起こるかの選択 */

if(bra<pht_rata) {
(*(drop_E)) = (*(drop_E)) + E + Comp_E;
Res = resolve((*(drop_E)));
s = Res * (*(drop_E)) / sqrt(2.0e0*log(2.0e0));
r1 = -genrand_real2() + 1.0e0;
r2 = -genrand_real2() + 1.0e0;
x1 = s * sqrt(-2.0e0*log(r1)) * cos(2.0e0*M_PI*r2);
x2 = s * sqrt(-2.0e0*log(r1)) * sin(2.0e0*M_PI*r2);

op = genrand_real2();
if(op<5.0e-1) {
(*(drop_E)) = (*(drop_E)) + x1;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E)), x1, Res, 1);
#endif

}
else {
(*(drop_E)) = (*(drop_E)) + x2;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E)), x2, Res, 1);
#endif

}
}
else {
phi = 2.0e0 * M_PI * genrand_real2();
z = 2.0e0 * genrand_real1() - 1.0e0;

new_p[0] = sqrt(1.0e0-z*z) * cos(phi);
new_p[1] = sqrt(1.0e0-z*z) * sin(phi);
new_p[2] = z;

#ifdef DEBUG3
```

```
printf("%e %e %e\n", new_p[0], new_p[1], new_p[2]);
#endif

if(bra<pht_rata+cmp_rata) {
temp[0] = (*(p)) - new_p[0];
temp[1] = (*(p+1)) - new_p[1];
temp[2] = (*(p+2)) - new_p[2];

R[0] = sqrt(pow((*(p)), 2.0e0) + pow((*(p+1)), 2.0e0) + pow((*(p+2)), 2.0e0));
R[1] = sqrt(pow(new_p[0], 2.0e0) + pow(new_p[1], 2.0e0) + pow(new_p[2], 2.0e0));
R[2] = sqrt(pow(temp[0], 2.0e0) + pow(temp[1], 2.0e0) + pow(temp[2], 2.0e0));

sangle = acos((R[0] * R[0] + R[1] * R[1] - R[2] * R[2]) / (2.0e0 * R[1] * R[0]));

#ifdef DEBUG1
int i;
for(i=0;i<36; i++) {
if(sangle<M_PI/3.6e1*((double)i + 1.0e0) && sangle>M_PI/3.6e1*((double)i)) {
Ahist[i]++;
}
}
#endif

new_E = 1.0e0 / ((1.0e0 - cos(sangle)) / E_elec + 1.0e0 / E);
Comp_E = Comp_E + (E - new_E);

scintilation(y, new_p, new_E, Comp_E, drop_E);
}
else {
scintilation(y, new_p, E, Comp_E, drop_E);
}
}

break;

}
}
}
```

```
scint = 1;

}
}
}
}

/************************ シンチレーター2 ************************/

if(scint-2) {
if(x[0]<L2[0]/2.0e0 && x[0]>-L2[0]/2.0e0) {
if(x[1]>L[1] && x[1]<L[1]+L2[1]) {
if(x[2]<L2[2]/2.0e0 && x[2]>-L2[2]/2.0e0) {

/******************** NaI中の透過距離の決定 ************************/

mu = (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)) * NaI_density; /* NaI
の線減衰係数の決定 */
pht_rata = phot_ab(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* 光
電吸収が起きる割合 */
cmp_rata = comp_scat(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* コ
ンプトン散乱が起きる割合 */
tau = - log(1.0e0 - genrand_real2()) / mu; /* 透過距離の決定 */

y[0] = x[0] + tau * (*(p)) / r;
y[1] = x[1] + tau * (*(p+1)) / r;
y[2] = x[2] + tau * (*(p+2)) / r;

#ifdef DEBUG
printf("%e %e %e\n", y[0], y[1], y[2]);
#endif

if(y[0]<L2[0]/2.0e0 && y[0]>-L2[0]/2.0e0) {
if(y[1]>L[1] && y[1]<L[1]+L2[1]) {
if(y[2]<L2[2]/2.0e0 && y[2]>-L2[2]/2.0e0) {

bra = genrand_real2(); /* 光電吸収、Compton効果、Rayleigh散乱のうちど
れが起こるかの選択 */
```

```
if(bra<pht_rata) {
(*(drop_E+1)) = (*(drop_E+1)) + E + Comp_E;
Res = resolve((*(drop_E+1)));
s = Res * (*(drop_E+1)) / sqrt(2.0e0*log(2.0e0));
r1 = -genrand_real2() + 1.0e0;
r2 = -genrand_real2() + 1.0e0;
x1 = s * sqrt(-2.0e0*log(r1)) * cos(2.0e0*M_PI*r2);
x2 = s * sqrt(-2.0e0*log(r1)) * sin(2.0e0*M_PI*r2);

op = genrand_real2();
if(op<5.0e-1) {
(*(drop_E+1)) = (*(drop_E+1)) + x1;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+1)), x1, Res, 2);
#endif

}
else {
(*(drop_E+1)) = (*(drop_E+1)) + x2;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+1)), x2, Res, 2);
#endif

}
}
else {
phi = 2.0e0 * M_PI * genrand_real2();
z = 2.0e0 * genrand_real1() - 1.0e0;

new_p[0] = sqrt(1.0e0-z*z) * cos(phi);
new_p[1] = sqrt(1.0e0-z*z) * sin(phi);
new_p[2] = z;

#ifdef DEBUG3
printf("%e %e %e\n", new_p[0], new_p[1], new_p[2]);
#endif
```

```
if(bra<pht_rata+cmp_rata) {
temp[0] = (*(p)) - new_p[0];
temp[1] = (*(p+1)) - new_p[1];
temp[2] = (*(p+2)) - new_p[2];

R[0] = sqrt(pow((*(p)), 2.0e0) + pow((*(p+1)), 2.0e0) + pow((*(p+2)), 2.0e0));
R[1] = sqrt(pow(new_p[0], 2.0e0) + pow(new_p[1], 2.0e0) + pow(new_p[2], 2.0e0));
R[2] = sqrt(pow(temp[0], 2.0e0) + pow(temp[1], 2.0e0) + pow(temp[2], 2.0e0));

sangle = acos((R[0] * R[0] + R[1] * R[1] - R[2] * R[2]) / (2.0e0 * R[1] * R[0]));

#ifdef DEBUG1
int i;
for(i=0;i<36; i++) {
if(sangle<M_PI/3.6e1*((double)i + 1.0e0) && sangle>M_PI/3.6e1*((double)i)) {
Ahist[i]++;
}
}
#endif

new_E = 1.0e0 / ((1.0e0 - cos(sangle)) / E_elec + 1.0e0 / E);
Comp_E = Comp_E + (E - new_E);

scintilation(y, new_p, new_E, Comp_E, drop_E);
}
else {
scintilation(y, new_p, E, Comp_E, drop_E);
}
}

break;

}
}
}

scint = 2;

}
```

```
}
}
}

/************************ シンチレーター3 ************************/

if(scint-3) {
if(x[0]>L[0] && x[0]<L[0]+L3[0]) {
if(x[1]<L3[1]/2.0e0 && x[1]>-L3[1]/2.0e0) {
if(x[2]<L3[2]/2.0e0 && x[2]>-L3[2]/2.0e0) {

/******************** NaI中の透過距離の決定 ************************/

mu = (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)) * NaI_density; /* NaI
の線減衰係数の決定 */
pht_rata = phot_ab(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* 光
電吸収が起きる割合 */
cmp_rata = comp_scat(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* コ
ンプトン散乱が起きる割合 */
tau = - log(1.0e0 - genrand_real2()) / mu; /* 透過距離の決定 */

y[0] = x[0] + tau * (*(p)) / r;
y[1] = x[1] + tau * (*(p+1)) / r;
y[2] = x[2] + tau * (*(p+2)) / r;

#ifdef DEBUG
printf("%e %e %e\n", y[0], y[1], y[2]);
#endif

if(y[0]>L[0] && y[0]<L[0]+L3[0]) {
if(y[1]<L3[1]/2.0e0 && y[1]>-L3[1]/2.0e0) {
if(y[2]<L3[2]/2.0e0 && y[2]>-L3[2]/2.0e0) {

bra = genrand_real2(); /* 光電吸収、Compton効果、Rayleigh散乱のうちど
れが起こるかの選択 */

if(bra<pht_rata) {
(*(drop_E+2)) = (*(drop_E+2)) + E + Comp_E;
Res = resolve((*(drop_E+2)));
```

```
s = Res * (*(drop_E+2)) / sqrt(2.0e0*log(2.0e0));
r1 = -genrand_real2() + 1.0e0;
r2 = -genrand_real2() + 1.0e0;
x1 = s * sqrt(-2.0e0*log(r1)) * cos(2.0e0*M_PI*r2);
x2 = s * sqrt(-2.0e0*log(r1)) * sin(2.0e0*M_PI*r2);

op = genrand_real2();
if(op<5.0e-1) {
(*(drop_E+2)) = (*(drop_E+2)) + x1;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+2)), x1, Res, 3);
#endif

}
else {
(*(drop_E+2)) = (*(drop_E+2)) + x2;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+2)), x2, Res, 3);
#endif

}
}
else {
phi = 2.0e0 * M_PI * genrand_real2();
z = 2.0e0 * genrand_real1() - 1.0e0;

new_p[0] = sqrt(1.0e0-z*z) * cos(phi);
new_p[1] = sqrt(1.0e0-z*z) * sin(phi);
new_p[2] = z;

#ifdef DEBUG3
printf("%e %e %e\n", new_p[0], new_p[1], new_p[2]);
#endif

if(bra<pht_rata+cmp_rata) {
temp[0] = (*(p)) - new_p[0];
temp[1] = (*(p+1)) - new_p[1];
```

```
temp[2] = (*(p+2)) - new_p[2];

R[0] = sqrt(pow((*(p)), 2.0e0) + pow((*(p+1)), 2.0e0) + pow((*(p+2)), 2.0e0));
R[1] = sqrt(pow(new_p[0], 2.0e0) + pow(new_p[1], 2.0e0) + pow(new_p[2], 2.0e0));
R[2] = sqrt(pow(temp[0], 2.0e0) + pow(temp[1], 2.0e0) + pow(temp[2], 2.0e0));

sangle = acos((R[0] * R[0] + R[1] * R[1] - R[2] * R[2]) / (2.0e0 * R[1] * R[0]));

#ifdef DEBUG1
int i;
for(i=0;i<36; i++) {
if(sangle<M_PI/3.6e1*((double)i + 1.0e0) && sangle>M_PI/3.6e1*((double)i)) {
Ahist[i]++;
}
}
#endif

new_E = 1.0e0 / ((1.0e0 - cos(sangle)) / E_elec + 1.0e0 / E);
Comp_E = Comp_E + (E - new_E);

scintilation(y, new_p, new_E, Comp_E, drop_E);
}
else {
scintilation(y, new_p, E, Comp_E, drop_E);
}
}

break;

}
}
}

scint = 3;

}
}
}
}
```

```
/************************ シンチレーター4 ************************/

if(scint-4) {
if(x[0]<-L[0] && x[0]>-L[0]-L4[0]) {
if(x[1]<L4[1]/2.0e0 && x[1]>-L4[1]/2.0e0) {
if(x[2]<L4[2]/2.0e0 && x[2]>-L4[2]/2.0e0) {

/******************** NaI中の透過距離の決定 ************************/

mu = (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)) * NaI_density; /* NaI
の線減衰係数の決定 */
pht_rata = phot_ab(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* 光
電吸収が起きる割合 */
cmp_rata = comp_scat(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* コ
ンプトン散乱が起きる割合 */
tau = - log(1.0e0 - genrand_real2()) / mu; /* 透過距離の決定 */

y[0] = x[0] + tau * (*(p)) / r;
y[1] = x[1] + tau * (*(p+1)) / r;
y[2] = x[2] + tau * (*(p+2)) / r;

#ifdef DEBUG
printf("%e %e %e\n", y[0], y[1], y[2]);
#endif

if(y[0]<-L[0] && y[0]>-L[0]-L4[0]) {
if(y[1]<L4[1]/2.0e0 && y[1]>-L4[1]/2.0e0) {
if(y[2]<L4[2]/2.0e0 && y[2]>-L4[2]/2.0e0) {

bra = genrand_real2(); /* 光電吸収、Compton効果、Rayleigh散乱のうちど
れが起こるかの選択 */

if(bra<pht_rata) {
(*(drop_E+3)) = (*(drop_E+3)) + E + Comp_E;
Res = resolve((*(drop_E+3)));
s = Res * (*(drop_E+3)) / sqrt(2.0e0*log(2.0e0));
r1 = -genrand_real2() + 1.0e0;
r2 = -genrand_real2() + 1.0e0;
```

```
x1 = s * sqrt(-2.0e0*log(r1)) * cos(2.0e0*M_PI*r2);
x2 = s * sqrt(-2.0e0*log(r1)) * sin(2.0e0*M_PI*r2);

op = genrand_real2();
if(op<5.0e-1) {
(*(drop_E+3)) = (*(drop_E+3)) + x1;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+3)), x1, Res, 4);
#endif

}
else {
(*(drop_E+3)) = (*(drop_E+3)) + x2;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+3)), x2, Res, 4);
#endif

}
}
else {
phi = 2.0e0 * M_PI * genrand_real2();
z = 2.0e0 * genrand_real1() - 1.0e0;

new_p[0] = sqrt(1.0e0-z*z) * cos(phi);
new_p[1] = sqrt(1.0e0-z*z) * sin(phi);
new_p[2] = z;

#ifdef DEBUG3
printf("%e %e %e\n", new_p[0], new_p[1], new_p[2]);
#endif

if(bra<pht_rata+cmp_rata) {
temp[0] = (*(p)) - new_p[0];
temp[1] = (*(p+1)) - new_p[1];
temp[2] = (*(p+2)) - new_p[2];

R[0] = sqrt(pow((*(p)), 2.0e0) + pow((*(p+1)), 2.0e0) + pow((*(p+2)), 2.0e0));
```

```
R[1] = sqrt(pow(new_p[0], 2.0e0) + pow(new_p[1], 2.0e0) + pow(new_p[2], 2.0e0));
R[2] = sqrt(pow(temp[0], 2.0e0) + pow(temp[1], 2.0e0) + pow(temp[2], 2.0e0));

sangle = acos((R[0] * R[0] + R[1] * R[1] - R[2] * R[2]) / (2.0e0 * R[1] * R[0]));

#ifdef DEBUG1
int i;
for(i=0;i<36; i++) {
if(sangle<M_PI/3.6e1*((double)i + 1.0e0) && sangle>M_PI/3.6e1*((double)i)) {
Ahist[i]++;
}
}
#endif

new_E = 1.0e0 / ((1.0e0 - cos(sangle)) / E_elec + 1.0e0 / E);
Comp_E = Comp_E + (E - new_E);

scintilation(y, new_p, new_E, Comp_E, drop_E);
}
else {
scintilation(y, new_p, E, Comp_E, drop_E);
}
}

break;

}
}
}

scint = 4;

}
}
}
}

/************************ シンチレーター5 ************************/
```

```
if(scint-5) {
if(x[0]<L5[0]/2.0e0 && x[0]>-L5[0]/2.0e0) {
if(x[1]<L5[1]/2.0e0 && x[1]>-L5[1]/2.0e0) {
if(x[2]>L[2] && x[2]<L[2]+L5[2]) {

/******************** NaI中の透過距離の決定 ************************/

mu = (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)) * NaI_density; /* NaI
の線減衰係数の決定 */
pht_rata = phot_ab(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* 光
電吸収が起きる割合 */
cmp_rata = comp_scat(E) / (phot_ab(E) + comp_scat(E) + Rayleigh_scat(E)); /* コ
ンプトン散乱が起きる割合 */
tau = - log(1.0e0 - genrand_real2()) / mu; /* 透過距離の決定 */

y[0] = x[0] + tau * (*(p)) / r;
y[1] = x[1] + tau * (*(p+1)) / r;
y[2] = x[2] + tau * (*(p+2)) / r;

#ifdef DEBUG
printf("%e %e %e\n", y[0], y[1], y[2]);
#endif

if(y[0]<L5[0]/2.0e0 && y[0]>-L5[0]/2.0e0) {
if(y[1]<L5[1]/2.0e0 && y[1]>-L5[1]/2.0e0) {
if(y[2]>L[2] && y[2]<L[2]+L5[2]) {

bra = genrand_real2(); /* 光電吸収、Compton効果、Rayleigh散乱のうちど
れが起こるかの選択 */

if(bra<pht_rata) {
(*(drop_E+4)) = (*(drop_E+4)) + E + Comp_E;
Res = resolve((*(drop_E+4)));
s = Res * (*(drop_E+4)) / sqrt(2.0e0*log(2.0e0));
r1 = -genrand_real2() + 1.0e0;
r2 = -genrand_real2() + 1.0e0;
x1 = s * sqrt(-2.0e0*log(r1)) * cos(2.0e0*M_PI*r2);
x2 = s * sqrt(-2.0e0*log(r1)) * sin(2.0e0*M_PI*r2);
```

```
op = genrand_real2();
if(op<5.0e-1) {
(*(drop_E+4)) = (*(drop_E+4)) + x1;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+4)), x1, Res, 5);
#endif

}
else {
(*(drop_E+4)) = (*(drop_E+4)) + x2;

#ifdef DEBUG4
printf("%e %e %e 0%d\n", (*(drop_E+4)), x2, Res, 5);
#endif

}
}
else {
phi = 2.0e0 * M_PI * genrand_real2();
z = 2.0e0 * genrand_real1() - 1.0e0;

new_p[0] = sqrt(1.0e0-z*z) * cos(phi);
new_p[1] = sqrt(1.0e0-z*z) * sin(phi);
new_p[2] = z;

#ifdef DEBUG3
printf("%e %e %e\n", new_p[0], new_p[1], new_p[2]);
#endif

if(bra<pht_rata+cmp_rata) {
temp[0] = (*(p)) - new_p[0];
temp[1] = (*(p+1)) - new_p[1];
temp[2] = (*(p+2)) - new_p[2];

R[0] = sqrt(pow((*(p)), 2.0e0) + pow((*(p+1)), 2.0e0) + pow((*(p+2)), 2.0e0));
R[1] = sqrt(pow(new_p[0], 2.0e0) + pow(new_p[1], 2.0e0) + pow(new_p[2], 2.0e0));
R[2] = sqrt(pow(temp[0], 2.0e0) + pow(temp[1], 2.0e0) + pow(temp[2], 2.0e0));
```

```
sangle = acos((R[0] * R[0] + R[1] * R[1] - R[2] * R[2]) / (2.0e0 * R[1] * R[0]));

#ifdef DEBUG1
int i;
for(i=0;i<36; i++) {
if(sangle<M_PI/3.6e1*((double)i + 1.0e0) && sangle>M_PI/3.6e1*((double)i)) {
Ahist[i]++;
}
}
#endif

new_E = 1.0e0 / ((1.0e0 - cos(sangle)) / E_elec + 1.0e0 / E);
Comp_E = Comp_E + (E - new_E);

scintilation(y, new_p, new_E, Comp_E, drop_E);
}
else {
scintilation(y, new_p, E, Comp_E, drop_E);
}
}

break;

}
}
}

scint = 5;

}
}
}
}

if(first) {
if(Comp_E>0.0e0) {
for(i=0; i<5; i++) {
if(scint==i+1) {
(*(drop_E+i)) = (*(drop_E+i)) + Comp_E;
```

```
Res = resolve((*(drop_E+i)));
s = Res * (*(drop_E+i)) / sqrt(2.0e0*log(2.0e0));
r1 = -genrand_real2() + 1.0e0;
r2 = -genrand_real2() + 1.0e0;
x1 = s * sqrt(-2.0e0*log(r1)) * cos(2.0e0*M_PI*r2);
x2 = s * sqrt(-2.0e0*log(r1)) * sin(2.0e0*M_PI*r2);
op = genrand_real2();
if(op<5.0e-1) {
(*(drop_E+i)) = (*(drop_E+i)) + x1;

#ifdef DEBUG4
printf("%e 1%d\n", (*(drop_E+i)), i+1);
#endif

}
else {
(*(drop_E+i)) = (*(drop_E+i)) + x2;

#ifdef DEBUG4
printf("%e 1%d\n", (*(drop_E+i)), i+1);
#endif

}
Comp_E = 0.0e0;
}
}
}
}

x[0] = x[0] + dL * (*(p)) / r;
x[1] = x[1] + dL * (*(p+1)) / r;
x[2] = x[2] + dL * (*(p+2)) / r;
d = sqrt(x[0]*x[0] + x[1]*x[1] + x[2]*x[2]);

first = 0;
}

#ifdef DEBUG
if((*(p))>0.0e0) {
```

```
printf("\n\n\n");
}
#endif

}

/*************************************************************************
* ヒストグラムを作る函数
*************************************************************************/

void histgram(double x, int *num)
{
double min, max;
int n;
register int i;

n = M;

for(i=0; i<n; i++) {
min = dX * ((double)i);
max = dX * (((double)i) + 1.0e0);
if(min<x && x<max) {
(*(num+i))++;
}
}
}

/*************************************************************************
* 光電効果による NaI シンチレーターの質量減衰係数
*************************************************************************/

double phot_ab(double E)
{
double c;

if(E<1.0e-2) {
c = 1.1e2;
}
else if(E<1.5e-2) {
```

```
c = 6.0e1;
}
else if(E<2.0e-2) {
c = 2.7e1;
}
else if(E<2.5e-2) {
c = 1.3e1;
}
else if(E<3.0e-2) {
c = 7.0e0;
}
else if(E<3.4e-2) {
c = 4.8e0;
}
else if(E<4.0e-2) {
c = 2.2e1;
}
else if(E<5.0e-2) {
c = 1.4e1;
}
else if(E<6.0e-2) {
c = 8.0e0;
}
else if(E<7.0e-2) {
c = 4.8e0;
}
else if(E<8.0e-2) {
c = 3.0e0;
}
else if(E<9.0e-2) {
c = 2.3e0;
}
else if(E<1.0e-1) {
c = 1.6e0;
}
else if(E<1.5e-1) {
c = 8.0e-1;
}
else if(E<2.0e-1) {
```

```
c = 2.8e-1;
}
else if(E<2.5e-1) {
c = 1.3e-1;
}
else if(E<3.0e-1) {
c = 7.5e-2;
}
else if(E<4.0e-1) {
c = 3.6e-2;
}
else if(E<5.0e-1) {
c = 2.0e-2;
}
else if(E<6.0e-1){
c = 1.25e-2;
}
else if(E<7.0e-1) {
c = 8.5e-3;
}
else if(E<8.0e-1) {
c = 5.9e-3;
}
else if(E<9.0e-1) {
c = 4.3e-3;
}
else if(E<1.0e0) {
c = 3.5e-3;
}
else {
c = 2.2e-3;
}

return c;
}


/*****************************************************************
* コンプトン効果による NaI シンチレーターの質量減衰係数
```

```
**********************************************************************/

double comp_scat(double E)
{
double c;

if(E<1.0e-2) {
c = 1.65e-1;
}
else if(E<2.0e-2) {
c = 1.6e-1;
}
else if(E<3.0e-2) {
c = 1.55e-2;
}
else if(E<4.0e-2) {
c = 1.5e-1;
}
else if(E<5.0e-2) {
c = 1.45e-1;
}
else if(E<7.0e-2) {
c = 1.4e-1;
}
else if(E<9.0e-2) {
c = 1.35e-1;
}
else if(E<1.0e-1) {
c = 1.3e-1;
}
else if(E<1.5e-1) {
c = 1.25e-1;
}
else if(E<2.0e-1) {
c = 1.1e-1;
}
else if(E<2.5e-1) {
c = 1.0e-1;
}
```

```
else if(E<3.0e-1) {
c = 9.25e-2;
}
else if(E<4.0e-1) {
c = 8.5e-2;
}
else if(E<5.0e-1) {
c = 7.7e-2;
}
else if(E<6.0e-1){
c = 7.5e-2;
}
else if(E<7.0e-1) {
c = 6.7e-2;
}
else if(E<8.0e-1) {
c = 6.3e-2;
}
else if(E<9.0e-1) {
c = 5.8e-2;
}
else if(E<1.0e0) {
c = 5.6e-2;
}
else {
c = 5.0e-2;
}

return c;
}

/*****************************************************************
* Rayleigh 散乱による NaI シンチレーターの質量減衰係数
*****************************************************************/

double Rayleigh_scat(double E)
{
double c;
```

```
if(E<1.0e-2) {
c = 2.2e0;
}
else if(E<1.5e-2) {
c = 1.7e0;
}
else if(E<2.0e-2) {
c = 1.2e0;
}
else if(E<2.5e-2) {
c = 8.0e-1;
}
else if(E<3.0e-2) {
c = 5.7e-1;
}
else if(E<4.0e-2) {
c = 4.1e-1;
}
else if(E<5.0e-2) {
c = 2.7e-1;
}
else if(E<6.0e-2) {
c = 1.8e-1;
}
else if(E<7.0e-2) {
c = 1.4e-1;
}
else if(E<8.0e-2) {
c = 1.1e-1;
}
else if(E<9.0e-2) {
c = 8.8e-2;
}
else if(E<1.0e-1) {
c = 7.3e-2;
}
else if(E<1.5e-1) {
c = 5.0e-2;
}
```

```
else if(E<2.0e-1) {
c = 2.6e-2;
}
else if(E<2.5e-1) {
c = 1.6e-2;
}
else if(E<3.0e-1) {
c = 1.1e-2;
}
else if(E<4.0e-1) {
c = 6.5e-3;
}
else if(E<5.0e-1) {
c = 4.3e-3;
}
else if(E<6.0e-1){
c = 2.9e-3;
}
else if(E<7.0e-1) {
c = 2.1e-3;
}
else if(E<8.0e-1) {
c = 1.6e-3;
}
else if(E<9.0e-1) {
c = 1.25e-3;
}
else if(E<1.0e0) {
c = 1.0e-3;
}
else {
c = 0.0e0;
}

return c;
}

/*****************************************************************
* NaI シンチレーターのエネルギー分解能
```

```
**********************************************************************/

double resolve(double E)
{
double R, K;

K = 9.674e-2;

R = K / sqrt(E);

return R;
}
```

参考文献

[1] http://www.math.sci.hiroshima-u.ac.jp/ m-mat/MT/mt.html

[2] 木村 逸郎、阪井 英次（1982）『放射線計測ハンドブック』日刊工業新聞社

[3] Micheal E.Peskin, Daniel V.Schroeder, *An Introduction to Quantum Field Theory*, Westview Press, 1995

[4] Lifshitz, E.M, Pitaevskii, L.P, Berektetskii, V.B, *Quantum Electrodynamics*, Butterworth-Heinemann, 1982

[5] P.A. Vetter, S.J. freedman, Phys. Rev. A **66**, 052505 (2002)