2023 年度　卒業論文

発光波形情報を用いた
シンチレータ検出器のエネルギー分解能の向上

広島大学 理学部 物理学科
高エネルギー宇宙・可視赤外線天文学研究室

B203866　宮島 渓太

主査： 高橋 弘充
副査： 伊藤 清一

2024 年 3 月 24 日

概 要

シンチレータ検出器はガンマ線や硬 X 線などの高エネルギーの電磁波をはじめとした種々の放射線を検出することができ、人工衛星等に搭載されて宇宙の物理現象の観測に役立てられている。
シンチレータは光電吸収あるいはコンプトン散乱によってガンマ線からエネルギーを受け取り、そのエネルギーに応じた数の可視光を発する。発せられる光量は数マイクロ秒かけて減衰していき、この発光の時間変化が検出器の信号波形となる。
ガンマ線によるシンチレーション発光波形がイベントのエネルギーによって異なることが報告されている。本研究では発光波形から得られる情報を用いて従来の手法によって得られたエネルギーを補正し、エネルギー分解能を向上することを目指した。
発光波形の取得には、光検出器 MPPC と毎秒約 2.5×10^8 回信号をサンプリングできる ADC を用いた。使用したシンチレータは CsI(Tl) と GAGG(Ce) である。
発光波形を特徴づけるようなパラメータを定義して、エネルギーの検出値との相関関係からエネルギーを補正することを検討した。結果として、GAGG(Ce) シンチレータでは同一のエネルギーでもイベントごとにシンチレーション発光波形が一定程度ばらついており、これが検出器のエネルギー分解能悪化の一因になっていることを発見した。この効果を打ち消すように従来の算出法で計算されたエネルギー値を補正し、エネルギー分解能を向上することを目指したが、有意なエネルギー分解能の向上は達成できなかった。
付随してシンチレータ検出器の信号を積分してエネルギーを求める際の、適切な積分時間のエネルギー依存性についても考察した。これによって GAGG(Ce) シンチレータを使った検出器で、観測したいエネルギー帯に応じて適切な積分時間を選べるようになった。

目 次

図 目 次

第1章　研究背景・目的

1.1　研究の背景

1.1.1　シンチレータ検出器

ガンマ線や荷電粒子などの放射線を検出するために、シンチレータ検出器が広く使われている。シンチレータ検出器は、放射線と反応した際に可視光を発するシンチレータ自身と、発光した可視光を検出する光検出器から構成される。

1.1.2　シンチレーション発光波形

ガンマ線がシンチレータに入射すると、コンプトン散乱もしくは光電吸収によってそのエネルギーの一部または全部がシンチレータに与えられ、与えられたエネルギーに応じた数の可視光光子が発生する。この可視光光子を数えればシンチレーションイベントのエネルギーを見積もることができる。可視光光子は PMT（光電子増倍管）や光検出器 MPPC（Multi-Pixel Photon Counter）などを用いて電気信号に変換される。シンチレータの発光は通常数マイクロ秒ほどの時間をかけて減衰する。電気信号もそれに従って減衰するので、イベントのエネルギーを正確に知るにはこの電気信号をアンプを用いて積分するか、本研究で行ったようにデジタルデータを計算機で処理する。

この電気信号の波形、すなわちシンチレータの発光は、複数の減衰する指数関数の足し合わせで表される。多くの場合足し合わせる指数関数の数は１つもしくは２つでおおよそ表すことができる。近年シンチレーション発光波形が、イベントのエネルギーによって異なっていることが報告されている [2]。

1.1.3　ガンマ線のエネルギーによる発光波形の違い

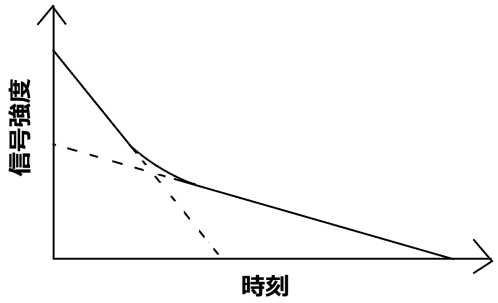


図 1.1: 簡略化した発行波形（縦軸対数）

シンチレーション発光波形を図 1.1 のように 2 つの減衰する指数関数の足し合わせと考える。縦軸が対数スケールであることに注意されたい。

CsI(Tl) シンチレータでは、この発光波形がエネルギー依存性を持つことが報告されている。エネルギーが大きくなるに従って波形がより長い尾を引くようになると考えられている。図 1.2 は図 1.1 で表したような波形がエネルギーによってどう変化するか、わかりやすいように誇張して描いている。実際にはひと目見ただけで区別できるほど大きな違いではない。

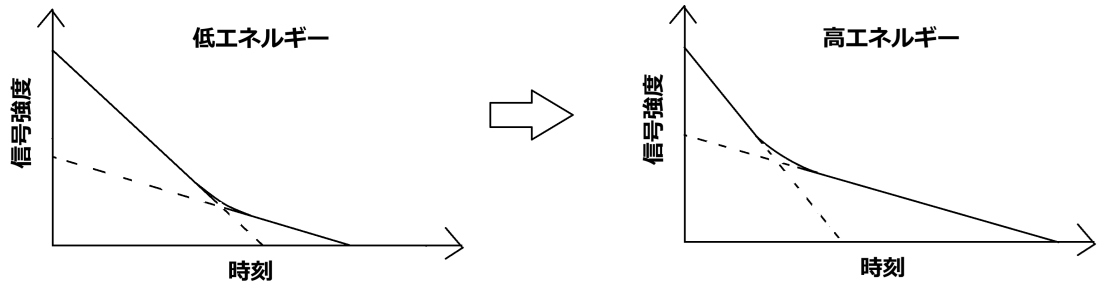


図 1.2: 発光波形のエネルギー依存性

1.2 研究の目的

シンチレータの発光波形情報を用いて、単なる信号の積分で得られたエネルギーを補正し、シンチレーション検出器のエネルギー分解能を向上することを目指す。具体的には発光波形を特徴づけるようなパラメータを導入し、その値とエネルギーとの相関を考察し、従来の方法で得られたエネルギーをより真のエネルギーに近い値に修正する方法を考える。

第2章　ガンマ線とシンチレータの反応

2.1　光電吸収とコンプトン散乱

線源から飛来したガンマ線はシンチレータに入射して、光電吸収もしくはコンプトン散乱によりシンチレータ中の電子にエネルギーを与える。

2.1.1　光電吸収

光電吸収とは、入射した電磁波のエネルギーのすべてが物質中の原子に吸収され、それに応じたエネルギーを持つ電子が原子の束縛から解かれることである。このとき電磁波は消滅する。この電子を光電子と呼び、光電子が持つエネルギー E_{e^-} は、

$$E_{e^-} = h\nu - E_b \tag{2.1}$$

と表せる。ここで ν は入射した電磁波のエネルギー、E_b は仕事関数である。仕事関数とは光電子が原子に束縛されていたエネルギーである。電子が欠損した準位に外殻電子が落ちる際には、仕事関数に対応する別個の X 線を放射する。これを特性 X 線と呼ぶ。よって、光電吸収プロセスでは、電子線と特性 X 線の 2 本の放射線が発生している。

2.1.2　コンプトン散乱

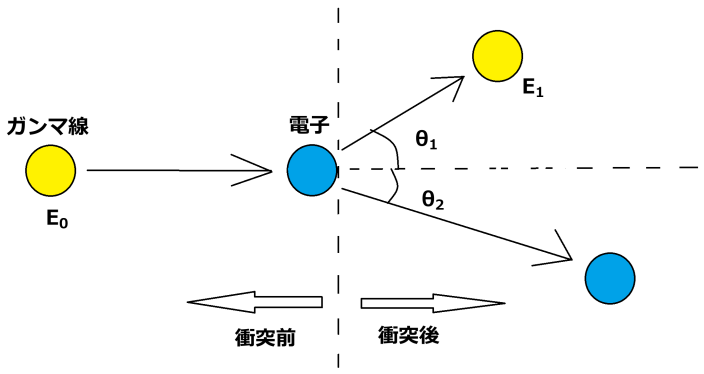


図 2.1: コンプトン散乱

コンプトン散乱は、光子が電子に衝突して散乱する効果のことである。光子は電子に衝突して、そのエネルギーの一部を与える。光子はその分のエネルギーを失うが、光電吸収とは異なり消滅することはない。

図 2.1 のように静止した電子に光子が衝突することを考えると、この衝突によって光子が電子に与えるエネルギーは

$$E_0 - E_1 = \frac{E_0^2(1-\cos\theta_1)}{mc^2 + E_0(1-\cos\theta_1)} \tag{2.2}$$

となる。その逆に衝突後の光子が持っているエネルギーは、

$$E_1 = E_0 - \frac{E_0^2(1-\cos\theta_1)}{mc^2 + E_0(1-\cos\theta_1)} \tag{2.3}$$

となる。式 (2.2) で $\theta_1 = \pi$ のときコンプトン散乱によって電子に与えられるエネルギーが最大になり、

$$E_0 - E_1 = \frac{2E_0^2}{mc^2 + 2E_0} \tag{2.4}$$

となる。このとき散乱後の光子が持っているエネルギーは、

$$E_1 = E_0 - \frac{2E_0^2}{mc^2 + 2E_0} \tag{2.5}$$

となる。

2.2 無機シンチレータの発光

高エネルギーの電磁波がシンチレータに入射すると前述の光電吸収もしくはコンプトン散乱により、シンチレータ中のひとつの電子にエネルギーが与えられる。（この時特性 X 線も発生している。）この高エネルギーを持った電子はシンチレータ内を動き回り、そのエネルギーがなくなるまで結晶中に多数の電子・正孔対を励起する。シンチレータには純粋結晶の禁制帯内に活性化物質（不純物）が作り出したエネルギー準位があり、励起された電子と正孔は活性化物質が作り出したエネルギー準位に入り励起状態となる [3]。これが脱励起することによってシンチレーションが発生する。この過程を図にしたのが図 2.2 である。このとき電子・正孔対がどの励起状態に入っているかによって遷移確率が異なるので、複数の減衰時間を持っている。これが 1.1 節で触れた、信号が複数の指数関数の足し合わせで表すことができる理由である。活性化物質が必要なのは、純粋結晶中で電子が禁制帯を飛び越えて価電子帯に戻り発光するのは効率の悪い過程であり、さらにギャップの幅が大きすぎて可視光とはならず検出するのに不都合だからである [3]。

2.3 本研究で用いたシンチレータ

本研究ではいずれも無機・固体の単結晶シンチレータである CsI(Tl)、GAGG(Ce) を用いた。形はいずれも 10 mm 角の立方体である。図 2.3 に示すとおり CsI シンチレータは少し白く濁った無色透明であり、図 2.4 のように GAGG シンチレータは黄色透明である。

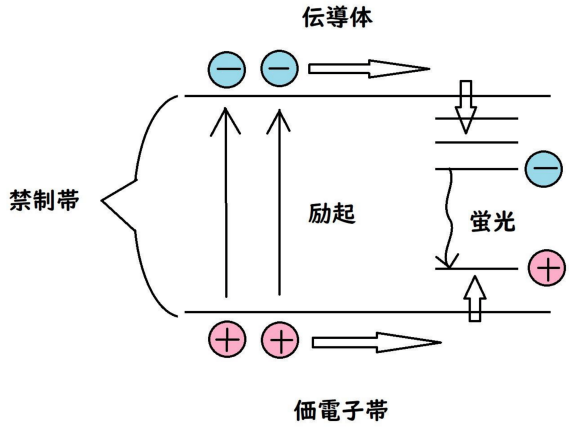


図 2.2: ガンマ線による励起でシンチレーションが起こる過程

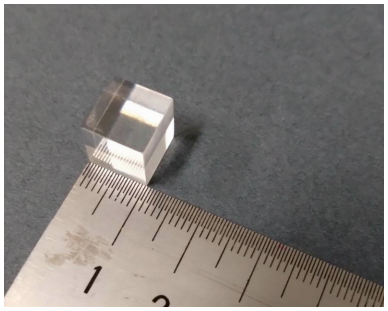


図 2.3: CsI(Tl) シンチレータの写真

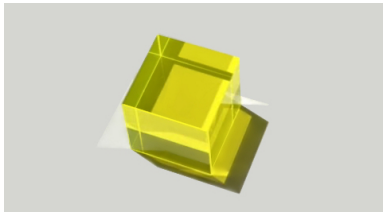


図 2.4: GAGG(Ce) シンチレータの写真 [1]

第3章　シンチレーション発光波形の取得

3.1　波形測定セットアップ

波形の測定に用いたセットアップの概念図を図 3.1 に示した。

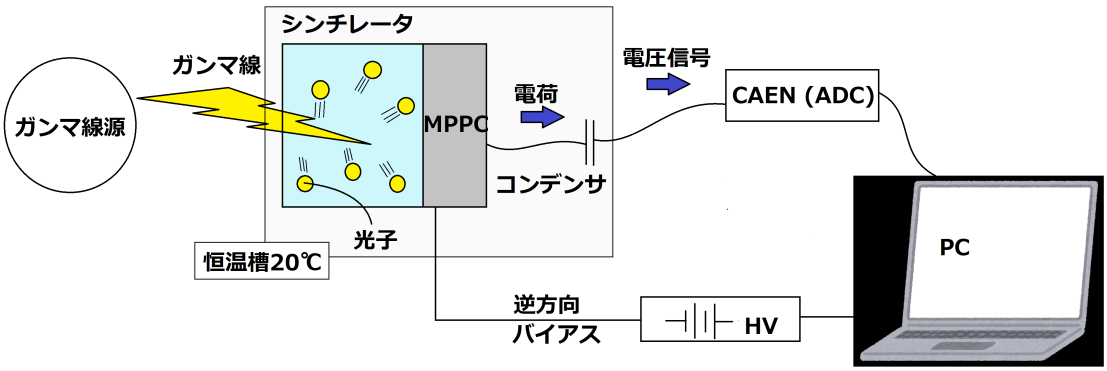


図 3.1: 波形測定セットアップ概念図

入射したガンマ線によってシンチレータにエネルギーが与えられるとそれに応じた可視光が発せられる。その可視光は直接、もしくはシンチレータ各面に貼ってある反射フィルムで反射された後光検出器 MPPC によって検出される。MPPC の各ピクセルは光子を受け取ると光電効果によって電子を出し、電子雪崩の原理で電子数を増幅させる。MPPC からの信号の高さは光子を捉えたピクセルの数に比例する [4]。この信号は ADC によってコンピュータの理解できるデジタルデータとして PC へと送られる。本研究で用いた ADC は CAEN（DT5720）で、１秒間に 2.5×10^8 回信号の高さ（電圧）のサンプルを取ることができ（250MS/s）、シンチレーション発光波形を記録することができる。

3.2　実験手順

シンチレータと MPPC を組み合わせてシンチレータ検出器とする際のセットアップを述べる。シンチレータの周りには ESR という可視光に対して反射率の優れたフィルムを巻く。これはシンチレータで発生した光子を MPPC にもれなく届けて、可視光の光子数が減ってポアソン分布の効果によりエネルギー分解能が悪化することを防ぐためである。シンチレータと MPPC の接着には光学グリスを用いる。このグリスにはシンチレータと MPPC の受光面との間の屈折率の急激な変化を抑え、全反射を防ぐ狙いがある。シンチレータと組み合わせた MPPC を回路に組み込み、図 3.2 のようにアルミの箱に詰めて恒温槽に入れる。本研究では恒温槽の温度はすべて+20.0 ℃とした。

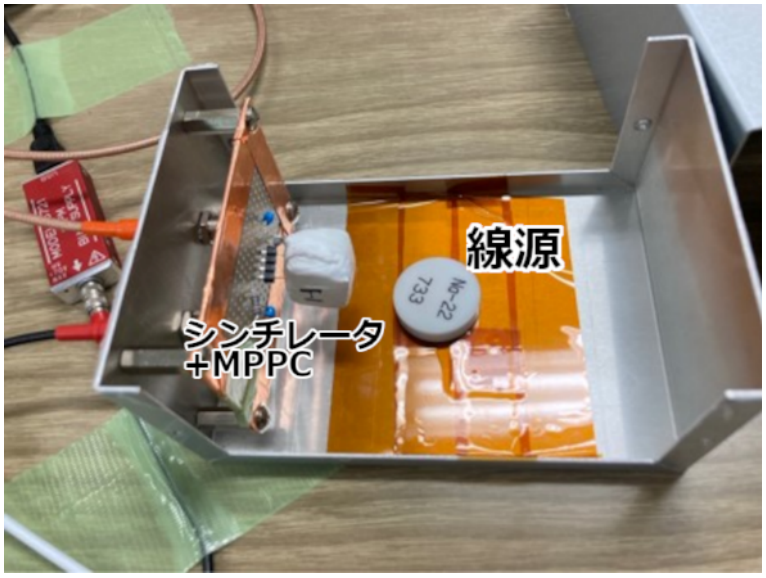


図 3.2: シンチレータと MPPC のセットされたアルミ箱

MPPC に逆バイアス電圧をかける。製造メーカーの浜松ホトニクスの推奨電圧 54.45V にした。ADC の電源を入れる。CAEN 用のソフトウェアである CoMPASS を用いてデータを取得する。

3.3 測定条件

実験の条件は以下のとおり。

- 10mm×10mm×10mm の CsI(Tl) 及び GAGG(Ce) シンチレータ
- 線源は ^{137}Cs・^{22}Na・^{241}Am・^{152}Eu
- 測定時の恒温槽温度+20.0 ℃
- MPPC の動作電圧 54.45V

3.4 取得された波形の概形

図 3.3 と図 3.4 にそれぞれのシンチレータで取得された波形の一例を示した。縦軸は底が e の対数での電気信号の高さであり、チャンネル単位である。横軸はイベント内でのサンプル番号であり、１つのサンプルが 4ns に相当する。両者のシンチレータで、総発光量は数万光子で同程度であるが、CsI に比べ GAGG のほうが減衰時間が短いため、波高が高くなり、到来光子のポアソン揺らぎが低減され波形がきれいに見えている。なお図 3.3・図 3.4 ともに 662keV のガンマ線を光電吸収したイベントである。

複数イベントの波形を平均すればポアソン統計によるゆらぎが小さくなり波形が見やすくなる。図 3.5 と図 3.6 にエネルギーの近いイベントの平均を示した。イベントのエネルギーは後述の 3.5 節で求めたエネルギー較正直線により計算している。この操作を行うプログラムを 6.1.3 に記した。 1.1.2 のとおり主に２つ

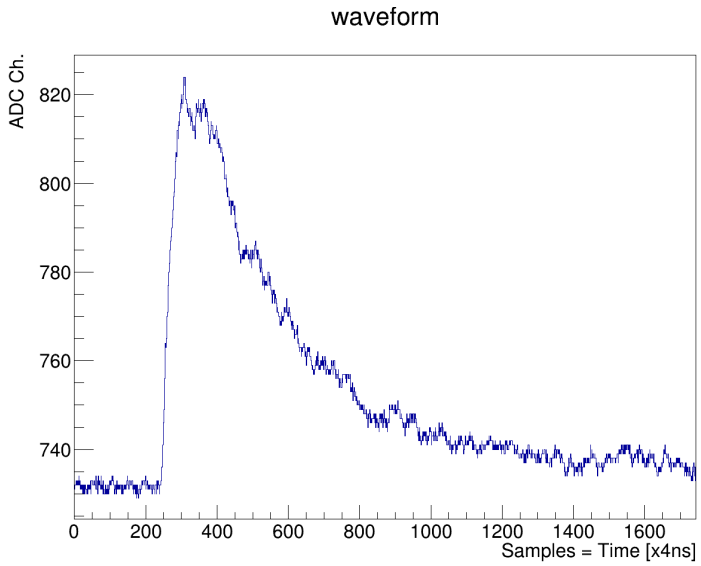


図 3.3: CsI シンチレータで取得された波形の一例

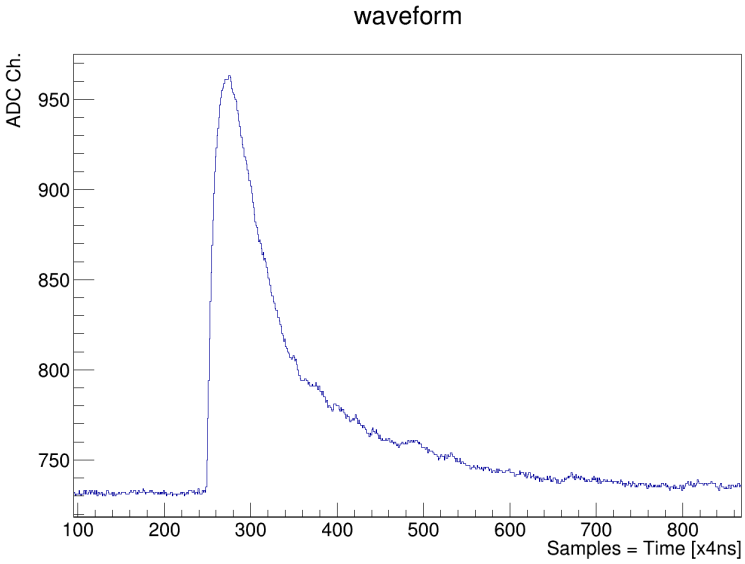


図 3.4: GAGG シンチレータで取得された波形の一例

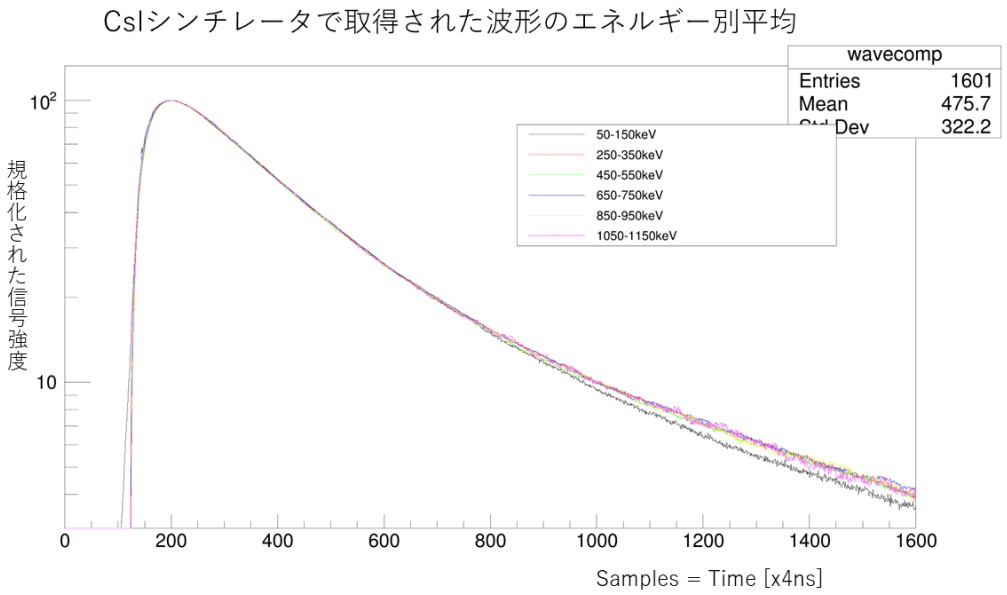


図 3.5: エネルギーごとに平均化した CsI シンチレータの発光波形

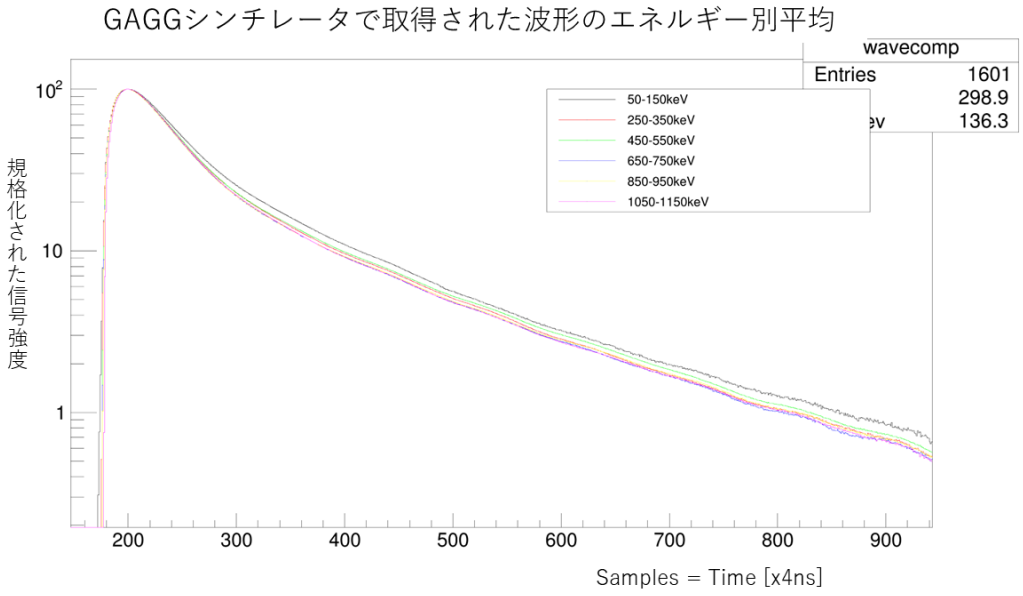


図 3.6: エネルギーごとに平均化した GAGG シンチレータの発光波形

の指数関数の足し合わせの形になっている。縦軸が対数であることに注意されたい。

3.5　検出されたガンマ線のエネルギーを波形から算出

3.5.1　波形からのエネルギースペクトル作成

取得された波形からエネルギーを算出するには、シンチレーション光によって信号が励起された部分を足し合わせる。この積分値に対してヒストグラムを作成するとエネルギースペクトルが得られる。この操作を行うプログラムを 6.1.1 に記した。これによって得られたのが図 3.7 である。

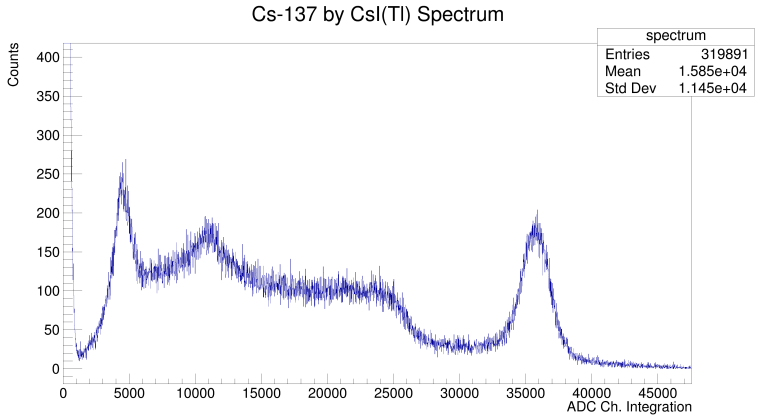


図 3.7: 波形の積分によって得られたスペクトル

この画像で見られるピークのうち、積分値 36000 付近にあるのが 662 keV ガンマ線の光電吸収によるもので、光電吸収ピークと呼ぶ。25000 以下にあるものはガンマ線がシンチレータ内の電子をコンプトン散乱していった後に脱出したため、エネルギーの一部のみがシンチレータに与えられたもので、コンプトン連続部と呼ぶ。式 (2.4) にある通りガンマ線のエネルギーによって上限が決定されるため、ある一定のエネルギー以上では観測されなくなり、このエネルギーをコンプトンエッジと呼ぶ。（今回は 25000 付近）横軸 12000 付近の盛り上がりは、後方散乱ピークと呼ばれる。これはシンチレータ外でコンプトン散乱をしてエネルギーを失ったガンマ線がシンチレータ内で光電吸収されることによるピークである。ガンマ線が十分遠方で散乱された場合、式 (2.3) の散乱角 $\theta_1 = \pi$ となり、式 (2.5) に近いエネルギーをもつガンマ線が多くなるからである。横軸 5000 ほどに見られるピークは、ガンマ線源 ^{137}Cs が崩壊して生じる $_{55}$Ba の特性 X 線である。

本研究ではガンマ線がシンチレータにすべてのエネルギーを与えて消失する光電吸収ピークを扱う。

3.5.2　エネルギー較正直線とエネルギー分解能

積分値の光電吸収ピークの中心値はエネルギーと線形に関係しており、複数の線源で取得した値を使って積分値とエネルギーを結びつける直線を作成する。これをエネルギー較正直線という。

光電吸収ピークの中心値と標準偏差を調べるため、これをガウス関数でフィットする。ガウス関数は式 (3.1) で定義する。

$$I\exp\{-\frac{(x-\mu)^2}{2\sigma^2}\} \tag{3.1}$$

プログラム 6.1.2 でフィットする。これによって図 3.7 における ^{137}Cs の 662 keV ガンマ線のピーク積分値 μ は 3.57×10^4、σ は 1.20×10^3 とわかる。

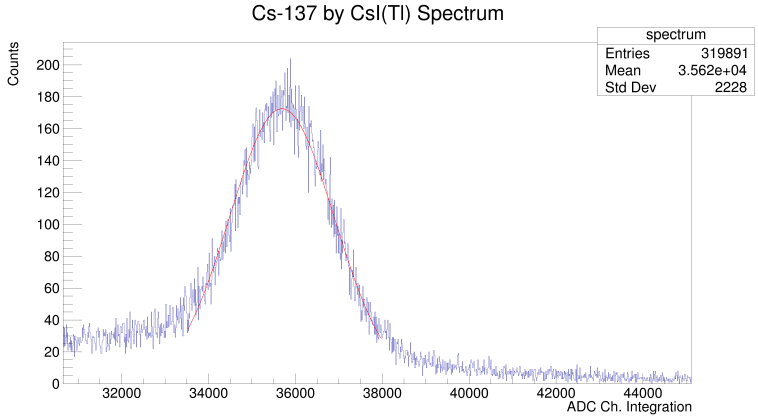


図 3.8: ガウス関数でフィットされた 662 keV 光電吸収ピーク

これと同様の操作を ^{241}Am の 59.5 keV ピーク、^{152}Eu の 122 keV および 344 keV ピーク、^{22}Na の 511 keV および 1275 keV ピークに対して行う。これによって表 3.1 を得た。このエネルギーとピーク積分値が直線

表 3.1: CsI について得られた各光電吸収ピークのチャンネル積分値と標準偏差

エネルギー [keV]	ピーク積分値 [$\times10^3$]	σ [$\times10^2$]
59.5	3.31	4.10
122	6.84	4.78
344	18.4	8.59
511	27.1	9.07
662	35.7	12.0
1275	63.9	14.7

になっているとしてフィットする。フィットされた直線を図 3.9 に示す。ただし図 3.9 中の a と b は式 (3.2) のものであり、チャンネル積分値とエネルギーを結びつける定数である。

$$ADC\ Ch.\ Integration = a\times Energy + b \tag{3.2}$$

図 3.9 中の各プロット点に付与されたパーセンテージはエネルギー分解能である。エネルギー分解能は、式 (3.3) で定義される。

$$Energy\ Resolution = \frac{FWHM_{Energy}}{Energy} \tag{3.3}$$

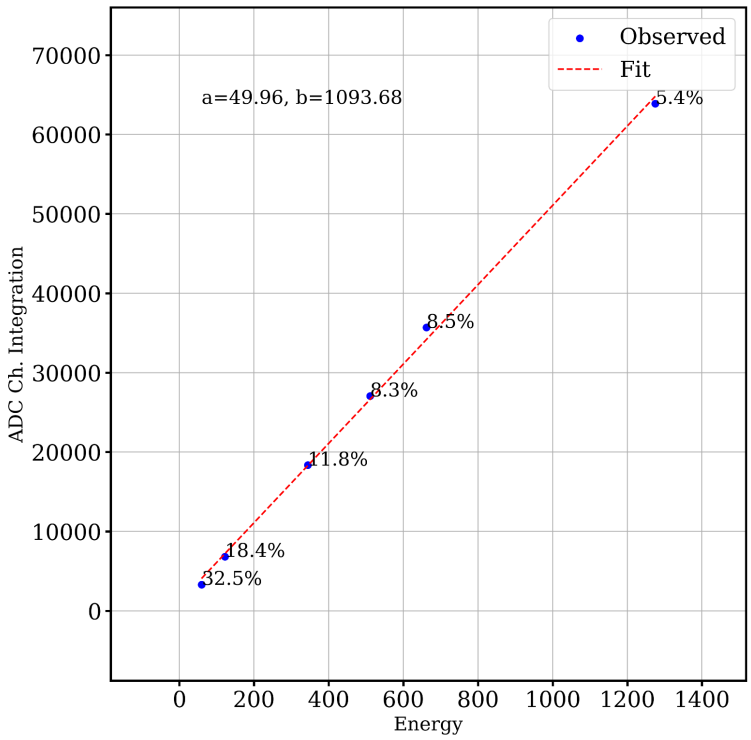


図 3.9: CsI(Tl) のエネルギー較正直線　各点に添えられている数値は、得られたエネルギー分解能を示す。

ここで $FWHM_{Energy}$ とは、検出されたエネルギーの半値全幅のことであり、$FWHM_{Energy} \simeq 2.35\sigma_{Energy}$ である。$Energy$ は線源から放射されるガンマ線のエネルギーとして知られているものである。ここで σ_{Energy} は表 3.1 の σ と式 (3.2) の a を用いて、$\sigma_{Energy} = \frac{\sigma}{a}$ と表せるので、結局式 (3.4) で計算できる。

$$Energy\ Resolution = \frac{2.35\sigma}{a \times Energy} \tag{3.4}$$

これより近くに他のピークがある場合、分解できない。このエネルギー分解能のパーセンテージが小さいほどエネルギーの近い異なるピーク同士を区別する能力が高いことになる。エネルギー分解能は放射線検出器の最も重要な性能のひとつである。

3.6　エネルギー分解能を良くするための適切な信号の積分時間

3.6.1　ノイズによるエネルギー分解能の悪化

一般論として、積分時間がながければ長いほどシンチレータの発光を漏れなく捉えることができ、MPPC 光検出器で発生した電子数のポアソン分布の性質によりエネルギー分解能が向上する。しかしながらエネルギーが小さいイベントは減衰によって発光信号が測定系のオフセットのゆらぎに飲まれてしまい、それ以上積分するのはあまり意味がないと考えられる。むしろ、エネルギー分解能の悪化の原因となりうる。

図 3.10 はシンチレータによって信号が励起されていないときのオフセットを 500ns 間（125samples）積分した値のヒストグラムである。このヒストグラムは正規分布を示しているが、FWHM が 100 ほどあり、

これは 3.4 節で求めたエネルギー較正直線の傾きを考慮すると 3 keV から 4 keV ほどとなる。したがって冗長な積分時間はエネルギー分解能の悪化に寄与しうると言える。

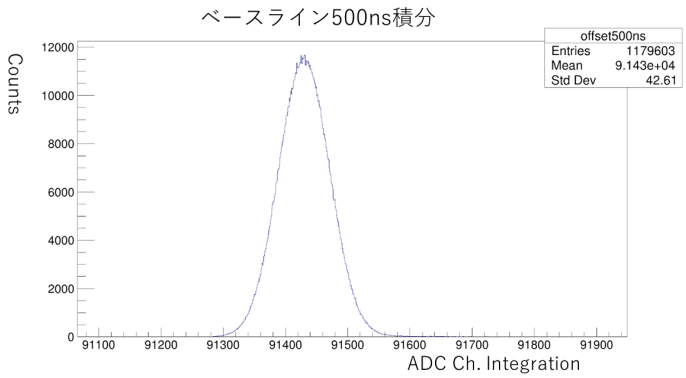


図 3.10: オフセットの 500ns 積分値のヒストグラム

3.6.2 GAGG: 積分時間によるエネルギー分解能の傾向

図 3.11 と図 3.12 に GAGG シンチレータで取得したエネルギーの異なる 2 つの波形を示した。それぞれイベントのエネルギーは 59.5 keV・1274 keV である。 2 つのグラフを見比べると、意味のあるサンプルもしくは時間の長さに差があることがわかる。 3.6.1 節の考察を踏まえると、エネルギー分解能を最良にする

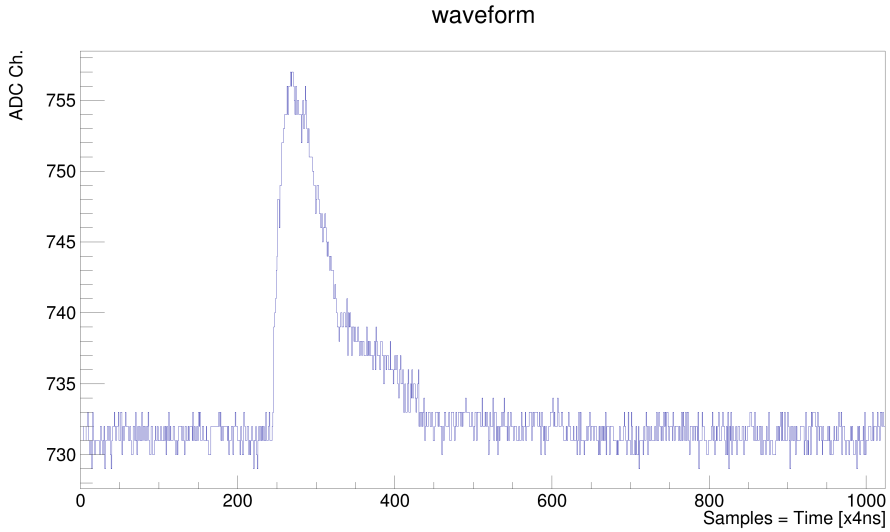


図 3.11: GAGG シンチレータの 59.9 keV イベント発光波形

積分時間は低エネルギーで短く、高エネルギーで長いという可能性がある。

信号の積分時間を 500ns・1μs・2μs・10μs と変えて 3.5 節の方法で解析を行った。解析結果を表 3.2 から表 3.5 に記した。 表 3.2 から表 3.5 までのエネルギー分解能を図 3.13 にまとめた。図 3.13 より全体的に

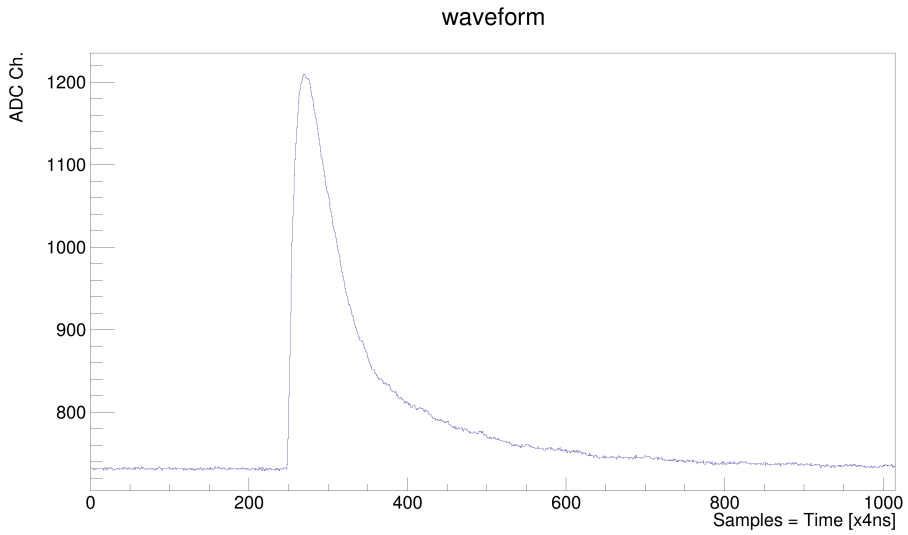


図 3.12: GAGG シンチレータの 1274 keV イベント発光波形

表 3.2: GAGG 発光信号積分時間 500ns での解析結果

エネルギー [keV]	ピーク積分値 [$\times 10^3$]	σ [$\times 10^2$]	エネルギー分解能 [%]
59.5	1.23	2.36	35.7
122	2.99	3.90	28.8
344	9.20	9.58	25.1
511	13.8	10.4	18.2
662	17.7	10.9	14.8
1275	33.0	15.2	10.7

最良の積分時間は 2μs 付近にあると予想できる。短い積分時間では信号をもれなく積分することができず、あまりに長い積分時間ではノイズの影響を受けるからである。低エネルギーのピークほど短い積分時間でのエネルギー分解能の悪化が小さく、高エネルギーのピークほど長い積分時間でのエネルギー分解能の悪化が小さい。したがって適切な積分時間はエネルギーが大きいほど長く、エネルギーが小さいほど短いと予想できる。

表 3.3: GAGG 発光信号積分時間 1μs での解析結果

エネルギー [keV]	ピーク積分値 [$\times 10^3$]	σ [$\times 10^2$]	エネルギー分解能 [%]
59.5	1.57	2.89	35.4
122	3.82	4.10	24.5
344	11.5	9.43	20.0
511	17.1	10.8	15.4
662	21.9	11.1	12.3
1275	40.8	14.6	8.3

表 3.4: GAGG 発光信号積分時間 2μs での解析結果

エネルギー [keV]	ピーク積分値 [$\times 10^3$]	σ [$\times 10^2$]	エネルギー分解能 [%]
59.5	1.79	3.21	35.1
122	4.31	4.00	21.3
344	12.9	8.77	16.5
511	19.2	9.06	11.5
662	24.6	9.89	9.7
1275	45.8	12.4	6.3

3.6.3 GAGG: 信号長さのエネルギー依存性

波形の積分時間をエネルギー分解能が最良になるものにするため、信号が測定系のベースラインに復帰するまでの信号の長さを評価したい。ベースラインのノイズによるゆらぎを知るため、各イベントのトリガー時刻より前のベースラインの値から、その標準偏差 σ_{noise} を計算する。信号の高さが $ADC\ Ch. = -\sigma_{\mathrm{noise}}$ となったとき信号が終了したと判定することにする。ここでは、「トリガー時刻」から「信号終了」までの時間を信号の長さと定義する。GAGG シンチレータを用いて取得した各イベントのエネルギーと信号の長さを図 3.14 に散布図として示した。線源は ^{22}Na であり、縦軸は前述の方法で判定した信号の長さ、横軸は積分時間 1μs で計算されたエネルギーである。これを出力するプログラムを 6.1.4 節に載せた。これによるとやはりエネルギーが大きくなるにつれて、ベースラインに復帰するまでの信号の長さが大きくなっている。

それぞれのエネルギーを代表する信号の長さを決定したい。それができればスペクトルを作成する際、まず 1μs 積分して仮のエネルギーを出し、仮のエネルギーに従ってエネルギー分解能を最良にする積分時間を決定できる。

図 3.14 の横軸の各 bin 中の *Signal Length* [ns] の平均値を求めた。これが図 3.15 の青線部分である。この青線の中心部分が各 bin の平均値であり、長さが標準誤差である。ただし標準誤差は各 bin の標準偏差をその bin 中のイベント数の平方根で除したものである。511 keV 以下では ^{22}Na の 511 keV 光電吸収ピークとそれに対するコンプトン連続部によってイベント数が多く標準誤差が小さくなっているが、それより大きいエネルギーのイベントは崩壊確率の低い 1275 keV ガンマ線のコンプトン連続部と光電吸収ピークしかないのでイベント数が少なく、標準誤差が大きくなっている。

表 3.5: GAGG 発光信号積分時間 10μs での解析結果

エネルギー [keV]	ピーク積分値 [$\times 10^3$]	σ [$\times 10^2$]	エネルギー分解能 [%]
59.5	1.86	6.10	65.6
122	4.35	7.31	38.3
344	13.0	10.8	20.1
511	19.5	11.1	13.9
662	25.0	11.5	11.1
1275	46.6	13.6	6.8

各エネルギーに対する信号の長さの代表値によって積分し直すプログラムを作成する際、図 3.15 の青線部分の値を記録しておいてその値を使うより、連続でなめらかな関数でフィットして、その関数から返ってくる値を使うほうが合理的で簡単である。また、青線のように隣接したエネルギーに対して大きく異なる積分時間を返すとピークが広がってエネルギー分解能の悪化につながる可能性もある。そこで青線を滑らかな関数でフィットした結果が、図 3.15 の赤線である。ただし関数は式 (3.5) で定義した。

$$Signal\ length = p_0 \times (Energy - p_1)^{p_2} + p_3 \tag{3.5}$$

参考として、フィットされた変数は表 3.6 となった。また、図を作成するプログラムを 6.1.5 に載せた。

表 3.6: フィットされた式 (3.5) の変数

p_0	p_1	p_2	p_3
6.67×10^4	-12.3	1.01×10^{-2}	-6.81×10^4

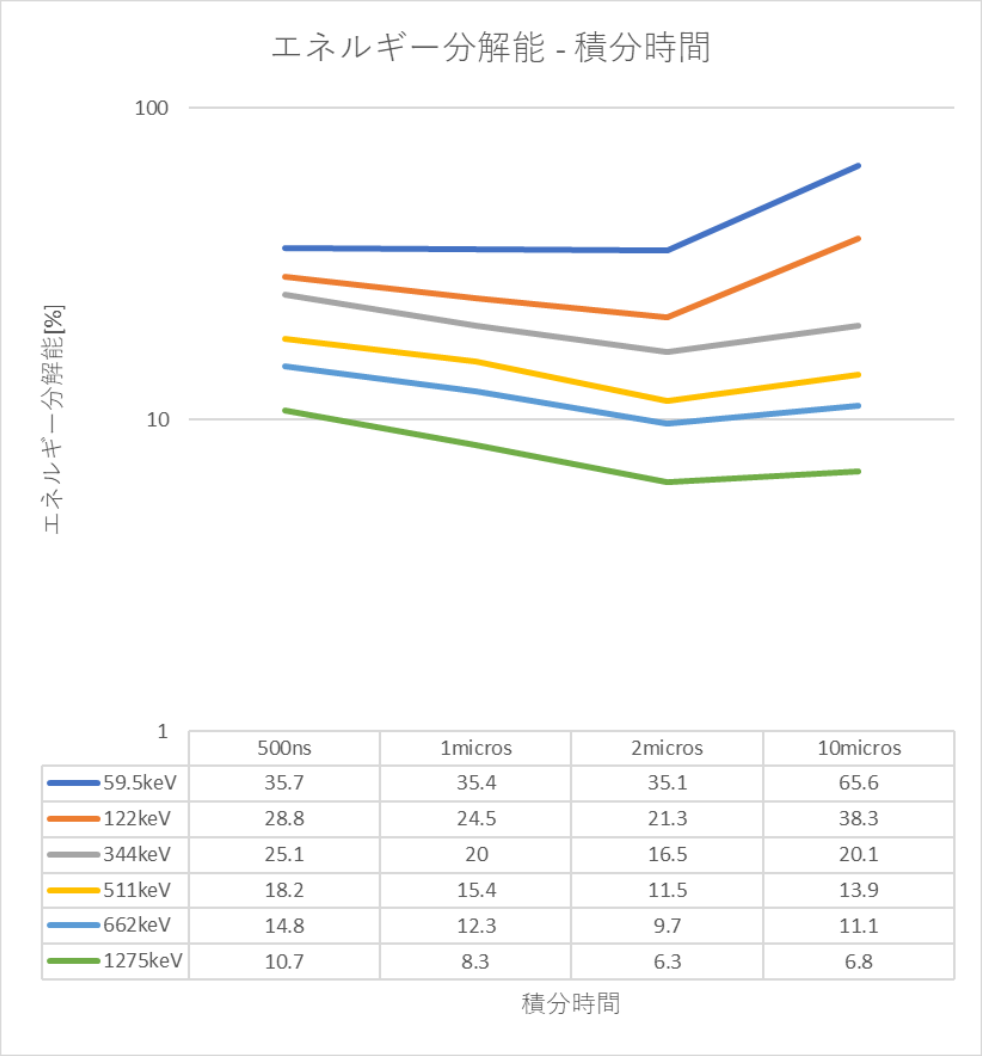
3.6.4 GAGG: 可変積分時間を用いたときのエネルギー分解能

6.1.4 節で求めた積分時間を用いてスペクトルを作成することを考える。手順としては、各イベントに対して以下の操作を行う。

- 1μs 間積分して仮のエネルギーを求める。
- 式 (3.5) に仮のエネルギーを代入し、積分時間を決定する。
- 決定された積分時間だけ信号を再度積分する。
- 積分で得た値をヒストグラム（エネルギースペクトル）に記録する。

この操作を行うプログラムを 6.1.6 節に載せた。

以上の操作によって表 3.7 の結果を得た。表 3.7 と 3.6.2 の結果を図 3.16 にまとめた。結果として 511 keV で 2μs 積分にわずかに劣ったほかは、すべてのエネルギーで最良のエネルギー分解能を得ることができた。



	500ns	1micros	2micros	10micros
59.5keV	35.7	35.4	35.1	65.6
122keV	28.8	24.5	21.3	38.3
344keV	25.1	20	16.5	20.1
511keV	18.2	15.4	11.5	13.9
662keV	14.8	12.3	9.7	11.1
1275keV	10.7	8.3	6.3	6.8

図 3.13: GAGG シンチレータで検出されたガンマ線の、積分時間によるエネルギー分解能の変化

3.6.5 CsI シンチレータでの実験結果

CsI シンチレータで取得したデータでも GAGG シンチレータ同様に 3.6.3 節・3.6.4 節の操作を行おうとした。図 3.17 は CsI シンチレータで取得した発光信号の長さである。300 keV 以下で GAGG シンチレータと同じように信号がエネルギーに従って長くなっていく様子がわかる。しかしながら、CsI シンチレータの発光時間が実験前に想定していたより長く、今回の取得時間では十分でなかったので信号の長さ（プログラム上で）6μs で頭打ちになってしまった。従って今回は CsI に関しては適切な積分時間のエネルギー依存性については考えられなかった。

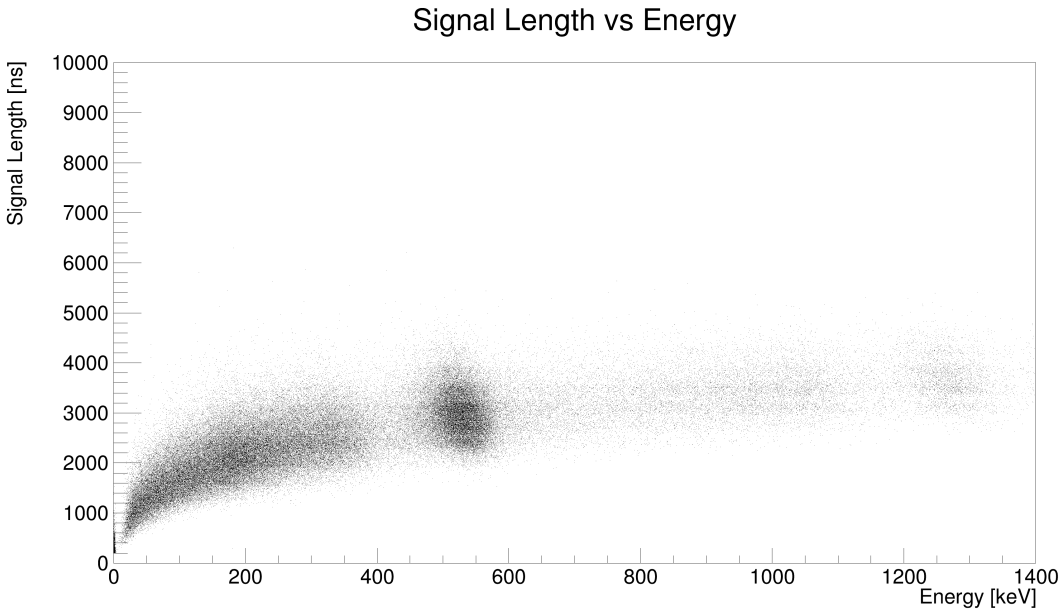


図 3.14: GAGG でのエネルギーに対する発光信号の長さ横軸は 1μs 積分で求めた

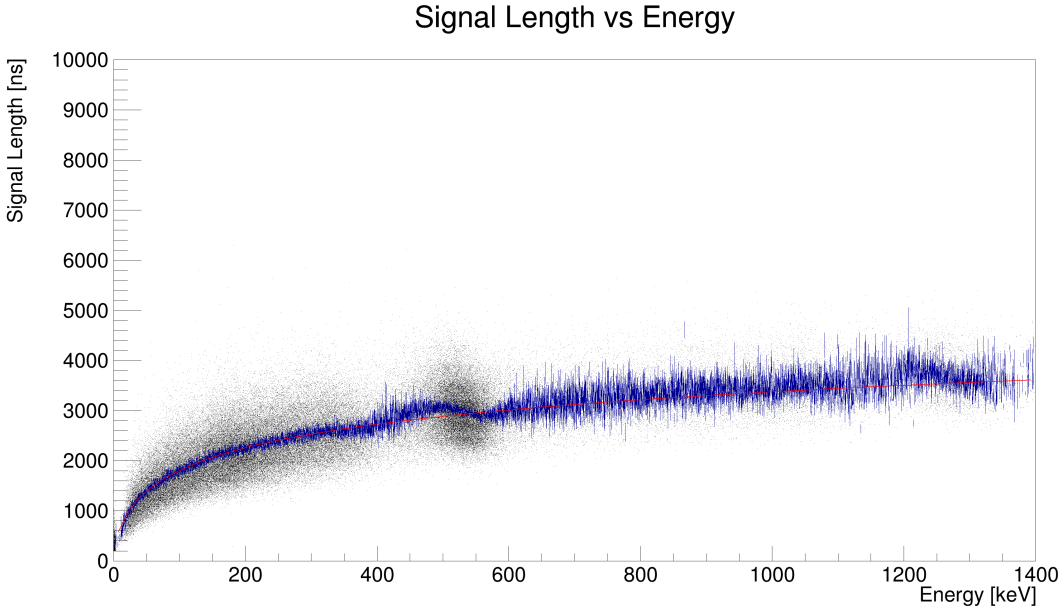


図 3.15: GAGG 発光信号のエネルギーごとの平均長さ
青線が各 X 値での Y 値の平均と標準誤差、赤線が青線をフィットした関数

表 3.7: GAGG 発光信号 可変積分時間での解析結果

エネルギー [keV]	ピーク積分値 [$\times 10^3$]	σ [$\times 10^2$]	エネルギー分解能 [%]
59.5	1.67	3.28	34.4
122	4.28	4.16	21.3
344	13.1	8.11	14.7
511	19.7	9.51	11.6
662	25.4	9.91	9.3
1275	47.5	11.8	5.8

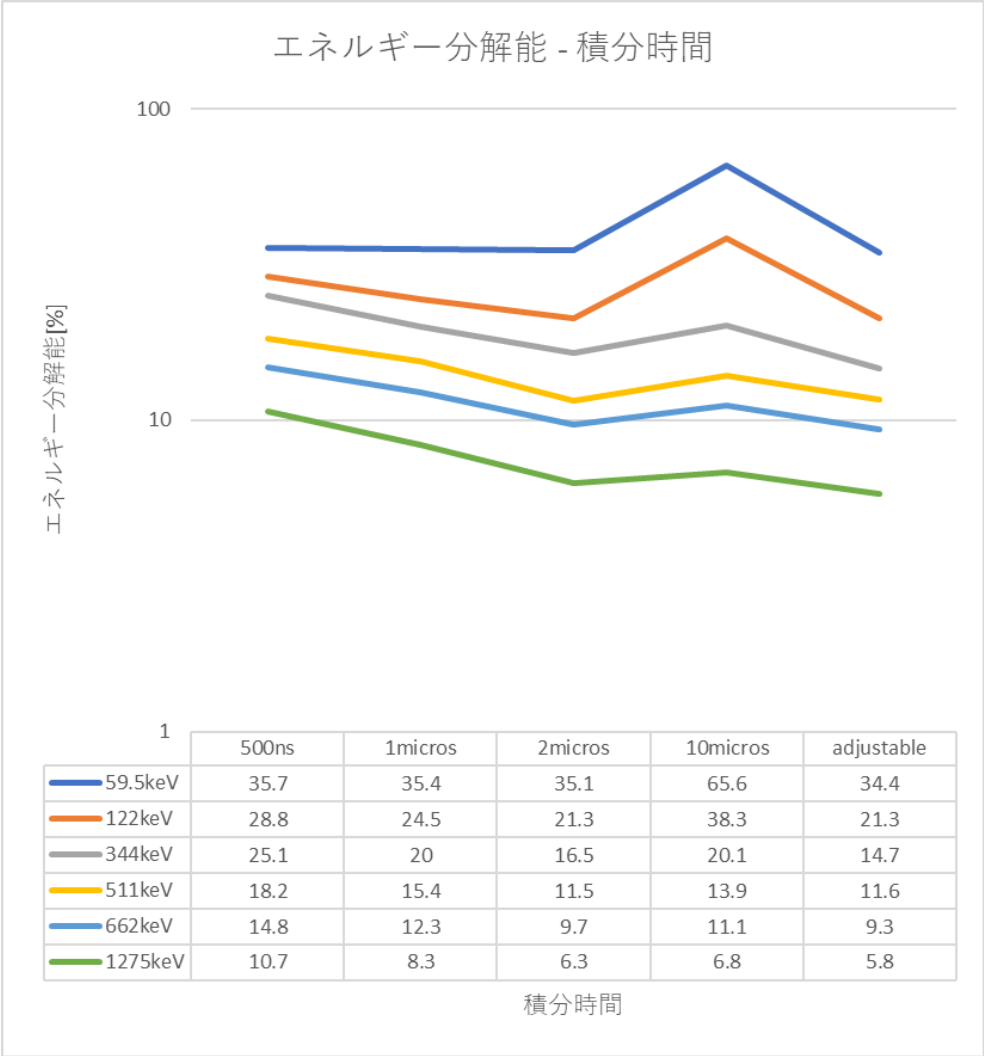


図 3.16: 積分時間を固定した場合と可変にした場合のエネルギー分解能

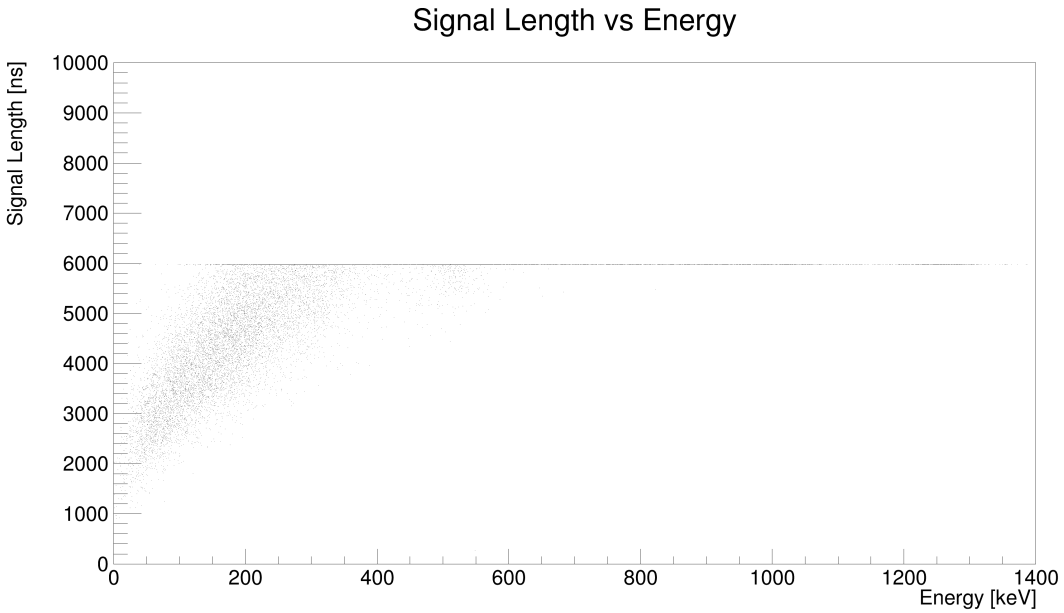


図 3.17: CsI でのエネルギーに対する発光信号の長さ

第4章　波形情報を用いたエネルギー分解能の向上

3.6.4 節では、波形の積分時間を可変にして、エネルギー分解能を向上することができることを示した。これは測定系の整形増幅器の shaping time というパラメータを調整することに対応する。この章では、本測定ではシンチレーション光の波形を保存していることから、波形情報そのもの（波形の長さ、減衰の速い成分と遅い成分の強度比）に着目してエネルギー分解能を向上できないかを実験した。

入射したガンマ線によってシンチレータに与えられたエネルギーが同一のイベント同士であっても、検出されるエネルギーが必ず同じとは限らない。例えば図 3.8 に見られるピークに含まれるイベントはほぼ全て ^{137}Cs の崩壊による 662 keV のガンマ線を光電吸収したものである。一般論として、ある核種の同一の崩壊モードにより発せられる放射線のエネルギーは厳密にすべて同じである。光電吸収ではガンマ線光子の全エネルギーを受け取るので、シンチレータに与えられるエネルギーはすべてのイベントで同一である。しかしながら実際は図 3.8 で見られるように光電吸収ピークは幅を持っている。検出されるイベントのエネルギーは、シンチレーションで放射される可視光光子のうちいくつが光検出器で検出できるかというポアソン統計の効果などによって揺らぐ。あるイベントの検出値が真のエネルギーからずれたとき、それに相関してずれるような物理量があればその物理量を用いてエネルギーのズレを補正できる可能性がある。以下ではイベントのエネルギー値としては 1μs の積分で求めたものを扱う。

4.1　GAGG シンチレータの発光信号長さを用いたエネルギーの補正

4.1.1　補正係数のエネルギー依存性を考慮しない場合

図 4.1 は図 3.15 から青線（各エネルギーでの信号長さの平均値と標準誤差）を取り除いたものである。511 keV の光電吸収ピークに注目する。前述の通りこのピークに含まれるイベントの真のエネルギーはすべて 511 keV で同一である。このピークの中でもエネルギーが大きい方にずれた（横軸で右側にずれた）イベントは信号が短くなる（縦軸下側にずれる）傾向があり、逆にエネルギーが小さい方にずれた（横軸で左側にずれた）イベントは信号が長くなる（縦軸上側にずれる）傾向があることが見て取れる。この傾向を用いて 511 keV の光電吸収ピークを補正し、エネルギー分解能が向上できないか調べる。6.1.7 節のプログラムで `double_t angle = 0.015` とする。これは 1μs の固定された積分時間で求めた仮のエネルギー *Energy* をもつイベントの信号の長さ *Signal Length* が、*Energy* を式 (3.5) に代入して得られる値よりどれだけ大きいか（小さいか）によってエネルギーに補正をかけるものである。補正係数 `double_t angle = 0.015` は光電吸収イベントの集団が横軸に対して垂直に見えるように選んだ。補正の結果が図 4.2 である。

同様の方式の処理を他のエネルギーの光電吸収ピークにも施す。ただし補正係数 `double_t angle` はすべて同一 (0.015) とし、赤線（信号の長さをフィットした関数）は図 3.15 で求めたものをそのまま用いた。これによって ^{137}Cs の 662 keV ピークは図 4.3 のように、^{152}Eu の 122 keV から 344 keV の諸ピークは図 4.4 のように補正された。 344 keV 以上のピークではよく補正されているように見えるが、122 keV ピークでは

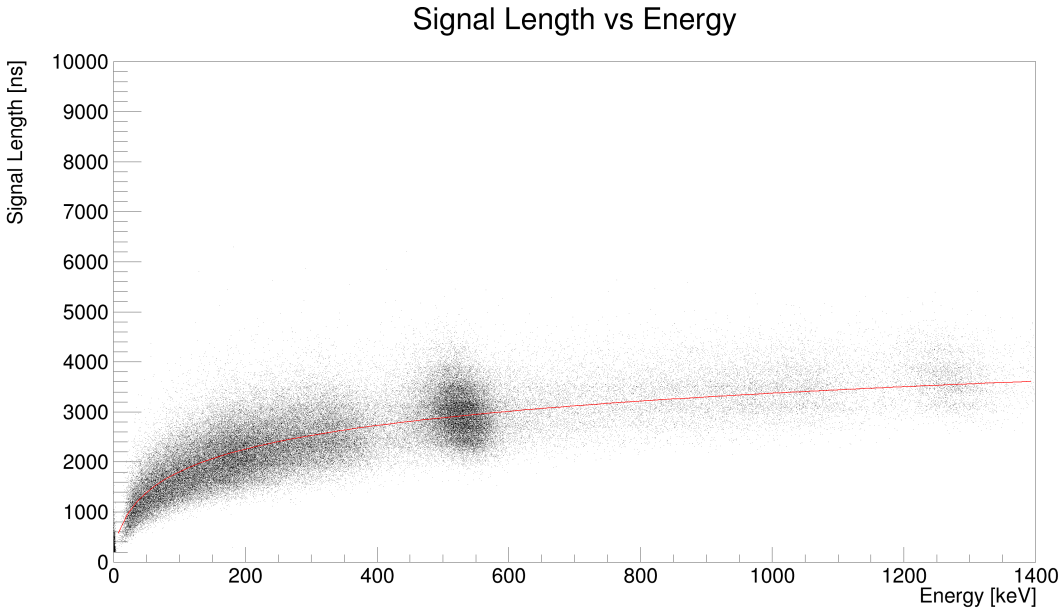


図 4.1: GAGG シンチレータで捉えられた ^{22}Na からのガンマ線の発光信号の長さ

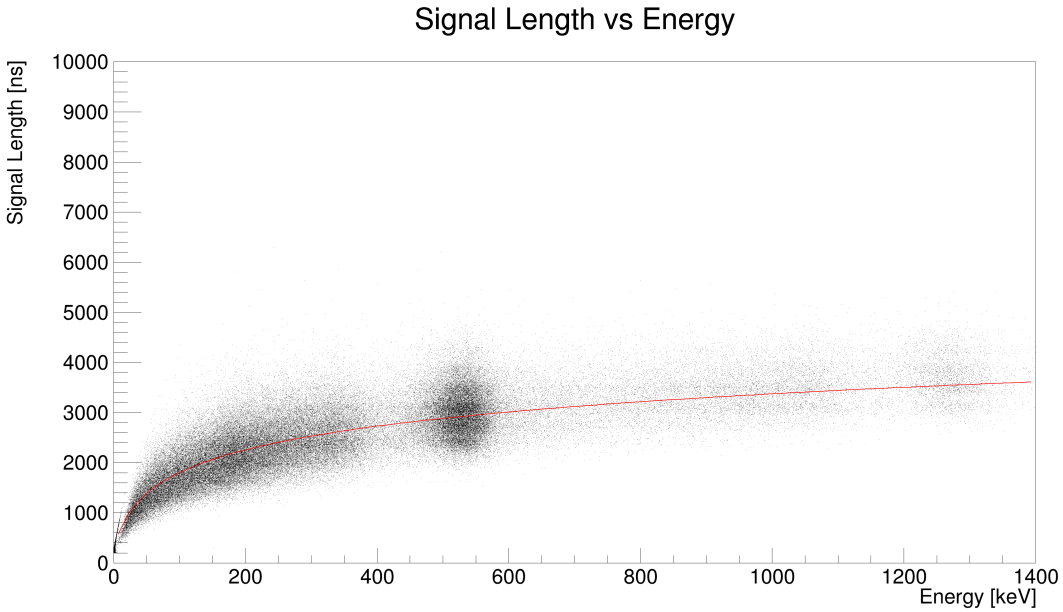


図 4.2: 信号の長さによって補正された、GAGG で取得された ^{22}Na からのガンマ線

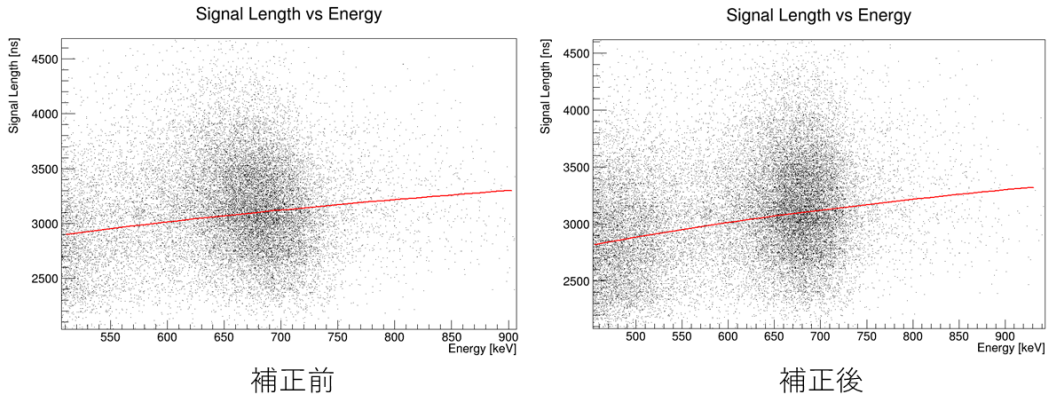


図 4.3: 信号の長さによって補正された、GAGG で取得された 662 keV のガンマ線

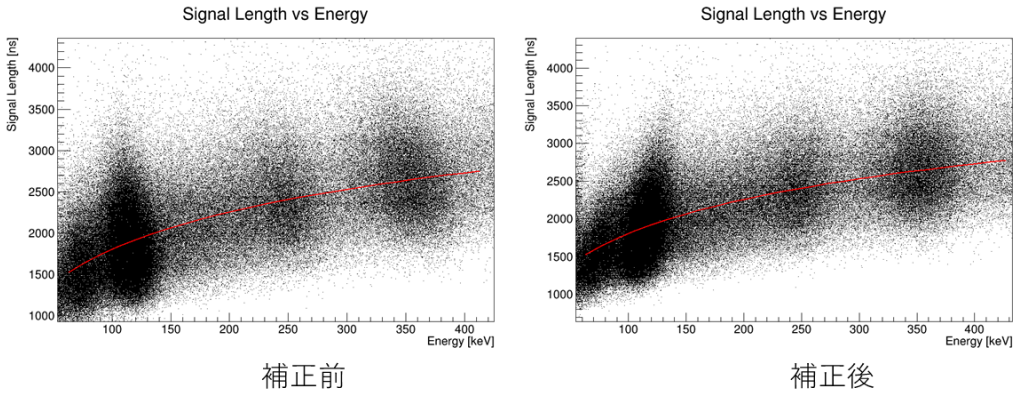


図 4.4: 信号の長さによって補正された、GAGG で取得された ^{152}Eu からのガンマ線

補正が過度であるようにみえる。これは補正係数のエネルギー依存性を考慮するべきであることを示唆している。^{241}Am の 59.5 keV の光電吸収ピークは図 4.5 のように、ADC の閾値付近のためイベント集団の形が他と違ったので補正の対象外とした。

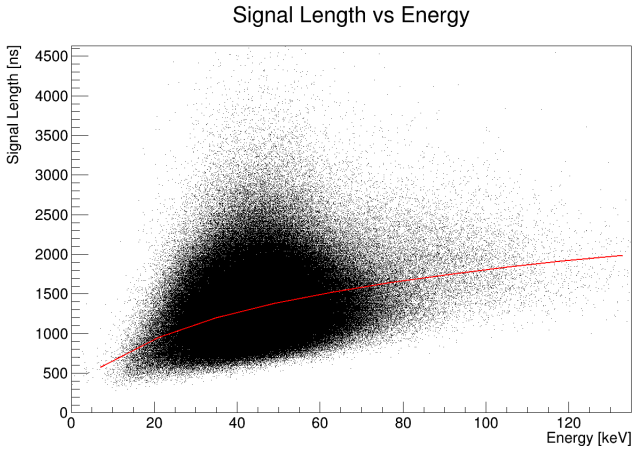


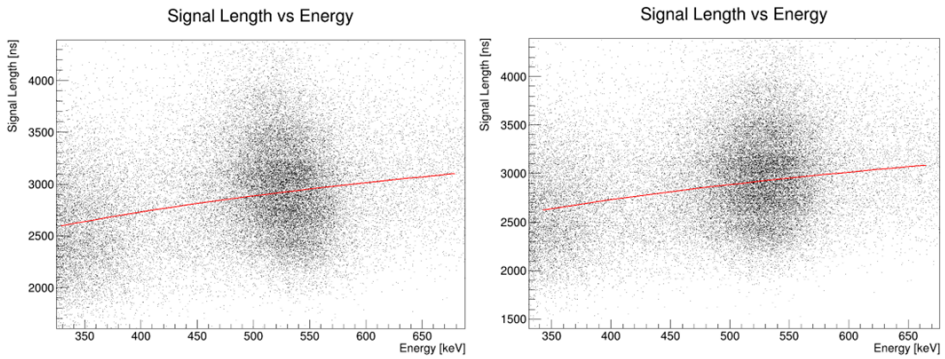
図 4.5: ^{241}Am の 59.5*keV* ガンマ線の信号の長さ

4.1.2 補正係数のエネルギー依存性を考慮する場合

122 keV では補正が過剰だったことから、補正係数がエネルギーに依存すると考えた。エネルギーが大きいほど補正係数が大きく、小さい時補正係数が小さいと仮定して、補正係数が信号の長さ自体に比例すると予想した。511 keV での補正係数が 4.1.1 節と同じく 0.015 となるようにすると、補正係数 $Angle(Energy)$ は式 (3.5) の関数を $Signal\ Length(Enregy)$ として、式 (4.1) で表される。

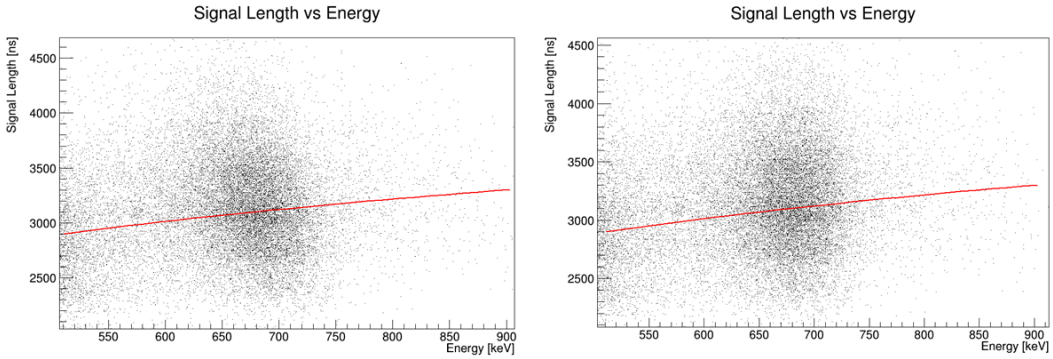
$$Angle(Energy) = 0.015 \times \frac{Signal\ Length(Energy)}{Signal\ Length(511\mathrm{keV})} \tag{4.1}$$

これを用いて補正を行うと、^{22}Na の 511 keV ピークは図 4.6、^{137}Cs の 662 keV ピークは図 4.7、^{152}Eu の 122 keV から 344 keV の諸ピークは図 4.8 のように補正された。 ^{152}Eu の 122 keV ピークの過補正が 4.1.1 節に比べると小さくなったが、それでもまだ過補正気味である。従って補正係数のエネルギー依存性に関してはさらなる考察が必要である。



補正前　　補正後

図 4.6: 信号の長さによってエネルギー依存性を考慮して補正された、GAGG で取得された 511 keV のガンマ線



補正前　　補正後

図 4.7: 信号の長さによってエネルギー依存性を考慮して補正された、GAGG で取得された 662 keV のガンマ線

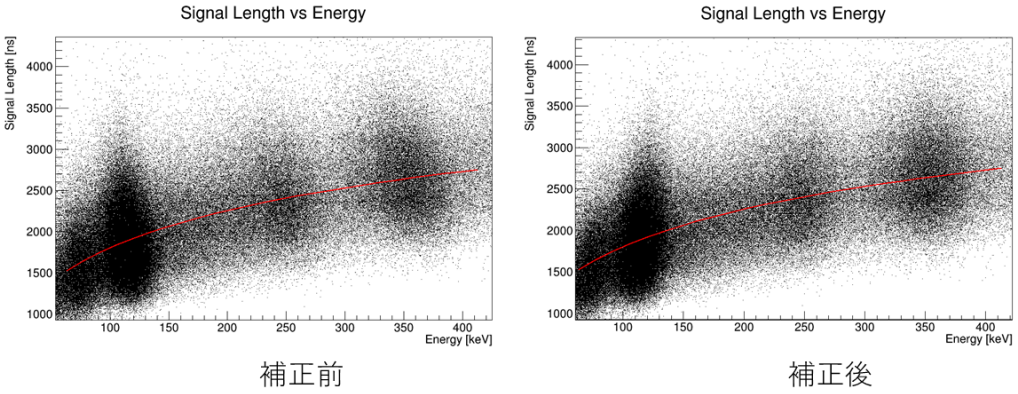


図 4.8: 信号の長さによってエネルギー依存性を考慮して補正された、GAGG で取得された ^{152}Eu からのガンマ線

4.1.3 補正されたエネルギーの分解能

補正前のエネルギー分解能は表 3.3 のとおりである。補正係数のエネルギー依存性を考慮しなかった場合のエネルギー分解能は表 4.1 のとおりである。ただし補正を施したものに関してはすでにエネルギーに直してあるものをフィットしている。補正係数のエネルギー依存性を考慮した場合のエネルギー分解能は表

表 4.1: エネルギー依存性を考慮しない補正を施した GAGG でのエネルギー分解能

エネルギー [keV]	$mean_{Energy}$ [keV]	σ_{Energy} [keV]	エネルギー分解能 [%]
122	114	13.1	27.0
344	351	28.23	18.9
511	526	32.5	14.5
662	677	32.3	11.5
1275	1267	39.6	7.3

4.2 のとおりである。表 3.3、表 4.1 および表 4.2 をまとめて表 4.3 に示した。

122 keV では予想通り過補正によってエネルギー分解能が悪化している。344 keV 以上ではすべて補正によりエネルギー分解能が良くなっている。122 keV の次に低エネルギーである 344keV では補正係数のエネルギー依存性を考慮したことによってエネルギー分解能が上昇している。511 keV では式 (4.1) よりどちらの補正も同じエネルギー分解能になっている。662 keV ではエネルギー依存性を考慮しないほうがエネルギー分解能が良かった。これは低エネルギーでは適切な補正係数が小さくなるのに対して、高エネルギーで

表 4.2: エネルギー依存性を考慮した補正を施した GAGG でのエネルギー分解能

エネルギー [keV]	$mean_{Energy}$ [keV]	σ_{Energy} [keV]	エネルギー分解能 [%]
122	115	12.3	25.2
344	353	26.4	17.6
511	527	32.5	14.5
662	675	34.6	12.3
1275	1267	39.3	7.3

表 4.3: 信号長さによる補正前後のエネルギー分解能

エネルギー [keV]	補正前 [%]	エネルギーに依存しない補正後 [%]	エネルギー依存性を考慮した補正後 [%]
122	24.5	27.0	25.2
344	20.0	18.9	17.6
511	15.4	14.5	14.5
662	12.3	11.5	12.3
1275	8.3	7.3	7.3

補正係数が大きくなるわけではないことを示唆している。

なお、エネルギー分解能が良くなった際のスペクトルの例として図 4.9 に補正前のスペクトルと、エネルギー依存性を考慮しない補正後のスペクトルを 511 keV で重ね書きした。青線が補正前、赤線が補正後である。赤線のほうがわずかに鋭くなっている。ピークチャンネル付近で赤線が大きくなっていることに注目するとわかりやすい。

4.1.4 信号の長さによるエネルギー補正の問題点

信号の長さとエネルギーの関係は、横軸のエネルギーを出す際の積分時間を十分長くすると図 4.1 のような、補正可能なものではなくなる。図 4.10 のようにエネルギーのずれと信号の長さのずれに相関が見られなくなる。また、積分時間を適切にしたもののほうが積分時間 1μs で得られたエネルギーを信号の長さによって補正するよりもエネルギー分解能が良い。

前述のことより、短い積分時間のときと長い積分時間のときとではエネルギーのずれの振る舞いに差異が生じることがわかる。これはイベントのエネルギーがずれるとき、発光波形の減衰の速い成分と減衰の遅い成分の振る舞いに違いがあるということを示唆していると考え、次節のような解析を行った。

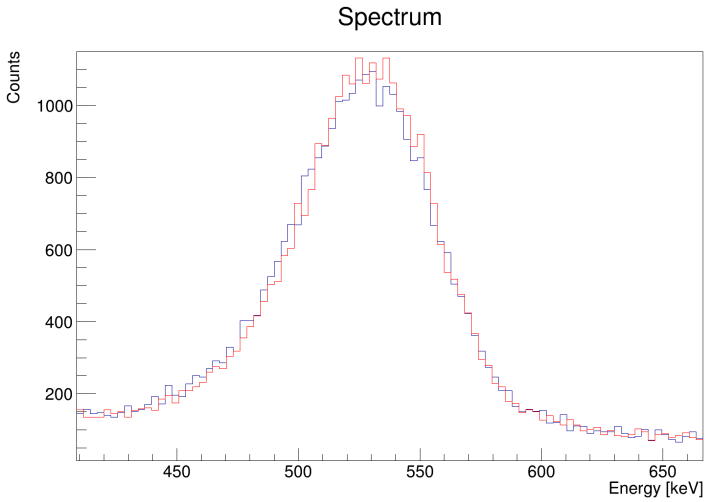


図 4.9: エネルギー依存性を考慮しない補正前後のエネルギースペクトル, 511 keV

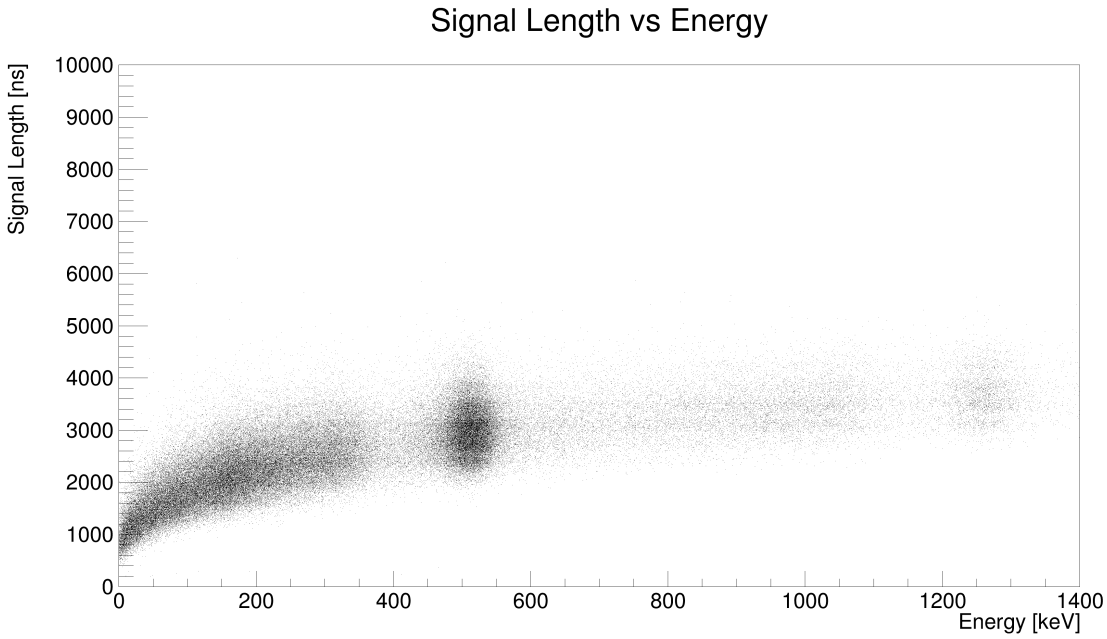


図 4.10: 横軸積分時間が 3μs のときの信号の長さとエネルギーの相関

4.2 発光波形情報の評価

発光波形を特徴づけるようなパラメータを設定したい。図 3.5 や図 3.6 に見られるように、CsI(Tl) シンチレータと GAGG(Ce) シンチレータの発光波形は減衰の速い指数関数と減衰の遅い指数関数の足し合わせからなっている。それぞれのシンチレータで、速い成分が支配的な部分の積分を I_{fast}、遅い成分が支配的な部分の積分を I_{slow} とあらわす。今回定義する値を、便宜上 $Yvalue$ と呼ぶことにすると、$Yvalue$ は式 (4.2) のように表される。

$$Yvalue = \frac{I_{\mathrm{slow}}}{I_{\mathrm{fast}}} \tag{4.2}$$

$Yvalue$ が大きいことは遅い成分の寄与が比較的大きいこと、$Yvalue$ が小さいことは速い成分の寄与が比較的大きいことを指す。

4.2.1 GAGG: 波形の評価

以下の GAGG(Tl) に関する部分では、エネルギー値として 3μs 積分で算出したものを用いる。理由は、図 3.15 から 511 keV 付近でのエネルギー分解能を最良にする信号の積分時間は約 3μs であると読み取れるからである。GAGG(Tl) の発光波形において、I_{fast} の積分範囲を図 3.6 の 200-300 Samples とする。I_{slow} の積分範囲を図 3.6 の 300-950 Samples とする。GAGG(Ce) シンチレータで ^{22}Na からのガンマ線を検出したときの、Yvalue とエネルギー値の散布図を図 4.11 に示す。この散布図を作るプログラムを 6.1.8 節に載せた。図 4.11 の yvalue に対して平均値と標準誤差をプロットすると図 4.12 になる。これによると $Yvalue$ の

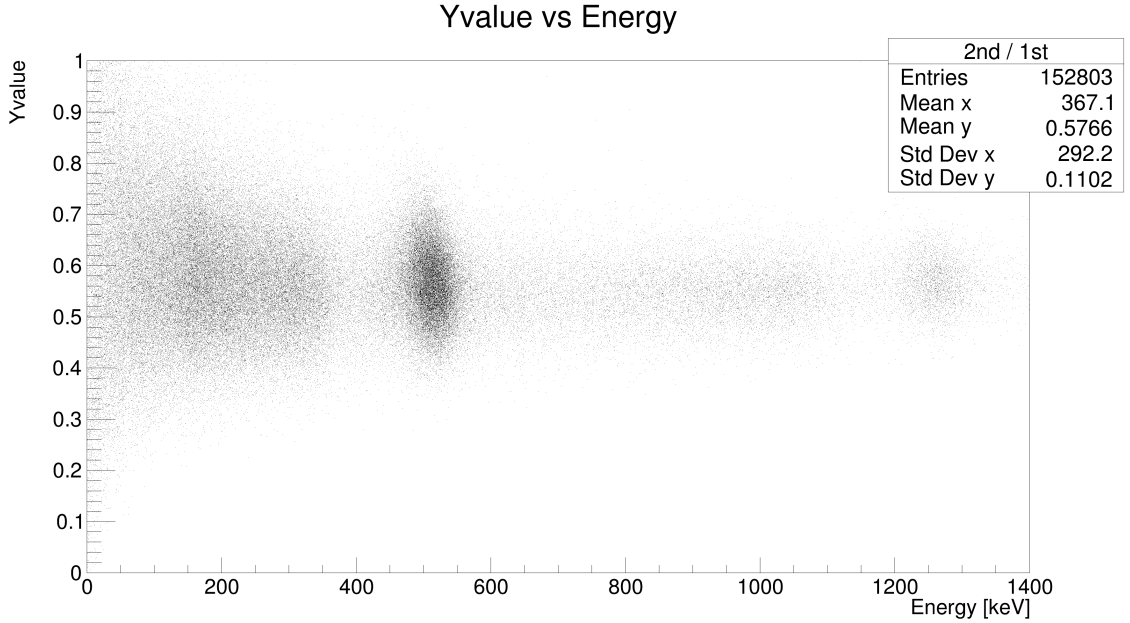


図 4.11: GAGG シンチレータで ^{22}Na からのガンマ線を検出したときの yvalue

平均値は 500 keV 以下ではエネルギーに対して右下がりになっており、その後緩やかに右上がりになる。これが図 3.6 のエネルギー別の違いと合致する。

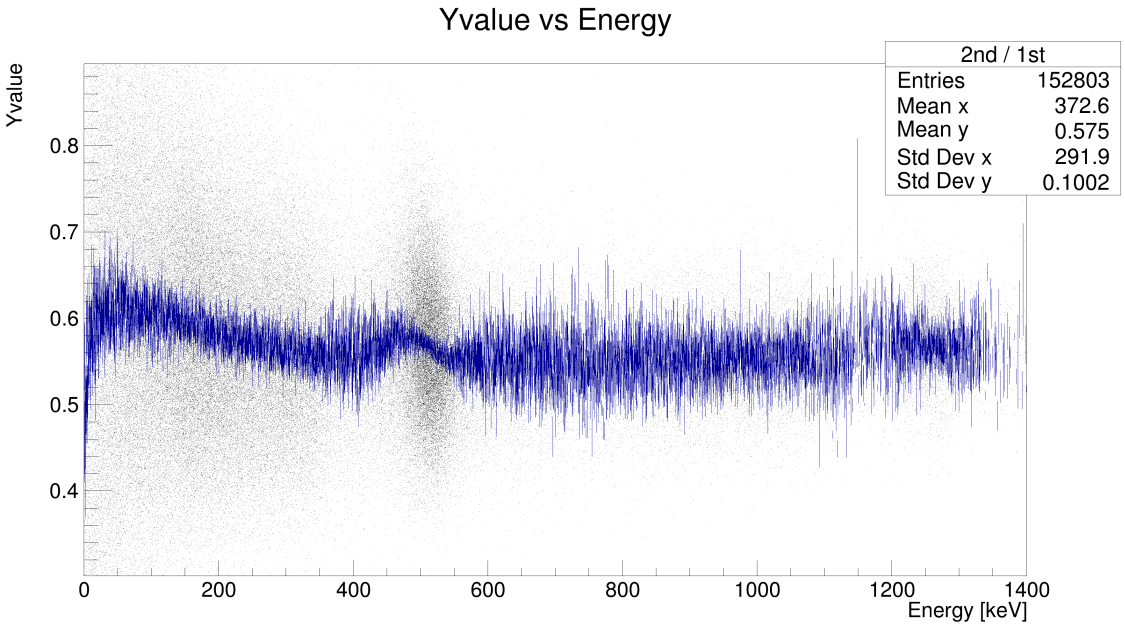


図 4.12: 図 4.11 に対して平均値と標準誤差を追加したもの

4.2.2 CsI: 波形の評価

CsI(Tl) の発光波形において、I_{fast} の積分範囲を図 3.5 の 200-600 Samples とする。I_{slow} の積分範囲を図 3.5 の 800-1200 Samples とする。CsI(Tl) シンチレータで ^{22}Na からのガンマ線を検出したときの、Yvalue とエネルギー値の散布図を図 4.13 に示す。図 4.13 の yvalue に対して平均値と標準誤差をプロットすると図 4.6 になる。これによると *Yvalue* の平均値は 150 keV ほどまで右上がりになっており、それ以上では一定になっている。これが図 3.9 のエネルギー別の違いと合致する。

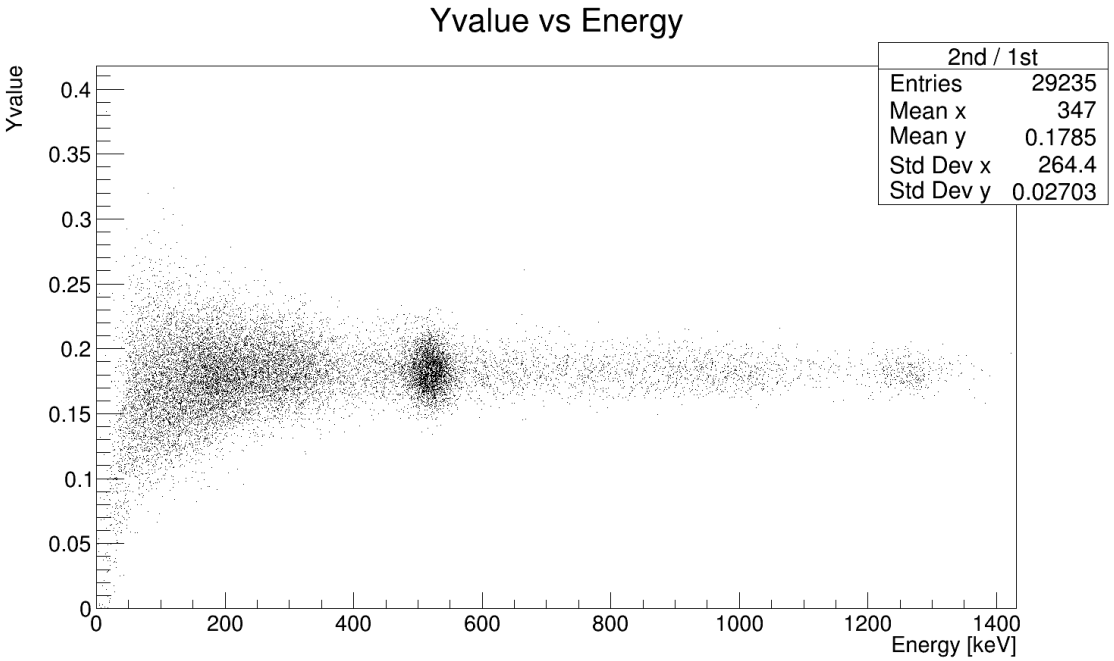


図 4.13: CsI シンチレータで ^{22}Na からのガンマ線を検出したときの yvalue

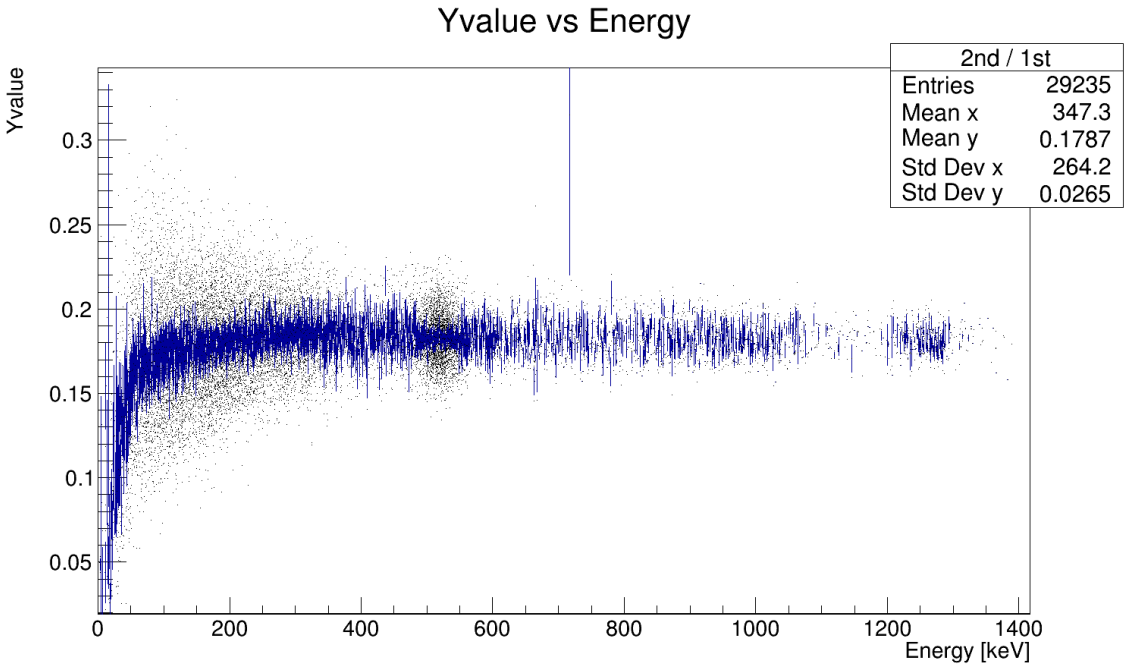


図 4.14: 図 4.13 に対して平均値と標準誤差を追加したもの

4.3　波形情報 Yvalue とエネルギー分解能

4.3.1　波形情報を用いた GAGG シンチレータのエネルギー分解能向上

図 4.11 を見ると、511 keV の光電吸収ピーク付近で、*Yvalue* とエネルギー値が負の相関を持っているように見受けられる。この図の 511 keV 付近を取り出したのが図 4.15 である。このとき σ_{Energy}=25.5 keV、エ

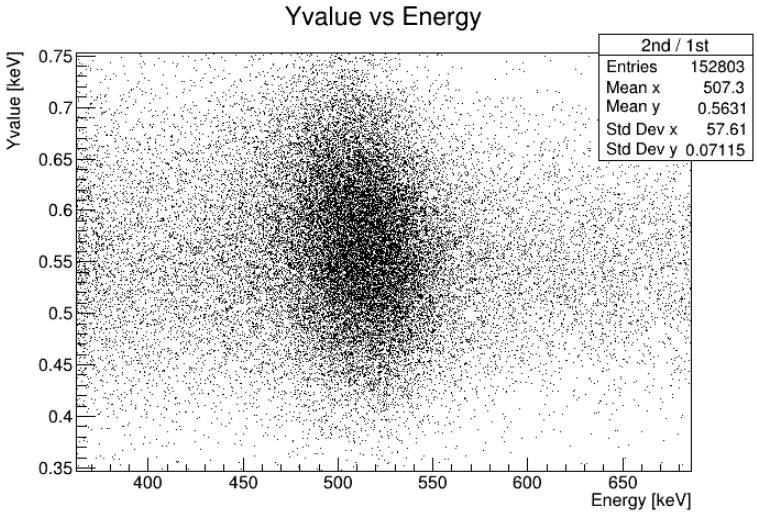


図 4.15: 図 4.11 の 511 keV 付近を取り出したもの

ネルギー分解能は 11.7 %だった。これを用いて、以下ではエネルギーが $511 \pm 2 \times \sigma_{Energy}$ の範囲に入っているものを 511 keV の光電吸収のイベントとして扱った。

エネルギーが $511 \pm 2 \times \sigma_{Energy}$ の範囲に入っているイベントの *Yvalue* のヒストグラムを作成した。（図 4.16）これによって $Mean_{Yvalue} = 0.568$、$\sigma_{Yvalue} = 0.0733$ とわかる。

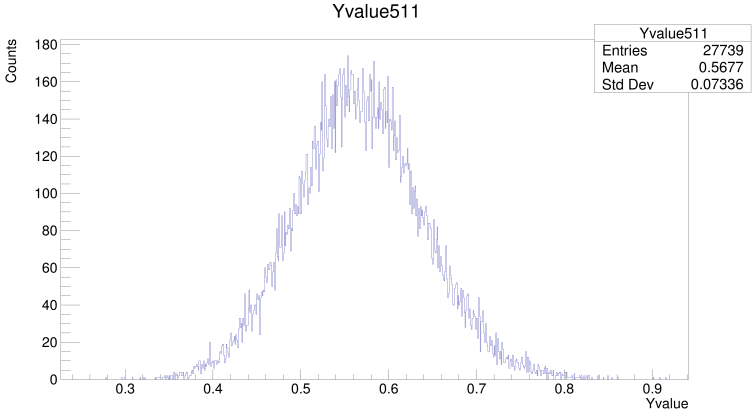


図 4.16: GAGG シンチレータで取得した 511 keV 付近の *Yvalue*

ここで、図 4.15 の領域を図 4.17 のように区分する。以下では図の青の領域を「*Yvalue* 上側」、赤の領域を「*Yvalue* 下側」と呼称する。

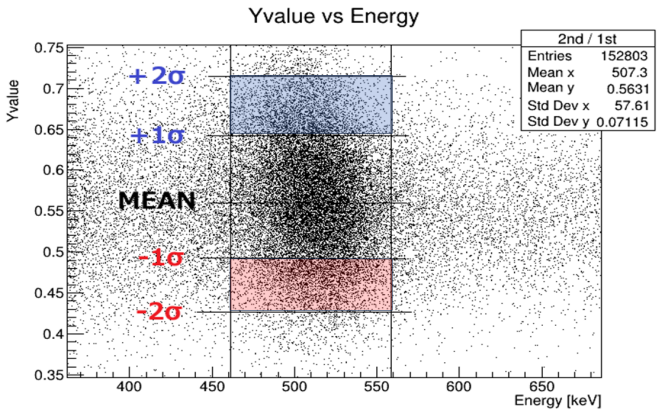


図 4.17: 511 keV のガンマ線イベント群の区分

Yvalue 上側に含まれるイベントと *Yvalue* 下側に含まれるイベントの平均波形が図 4.18。*Yvalue* 下側のイベント数を、*Yvalue* 上側のイベント数に合わせるように規格化している。*Yvalue* という量を設定した通

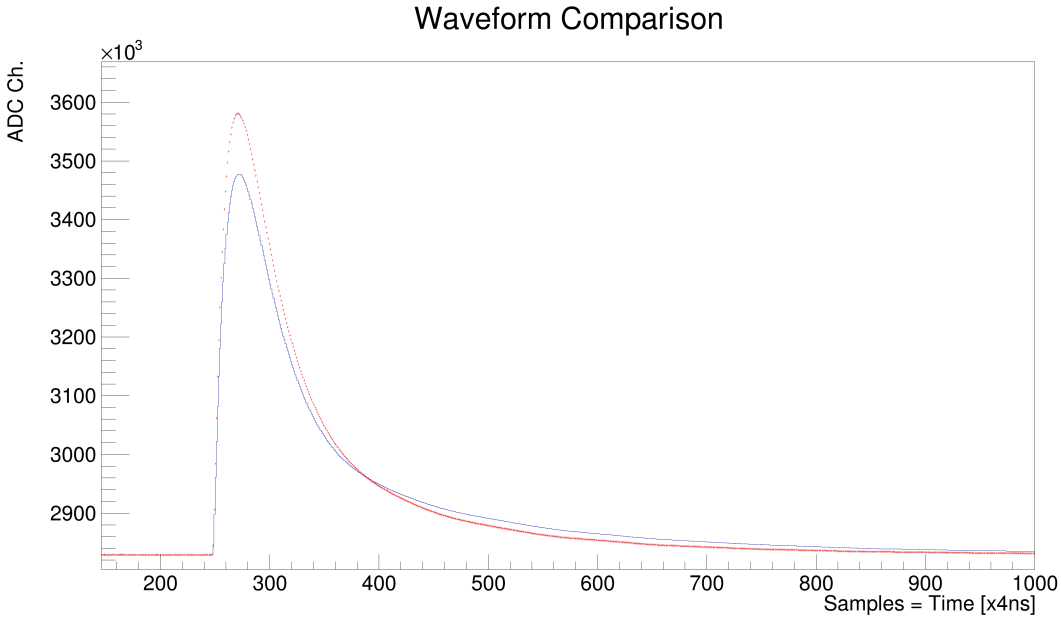


図 4.18: *Yvalue* 上側と下側の平均波形　青線が上側、赤線が下側である。

り、上側（図 4.18 の青線）は減衰の遅い成分が比較的強く、下側（図 4.18 の赤線）は減衰の速い成分が比較的強い。*Yvalue* 上側と下側に含まれるイベント数は、ガウス関数の性質よりそれぞれ約 13.6 %なので、それなりの数のイベントの波形が図 4.18 のように変化しているということである。

Yvalue の上側と下側に分けてエネルギースペクトルを作成したのが図 4.19 である。*Yvalue* の上側のエネルギースペクトル（図 4.19 の青線）の中心値は 505.9 keV、*Yvalue* の下側のエネルギースペクトル（図 4.19 の赤線）の中心値は 513.4 keV である。したがってこれらは 7.5 keV だけ異なるピークを持つ。

以上を総合すると、GAGG(Ce) シンチレータでは、同一のエネルギーを持つ 511 keV のガンマ線に対す

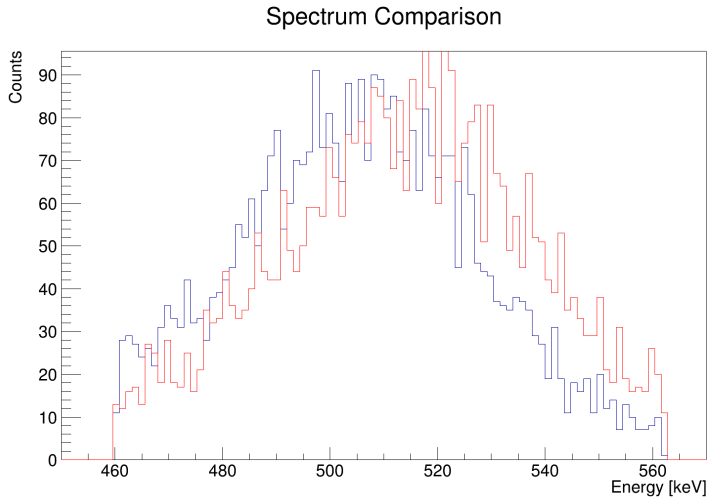


図 4.19: *Yvalue* 上側と下側のエネルギースペクトル　青線が上側、赤線が下側である。

る発光波形が異なることがあり、減衰の速い成分が強まる、もしくは減衰の遅い成分が強まる。減衰の速い成分が強まったときエネルギーの検出値は大きくなる傾向があり、減衰の遅い成分が強まったときエネルギーの検出器は小さくなる傾向がある。減衰の速い成分と減衰の遅い成分は、2.2 節にあるとおり、シンチレーションが起こったときのエネルギー準位が異なる。異なる励起状態からの発光は波長が異なる。検出器のどこかに検出効率の波長依存性があることが理由として考えられる。候補としては、光検出器 MPPC の感度の波長依存性 [4] や、もしくは図 2.4[1] からわかるように GAGG シンチレータ自身に色がついており、透過率の波長依存性があることなどがある。

ここまでで得られた情報によって GAGG(Ce) シンチレータで得られた 511 keV のガンマ線（図 4.15) によるイベントのエネルギーを補正する。*Yvalue* の上側と下側それぞれの領域での *Yvalue* の平均同士の差は、ガウス関数の性質より $2.67\sigma_{Yvalue} = 0.196$ である。また、それぞれのエネルギースペクトルのピークの差は 7.5 keV だったので、*Yvalue* の中心地からのズレに対するエネルギーの補正係数は 7.5/0.196=38.4。つまり *Yvalue* が中心値から 1 ずれている場合、エネルギー値に 38.4 keV 足して補正する。このプログラムを 6.1.9 節に載せた。この補正を施したのが図 4.20 である。図 4.15 で見られた負の相関が解消されているように見える。エネルギースペクトルはこれらの散布図の X 軸（エネルギー）に対する投射なので、散布図でのイベントのまとまりの軸が X 軸（エネルギー）に対して垂直に見えることは良いことである。

波形情報を用いた補正後のエネルギー分解能は 11.3±0,1 %となった。また、補正前データに関しても新たにエネルギースペクトルを作成し、補正後データと同じフィット区間でフィットしたところ、エネルギー分解能は 11.4±0.1 %であった。したがってエネルギー分解能は、

11.4±0.1 %から、11.3±0,1 %

になった。これは有意な差とは言いづらい。

各イベントの発光波形が変化することによってエネルギー分解能の悪化が起こる可能性があることは示せた。しかしながら今回想定したような発光波形の変化のエネルギー分解能の悪化に対する寄与は、他の様々な悪化の要因に比べて小さく、発光波形の変化の効果によってずれたエネルギー値を補正しても直ちにエネルギー分解能の向上が見込めるほどではないとわかった。

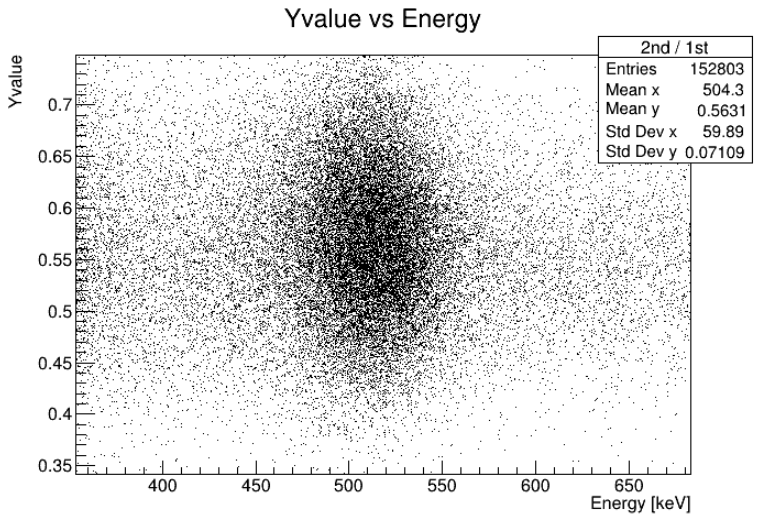


図 4.20: 波形情報を用いて補正された GAGG シンチレータによる 511 keV のガンマ線

4.3.2 波形情報を用いた CsI シンチレータのエネルギー分解能向上

図 4.13 からわかるとおり、CsI(Tl) で取得された *Yvalue* とエネルギー値には GAGG(Ce) のような補正すべき関係が見られない。4.3.1 で考察したように、波形の変化によるエネルギーのずれがシンチレータの可視光透過率の波長依存性によっているならば、図 2.3 のように無色透明のシンチレータには透過率の波長依存性がなく、波形の変化によるエネルギーのズレが小さいと予想できる。

第5章　まとめと課題

シンチレーション発光波形から得られる情報を用いて、従来の手法で求められるエネルギー値の補正を行い、エネルギー分解能を向上することを目指した。実験にはCsI(Tl) シンチレータ、GAGG(Ce) シンチレータという、どちらも発光波形が減衰の速い指数関数と減衰が遅い指数関数、合わせて２つの関数の足し合わせで表すことができるシンチレータを用いた。発光に対する減衰の速い成分の寄与と減衰の遅い成分の寄与を比較するパラメータを定義し、これとエネルギー値の相関関係について調べた。これによって、GAGG(Ce) シンチレータで取得されたエネルギー 511 keV のイベント群について、減衰が速い成分が比較的強くなったイベントはエネルギー値が真の値より大きく観測される傾向があり、逆に減衰が遅い成分が比較的強くなったイベントではエネルギー値が真の値より小さく観測される傾向があることがわかった。これがエネルギー分解能悪化の原因の一つになりうる。この関係を逆に利用して、減衰が速い成分が強くなっているイベントでは計測されたエネルギー値よりも真の値は小さい、減衰が遅い成分が強くなっているイベントでは計測されたエネルギーよりも真の値は大きいと推理し、エネルギー値の補正を行った。結果としては、エネルギー分解能の有意な向上は見られなかった。これは現時点では、波形が変化することによるエネルギー分解能悪化の効果は、その他の様々なエネルギー分解能悪化の原因によるものに比べて小さいことを示している。しかしながら将来的に他のエネルギー分解能悪化の要因が取り除かれていけば、本研究で指摘したような効果が無視できなくなり、本研究で提案したエネルギー分解能向上の手法が有効になるだろう。また、CsI(Tl) シンチレータでは上記のような傾向が認められなかった。付随して、GAGG(Ce) シンチレータの発光信号を積分する際の、最適な積分時間のエネルギー依存性に関しても考察し、観測したいエネルギー帯に応じて適切な積分時間を選べるような関数を求めた。

今回は主にGAGG(Ce) シンチレータで捉えられた511 keV のガンマ線の光電吸収ピークについて解析したが、その他のエネルギーについても解析し、波形の変化がエネルギー分解能に与える影響のエネルギー依存性についても調べたい。

同時に、物理学的観点からの考察をさらに深めたい。

第6章　補遺

6.1　プログラム

本研究で用いた自作のプログラムを記す。いずれも CINT で利用することを想定したマクロである。バージョンは ROOT 6.30/02[5] である。

以下のマクロを使用する際は、ターミナルで

```
$ root Data_*.root
```

とするか、ROOT を起動した上で

```
TFile *_file0 = TFile::Open("Data_*.root");
```

として取得した `Data_*.root` を予め読み込ませておく。

CINT のマクロを実行するには、

```
.x /pass/*.C
```

として`*.C` を実行する。`/pass/`はマクロのパスで、相対パスでも絶対パスでも動作する。

hogehoge.C の void hogehoge() が引数を持つ場合、

```
.x /pass/hogehoge.C(int, "char")
```

ここで hogehoge.C と (int, ”char”) の間にスペースを入れないよう注意する。

全体を通して `double_t a` と `double_t b` は式 (3.2) で定義した較正直線の係数である。よって使用する際には予めエネルギーの較正を行い、プログラムを書き換える。

6.1.1　makespectrum.C

実験で得られた ROOT ファイルから積分値のスペクトルを作成するマクロである。

プログラム 6.1: ”makespectrum.C”

```
1  //makespectrum.C
2
3  void makespectrum() {
4          TArrayS *sample = 0;
5          Data->SetBranchAddress("Samples", &sample);
6          int nevents = Data->GetEntries();
7          TH1D* spec = new TH1D("spectrum", "Cs-137 by CsI(Tl) Spectrum", 6000, 0,
               80000);
8          Data->GetEntry(1);
9          int nsamples = sample->GetSize();
10         for(int j = 0; j < nevents; j++) {
11                 Data->GetEntry(j);
```

```
12                 double_t sum = 0.;
13                 double_t offsum = 0.;
14                 for(int i = 0; i < 200; i++) {
15                         offsum += sample->GetAt(i);
16                 }
17                 double_t offmean = offsum / 200.;
18                 for(int i = 250; i < nsamples; i++) {
19                         sum += sample->GetAt(i) - offmean;
20                 }
21                 spec->Fill(sum);
22         }
23         spec->Draw();
24         spec->GetXaxis()->SetTitle("ADC Ch. Integration");
25         spec->GetYaxis()->SetTitle("Counts");
26 }
```

6.1.2 fitspectrum.C

6.1.1 節で作成したスペクトルをフィットするマクロである。このプログラムでも 6.1.1 節と同じ方法でスペクトルを作成しているので、独立に動作する。

プログラム 6.2: ”fitspectrum.C”

```
1  //fitspectrum.C
2
3  void fitspectrum(double_t fitmin, double_t fitmax) {
4          TArrayS *sample = 0;
5          Data->SetBranchAddress("Samples", &sample);
6          int nevents = Data->GetEntries();
7          TH1D* spec = new TH1D("spectrum", "Cs-137 by CsI(Tl) Spectrum", 6000, 0,
               80000);
8          Data->GetEntry(1);
9          int nsamples = sample->GetSize();
10         for(int j = 0; j < nevents; j++) {
11                 Data->GetEntry(j);
12                 double_t sum = 0.;
13                 double_t offsum = 0.;
14                 for(int i = 0; i < 200; i++) {
15                         offsum += sample->GetAt(i);
16                 }
17                 double_t offmean = offsum / 200.;
18                 for(int i = 250; i < nsamples; i++) {
19                         sum += sample->GetAt(i) - offmean;
20                 }
21                 spec->Fill(sum);
22         }
23         spec->Draw();
```

```
24          spec->Fit("gaus","","",fitmin,fitmax);
25          spec->GetXaxis()->SetTitle("ADC Ch. Integration");
26          spec->GetYaxis()->SetTitle("Counts");
27  }
```

6.1.3 averagewaveform.C

指定したエネルギー区間で信号波形を平均し描画するマクロである。

プログラム 6.3: ”averagewaveform.C”

```
1   //averagewaveform.C
2
3   void averagewaveform(double_t ene_min, double_t ene_max, int col, const char* label) {
4           double_t a = 32.25; double_t b = 134.63;
5           TArrayS *sample = 0;
6           Data->SetBranchAddress("Samples", &sample);
7           Data->GetEntry(1);
8           int nsamples = sample->GetSize();
9           int nevents = Data->GetEntries();
10          TH1D* waveform = new TH1D("waveform", "waveform", nsamples-1, 0, nsamples-1);
11          TH1D* wavecomp = new TH1D("wavecomp", label, 1600, 0, 1600);
12          for(int j = 0; j < nevents; j++) {
13                  Data->GetEntry(j);
14                  double_t sum = 0;
15                  double_t bgsum = 0;
16                  for(int i = 0; i < 200; i++) {
17                          bgsum += sample->GetAt(i);
18                  }
19                  double_t bg = bgsum / 200.;
20                  for(int i = 250; i < 500; i++) {
21                          double_t hight = sample->GetAt(i) - bg;
22                          sum += hight;
23                  }
24                  double_t ene = (sum-b)/a;
25                  if(ene>=ene_min && ene<ene_max){
26                          for(int i = 200; i < nsamples; i++) {
27                                  double_t hight = sample->GetAt(i) - bg;
28                                  waveform->SetBinContent(i,waveform->GetAt(i)+hight);
29                          }
30                  }
31          }
32          double_t max = 1.;
33          int i_max = 0;
34          for(int i = 0; i < nsamples; i++){
35                  double_t hight = waveform->GetAt(i);
36                  if(hight>max){
37                          max = hight;
```

```
38                          i_max=i;
39                  }
40          }
41          waveform->Scale(100./max);
42          for(int i = 0; i <= 1600; i++){
43                  wavecomp->SetBinContent(i, waveform->GetAt(i_max - 200 + i));
44          }
45          wavecomp->SetLineColor(col);
46          wavecomp->Draw("same");
47          wavecomp->GetXaxis()->SetTitle("Sample = Time [x4ns]");
48          wavecomp->GetYaxis()->SetTitle("Normalized ADC Ch.");
49          wavecomp->SetStats(0);
50  }
```

6.1.4 signallength.C

各イベントの信号の長さとエネルギーを散布図にプロットするマクロである。

プログラム 6.4: ”signallength.C”

```
1  //signallength.C
2
3  void signallength() {
4          TArrayS *sample = 0;
5          Data->SetBranchAddress("Samples", &sample);
6          int nevents = Data->GetEntries();
7          TH2D* hist = new TH2D("time vs integration", "Signal Length vs Energy", 5000, 0
               ,1400, 10000, 0, 10000);
8          Data->GetEntry(1);
9          int nsamples = sample->GetSize();
10         for(int j = 0; j < nevents; j++) {
11                 //evaluate offset
12                 Data->GetEntry(j);
13                 TH1I* offset = new TH1I("offset", "offset", 20, 720, 740);
14                 for(int i = 0; i < 200; i++) {
15                         Int_t off = sample->GetAt(i);
16                         offset->Fill(off);
17                 }
18                 double_t offmean = offset->GetMean();
19                 double_t offstddev = offset->GetStdDev();
20                 delete offset;
21                 //set y: integration time
22                 int timing = 250;
23                 for(int i = 300; i < nsamples; i++) {
24                         double_t hight = sample->GetAt(i) - offmean;
25                         if(hight < -1. * offstddev) {
26                                 timing = i;
27                                 break;
```

```
28                      }
29                      if(i == nsamples -1) {timing = i;}
30              }
31              Int_t time_ns = 4 * (timing - 250);
32              //set x: full integration 1 mic
33              double_t sum = 0.;
34              for(int i = 250; i < 250 + 250; i++) {
35                      double_t hight = sample->GetAt(i) - offmean;
36                      sum += hight;
37              }
38              double_t a = 32.25; double_t b = 134.63;
39              double_t ene = (sum - b) / a;
40              //fill the content to the hist
41              hist->Fill(ene, time_ns);
42      }
43      hist->Draw("scat");
44      hist->GetXaxis()->SetTitle("Energy [keV]");
45      hist->GetYaxis()->SetTitle("Signal Length [ns]");
46      hist->SetStats(0);
47  }
```

6.1.5 fitprofilesignallength.C

3.6.3 で求められた散布図の、各横軸 bin 中の信号長さの平均とその標準誤差をプロットし、さらに平均値をフィットするマクロである。

プログラム 6.5: ”fitprofilesignallength.C”

```
1  //fitprofilesignallength.C
2
3  void fitprofilesignallength() {
4          TArrayS *sample = 0;
5          Data->SetBranchAddress("Samples", &sample);
6          int nevents = Data->GetEntries();
7          TH2D* hist = new TH2D("time vs integration", "Signal Length vs Energy", 5000, 0
               ,1400, 10000, 0, 10000);
8          Data->GetEntry(1);
9          int nsamples = sample->GetSize();
10         for(int j = 0; j < nevents; j++) {
11                 //evaluate offset
12                 Data->GetEntry(j);
13                 TH1I* offset = new TH1I("offset", "offset", 20, 720, 740);
14                 for(int i = 0; i < 200; i++) {
15                         Int_t off = sample->GetAt(i);
16                         offset->Fill(off);
17                 }
18                 double_t offmean = offset->GetMean();
```

```
19                 double_t offstddev = offset->GetStdDev();
20                 delete offset;
21                 //set y: integration time
22                 int timing = 250;
23                 for(int i = 300; i < nsamples; i++) {
24                         double_t hight = sample->GetAt(i) - offmean;
25                         if(hight < -1. * offstddev) {
26                                 timing = i;
27                                 break;
28                         }
29                         if(i == nsamples -1) {timing = i;}
30                 }
31                 Int_t time_ns = 4 * (timing - 250);
32                 //set x: full integration 1 mic
33                 double_t sum = 0.;
34                 for(int i = 250; i < 250 + 250; i++) {
35                         double_t hight = sample->GetAt(i) - offmean;
36                         sum += hight;
37                 }
38                 double_t a = 32.25; double_t b = 134.63;
39                 double_t ene = (sum - b) / a;
40                 //fill the content to the hist
41                 hist->Fill(ene, time_ns);
42         }
43         hist->Draw("scat");
44         hist->GetXaxis()->SetTitle("Energy [keV]");
45         hist->GetYaxis()->SetTitle("Signal Length [ns]");
46         hist->SetStats(0);
47         TProfile *prof = hist->ProfileX();
48         prof->Draw("same");
49         TF1 *func = new TF1("func", "[0] * std::pow(x-[1], [2]) + [3]", 0, 1400);
50         func->SetParameters(20000, -10, 0.03, -20000);
51         prof->Fit(func);
52 }
```

6.1.6 spectrum_adjustable.C

プログラム 6.6: ”spectrum_adjustable.C”

```
1  //spectrum_adjustable.C
2
3  void spectrum_adjustable() {
4          TArrayS *sample = 0;
5          Data->SetBranchAddress("Samples", &sample);
6          int nevents = Data->GetEntries();
7          TH1D* spec = new TH1D("spectrum", "spectrum", 3000, 0, 50000);
8          int nsamples = 2496;
```

```
        for(int j = 0; j < nevents; j++) {
                Data->GetEntry(j);
                double_t sum = 0.;
                double_t offsum = 0.;
                //evaluate offset
                for(int i = 0; i < 200; i++) {
                        offsum += sample->GetAt(i);
                }
                //evaluate energy 1mic
                double_t offmean = offsum / 200.;
                for(int i = 250; i < 250 + 250; i++) {
                        sum += sample->GetAt(i) - offmean;
                }
                double_t a = 32.25; double_t b = 134.63;
                double_t ene = (sum - b) / a;
                //decide integration time
                double_t p0 = 66680; double_t p1 = -12.3013;
                double_t p2 = 0.0100911; double_t p3 = -68127.3;
                double_t integration_ns = p0 * std::pow(ene - p1, p2) + p3;
                double_t integration_samples = integration_ns / 4.;
                //actually integrate
                double_t sum2 = 0.;
                for(int i = 250; i < 250. + integration_samples; i++) {
                        sum2 += sample->GetAt(i) - offmean;
                }
                spec->Fill(sum2);
        }
        spec->Draw();
        spec->SetStats(0);
}
```

6.1.7 signallengthcorrection.C

プログラム 6.7: ”signallengthcorrection.C”

```
//signallengthcorrection.C

void signallengthcorrection(double_t angle) {
        TArrayS *sample = 0;
        Data->SetBranchAddress("Samples", &sample);
        int nevents = Data->GetEntries();
        TH2D* hist = new TH2D("time vs integration", "Signal Length vs Energy", 5000, 0
             ,1400, 10000, 0, 10000);
        Data->GetEntry(1);
        int nsamples = sample->GetSize();
        TF1 *func = new TF1("func", "[0] * std::pow(x-[1], [2]) + [3]", 0, 1400);
        func->SetParameters(66680,-12.3013,0.0100911,-68127.3);
```

```
        for(int j = 0; j < nevents; j++) {
                //evaluate offset
                Data->GetEntry(j);
                TH1I* offset = new TH1I("offset", "offset", 20, 720, 740);
                for(int i = 0; i < 200; i++) {
                        Int_t off = sample->GetAt(i);
                        offset->Fill(off);
                }
                double_t offmean = offset->GetMean();
                double_t offstddev = offset->GetStdDev();
                delete offset;
                //set y: integration time
                int timing = 250;
                for(int i = 300; i < nsamples; i++) {
                        double_t hight = sample->GetAt(i) - offmean;
                        if(hight < -1. * offstddev) {
                                timing = i;
                                break;
                        }
                        if(i == nsamples -1) {timing = i;}
                }
                Int_t time_ns = 4 * (timing - 250);
                //set x: full integration 1 mic
                double_t sum = 0.;
                for(int i = 250; i < 250 + 250; i++) {
                        double_t hight = sample->GetAt(i) - offmean;
                        sum += hight;
                }
                double_t a = 32.25; double_t b = 134.63;
                double_t ene = (sum - b) / a;
                //modify the energy
                double_t newene = ene + angle * (time_ns - func->Eval(ene));
                //fill the content to the hist
                hist->Fill(newene, time_ns);
        }
        hist->Draw("scat");
        hist->GetXaxis()->SetTitle("Energy [keV]");
        hist->GetYaxis()->SetTitle("Signal Length [ns]");
        hist->SetStats(0);
        func->Draw("same");
}
```

6.1.8 yvalue.C

プログラム 6.8: ”yvalue.C”

```
//yvalue.C
```

```
2
3  void yvalue() {
4          TArrayS *sample = 0;
5          Data->SetBranchAddress("Samples", &sample);
6          int nevents = Data->GetEntries();
7          TH2D* hist = new TH2D("2nd / 1st", "Yvalue vs Energy", 5000, 0 ,1400, 10000,
               0, 1);
8          int nsamples = 2496;
9          TF1 *func = new TF1("func", "[0] * std::pow(x-[1], [2]) + [3]", 0, 1400);
10         func->SetParameters(66680,-12.3013,0.0100911,-68127.3);
11         double_t a = 36.777; double_t b = 932.74;
12         for(int j = 0; j < nevents; j++) {
13                 Data->GetEntry(j);
14                 //evaluate offset
15                 Data->GetEntry(j);
16                 double_t offsum = 0.;
17                 for(int i = 0; i < 200; i++) {
18                         Int_t off = sample->GetAt(i);
19                         offsum += off;
20                 }
21                 double_t offmean = offsum / 200.;
22                 //find the peak
23                 double_t peakhight = 0.;
24                 int peakat = 0;
25                 for(int i = 250; i < nsamples; i++) {
26                         double_t hight = sample->GetAt(i) - offmean;
27                         if(hight > peakhight) {
28                                 peakhight = hight;
29                                 peakat = i;
30                         }
31                 }
32                 if(peakat > 1000) {continue;}
33                 //integrate the 2nd content;
34                 double_t integrate2 = 0.;
35                 for(int i = peakat + 100; i < peakat + 750; i++) {
36                         double_t hight = sample->GetAt(i) - offmean;
37                         integrate2 += hight;
38                 }
39                 //integrate the 1st content;
40                 double_t integrate1 = 0.;
41                 for(int i = peakat; i < peakat + 100; i++) {
42                         double_t hight = sample->GetAt(i) - offmean;
43                         integrate1 += hight;
44                 }
45                 //set y: integrate2 / fullintegrate
46                 double_t yvalue = integrate2 / integrate1;
47                 //set x: integration 3 mic
48                 double_t sum = 0.;
```

```
49                 for(int i = 250; i < 250 + 750; i++) {
50                         double_t hight = sample->GetAt(i) - offmean;
51                         sum += hight;
52                 }
53                 double_t ene = (sum - b) / a;
54                 //fill the content to the hist
55                 hist->Fill(ene, yvalue);
56         }
57         hist->Draw("scat");
58         hist->GetXaxis()->SetTitle("Energy [keV]");
59         hist->GetYaxis()->SetTitle("Yvalue");
60 }
```

6.1.9 wavecorrection.C

プログラム 6.9: ”wavecorrection.C”

```
1  //wavecorrection.C
2
3  void wavecorrection() {
4          TArrayS *sample = 0;
5          Data->SetBranchAddress("Samples", &sample);
6          int nevents = Data->GetEntries();
7          TH2D* hist = new TH2D("2nd / 1st", "Yvalue vs Energy", 5000, 0 ,1400, 10000,
               0, 1);
8          int nsamples = 2496;
9          TF1 *func = new TF1("func", "[0] * std::pow(x-[1], [2]) + [3]", 0, 1400);
10         func->SetParameters(66680,-12.3013,0.0100911,-68127.3);
11         double_t a = 36.777; double_t b = 932.74;
12         for(int j = 0; j < nevents; j++) {
13                 Data->GetEntry(j);
14                 //evaluate offset
15                 Data->GetEntry(j);
16                 double_t offsum = 0.;
17                 for(int i = 0; i < 200; i++) {
18                         Int_t off = sample->GetAt(i);
19                         offsum += off;
20                 }
21                 double_t offmean = offsum / 200.;
22                 //find the peak
23                 double_t peakhight = 0.;
24                 int peakat = 0;
25                 for(int i = 250; i < nsamples; i++) {
26                         double_t hight = sample->GetAt(i) - offmean;
27                         if(hight > peakhight) {
28                                 peakhight = hight;
29                                 peakat = i;
```

```
30                      }
31              }
32              if(peakat > 1000) {continue;}
33              //integrate the 2nd content;
34              double_t integrate2 = 0.;
35              for(int i = peakat + 100; i < peakat + 750; i++) {
36                      double_t hight = sample->GetAt(i) - offmean;
37                      integrate2 += hight;
38              }
39              //integrate the 1st content;
40              double_t integrate1 = 0.;
41              for(int i = peakat; i < peakat + 100; i++) {
42                      double_t hight = sample->GetAt(i) - offmean;
43                      integrate1 += hight;
44              }
45              //set y: integrate2 / fullintegrate
46              double_t yvalue = integrate2 / integrate1;
47              //set x: integration 3 mic
48              double_t sum = 0.;
49              for(int i = 250; i < 250 + 750; i++) {
50                      double_t hight = sample->GetAt(i) - offmean;
51                      sum += hight;
52              }
53              double_t ene = (sum - b) / a;
54              //fill the content to the hist
55              double_t angle = 38.285;
56              double_t ymean = 0.5677;
57              double_t newene = ene + angle * (yvalue - ymean);
58              hist->Fill(newene, yvalue);
59      }
60      hist->Draw("scat");
61      hist->GetXaxis()->SetTitle("Energy [keV]");
62      hist->GetYaxis()->SetTitle("Yvalue");
63  }
```

謝辞

本研究をするに当たりご支援・ご協力をくださった皆様に、この場をお借りしてお礼申し上げます。特に、指導にあたってくださった “ひろたかさん” こと高橋弘充准教授には大変お世話になりました。約半年間に渡って、懇切丁寧に教えていただきました。また、卒論提出に漕ぎ着けるまでに様々なトラブルがありましたが、毎回対処していただきました。本当にありがとうございました。研究室の教員の皆さまには、ミーティングなどの機会でご助言をいただき、とても勉強になりました。ありがとうございました。研究室の先輩方にも研究のこと・その他のことでたくさんアドバイスをいただきました。また、機会があるごとにご飯やお酒をご馳走していただきました。研究に関わることからくだらないことまでいろいろなお話ができ、とても楽しかったです。ありがとうございました。同期の B4 の皆さま、いつも私のムダ話に付き合ってくれてありがとうございました。私がトラブルで落ち込んだときも明るく励ましてくれてありがとうございました。今後あまりお会いできなくなりそうなのが本当に残念です。家族には、大学生の期間は主に経済的にご支援をいただきました。私がここまでやってこれたのは、ここに挙げさせていただいた方を始めとしたたくさんの方々のおかげです。本当にお世話になりました。

最後に皆様、私のことは忘れてもデータのバックアップを取ることは忘れないでください。

参考文献

[1] C&A Corporation: GAGG(Ce)：ガンマ線検出器. https://www.c-and-a.jp/products_details/products_detail_jp_GAGG.html, 2024 年 2 月 6 日閲覧

[2] W Wolszczak and P Dorenbos: Time-resolved gamma spectroscopy of single events, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, **886**, 30/35 (2018)

[3] Glenn F. Knoll: 『放射線計測ハンドブック 第 4 版』, オーム社 (平成 25 年 9 月 25 日). 神野郁夫, 木村逸郎, 坂井英次 訳

[4] 浜松ホトニクス: 技術資料/MPPC. https://www.hamamatsu.com/content/dam/hamamatsu-photonics/sites/documents/99_SALES_LIBRARY/ssd/mppc_kapd9008j.pdf, 2024 年 2 月 6 日閲覧

[5] ROOT: analyzing petabytes of data, scientifically. urlhttps://www.root.cern