ラドン検出器の作成

小林親司　福岡亮輔

平成20年4月1日

概 要

現在、宇宙線研究室では 3 次元飛跡検出が出来る『μ TPC』という検出器が稼動している。この検出器の構成物質からラドンが空間中に染み出すことでバックグラウンドとなり、検出に無視できない影響を与えることが分かっている。もしラドン放出量の少ない検出器構成物質を判別出来れば、それらを選択して検出器を構成することでバックグラウンドは低減される。

本レポートでは、物質から染み出すラドン量を測定できる「ラドン検出器」の製作を目標にして、 検出器に用いる各装置の製作及びいくつかの予備実験を行った。最終的に「ラドン検出器」の完成までには至らなかったが、完成に向けて製作した装置及び行った予備実験について述べる。

目 次

第1章　実験目的・動機

1.1　ウラン系列 ^{222}Rn

一般に物質にはウラン系列及びトリウム系列の放射性核種が含まれている。Rn はこれらの系列の唯一の希ガスである。ウラン系列の放射性核種が崩壊を繰り返して ^{222}Rn になると、その一部が物質中から飛び出して空間中を飛び回るようになる。^{222}Rn は空間中に出た後にも、半減期 3.8 日で図 1.1 のように崩壊を繰り返し、それに伴いα、β及びγ線を放出する。このようにして発生する放射線は、様々な測定のバックグラウンドになることが知られている。

一方で ^{222}Rn の同位体であるトリウム系列の ^{220}Rn も希ガスであるが、半減期が 55.6 秒と短いためほとんど空間中に飛び出さない。この 2 つを区別するため一般に、^{222}Rn をラドン、^{220}Rn をトロンと呼ぶ。

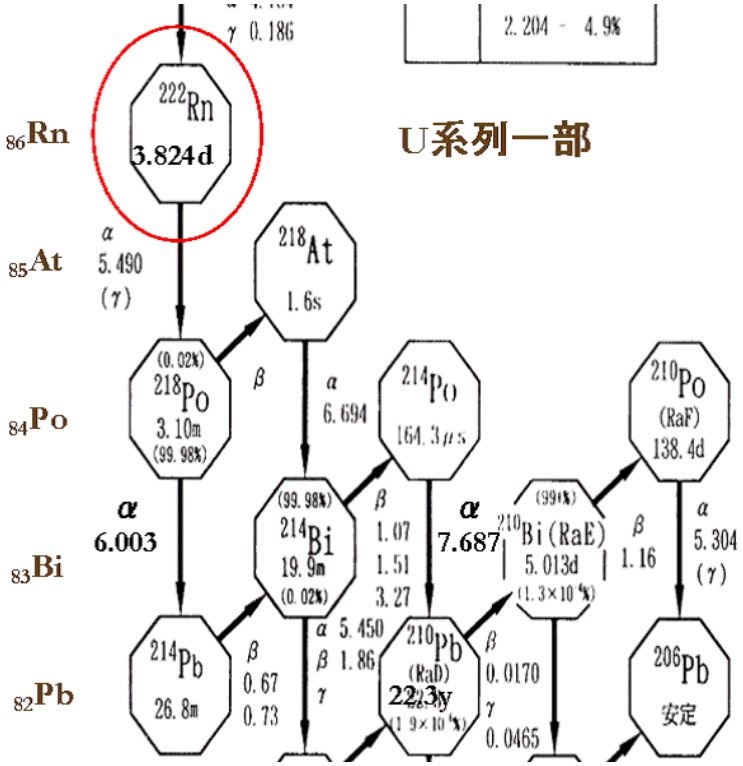


図 1.1: ウラン系列壊変表一部

1.2　μ TPC でのラドンの働き

μ TPC とは宇宙線研究室で使用されている、3 次元飛跡検出装置であり外観は図 1.2 のようになっている。

前節で述べたように、物質にはウラン及びトリウム系列の放射性核種が含まれており、μ TPC を構成するチャンバー内壁、回路等も例外ではない。これらから、ウラン系列放射性核種が崩壊し ^{222}Rn となって空間中に染み出し、さらに空間中で崩壊が進むことで放射線を放出、その放射線が検出器に入る、といった具合でバックグラウンドとなる。実際にラドンが μ TPC のバックグラウンドになっている裏付けとして、図 1.1 での崩壊 α 線のエネルギーピークが、3.8 日の時定数 (^{222}Rn の半減期) で増大しているという事実がある。(すべての崩壊 α 線が ^{222}Rn と同じ減衰因子である理由は後述の永続平衡を参照)。

μ TPC において、ラドンによるバックグラウンドの低減は大きな課題のひとつである。

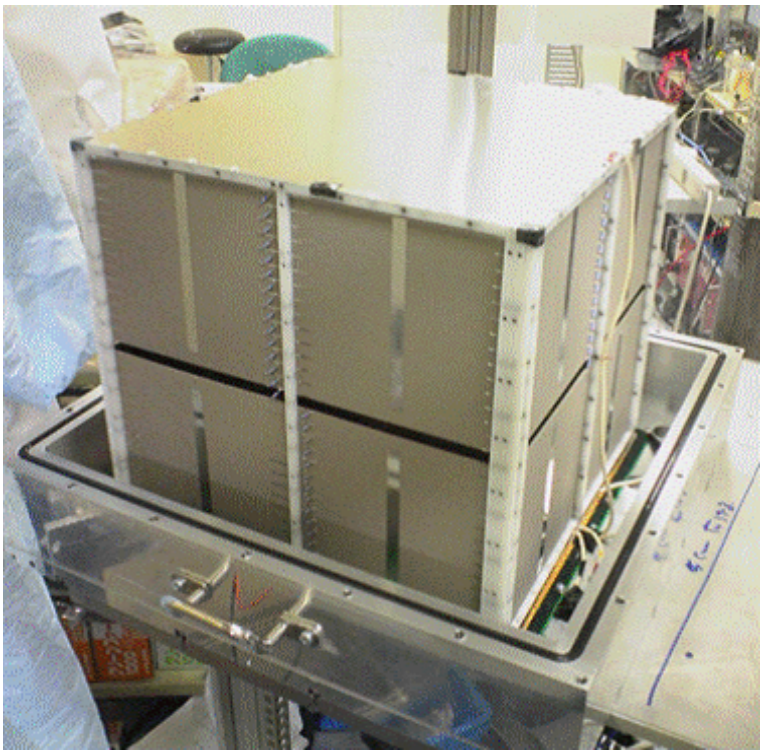


図 1.2:　μ TPC 外観
大きさは底辺 [23cm × 28cm] ×高さ [31cm]

1.3　実験目標及び動機

本レポートでは、物質から放出されるラドン量を測定できる「ラドン検出器」の製作を実験目標とした。即ち、真空容器中に試料を置き、時間経過と共に ^{222}Rn の崩壊によるピークがどのように増大するかを測定出来る検出器を製作する。試料ごとのピークの高さを比べれば、それぞれのラドン放出量を比べることが出来るようになる。

実験動機は、この検出器を用いて前節で述べたμ TPC の構成物質を、ラドン放出量のより少ないものに選択することによるバックグラウンドの低減である。

第2章　実験原理・実験装置

2.0.1　検出原理及び検出器の構成

^{222}Rn の崩壊核種である ^{218}Po は空気中で通常 90 %以上が陽イオンとして存在する。この ^{218}Po を静電捕集し、崩壊時のα線を Si-PPD で検出することで ^{222}Rn の濃度を測定することができる。負の高電圧により ^{218}Po を静電捕集することで非常に高い検出感度が期待できる。

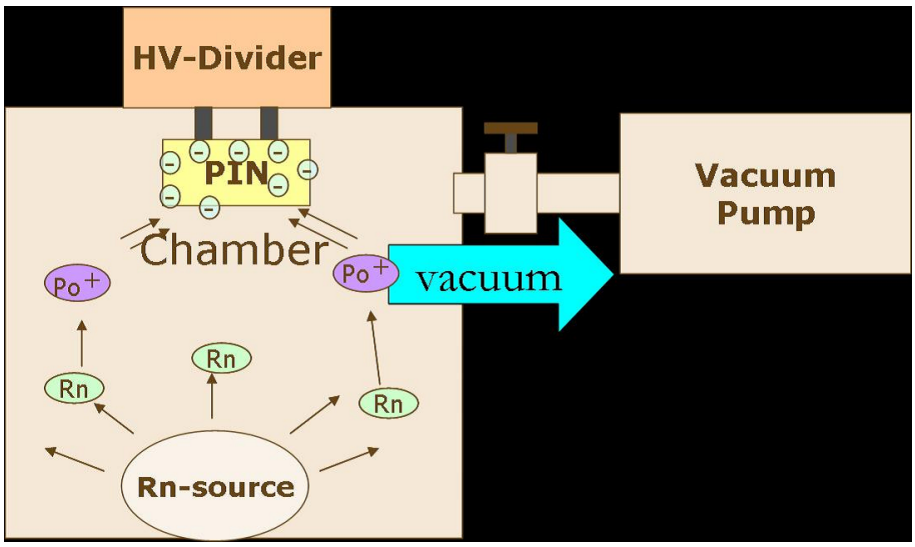


図 2.1: 実験装置の概略図

実験手順を以下に示す。

　試料を chamber の中にいれ、真空に引く。　PIN を-1.5 ｋ V 程度に印加する。（試料から Rn が染み出す。さらに娘核である Po も生じる）　PIN に入るα線を測定する。

＊＊　の印加電圧は静電捕集用の電圧である。　においては通常のスペクトル取得を時間経過とともに記録をする。

2.0.2 Rn,Poの生成　放射性壊変と永続平衡

放射性壊変の壊変法則はNを時間tにおける原子数、Aを時間tにおける放射能、λを壊変定数、Tを半減期として

$-\frac{dN}{dt} = \lambda N = A$

$N = N_0 e^{-\lambda t}$

$A = A_0 e^{-\lambda t} = A_0(\frac{1}{2})^{\frac{t}{T}}$

と表される。これが一つの放射性元素の壊変を表す。Th系列やU系列などの連続的な壊変は逐次改変といわれる。逐次改変はi番目の崩壊核種の各々定数を下付のiで示せば

$-\frac{dN_i}{dt} = \lambda_{i-1} N_{i-1} - \lambda_i N_i$

という連立微分方程式で書かれる。(永続平衡や過渡平衡は逐次壊変の近似の仕方)

厳密に原子数の時間変化を追う場合にはすべての微分方程式を解く必要があるが一般的には過渡平衡と永続平衡を測定対象の時間スケールに合わせて近似する。

放射線壊変全体を扱う場合は主にThやU以降をすべて永続平衡と仮定して扱う。この場合の永続平衡はThの半減期$1.4 * 10^{10} y$のスケールで考えたときにそれ以降の核種の半減期はとても短いとする考え方で、この平衡を仮定したときＴｈ以外の核種に対して

$\cdots = \lambda_{i-1} N_{i-1} = \lambda_i N_i = \cdots$

が成り立つと考える。

今回の実験ではU系列のRnの量が知りたいので前後の崩壊核種の半減期を考える。

まず後ろを考えるとRnの半減期3.8dに対し、Poの半減期が$3.1m$で十分短い。なのでPoはRnの半減期のスケール(数日程度)の測定においてはRnと永続平衡状態であると考える。(PoはRnとほぼ等量存在すると考える。)一方前の核種を見るとRaの半減期は$1.6 * 10^3$yでとても長い。この場合Rnの存在量を考える際にはRaとRnの過渡平衡を考える必要がある。(Raのタイムスケールで考えるとRnも含めて永続平衡で考えればよいが今回はRnのスケールで考えるため。)

Raに対してλ_1, N_1、Rnに対してλ_2, N_2を考え、まずN_1を一次近似する。t=0で$N_1 = N_{10}$とすれば

$-\frac{dN_1}{dt} = \lambda_1 N_{10}$

ここから

$N_1 = N_{10} - \lambda_1 N_{10} t$

逐次壊変の式から

$\frac{dN_2}{dt} = \lambda_1 N_{10} - \lambda_2 N_2$

なのでt=0で$N_2 = 0$のもとでこの式を解けば

$N_2 = \frac{\lambda_1}{\lambda_2} N_{10} + e^{-\lambda t}$

と計算される。

2.1 実験装置

2.1.1 chamber

試料の容器として activeshop 社製のステンレス密閉容器を用いた。この容器は内部が電解研磨されていて、ステンレス容器の内部で生じた Rn が表面に染み出しにくくなっている。また本体のふちにシリコン製のパッキンが着いており、機密性を保てるようになっている。容器の大きさは図の通りである。

今回はこの容器内部を真空に引き、シグナルを取り出したいという動機から蓋部分の設計をした。下の図が実際に発注した設計図である。設計に当たっては Jw-cad というソフトをつかった。

設計に当たっては基盤を固定するためのねじ穴 (貫通しない) が必要であること、真空を引く貫通穴が必要であること、真空の際にかかる外圧に耐えうるかという 3 点に注意した。

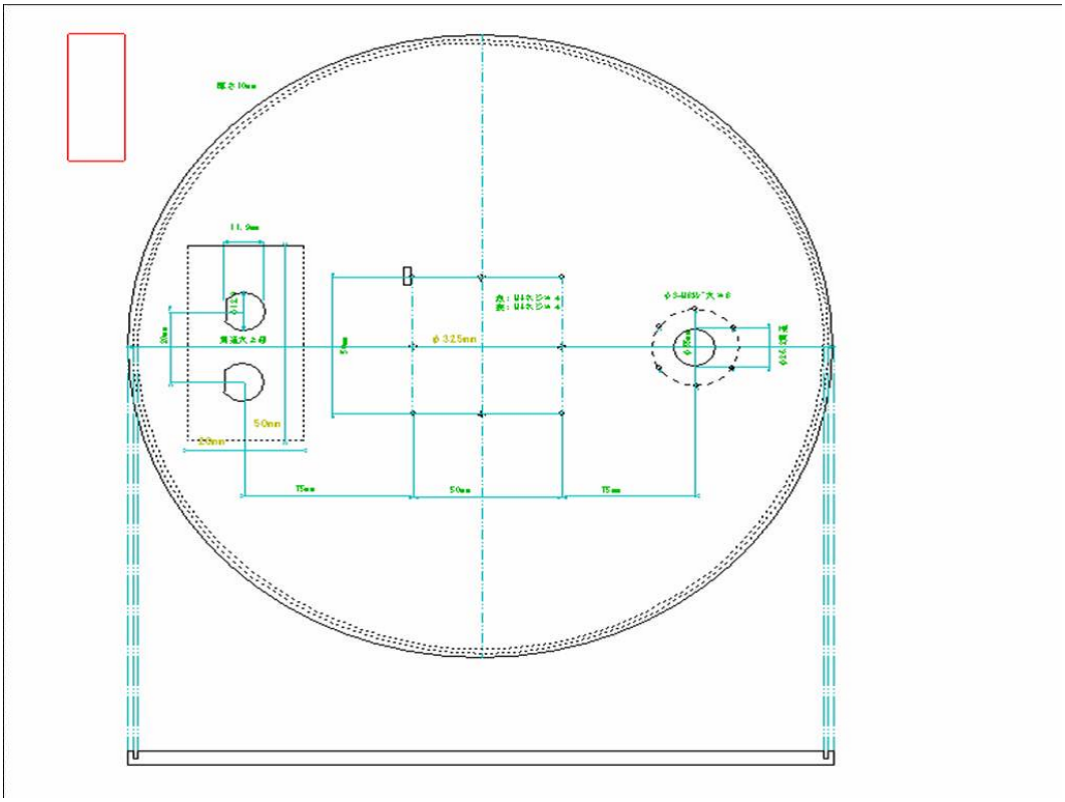


図 2.2: 蓋の設計図

2.1.2 SiPhotoDiode

受光面のP型領域と基盤側のN型領域は、PN接合を形成し光電変換部として働く。Si-PPDの場合はP層は１μm以下の厚さになる。P層とN層の接合部の中性領域は空乏層と呼ばれる。Si-PPDに光が照射され、その光エネルギーがバンドギャップエネルギーよりも大きいと荷電子帯の電子は伝導帯に励起され、もとの荷電子帯に正孔をのこす。この電子-正孔対が空乏層内で生じるとバイアス電場によりそれぞれP層、N層に加速される。このような電子-正孔対は入射光量に比例して発生し、各々電極を通じて取り出せばパルスとして検出される。Si-PPDではダークカウントと呼ばれる、熱的に発生した暗電流によるパルスも発生する。このパルスがダークカウントとして測定され、検出誤差の原因となる。バイアス電圧を高くすると検出効率が上がるが、その分ダークカウントも大きくなる。ダークカウントは温度が低いほど値が小さくなる。

PPD表面の黒い部分は受光面で下図のP層に当たる。α線も空乏層内で生じた電子正孔対はバイアス電場により正極と負極に流れパルスとして検出される。α線はエネルギー損失が大きく空気や物質と反応し直ちにエネルギーを落としてしまうので防塵用の絶縁膜はないものを使った。

PIN － Photo － Diodeの典型的な図を以下に示す。

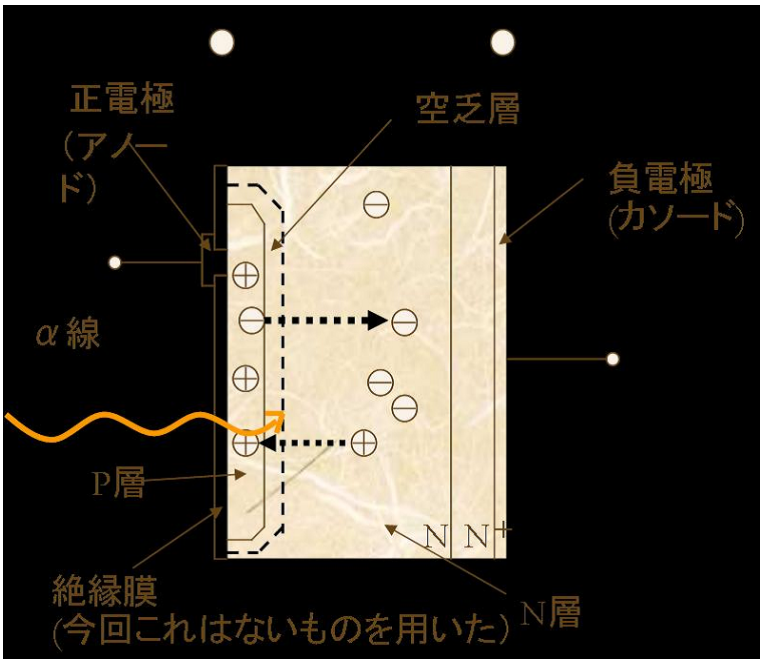


図 2.3: 典型的なPINPhotoDiode

予備実験から本実験まで通じてHAMAMATSU社製のSi-PPDを使用した。型番はS3590-02で有感帯は10mm × 10mm、Windowlessのものを使った。

2.1.3 ClearPulse社　579CsI型アンプ

本実験ではPINを取り付けるアンプを自作したが、予備実験においてはCP社製の既製品アンプModel 579CsI型(プリアンプを含む)を用いた。

図 2.4: CP社579CsI型アンプ

これにはバイアス回路、プリアンプ回路、ポールゼロキャンセル回路、アンプ回路等が含まれる。既製品なので詳細は省略する。

2.1.4 ClearPulse社　Cs515型プリアンプ

後述の通り今回の実験は予備実験から段階的に進めていったのだが、第2段階(Step 2)以降はClearPulse社579CsI型アンプは使用しなかった。代わりにClearPulse社電荷有感型前置増幅器Cs515型(IC素子)を組み込んだ回路を自作し、使用した。

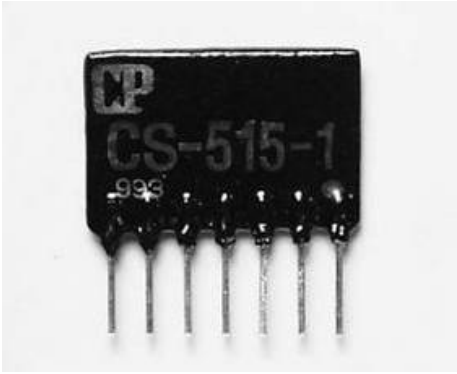


図 2.5: CP社プリアンプ　Cs-515

第3章　本実験に向けて

3.1　本実験への道

今回の実験では、装置の動作確認や環境設定、解析方法を確認しながら、徐々に本実験に近い状態を再現していくという段階的な実験を行った。実験段階は Step1 から Step4 まであり、おおまかな特徴は表 3.1 にまとめた。また、以下の点は各 Step において共通である。

- α線源マントル (後述)、Si-PIN フォトダイオードを使用
- デジタルデータ取得にオシロスコープを使用 (後述)
- PIN の両端に掛かる電位差 (逆バイアス電圧) は 30V

実験段階	PIN位置	PIN両端電圧	579/CsI	備考
Step1	回路上	0V,30V	使用	ｱﾝﾌﾟ製作
Step2	回路上	0V,30V	不使用	回路製作
Step3	ﾁｬﾝﾊﾞｰ内	-1457V, -1427V	不使用	HV回路 製作
Step4	ﾁｬﾝﾊﾞｰ内	-1457V, -1427V	不使用	内部真空

図 3.1: 実験段階表

以下に各々の Step の簡単な説明を行う。

Step1

クリアパルス社製 579/CsI 型プリアンプと自作アンプ回路を用いて、α線源からスペクトルを得る段階。アンプ回路には、LF356N 型アンプを組み込

む。使用電源電圧は30V。データ取得及び解析技術の向上を主な目的とする。
以降のStepでも同線源を用いるので、ここでのスペクトルと以降のStepで取れるスペクトルを比べることで、各Stepでうまくα線が見えてるかを判断する指標になる。

Step2

PINの両端に逆バイアス電圧を印加させるための回路及びプリアンプ、アンプ回路を作成し、これらを用いてα線源からスペクトルを得る段階。プリアンプ回路には、クリアパルス社製ハイブリットIC(電荷有感型前置増幅器):CS515型を組み込む。また、アンプ回路にはLF356N型アンプを組み込む。使用電源電圧は30V。回路の十分な分解能及び計数率を得ることを、主な目的とする。

Step3

-1.5kVの電源電圧を使用、その電圧をPINへの逆バイアス電圧と静電捕集用電圧に分けるHV-dividerを自作し、Step2でのプリアンプ、アンプと合わせて用い、α線源からスペクトルを得る段階。このStepからPIN及び線源は本実験で用いるチャンバー内部に設置する。チャンバー及び回路の環境を整え、HVをかける時の安定性を得る(放電、過電流を無くす)ことと、十分な分解能及び計数率を得ることを目的とする。

Step4

Step3の状態からさらにチャンバー内部を真空にして、α線源からスペクトルを得る段階。このStepでは、本実験で用いる試料の代わりにα線源を使用する以外では、本実験と同じセッティングである。ここで十分な真空度と真空状態の保持を確認し、解析に十分なスペクトルが得られれば、本実験に突入する。

なお、装置の製作、動作確認はStep4まで行ったが実際に結果が出たのはStep2までである。

3.2 α線源:マントル

今回、高エネルギーα線源として図3.2のマントルを用いた。マントルは6-6ナイロン製で本来はランタンの発光体として使用されているが、トリウムが添加されているのでトリウム系列の線源として使用することが出来る。トリウム系列の壊変表を図3.3に示す。

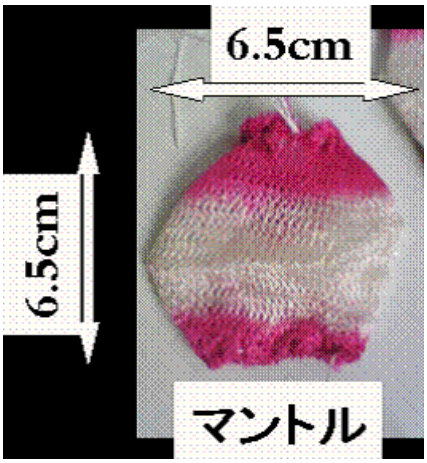


図 3.2: マントル

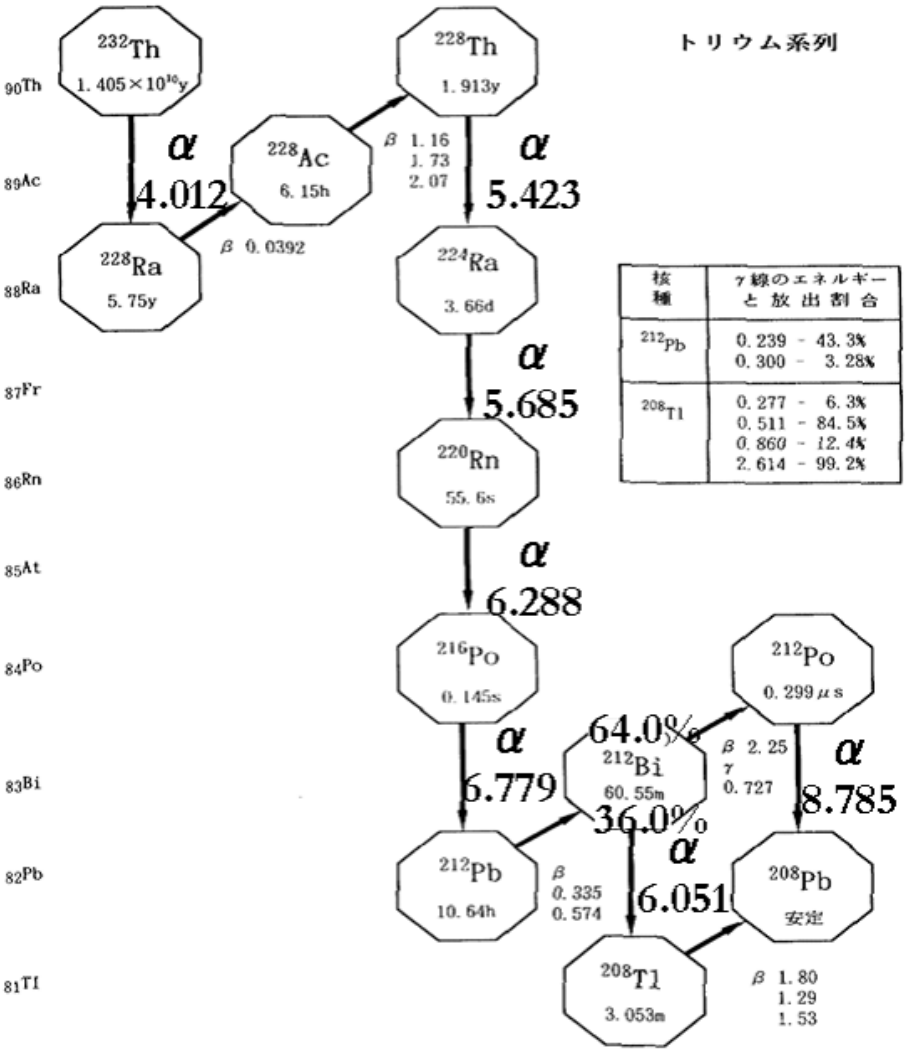


図 3.3: Th 系列壊変表

3.3 データ取得

今回の実験では、データの取得はオシロスコープ (Tektronix 社/TDS3034B 型) を用いて行った。方法は、オシロスコープを PC(Linux) を Lan ケーブルで接続し、デジタルデータ取得のプログラムを神岡グループが使用しているものを改変して使用した。プログラムソースは付録を参照。以下特徴を示す。

- データ取得レート:約 1Hz(製造元曰く最大 30Hz)
- サンプリングレート:10^7[Hz](100[Hz]〜5×10^{11}[Hz] で可変)
- 垂直分解能 (縦のビット数):9Bit(スペックシートより)

データ取得レートは、波形データを得る割合である。

サンプリングレートとは、データを数値化するとき、時間軸に対して数値化を行なう割合のことを指す。1 つの波形データは、10000 個の [横軸:[sec]、縦軸:[V]] の値に数値化される。一方で 1 つの波形の時間幅はオシロスコープのスケール調整によって 20[ns]〜100[s] まで任意に変えられる。よって、上に示したサンプリングレートの可変範囲を得る。

垂直分解能とは、縦軸がどれだけ細かく数値化されるか (オシロスコープで得られる縦軸の範囲を何 Bit で分割してるか) を示す値である。
取得できる縦軸の範囲はオシロスコープの縦軸スケールを調整することで、変えられる。データを取得する際には、すべての波高が縦軸の範囲に収まるようにスケール調整を行う必要がある。

3.4 解析

データ解析には、解析ソフト ROOT を用いた。ここでは、スペクトル取得の方法及びスペクトルの解析方法を述べる。

3.4.1 ファイル形式の変換

オシロスコープから得られたデータは tds 形式であるが、この形式は ROOT では扱えない。そこで tds 形式のデータを ROOT で扱える root 形式のデータに変換する必要がある。ここでも前節データ取得と同様に、プログラムを用いてファイル形式の変換を行った。プログラムソースは付録参照。

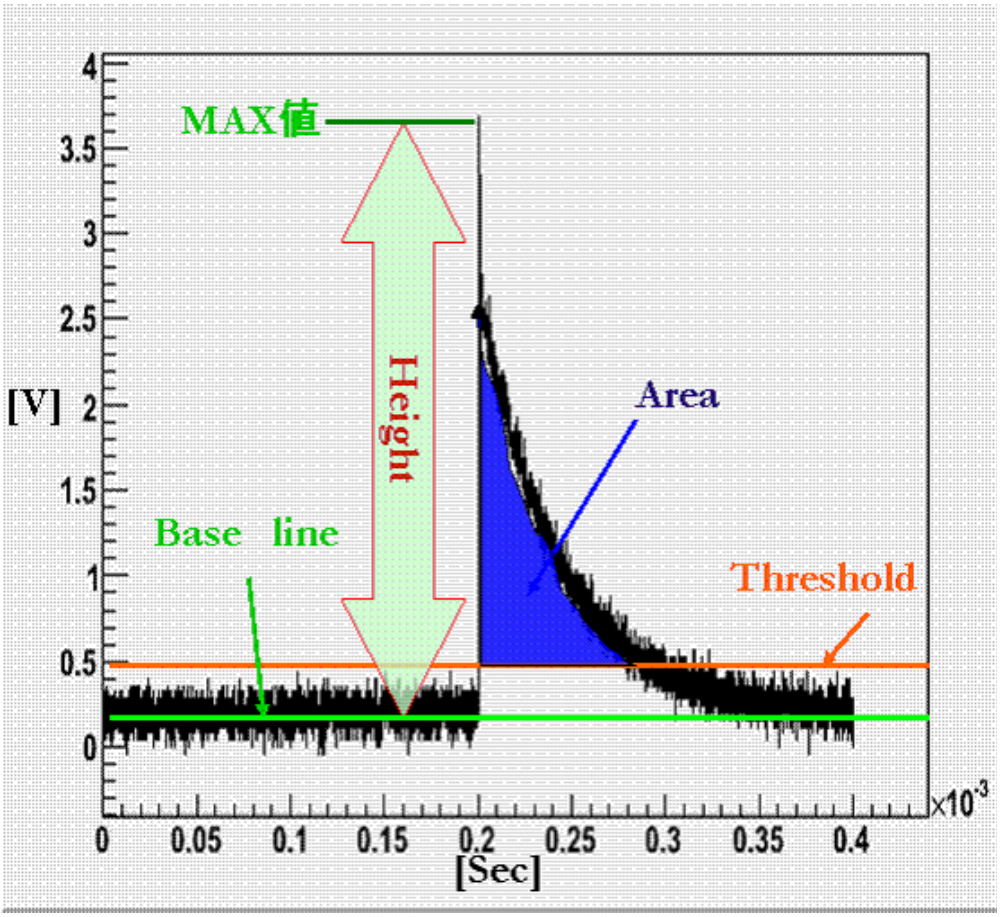


図 3.4: 波形サンプル

3.4.2 スペクトル取得

今回は 2 通りの手法でスペクトルを取得した。

1. パルスの高さのヒストグラム

   図 3.4 における、Height の値をヒストグラムにしてスペクトルを得るのが 1 つ目の手法である。即ち、各データで図の Max の値 (パルスの最高値) と Base line の値の差をヒストグラムにする。図の Base line は、パルスが立つ前 ($0.2 \times 10^{-3}[sec]$ より前) の範囲の平均値に相当する。

2. パルスの面積分のヒストグラム

   図 3.4 における、Area の値をヒストグラムにしてスペクトルを得るのが 2 つ目の手法である。即ち、各データで図の Threshold の値を閾値として、それを超える値の範囲を積分した時の値をヒストグラムにする。Threshold の

値は、Base line の値から適当な値 (パルスが立つ前の部分で積分しないように定めた値) を加えたものである。

2 つの手法のうち分解能が良い手法を用いるべきなのだが、現段階でどちらが良いか判断出来なかったため、2 つ得るスペクトルのそれぞれについて解析を行った。(プログラムソースは、付録を参照)

3.4.3 解析方法

α 線源マントルに含まれる、トリウム系列の放射線核種が永続平衡の状態にあることを仮定し、それぞれの崩壊 α 線のピークに対してガウシアンで重ね合わせると、図 3.5 のようになる。ちなみにそれぞれのガウシアンは一般的な分解能であると思われる値 (FWHM:352.5[KeV]) を仮定した。

実際に取れるスペクトルもこのようになると期待できるので、実際のスペクトルと図 3.5 の一致の程度をみることで解析を行う。

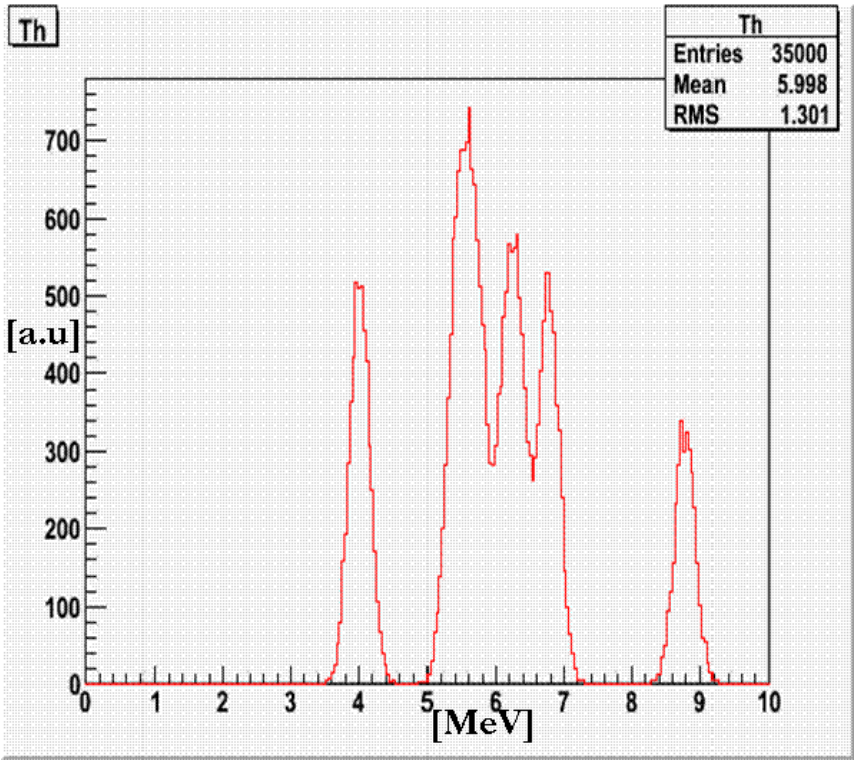


図 3.5: Th 系列ガウシアン重ね書き

3.4.4 エネルギー較正

得られた各スペクトルのピークが Gauss 分布に従うとして、fitting を行う。各スペクトルの x 軸に対し、Gaussian Function は

$$G(x) = a \exp\left\{-\frac{1}{2}\left(\frac{x-b}{c}\right)^2\right\} \tag{3.1}$$

である。

トリウム系列壊変表を参照して、GaussianFitting で得られた線源の各ピークの中心値 b に対応するエネルギーの値 (MeV) を特定し、各スペクトルで [x 軸] 対 [MeV] のデータを作成する。これを一次関数と仮定し、

$$f(x) = d * x + e \tag{3.2}$$

として、直線で Fitting を行い d と e の値を求めれば、各スペクトルの横軸をエネルギーに直すことが出来る。この作業をエネルギー較正という。

解析の際には、エネルギー較正を行ったスペクトルと前小節の図 3.5 の重ね書きを行う。

3.5 予備実験 Step1

3.5.1 測定

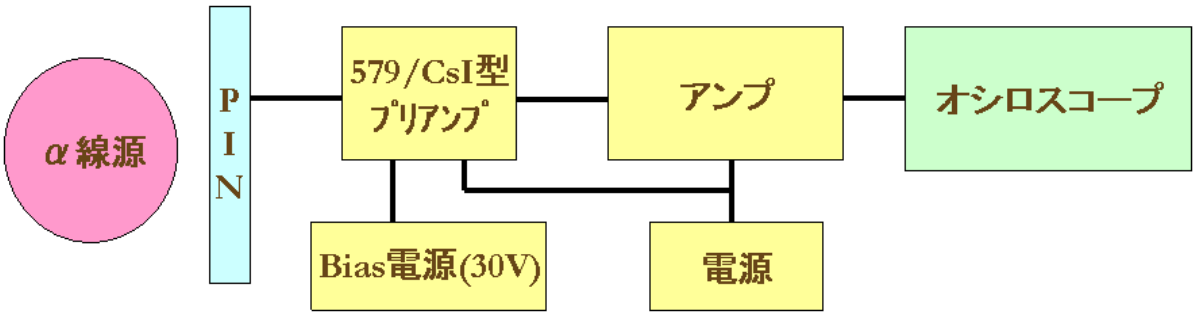


図 3.6: Step1:Setup

以下の手順を踏んで測定を行った。

1. Setup を図 3.6 に示すように行った。図 3.7 は Step1 で使用したクリアパルス社製 579/CsI 型プリアンプである。

2. オシロスコープのスケールを調整した後、オシロスコープと LAN ケーブルで接続された PC のプログラムを走らせ、十分な時間データを取得した。

また、図 3.8 に自作したアンプの回路図を示す。これは非反転増幅器であり理論的に 50 倍まで可変で増倍できる。今回は増倍率を最大の 50 倍に設定した。

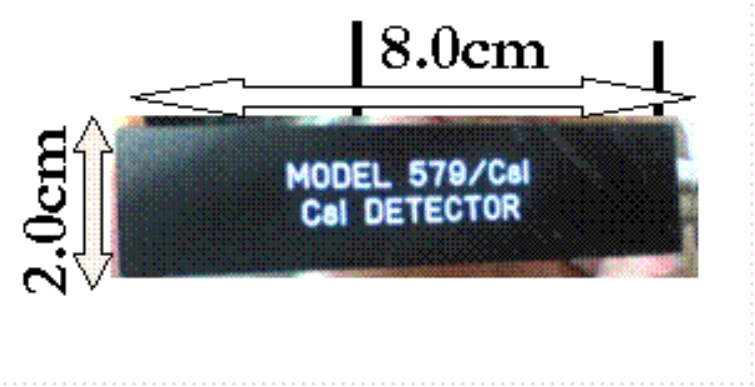


図 3.7: CP 社製 579/CsI 型

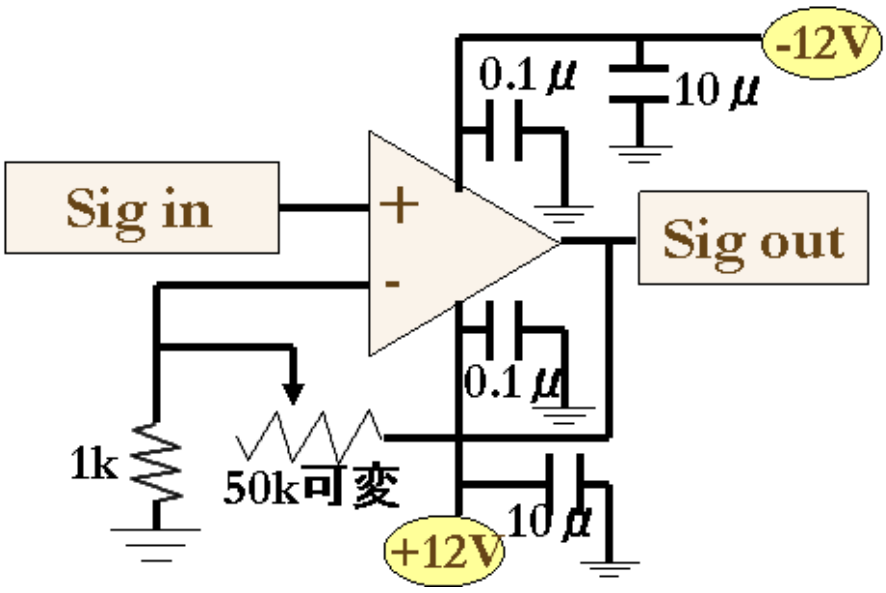


図 3.8: 自作アンプ

3.5.2 結果

考察 1

実験により得られたスペクトルに、図 3.5 で仮定したトリウム系列のスペクトルを重ね書きし、図 3.9 に示す。エネルギー較正は暫定的に実験で得たスペクトルの 1 番高いところのピークを、トリウム系列崩壊 α 線の最も高いエネルギーの中心値となるように行った。図を見てみると、測定で得たスペクトルは 8MeV 付近にピークがあるのでこれでは説明が付かない。

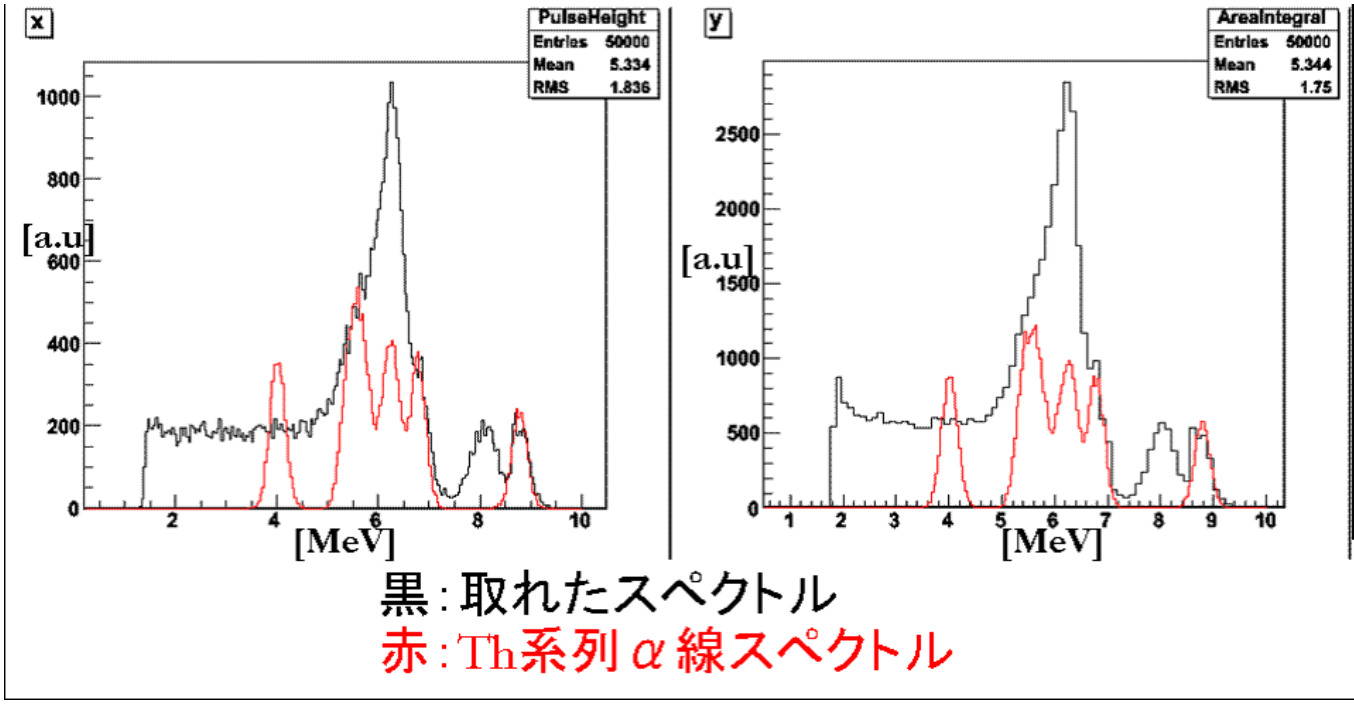


図 3.9: Step1 結果 1

考察 2

そこで、もう 1 つの仮定を取り入れた解析を行った。その仮定とは、トリウム系列以外に、ウラン系列の放射線核種がマントルに含まれているとしたものである。第 1 章で述べたとおり、ウラン系列の放射性核種は一般に物質中に含まれているので可能性は考えられる。

ウラン系列含有量をトリウム系列の半分と仮定し、前節の『解析方法』の小節と同様の方法で、ウラン系列から得るスペクトルを図 3.5 のスペクトルに足し合わせ、測定で得たスペクトルに重ね書きした。図 3.10 における青色のスペクトルがトリウム及びウラン系列のガウシアンでの重ね合わせである。

エネルギー較正は高いエネルギーのピーク 3 つに対して行った。Height 値のスペクトルに対するエネルギー較正の直線 Fitting のグラフを図 3.11 に、同様に Area 値のものを図 3.12 に、直線 Fitting 結果の値を表 3.1 に示す。

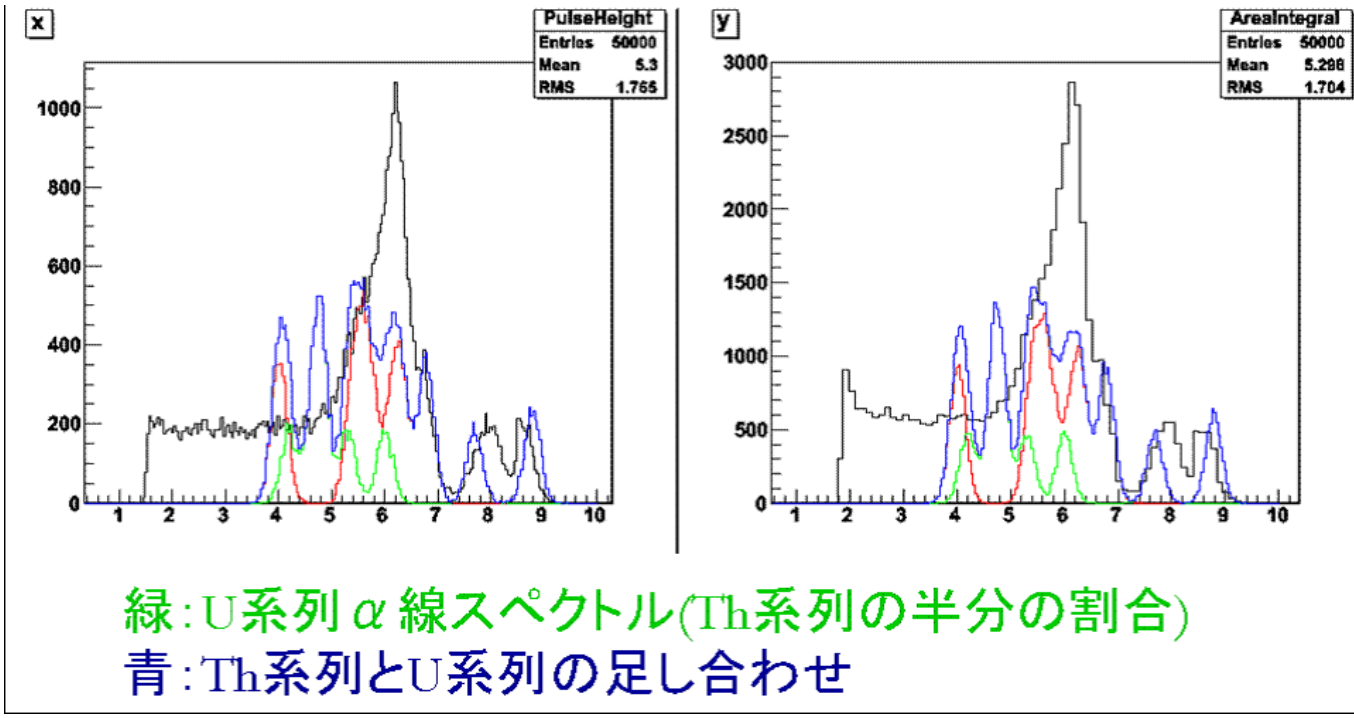


図 3.10: Step1 結果 2

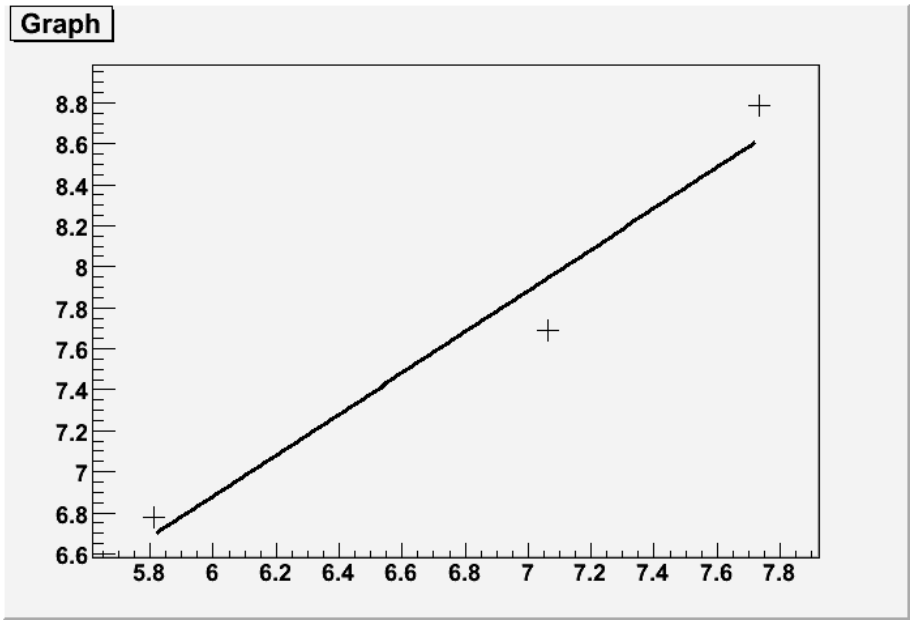


図 3.11: Height エネルギー較正

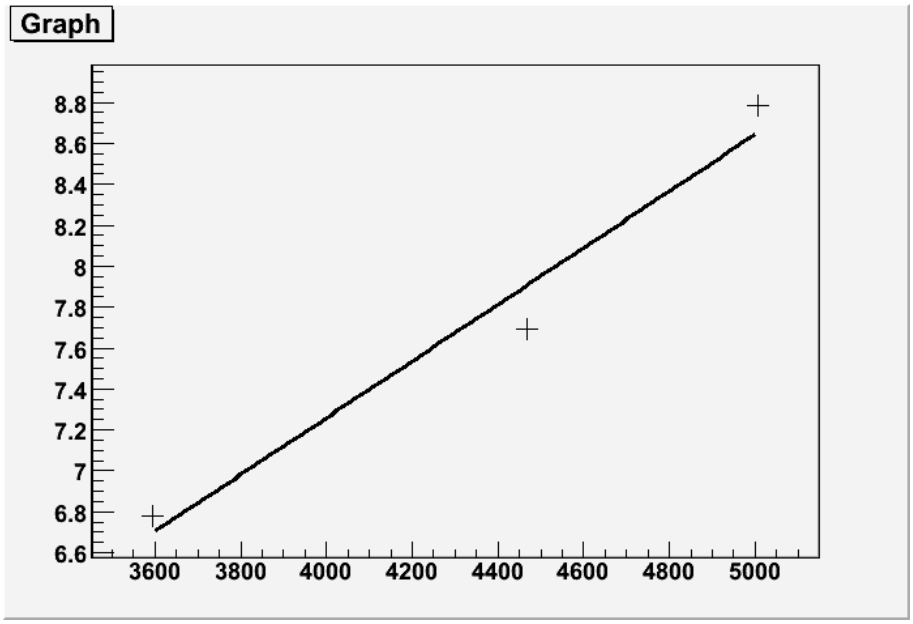


図 3.12: Area エネルギー較正

	d(傾き)	e(切片)
Height Cal	1.00±0.725	0.860±5.01
Area Cal	$1.38\times10^{-3}\pm9.92\times10^{-4}$	1.73±4.36

表 3.1: エネルギー較正表

図を見てみると、前の検証では明らかに合わなかった 8MeV 付近のピークが今回では近くに見え、より実験で得たスペクトルに近くなった。しかし、

1. 6MeV 付近に非常に大きなピークがある。
2. エネルギー較正の Error の値が大きく、ピークがずれている。
3. 5MeV 以下のピークがない。

と言った疑問も残る。原因として、

- バックグラウンドのピーク

  バックグラウンドがピークになっていて、実際には見えないはずのピークが見えている可能性がある。

- エネルギー損失の影響

  α線の空気中及びPINのP層でのエネルギー損失を考慮してないので、実際はスペクトルがずれる。さらには、マントル表面からα線が放出されるときにもエネルギー損失が起きる。(α線のマントル内部での最大飛程はわずか80 μ m程度) これにより低エネルギー(5MeV以下) α線のピークが見えてないのかもしれない。

- 崩壊β線のピーク

  トリウム系列崩壊β線には3MeV以上のものが存在する。前述のとおりα線はエネルギー損失が大きいので、このピークがα線の6MeV付近でのピークに重なる可能性がある。(ほかにも2MeV以上のいくつかの崩壊β線がα線の3〜5MeVにかぶっている可能性もある)

といったことが考えられるが、時間の関係もありここは検証出来ないまま次のStepへと進んだ。

3.6 予備実験Step2

3.6.1 測定

図 3.13: Step2:Setup

データ取得を行ったオシロスコープやパソコンはStep1と同じようにセッティングした。

1．Setupを図3.11に示すように行った。図中のCS515は実験装置の項目で述べたCP社製のプリアンプである。

2．Step1と同様に十分な時間データを取得した。

Step2ではPINに逆バイアスをかける回路、またCS515を組み込んだプリアンプ回路、また反転増幅回路を自作した。はじめはバイアス回路とアンプ&プリアンプ回路は別々に作ったが、ノイズ対策の一つとして同じ基板に載せた。非反転アンプはStep1のものと同様可変抵抗を用い、増倍率を可変にした。

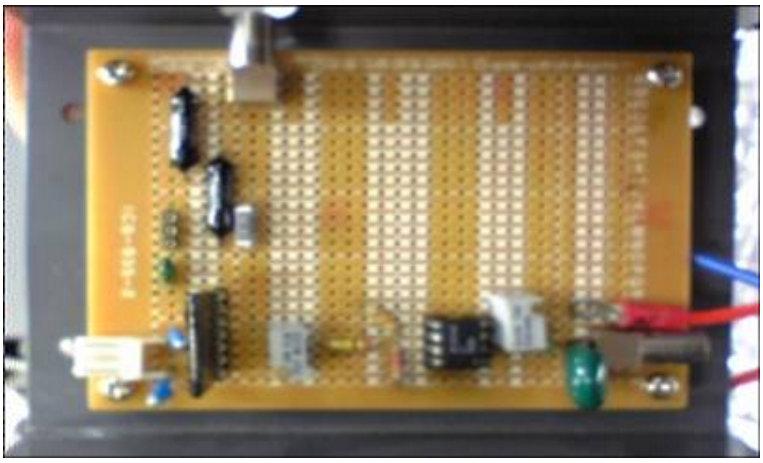


図 3.14: bias & preamp & amp

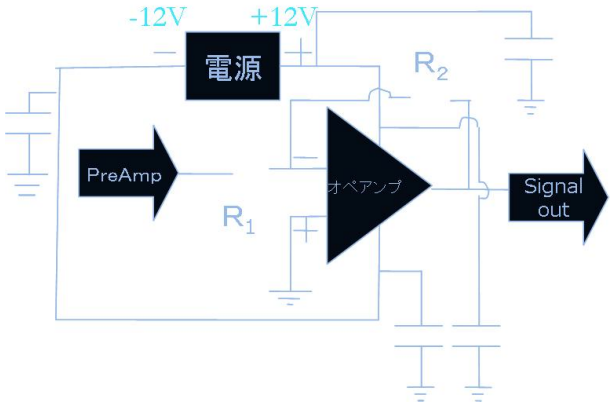


図 3.15: 反転増幅器の回路図

3.6.2 結果

実験で得られたスペクトルは以下のようになった。Step 1 のとき同様 Th 系列のスペクトルを重ねてみてもエネルギー較正を出来るようなピークが得られなかった。この Step のスペクトルは試行錯誤を重ねたため本当はもっとたくさんあるのだが結果としてどれ一つうまく較正できなかったので一例を示すにとどめる。ピークが得られなかった原因は Step1 と同様の考察が考えられるがこの実験を行ったときに単に線源を多くするためにマントルを重ねてしまっていたので、それによるピークの崩れがあるかもしれないと発表会の直前に気づいた。

スペクトルの左下のグラフは横軸に pulseheight、縦軸に areaintegral を取った 2 次元ヒストグラムである。これは時定数の違うノイズの混ざり具合を見るためのもので大体直線になっていればおかしな波形は多くないと分かる。この解析はスペクトルがうまく取れない原因を探るための解析としてこの Step からやり始めた。

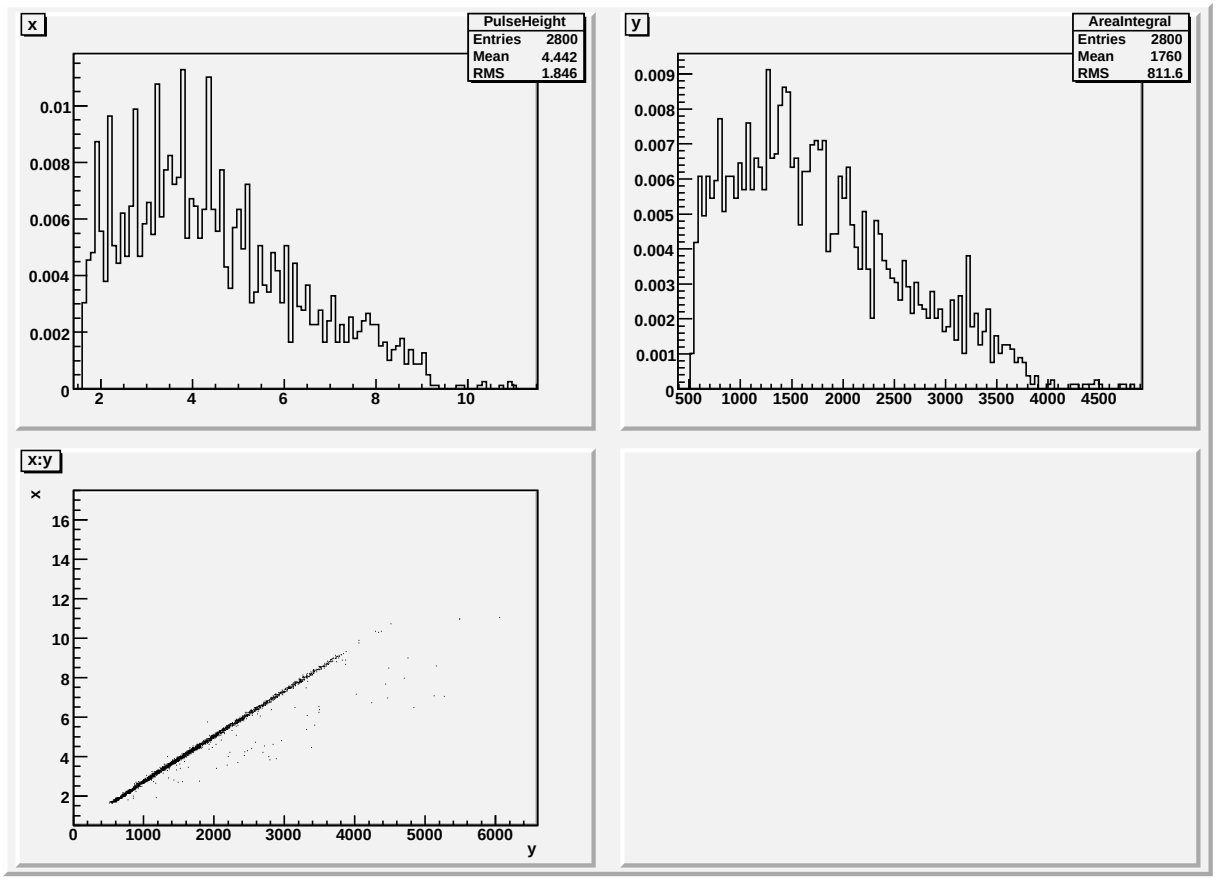


図 3.16: Step2 取得スペクトル

3.7 予備実験 Step3

3.7.1 測定

データ取得を行ったオシロスコープやパソコンは Step1 と同じようにセッティングした。

自作した Bias-HVdivider の回路図を以下に載せる。この回路では PIN を-1500V 程度に印加すると同時に約 30V 逆バイアスがかかるように設計した。

1.Setup を図に示すように行った。

2. 回路をすべて Chamber の蓋部分に設置した。
3. PIN を設置せずに HV による放電や不具合が起きないか確認した。
4. PIN と α 線源を Chamber の蓋の下部に設置しスペクトルを取得した。

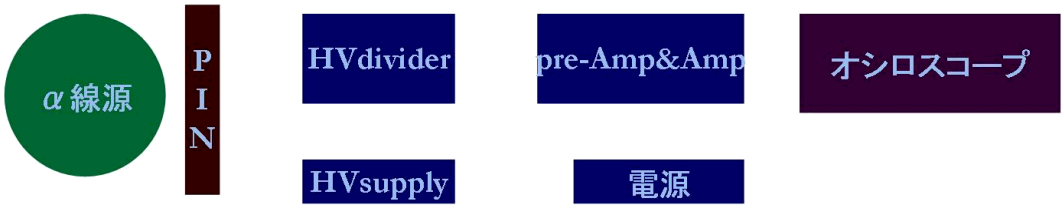


図 3.17: Step3:Setup

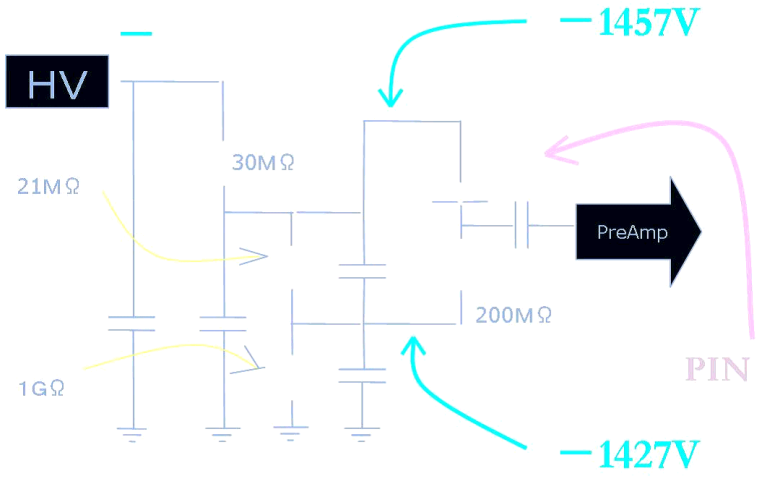


図 3.18: HV-divider 回路図

3.7.2 静電遮蔽箱

この Step では静電遮蔽のためのパンチメタルの箱を作成した。Chamber は金属製だが表面の抵抗を測定すると 0 とはみなせない抵抗値があった。これは装置の項で述べた電解研磨に関係していると思われる。実際調べてみたところ電解研磨により金属表面に不動態膜が形成されていることが分かった。

作成した静電遮蔽箱は底面が 39cm 角の正方形で高さが 38cm 程度の箱で、蓋は金属の代わりにダンボールにアルミホイルを貼り付けた蓋を用いた。

パンチメタルの遮蔽では不十分だったので表面にアルミホイルを貼り、さらにその上から遮光することでかなりのノイズの減少に成功した。

3.7.3 結果

Step3 でも Step2 と同様較正を行えるようなスペクトルは得られなかった。またバックグラウンドを差し引いて解析を行うため線源の on/off のスペクトル取得を行ってみたところ、バックグラウンドスペクトル内にはっきりとしたピークが見

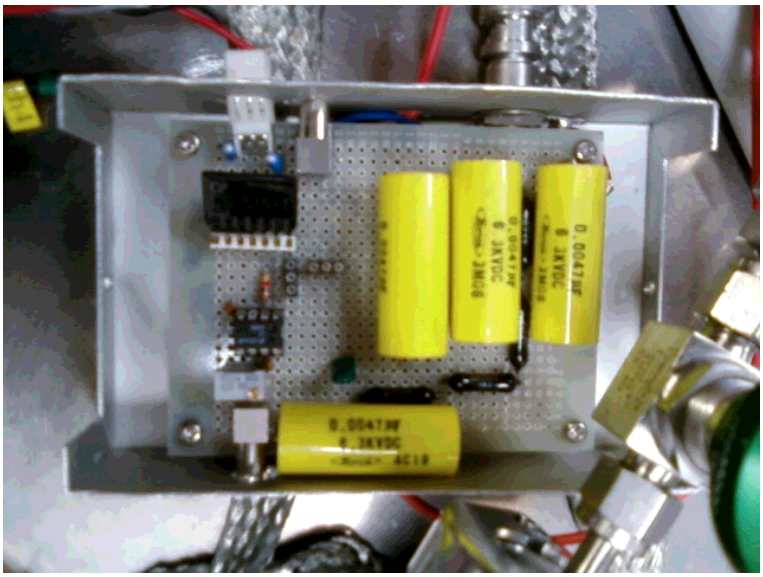


図 3.19: HV-divider

られ、原因の究明に前の段階に立ち返って実験を行うことになった。

3.8 予備実験 Step4

3.8.1 測定

Step4 では Step3 と同じセッティングでさらに Chamber 内部を真空状態にして測定を行う予定だった。発表会直前に強行して実験を行ったところ原因不詳の不具合が生じ、プリアンプを壊してしまった。

スペクトルも結局取得にいたらなかった。

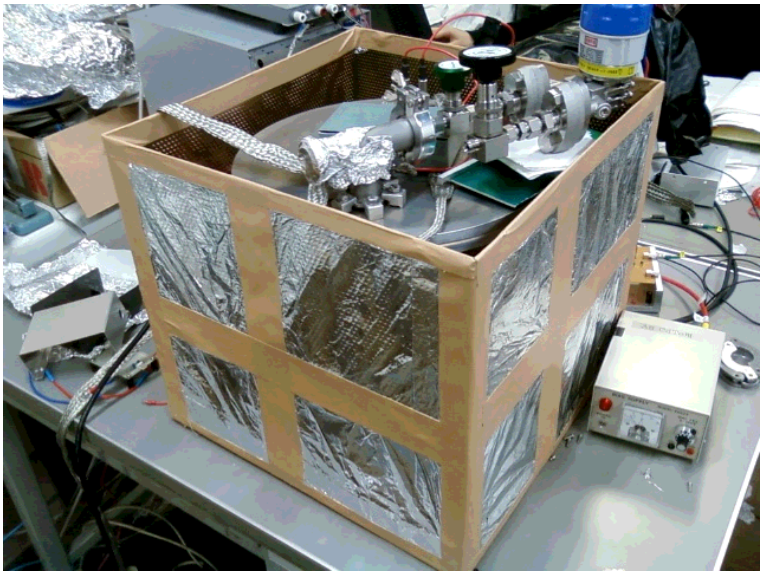


図 3.20: 遮蔽箱

第4章　まとめ

今回の実験ではRn-detecterを目標に進めていったが本実験にまで到達しなかったので残った課題をあげてまとめに代えることにする。

Step1では比較的ピークのはっきりしたスペクトルを取得できたが、Step3のBGのピークが実はこの段階にも現れることが後にわかり、さらに考察を要することがわかった。

α線源としてのランタンは設置の状況によって得られるスペクトルが繊細に変わるので、再現性のある設置の仕方を考えるべきだった。（厚みはなるべく薄く、角度はなるべく付かないように。）

回路によっては振動に弱いもの、アースの甘いものがあったように思う。

Step3以降は原因の分からない不安定性に悩まされた。主にPreAmpの電源電流にその不安定性は現れた。HVをかけるとPreAmpの電源電流が安定するのに時間が掛かったり、遮蔽箱の蓋をつけたりはずしたりするとPreAmpの電源電流に著しい影響があった。

Step4の真空を引いてからはさらに大きく電源電流がゆれ、またHV電源から流れる電流にも影響があった。このときは真空を引く前にアンプの電源を入れており、そのまま真空に引いていったところ許容電流以上の電流が流れた。そのため以降は真空を引くのも、HVをかけるのもアンプの電源電流は落として行った。

時間の関係でいたらなかったがBGの特定と再現性の追及によりStep2、3以降の実験も結果が得られると思われる。

今回の実験では装置の作成を多く行ったが、作成に当たっては再現性という点を強く意識することが必要だと感じた。

第5章　付録

5.1　プログラムソース集

5.1.1　オシロスコープからデータ取得するプログラム

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <string.h>
#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <time.h>

int init_connect(char *,int );
int url_encode(char *, char *, int);
int setStopAfter(char *,char *);
int getStopAfter(char *,char *, int );
int setSingleSequence(char *);
int isSingleSequence(char *);
int setRunStop(char *);
int isRunStop(char *);
int setState(char *,char *);
int getState(char *,char *, int );
int setStateRun(char *);
int isStateRun(char *);
int setStateStop(char *);
int isStateStop(char *);
int sendCommand(char *,char *, char *, int );
int getWaveform(char *,int , char *, int ,int);
int strip_http_headers(char *, int);
```

```
int debug=0;
int init_connect(char *remotehostname,int port){
  int     sock;
  static  struct  sockaddr_in sock_name;
  struct  hostent             hostentstruct;  /* Storage for hostent data.  */
  struct  hostent             *hostentptr;    /* Pointer to hostent data.   */
  static  char                hostname[256];  /* Name of local host.        */
  int     retval;                             /* helpful for debugging */
  if ((sock = socket (AF_INET, SOCK_STREAM, 0)) == -1) {
perror( "socket");
return -1;
  }
  if ((hostentptr = gethostbyname (remotehostname)) == NULL) {
perror( "gethostbyname");
close(sock);
return -1;
  }
  hostentstruct = *hostentptr;
  sock_name.sin_family = hostentstruct.h_addrtype;
  sock_name.sin_port = htons(port);
  sock_name.sin_addr = * ((struct in_addr *) hostentstruct.h_addr);
  retval = connect(sock,(struct sockaddr *)(&sock_name), sizeof (sock_name));
  if (retval != 0) {
perror("connect");
close(sock);
return -1;
  }
  return sock;
}

int url_encode(char *src, char *dst, int len){
  int i;
  int srclen = strlen(src);
  if (len < srclen*3){
return -1; /* too short to store everything */
  }
  for(i = 0 ; i < srclen ; i++){
snprintf((dst+(i*3)),4,"%%.2x",src[i]);
  }
```

```
  return (srclen*3);
}

int setStopAfter(char *hostname,char *strMode)
{
  int ret;
  char cmd_str[1024]="ACQuire:STOPAfter \0";
  if (strlen(strMode) > 1000){
    /** return NULL;**/
    return 0;
  }
  strncat(cmd_str, strMode, 1024);
  ret = sendCommand(hostname, cmd_str, NULL, 0);

  return ret;

}

int getStopAfter(char *hostname, char *buf, int maxlen)
{
  int ret;
  char cmd_str[1024]="ACQuire:STOPAfter?\0";
  ret = sendCommand(hostname, cmd_str, buf, maxlen);
  return ret;
}

int setSingleSequence(char *hostname)
{
  int ret;
  ret = setStopAfter(hostname, "SEQuence\0");
}

int isSingleSequence(char *hostname)
{
  int ret;
  char status[1024];
  int  len = 1024;
  ret = getStopAfter(hostname, status, len);
  if (strstr(status,"SEQ") == status){
```

```
printf("In the single SEQ.\n");
return 1;
  }else{
printf("Not in the single SEQ.\n");
return 0;
  }

}

int setRunStop(char *hostname)
{
  int ret;
  ret = setStopAfter(hostname, "RUNSTop\0");

  return ret;

}

int isRunStop(char *hostname)
{
  int ret;
  char status[1024];
  int  len = 1024;

  ret = getStopAfter(hostname, status, len);

  if (strstr(status,"RUNSTop") != status){
return 1;
  }else
return 0;
}

int setState(char *hostname, char *state)
{
  int ret;
  char cmd_str[1024]="ACQuire:STATE \0";

  if (strlen(state) > 1000){
    /** return NULL;**/
```

```
    return 0;
  }
  strncat(cmd_str, state, 1024);
  ret = sendCommand(hostname, cmd_str, NULL, 0);

  return ret;

}

int
getState(char *hostname, char *buf, int maxlen)
{
  int ret;
  char cmd_str[1024]="ACQuire:STATE?\0";

  ret = sendCommand(hostname, cmd_str, buf, maxlen);

  return ret;

}

int
setStateRun(char *hostname)
{
  int ret;

  ret = setState(hostname, "RUN");

  return ret;

}

int
isStateRun(char *hostname)
{
  int ret;
  char status[1024];
  int  len = 1024;
```

```
  ret = getState(hostname, status, len);

  if (  (strstr(status,"1") == status)
  ||(strstr(status,"RUN") == status)
  ||(strstr(status,"ON") == status)){
return 1;
  }else
return 0;
}

int
setStateStop(char *hostname)
{
  int ret;
  ret = setState(hostname, "STOP");
  return ret;
}

int
isStateStop(char *hostname)
{
  int ret;
  char status[1024];
  int  len = 1024;

  ret = getState(hostname, status, len);

  if (  (strstr(status,"0") == status)
  ||(strstr(status,"STOP") == status)
  ||(strstr(status,"OFF") == status)){
return 1;
  }else
return 0;
}

int
sendCommand(char *hostname,char *cmd_str, char *buf, int maxlen)
{
  int fd;
```

```
  char url_string[4096];
  char tmp_str[4096];
  char ack='\n';
  char *msg,*lasts;

  int ret,len;

  int flag;



#define FALSE 0
#define TRUE  1

  fd = init_connect(hostname,80);
  ret = url_encode(cmd_str, tmp_str, 4000);
  snprintf(url_string,4096,"GET /?command=%s HTTP/1.1\n",tmp_str);
  ret = send(fd,url_string,strlen(url_string),0);

  if (ret < 0){
perror("Failed to send URL request.");
return ret;
  }

  memset(tmp_str,0,sizeof(tmp_str));
  ret = recv(fd,tmp_str,4096,0);

  len = 0;
  if (strstr(tmp_str,"Continue") != NULL){
    send(fd,&ack,1,0);
    while( (4096-len) > 0){
      ret = recv(fd,tmp_str+len,4096-len,0);
      if (ret <=0 ) break;
      len += ret;
    }
  }
  flag = FALSE;

  msg = strtok_r(tmp_str,"\r\n",&lasts);
```

```
  while(flag == FALSE){
    if (  (strstr(msg,"HTTP/1.1") == msg)
  ||(strstr(msg,"Date:")    == msg)
  ||(strlen(msg)            ==   0)){
      msg = strtok_r(NULL,"\r\n",&lasts);
      if (msg == NULL){
flag = TRUE;
ret = 0;
  }
}else{
  flag = TRUE;
  ret  = strlen(msg);
}
  }

  if (ret > maxlen){
len = maxlen;
  }else{
len = ret;
  }

  if (len >0){
memcpy(buf,msg,len);
  }
  shutdown(fd,2);
   close(fd);
  return ret;
}

int getWaveform(char *hostname,int ch, char *buf, int maxlen,int outmode)
{

  int ret=0;
  int len=0;
  int fd;
  int i;
  char ack='\n';
  int content_length=0;
#define NCMD 6
```

```
#define CMDLEN 256
  char command_string[NCMD*CMDLEN];
  char command_strings[NCMD][CMDLEN]
    = {"POST /getwmf.isf HTTP/1.1\r\n\0",
       "Host:\0"
       "Connection: close\r\n\r\n",
       "command=select:ch%d on\r\n\0",
       "command=save:waveform:fileformat internal\r\n\0",
       "wfmsend=Get\r\n\0"};
  memset(command_string,0,sizeof(command_string));
  fd = init_connect(hostname,80);
  if(debug)   fprintf(stderr,"fd=%d\n",fd);
  snprintf(command_strings[1],256,"Host: %s\r\n\0",hostname);
  snprintf(command_strings[3],256,"command=select:CH%d on\r\n\0",ch);
  if(outmode)
  snprintf(command_strings[4],256,"command=save:waveform:fileformat spreadsheet\r\
  else
  snprintf(command_strings[4],256,"command=save:waveform:fileformat internal\r\n\0
  snprintf(command_strings[5],256,"wfmsend=Get\r\n\0");

  for(i = 3 ; i < NCMD ; i++){
content_length+=strlen(command_strings[i]);
  }
  snprintf(command_strings[2],256,"Connection: close\r\n\r\n");
  snprintf(command_string,sizeof(command_string),"%s%s%s%s%s%s",command_strings[0]

  if(debug)fprintf(stderr,"%s",command_string);

  ret = send(fd,command_string,strlen(command_string),0);
  if (ret < 0){
perror("Failed to send URL request.");
return ret;
  }
  else if(debug) fprintf(stderr,"communication with the host: OK\n");
  if(debug)  fprintf(stderr,"maxlen=%d len=%d\n",maxlen,len);
  while( (maxlen-len) > 0){
ret = recv(fd,buf+len,maxlen-len,0);
if (ret <=0 ) break;
if (strstr(buf+len,"Continue") != NULL){
```

```
  send(fd,&ack,1,0);
}
len += ret;
  }
  if (ret < 0){
perror("Failed to retrive waveform");
return ret;
  }
  if(debug) fprintf(stderr,"Received length: %d\n",len);
  ret = strip_http_headers(buf,len);
  if(debug)  fprintf(stderr,"Data length: %d\n",ret);
  shutdown(fd,2);
  close(fd);
  return ret;
}

int
strip_http_headers(char *buf, int len)
{

  char *data_ptr = buf;
  int data_len   = len;
  char *ctmp,*lasts;
  char *httpmsgs[2]={NULL,NULL};
  httpmsgs[0] = strstr(data_ptr,"HTTP/1.1 200 OK");
  httpmsgs[1] = strstr(data_ptr,"HTTP/1.1 100 Continue");
  if (httpmsgs[0] == NULL){
fprintf(stderr,"There seems to be some trouble in the response.\n");
  }
  data_ptr = (httpmsgs[0]>httpmsgs[1])?httpmsgs[0]:httpmsgs[1];
  data_ptr = strstr(data_ptr,"\r\n\r\n");
  data_ptr+=4;
  data_len -= (data_ptr-buf);
  memmove(buf,data_ptr,data_len);
  return data_len;
}

int main(int argc, char **argv)
{
```

```
int ret;
int fd[3];
int i,thisch;
int num_event =1;
int n_channel = 4;
int outmode =0;
char output_fname[1024]="TDS";
char output_fname_head[1024]="TDS";
char hostname[256]="TDS\0";
char str[1000];
char buf[1024*1024];

fprintf(stderr,"***read_TDS***\n");
fprintf(stderr,"read_TDS ch(1-4) num [out_filename] [1/0 (output format 1 for as
/**commad line parameters**/
if(argc>1)    num_event=atoi(argv[1]);
if(argc>2)    n_channel=atoi(argv[2]);
if(argc>3)    sprintf(output_fname_head,"%s",argv[3]);
if(argc>4)    outmode=atoi(argv[4]);
if(argc>5)    strcpy(hostname,argv[5]);
fprintf(stderr,"num of events=%d\n",num_event);
fprintf(stderr,"num of channels=%d\n",n_channel);
if(outmode)
fprintf(stderr,"output file=%s_*_ch?.csv (ascii mode)\n",output_fname_head);
else
fprintf(stderr,"output file=%s_*_ch?.tds (TDS internal mode)\n",output_fname_hea
fprintf(stderr,"hostname=%s\n\n",hostname);

/**initialize**/
if(debug)  printf("Set the state to STOP.\n");
ret = setStateStop(hostname);
if(debug) printf("Waiting for the state to be STOPPED\n");
ret = 0;
while (ret == 0){
  // usleep(0);
  ret = isStateStop(hostname);
}
if(debug) printf("Set the state to single seq.\n");
```

```
  ret = setSingleSequence(hostname);

  /**data loop**/
 int start = time(0);
  for (i = 0 ; i < num_event ; i++){
    fprintf(stderr,"event %d/%d\r",i+1,num_event);

    if(debug)printf("Start the run\n");
    ret = setStateRun(hostname);
    if(debug)printf("Waiting for the trigger\n");
    if(debug)printf("(Waiting for the state to be STOPPED)\n");
    ret = isStateStop(hostname);
    while (ret == 0){
      // usleep(0);
      ret = isStateStop(hostname);
      if(debug) printf(".");
      fflush(stdout);
    }
    if (debug)    printf("\n");
    if(debug) printf("Retriving the waveform.\r");

    for(thisch=0;thisch<n_channel;thisch++){
      if(debug)fprintf(stderr,"%d",thisch);
      if(outmode)
sprintf(output_fname,"%s_%d_ch%d.cvs",output_fname_head,i,thisch+1);

      else
sprintf(output_fname,"%s_%d_ch%d.tds",output_fname_head,i,thisch+1);
      fd[0] = open(output_fname,O_RDWR | O_CREAT,0644);
      if (fd[0] < 0){
perror("Failed to open output file.");
exit(1) ;
      }
      ret = getWaveform(hostname, thisch+1, buf, 1024*1024,outmode);
      ret = write(fd[0],buf,ret);
    }
    if(debug)    fprintf(stderr,"%s ",buf);

    if(debug)fprintf(stderr,"event number = %d\n",i+1);
```

```
    if (ret < 0){
      perror("Failed to dump data to disk..");
      close(fd[0]);
      exit(1);
    }
    if (debug)    fprintf(stderr,"clsing %d\n",i);
    close(fd[0]);


    if (i == 9) { int goon = time(0);
      int ten = difftime(goon,start);
      fprintf(stderr,"\n%dsec at %d events\n",ten,i+1);
  }
if (i%100 == 99){ int goon = time(0);
    int hundret = difftime(goon,start);
    fprintf(stderr,"\n%dsec at %d events\n",hundret,i+1);
  }


  }
int end = time(0);
 int diff = difftime(end,start);
 fprintf(stderr,"\ndone!!!.\nworktime is %d\n",diff);

    exit(0);

}
```

5.1.2 ファイル形式変換のプログラム

ソースは2つあり。以下”convert.cxx”のソース。

```
#include <stdio.h>
#include <stdlib.h>
#include "COscillo.h"
#include <TGraph.h>
#include <TFile.h>
```

```
int main(int argc, char *argv[]){
  int n_event = 1;
  int n_channel=4;
  char input_fname[1024]="TDS";

  fprintf(stderr,"***convert***\n");
  fprintf(stderr,"convert ch(1-4) num [in_filename(head)]\n");
  /**commad line parameters**/
  if(argc>1)    n_event=atoi(argv[1]);
  if(argc>2)    n_channel=atoi(argv[2]);
  if(argc>3)    sprintf(input_fname,"%s",argv[3]);
  fprintf(stderr,"num of events:%d\n",n_event);
  fprintf(stderr,"num of channel:%d\n",n_channel);
  fprintf(stderr,"input file:%s_*_ch?.tds\n",input_fname);
  fprintf(stderr,"output file:%s_*_ch?.root\n",input_fname);

  COscillo oscillo("dummy");
  TGraph gr;
  oscillo.ConvertToTGraph(input_fname, n_event, n_channel, &gr);
  //oscillo.ConvertToTGraph(argv[3], &gr);
  oscillo.Close();
  fprintf(stderr,"\ndone.\n");
  //  TFile file("out.root", "recreate");
  //gr.Write("graph");
  //file.Close();
    return 0;
}
```

ここから”COscillo.h”のソース

```
#ifndef COscillo_H
#define COcsillo_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
```

```
#include <unistd.h>

#include "TSocket.h"
#include "TString.h"
#include "TGraph.h"
#include "TFile.h"

class COscillo
{
public:
  COscillo(const char* h, int p=80){host=h; port=p;}
  ~COscillo(){Close();}

  //  bool GetWaveForm(const char* file);
  bool ConvertToTGraph(const char* file, int n_event,int n_channel,TGraph *gr);
  // bool GetWaveFormByTGraph(TGraph *gr);
  void Close(){};

private:
  const char* host;
  int port;
  int iNumPoints;            /* The number of points in the array   */
  double dXIncr;              /* Delta X between points */
  double dYMult;
  double dYOff;
  double dPtoff;
  double dYZero;

  //  void SkipHeader(FILE* fp);
  bool GetParamDouble(char* buf, char* pat, double *dValue);
  bool GetParamInteger(char* buf, char* pat, int *iValue);
  bool AdvanceToData(int iHandle);
  bool ScaledSimple2 (int iInFileHandle, TGraph *gr);
};


/**void COscillo::SkipHeader(FILE* fp)
{
  int line_size = 0;
```

```
  while(true){
    switch(fgetc(fp)){
    case '\r':
      break;
    case '\n':
      if(line_size == 0){
return;
      }else{
line_size = 0;
break;
      }
    default:
      line_size++;
      break;
    }
  }
}
**/
 /**
bool COscillo::GetWaveForm(const char* file)
{
  TSocket socket(host, port);

  TString str = "POST /getwmf.isf HTTP/1.1\r\n";
  str += "Host: "; str+=host; str+="\r\n";
  str += "Connection: close\r\n";
  str += "\r\n";
  str += "command=select:ch1 on\r\n";
  str += "command=save:waveform:fileformat internal\r\n";
  str += "wfmsend=Get\r\n";

  socket.SendRaw(str.Data(), str.Length());

  TString tmpFile(file);
  tmpFile.Append("~");

  FILE* fp = fopen(tmpFile.Data(), "wb");
```

```
const int size = 5120;
char buf[size];
int len = socket.RecvRaw(buf, size);

while(len>0){
  fwrite(buf, sizeof(char), len, fp);
  len = socket.RecvRaw(buf, size);
}
socket.Close();
fclose(fp);

char cHead[9];
int  iRes;
FILE* fp_in = fopen(tmpFile, "rb");

while(true){
  fscanf(fp_in, "%s %d", cHead, &iRes);
  if(iRes == 100){
    this->SkipHeader(fp_in);
  }else if(iRes >= 200 && iRes < 300){
    this->SkipHeader(fp_in);
    break;
  }else{
    printf("error, unkown responce code, %s %d\n", cHead, iRes);
    return false;
  }
}

 FILE* fp_out = fopen(file, "wb");
 len = fread(buf, sizeof(char), size, fp_in);
while(len > 0 || !feof(fp_in)){
  fwrite(buf, sizeof(char), len, fp_out);
  len = fread(buf, sizeof(char), size, fp_in);
}
fclose(fp_in);
fclose(fp_out);

remove(tmpFile.Data());
```

```
  return true;
}
 **/

bool COscillo::GetParamDouble(char* buf, char* pat, double *dValue)
{
  char* found = strstr(buf, pat);
  if(found == NULL) return false;
  found += strlen(pat);
  *dValue = atof(found);
  return true;
}

bool COscillo::GetParamInteger(char* buf, char* pat, int *iValue)
{
  char* found = strstr(buf, pat);
  if(found == NULL) return false;
  found += strlen(pat);
  *iValue = atoi(found);

  return true;
}

bool COscillo::AdvanceToData(int iHandle)
{
  char c;
  int iStat;
  int iNumDigits;

  /* Find the '#' the marks the binary field */
  c = ' ';
  while (c != '#'){
    iStat = read (iHandle, &c, 1);
    if (iStat != 1){
      printf ("*** Error - reading the input file.\n");
      return false;
    }
  }
```

```
  /* The value in variable 'c' is the ASCII for the number of digits
     to skip over.
   */
  iStat = read (iHandle, &c, 1);
  if (iStat != 1){
    printf ("*** Error - reading the input file.\n");
    return false;
  }

  iNumDigits = (int) (c - '0');
  while (iNumDigits > 0){
    iStat = read (iHandle, &c, 1);
    if (iStat != 1){
      printf ("*** Error - reading the input file.\n");
      return false;
    }
     --iNumDigits;
  }
  return true;
}

bool COscillo::ScaledSimple2 (int iInFileHandle, TGraph *gr)
{
  int lLcvPoint;       /* Used to step through the points */
  char c[2];           /* Buffer to read the point        */
  double dPoint;
  double dTime;
  int iStat;

  for (lLcvPoint = 0; lLcvPoint < iNumPoints; lLcvPoint++){
    iStat = read (iInFileHandle, c, 2);
    if (iStat != 2){
      printf ("*** Error - read data points.\n");
      return false;
    }

    short sPoint = (c[1]&0xFF) | ((c[0]&0xFF)<<8);
    dPoint = dYZero + dYMult * ((double)sPoint  - dYOff);
    dTime = dXIncr * ((double) lLcvPoint - dPtoff);
```

```
    //iStat = fprintf (fOutFile, "%lg,%g\n", dTime, dPoint);
    //if (iStat < 0){
    //  printf ("*** Error - writing output file.\n");
    //  return false;
    //}
    gr->SetPoint(lLcvPoint, dTime, dPoint);
  }
  return true;
}

bool COscillo::ConvertToTGraph(const char* file, int n_event,int n_channel,TGraph
{
  char infname[128],outfname[128];
  int iBit_NR = 16;
  int debug=0;
  int index,ch,start;

  fprintf(stderr,"start number??");
  scanf("%d",&start);

  for(index=start;index<n_event;index++){
    fprintf(stderr,"converting %d  / %d\r",index+1,n_event);
    for(ch=0;ch<n_channel;ch++){
      sprintf(infname,"%s_%d_ch%d.tds",file,index,ch+1);
      int iHandle = open (infname, O_RDONLY);
      // get header;
      char buf[512];
      read(iHandle, buf, 512);
      buf[511] = '\0';

      sprintf(outfname,"%s_%d_ch%d.root",file,index,ch+1);
      if(debug)fprintf(stderr,"converting %s to %s.",infname,outfname);

      lseek (iHandle, 0L, SEEK_SET);
    if(debug)fprintf(stderr,".");
    //bool bEnvelop = false;      /* True if this is an envelop */
    if(!this->GetParamInteger(buf, "NR_PT", &iNumPoints)){
      if(debug)fprintf(stderr,"NumPoints %d\n", iNumPoints);
    return false;
```

```
}
if(!this->GetParamDouble(buf, "XINCR", &dXIncr)) return false;
//printf("XIncr %g\n", dXIncr);
if(!this->GetParamDouble(buf, "YMULT", &dYMult)) return false;
//printf("YMult %g\n", dYMult);
if(!this->GetParamDouble(buf, "YOFF", &dYOff)) return false;
//printf("YOff %g\n", dYOff);
if(!this->GetParamDouble(buf, "PT_OFF", &dPtoff)) return false;
//printf("PT_Off %g\n", dPtoff);
if(!this->GetParamDouble(buf, "YZERO", &dYZero)) return false;
//printf("YZero %g\n", dYZero);
if(!this->GetParamInteger(buf, "BIT_NR", &iBit_NR)) return false;
//printf("Bit %d\n", iBit_NR);

// move to data start point
if(!this->AdvanceToData(iHandle)) return false;
if(debug)fprintf(stderr,"numofpoints=%d\n", iNumPoints);
//convert data
//TGraph gr(iNumPoints);
if(debug)fprintf(stderr,"iBIT=%d\n",iBit_NR);

gr->Set(iNumPoints);

switch(iBit_NR){
case 8:
  //ScaledSimple (iInFileHandle, &gr);
  break;
case 16:
  this->ScaledSimple2 (iHandle, gr);
  break;
default:
  break;
}
TFile tfile(outfname, "RECREATE");
(*gr).Write("graph");
 tfile.Close();
 if(debug)fprintf(stderr,"done");
 close(iHandle);
  }
```

```
  }
  return true;

}

#endif // COcsillo_H
```

5.1.3 波形データの Height、Area 値を出力するプログラム

```
{
#include <fstream.h>
  gROOT->Reset();
  int n,j,l,m,r;
char fn[30];
FILE *fp;
 double max[100000],th,area[100000],base;
double *v;

 printf("何番目のデータから？");
 scanf("%d",&l);
 printf("何番目のデータまで？");
 scanf("%d",&m);

 fp=fopen("alp_hist.dat","a");

for(int j=l;j<m;j++) {
  sprintf(fn,"man_%d_ch1.root",j);
  TFile f1(fn);
  TGraph wf1=(TGraph) f1.Get("graph");
  v=wf1.GetY();
  base = 0;

  for(int r=0;r<999;r++){
    base += v[r];
}
  th = (base / 1000) + 0.2 ;
  for(int i=0;i<10000;i++){
```

```
    if(v[i]>th)area[j] += (v[i] - th);
    if(v[i]>max[j])max[j]=v[i];

  }
  fprintf(fp,"%e %e %e\n",max[j]-(base/1000),area[j],base / 1000);
  if(j%100 == 0) printf("%d\n",j);
 }
 fclose(fp);


 printf("DONE!!! to %d\n",j);
}
```

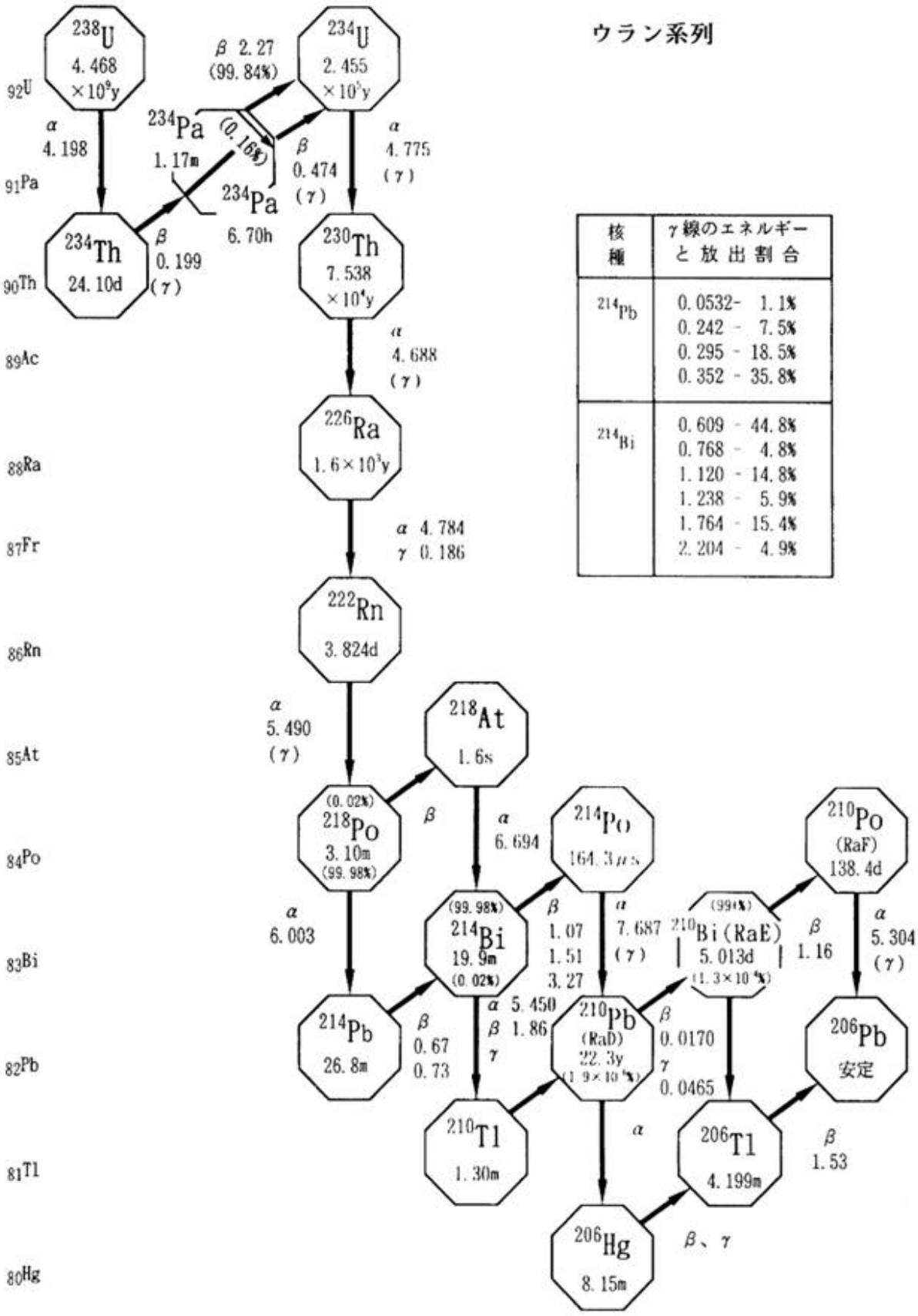
5.1.4 data から較正したヒストグラムを作るプログラム

```
{
#include<fstream.h>
#include<math.h>
#include<time.h>

gROOT->Reset();

 int h,i=0;
 double x,y,z,c,c2;


 c =1.003311;
 c2 =0.001382552;

 TCanvas *c1 =new TCanvas("c1","pulseheight-areaintegral");

 c1->Divide(2,2);

 TNtuple *n1 =new TNtuple("alp_cal","n1","x:y:z");

   ifstream deta("alp_hist.dat");

   while(deta>>x>>y>>z)n1->Fill((x*c)+0.8601651,(y*c2)+1.726739,z);
```

```
deta.close();

 c1->cd(1);
 n1->Draw("x>>PulseHeight(256,0,)");
 PulseHeight->Draw();
 c1->cd(2);

 n1->Draw("y>>AreaIntegral(512,0,)");
 AreaIntegral->Draw();

c1->cd(3);

 n1->Draw("x:y>>pro(2048)");

c1->cd(4);

 n1->Draw("z>>base(256)");

 c1.cd(1);

}
```

謝辞

実験を進めるにあたって、実験の進め方や装置の扱い方などを指導して頂き、困ったときや PPT 作成時には数多くの助言をして下さった助教の身内さんには、非常にお世話になりました。本当にありがとうございました。また、他の宇宙線研究室のスタッフ・TA・院生の方々にもお世話になり、感謝しております。最後に、P6 の仲間として一緒にやってきた他の班のみんなとは、楽しくお互い励ましあいながら実験を進められる関係で、精神的支えとなってくれました。みんな、本当にお疲れさま。



核種	γ線のエネルギーと放出割合
^{214}Pb	0.0532- 1.1% 0.242 - 7.5% 0.295 - 18.5% 0.352 - 35.8%
^{214}Bi	0.609 - 44.8% 0.768 - 4.8% 1.120 - 14.8% 1.238 - 5.9% 1.764 - 15.4% 2.204 - 4.9%

表 5.1: ウラン系列壊変表