Portable micro-TPC Assembly (POMTA)を用いたミューオンの天頂角分布の測定

2006年3月15日

瀧川　庸二朗

筒井　亮

山中　透

目次

1 μ-PIC (micro-Pixel Chamber)

1.1 μ-PIC とは何か？

μ-PIC (micro-Pixel Chamber) は，ピクセル型ガス検出器で，10 cm 四方の正方形の中に輪切りにした比例計数管が 400 μm の間隔で 256 個埋め込まれている。実験で用いた μ-PIC は，$256 \times 256 = 65,536$ ピクセルの検出器である。その構造を図 1.1 に，その装置の写真を図 1.2 に記した。

検出器内に放射線が通過すると，検出器内のガス原子との相互作用により，ガス原子を電離し，電子と陽イオンを発生する。発生した電子は Drift plane と μ-PIC の面の間の電場によって μ-PIC 面に到達する。その後 Anode-Cathode 間の電場によって，電子なだれ増幅をしながら Anode に移動する。また，そのとき発生した陽イオンは Cathode に捕獲される。一つのピクセルに対応する Anode-Cathode からの情報は縦-横の座標として読み出され，同時刻の情報を座標として記録する。

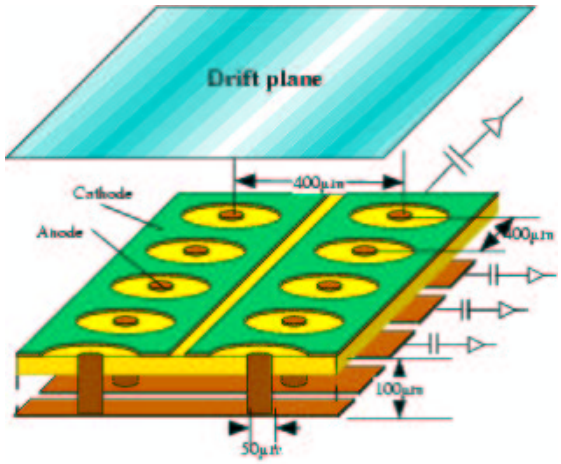

図 1.1：μ-PIC の構造*

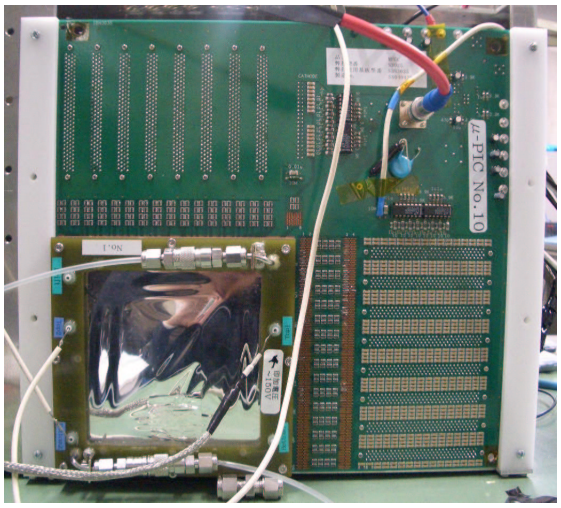

図 1.2：μ-PIC 検出器の写真

1.2 μ-PIC の増幅率

μ-PIC の増幅率（ガス増幅率）を測定するために，μ-PIC にガス（C_2H_6 10%, Ar BALANCE）を μ-PIC に 100 cc/min で封入し，安定するまで 1 時間ほど待った。

X 線源として，^{55}Fe を用い，検出器から数 cm 離して設置し，Drift plane に -150 V をかけ Anode の電圧を変化させて測定を行った。図 1.3 は，実験で用いた ASD の写真で，図 1.4 はその接続部の写真である。

実験結果を図 1.5 に記した。この結果から μ-PIC の増幅率を計算する。

X 線源の ^{55}Fe から放出された 5.9 keV の X 線と μ-PIC 内の Ar 原子との相互作用，主に光電効果によって Ar 原子は電離され，電子が放出される。その電子はさらに別の Ar 原子に束縛されている電子を電離し，最終的に X 線のエネルギーを Ar の W 値*で割った数の電子が発生する。それらの電子は Drift plane と μ-PIC 基盤との電位差によって加速され，μ-PIC 面に到達する。そして，Anode-Cathode 間の電場で Anode に捕獲されるまでの間に電子なだれを起こして増幅される。ここでの増幅率を μ-PIC の増幅率と考える。

Anode に捕獲された電子は μ-PIC 基盤から ASD (Amplifier Shaper Discriminator) に移動し，そこでさらに増幅される。実験では，1 pC の信号が入力されたとき 300 mV として出力される ASD を用いた。

---

*http://www-cr.scphys.kyoto-u.ac.jp/research/mu-PIC/NEWAGE/newage_about_e.htm から引用 (2006/03/16)。
*気体中でイオン対 1 個を作るのに必要な平均エネルギー。

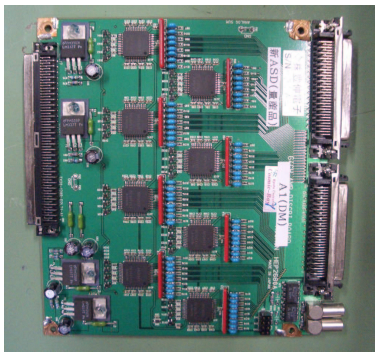

図 1.3： ASD の写真

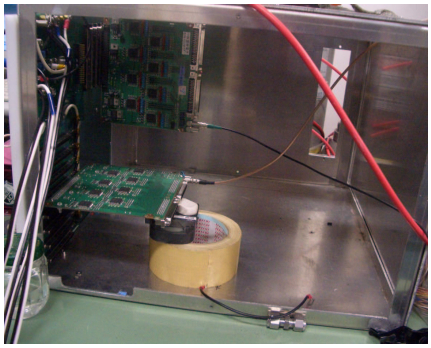

図 1.4： 検出器の ASD 接続部

μ-PIC の増幅率を $A_{\mu\mathrm{PIC}}$ とすると，次式が成り立つ。

$$\frac{5.9\ \mathrm{keV}}{26\ \mathrm{eV}} \times (1.602 \times 10^{-19}\ \mathrm{C}) \times A_{\mu\mathrm{PIC}} \times \left.\frac{300\ \mathrm{mV}}{1\ \mathrm{pC}}\right|_{\mathrm{ASD}} = \mathrm{Pulse\,Hight\ mV} \tag{1.1}$$

ここで，5.9 keV は X 線のエネルギー，26 eV は Ar 原子の W 値である。X 線のエネルギーを W 値で割った数だけ電子が発生し，それらの電子の総電荷を出すために，素電荷が掛けてある。(1.1) を用いて，図 1.5 の縦軸を増幅率に変換したグラフを図 1.6 に記した。

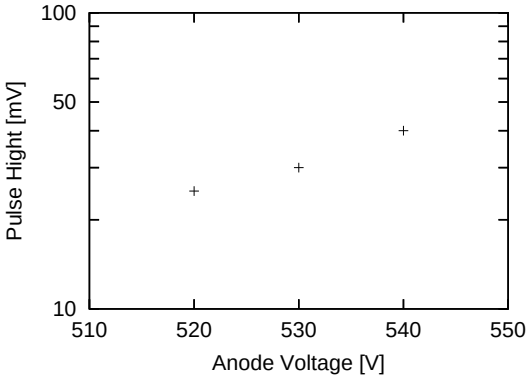

図 1.5： Anode の電圧と Pulse Hight の相関—線源 ^{55}Fe (Drift plane -150 V)

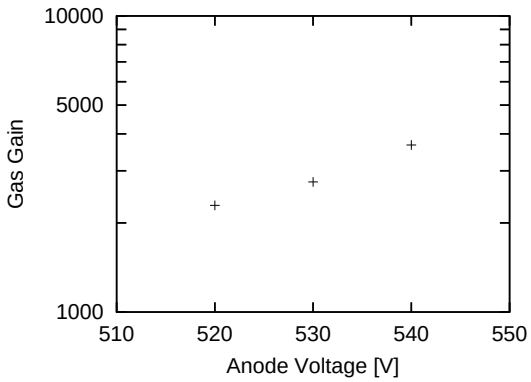

図 1.6： Anode の電圧と μ-PIC の Gas Gain の相関—線源 ^{55}Fe (Drift plane -150 V)

2 GEM (Gas Electron Multiplier)

2.1 GEM とは何か？

GEM は厚さ 50 μm の Kapton*に銅を 5 μm コーティングし，140 μm 間隔の 6 角形のパターンに 70 μm の穴を開けた素材である。この穴を電子が通過するとき，電子なだれが起きるように両端に強電場をかけられるのが特徴である。図 2.1 はその拡大図である。

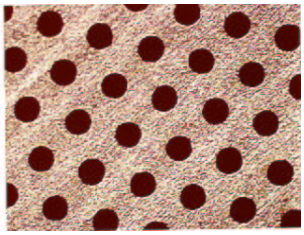

図 2.1 : GEM の拡大写真†

2.2 GEM の増幅率

図 2.2 のように，μ-PIC と Drift plane の間に GEM を取り付けて，測定を行った。

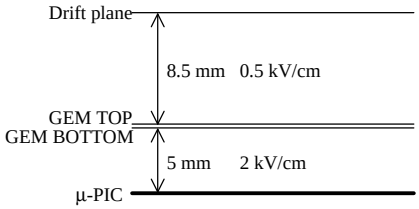

図 2.2 : μ-PIC と GEM の構成図

Drift plane を -1600 V，GEM の下側を -1000 V，Anode を $+500$ V に設定し，GEM の上側の電圧を -1250 V から -1300 V まで 10 V 刻みで変化させ，そのときの平均の Pulse Hight を測定した。その結果が図 2.3 である。線源は ^{55}Fe を用いた。

ここで，1.2 節の図 1.6 のグラフで最小二乗法により，直線の式を求め，Anode の電圧が 500 V のときの増幅率を計算すると，$A_{\mu\mathrm{PIC}}$ =1408 であった。このことを用いて，GEM の増幅率 A_{GEM} を計算する。計算式は (2.1) で表され，その計算結果を図 2.4 に記した。

$$\frac{5.9\ \mathrm{keV}}{26\ \mathrm{eV}} \times (1.602 \times 10^{-19}\ \mathrm{C}) \times A_{\mu\mathrm{PIC}} \times A_{\mathrm{GEM}} \times \frac{300\ \mathrm{mV}}{1\ \mathrm{pC}} = \mathrm{PulseHight\ mV} \tag{2.1}$$

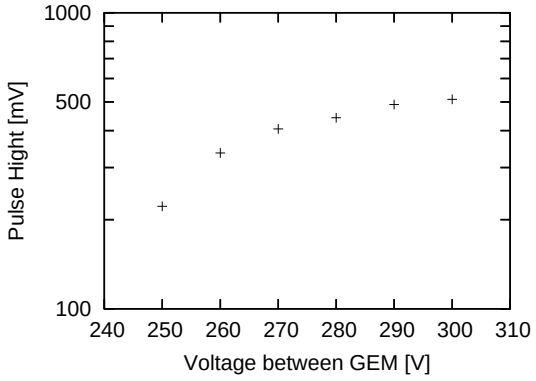

図 2.3 : GEM 間の電圧と Pulse Hight の相関—線源 ^{55}Fe (Drift plane -1600 V, GEM bottom -1000 V, Anode $+500$ V)

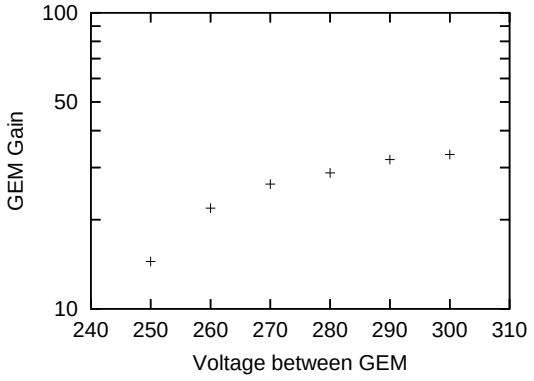

図 2.4 : GEM 間の電圧と GEM Gain の相関—線源 ^{55}Fe (Drift plane -1600 V, GEM bottom -1000 V, Anode $+500$ V)

*超耐寒・超耐熱性（$-269 \sim 400$ °C）ポリイミドフィルム。ピロメリット酸二無水物と 4,4′-ジアミノジフェニルエーテルの重合から製造。

†http://www.physics.purdue.edu/msgc/postout/gem.html (2006/03/16) から引用。

[1] の Fig.2 は，GEM Voltage に対して直線（片対数グラフ）で増加しているが，図 2.4 は直線となっていない。一見，250 V の値を除けば，直線になるように見える。しかし，[1] と比較すると，GEM Voltage が 260 - 290 V の Gain が高くなっていることが分かった。GEM Voltage が 300 V のときは GEM Gain が 30 となり，両者はほぼ一致している。GEM Voltage が 250 V のところは 2 - 3 若干高い程度である。

実験ではオシロスコープのピークの値（オシロスコープの表示）をいくつか読んでそれを平均したのだが，この Analog 作業に原因があるのかもしれない。おそらく，測定上の都合で，途中オシロスコープのトリガーレベルを変えたことが原因かもしれない。

次に，GEM 間の電圧は一定にして，GEM の下側と μ-PIC 面の間（Induction Field）の電圧（電場）を変化させる実験を行った。GEM の下側の電圧を -1000 V から -750 V まで変化させ，同時に GEM の上側の電圧も上げていき，GEM 間の電圧は一定に保った。ただし，GEM の上側と Drift plane の間の電圧は一定には保っていない。実験の詳細は省略するが，GEM の上側と Drift plane の間の電圧は増幅率にはほとんど影響しないことも確かめた。

この実験の結果を図 2.5 に，(2.1) を用いて求めた GEM の増幅率を図 2.6 に記した。

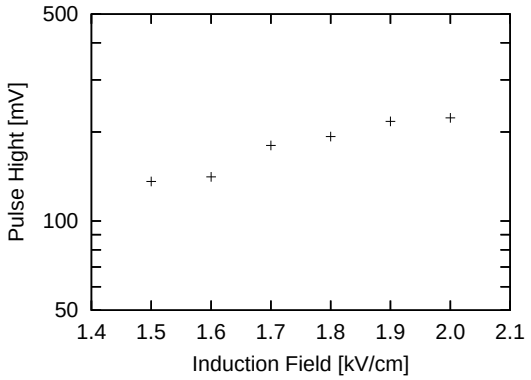

図 2.5：Induction Field の電場と Pulse Hight の相関—線源 ^{55}Fe (Drift plane -1600 V, Voltage between GEM 250 V, Anode $+500$ V)

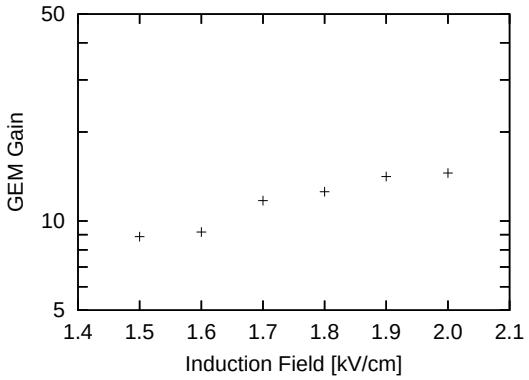

図 2.6：Induction Field の電場と Pulse Hight の相関—線源 ^{55}Fe (Drift plane -1600 V, Voltage between GEM 250 V, Anode $+500$ V)

[1] の Fig.3 では，Induction Field に対して，GEM Gain が直線的に増加（片対数グラフ）している。Induction Field の電場が小さいと GEM で増幅した電子が再び GEM に吸収されてしまうので，Induction Field を上げると吸収される電子の量が減り，GEM Gain が増加する。

3　スペクトルの取得

μ-PIC，GEM，ASD で増幅された電子の信号を電圧に変換し，そのアナログ信号の波形を計算機に取り込み，そのデータからスペクトルの解析を行った。

3.1　実験原理と方法

増幅された電子は電場によって Anode に捕獲され，電子と同時に発生した正イオンは Cathode に捕獲される。したがって，Anode 側からは負の信号が，Cathode 側からは正の信号が送られる。Anode または Cathode の信号を FADC（Flash Analog-to-Digital Converter）で読み込むのだが，FADC は STOP 信号が入ったときから一定時間遡った情報を読み込むようになっている。よって，信号が入ってきてすぐ STOP 信号を FADC に入力したのでは，波形は得られない。波形を得るためには，STOP 信号を本来の信号から適度に遅らせることで，波形のデータを読み込めるようにする必要がある。

実験装置の配線図を図 3.1 に，また，それぞれの実験装置で行われる処理を図 3.2 に記した。

まず，Anode と Cathode の信号を同期させるために，Cathode の信号はあらかじめ反転させた（LINEAR)。次に，Anode と Cathode からの信号を DISCRIMINATOR に入力した。DISCRIMINATOR は，閾値以上の信号が入ってきたときに，その電圧に応じた方形波を出力する装置である。

DISCRIMINATOR から COINCIDENCE に入力し，同期信号を出力させ，それを再び DISCRIMINATOR に入力し，さらにその信号を 2 つに分配した（本来は，再び DISCRIMINATOR に信号を通さずにそのまま GATE GENERATOR に信号を送るのがよい)。2 つの信号とも GATE GENERATOR に入力するが，そのうち 1 つの信号は，2 μsec 遅らせて，STOP 信号として FADC に入力した。もう 1 つの信号は再び DISCRIMINATOR の VETO に戻す。VETO に信号が入ると，次に解除の信号がくるまで，信号の出力が止まる。VETO をかけていないと，STOP 信号によって FADC の書き込みが停止し読み込みが始まった段階で，DISCRIMINATOR からの STOP 信号が来てしまい，読み込みが不完全となってしまう。

FADC の読み込みが終了すると，INTERRUPT REGISTER から VETO 解除の信号が出力されて，再び測定が始まる仕組みになっている。

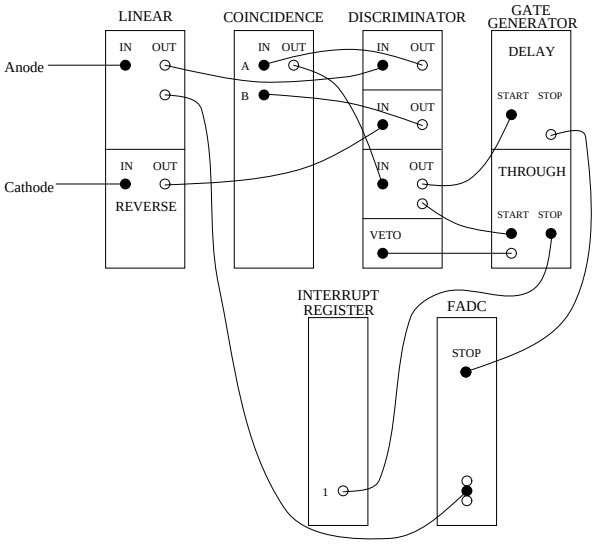

図 3.1：実験装置の配線図

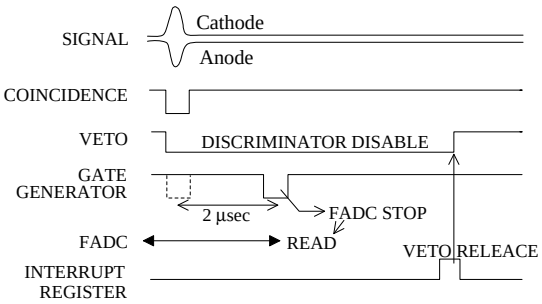

図 3.2：スペクトル測定の実験装置から出る信号とその役割

3.2 実験結果と解析

Drift plane に −1600 V，GEM top に −1250 V，GEM bottom に −1000 V，Anode に +500 V の電圧をかけて実験を行った。GEM，μ-PIC，ASD で増幅されたアナログ信号の波形は，図 3.3 のようになった。線源として，^{55}Fe，^{109}Cd を用いて，それぞれ 2000 発のデータを取得した。

増幅率は一定なので，入射 X 線のエネルギーに比例した数の電子が発生する。それらの電子の総電荷を電圧へと変換した結果が図 3.3 である。したがって，ピークの積分値は総電荷，はじめに発生した電子，さらに入射 X 線のエネルギーに比例することになる。

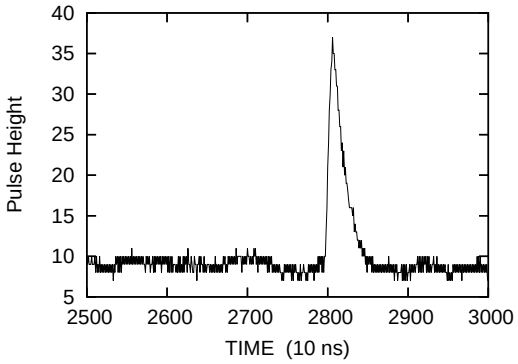

図 3.3：ASD からのアナログ信号の波形

以上より，まずノイズを除去してから，ピークの立っている部分を積分し，その値を記録しておく。それから，2000 発のそれらの積分値をヒストグラムにする。図 3.4，図 3.5 にそのヒストグラムを示した。

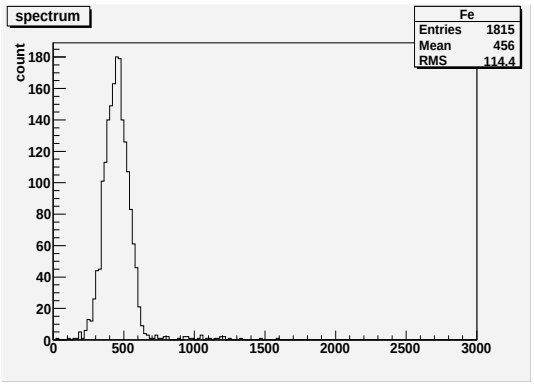

図 3.4：^{55}Fe のスペクトル

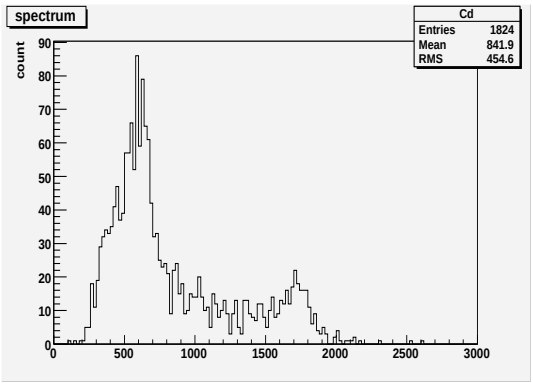

図 3.5：^{109}Cd のスペクトル

^{55}Fe の 5.9 keV のピークと，^{109}Cd のスペクトルに見られる Cu の特性 X 線 8 keV のピークを Gauss fit して，エネルギー較正を行うと，次式が得られた。

$$E = 15.1 \times 10^{-3}\mathrm{CH} - 0.85 \tag{3.1}$$

(3.1) を用いて，^{109}Cd の 22 keV に対応する CH 数を計算すると，1400 CH になった。しかし，図 3.5 と若干のずれがあるので，^{55}Fe の 5.9 keV のピークの情報は忘れて，^{109}Cd のスペクトルに見られる Cu の特性 X 線 8 keV のピークと原点だけを用いてエネルギー較正を行うと，

$$E = 13.6 \times 10^{-3}\mathrm{CH} \tag{3.2}$$

が得られた。こうすると，^{109}Cd の 22 keV に対応する CH が 1613 CH となり，さきほどより図 3.5 に一致する。

^{55}Fe の 5.9 keV のピークの情報を除いてうまくいく理由として，図 3.4 にエスケープピークが見られないこととも関係していて，Threshold が高すぎたのではないかということが挙げられる。

しかし，エネルギー分解能は前者が 13.6 % (0.8 keV) で，後者が 43.1 % (2.54 keV) であった。正確なスペクトルの測定にはもう少し実験が必要である。

4　2-D imaging

μ-PIC は，$\sim$ 100 μm の高い位置分解能をもつのが特徴である。したがって μ-PIC を用いれば 2 次元のイメージがかなり正確に得られるはずである。そこで、自転車の鍵とテストチャートを用いて２次元のイメージの取得を試みた。

4.1　実験原理

μ-PIC に X 線が進入すると，ガス原子が電離し，電子・正イオン対が発生し，それが μ-PIC や GEM で増幅される。μ-PIC の一つのピクセルに電子・正イオンが到達すると，そのピクセルの縦ー横の座標が Anode と Cathode に分けて出力される。よって片側の信号だけを読んだだけでは縦と横の一方の情報だけしか得ることができない。実際にどの座標のピクセルが反応したかを知るためには，同じ時刻に来た Anode と Cathode の情報を見つけてやる必要がある。この実験では，Encoder がその機能を果たしている。FPGA (Field Programmable Gate Array) に 2-D imaging 用のプログラムを入力すると，Encoder からは 32 bit のデータが Memory Board に送られ，そのうち 10 bit ずつが Anode，Cathode の座標に割り当てられる。つまり，Anode，Cathode の座標は $2^{10} = 1024$ CH で与えられることになる。

μ-PIC の前に線源を置き，遮蔽するものがなければ，データを plot したときに，全体に点が散らばる。線源の前に遮蔽するものを置くと，X 線はその遮蔽された部分の裏側には到達できないため，plot したとき点が描かれない。よって，遮蔽物があるところは白く，ないところは黒く表示されることになる。

4.2　実験方法

Anode，Cathode のすべてのチャンネル（8 CH $\times$ 2）に ASD を取り付け，μ-PIC の全面をアクティブにした。図 4.1 はその ASD の接続部の写真で，装置の配線は図 4.2 のようにした。

1 つの ASD から 2 つのケーブルを通して 128 CH 分のデータが出力され，8 つの ASD から合計 1024 CH 分の CH 分のデータが出力される。16 $\times$ 2 本のケーブルを Encoder に接続し，Encoder から Memory Board へは 1 本のケーブルを接続している。

Drift plane に -1600 V，GEM TOP に -1250 V，GEM BOTTOM に -1000 V，Anode に $+500$ V をかけた。線源は ^{55}Fe を用い μ-PIC の gas package から 5 - 10 cm 離して設置し，図 4.3 のように，μ-PIC の前に鍵とテストチャートを取り付け，X 線を遮蔽した。線源からの応答が 100,000 発になるまで測定した。

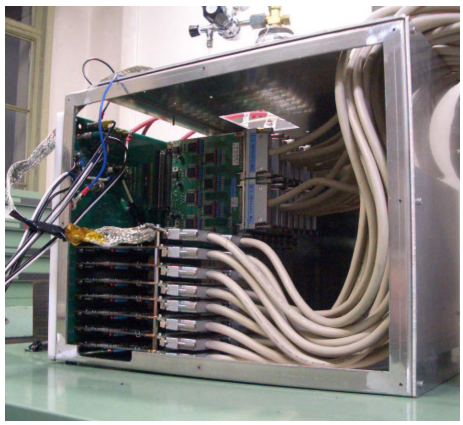

図 4.1：実験装置の ASD 接続部

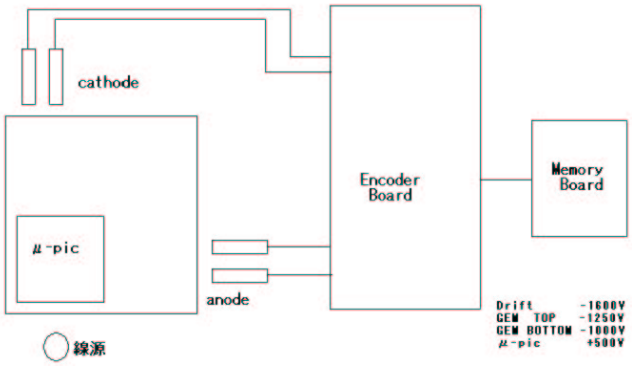

図 4.2：2-D image 検出のための配線図

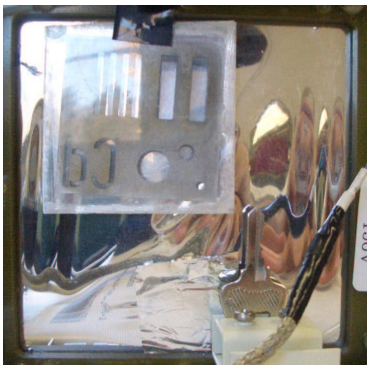

図 4.3：テストチャートと鍵

4.3 実験結果

測定結果の 2 次元ヒストグラムを図 4.4 に記した。非常に鮮明な図が得られた。感動した！

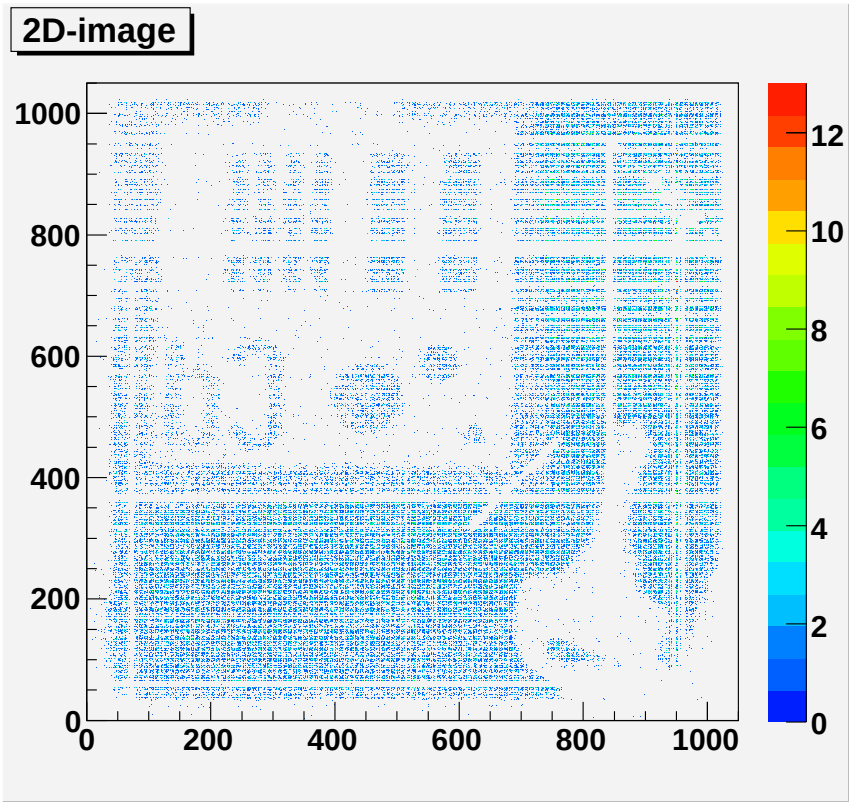

図 4.4：テストチャートと鍵の 2 次元画像

5　3-D Tracking

α 線や β 線の軌跡の 3-D 画像の取得を目指す。

5.1　実験原理

3-D Tracking の実験は 2-D imaging の応用で，2 次元座標に時間情報を加えることで，3 次元情報を得る。その原理図を図 5.1 に記した。

放射線は package 内をガス原子を電離させながら通過する。電離された電子は Drift plane から GEM，Anode 間の電場によって，μ-PIC へと運ばれる。放射線の速度が Drift 速度に比べて十分大きいときは，放射線がガス原子を電離する時間は同じだと思ってよい ($t'_0 = t'_1 = \cdots = t'_i = \cdots$)。よって，$\mu$-PIC に到達した時刻とある基準となる時刻 ($t_0$) の差 ($t_i - t_0$) に Drift 速度を掛けることで，ある基準となる高さ (h_0) からの高さ ($h_i - h_0$) が求まる。

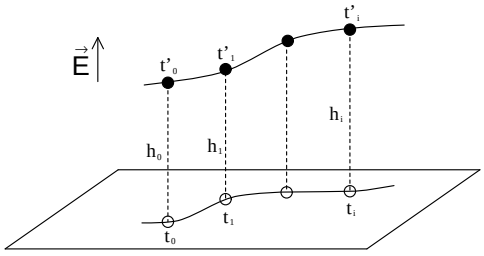

図 5.1：3-D Tracking の原理図

3-D Tracking の実験をするにあたり，Encoder の設定を 2-D imaging の設定から変更して，10 bit の Anode の座標，10 bit の Cathode の座標，12 bit の時間の情報を得られるようにした。

このように，μ-PIC を使って，時間情報から 3 次元軌跡を得られるようにした装置を micro-TPC（Time Projection Chamber）と呼んでいる。

5.2　実験方法

図 5.3 に実験装置の写真を載せた。厚さ 10 cm の package を用いた。図 5.4 は，package の内部の写真で，package の壁面に mantle* を付けておいた。mantle からは α 線が放出されている。

Drift plane に -4250 V，GEM TOP に -750 V，GEM BOTTOM に -500 V，Anode に $+450$ V の電圧をかけて測定した。β 線源として ^{90}Sr/^{90}Y を用いた。線源は package の窓から 3 cm 離して設置した。

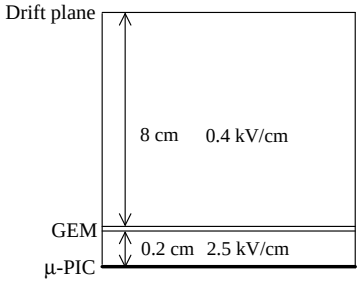

図 5.2：3-D Tracking 用の μ-PIC の構成図

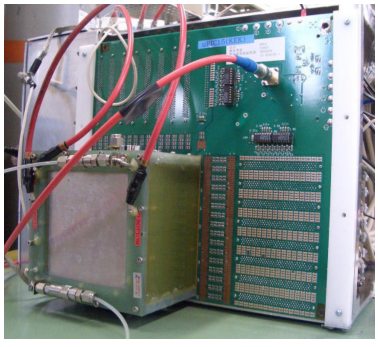
図 5.3：3-D Track 検出のための μ-PIC 基盤


図 5.4：gas package 内部の写真（mantle 取り付け部）

*GAS LANTERN に用いるマントルで，α 線を放出する。

5.3 実験結果

5.3.1 α 線の軌跡

実験の結果 α 線だと思われる軌跡を図 5.5 に記した。

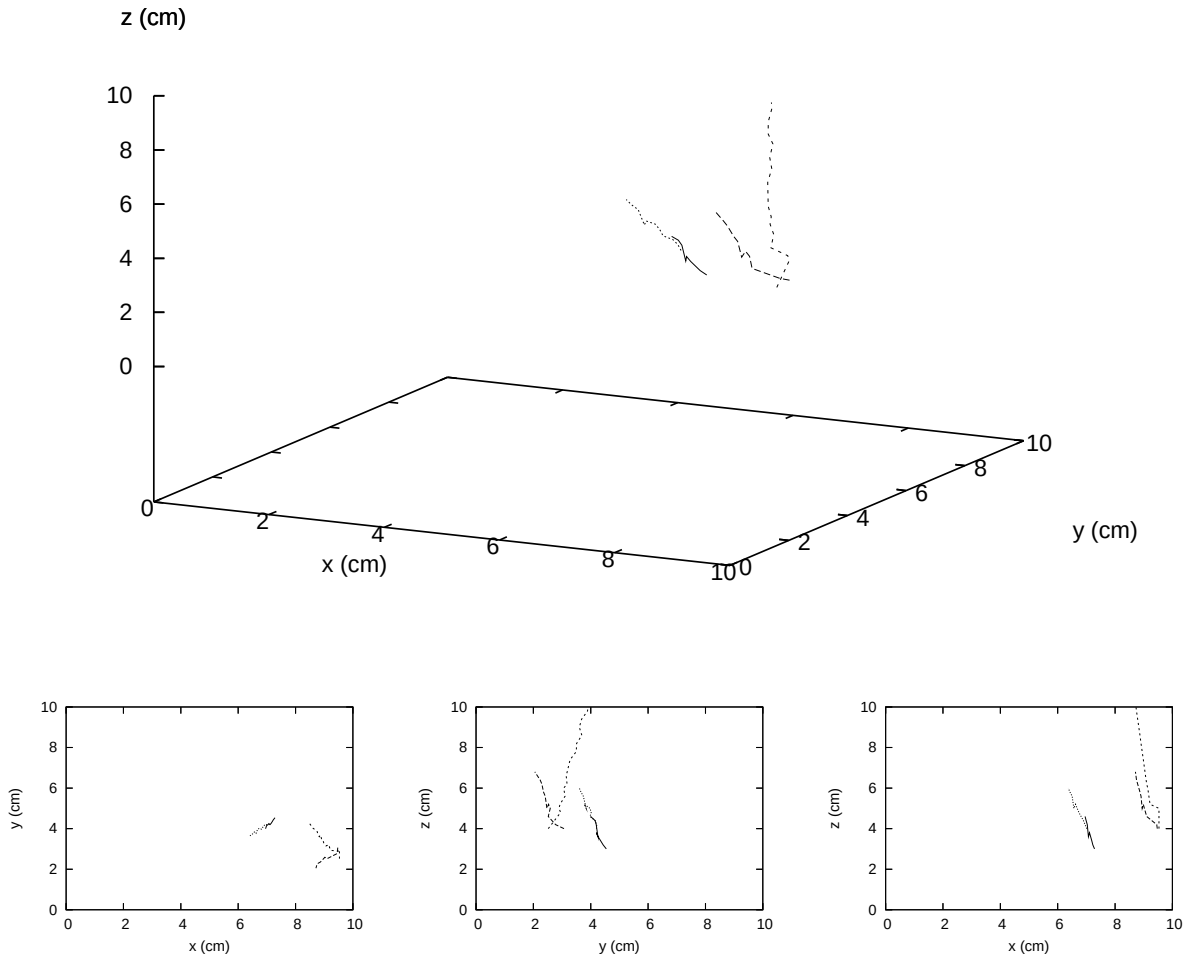

図 5.5 : α 線の 3-D Tracking—上図は 3 次元画像で，下図は xy，yz，zx 平面に射影した図である。

5.3.2 β 線の軌跡

実験の結果 β 線だと思われる軌跡を図 5.6 に記した。

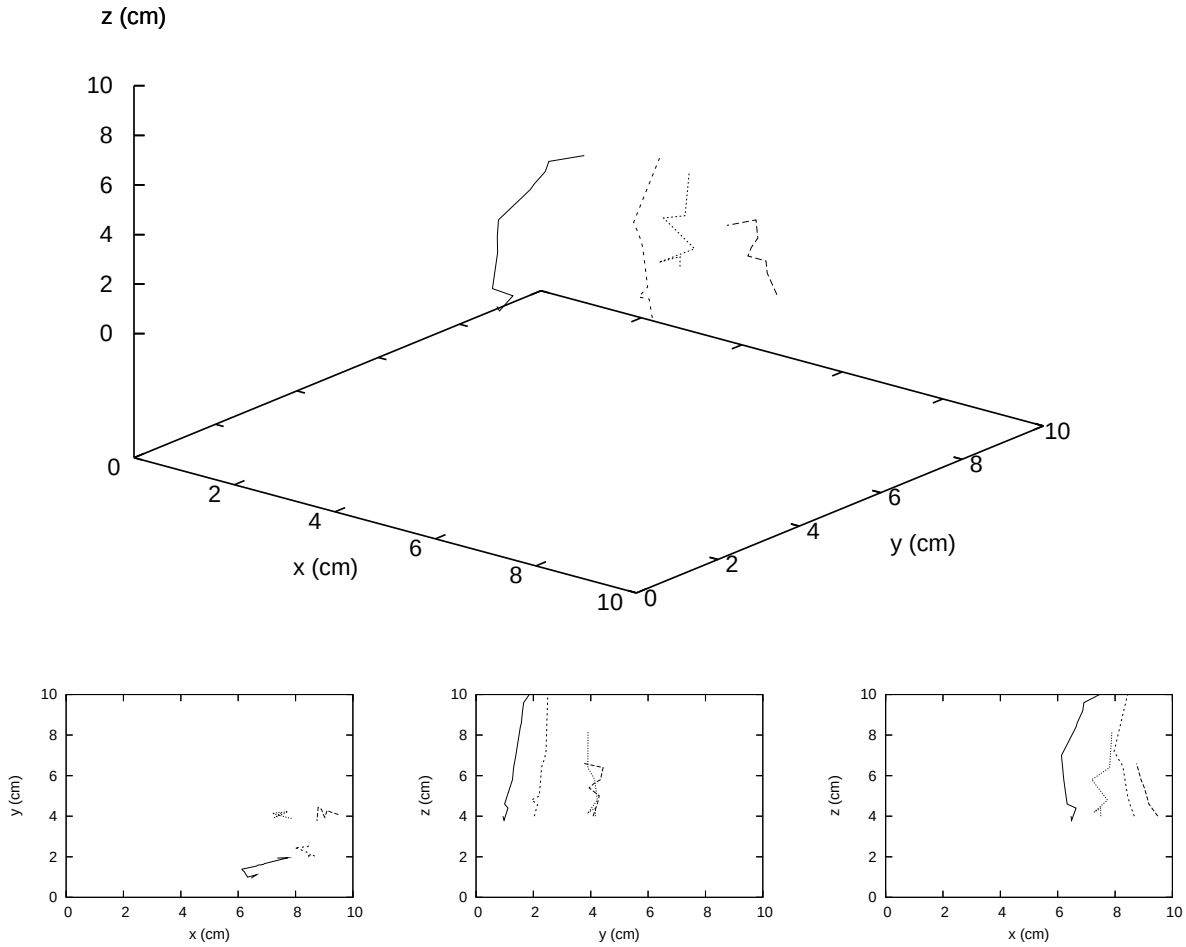

図 5.6 : β 線の 3-D Tracking—上図は 3 次元画像で，下図は xy，yz，zx 平面に射影した図である。

α 線は質量が重いので，直線的な軌跡となり，β 線は質量が軽いので，散乱されているのが分かる。

しかし，これが本当に β 線かどうかはこれだけでは分からない。今の段階では，そうらしいとしか言えない。

5.4 α 線の放射状況

α 線の放射状況を確認するために，10000 回のイベントを測定し，ノイズ除去（6.3.2 節）をしてから，その軌跡のヒストグラムを作成した。z 軸の基準を 4 cm として図 5.7 を描いた。ここで，xy 射影図が μ-PIC 面を表している。

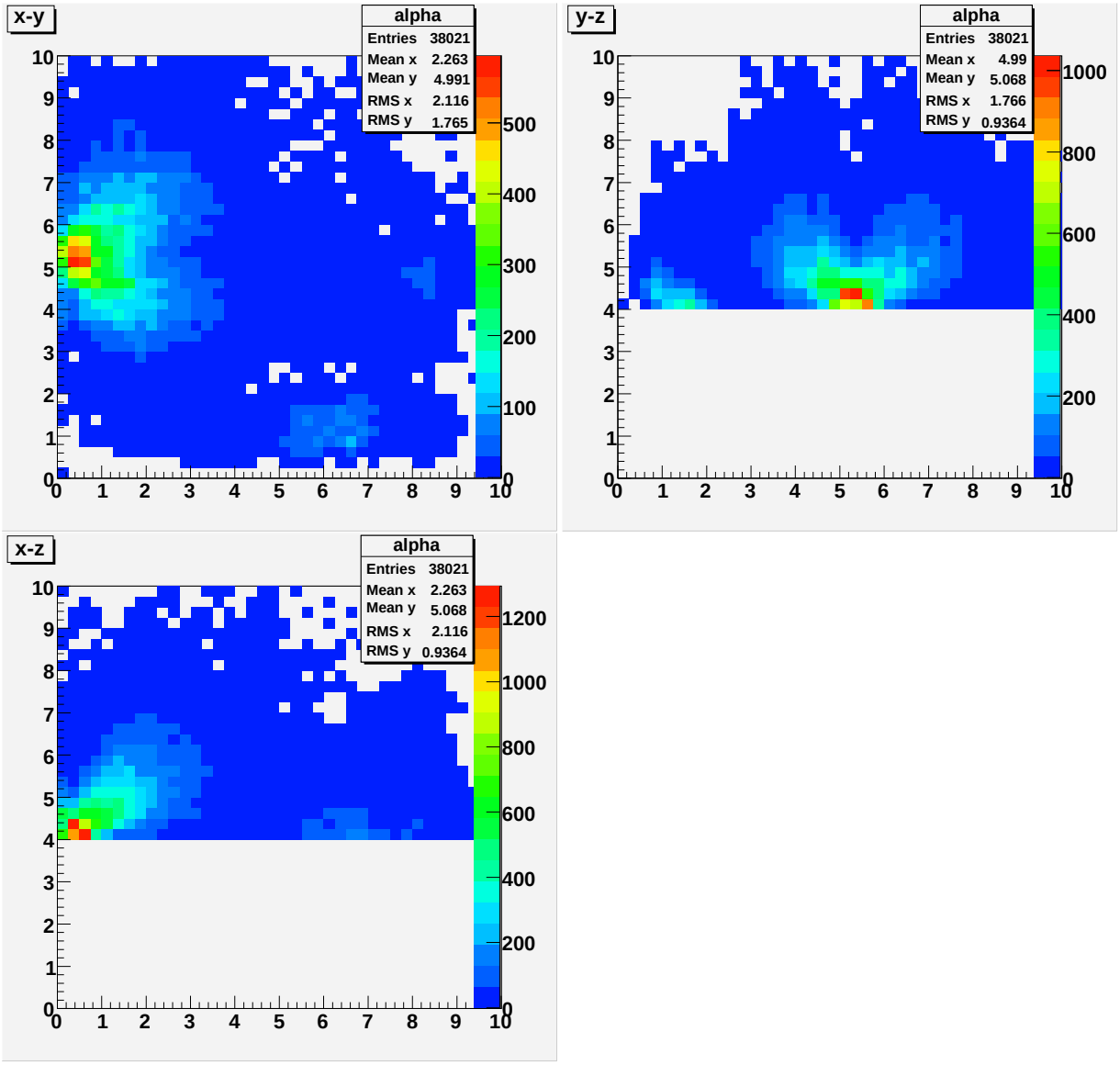

図 5.7：α 線の放射状況—左上図：xy 射影（横軸 x，縦軸 y），右上図：yz 射影（横軸 y，縦軸 z），左下図：xz 射影（横軸 x，縦軸 z）

6　muon の天頂角分布

muon*と Ar ガス原子との電磁相互作用によって，muon の軌跡に沿って電子が発生することを利用して，muon の軌跡を測定し，その軌跡から天頂角分布を解析する。

6.1　宇宙線

地球には，陽子をはじめ鉄以上の重さの原子核にいたるさまざまな種類の原子核に加え，電子や陽電子などが 1 cm^2 あたり毎秒 1 個の割合で降りそそいでいる。地球の大気圏に到達する前の宇宙線を 1 次宇宙線と呼ぶ。そのエネルギーは 10^7 eV から 10^{20} eV 以上にまでおよんでいる。

太陽系外からの宇宙線は太陽風の磁場の乱れの影響を受けて，~1 GeV の宇宙線は散乱されてしまい，地球には到達できない。さらに，10 GeV 程度までの宇宙線は，地磁気効果の影響を強く受ける。地磁気効果とは，地球を取り巻く双極子型の磁場によって，10 GeV 程度以下の宇宙線の赤道方向からの侵入を妨げられるというものだ。ただし，磁場による曲げ方が少ない分，極方向からの侵入は許される。

1 次宇宙線のエネルギースペクトルは，それぞれのエネルギー領域によってその観測の仕方が異なり，1 GeV 以下から 10 GeV にいたるエネルギー領域の宇宙線は地磁気の影響があり，地球には到達できない（極方向からは侵入できるが）ので，地球から遠く離れた位置にある人工衛星によって直接観測される。1 GeV から 10 GeV の領域は地球上での観測結果から地磁気効果を差し引いて推定することができる。10 GeV 以上から 10^{13} eV 程度までは大気の頂上付近で露出された原子核乾板の中で起こる核反応（ジェット）を測定することで求められる。10^{13} eV 以上の宇宙線は flux が少なく，直接に測定することが困難となるため，大気中での空気シャワーから間接的に求められる。

1 次宇宙線がそのまま地表に到達することはほとんどなく，空気シャワーによって，10~15 世代を重ねるうちに，さまざまな素粒子に形を変え，それらが降りそそいでいる。これを 2 次宇宙線とよぶ。度重なる核子カスケードと電子カスケードシャワーによって，1 次宇宙線のエネルギーは多数の π，K，N，Y に分割される。π^0 は寿命が短く 2 個の光子に崩壊†して電子カスケードシャワーの源となる。$\pi^{\pm}$‡や K§の崩壊によって muon が生成される。エネルギーが高ければ，muon は μ - e 崩壊¶もしないで，減衰することなくそのほぼすべてが地表に到達する。一方，空気シャワーの 90 % を占める電子-光子成分や核カスケードを構成する粒子はある深さで極大に達してから減衰する。

また，空気シャワーの横の広がりは，世代が新しい衝突を反映するエネルギーが高い粒子ほど中心に密集し，後代のエネルギーの低い粒子は中心から遠くなる [2]。

図 6.1 に空気シャワーの様子を記した。

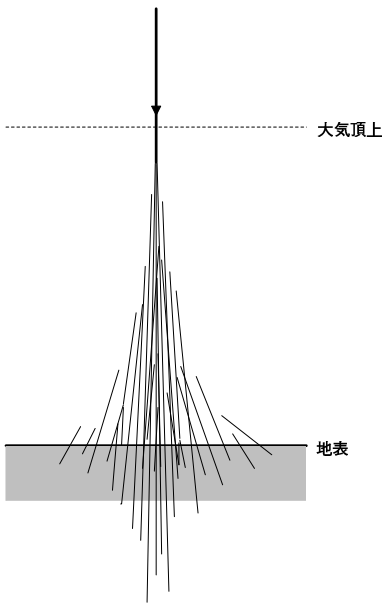

図 6.1：空気シャワー

*質量 105 MeV/c^2，荷電 $+e$ または $-e$ の lepton
†$\pi^0 \to 2\gamma$
‡$\pi^{\pm} \to \mu^{\pm}\nu_{\mu}$（10 GeV 以下）
§$K^{\pm} \to \mu^{\pm}\nu_{\mu}$（主に）
¶$\mu^{\pm} \to e^{\pm}\bar{\nu_e}\nu_{\mu}$

6.2 実験方法

muon の軌跡を測定するにあたり，muon が検出器内に侵入したときにだけ，Memory Board の情報を読み込みたい。そのために，plastic scintillator を用いた。muon が plastic scintillator を通過すると，シンチレーション光が発生する。それを光電子増倍管で増幅し，信号として利用する。plastic scintillator から出た信号は，DISCRIMINATOR に通して logic pulse に変換してから，COINCIDENCE に通す。さらにそれを NIM* TO LVDS† CONVERTER に通して LVDS に変換してから，Encoder に送る。

Encoder は，trigger 信号が来たら Memory board にデータを送信する。また，trigger 信号が来てからのデータ量などの信号が Encoder から LVDS TO NIM CONVERTER に送られる。そして，LVDS TO NIM の OUTPUT として，trigger 信号が来てから μ-PIC からの座標情報が入ってくると Drift End が，そのデータ量が多いとそれ分だけ多く立つ Enable が，その分だけ Drift End から遅れる Interrupt が出力される。実験では，時間の遅れでデータ量を反映している Interrupt を INTERRUPT REGISTER に送り，データの読み込みをはじめた。

また，Encoder の設定を変更して，32 bit のデータのうち，Anode の座標に 9 bit，Cathode の座標に 9 bit，時間情報に 9 bit，Event Id に 5 bit 割り当てた。それによって，Anode，Cathode の座標が 512 CH になった。また Event Id とは，その座標の情報がどの軌跡によるものかを示し，1 つの軌跡に対して同じ値が付される。

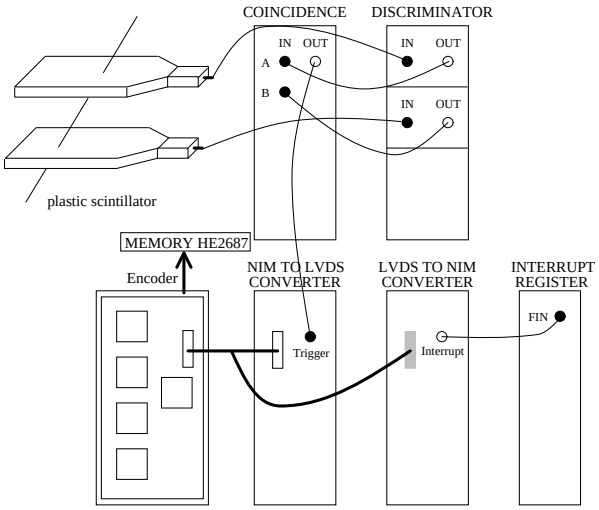

図 6.2：POMTA の配線図

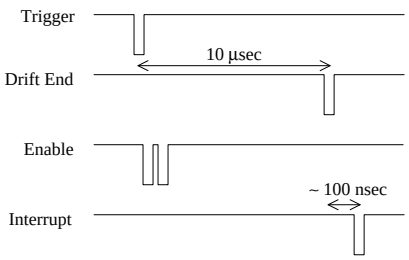

図 6.3：LVDS TO NIM CONVERTER の OUTPUT

*Nuclear Instruments Module
†Low Voltage Differential Signaling

6.3 解析準備

6.3.1 天頂角分布

図 6.4 のように，極座標（r, θ, φ）を決め，xz 射影が z 軸となす角を θ_x，yz 射影が z 軸となす角を θ_y とする。実験では，3 次元軌跡の情報が得られるわけだが，xz 射影と yz 射影とに分けて最小二乗法で直線 fit して，それぞれの直線の傾きを求める。それによって，θ_x（$\tan\theta_x$），θ_y（$\tan\theta_y$）が決まる。ここで，得られた θ_x と θ_y から極座標の角度 θ，φ を逆算して，天頂角分布の情報を収得する。その方程式は，以下の通り簡単に求められる。

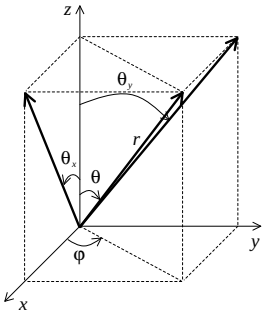

図 6.4：極座標表示

まず，デカルト座標 x，y，z と極座標 r，θ，φ の関係は，

$$x = r\sin\theta\cos\phi, \quad y = r\sin\theta\sin\phi, \quad z = r\cos\theta \tag{6.1}$$

である。また，射影角 θ_x，θ_y との関係は，

$$\tan\theta_x = \frac{x}{z}, \qquad \tan\theta_y = \frac{y}{z} \tag{6.2}$$

である。(6.1)，(6.2) より，次式が得られる。

$$\tan^2\theta = \tan^2\theta_x + \tan^2\theta_y, \quad \tan\varphi = \frac{\tan\theta_y}{\tan\theta_x} \tag{6.3}$$

ここで求めた θ，φ を用いて，天頂角分布を描くことができる。

6.3.2 ノイズ除去

実際で得られたデータを plot してみると，muon の軌跡とは明らかに違う点（ノイズ）が確認されることがあるため，このまま直線の傾きを求めたのではその分だけ正確な値からずれてしまう。そこで，ノイズを除去することを考える。plot した結果が直線にどれだけ近いかを示す散らばりの尺度として，分散または標準偏差を用いることにした。標準偏差は分散の平方根で，標準偏差 σ は次のように表される [3]。

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(x_i - \overline{x})^2} \tag{6.4}$$

ここで，n はデータ数，$\overline{x}$ は 1 回目の最小二乗法で得られる結果である。

標準偏差 σ が基準値より大きいとき，そのイベントは散らばり度が大きく，そのイベントはデータとして捨ててしまいたい。しかし，その基準値は実験の系による値なので，実験では実際の標準偏差とそのイベントを見比べて適当な値を決定した。

また，標準偏差がその基準値より小さくても，そのイベントの中で，明らかに直線から逸脱しているノイズを除去するために，残差*[4] の絶対値が σ（68.3 %）以上のデータを取り除いた。これ以上あげると，データ数が激減してしまい，いろいろ試した結果この値に落ち着いた。

*残差：$e_i = x_i - \overline{x}$

6.4 実験結果

6.4.1 μ-PIC 垂直

μ-PIC を垂直にして，package の上下からシンチレータで挟んだ。その結果を図 6.5 ，図 6.6 に記した。なお，Drift plane -4250 V，GEM TOP -750 V，GEM BOTTOM -500 V，Anode $+480$ V に設定した。

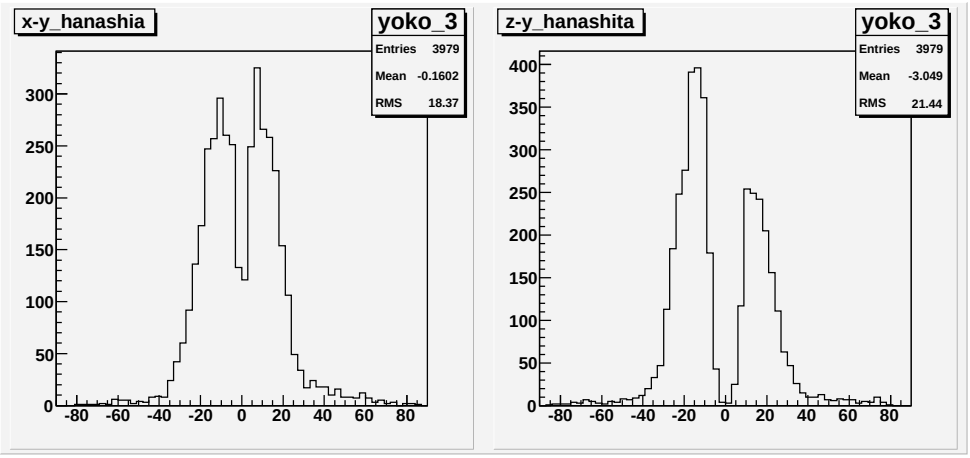

図 6.5 : muon の天頂角分布の射影図（横軸：度）。左図：xy 射影，右図：zy 射影（μ-PIC 垂直，シンチレータを上下に設置）

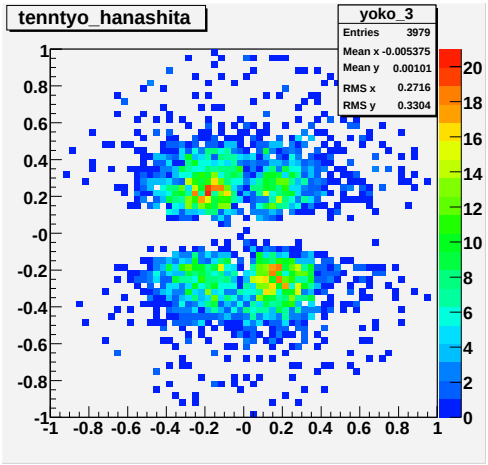

図 6.6 : muon の天頂角分布図。（μ-PIC 垂直，シンチレータを上下に設置）

μ-PIC を垂直にして，package の下面に両方のシンチレータを置いた。その結果を図 6.7 ，図 6.8 に記した。

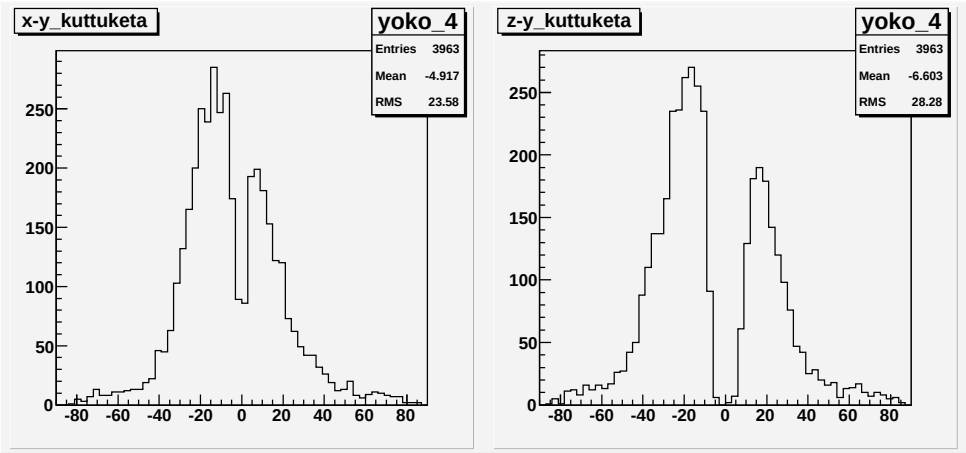

図 6.7 : muon の天頂角分布の射影図（横軸 : 度）。左図 : xy 射影，右図 : zy 射影（μ-PIC 垂直，両方のシンチレータを下面に設置）

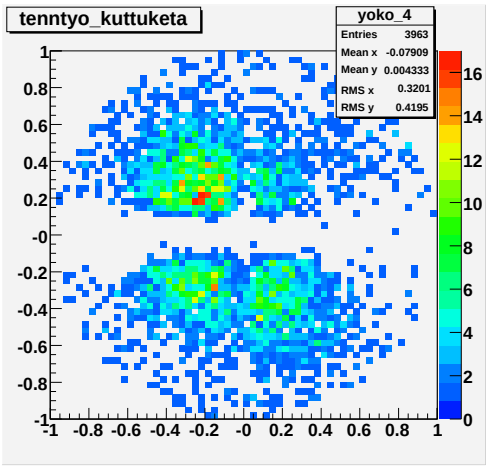

図 6.8 : muon の天頂角分布図。（μ-PIC 垂直，両方のシンチレータを下面に設置）

図 6.5 と図 6.7 を比較すると，シンチレータをくっつけたときの方が天頂角の広がりが大きくなっていることがわかる。これは，シンチレータを離して測定したときは，離した分だけ，天頂角が制限を受けるからである。この結果は図 6.6 と図 6.8 を比較しても分かる。

また，天頂角が 0° 付近のイベントが極端に少ないのはそのイベントが同時刻イベントだからである。同時刻イベントとは，muon が μ-PIC 面に対して平行に入射したときのことで，そのとき，muon の軌跡に沿って発生した電子と μ-PIC 基盤との距離がすべて等しいために，それぞれの電子に起因する電子なだれ群が同時に μ-PIC へと到達してしまう。

ここで，Encoder の設定を確認する。μ-PIC に電子なだれ群が到達した座標が必ずしも 1 つのピクセルだけではなく周辺のピクセルにも到達するために，ある範囲のピクセルの情報の最大と最小の座標をそれに対応する FPGA が 4 つの FPGA を統括する 5 つ目の FPGA に伝えて，その平均を座標としている。その範囲があまりにも大きいとそれから得られた座標が正確とはいえないので，その範囲に制限を加えている。つまり，同時刻に到達したピクセルの座標が制限の範囲を越えたところにあると，その情報は消されてしまうのである。

よって，同時刻イベントの軌跡はこの装置の特性として測定することができないことが分かった。目的は天頂角分布を求めることなので，天頂角が 0° 付近のイベントも測定してやる必要がある。

6.4.2 μ-PIC 水平

次に，μ-PIC を水平にした。こうすれば，この場合の同時刻イベントが天頂角 90° のイベントとなるので，天頂角 0° 付近のイベントが問題なく測定できる。

まず，シンチレータを package に挟んで，測定した。その結果を図 6.9 ，図 6.10 に記した。

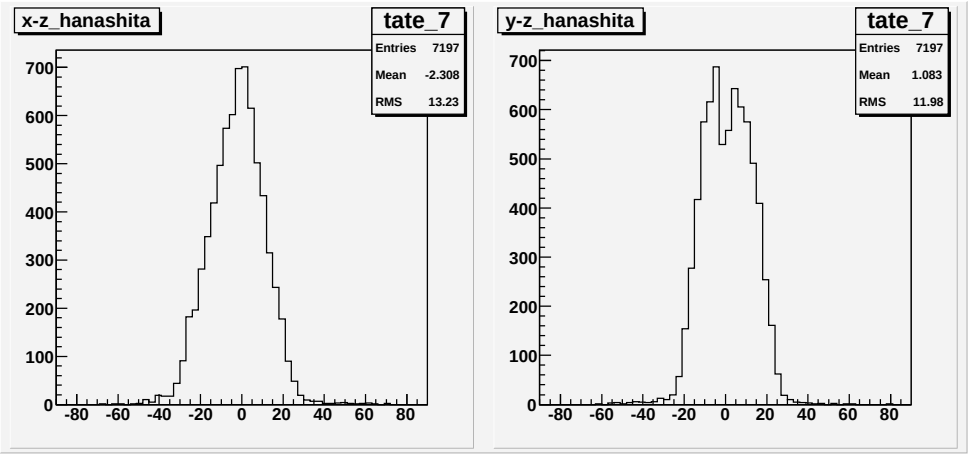

図 6.9 : muon の天頂角分布の射影図（横軸：度）。左図：xz 射影，右図：yz 射影（μ-PIC 水平，シンチレータを上下に設置）

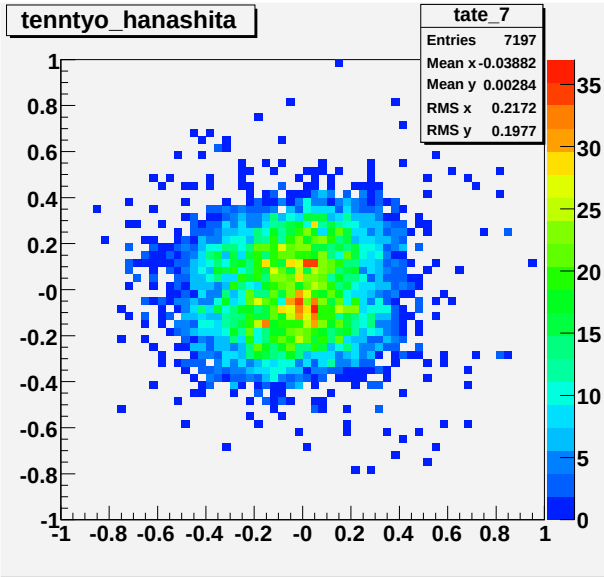

図 6.10 : muon の天頂角分布図。（μ-PIC 垂直，シンチレータを上下に設置）

図 6.10 を図 6.5 と比較すると，天頂角 0° 付近のデータが確かに得られた。しかも，天頂角 0° 付近がもっとも多いことも確かめられた。

μ-PIC を水平にして，package の下面に両方のシンチレータを置いた。その結果を図 6.11 ，図 6.12 に記した。

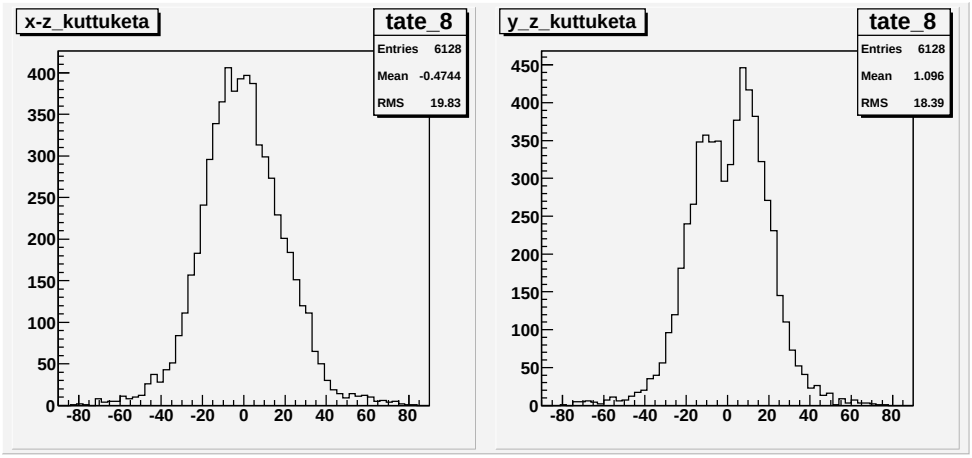

図 6.11 : muon の天頂角分布の射影図（横軸：度）。左図：xz 射影，右図：yz 射影（μ-PIC 水平，両方のシンチレータを下面に設置）

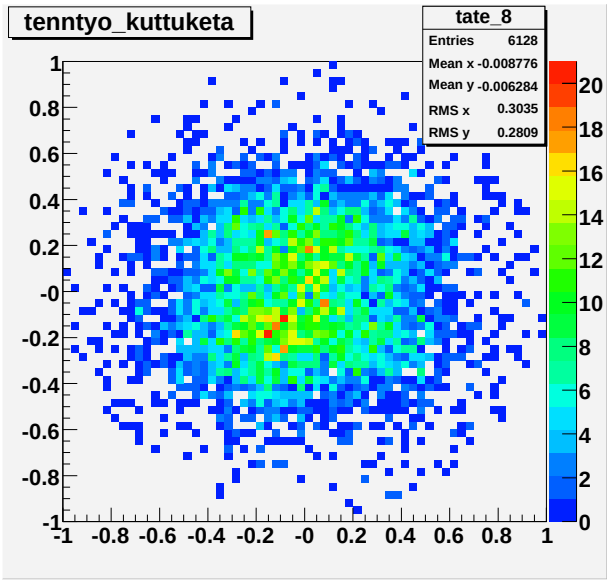

図 6.12 : muon の天頂角分布図。（μ-PIC 垂直，両方のシンチレータを下面に設置）

図 6.12 を図 6.10 と比較すると，図 6.12 の方が天頂角半径が大きいことも分かる。

これでようやく準備がととのった。

7　POMTA（Portable Micro-TPC Assembly）

micro-TPC を移動可能にした varphi 装置 POMTA を組み立て，移動性を利用して，障害物のない外，建物がある場所，地下などでの測定を目指す。

7.1　実験室での天頂角分布の測定

muon の検出器をすべてコマ付きのラックに積んで，POMTA を製作した。図 7.1 がその完成図である。この POMTA は宇宙線研究室の 2 号機である。POMTA#2 で muon を測定した。その結果を図 7.2 ，図 7.3 に示した。

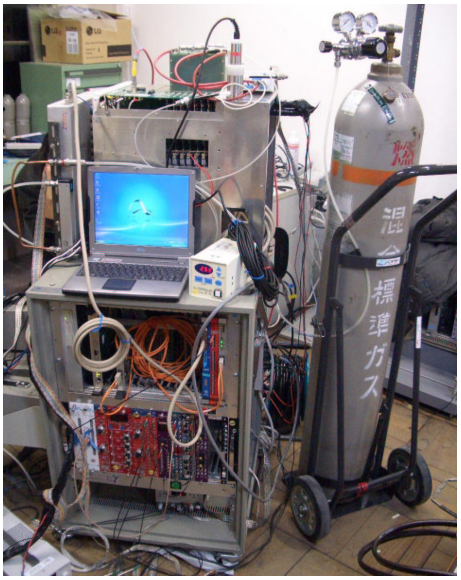

図 7.1 : POMTA#2

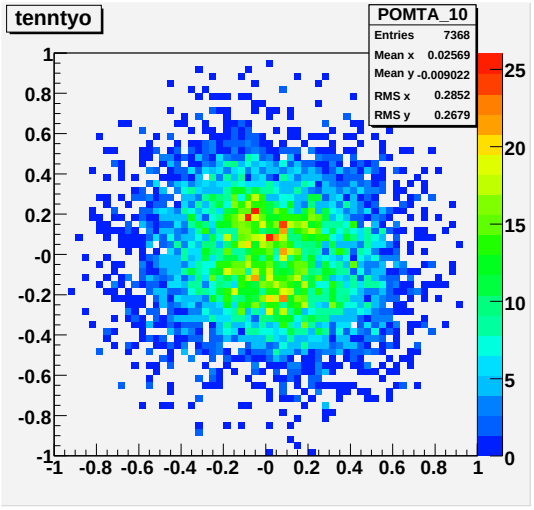

図 7.2 : muon の天頂角分布図。（μ-PIC 垂直，両方のシンチレータを下面に設置）

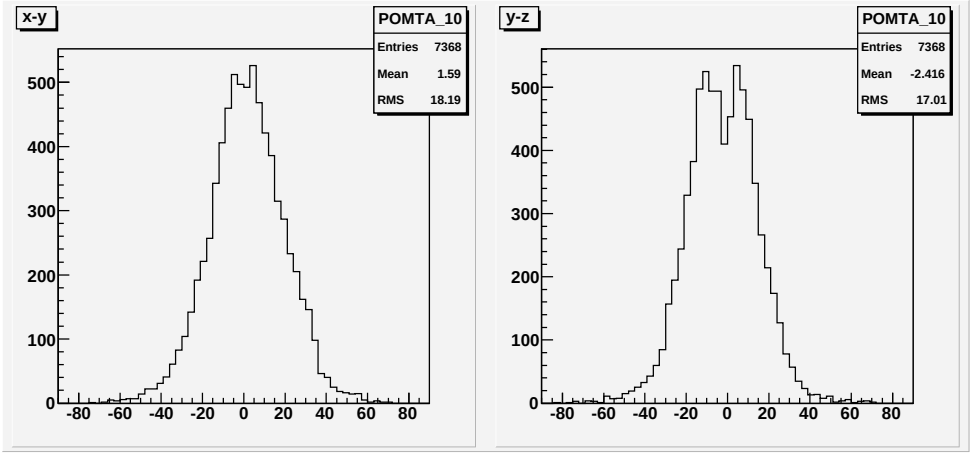

図 7.3 : muon の天頂角分布の射影図（横軸 : 度）。左図 : xz 射影，右図 : yz 射影（μ-PIC 水平，両方のシンチレータを下面に設置）

7.2　有効面積（検出可能な面積）

μ-PIC は 10 cm×10 cm の正方形の形をしているが，実際に検出可能な面積（有効面積）は muon の天頂角 θ によって異なる。有効面積の計算をするのに次の 2 つの場合を考える必要がある。

図 7.4 のように，下側のシンチレータと μ-PIC の間隔が H だとすると，検出のためには μ-PIC と両方のシンチレータを通過する必要があるので，$^{\exists}\theta$ に対して図 7.4 の muon の軌跡より左側を通過する muon は検出できない。よって，図 7.5 のような有効面積となる。

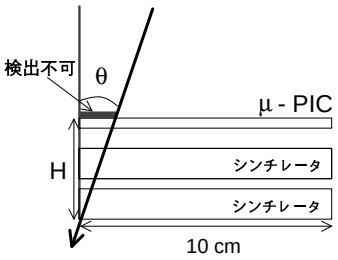

図 7.4：有効面積説明図 1

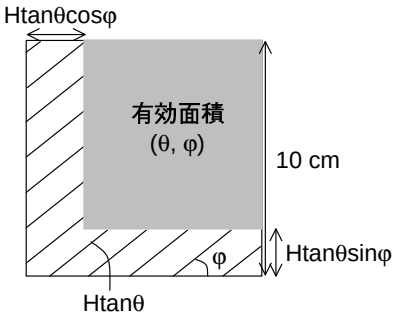

図 7.5：有効面積 1

また，正確な muon の軌跡を得るためにデータ解析の段階で 5 hit より少ないイベントのデータは捨てたことを考慮すると，μ-PIC を通過する前に 5 hit 以上の hit を残す必要があるので，図 7.6 の muon の軌跡より右側の部分は検出できない。この部分は，muon の速度にも関係があり，実験データから平均的な 5 hit 分の muon の軌跡の長さは $L = 3$ cm であった。よって，有効面積は図 7.7 のようになる。

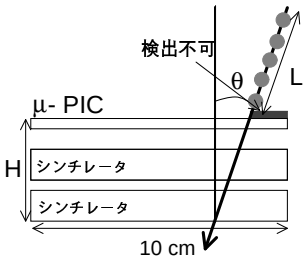

図 7.6：有効面積説明図 2

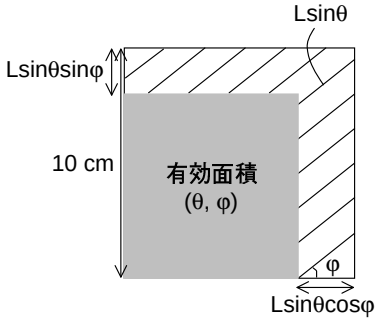

図 7.7：有効面積 2

図 7.5 と図 7.7 を統合すると，図 7.8 のような有効面積となる。

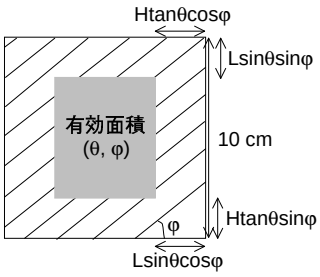

図 7.8：有効面積 total

有効面積 $S_{\rm eff}$ の式は次のようになる。

$$S_{\rm eff}(\theta,\varphi) = 10^2 - \left\{10(\sin\varphi+\cos\varphi)(H\tan\varphi + L\sin\theta) - \frac{\sin 2\varphi}{2}(H\tan\theta + L\sin\theta)^2\right\} \qquad (7.1)$$

(7.1) を φ について平均をとると，(7.2) になる。

$$\overline{S_{\rm eff}}(\theta = \frac{1}{2\pi}\int_0^{2\pi} S_{\rm eff} d\varphi = 10^2 - \frac{40}{\pi}(H\tan\theta + L\sin\theta) - \frac{2}{\pi}(H\tan\theta + L\sin\theta)^2 \tag{7.2}$$

また，図 7.9 と図 7.10 に $S_{\rm eff}$ のグラフを描いた。

実際にデータを解析するときは，天頂角の違いによる有効面積の違いを重みとしてつけて得られたデータを補正する。

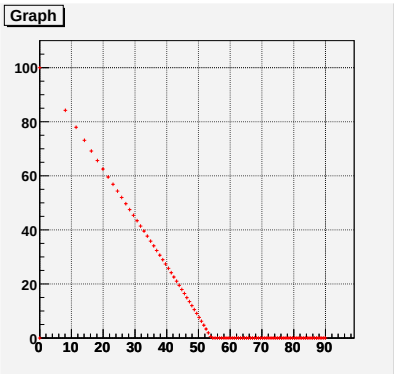

図 7.9：天頂角 θ と有効面積 $S_{\rm eff}$ の関係

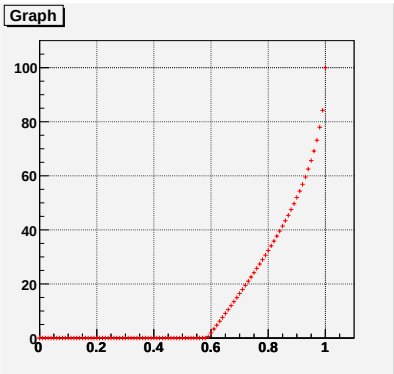

図 7.10：$\cos\theta$ と有効面積 $S_{\rm eff}$ の関係

7.3 $\cos^n$ 則

まず，(6.3) を用いて 6 節で得られたデータから天頂角 θ を計算した。muon が天頂軸に対して対象的に降り注ぐと仮定して，φ について平均をとり，muon の強度を見積もる。

$\cos\theta$ のヒストグラムを図 7.11 ，図 7.12 に示した。$\cos^n$ で fit すると，$n \sim 10$ となった。

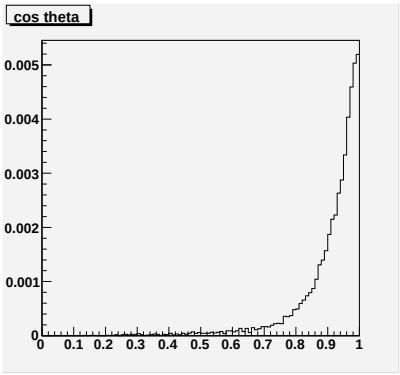

図 7.11 : $\cos\theta$ のヒストグラム（縦軸：強度 [$\mathrm{cm^{-2}s^{-1}str^{-1}}$]）

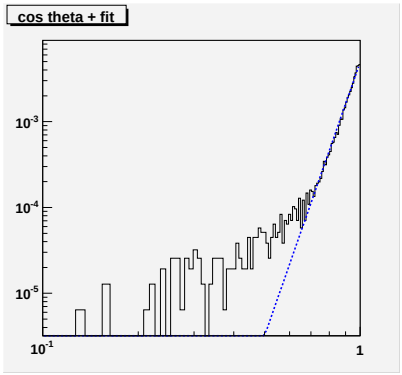

図 7.12 : $\cos\theta$ のヒストグラムの拡大図（両対数グラフ）（縦軸：強度 [$\mathrm{cm^{-2}s^{-1}str^{-1}}$]）

7.3.1 検出効率

シンチレータと μ-PIC の検出効率が異なるために，その分の補正を考える。そのためにシンチレータの全強度と muon の全強度を比較する。

シンチレータの全強度 A はデータ数，データ取得時間，シンチレータの面積 $10 \times 10 = 100\ \mathrm{cm^2}$ を用いて，$6.75 \times 10^{-3}\ \mathrm{cm^{-2}s^{-1}}$ となった。muon の全強度 B は生のデータに 7.2 節の有効面積の重みをかけ，それから全立体角で積分して求めた。規格化因子を f とすると，次の式が成り立つ。

$$A = fB \tag{7.3}$$

6 節のデータでは，$f = 60$ となった。

7.3.2 データ解析

6 節のデータに有効面積の重みをかけて，さらに規格化因子 f をかけた解析結果を図 7.13 ，図 7.14 に示した。

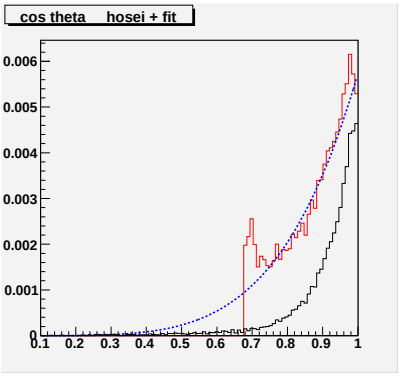

図 7.13 : $\cos\theta$ 補正のヒストグラム（縦軸:強度 [$\mathrm{cm^{-2}s^{-1}str^{-1}}$]）

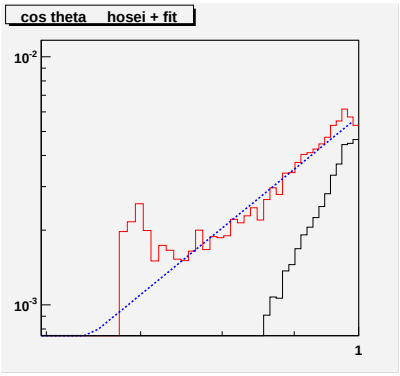

図 7.14 : $\cos\theta$ 補正のヒストグラムの拡大図（両対数グラフ）（縦軸：強度 [$\mathrm{cm^{-2}s^{-1}str^{-1}}$]）

$\cos^n$ で fit すると，$n \sim 6.6$ となった。n の値は 5 hit 分の軌跡の長さが 4 cm のときで 5.8，5 cm のときで 4.6 になった。その軌跡の長さによって，n の値は大きく左右されることが分かる。[5] によると，2 次宇宙線の天頂角分布はエネルギーが高い方は $\sim \cos^2\theta$，低い方は $\sim \cos^3\theta$ になるので，実験で求めた値はまだ大きい。しかし，この差を埋める解析まではいかなかった。

ただ，muon の強度の鉛直成分は，実験では $0.6\times10^{-2}\mathrm{cm}^{-2}\mathrm{s}^{-1}\mathrm{str}^{-1}$，[5] では $\sim 0.8\times10^{-2}\ \mathrm{cm}^{-2}\mathrm{s}^{-1}\mathrm{str}^{-1}$ となりほぼ一致していた。

ここまではよかったのだが，いよいよ外にもって測定をはじめようかとしているときに，GEM がつながり，さらに FPGA のプログラムの入力がうまくいかなくなり，当初の目標を達成することはできなくなった。

外で測定できていたら，障害物による muon の遮蔽の度合や地上と地下での muon の強度などの比較ができたかもしれないので，非常に残念だった。

付録

A μ-PIC 検出器動作プログラム in kazu

A.1 1 次元の main.cxx

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/param.h>
#include <vmedrv.h>
#include "rpv160.h"
#include "rpv130.h"
#define BASE 0x400000
#define BASEi 0x2000

int main(int argc,char *argv[]){
  int        dev,devi;
  size_t     map,mapi;
  char       *addr,*addri;
  int FADC_depth=3000;

  if(argc >= 2){
    FADC_depth = atoi(argv[1]);
  }

  addr = (char *)init_fadc( &dev, BASE, &map );
  addri = (char *)init_rpv130( &devi, BASEi, &mapi );
  set_fadc((unsigned int)addr);
  clear_fadc((unsigned int)addr);
  start_fadc((unsigned int)addr);
  pulse_rpv130((unsigned int) addri, 1);

  int num = data_num_fadc((unsigned int) addr);
  int ch[8][num];

  for( int i = 0; i < num; i ++ ){
    for( int j = 0; j < 8; j ++ ){
      ch[j][i]=0;
    }
  }
  for( int i = 0; i < 8; i ++ ){
    read_fadc((unsigned int) addr, num, i, &ch[i][0] );
  }

  if(num>FADC_depth){
    for( int i = num-FADC_depth; i < num; i ++ ){
      for( int j = 0; j < 8; j ++ )printf( "%d\t", ch[j][i] );
      printf("\n");
    }
  }
  else{
    for( int i = 0; i < FADC_depth-num; i ++ ){
      for( int j = 0; j < 8; j ++ )printf( "0\t");
      printf("\n");
    }
    for( int i = FADC_depth-num; i < FADC_depth; i ++ ){
```

```
      for( int j = 0; j < 8; j ++ ){
printf( "%d\t", ch[j][i+num-FADC_depth] );
      }
      printf("\n");
    }
  }

  end_fadc( dev, map, (unsigned int) addr);
  end_rpv130( devi, mapi, (unsigned int) addri);
  return 0;
}
```

A.2 2 次元の memory.cxx

```
#include <fstream.h>
#include <stdio.h>
#include <iomanip.h>
#include "memory.h"

int read_memory_2D( unsigned int addr, int num ){
  int         *ptr;
  int         flag;
  unsigned    tmp;
  int         a_center;
  int         c_center;
  int clock;

  do{
    ptr = (int *)( addr + 0x4 );
    flag = *ptr;
  }while( flag > 0 );

  if( flag == 0x0 ){
    for( int i = 0; i < num; i++ ){
      ptr = (int *)( addr + 0x0 );
      tmp = (unsigned)*ptr;
      c_center = ( tmp ) & 0x7ff;
      a_center = ( tmp >> 11 ) & 0x7ff;
      clock =  ( tmp >> 22 ) & 0x3ff;

      cout << a_center << "\t"
           << c_center << "\t"
   << clock << endl;
    }
  }
    return flag;
};

int read_memory_miuchi( unsigned int addr, int num ){
  int         *ptr;
  int         flag;
  unsigned    tmp;
  int         a_center;
  int         c_center;
  int clock;

  do{
    ptr = (int *)( addr + 0x4 );
    flag = *ptr;
  }while( flag > 0 );

  if( flag == 0x0 ){
    for( int i = 0; i < num; i ++ ){
      ptr = (int *)( addr + 0x0 );
      tmp = 0xffffffff - (unsigned)*ptr;
      a_center = ( tmp ) & 0x3ff;
      c_center = ( tmp >> 10 ) & 0x3ff;
      clock =  ( tmp >> 20 ) & 0xfff;
```

```
      cout << a_center << "\t"
           << c_center << "\t"
   << clock
   << endl;
    }
  }
  return flag;
};

int start_memory(unsigned int addr){
  int         *ptr;
  int         flag;

  do{
    ptr = (int *)( addr + 0x4 );
    flag = *ptr;
  }while( flag );

  usleep( 50 );
  ptr = (int *)( addr + 0x0 );
  *ptr = 0x1;
  usleep( 100 );

  return flag;
};

void set_memory(unsigned int addr, int num){
  int         *ptr;

  ptr = (int *)(addr + 0x8);
  *ptr = num;
};

void abort_memory(unsigned int addr){
  int         *ptr;

  ptr = (int *)(addr + 0x0);
  *ptr = 0x0;
};

void reset_memory(unsigned int addr){
  int         *ptr;

  ptr = (int *)(addr + 0x0);
  *ptr = 0x2;
};

unsigned int init_memory(int *dev, long base, size_t *mapsize){
  int         offset, pagesize;
  char        *addr;

  cerr << "\t memory base address:     " << base << endl;

  if( (*dev = open("/dev/vmedrv32d32", O_RDWR)) == -1 ){
    perror( " open : /dev/vmedrv32d32 " );
    exit(1);
  }

  pagesize = getpagesize();
  *mapsize = 0x1000;
  offset = base % pagesize;
  base = base - offset;
  addr = (char *)mmap( 0, *mapsize, PROT_READ|PROT_WRITE,
                       MAP_SHARED, *dev, base );
  if( addr == MAP_FAILED ){
    perror( " mmap : memory board failed. " );
    exit(1);
  }
  addr += offset;
```

```
  cerr << "\t pagesize:                " << pagesize << endl;
  cerr << "\t mapsize:                 " << *mapsize << endl;
  cerr << "\t memory address on linux: " << (unsigned int)addr << endl;

  return (unsigned int)addr;
};
```

A.3 2 次元の main.cxx

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <fstream.h>
#include <iomanip.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/param.h>
#include <vmedrv.h>
#include "memory.h"
#include "main.h"

using namespace std;

int take_data(){
  int flag = 0;

  // HE2687
  read_memory_2D( (unsigned int)addr_he2687, num_event );
  return flag;
};

int main( int argc, char **argv ){
  long   start_time, end_time, elapsed_time;
  double rate;

  // set default values
  num_event     = 10000;

  if(argc>1)num_event=atoi(argv[1]);
  cerr << "num of event" << num_event<<endl;
  // HE2687
  addr_he2687 = (char *)init_memory( &dev_he2687, BASE_HE2687, &map_he2687 );
  reset_memory( (unsigned)addr_he2687 );
  set_memory( (unsigned int)addr_he2687, num_event );
  abort_memory( (unsigned int)addr_he2687 );

  // DAQ start
  cerr << "DAQ start...";

  start_memory( (unsigned int)addr_he2687 );
  take_data();

  cerr << "...done." << endl;
  return 0;
}
```

A.4 3 次元の main.cxx

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <fstream.h>
#include <iomanip.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/param.h>
#include <vmedrv.h>
#include <time.h>
#include "memory.h"
#include "main.h"

using namespace std;

int take_data(){
  int flag = 0;

  // HE2687
  read_memory_miuchi( (unsigned int)addr_he2687, num_event );
  return flag;
};

long Now( void ){
  time_t timer, now;

  now = time( &timer );
  return (long)now;
}

int main( int argc, char **argv ){
  long   start_time, end_time, elapsed_time;
  double rate;

  // set default values
  num_event     = 10000;

  if(argc>1)
    num_event = atoi(argv[1]);

  cerr << "num of event" << num_event<<endl;
  // HE2687
  addr_he2687 = (char *)init_memory( &dev_he2687, BASE_HE2687, &map_he2687 );
  reset_memory( (unsigned)addr_he2687 );
  set_memory( (unsigned int)addr_he2687, num_event );
  abort_memory( (unsigned int)addr_he2687 );

  // DAQ start
  start_time = Now();
  cerr << "DAQ start...";

  start_memory( (unsigned int)addr_he2687 );
  take_data();
  end_time = Now();
  elapsed_time = difftime( end_time , start_time);
  rate =(double) num_event / elapsed_time;

  cerr << "...done." << endl;
  cerr << "time: " << elapsed_time << " sec" << endl;
  cerr << "rate: " << rate         << " Hz"  << endl;

  return 0;
}
```

A.5 muon の memory.cxx

```
#include <fstream.h>
#include "memory.h"
#include "MIudaq_val.h"

int read_memory (unsigned int addr , unsigned int *data , int num ){
  int         *ptr;
  int         flag;
  unsigned    tmp;
  int         a_center;
  int         c_center;
  int clock;
  int id;
  int escape = 0;

  do{
    ptr = (int *)( addr + 0x4 );
    flag = *ptr;
  }while( flag > 0 );

  if( flag == 0x0 ){
    for( int i = 0; i < num; i ++ ){
      ptr = (int *)( addr + 0x0 );
      tmp = 0xffffffff - (unsigned)*ptr;
      c_center = ( tmp >> 9 ) & 0x1ff;
      a_center = ( tmp >> 18 ) & 0x1ff;
      clock = ( tmp ) & 0x1ff;
      id = ( tmp >> 27 ) & 0x1f;

      if((a_center == 0) && (c_center == 0) && (clock == 0) && (id == 0))
escape = 1;
      if(escape == 0){
cout << i+1 <<"\t";
cout << c_center << "\t"
     << a_center << "\t"
     << clock << "\t"
     << id << endl;
      }
    }
  }
  return flag;
};

int start_memory(unsigned int addr){
  int         *ptr;
  int         flag;

  do{
    ptr = (int *)(addr + 0x4);
    flag = *ptr;
  }while(flag);

  usleep(50);
  ptr = (int *)(addr + 0x0);
  *ptr = 0x1;
  usleep(100);

  return flag;
};

void set_memory(unsigned int addr, int num){
  int         *ptr;

  ptr = (int *)(addr + 0x8);
  *ptr = num;
};

void abort_memory(unsigned int addr){
  int         *ptr;
```

```
  ptr = (int *)(addr + 0x0);
  *ptr = 0x0;
};

void reset_memory(unsigned int addr){
  int         *ptr;

  ptr = (int *)(addr + 0x0);
  *ptr = 0x2;
};

unsigned int init_memory(int *dev, long base, size_t *mapsize){
  int         offset, pagesize;
  char        *addr;

  printf("   memory base address   : 0x%lx\n", base);

  if((*dev = open("/dev/vmedrv32d32", O_RDWR)) == -1){
    perror(" open : /dev/vmedrv32d32 ");
    exit(1);
  }

  pagesize = getpagesize();
  *mapsize = 0x1000;
  offset = base % pagesize;
  base = base - offset;
  addr = (char *)mmap(0, *mapsize, PROT_READ|PROT_WRITE,
                      MAP_SHARED, *dev, base);
  if(addr == MAP_FAILED){
    perror(" mmap : memory board failed. ");
    exit(1);
  }
  addr += offset;

  printf("        pagesize         : 0x%x\n", pagesize);
  printf("        mapsize          : 0x%x\n", *mapsize);
  printf(" memory address on linux : 0x%x\n", (unsigned int) addr);

  return (unsigned int)addr;
};

int end_memory(int dev, size_t mapsize, unsigned int addr){
  munmap((char *) addr, mapsize);
  close(dev);
  printf(" memory : unmapped.\n");
  return 0;
};
```

A.6 muon の MIudaq.cxx

```
#include <fstream.h>
#include "memory.h"
#include "MIudaq_val.h"

int read_memory (unsigned int addr , unsigned int *data , int num ){
  int         *ptr;
  int         flag;
  unsigned    tmp;
  int         a_center;
  int         c_center;
  int clock;
  int id;
  int escape = 0;

  do{
    ptr = (int *)( addr + 0x4 );
    flag = *ptr;
  }while( flag > 0 );
```

```
  if( flag == 0x0 ){
    for( int i = 0; i < num; i ++ ){
      ptr = (int *)( addr + 0x0 );
      tmp = 0xffffffff - (unsigned)*ptr;
      c_center = ( tmp >> 9 ) & 0x1ff;
      a_center = ( tmp >> 18 ) & 0x1ff;
      clock = ( tmp ) & 0x1ff;
      id = ( tmp >> 27 ) & 0x1f;

      if((a_center == 0) && (c_center == 0) && (clock == 0) && (id == 0))
escape = 1;
      if(escape == 0){
cout << i+1 <<"\t";
cout << c_center << "\t"
     << a_center << "\t"
     << clock << "\t"
     << id << endl;
      }
    }
  }
  return flag;
};

int start_memory(unsigned int addr){
  int         *ptr;
  int         flag;

  do{
    ptr = (int *)(addr + 0x4);
    flag = *ptr;
  }while(flag);

  usleep(50);
  ptr = (int *)(addr + 0x0);
  *ptr = 0x1;
  usleep(100);

  return flag;
};

void set_memory(unsigned int addr, int num){
  int         *ptr;

  ptr = (int *)(addr + 0x8);
  *ptr = num;
};

void abort_memory(unsigned int addr){
  int         *ptr;

  ptr = (int *)(addr + 0x0);
  *ptr = 0x0;
};

void reset_memory(unsigned int addr){
  int         *ptr;

  ptr = (int *)(addr + 0x0);
  *ptr = 0x2;
};

unsigned int init_memory(int *dev, long base, size_t *mapsize){
  int         offset, pagesize;
  char        *addr;

  printf("   memory base address   : 0x%lx\n", base);

  if((*dev = open("/dev/vmedrv32d32", O_RDWR)) == -1){
    perror(" open : /dev/vmedrv32d32 ");
    exit(1);
```

```
  }

  pagesize = getpagesize();
  *mapsize = 0x1000;
  offset = base % pagesize;
  base = base - offset;
  addr = (char *)mmap(0, *mapsize, PROT_READ|PROT_WRITE,
                      MAP_SHARED, *dev, base);
  if(addr == MAP_FAILED){
    perror(" mmap : memory board failed. ");
    exit(1);
  }
  addr += offset;

  printf("        pagesize         : 0x%x\n", pagesize);
  printf("        mapsize          : 0x%x\n", *mapsize);
  printf(" memory address on linux : 0x%x\n", (unsigned int) addr);

  return (unsigned int)addr;
};

int end_memory(int dev, size_t mapsize, unsigned int addr){
  munmap((char *) addr, mapsize);
  close(dev);
  printf(" memory : unmapped.\n");
  return 0;
};
```

B 解析用シェルスクリプト

B.1 1 次元解析用のシェルスクリプト

```
#!/bin/bash/

i=1
loop=2000

while [ $i -le $loop ]; do

    ./integral filename >> out
    i=`expr $i + 1`

done
```

B.2 muon 測定の C 言語をまとめてコンパイルするシェルスクリプト

```
#!/bin/sh

gcc -o time_to_cm time_to_cm.c

gcc -o muonsute_variance muonsute_variance.c -O -L. -llapack -lblas -lm -lg2c

gcc -o than3 than3.c

gcc -o muon_degree muon_degree.c -O -L. -llapack -lblas -lm -lg2c

gcc -o tenntyo tenntyo.c -O -L. -llapack -lblas -lm -lg2c
```

B.3　muon 測定のデータをすばやく出すためのシェルスクリプト

```
#!/bin/sh

sed -n '/[a-z]/!p' filename > out_1

awk '{a[NR]=$0}END{for(i=NR;i>0;i--)print a[i]}' out_1 > out_2

./time_to_cm out_2 10000 > out_3

./muonsute_variance out_3 > out_4

awk '{a[NR]=$0}END{for(i=NR;i>0;i--)print a[i]}' out_4 > out_5

./than3 out_5 10000 > out_6

./muon_degree out_6 > outfile_1

./tenntyo out_6 > outfile_2
```

C　解析用プログラム（C 言語）

最小 2 情報に LAPACK（Linear Algebra Package）を用いた。http://www.netlib.org/lapack/を参照のこと。尚，コンパイルは gcc -o *"object file" "program file.c"* -O -L. -llapack -lblas -lm -lg2c を用いること。

C.1　取得したアナログ信号のエネルギー（波高）を求めるプログラム integral3.c

```
#include <stdio.h>
#include <stdlib.h>
#define NOISE1 2750
#define INTEG  150
#define NOISE2 100

int main(int argc,char *argv[]){
  FILE *fp;
  int i,j,noise_num;
  int dt[8];
  double noise1,noise2;
  double mean_noise,mean_noise2,integral;

  if(argc != 2){
    printf("enter (./integral filename)\n");
    exit(1);
  }
  if((fp=fopen(argv[1],"r"))==NULL){
    printf("not found\n");
    exit(1);
  }

  noise_num = 0;
  noise1 = 0;
  for(i=0;i<NOISE1;i++){
    for(j=0;j<8;j++){
      fscanf(fp,"%d",&dt[j]);
    }
    if(dt[0] > 0){
      noise_num++;
      noise1 += dt[0];
    }
  }
  if(noise_num > 0) mean_noise = noise1/noise_num;
  else              mean_noise = 0;

  noise_num = 0;
  integral = 0;
  for(i=0;i<INTEG;i++){
```

```
    for(j=0;j<8;j++){
      fscanf(fp,"%d",&dt[j]);
    }
    if(dt[0] > 0){
      integral += dt[0];
      noise_num++;
    }
  }

  noise2 = 0;
  mean_noise2 = 0;
  for(i=0;i<NOISE2;i++){
    for(j=0;j<8;j++){
      fscanf(fp,"%d",&dt[j]);
    }
    noise2 += dt[0];
  }
  mean_noise2 = noise2/NOISE2;

  if(mean_noise > 0){
    integral -= (mean_noise + mean_noise2)/2*(noise_num);

    if(integral > 0)
      printf("%f %s\n",integral,argv[1]);
  }
  return 0;
}
```

C.2 5 hit 以上のイベントを残して，x，y，z 座標を cm 単位に変換するプログラム time_to_cm.c

```
#include <stdio.h>
#include <stdlib.h>
#define M 70

int main(int argc, char *argv[])
{
  FILE   *fp;
  int    h, i, j, k, l, lmax;
  int    hits;
  int    Eventnum;
  double dt[M][5];
  double initial_time;

  if(argc != 3){
    printf("enter(time_to_cm filename Eventnum)\n");
    exit(1);
  }
  if((fp=fopen(argv[1], "r")) == NULL){
    printf("not found\n");
    exit(1);
  }
  Eventnum = atoi(argv[2]);

  for(h = 0; h < Eventnum; h ++){
    for(j = 0; j < 5; j ++){
      fscanf(fp, "%lf", &dt[0][j]);
    }
    hits = dt[0][0];

    if(hits > 2){
      for(i = 1; i < hits; i ++){
for(j = 0; j < 5; j ++){
  fscanf(fp, "%lf", &dt[i][j]);
}
      }

      initial_time = dt[hits-1][3];
```

```
      for(i = 0; i < hits; i ++){
dt[i][1] = (dt[i][1]/51.2);
dt[i][2] = (dt[i][2]/51.2);
dt[i][3] = (dt[i][3]-initial_time)*0.084 + 4;
      }

      for(i = 0; i < hits; i ++){
printf("%.1f\t",dt[i][0]);
for(j = 1; j < 4; j ++){
  printf("%6.3lf\t",dt[i][j]);
}
printf("%.1f\t%d\n",dt[i][4],Eventnum - h);
      }
    }
    else{
      for(i = 1; i < hits; i ++){
for(j = 0; j < 5; j ++){
  fscanf(fp, "%lf", &dt[i][j]);
}
      }
    }
  }
  fclose(fp);

  return 0;
}
```

C.3 最小 2 乗法をしてから，ノイズを落とすプログラム muonsute_variance.c

```
// using clapack
//  a,b of y = ax + b wo saisyo_nijyo_hou_no_suitei
// xz,yz sya_ei

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define Enum 10000
#define M 80     // max number of the date
#define N 2      // unknown number

int main(int argc, char *argv[])
{
  FILE *fp;
  int h, i, j, k;
  int flag, flagg, hits;
  double dt[M*N][6];
  double A1[M*N], B1[M], work[M+N], A2[M*N], B2[M], lenxy, lenyz, l;
  double variance_xz, variance_yz;
  double deviation_xz, deviation_yz;
  double std_deviation_xz, std_deviation_yz;
  long m, n, lda, ldb, info, ldw, nrhs;

  if(argc != 2){
    printf("enter(./muonsute filename)\n");
    exit(1);
  }
    if((fp=fopen(argv[1], "r")) == NULL){
    printf("not found\n");
    exit(1);
  }

  for(h = 0; h < Enum; h ++){
    for(j = 0; j < 6; j ++){
      if(fscanf(fp, "%lf",&dt[0][j]) == EOF ) exit(1);
    }
    hits = dt[0][0];

    for(i = 1; i < hits; i ++){
```

```
      for(j = 0; j < 6; j ++){
fscanf(fp, "%lf",&dt[i][j]);
      }
    }

    for(k = hits; k < hits*N; k ++){
      for(j = 0; j < 5; j ++){
dt[k][j] = 1.0;
      }
    }

    for(i = 0; i < hits; i++){
      A1[i] = dt[i][3], A1[i+hits] = dt[i+hits][3];
      B1[i] = dt[i][1];
    }
    m=hits, n=N, lda=hits, ldb=hits, info, ldw=hits+N, nrhs=1;
    // saisyo_nijyohou_ clapack   DGELS
    dgels_("N", &m, &n, &nrhs, A1, &lda, B1, &ldb, work, &ldw, &info);

    for(i = 0; i < hits; i++){
      A2[i] = dt[i][3], A2[i+hits] = dt[i+hits][3];
      B2[i] = dt[i][2];
    }
    m=hits, n=N, lda=hits, ldb=hits, info, ldw=hits+N, nrhs=1;
    // saisyo_nijyohou_ clapack   DGELS
    dgels_("N", &m, &n, &nrhs, A2, &lda, B2, &ldb, work, &ldw, &info);

    variance_xz = 0;
    for(i = 0; i < hits; i ++){
      variance_xz +=
(1/((double) hits)) * pow((B1[0]*dt[i][3]+B1[1]-dt[i][1]),2.0);
    }
    std_deviation_xz = sqrt(variance_xz);

    variance_yz=0;
    for(i = 0; i < hits; i ++){
      variance_yz +=
(1/((double) hits)) * pow((B2[0]*dt[i][3]+B2[1]-dt[i][2]), 2.0);
    }
    std_deviation_yz = sqrt(variance_yz);

    flag = 0;
    l = 1.0;

    if(std_deviation_xz > 2.5) flag = 1;
    if(std_deviation_yz > 2.5) flag = 1;

    if(flag == 0){
      for(i = 0; i < hits; i ++){
flagg = 0;

deviation_xz = fabs(B1[0]*dt[i][3]+B1[1] - dt[i][1]);
deviation_yz = fabs(B2[0]*dt[i][3]+B2[1] - dt[i][2]);

if(deviation_xz > std_deviation_xz) flagg = 1;
if(deviation_yz > std_deviation_yz) flagg = 1;

if(flagg == 0){
  printf("%.lf\t%lf\t%lf\t%lf\t%.lf\t%.lf\n",
 l,dt[i][1],dt[i][2],dt[i][3],dt[i][4],dt[i][5]);
  l = l + 1.0;
}
      }
    }
  }
  fclose(fp);

  return 0;
}
```

C.4　3 hit 以上のイベントを残すプログラム muon_than3.c

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

int main(int argc, char *argv[])
{
  FILE   *fp;
  int    i, j, k;
  int    numevent;
  int    hits;
  double dt[6];

  if(argc != 3){
    printf("enter(than5 filename numevent)\n");
    exit(1);
  }
  if((fp=fopen(argv[1], "r"))==NULL){
    printf("not found\n");
    exit(1);
  }

  numevent = atoi(argv[2]);
  hits = 0;

  for(k = 0; k < numevent; k ++){
    for(j = 0; j<6; j ++){
      if(fscanf(fp, "%lf",&dt[j]) == EOF)exit(1);
    }
    hits = dt[0];

    if(hits > 2){
      printf("%.lf\t%lf\t%lf\t%lf\t%.lf\t%.lf\n",
     dt[0],dt[1],dt[2],dt[3],dt[4],dt[5]);

      for(i = 1; i < hits; i ++){
for(j = 0; j < 6; j ++) fscanf(fp, "%lf",&dt[j]);

printf("%.lf\t%lf\t%lf\t%lf\t%.lf\t%.lf\n",
                dt[0],dt[1],dt[2],dt[3],dt[4],dt[5]);
      }
    }
    else{
      for(i = 1; i < hits; i ++){
for(j = 0; j < 6; j ++){
  fscanf(fp, "%lf", &dt[j]);
}
      }
    }
  }
  fclose(fp);

  return 0;
}
```

C.5　最小 2 乗法をして，天頂角を出すプログラム muon_degree.c

```
// using clapack
//  a,b of y = ax + b wo saisyo_nijyo_hou_no_suitei
// zx,zy sya_ei

#include <stdio.h>
#include <stdlib.h>
#define Enum 10000
#define M 80     // max number of the date
#define N 2      // unknown number

int main(int argc, char *argv[])
```

```
{
  FILE *fp, *fout;
  int h, i, j, k, lmax, flag, hits;
  double dt[M*N][6];
  double A1[M*N], B1[M], work[M+N], A2[M*N], B2[M], lenxy, lenyz, l;
  double theta, tantheta;
  long m, n, lda, ldb, info, ldw, nrhs;

  if(argc != 2){
    printf("enter(./muon_degree filename)\n");
    exit(1);
  }
   if((fp = fopen(argv[1], "r")) == NULL){
    printf("not found\n");
    exit(1);
  }

  for(h = 0; h < Enum; h ++ ){
    for(j = 0; j < 6; j ++){
      if(fscanf(fp, "%lf", &dt[0][j]) == EOF ) exit(1);
    }
    hits = dt[0][0];

    for(i = 1; i < hits; i ++){
      for(j = 0; j < 6; j ++){
fscanf(fp, "%lf", &dt[i][j]);
      }
    }
    for(k = hits; k < hits*N; k ++){
      for(j = 0; j < 6; j ++){
dt[k][j] = 1.0;
      }
    }

    for(i = 0; i < hits; i ++){
      A1[i] = dt[i][3], A1[i+hits] = dt[i+hits][3];
      B1[i] = dt[i][1];
    }

    m=hits, n=N, lda=hits, ldb=hits, info, ldw=hits+N, nrhs=1;
    // saisyo_nijyohou_ clapack   DGELS
    dgels_("N", &m, &n, &nrhs, A1, &lda, B1, &ldb, work, &ldw, &info);

    for(i = 0; i < hits; i ++){
      A2[i] = dt[i][3], A2[i+hits] = dt[i+hits][3];
      B2[i] = dt[i][2];
    }

    m=hits, n=N, lda=hits, ldb=hits, info, ldw=hits+N, nrhs=1;
    // saisyo_nijyohou_ clapack   DGELS
    dgels_("N", &m, &n, &nrhs, A2, &lda, B2, &ldb, work, &ldw, &info);

    tantheta = sqrt(B1[0]*B1[0] + B2[0]*B2[0]);
    theta = atan(tantheta);

    printf("%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%.lf\n",
   atan(B1[0])*180/3.1415, atan(B2[0])*180/3.1415,
   theta*180/3.1415, sin(theta), cos(theta),tantheta, dt[0][5]);
  }
  fclose(fp);

  return 0;
}
```

C.6　天頂角分布を求めるプログラム muon _tenntyo.c

```
// using clapack
#include <stdio.h>
```

```
#include <stdlib.h>
#include <math.h>
#define Enum 10000
#define M 80     // max number of the date
#define N 2      // unknown number

int main(int argc , char *argv[] )
{
  FILE *fp;
  int h, i, j, k, lmax, flag, sig, hits;
  double dt[M*N][6];
  double A1[M*N], B1[M], work[M+N], A2[M*N], B2[M], lenxz, lenyz;
  long m, n, lda, ldb, info, ldw, nrhs;
  double xx, yy;// tenntyo .xx=sin(the)cos(phi),yy=sin(the)cos(phi)

  if(argc!=2){
    printf("enter(muon_degree input_filename > out)\n");
    exit(1);
  }
  if((fp=fopen(argv[1],"r"))==NULL){
    printf("not found\n");
    exit(1);
  }

  for(h = 0; h < Enum; h ++ ){
    for(j = 0; j<6; j++){
      if(fscanf(fp,"%lf",&dt[0][j]) == EOF)exit(1);
    }
    hits=dt[0][0];

    for(i = 1; i < hits; i ++){
      for(j=0; j<6; j++){
fscanf(fp, "%lf", &dt[i][j]);
      }
    }
    for(k = hits; k < hits*N; k ++){
      for(j=0; j<6; j++){
dt[k][j] = 1.0;
      }
    }

    for(i = 0; i < hits; i ++){
      A1[i] = dt[i][3], A1[i+hits] = dt[i+hits][3];
      B1[i] = dt[i][1];
    }
    m=hits, n=N, lda=hits, ldb=hits, info, ldw=hits+N, nrhs=1;
    // saisyo_nijyohou_ clapack   DGELS
    dgels_("N", &m, &n, &nrhs, A1, &lda, B1, &ldb, work, &ldw, &info);

    for(i = 0; i < hits; i ++){
      A2[i] = dt[i][3], A2[i+hits] = dt[i+hits][3];
      B2[i] = dt[i][2];
    }
    m=hits, n=N, lda=hits, ldb=hits, info, ldw=hits+N, nrhs=1;
    // saisyo_nijyohou_ clapack   DGELS
    dgels_("N", &m, &n, &nrhs, A2, &lda, B2, &ldb, work, &ldw, &info);

    if((B1[0] >= 0) && (B2[0] >= 0)){
      xx =    sqrt(1-1/(pow(B2[0],2)+pow(B1[0],2)+1))*cos(atan((B2[0])/B1[0]));
      yy =    sqrt(1-1/(pow(B2[0],2)+pow(B1[0],2)+1))*sin(atan((B2[0])/B1[0]));
    }
    if((B1[0] >= 0) && (B2[0] < 0)){
      xx =    sqrt(1-1/(pow(B2[0],2)+pow(B1[0],2)+1))*cos(atan((B2[0])/B1[0]));
      yy =    sqrt(1-1/(pow(B2[0],2)+pow(B1[0],2)+1))*sin(atan((B2[0])/B1[0]));
    }
    if((B1[0] < 0) && (B2[0] >= 0)){
      xx = -1*sqrt(1-1/(pow(B2[0],2)+pow(B1[0],2)+1))*cos(atan((B2[0])/B1[0]));
      yy =    sqrt(1-1/(pow(B2[0],2)+pow(B1[0],2)+1))*sin(atan((B2[0])/B1[0]));
    }
    if((B1[0] < 0) && (B2[0] < 0)){
```

```
      xx = -1*sqrt(1-1/(pow(B2[0],2)+pow(B1[0],2)+1))*cos(atan((B2[0])/B1[0]));
      yy =    sqrt(1-1/(pow(B2[0],2)+pow(B1[0],2)+1))*sin(atan((B2[0])/B1[0]));
    }
    printf("%lf\t%lf\t%.lf\n", xx, yy, dt[0][5]);
  }
  fclose(fp);

  return 0;
}
```

D　解析用プログラム（root）

マクロの使用方法は以下の通りである。
.L *file name*.C
.x main();

D.1　2 次元画像に用いたマクロ

```
#include <fstream.h>

fe(){
  TCanvas *c1 = new TCanvas("c1","2D",0,0,400,400);
  gStyle->SetOptStat(0000000);
  TH2F *h1 = new TH2F("Fe","2D-image",1050,0,1050,1050,0,1050);

  Double_t x,y;
  ifstream data("filename");
  while(data >> x >> y)h1->Fill(1024 - x , y);
  data.close();

  h1->Draw("COLTZ");
}
```

D.2　muon の $\cos^n$ 則に用いたマクロ

```
#include <fstream.h>
void cos(){
  Double_t len  = 4.0;   //muon no hiseki
  Double_t sci  = 6.0;   //scintillator kyori
  Double_t time = 14832;
  Double_t norm = 59.68; //kikakuka

  TCanvas *c1 = new TCanvas("c1", "POMTA", 5, 5, 350, 350);
  gPad->SetLogx();
  gPad->SetLogy();

  ifstream data("outfile_1");
  Double_t x, y, z, w, k, l, m;
  TH1F *h1 = new TH1F("POMTA","cos theta", 100, 0.1, 1);
  while(data >> x >> y >> z >> w >> k >> l >> m)
    h1->Fill(k,0.01/time/6.283*norm);
  data.close();
  h1->Draw();

  TCanvas *c2=new TCanvas("c2","POMTA", 5, 5, 350, 350);
  gPad->SetLogx(0);
  gPad->SetLogy(0);

  ifstream data("outfile_1");
  Double_t x, y, z, w, k, l, m;
  TH1F *h2 = new TH1F("POMTA","cos theta + fit", 100, 0.1 ,1);
  while(data >> x >> y >> z >> w >> k >> l >> m)
    h2->Fill(k, 0.01/time/6.283*norm);
  data.close();
```

```
  h2->Draw();

  TF1 *fn2 = new TF1("fn2", "[0]*x**[1]", 0.0, 1.0);
  fn2->SetParameters(0.000,8);
  fn2->SetParNames("A", "n");
  fn2->SetLineStyle(2);
  fn2->SetLineColor(4);
  h2->Fit("fn2");

  TCanvas *c3 = new TCanvas("c3", "POMTA", 5, 5, 350, 350);
  gPad->SetLogx(0);
  gPad->SetLogy(0);

  ifstream data("outfile_1");
  Double_t x, y, z, w, k, l, m;
  Double_t menseki;
  menseki = 100-12.73*(sci*l+len*w)+0.318*(sci*l+len*w)*(sci*l+len*w);
  TH1F *h3 = new TH1F("POMTA","cos theta     hosei", 100, 0.1, 1);
  while(data >> x >> y >> z >> w >> k >> l >> m){
    if(k > 0.68 && k < 1)
      h3->Fill(k,1/menseki/time/6.283*norm);
  }
  data.close();
  h3->SetLineColor(2);
  h3->Draw();
  h1->Draw("same");

}
```

謝辞

実験を進めるにあたって宇宙線研究室の身内さんには実験への取り組み方，機材の扱い方，Unix のことからデータの解析方法まで全てにおいてお世話になりました。TA の服部さんにも実験の初期にはほぼ付きっ切りで面倒をみてもらい大変お世話になりました。

何度も大きな壁に突き当たり実験が何日も進まなくなることも何度かありましたが，その度にやってきてちょちょいと問題を解決していく身内さんが憧れでもあり，また憎らしくもありました。

実験の合間におもしろいお話をしていただいた乾さん，ノートパソコンを貸していただいた岡田さん，発表会前に Power Point の本を貸していたただいた小沢さん，いつも線源を借りに行くときにお世話になった内山さんをはじめ宇宙線研究室の皆様にも大変お世話になりました。こうして無事（?）卒業研究を終えることができたのも皆様のおかげです。本当にありがとうございました。

最後になりましたが，一年間，時に励ましあいながら Y.T の声のこだまする P6 部屋で一緒に卒業研究を乗り越えてきた P6 みんなへ。ありがとう，そしてさようなら。

参考文献

[1] K. Hattori *et al.*, *Micro Pixel Chamber Operation with Gas Electron Multiplier* (2005).

[2] 小田稔, 宇宙線, 改訂版, 裳華房, 199–204, 261–271 (1972).

[3] 松原望, 統計学入門, 東京大学出版会, 35–37 (1991).

[4] G. L. Squires, *Practical Physics*, 4th edition, Cambridge University Press (2001).
重川秀実，山下理恵，吉村雅満，風間重雄訳，いかにして実験をおこなうか — 誤差の扱いから論文作成へ —，丸善 (2006).

[5] 福井崇時, 粒子物理計測学入門, 共立出版, 165–166 (1992).