微細分割されたハドロンカロリメータ内における粒子飛跡からのエネルギー測定

信州大学　理学部　物理科学科
高エネルギー物理学研究室

学籍番号　**12S2002H**
氏名　飯野貴浩

指導教官

竹下　徹　教授
長谷川　庸司　准教授

はじめに

現在高エネルギー物理学では、加速した粒子を衝突させることで内部構造を明らかにし、新粒子の発見を目指す実験が行われている。その実験において衝突により発生した粒子を個々に識別する必要があり、その方法として測定器内部で粒子が作るジェットの全粒子を識別する方法が有力視されている。ジェットとは衝突によって生じた粒子がハドロン化したとき、それらのハドロンが進行歩行に束になって放出される現象である。また、高エネルギー粒子が物質内に入射することで、無数に発生した粒子が入射方向に集中して放出される。この現象をシャワーと呼び、光子や電子が作るシャワーを電磁シャワー、ハドロンが作るジェットをハドロンシャワーと呼ぶ。ゆえに、ジェットを構成する粒子のエネルギーを測定するためには、シャワー内粒子のエネルギーを測定する必要がある。

シャワーを構成する粒子のエネルギーを測定する際、カロリメータという測定器を粒子の通過位置をある程度正確に検出できるように細分化して使用する。カロリメータは光子や電子のエネルギーを測定する電磁カロリメータ（Electromagnetic Calorimeter：略称 ECAL）とハドロンのエネルギーを測定するハドロンカロリメータ（Hadron Calorimeter : 略称 HCAL）の２種類に分類されている。ハドロンが HCAL 内に落とすエネルギーは反応ごとに激しく変動する。そのためジェットを構成するハドロンのエネルギーを測定するためにはハドロンシャワーを構成する粒子一個一個の HCAL 内におけるエネルギーを反応ごとに測定することが求められる。そこで本研究では、ハドロンの物質内のおける反応をいくつかに種別し、安定な状態での正確なエネルギー測定法を確立することにより、ジェットエネルギー分解能改善の一助とすることを目標とした。具体的には、ハドロンシャワー特有のトレース状飛跡が磁場内で曲率を持つことを利用した。本実験では HCAL を Geant4というシミュレーションソフトを使用して作成し、得られたデータを root によって解析した。

前半では HCAL の構造ごとに入射エネルギーと荷電粒子のトレースの測定エネルギーの比及び測定値の二乗平均平方根（RMS）を評価し、構造の変更による大きな改善は望めないことがわかった。後半では入射粒子のエネルギーが HCAL 内で常に減衰し続けていることに注目し、解析方法の改善について考察した。

目次

1. 研究概要

1.1 目的

本実験の目的は、HCAL のジェットエネルギー分解能の向上であり、そのためのシミュレーションと得られたデータの解析を以下の手順で実行した。

1. 0.5GeV のエネルギーを持つミュー粒子1000個を HCAL 内に入射し、エネルギーが落ちた検出層のタイルを記録する。同様のシミュレーションを1.0GeV,1.5GeV,2.0GeV でも行う。

2. エネルギーが落ちたタイルの位置から粒子の飛跡をグラフ上にプロットし、root を用いてフィッティングを行うことで曲率半径を求め、曲率半径からエネルギーを求める。

3. 求まったエネルギーをヒストグラムにまとめ、その Mean 値（平均値）と RMS 値を用いて入射エネルギー毎の「測定エネルギー/入射エネルギー」をプロットし変化を見る。

1.2 ILC とハドロンカロリメータ

国際リニアコライダー（International Linear Collider : 略称 ILC）とは、現在計画されている電子と陽電子の衝突実験を行うための直線型加速器である。衝突によって様々な粒子が高エネルギーの状態で現れ、発生したそれらを測定器でキャッチし、得られたデータを解析する。その測定器の一つがハドロンのエネルギーを測定するハドロンカロリメータ（HCAL）であり、本実験では次世代の加速器である ILC での使用に足るだけの高いエネルギー分解能を持つデータ解析法を研究した。カロリメータは粒子にストップをかける吸収層と実際に粒子を検出する検出層の二層を重ねたレイヤーを更に重ねあわせて作成されている。本実験では吸収層に鉄を、検出層にシンチレータを用いた。

2.　ハドロンの物質内での振る舞い

シミュレーションによる、HCAL 内に入射された高エネルギーのハドロンの様子を図1に示す。

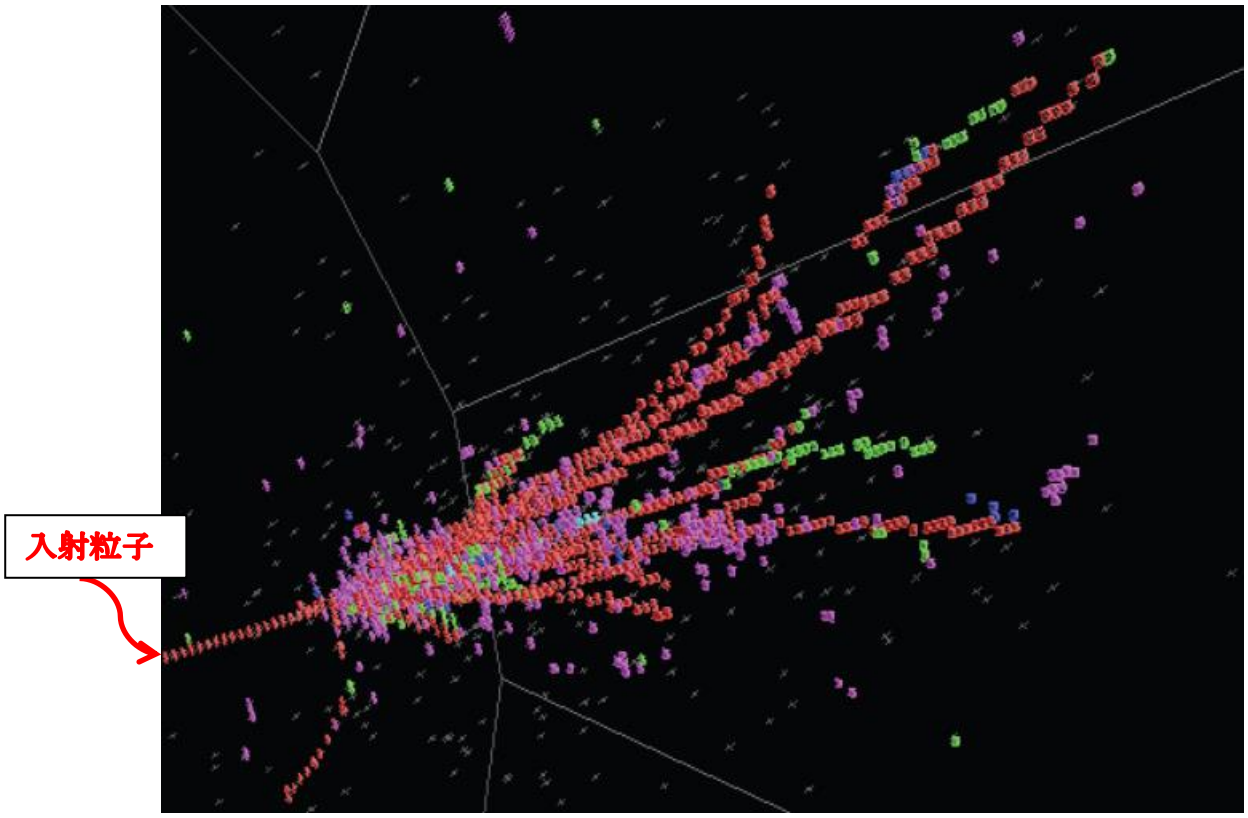

図1　シミュレーションによるハドロンシャワーの様子。入射粒子は100GeV の負の電荷を持つパイ中間子トレース状飛跡が磁場の影響により曲率を持っている様子が赤いプロットで示されている。
[筑波大学　山口佳博氏の修士論文より引用]

ハドロンの HCAL 内における振る舞いを、電磁シャワーを起こす電磁過程とハドロンが核子と相互作用をするハドロン過程の２種類に分類し、更にハドロン過程を核子反応過程と MIP トレース過程の２種類に分類した。図2にハドロンシャワーの概略を示し、以下に詳細を述べる。

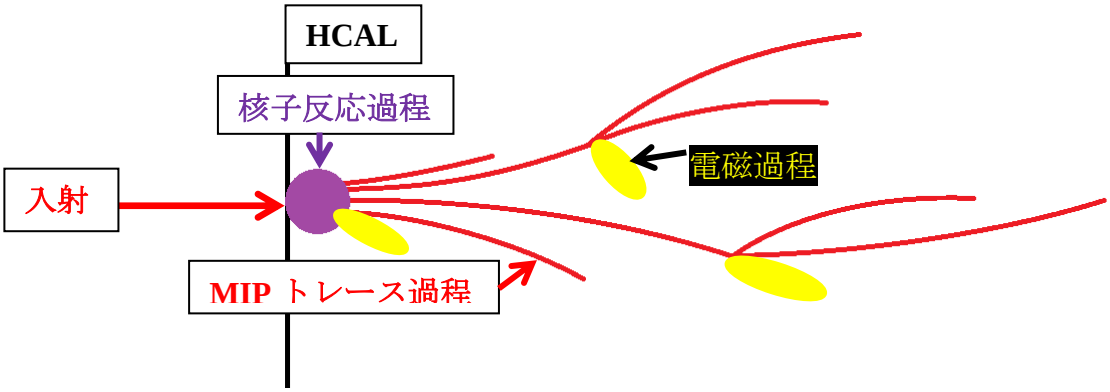

図2　ハドロンシャワーの概略図。入射粒子が物質内でシャワーを起こしパイ中間子を放出する。放出されたのが荷電パイ中間子であれば安定な状態で飛跡を描いた後、さらにハドロンを放出する。中性パイ中間子であれば光子2個を放出し電磁過程を取る。

2.1 電磁過程

中性パイ中間子を経て高エネルギーの光子２個を放出。その光子が核子と反応を起こし電子ー陽電子対を生成する。それらは物質内部で制動放射を起こし高エネルギーの光子を放出、光子が電子ー陽電子対を生成する。この反応が繰り返され粒子がシャワーのように広がっていく現象（電磁シャワー）を起こす過程。図3に概略を示す。

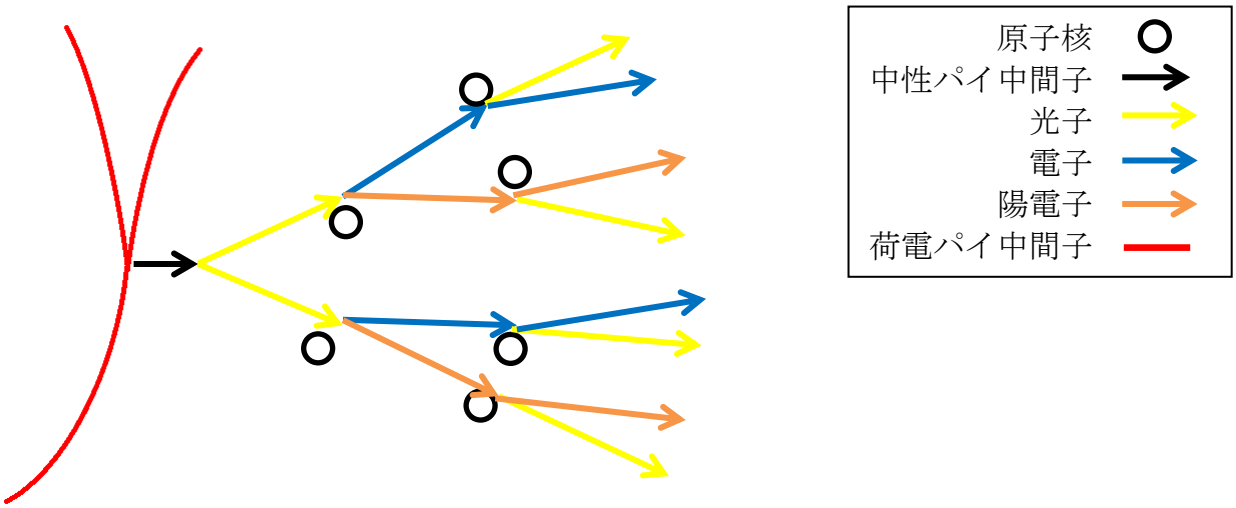

図3　電磁過程の概略図。中性パイ中間子は寿命が非常に短いため、荷電パイ中間子から直接電磁シャワーが発生しているように見える。

2.2 核子反応過程

ハドロンが HCAL 内の陽子や中性子と反応する過程。この過程においては、反応の結果大きなエネルギーを落としたり、電気的に中性になる場合がある。中性の粒子は電子と相互作用をしないので、シンチレータでエネルギーを検出できなくなり、エネルギーが消えて見える場合がある。ハドロンが HCAL 内で落とすエネルギーが反応ごとに大きく異なるのは核子反応過程におけるゆらぎに由来する。

2.3 MIP トレース過程

核子反応過程において生成された荷電ハドロンが、MIP（Minimum Ionizing Particle : 最小電離粒子）の状態でトレース状飛跡を描いて移動する過程。MIP とは粒子自身は変化せず、周囲の原子を電離する粒子の状態。飛跡は磁場内でエネルギーに対応した曲率を持つ。

本実験では安定な状態である MIP トレース過程におけるエネルギーを曲率から求めることを試みた。そのため、パイ中間子の MIP トレース過程と近似できる振る舞いをする安定な粒子であるミュー粒子を用いた。

3. シミュレーション及び解析の方法と結果

3.1 Geant4

Geant4とは粒子が物質内を通過するときの反応をコンピュータ上で再現できるシミュレーションソフトウェアである。使用するカロリメータの材質や形状を、ソースコードを書き換えることで設定し、入射する粒子の種類、エネルギー、個数、位置及び空間にかける磁場等を設定したマクロファイルを実行することで、粒子の入射シミュレーションを行うことができる。Geant4を実際に用いた例として､同じ条件のカロリメータに異なるエネルギーのミュー粒子を入射した場合のイベントディスプレイを図４に示す。

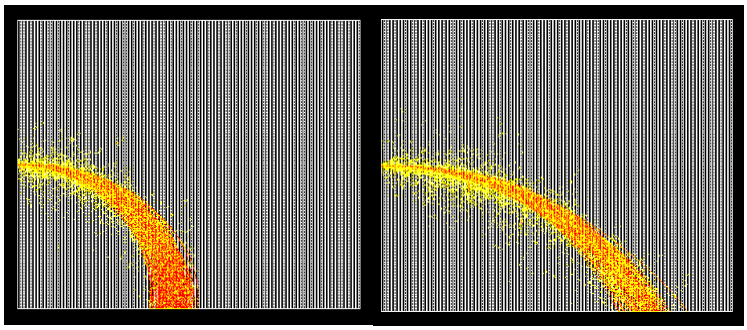

図4　Geant4で実行したシミュレーションのイベントディスプレイ

左が1.0GeV、右が2.0GeV である。いずれも入射粒子数は1000個、印加されている磁場の大きさは3.5T で方向は紙面垂直奥向である。粒子の飛跡が赤い曲線で、エネルギーの落ちた点が黄色い点で示されている。

3.2 シミュレーションに用いたカロリメータ

本研究で用いたカロリメータの概略を図5に示す。

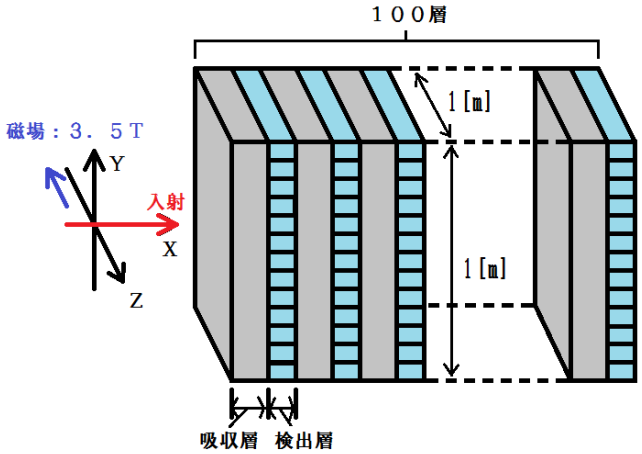

図5　シミュレーションで用いたカロリメータの概略図

粒子の通過位置はエネルギーの落ちたストリップがどのレイヤーの下から何番目にあったかで検出する。また、設定した座標軸は右手系で、中心の位置はカロリメータの左端面の中央にして粒子の入射位置と一致するようにし、入射方向に X 座標を取った。

吸収層には鉄を、検出層には Y 軸方向にスプリットしたプラスチックシンチレータ（ビニルトルエン）を使用する。レイヤーの数は100、粒子の入射位置はカロリメータの中心、印加した磁場は大きさ3.5T で Z 軸に反平行方向である。吸収層の厚さ及び検出層のストリップ幅を10mm、20mm の場合でそれぞれシミュレーションを行う。

3.3　エネルギーの導出方法

Geant4を用いて行ったシミュレーション内で、エネルギーの落ちたストリップの位置から粒子の通過位置を検出し、グラフ上にプロットする。そのプロットに合わせて円の方程式でフィッティングを行うことで飛跡の曲率半径を求める。フィッティングに用いた式を以下に示す。

$$y = \sqrt{R^2[\mathrm{m}] - (x-a)^2} + b$$

a と b は中心座標の X 座標と Y 座標、R は飛跡の曲率半径であり、曲率半径から粒子のエネルギーを導出する。

用いた式を以下に示す。

$$E[\mathrm{GeV}] = \frac{0.3B[\mathrm{T}]R[\mathrm{m}]}{\beta}, \beta = \frac{v}{c}$$

c は光速、vは粒子の速さであり、本実験ではβ~1である。フィッティングは root を用いて行った。図６にフィッティングの様子を示す。

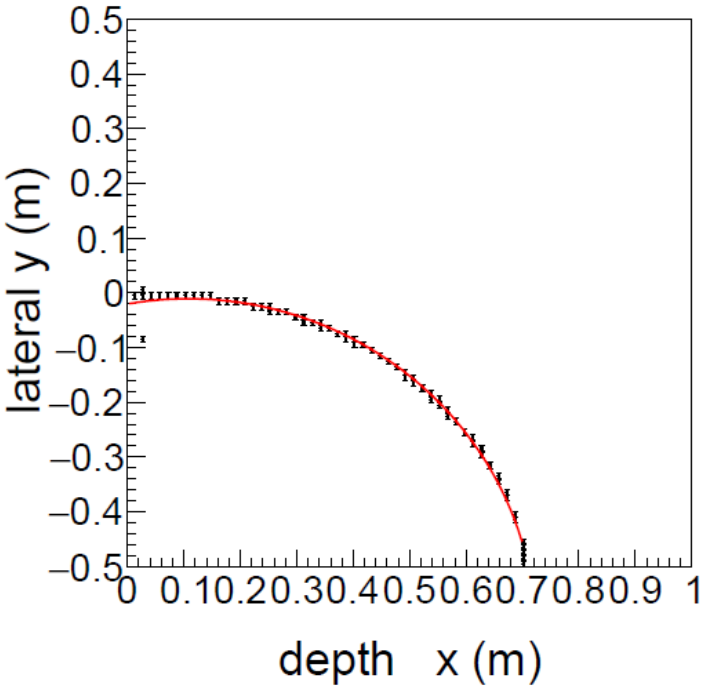

図６　1.0GeV のミュー粒子をカロリメータの中心から X 軸方向に一個入射した場合の通過位置を表すプロットとそのフィッティング曲線。カロリメータは吸収層の厚さが10mm、検出層の分割幅が10mm。この例での測定結果はそれぞれ a=0.11m , b=-0.63m , R=0.62m , E=0.65GeV

入射ごとにフィッティングを行い、曲率半径からエネルギーを求め、得られたデータを用いてヒストグラムを作成する。図７に入射エネルギー1.0GeV のミュー粒子1000個分の測定データを用いて作成したヒストグラムを示す。

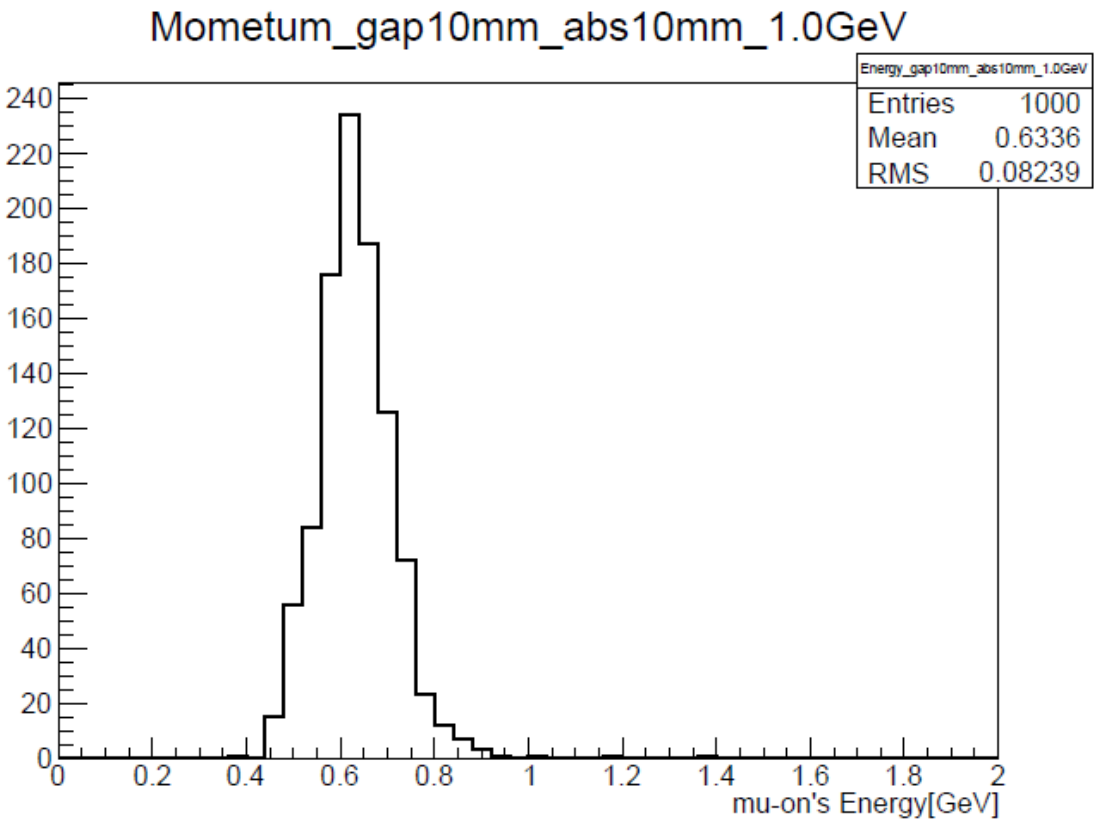

図7　入射粒子1000個の測定データによるヒストグラム

作成したヒストグラムの Mean 値を測定エネルギーとして扱う。測定エネルギーを入射エネルギーで除したものと RMS 値を入射エネルギーで除したものを用いて入射エネルギーごとの測定エネルギーの値を評価する。

3.4 吸収層の厚さごとの結果

吸収層の厚さが10mm と20mm の場合の結果を図8に示す。なお、検出層の分割幅は10mm である。

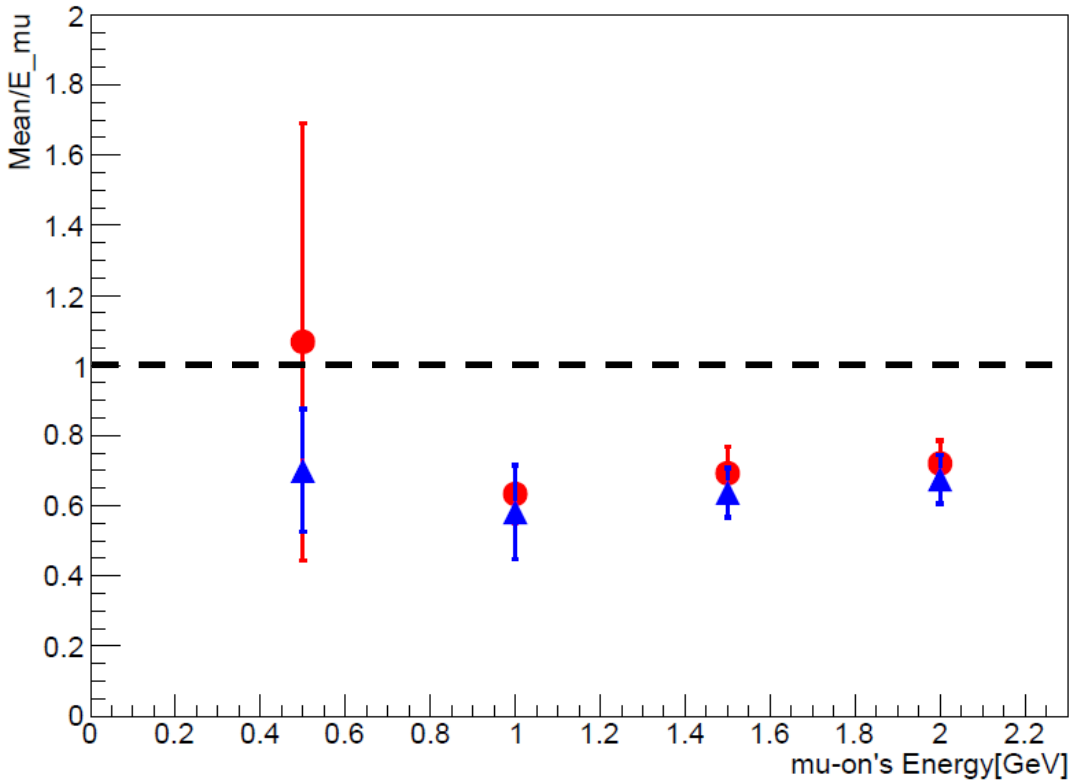

図8　吸収層の厚さごとの入射エネルギーと測定エネルギーの比。
赤いプロットが10mm、青いプロットが20mm。破線は比が1になる位置を表す。

横軸に入射エネルギー、縦軸に入射エネルギーと測定エネルギーの比の値を取った。RMS 値をプロットごとの縦棒とし、表示している。入射エネルギーが0.5GeV の点における比が1に近く、一見すると良い値をとっているように見えるが、RMS 値は一番大きくなっている。そこで、実際にどのような値をとっているのかヒストグラムを見て確認する。図9にそのヒストグラムを示す。

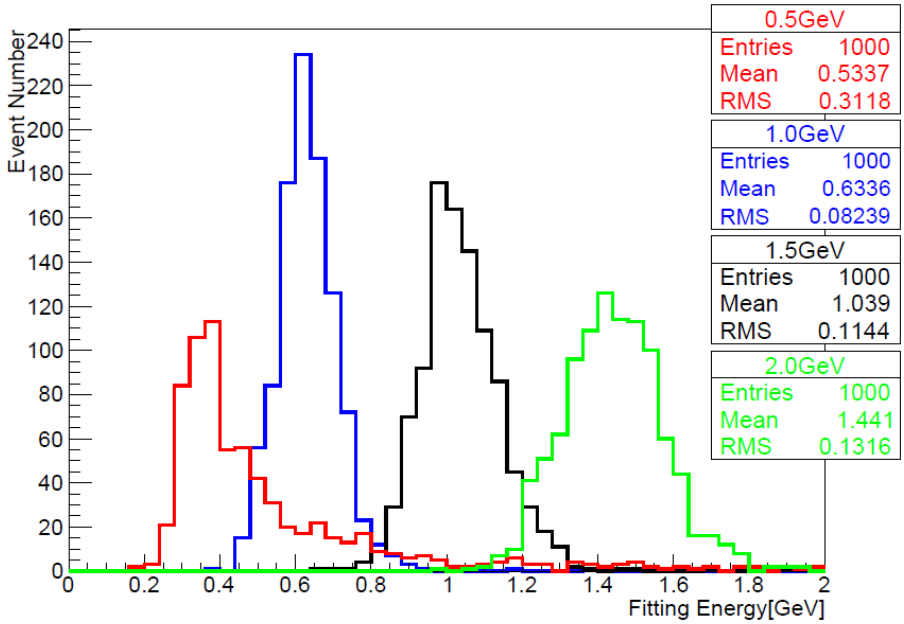

図9　入射エネルギー0.5,1.0,1.5,2.0GeV のミュー粒子1000個の測定データを用いたヒストグラム。

吸収層の厚さは10mm で、得られたデータ1000を全て用いた。

本来ありえない0.5GeV 以上の値（異常値）を持つ測定データが多数存在している様子が確認できる。そこで$\frac{\chi^2}{NDF} < 10$（NDF：自由度）の値のみを用いることで、異常値をカットしたデータを用意してヒストグラムを作ったところ、図10のようになった。

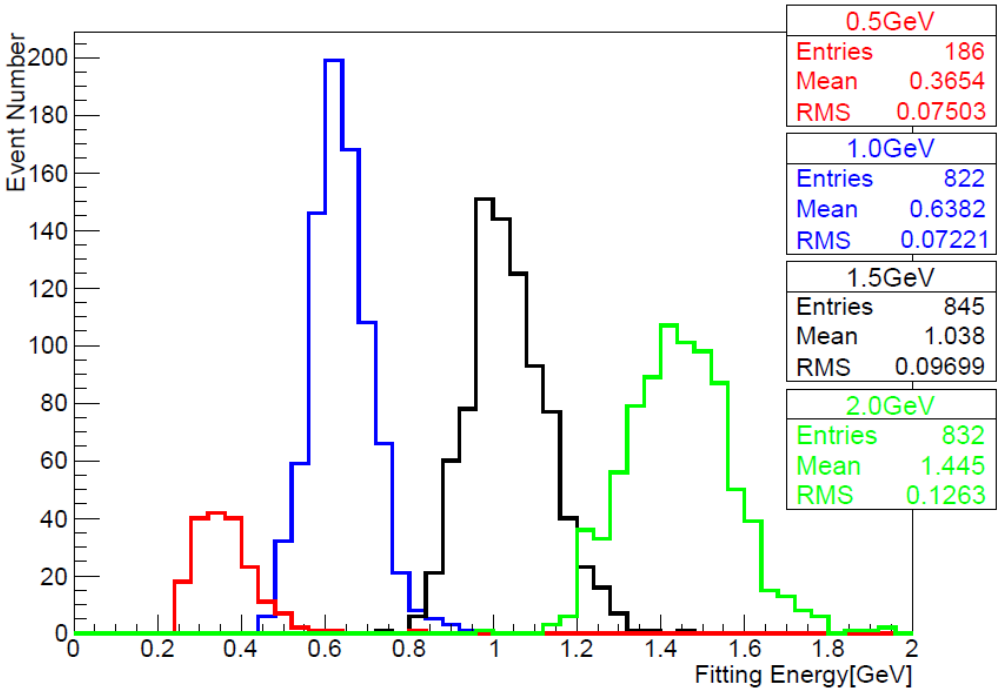

図10　円にフィッティングした時の$\frac{\chi^2}{NDF}$の値が10未満($\frac{\chi^2}{NDF} < 10$)となる測定データのみを用いたヒストグラム。

この値を測定データとして用いて作ったプロットが以下の図11である。

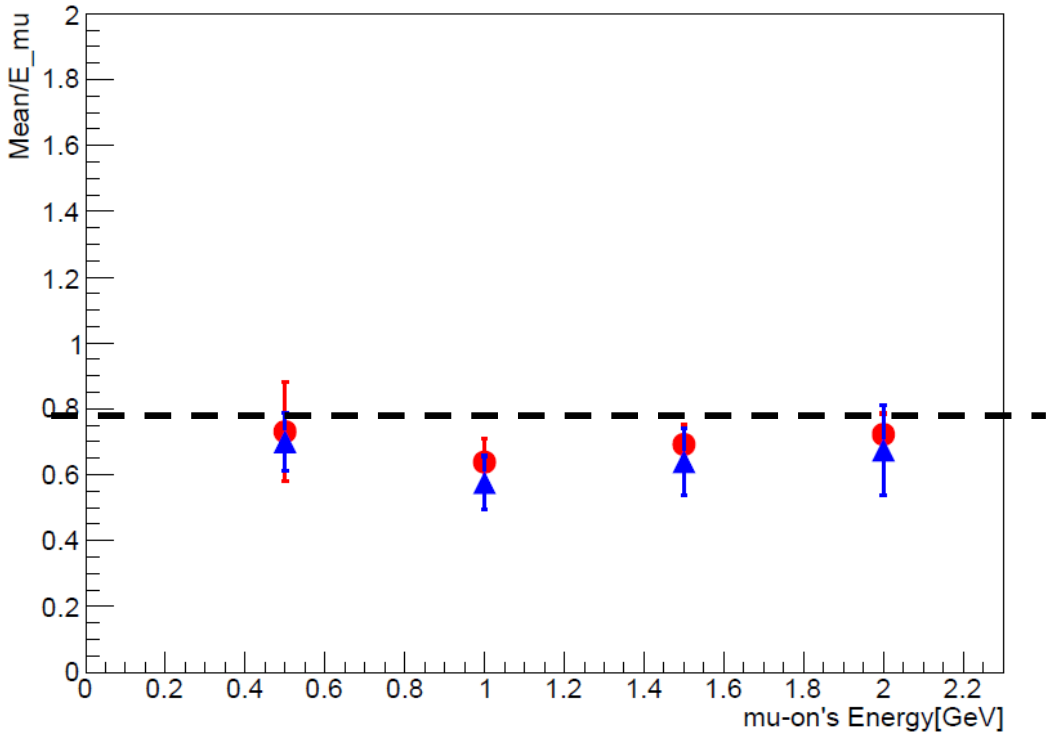

図11　図10のヒストグラムの Mean 値と RMS 値を用いたプロット。

この図から、入射エネルギーごとの測定エネルギー/入射エネルギーの値はほぼ一定になることが

わかる。具体的には以下の表1、表2のようになる。

入射エネルギー[GeV]	0.5	1.0	1.5	2.0
測定/入射エネルギー	0.74	0.64	0.70	0.73

表1　吸収層の厚さが10mm の場合

入射エネルギー[GeV]	0.5	1.0	1.5	2.0
測定/入射エネルギー	0.71	0.58	0.64	0.68

表2　吸収層の厚さが20mm の場合

よって、以下の測定データは全て円にフィッティングした時$\frac{\chi^2}{NDF}$の値が10未満($\frac{\chi^2}{NDF} < 10$)となる測定データのみを用いる。

3.5 検出層の分割幅ごとの結果

検出層の分割幅が10mm と20mm の場合の結果を図12に示す。なお吸収層の厚さは10mm である。

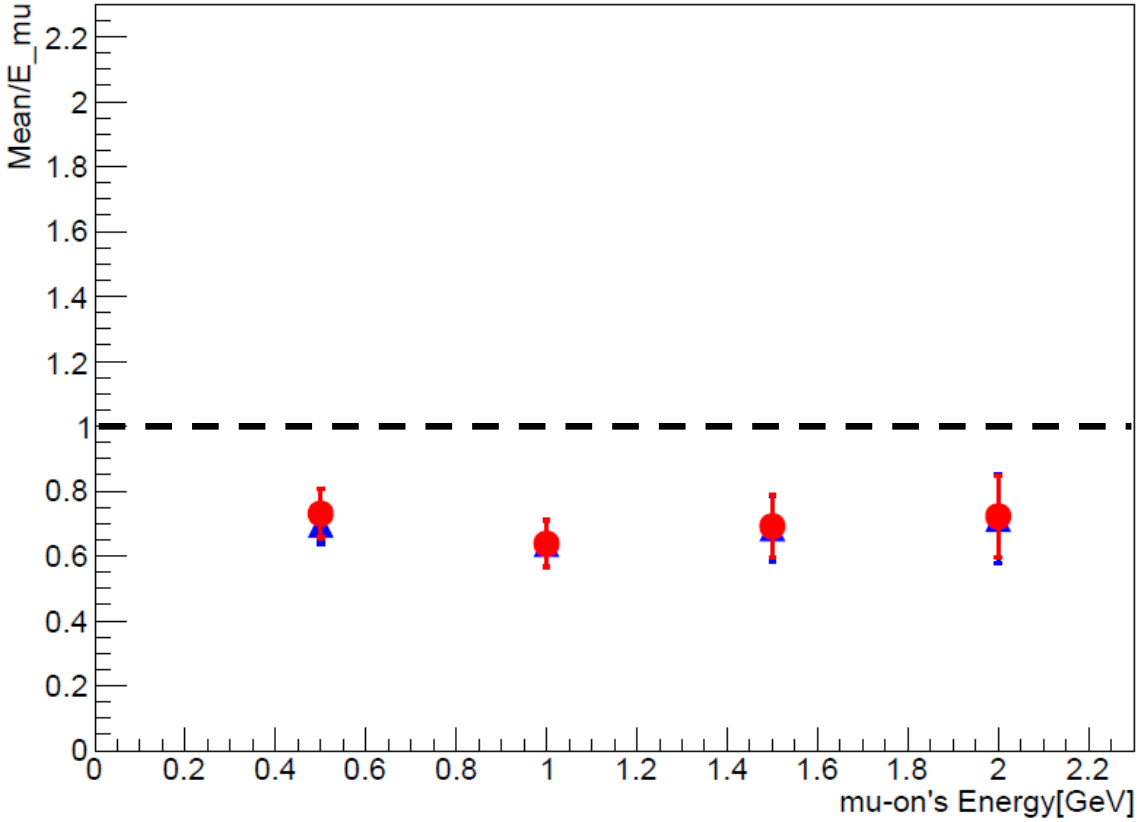

図12　検出層の分割幅ごとの入射エネルギーと測定エネルギーの比。赤いプロットが10mm、青いプロットが20mm である。

具体的な比の値を表3、表4に示す。

入射エネルギー[GeV]	0.5	1.0	1.5	2.0
測定/入射エネルギー	0.74	0.64	0.70	0.73

表3　分割幅が10mm の場合

入射エネルギー[GeV]	0.5	1.0	1.5	2.0
測定/入射エネルギー	0.70	0.64	0.69	0.73

表4　分割幅が20mm の場合

また、吸収層の厚さと検出層の分割幅を同時に変えた場合を図13に示す

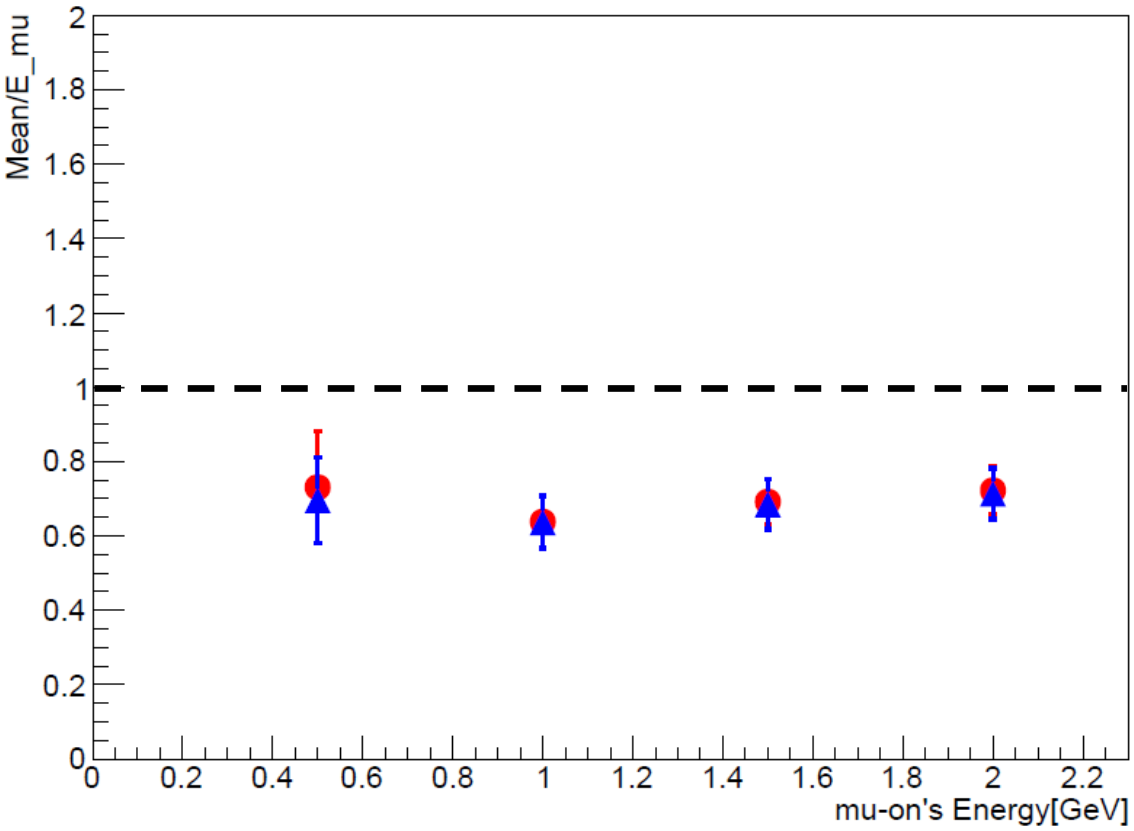

図13　吸収層の厚さ及び検出層の分割幅がともに10mm の場合とともに20mm の場合の比較プロット。赤いプロットが10mm、青いプロットが20mm

具体的な比の値を表5、表6に示す。

入射エネルギー[GeV]	0.5	1.0	1.5	2.0
測定/入射エネルギー	0.74	0.64	0.70	0.73

表5　厚さ及び分割幅が10mm の場合

入射エネルギー[GeV]	0.5	1.0	1.5	2.0
測定/入射エネルギー	0.70	0.58	0.64	0.67

表6　厚さ及び分割幅が20mm の場合

3.6 フィッティングの範囲を変えた時の結果

これまでの結果から、測定エネルギーはいずれも入射エネルギーより小さくなっていることがわかる。これは、粒子が HCAL 内で移動中常にエネルギーを失い続けていることに由来している。これまでフィッティングの式に円の方程式を用いてきたのは、磁場中を移動する荷電粒子には常に進行方向と垂直にローレンツ力が働くため、円軌道を描くと考えたためである。しかし実際には常にエネルギーが減衰し続けており、それは粒子の軌道が円ではなく螺旋であることを意味する。そこで、移動とともにエネルギーが減衰する様子を確かめるため、軌道を2分割または3分割しそれぞれでフィッティングを行った。

3.6.1 2分割及び3分割した場合の結果

吸収層の厚さ及び検出層の分割幅を10mm とし、粒子の起動をプロットの個数で2等分または３等分し、それぞれでフィッティングを行って、測定エネルギーを求めた。一例として入射エネルギーが1.0GeV でのフィッティングを図14、図15に示す。

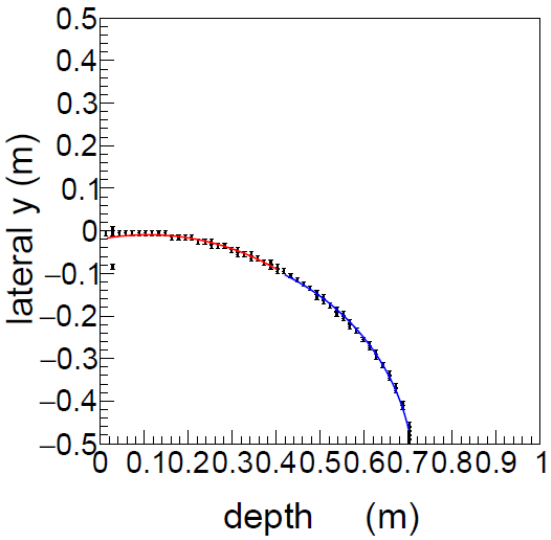

図14 プロットを前後半で2等分した場合のフィッティング。

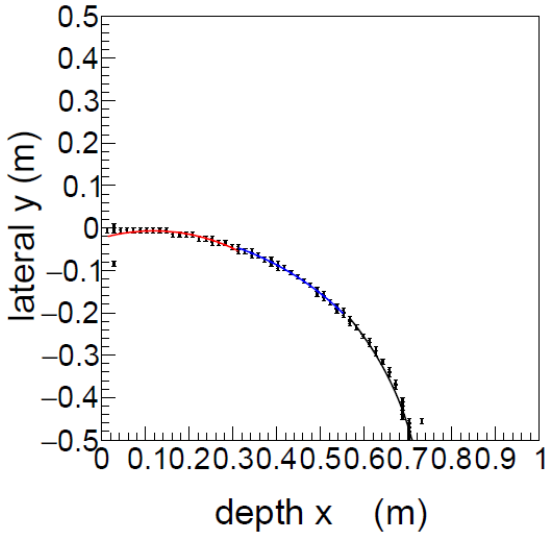

図15 プロットを3等分した場合のフィッティング。
入射粒子は入射エネルギーが1.0GeV のミュー粒子

これらから得られたデータを用いたヒストグラムを図16、図17に示す。

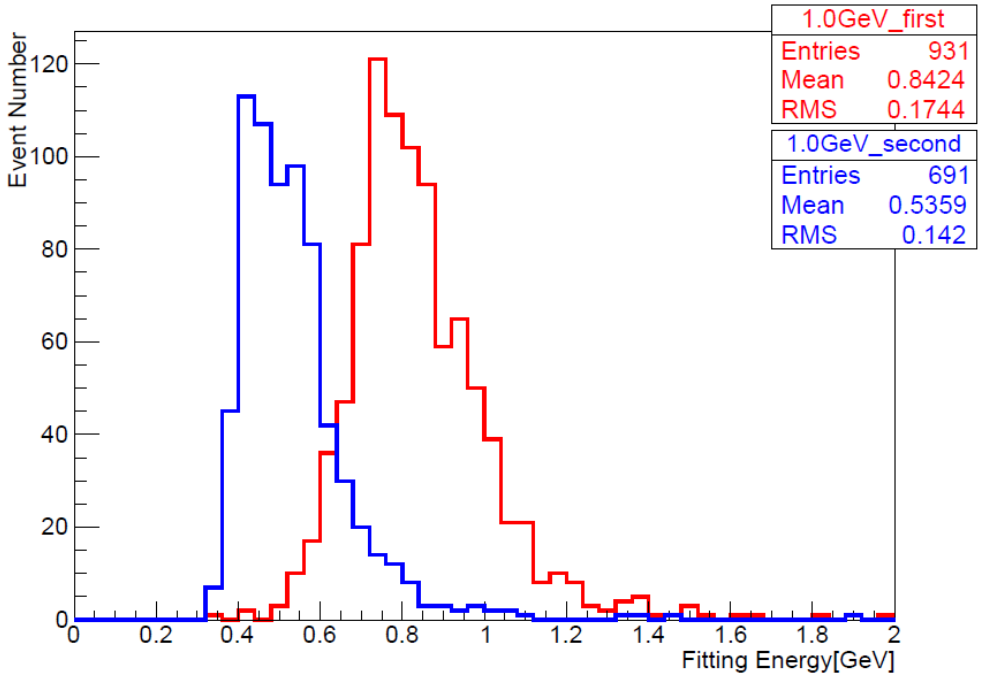

図16　赤が前半、青が後半のヒストグラム

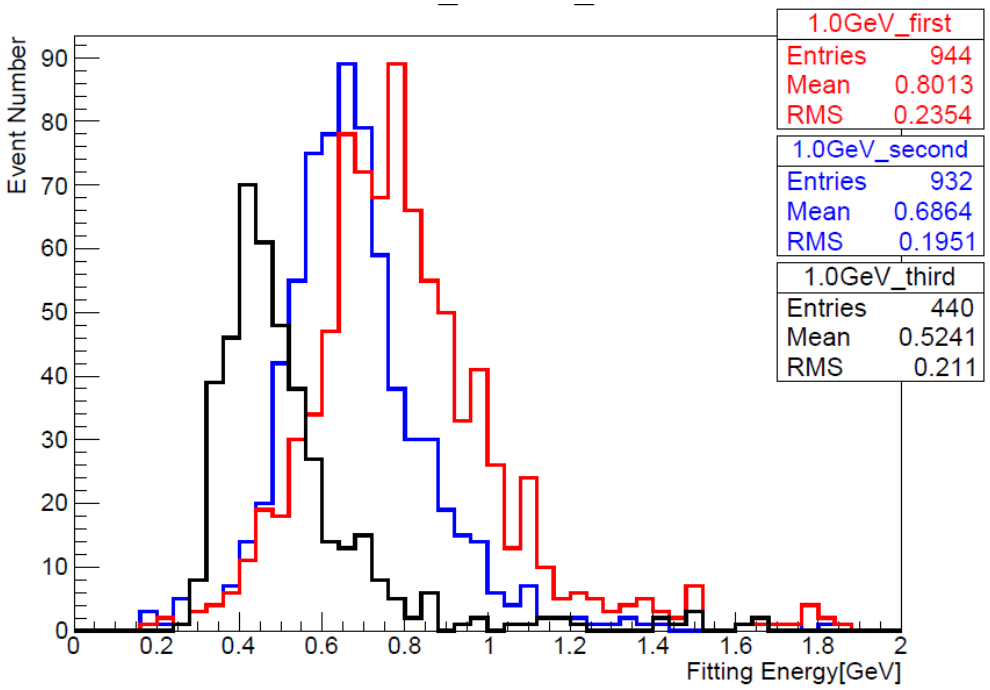

図17　三分割した時のフィッティングで求めたエネルギーのヒストグラム。
赤が始めの３分の１、青が中央、黒が最後の３分の１を表す

これらから、粒子のエネルギーが HCAL 内で常に減衰し続けていることがわかる。なお、図23の黒のヒストグラムの Mean 値がピークの位置と一致していないが、これは3.5節でも示した異常と同じ由来で発生しており、エネルギーの減衰を表すものである。

吸収層の厚さ及び検出層の分割幅がともに10mm の時の入射エネルギーごとの測定エネルギーと入射エネルギーの比の値を、２等分と３等分で比較したものを図18、図19に示す。ただし、すべての入射エネルギーで、フィッティングによる$\frac{\chi^2}{NDF}$が10未満となる測定データのみを用いた。

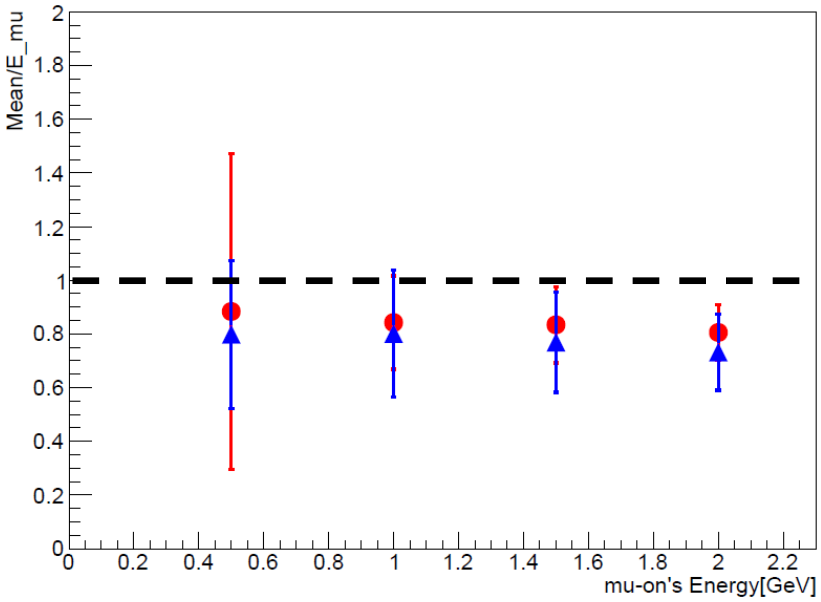

図18　2等分の前半と３等分の始めの３分の１を比較したプロット。
赤が2等分、青が3等分を表す

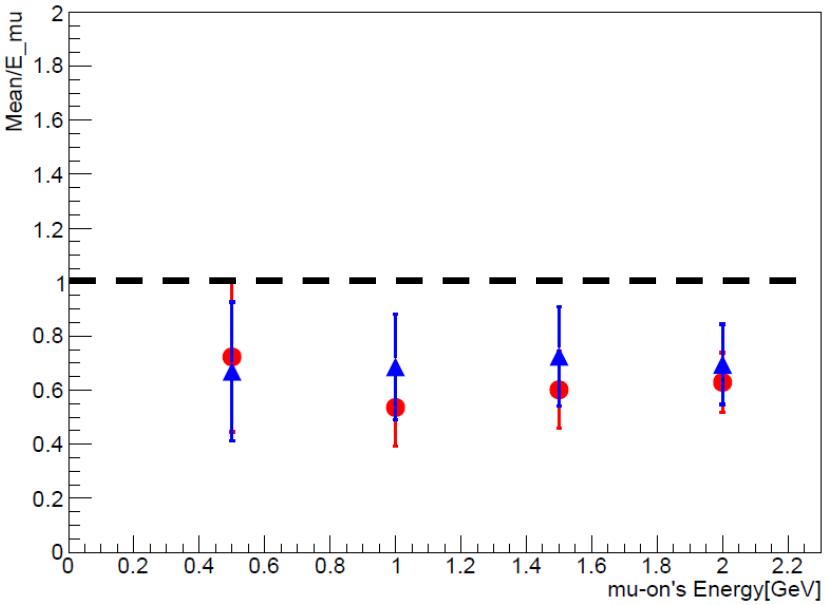

図19　2等分の後半と３等分の中央の３分の１を比較したプロット。
赤が2等分、青が3等分を表す

これらのグラフから、入射エネルギーが0.5GeV の場合にエネルギーが減衰することで異常値が増加し、それに反して1.0GeV 以上の場合はほぼ一定の値を取っていることがわかる。ゆえに、飛跡全体のエネルギーを一意に求める方法として、細分化したフィッティングで求めた値を用いた重み付き平均をとることを考えた。

フィッティングの範囲が2分の1及び3分の1の場合に、各部分でのフィッティングから求まった Mean 値を要素として重み付き平均を取った場合のプロットを図20に、具体的な数値を表7、表8に示す。エラーバーには標準誤差を用いた。

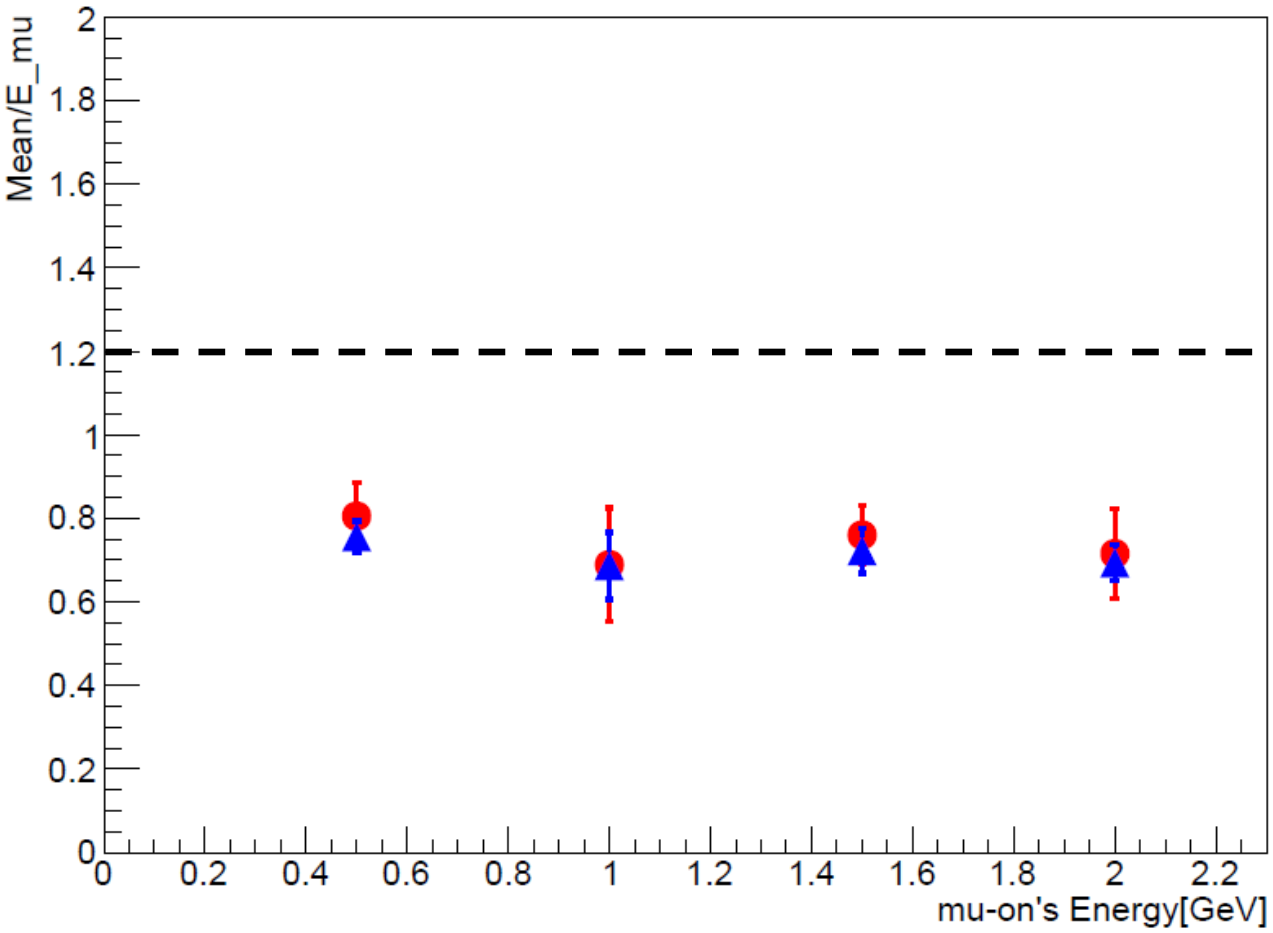

図20　赤が2分の1、青が3分の1での結果を表す。

入射エネルギー[GeV]	0.5	1.0	1.5	2.0
測定/入射エネルギー	0.81	0.69	0.76	0.72

表7　フィッティング範囲が2分の1の場合

入射エネルギー[GeV]	0.5	1.0	1.5	2.0
測定/入射エネルギー	0.76	0.79	0.75	0.71

表8　フィッティング範囲が3分の1の場合

また、入射エネルギーと測定エネルギーの比が一定になると仮定し、それぞれの結果の分散を求めた。範囲を2分の1とした結果の分散を VER2、3分の1とした結果の分散を VER3とおくと

$$\mathrm{VER2}\sim 4.9\times 10^{-4}$$
$$\mathrm{VER3}\sim 2.2\times 10^{-4}$$

となった。この事から、フィッティングの範囲をさらに狭くすることで、より正確な測定結果が得られることが予測できたので、フィッティングの範囲を10分の1とし、重み付き平均を求めた。

3.6.2 さらなる細分化の結果

一例として入射エネルギーが1.0GeV でのフィッティングの様子を図21に示す。

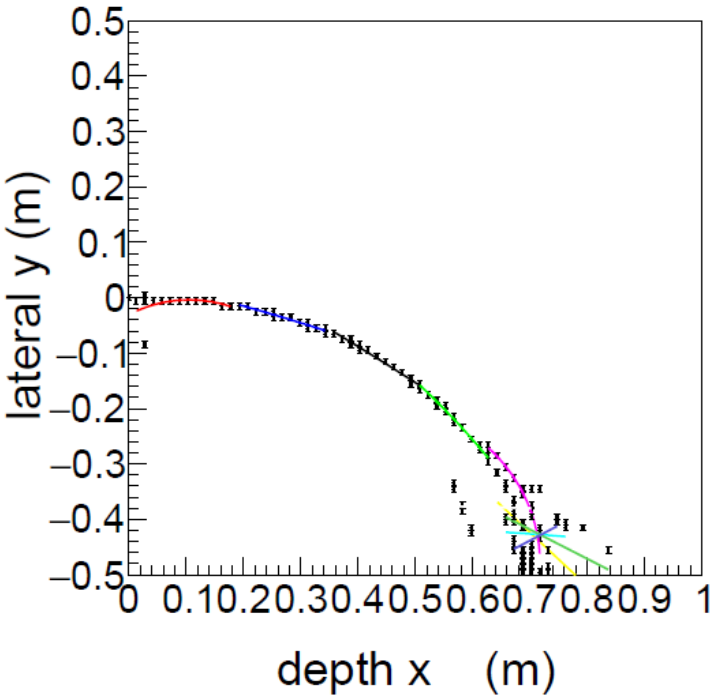

図21　10分の 1 の範囲でのフィッティング

フィッティングの範囲を飛跡全体の10分の 1 とすることにより1000粒子のデータが10パターン得られるので、それぞれの Mean 値からフィッティングによる$\frac{\chi^2}{NDF}$が10未満となるデータのみを要素として用いて、重み付き平均を取る。この際、正しく行われていないフィッティングから得られたデータは$\frac{\chi^2}{NDF} < 10$の条件付で省かれているものと考える。実際に用いた式を以下に示す。

$$\frac{\sum_{i=1}^{10} x_i w_i}{\sum_{i=1}^{10} w_i} = \bar{x}$$

ただしx_iは各データの Mean 値であり

$$w_i = \frac{1}{\Delta {x_i}^2}\ , \Delta x_i = \langle x \rangle - x_i\ , \langle x \rangle = \frac{\sum_{i=1}^{10} x_i}{10}$$

とした。求めた平均値を入射エネルギーで除したものに、標準誤差のエラーバーをつけたプロットを図22に、具体的な数値を表9に示す。

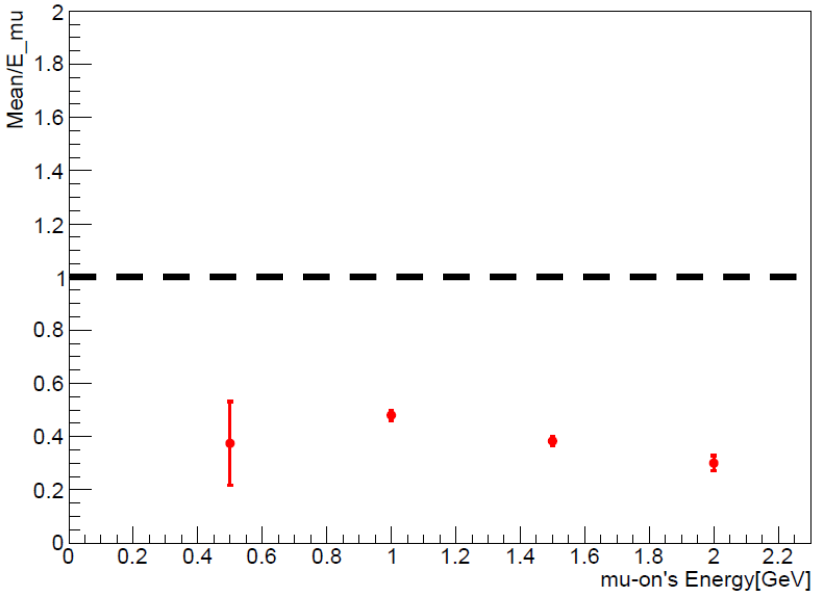

図22　フィッティング範囲が10分の1

入射エネルギー[GeV]	0.5	1.0	1.5	2.0
測定/入射エネルギー	0.37	0.48	0.38	0.31

表9　フィッティング範囲が10分の1

これらの分散を VER10として求めると

$$\mathrm{VER10}\sim 3.3\times 10^{-3}$$

となり、分散は大きくなった。これは図21に示されているように、エネルギーの落ちた点が粒子の通過位置以外にも存在し、フィッティングがうまく行われていないことに由来すると考えられる。

標準誤差の大きさについては、入射エネルギーが0.5GeV の点は、先述した通過位置から離れた点が非常に多く存在していることによるものと考えられる。図23に一例を示す。

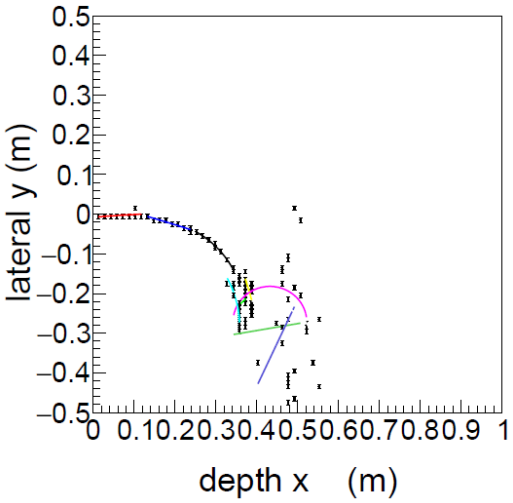

図23　10分の 1 の範囲でのフィッティング

それに対して1.0GeV 以上の点では、入射エネルギーの上昇に従って標準誤差も大きくなっているが、これは入射エネルギーが大きいほど HCAL 内の飛跡が長くなり、落とすエネルギーの量も大きくなるため、各部分での測定エネルギーに差が生じているものと考えられる。

4. まとめ

本研究では、Geant4用いてハドロンカロリメータを作成し、磁場内における安定な状態の荷電粒子が持つエネルギーを、飛跡の曲率から求めるシミュレーションを行った。

① カロリメータの形状を変えた場合の比較
吸収層の厚さと検出層のストリップ幅を10mm と20mm の間で変化させたが、結果に大きな違いは見られなかった。

② フィッティングの範囲を変更した場合の測定エネルギー
粒子の飛跡を等分し、それぞれの範囲でフィッティングを行い、導出したエネルギーの重み付き平均を求めた。2等分した場合に比べ3等分した場合は入射エネルギーごとの標準誤差も、全体の分散も小さかったため、測定エネルギーと入射エネルギーの比が一定に近づくという結果が得られることを期待して10等分した。しかし、フィッティングが正しく行われないなどの原因から、期待した結果は得られなかった。

以上の結果から、吸収層の厚さや検出層のストリップ幅を10mm、20mm の範囲で変更する場合、得られるデータに大きな変化は見られないことがわかった。
また、フィッティングの範囲を飛跡の部分ごとに分割し、得られた結果の重み付き平均を取ることで、比較的安定した測定データが得られることがわかった。よって HCAL 内粒子のエネルギーを求めたい場合、飛跡を細かく分割し、各部分で円の関数でフィッティングすることでエネルギーを求める方法が、測定方法の一つとして挙げられる。

5. 今後の展望

① 入射エネルギーパターンの増加

本研究では入射エネルギーごとの測定エネルギーの変化を観測するため、入射エネルギーを 0.5 , 1.0 , 1.5 , 2.0GeV で行った。その結果入射エネルギーと測定エネルギーの比がほぼ一定の値を取ったため、測定方法の改善により比は一定の値に近づくという仮定の元、研究を進めていった。そこで、実際に一定になるかどうかの検証として、入射エネルギーをより大きな範囲で、あるいはより細かく取ってのシミュレーションをすることが考えられる。

② フィッティングの範囲を狭めた状態でのカロリメータの変形

3.4節、及び3.5節での結果は吸収層の厚さや検出層のストリップ幅の10mm 程度の違いは測定結果に大きな影響を及ぼさないというものだった。これは言い換えれば10mm 程度の位置分解能の違いは全体の曲率に影響を及ぼさないということになる。実用段階において、必ずしも3.4節及び3.5節で行った程度に長いフィッティングをすることができるとは限らない。よって、吸収層の厚さと検出層のストリップ幅とともにフィッティングの長さも変化させることで、どの程度のフィッティング距離から、変形の影響が現れるのかを検証することができると考えられる。

③ 飛跡から離れた点の削除

本研究での飛跡の検出には、エネルギーが落ちた検出層ストリップの位置情報を用いているが、これは必ずしも飛跡のみをもたらすわけではない。フィッティングの最適化のために、粒子の通過位置を表すもの以外の点を削除したい。考えられる削除のポイントは2箇所ある。1箇所目はエネルギーが落ちたスプロットに位置を記録する段階、2箇所目はフィッティングを行う段階である。このどちらかに、何らかの基準により、粒子の通過位置を表す点以外の点を除くアルゴリズムを加えることで、正確に粒子の飛跡をなぞるフィッティングを行うことができると考えられる。

謝辞

本研究を行うに際して、竹下徹先生、長谷川庸司先生、研究員の小寺克茂様にご指導、ご教授を賜りました。定期報告会では現在の課題や、どのような最終目的が設定可能かなどを丁寧に解説していただけたため、今できる改善点やその後に向けてどのようなことができるようになっておくべきかなどを明確に認識することができました。竹下先生からいただいたご助言で、研究の方向性を明確にすることができました。長谷川先生には Geant4を使用するにあたってのプログラミングについてお力をお借りしました。また、小寺様には本研究のモチベーションとなる ILC について、またデータ解析のためのプラグラミングについて何度も質問に伺いましたが、その都度わかりやすく教えていただきました。そして、解析手法について同学科の北山佳治さんにご助言いただきました。

お世話になった方々に、この場を借りまして厚く御礼申し上げます。
最後に、大学で勉強と貴重な経験をさせてくださった両親に心より感謝申し上げます。

誠に有難うございました。

参考文献

[1] Ashok Das , Thomas Ferbel 著　末包文彦 , 白井淳平 , 湯田春雄 訳
『素粒子･原子核物理の基礎 ～実験から統一理論まで～』 (共立出版　2011)

[2] 伊佐見将 『Geant4を用いた ECAL シミュレーション』 (2015)

[3] 中尾知博 『Geant4によるカロリメータの性能評価』 (2013)

[4] Ties Behnke et al 『The International Linear Collider Technical Design Reports―Volume 4: Detectors』 (2013)

補遺

本研究で使用したプログラムのソースコードを載せる。

B4DetedtorConstruction.cc

このプログラムを用いて、カロリメータの形状を設定した。

＝＝＝

```
#include "B4DetectorConstruction.hh"

#include "G4Material.hh"
#include "G4NistManager.hh"

#include "G4Box.hh"
#include "G4LogicalVolume.hh"
#include "G4PVPlacement.hh"
#include "G4PVReplica.hh"
#include "G4GlobalMagFieldMessenger.hh"
#include "G4AutoDelete.hh"

#include "G4FieldManager.hh"
#include "G4TransportationManager.hh"
#include "G4ChordFinder.hh"

#include "G4GeometryManager.hh"
#include "G4PhysicalVolumeStore.hh"
#include "G4LogicalVolumeStore.hh"
#include "G4SolidStore.hh"

#include "G4VisAttributes.hh"
#include "G4Colour.hh"

#include "G4PhysicalConstants.hh"
#include "G4SystemOfUnits.hh"

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

G4ThreadLocal
G4GlobalMagFieldMessenger* B4DetectorConstruction::fMagFieldMessenger = 0;

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

B4DetectorConstruction::B4DetectorConstruction()
 : G4VUserDetectorConstruction(),
    fAbsorberPV(0),
    fGapPV(0),
    fCheckOverlaps(true)
{
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

B4DetectorConstruction::~B4DetectorConstruction()
{
}
```

```
//....oooOO0OOooo.......oooOO0OOooo.......oooOO0OOooo.......oooOO0OOooo......

G4VPhysicalVolume* B4DetectorConstruction::Construct()
{
   // Define materials
   DefineMaterials();

   // Define volumes
   return DefineVolumes();
}

//....oooOO0OOooo.......oooOO0OOooo.......oooOO0OOooo.......oooOO0OOooo......

void B4DetectorConstruction::DefineMaterials()
{
   // Lead material defined using NIST Manager
   G4NistManager* nistManager = G4NistManager::Instance();
   nistManager->FindOrBuildMaterial("G4_Fe");
   nistManager->FindOrBuildMaterial("G4_PLASTIC_SC_VINYLTOLUENE");
   nistManager->FindOrBuildMaterial("G4_AIR");

   // Liquid argon material
   G4double a;    // mass of a mole;
   G4double z;    // z=mean number of protons;
   G4double density;
   new G4Material("liquidArgon", z=18., a= 39.95*g/mole, density= 1.390*g/cm3);
            // The argon by NIST Manager is a gas with a different density

   // Vacuum
   new G4Material("Galactic", z=1., a=1.01*g/mole,density= universe_mean_density,
                           kStateGas, 2.73*kelvin, 3.e-18*pascal);

   // Print materials
   G4cout << *(G4Material::GetMaterialTable()) << G4endl;
}

//....oooOO0OOooo.......oooOO0OOooo.......oooOO0OOooo.......oooOO0OOooo......

G4VPhysicalVolume* B4DetectorConstruction::DefineVolumes()
{
   // Geometry parameters    //吸収層の厚さを設定
   G4int nofLayers = 100;
   G4double absoThickness = 20.*mm;
   G4double gapThickness =    5.*mm;
   G4double calorSizeXY    = 100.*cm;

   G4double layerThickness = absoThickness + gapThickness;
   G4double calorThickness = nofLayers * layerThickness;
   G4double worldSizeXY = 1. * calorSizeXY;
   G4double worldSizeZ    = 1. * calorThickness;

   // Get materials
   G4Material* defaultMaterial = G4Material::GetMaterial("Galactic");
   G4Material* absorberMaterial = G4Material::GetMaterial("G4_Fe");
   G4Material* gapMaterial = G4Material::GetMaterial("G4_PLASTIC_SC_VINYLTOLUENE");

   if ( ! defaultMaterial || ! absorberMaterial || ! gapMaterial ) {
      G4ExceptionDescription msg;
```

```
    msg << "Cannot retrieve materials already defined.";
    G4Exception("B4DetectorConstruction::DefineVolumes()",
      "MyCode0001", FatalException, msg);
}

//
// World
//
G4VSolid* worldS
   = new G4Box("World",                    // its name
                worldSizeXY/2, worldSizeXY/2, worldSizeZ/2); // its size

G4LogicalVolume* worldLV
   = new G4LogicalVolume(
                worldS,                    // its solid
                defaultMaterial,    // its material
                "World");                  // its name

G4VPhysicalVolume* worldPV
   = new G4PVPlacement(
                0,                         // no rotation
                G4ThreeVector(),    // at (0,0,0)
                worldLV,                   // its logical volume
                "World",                   // its name
                0,                         // its mother  volume
                false,                     // no boolean operation
                0,                         // copy number
                fCheckOverlaps);    // checking overlaps

//
// Calorimeter
//
G4VSolid* calorimeterS
   = new G4Box("Calorimeter",          // its name
                calorSizeXY/2, calorSizeXY/2, calorThickness/2); // its size

G4LogicalVolume* calorLV
   = new G4LogicalVolume(
                calorimeterS,          // its solid
                defaultMaterial,    // its material
                "Calorimeter");      // its name

new G4PVPlacement(
                0,                         // no rotation
                G4ThreeVector(0,0,0),    // at (0,0,0)
                calorLV,                   // its logical volume
                "Calorimeter",        // its name
                worldLV,                   // its mother  volume
                false,                     // no boolean operation
                0,                         // copy number
                fCheckOverlaps);    // checking overlaps

//
// Layer
//
G4VSolid* layerS
   = new G4Box("Layer",                    // its name
                calorSizeXY/2, calorSizeXY/2, layerThickness/2); // its size
```

```
  G4LogicalVolume* layerLV
    = new G4LogicalVolume(
                    layerS,              // its solid
                    defaultMaterial,    // its material
                    "Layer");            // its name

          //
          // Absorber
          //
          G4VSolid* absorberS
            = new G4Box("Abso",                    // its name
                        calorSizeXY/2, calorSizeXY/2, absoThickness/2); // its size

          G4LogicalVolume* absorberLV
            = new G4LogicalVolume(
                        absorberS,           // its solid
                        absorberMaterial, // its material
                        "Abso");             // its name

          fAbsorberPV
            = new G4PVPlacement(
                        0,                     // no rotation
                        G4ThreeVector(0., 0., -gapThickness/2), // its position
                        absorberLV,          // its logical volume
                        "Abso",              // its name
                        layerLV,             // its mother  volume
                        false,               // no boolean operation
                        0,                     // copy number
                        fCheckOverlaps);    // checking overlaps

          //
          // Gap
          //

        G4int ndivx = 1.;
        G4int ndivy = 50.;   //  検出層の分割幅を設定

          G4VSolid* gapS
            = new G4Box("Gap",                     // its name
                        calorSizeXY/ndivx/2, calorSizeXY/ndivy/2, gapThickness/2); // its size

          G4LogicalVolume* gapLV
            = new G4LogicalVolume(
                        gapS,                  // its solid
                        gapMaterial,         // its material
                        "Gap");              // its name

        for (G4int xi = 0 ; xi < ndivx ; ++xi ) {
                for (G4int yi = 0 ; yi < ndivy ; ++yi ) {
          fGapPV
            = new G4PVPlacement(
                        0,                     // no rotation
                        G4ThreeVector(-calorSizeXY/2+(0.5+xi)*calorSizeXY/ndivx,
-calorSizeXY/2+(0.5          +yi)*calorSizeXY/ndivy, absoThickness/2), // its position
                        gapLV,               // its logical volume
                        "Gap",               // its name
                        layerLV,             // its mother  volume
                        false,               // no boolean operation
                        ndivy*xi+yi,         // copy number
```

```
                                    fCheckOverlaps);    // checking overlaps
                        }
                }

// calorThickness/layerThickness

for (G4int zi = 0 ; zi < calorThickness/layerThickness ; ++zi) {
   flayerPV
      = new G4PVPlacement(
                        0,                          // no rotation
                        G4ThreeVector(0.,0.,-calorThickness/2+(zi+0.5)*layerThickness), // its position
                        layerLV,            // its logical volume
                        "Layer",                    // its name
                        calorLV,                  // its mother    volume
                        false,                     // no boolean operation
                        zi,                         // copy number
                        fCheckOverlaps);    // checking overlaps

}


   //
   // print parameters
   //
   G4cout
      << G4endl
      << "------------------------------------------------------------" << G4endl
      << "---> The calorimeter is " << nofLayers << " layers of: [ "
      << absoThickness/mm << "mm of " << absorberMaterial->GetName()
      << " + "
      << gapThickness/mm << "mm of " << gapMaterial->GetName() << " ] " << G4endl
      << "------------------------------------------------------------" << G4endl;

   //
   // Visualization attributes
   //
   worldLV->SetVisAttributes (G4VisAttributes::Invisible);

   G4VisAttributes* simpleBoxVisAtt= new G4VisAttributes(G4Colour(1.0,1.0,1.0));
   simpleBoxVisAtt->SetVisibility(true);
   calorLV->SetVisAttributes(simpleBoxVisAtt);

   //
   // Always return the physical World
   //
   return worldPV;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void B4DetectorConstruction::ConstructSDandField()
{
   // Create global magnetic field messenger.
   // Uniform magnetic field is then created automatically if
   // the field value is not zero.
   G4ThreeVector fieldValue = G4ThreeVector();
   fMagFieldMessenger = new G4GlobalMagFieldMessenger(fieldValue);
   fMagFieldMessenger->SetVerboseLevel(1);
```

```
   // Register the field messenger for deleting
   G4AutoDelete::Register(fMagFieldMessenger);
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
```

＝＝

B4EventAction.cc

このプログラムで、入射ごとの粒子の通過位置を記録した。

＝＝

```
#include "B4aEventAction.hh"
#include "B4RunAction.hh"
#include "B4Analysis.hh"

#include "G4RunManager.hh"
#include "G4Event.hh"
#include "G4UnitsTable.hh"

#include "Randomize.hh"
#include <iomanip>

#include <fstream>
#include <sstream>
#include <iostream>

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

B4aEventAction::B4aEventAction()
 : G4UserEventAction(),
    fEnergyAbs(0.),
    fEnergyGap(0.),
    fTrackLAbs(0.),
    fTrackLGap(0.)
{}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

B4aEventAction::~B4aEventAction()
{}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void B4aEventAction::BeginOfEventAction(const G4Event* /*event*/)
{
   // initialisation per event
   fEveNumber = 0.;
   fEnergyAbs = 0.;
   fEnergyGap = 0.;
   fTrackLAbs = 0.;
   fTrackLGap = 0.;
   for (G4int i=0; i<10000; ++i) {
      fEnergyAbsbyLyr[i] = 0.;
      fEnergyGapbyLyr[i] = 0.;
      fTileNumber[i] = 0.;
      fLayerNumber[i] = 0.;
```

```
    }
}

//....oooOO0OOooo.......oooOO0OOooo.......oooOO0OOooo.......oooOO0OOooo.....

void B4aEventAction::EndOfEventAction(const G4Event* event)
{
   // Accumulate statistics
   //

   // get analysis manager
   G4AnalysisManager* analysisManager = G4AnalysisManager::Instance();

   // fill histograms
   analysisManager->FillH1(1, fEnergyAbs);
   analysisManager->FillH1(2, fEnergyGap);
   analysisManager->FillH1(3, fTrackLAbs);
   analysisManager->FillH1(4, fTrackLGap);

   // fill ntuple
   fEveNumber++;
   analysisManager->FillNtupleDColumn(0, fEveNumber);
   analysisManager->FillNtupleDColumn(1, fEnergyAbs);
   analysisManager->FillNtupleDColumn(2, fEnergyGap);
   analysisManager->FillNtupleDColumn(3, fTrackLAbs);
   analysisManager->FillNtupleDColumn(4, fTrackLGap);
   for (G4int i = 0; i<1000; ++i) {
      analysisManager->FillNtupleDColumn(i*4    +5, fEnergyAbsbyLyr[i]);
      analysisManager->FillNtupleDColumn(i*4+1+5, fEnergyGapbyLyr[i]);
      if (fEnergyGapbyLyr[i] != 0) {
         analysisManager->FillNtupleDColumn(i*4+2+5, fTileNumber[i]);
         analysisManager->FillNtupleDColumn(i*4+3+5, fLayerNumber[i]);
      }
   }
   analysisManager->AddNtupleRow();

   //      |std::ios::app : << event->GetEventID()

   std::cout << "test" << event->GetEventID();

   std::stringstream ss;

// 粒子の通過位置をメートル単位で記録

if ( event->GetEventID() < 10 ) {
{
   ss.str("");
   ss << "/home/iino/pfa/trackdata/abs20mm/gap20mm/Plot_mu-_2.0_"    << "00" << event->GetEventID() <<
".txt";
   std::ofstream ofs(ss.str().c_str(),std::ios::app);
      for (G4int i = 0; i<10000; ++i) {
            if (fEnergyGapbyLyr[i] != 0) {
   ofs << 0.02+0.005/2+i/100*0.025 << "             " << 0.005+i%100*0.02 << std::endl;
         }
      }
   ofs.close();
}
}
```

```
  else if ( event->GetEventID() < 100 ) {
{
   ss.str("");
   ss << "/home/iino/pfa/trackdata/abs20mm/gap20mm/Plot_mu-_2.0_"    << "0" << event->GetEventID() <<
".txt";
   std::ofstream ofs(ss.str().c_str(),std::ios::app);
      for (G4int i = 0; i<10000; ++i) {
            if (fEnergyGapbyLyr[i] != 0) {
   ofs << 0.02+0.005/2+i/100*0.025 << "            " << 0.005+i%100*0.02 << std::endl;
         }
      }
   ofs.close();
}
}

  else    if( event->GetEventID() < 1000 ) {
{
   ss.str("");
   ss << "/home/iino/pfa/trackdata/abs20mm/gap20mm/Plot_mu-_2.0_"    << event->GetEventID() << ".txt";
   std::ofstream ofs(ss.str().c_str(),std::ios::app);
      for (G4int i = 0; i<10000; ++i) {
            if (fEnergyGapbyLyr[i] != 0) {
   ofs << 0.02+0.005/2+i/100*0.025 << "            " << 0.005+i%100*0.02 << std::endl;
         }
      }
   ofs.close();
}
}


  //

   // Print per event (modulo n)
   //
   G4int eventID = event->GetEventID();
   G4int printModulo = G4RunManager::GetRunManager()->GetPrintProgress();
   if ( ( printModulo > 0 ) && ( eventID % printModulo == 0 ) ) {
      G4cout << "---> End of event: " << eventID << G4endl;

      G4cout
         << "      Absorber: total energy: " << std::setw(7)
                                             << G4BestUnit(fEnergyAbs,"Energy")
         << "            total track length: " << std::setw(7)
                                             << G4BestUnit(fTrackLAbs,"Length")
         << G4endl
         << "              Gap: total energy: " << std::setw(7)
                                             << G4BestUnit(fEnergyGap,"Energy")
         << "            total track length: " << std::setw(7)
                                             << G4BestUnit(fTrackLGap,"Length")
         << G4endl;
   }
}

//....oooOO0OOooo.....oooOO0OOooo.....oooOO0OOooo.....oooOO0OOooo.....

==========================================================================
```

calucMomentum.C

B4EventAction.cc で作成した粒子の通過位置情報をもとにプロットグラフを作り、それに合わせて円の関数でフィッティングを行い、エネルギーを求める。

＝＝

```
calucMomentum(){

        Double_t depth[400];
        Double_t depthErr[400];
        Double_t lateralY[400];
        Double_t lateralYErr[400];

        Double_t width = 0.020;
        Double_t thick = 0.005;

        Double_t FieldStrength = 3.5;

        Double_t xlo = 0;
        Double_t ylo = -0.5;
        Double_t xhi = 1.0;
        Double_t yhi = 0.5;

        char outData[256] = "../results.dat",
                outData2[256] =
"../histdata/abs10mm/gap10mm/momentumHist_abs10mm_gap10mm_10GeV.dat",
                outData3[256] =
"../histdata/abs10mm/gap10mm/reducedChi_1Divide_abs10mm_gap10mm_10GeV.dat",
                outData4[256] =
"../histdata/abs10mm/gap10mm/NDF_Divide_abs10mm_gap10mm_10GeV.dat",
                outData5[256] =
"../histdata/abs10mm/gap10mm/momentumHist_reducedChi_abs10mm_gap10mm_10GeV.dat";

        ofstream fresults;
        fresults.open(outData);

        ofstream fmomentumHist_abs10mm_gap10mm_10GeV;
        fmomentumHist_abs10mm_gap10mm_10GeV.open(outData2);

        ofstream freducedChi_1Divide_abs10mm_gap10mm_10GeV;
        freducedChi_1Divide_abs10mm_gap10mm_10GeV.open(outData3);

        ofstream fNDF_Divide_abs10mm_gap10mm_10GeV;
        fNDF_Divide_abs10mm_gap10mm_10GeV.open(outData4);

        ofstream fmomentumHist_reducedChi_abs10mm_gap10mm_10GeV;
        fmomentumHist_reducedChi_abs10mm_gap10mm_10GeV.open(outData5);

        TGraphErrors * graph[1000];
        TF1 * func[1000];
        TCanvas * c1 = new TCanvas("c1","",800,800);
        TH1F * frame[1000];
        c1->Divide(1,1);

   int maxevent = 1000;
        for ( int i = 0; i < maxevent; i ++ ) {
```

```
                char funcname[256];
                sprintf(funcname, "func_%d", i );

                char datafile[256];
                if ( i < 10 ) {
                        sprintf (datafile, "../trackdata/abs10mm/gap10mm/Plot_mu-_1.0_00%d.txt", i );
                } else if ( i <100 ) {
                        sprintf (datafile, "../trackdata/abs10mm/gap10mm/Plot_mu-_1.0_0%d.txt", i );
                } else {
                        sprintf (datafile, "../trackdata/abs10mm/gap10mm/Plot_mu-_1.0_%d.txt", i );
                }

        std::cout << " file name = " << datafile << std::endl;
        ifstream data( datafile );

                        int index = 0;
        while (!data.eof()) {
           data >> depth[index] >> lateralY[index];
                                index ++;
        }
        data.close();
                        Double_t depthMax = 0;
                        for ( Int_t j = 0; j < index-1 ; j++ ) {
                                if ( depthMax < depth[j] ) {
                                        depthMax = depth[j];
                                }
                                lateralY[j] = lateralY[j] - 0.5;
                                depthErr[j] = thick/(2*sqrt(3));
                                lateralYErr[j] = width/(2*sqrt(3));
                        }

// プロットグラフを作り、フィッティングを行う
                func[i] = new TF1(funcname,circle, 0.0, depthMax*1.0, 3);
                c1->cd(i+1);
           frame[i] = gPad->DrawFrame(xlo, ylo, xhi, yhi);
      gPad->SetLeftMargin(0.25);
      gPad->SetBottomMargin(0.25);

                frame[i]->GetXaxis()->SetTitleOffset(1.2);
                frame[i]->GetYaxis()->SetTitleOffset(1.2);
                frame[i]->GetXaxis()->SetTitleSize(0.06);
                frame[i]->GetYaxis()->SetTitleSize(0.06);
                frame[i]->GetXaxis()->SetLabelSize(0.05);
                frame[i]->GetYaxis()->SetLabelSize(0.05);
                frame[i]->GetXaxis()->CenterTitle();
                frame[i]->GetYaxis()->CenterTitle();

          frame[i]->GetXaxis()->SetTitle( "depth      x (m)" );
          frame[i]->GetYaxis()->SetTitle( "lateral y (m)" );



                        graph[i] =    new TGraphErrors( index-1, depth, lateralY, depthErr, lateralYErr);
                        graph[i]->Draw("p");
                        func[i]->SetParameter(0, 1.0);
                        func[i]->SetParameter(1, 0.5);
                        func[i]->SetParameter(2, -1.5);
                        graph[i]->Fit(funcname,"R");
```

```
                                Double_t chisq = func[i]->GetChisquare();
                                Double_t ndf      = func[i]->GetNDF();
                                Double_t reducedChi = chisq/ndf;

        c1->Print("../figs/1Divide/abs10mm/gap10mm/muonTrack_1divide_gap10mm_abs10mm_1.0GeV.pdf")
;

                                // エネルギーを求める
                                Double_t *parm =    func[i]->GetParameters();
                                Double_t *parmErr =    func[i]->GetParErrors();
                                Double_t momentum = 0.3* FieldStrength * parm[0];
                                Double_t momentumErr = 0.3* FieldStrength * parmErr[0];
                                Double_t originX = parm[1];
                                Double_t originY = parm[2];
                                std::cout << momentum << " "
                                          << momentumErr << " "
                                          << originX << " "
                                          << originY
                                          << std::endl;

                                fresults << momentum << " "
                                         << momentumErr << " "
                                         << originX << " "
                                         << originY
                                         << std::endl;

                                fmomentumHist_abs10mm_gap10mm_10GeV << momentum << std::endl;

                                freducedChi_1Divide_abs10mm_gap10mm_10GeV << reducedChi << std::endl;

                                fNDF_Divide_abs10mm_gap10mm_10GeV << ndf << std::endl;

                                fmomentumHist_reducedChi_abs10mm_gap10mm_10GeV << momentum << " "
<< reducedChi << std::endl;

        }


}

////////////////// Definition functions ////////////////////////

Double_t circle( Double_t * x, Double_t *par)
{
   Float_t xx = x[0];
   Double_t f = std::sqrt( par[0]*par[0] - (xx-par[1])* (xx-par[1])) + par[2]; //  フィッティング関数を指定
   return f;
}

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝
```

Hist.C

calucMomentum.C で求めたエネルギーを用いてヒストグラムを作成する。

＝＝

```
{
TCanvas* c1;
c1=new TCanvas("Energy_gap10mm_abs10mm","Mometum_gap10mm_abs10mm",10,10,800,600);

TH1S h1("0.5GeV","Energy_gap10mm_abs10mm",50,0.0,2.0);
TH1S h2("1.0GeV","Energy_gap10mm_abs10mm",50,0.0,2.0);
TH1S h3("1.5GeV","Energy_gap10mm_abs10mm",50,0.0,2.0);
TH1S h4("2.0GeV","Energy_gap10mm_abs10mm",50,0.0,2.0);

#include <fstream.h>

ifstream data1("../histdata/abs10mm/gap10mm/momentumHist_abs10mm_gap10mm_05GeV.dat");
ifstream data2("../histdata/abs10mm/gap10mm/momentumHist_abs10mm_gap10mm_10GeV.dat");
ifstream data3("../histdata/abs10mm/gap10mm/momentumHist_abs10mm_gap10mm_15GeV.dat");
ifstream data4("../histdata/abs10mm/gap10mm/momentumHist_abs10mm_gap10mm_20GeV.dat");

double x1,x2,x3,x4;

int index =0;
while (data1 >> x1 ) h1.Fill(x1);
while (data2 >> x2 ) h2.Fill(x2);
while (data3 >> x3 ) h3.Fill(x3);
while (data4 >> x4 ) h4.Fill(x4);

data1.close();
data2.close();
data3.close();
data4.close();
//data5.close();

TH1F *frame = gPad->DrawFrame(0.,0.,6,80);
frame->Draw("sames");
frame->SetTitle("Energy_gap10mm_abs10mm_1.0GeV");
frame->GetXaxis()->SetTitle("mu-on's Energy[GeV]");

h2->Draw();

h1->Draw("sames");

h3->Draw("sames");

h4->Draw("sames");

c1->Update();

TPaveStats *st1 = (TPaveStats*)h1->FindObject("stats");
h1->SetStats(1);
h1->GetXaxis()->SetTitle("Fitting Energy[GeV]");
h1->SetLineColor(2);
st1->SetTextColor(2);
h1->SetLineWidth(2);

st1->SetX1NDC(0.78);
```

```
st1->SetX2NDC(0.98);
st1->SetY1NDC(0.775);
st1->SetY2NDC(0.935);

TPaveStats *st2 = (TPaveStats*)h2->FindObject("stats");
h2->SetStats(1);
h2->GetXaxis()->SetTitle("Fitting Energy[GeV]");
h2->SetLineColor(4);
st2->SetTextColor(4);
h2->SetLineWidth(2);

st2->SetX1NDC(0.78);
st2->SetX2NDC(0.98);
st2->SetY1NDC(0.605);
st2->SetY2NDC(0.765);

TPaveStats *st3 = (TPaveStats*)h3->FindObject("stats");
h3->SetStats(1);
h3->GetXaxis()->SetTitle("Fitting Energy[GeV]");
h3->SetLineColor(1);
st3->SetTextColor(1);
h3->SetLineWidth(2);

st3->SetX1NDC(0.78);
st3->SetX2NDC(0.98);
st3->SetY1NDC(0.435);
st3->SetY2NDC(0.595);

TPaveStats *st4 = (TPaveStats*)h4->FindObject("stats");
h4->SetStats(1);
h4->GetXaxis()->SetTitle("Fitting Energy[GeV]");
h4->SetLineColor(3);
st4->SetTextColor(3);
h4->SetLineWidth(2);

st4->SetX1NDC(0.78);
st4->SetX2NDC(0.98);
st4->SetY1NDC(0.265);
st4->SetY2NDC(0.425);
```

==========