Jupiter による
カロリメータのシミュレーション

信州大学理学部物理科学科
高エネルギー物理学研究室
04S2027B
牧野正樹

平成 20 年 3 月 23 日

概要

高エネルギー物理学において、シミュレーションの方法がある。実際の測定に用いる測定器には費用がかかる。シミュレーションを行い、性能を評価することが必要になる。測定器はいくつかの部分から構成されているが、エネルギーを測定するカロリメータがある。本研究では、Jupiter とよばれるシミュレーションソフトにより、カロリメータに関するシミュレーションを行った。

前半では、エネルギーの測定精度を表すエネルギー分解能と、横方向のエネルギーの拡がりを表すモリエール半径を評価した。エネルギー分解能では、入射エネルギーとの関係を見た。光子の場合、入射エネルギー E に対して、エネルギー分解能は $(0.156 \pm 0.002)/\sqrt{E[\mathrm{GeV}]} \oplus (0.000 \pm 0.006)\%$ となった。

後半では、π^0 の崩壊について考え、崩壊してできた 2 つの光子から、π^0 の不変質量と、崩壊でできた光子の角度分解能についてを調べた。π^0 が崩壊してできた 2 つの光子のエネルギーと開き角から、不変質量が計算できる。入射エネルギーが 1GeV から 5GeV の π^0 の不変質量の平均および誤差は、$M = 134.4 \pm 0.3\mathrm{MeV}$ となった。角度分解能は、光子の入射エネルギーとの関係を見た。光子の入射エネルギー E に対して、角度分解能 $\sigma = (0.122 \pm 0.001)/\sqrt{E[\mathrm{GeV}]} \oplus (0.022 \pm 0.001)\mathrm{rad}$ となった。

目 次

1 序章

1.1 目的

本研究の動機は、シミュレーションにより物理現象の考察をすることである。本研究の目的は、次の通りである。

- 精度を表すエネルギー分解能、角度分解能を求め、カロリメータの性能を調べる。
- モリエール半径を計算し、エネルギーの拡がりについて調べる。
- π^0 から崩壊してできた 2 つの光子のエネルギーと開き角から、π^0 の不変質量を求め、正しい値が得られるかどうかを確認する。

2 Jupiter および理論

2.1 Jupiter

2.1.1 Jupiter

Jupiter とは、“JLC unified particle interaction and tracking emulator” の略である。粒子と物質の相互作用をシミュレーションする Geant4 をベースとして作られた。ILC 加速器を用いた実験における測定器のシミュレータである。

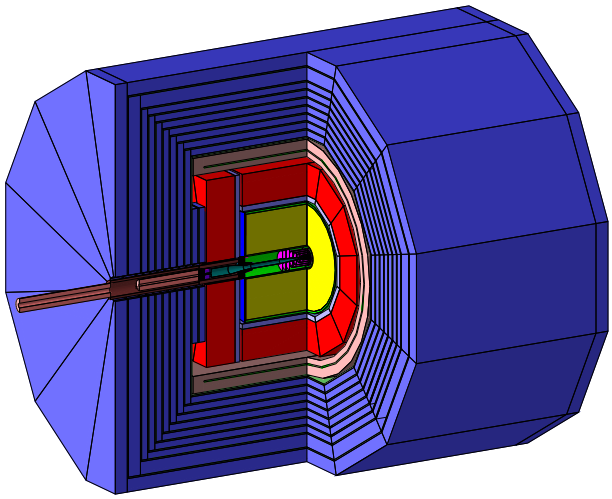

図 1: 測定器

測定器を図 1 に示す。構成は、内側から順に次のようになっている。

- Vertex Detector
  電子・陽電子の衝突点の極めて近傍に設置され、荷電粒子の飛跡を高い精度で測定する。CCD(電荷結合素子) のような 2 次元分解能の高いモジュールが想定されている。要求される位置分解能は、$\sigma = 4.0\mu$m である。

- Intermediate Tracker
  Time Projection Chamber と併せてより多くの荷電粒子の通過位置を測定し運動量分解能を向上させる。シリコンストリップ型の検出器である。位置分解能は、$\sigma = 4.0\mu$m である。

- Time Projection Chamber
  荷電粒子の通過位置を測定し、運動量を測定する。ビーム軸に平行な電場により、電子を端部方向にドリフトさせて 3 次元飛跡を測定する。曲率

から荷電粒子の運動量を測定する。位置分解能は、z 方向で $\sigma_z = 1.0$mm である。

- Calorimeter
  粒子を物質中で止め、吸収されたエネルギーの一部を測定することによって、粒子のエネルギーを測定する。電磁シャワーから出る光量を MPPC(Multi-Pixel Photon Counter) という光検出器で測定し、それからエネルギーに変換する。以下のエネルギー分解能が要求される。

$$\text{電磁カロリメータ} \quad \frac{\sigma}{E} = \frac{15}{\sqrt{E[\text{GeV}]}} \oplus 1\%$$
$$\text{ハドロンカロリメータ} \quad \frac{\sigma}{E} = \frac{40}{\sqrt{E[\text{GeV}]}} \oplus 2\%$$

- Solenoid Magnet
  荷電粒子の運動量を測定するために、飛跡を曲げる磁場をかける。磁場の大きさは 3T までである。

- Muon Detector
  μ 粒子の同定を行い、運動量を測定する。Muon Detector のシグナルと飛跡検出器の情報をつなげることで、ミューオンの識別を行う。位置分解能は、$\sigma = 0.5$mm である。

2.1.2 カロリメータ

エネルギーを測定する部分がカロリメータである。カロリメータには CLX と CAL の 2 種類ある。CAL は衝突点方向を向くタワーによって構成されている。ここでは使用した CLX について述べる。

CLX を図 2 に示す。CLX は 12 角形をしており、バレル部とエンドキャップ部からなっている。中心から 210cm から 230cm の部分に電磁カロリメータ (ECAL)、230cm から 350cm の部分にハドロンカロリメータ (HCAL) があり、吸収層と検出層が交互に重なっている。

ECAL は、タングステン 3mm、シンチレータ 2mm、空気 1mm が 33 層になっており、タングステンの部分が吸収層、シンチレータが検出層になっている。HCAL は、鉄 20mm、シンチレータ 5mm、空気 1mm が 46 層になっており、鉄の部分が吸収層、シンチレータが検出層になっている。検出層は 1cm × 1cm のタイル型になっている。エネルギーの多くは吸収層の部分に吸収され、一部をシンチレータで検出する。

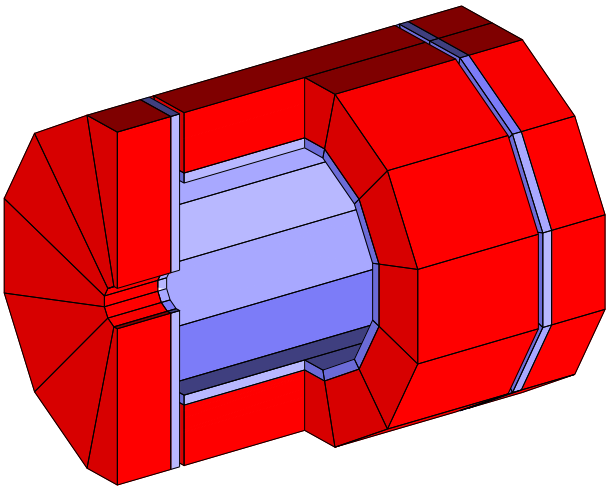

図 2: カロリメータ CLX

2.2 電子・光子の拡がり

2.2.1 電磁カスケードシャワー

高エネルギーの光子は、物質中で原子核の電磁場と相互作用をして、電子陽電子対に変換される (電子・陽電子対生成)。これによって生じた電子、陽電子は、原子核の作る強い電場によって加速度を受け、光子を放出してエネルギーを失う (制動放射)。これを繰り返すと、最初の光子に比べてエネルギーの低い電子、陽電子および光子がシャワー状に多数発生する。これを電磁カスケードシャワーという。入射粒子が電子の場合にも同様のプロセスで電磁カスケードシャワーが形成される。

電磁シャワーの粒子の数は、電子および陽電子のエネルギーが臨界エネルギー E_c になるまで増え続ける。陽子数を Z に対して、E_c は

$$E_c = \frac{800}{Z + 1.2}[\mathrm{MeV}] \tag{1}$$

で与えられる [1]。

縦方向の電磁シャワーを記述する単位として放射長 X_0 がある。X_0 は、物質中で電子のエネルギーが平均して $1/e$ に減少する長さであり、

$$X_0 = \frac{716.4A}{Z(Z+1)\ln(287/\sqrt{Z})}[\mathrm{g/cm^2}] \tag{2}$$

で与えられる [1]。A は質量数である。

2.2.2 モリエール半径

電磁シャワーはクーロン散乱によって横方向にも拡がる。横方向のシャワーの拡がりは、モリエール半径 R_{M} で記述される。R_{M} は

$$R_M = \frac{E_s}{E_c} \cdot X_0[\mathrm{g/cm^2}] \tag{3}$$

となる [1]。E_s は特性エネルギーとよばれ、

$$E_s = m_e c^2 \sqrt{\frac{4\pi}{\alpha}} = 21.2\mathrm{MeV} \tag{4}$$

である。

半径 R_{M} 内に、シャワーの全エネルギーの 90% が収まる。

2.3 π^0 の崩壊

2.3.1 π^0

π^0 は、質量 $134.98\mathrm{MeV}/c^2$ である。寿命 8.4×10^{-17}秒 で、電磁相互作用により 2 つの光子に崩壊する [2]。

$$\pi^0 \to 2\gamma \tag{5}$$

2.3.2 不変質量

π^0 の不変質量 M について、$c = 1$ とする単位系で考える。

崩壊する前の 4 元運動量を p、崩壊した一方の光子のエネルギーを E_1、運動量を $\boldsymbol{p_1}$、4 元運動量を p_1、もう一方の光子のエネルギーを E_2、運動量を $\boldsymbol{p_2}$、4 元運動量を p_2 とする。4 元運動量の保存より、

$$p = p_1 + p_2 \tag{6}$$

式 (6) の 2 乗は

$$p^2 = (p_1 + p_2)^2 = M^2 \tag{7}$$

4 元運動量は

$$p_1 = (E_1,\ \boldsymbol{p_1}) \tag{8}$$

$$p_2 = (E_2,\ \boldsymbol{p_2}) \tag{9}$$

であるから、式 (7) は

$$\begin{aligned} M^2 &= (E_1 + E_2,\ \boldsymbol{p_1} + \boldsymbol{p_2})^2 \\ &= (E_1 + E_2)^2 - (\boldsymbol{p_1} + \boldsymbol{p_2})^2 \end{aligned} \tag{10}$$

光子の質量は 0 であるから、$|\boldsymbol{p_1}| = E_1,\ |\boldsymbol{p_2}| = E_2$ とする。$\boldsymbol{p_1}$ と $\boldsymbol{p_2}$ のなす角を θ とすると、

$$M^2 = 2E_1E_2(1 - \cos\theta) \tag{11}$$

となる。

3　解析と結果

3.1　データ

入射粒子、方向、エネルギー、イベント数を決めて実行する。入射方向は、ビーム軸を z 軸、それに垂直に x 軸, y 軸をとり、曲座標での θ, ϕ を決める。得られるデータは、カロリメータのシンチレータの部分のセルの中心の座標と、そのセルで測定されたエネルギーの値である。

3.2　電子および光子のエネルギーの分布

3.2.1　エネルギー分解能

入射方向を $\theta = 80°, \phi = 90°$、入射粒子を 1GeV の電子として実行した。$\theta = 90°$ 方向には、TPC の中央仕切面があるため、$\theta = 80°$ とする。カロリメータの検出層のすべてのセルのエネルギーの和を求めた。これを 1000 イベントで行った結果を図 3 に示す。曲線は、ガウス分布としてフィッティングしたものである。

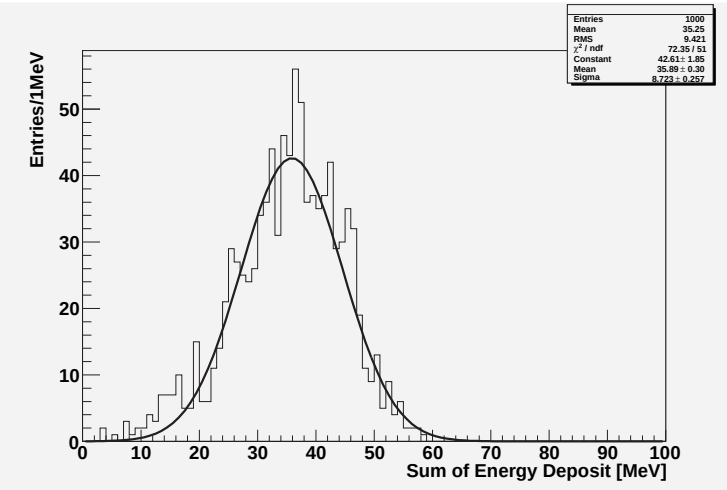

図 3: 1GeV の電子のシンチレータのエネルギー和の分布

中心値 $= 35.89 \pm 0.30$MeV, 標準偏差 $= 8.723 \pm 0.257$MeV となった。エネルギーの精度を表す指標となるエネルギー分解能について考える。エネルギー分解能は、ガウスフィットしたものから、標準偏差/中心値 により求める。エネルギー分解能は、$24.3 \pm 0.7\%$ となる。

入射エネルギーを変化させ、それぞれの場合のエネルギー分解能の値を求めた。これを表 1 に示す。横軸を $1/\sqrt{\text{入射エネルギー}}$、縦軸をエネルギー分解能としたグラフを図 4 に示す。

表 1: 電子のエネルギー分解能

入射エネルギー [GeV]	エネルギー和 [MeV] 中心値	標準偏差	エネルギー分解能 [%]
1	35.89 ± 0.30	8.723 ± 0.257	24.3 ± 0.7
1.5	60.39 ± 0.29	8.986 ± 0.227	14.9 ± 0.4
2	82.67 ± 0.29	8.862 ± 0.246	10.7 ± 0.3
5	208.7 ± 0.5	15.27 ± 0.41	7.32 ± 0.20
7	292.7 ± 0.6	17.47 ± 0.54	5.97 ± 0.18
10	418.6 ± 0.7	20.26 ± 0.66	4.84 ± 0.16

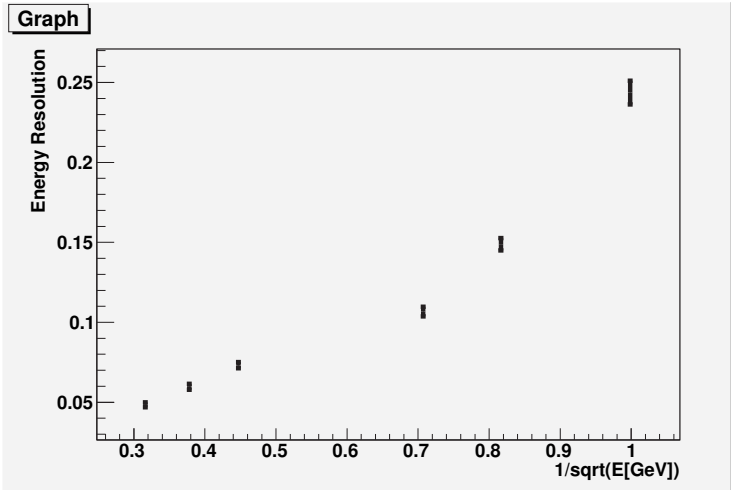

図 4: 電子のエネルギー分解能

入射粒子を光子に変えて、電子の場合と同様にエネルギー分解能を求めた。これを表 2、図 5 に示す。

光子の場合は直線でうまくフィットするが、電子の場合は直線状にならない。磁場の影響で電子が曲げられたためではないかと考えられる。1GeV の電子の場合には、磁場で曲げられて、エンドキャップ部に入る。このため、1GeV, 2GeV の場合にはデータを別に処理する必要がある。

光子の場合について、横軸に入射エネルギー、縦軸にエネルギー分解能をとったものを、図 6 に示す。$\sqrt{a^2/E + b^2}$ でフィッティングを行い、$a =$

表 2: 光子のエネルギー分解能

入射エネルギー [GeV]	エネルギー和 [MeV] 中心値	標準偏差	エネルギー分解能 [%]
1	41.47 ± 0.22	6.751 ± 0.175	16.3 ± 0.4
2	82.89 ± 0.31	9.259 ± 0.233	11.2 ± 0.3
5	209.8 ± 0.5	14.89 ± 0.44	7.10 ± 0.21
7	291.7 ± 0.6	16.69 ± 0.51	5.72 ± 0.18
10	417.2 ± 0.7	19.58 ± 0.66	4.69 ± 0.16

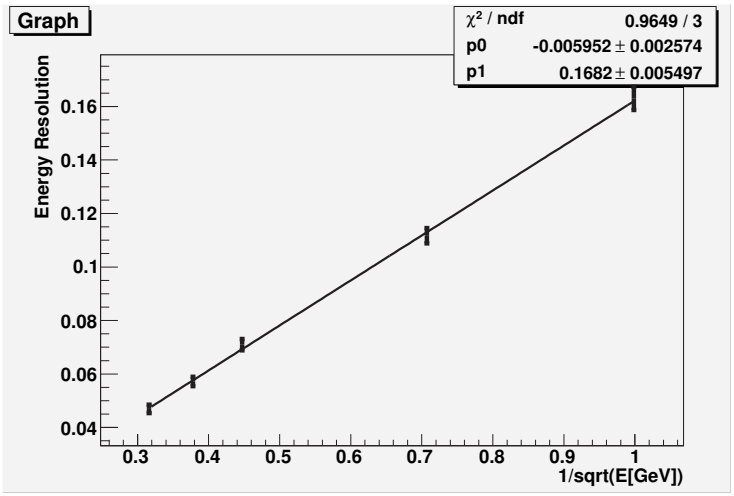

図 5: 光子のエネルギー分解能

0.156 ± 0.002, $b = 0.000 \pm 0.006$ となった。エネルギー分解能は、$(0.156 \pm 0.002)/\sqrt{E[\mathrm{GeV}]} \oplus (0.000 \pm 0.006)\%$ となる。この値は、ビームテストの場合の $16\%/\sqrt{E[\mathrm{GeV}]}$[3] と近い値となった。

3.2.2 モリエール半径

10GeV の光子を $\theta = 90°$, $\phi = 90°$ (y 軸方向) に入射させ、モリエール半径を計算する。計算方法は次の通りである。

1. 全てのセルのエネルギー和を求める。これを E_{total} とする。
2. 入射軸 (y 軸) からの距離が小さい順にエネルギーを加えていく。このエネルギー和を E とする。E/E_{total} が 90% を超えるまで続ける。

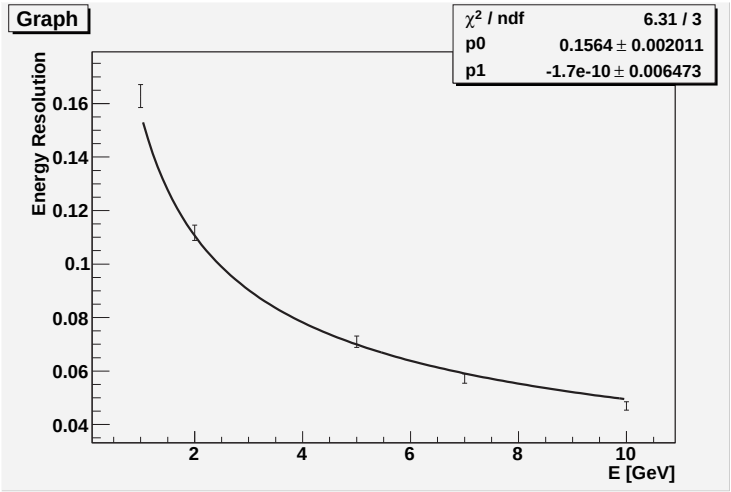

図 6: 光子のエネルギー分解能

3. E/E_{total} が 90% を超えたときの入射軸からの距離をモリエール半径とする。

1000 イベントを重ねて計算した結果は 57.1mm となった。

一方、井手康裕氏の論文 [4] によると、理論値は 17.6mm である。

図 7 にエネルギーの拡がりの様子を示す。横軸は y 軸からの距離、縦軸は y 軸からの距離が同じ点のエネルギーを、1000 イベントについて加えた値である。y 軸から近い部分に多くのエネルギーが吸収されている。一番左側の点 ($R = 5\sqrt{2}\mathrm{mm}$) よりも左側から二番目の点 ($R = 5\sqrt{10}\mathrm{mm}$) の方が大きいのは、セルの数によるためである。一番左側の点は、x, z 座標の組み合わせが $(\pm 5, \pm 5)$ となる 4 つの部分のエネルギー和であるのに対して、左側から二番目の点は、$(\pm 5, \pm 15)$, $(\pm 15, \pm 5)$ となる 8 つの部分エネルギー和であるからである。他の部分も同様である。

$(x, y, z) = (0, 0, 50)\mathrm{mm}$ から y 軸に平行に入射して同様に行い、モリエール半径を計算したが、変化はなかった。

モリエール半径が 57.1mm になった原因については、粒子の拡がり、計算方法についてより詳細に調べる必要がある。また、図 7 で横軸の幅について考慮する必要がある。図 7 では、入射軸からセルの中心までの距離の点についてのエネルギーの和を求めてあるが、セルの中心以外の部分も考える必要があり、ある点とその隣の点との境界を求める必要がある。

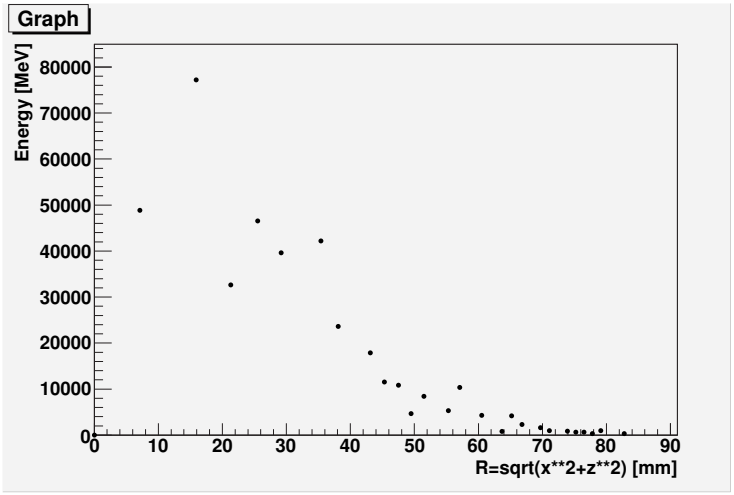

図 7: 光子のエネルギーの拡がりの様子

3.3 エネルギー較正

3.3.1 検出層のエネルギーと入射エネルギー

エネルギーを測定するのは検出層の部分であるが、吸収層の部分にもエネルギーが落ちる。カロリメータに入射したエネルギーを求めるには、吸収層の部分も考える必要がある。検出層での値から、入射エネルギーを求めることを考える。

ある入射エネルギーの光子を入射させる。3.2.1 節と同様に、シンチレータの全ての領域の部分のエネルギーの和のヒストグラムを作り、それをガウス分布でフィッティングしたものの平均を求める。ここでは 100 イベントで行った。入射エネルギーを 1GeV から 10GeV まで変化させ、検出層のエネルギーの和との関係を求める。

結果は表 3 のようになった。これをグラフにしたものを図 8 に示す。縦軸に入射エネルギー、横軸にシンチレータの部分のエネルギーの和をとってある。

フィットした直線の傾きは、24.0 ± 0.8 となった。この値は、シンチレータのエネルギー和に対する入射エネルギーの割合となる。シンチレータのエネルギーから入射エネルギーを求める場合は、24.0 を掛ければ求められることになる。

表 3: 光子の入射エネルギーとシンチレータのエネルギー和

入射エネルギー [GeV]	シンチレータのエネルギー和 [MeV]
1	41.1 ± 7.0
2	82.0 ± 9.8
3	126.0 ± 12.4
4	164.4 ± 11.4
5	208.9 ± 15.5
6	246.1 ± 16.9
7	293.7 ± 16.4
8	332.8 ± 17.3
9	373.7 ± 18.8
10	417.3 ± 19.7

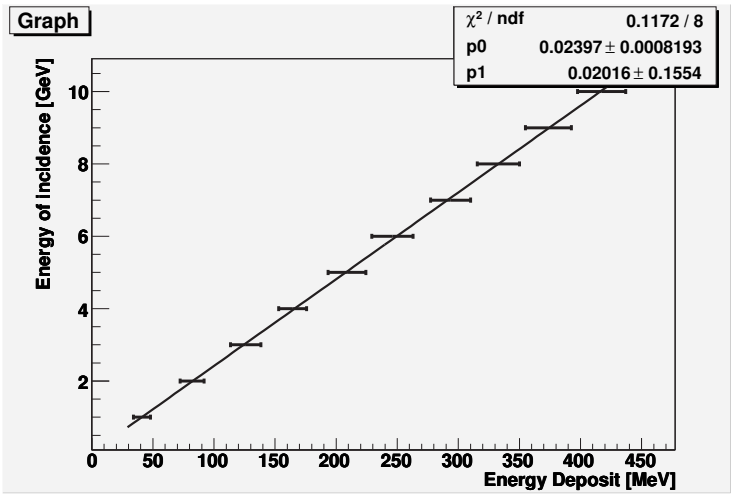

図 8: 検出エネルギーと入射エネルギーの関係

3.3.2 2 つの光子のエネルギー和

5GeV の π^0 を $\theta = 90°$, $\phi = 90°$ (y 軸方向) に入射させ、崩壊してできた 2 つの光子のエネルギーの和を計算した。2.3.1 で述べたように、π^0 は、すぐに崩壊するので y 軸方向で問題ないと考えられる。計算方法は次の通りである。

1. 空間上に生成点を通る軸を考える。軸の回り 5cm 以内にあるセルのエネルギー和を計算する。軸を少しずつ動かしながら、軸の回り 5cm 以内のエネルギー和が最大となる軸を見つける。このエネルギー和を E_1' とする。E_1' がエネルギーが大きい方の光子のエネルギーの検出層の部

分で測定された値となる。5cm という大きさは、小さすぎると光子の全てのエネルギーを収めることができず、大きすぎると 2 で述べるもう一方の光子のエネルギーと重なってしまうため、適当な大きさとして 5cm としてある。

2. 1 と同様にして、軸の回りのエネルギー和が 2 番目に大きくなる軸を見付ける。ただし、2 つの軸が近すぎて、半径 5cm の円柱同士が重ならないようにする。このときのエネルギー和を E_2' とする。これがエネルギーが小さい方の光子のエネルギーの検出層の部分で測定された値となる。

3. 吸収層の部分も考える。3.3.1 節で求めた値 24.0 を用いて、$E_1 = 24.0 \times E_1'$, $E_2 = 24.0 \times E_2'$ を計算する。E_1, E_2 は、吸収層の部分も考慮に入れた光子のエネルギーとなる。

4. 2 つの光子のエネルギー和 $E_1 + E_2$ を求める。

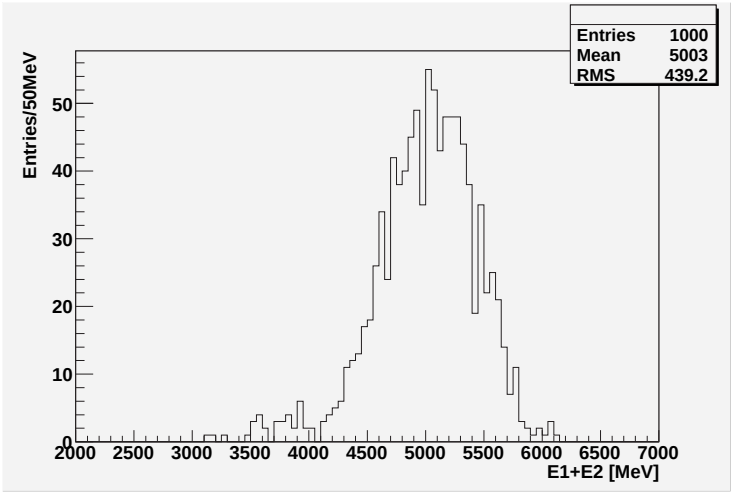

図 9: $E_1 + E_2$ の分布

$E_1 + E_2$ の分布の結果を図 9 に示す。π^0 の入射エネルギー 5GeV を中心に分布していることがわかる。

3.4 π^0 の不変質量と角度分解能

3.4.1 不変質量

実際の実験では、多くの光子が測定される。その光子から元の粒子を調べることが必要になる。2.3.1 節で述べたように、π^0 は 2 つの光子に崩壊する。測定された光子の 2 つの組から、元の粒子が π^0 であるか確かめることができる。ここでは、π^0 を入射させ、崩壊してできた 2 つの光子から、π^0 の不変質量を求めることを考える。

5GeV の π^0 を $\theta = 90°, \phi = 90°$ (y 軸方向) に入射させ、不変質量を求めた。2.3.1 で述べたように、π^0 は、すぐに崩壊するので y 軸方向で問題ないと考えられる。計算方法は次の通りである。

1. 3.3.2 節と同じ方法で E_1, E_2 を求める。
2. E_1 の軸と E_2 の軸のなす角 θ を求める。
3. 式 (11) より、不変質量 M を計算する。

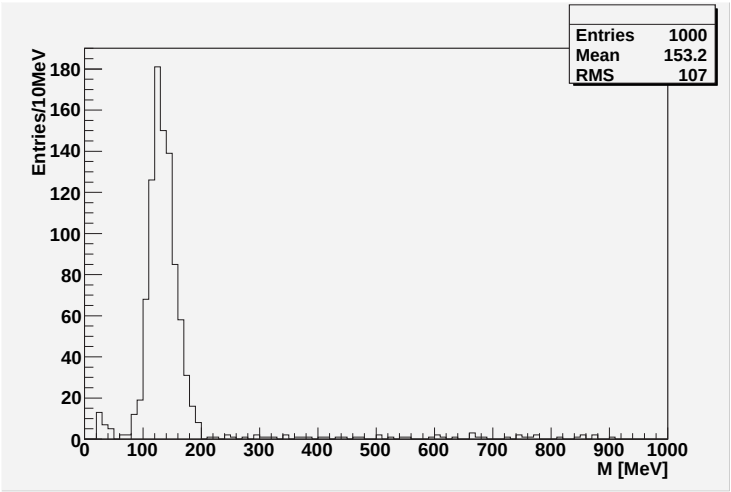

図 10: 再構成された π^0 の不変質量分布

1000 イベントで計算したときの結果を図 10、図 11 に示す。図 11 は図 10 を拡大したものである。ガウス分布でフィッティングしてある。平均および誤差は、$M = 133.1 \pm 0.8$MeV となり、誤差の 3 倍の範囲内で一致した。

不変質量の計算において、一部うまく計算できない場合がある。M の値が小さい値と非常に大きい値になる場合である。小さい値については、θ が小さ

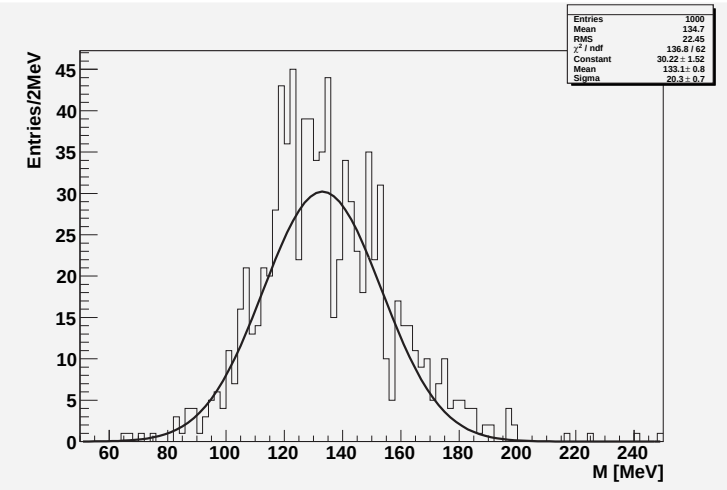

図 11: 再構成された π^0 の不変質量分布 (拡大)

く、2 つの光子が近くなった場合で、エネルギーが重なっているときである。軸の周り 5cm 以内の部分が重なってしまい、2 つの光子のエネルギーがうまく計算できなかったと考えられる。大きい値については、逆に θ が大きくて、その結果 M も大きくなったと考えられる。この場合は、2 つの軸の 5cm の部分にすべてのエネルギーが収まっていないことが多い。エネルギーが拡がっていて、光子の軸がうまく計算できていない可能性がある。

このように、2 つの光子から不変質量を計算することにより、元の粒子が π^0 かどうかについて判別することができる。多数の光子がある場合でも、光子を 2 つの組にして選び、元の粒子が π^0 かどうか判別できる。しかし、一部の場合については計算できない。π^0 の崩壊した 2 つの光子の角度が小さいときにどう処理するかを検討することと、2 つの光子の角度が大きいときの原因を考えることが必要になる。

3.4.2 入射エネルギーと π^0 の不変質量

3.4.1 節では入射エネルギーが 5GeV のときについての π^0 の不変質量を求めた。ここでは、入射エネルギーと π^0 の不変質量の関係を調べるために、入射エネルギーを変化させて行った。

入射する π^0 のエネルギーを 1GeV から 5GeV まで変化させ、3.4.1 節と同様に、不変質量を求めた。π^0 を $\theta = 45°$ から 135° の角度で一様に入射した。

ヒストグラムをガウス分布でフィットしたときの結果を表 4 に示す。入射エネルギーと π^0 の不変質量の関係を図 12 に示す。平均はエネルギーによら

表 4: 入射エネルギーと π^0 の不変質量

入射エネルギー [GeV]	π^0 の不変質量 [MeV]	
	中心値	標準偏差
1	134.7 ± 0.9	26.83 ± 0.89
2	134.3 ± 0.8	20.97 ± 0.58
3	135.3 ± 0.7	19.83 ± 0.52
4	133.6 ± 0.7	19.97 ± 0.62
5	134.0 ± 0.7	19.08 ± 0.54

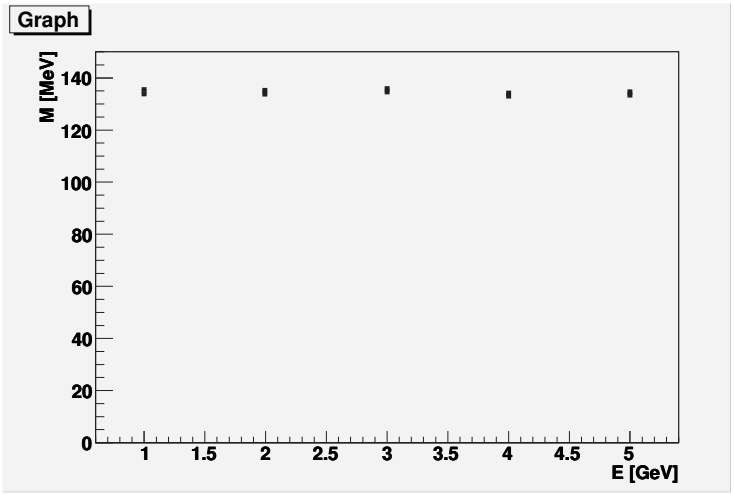

図 12: 入射エネルギーと π^0 の不変質量

ず、134.98MeV に近い値で、ほぼ一定でになっていることがわかる。標準偏差は、入射エネルギーが 1GeV のときだけ大きくなっているが、原因は不明である。1GeV から 5GeV について、すべて同じ計算方法で求めてある。

入射エネルギー 1GeV から 5GeV のときの π^0 の不変質量の平均および誤差は、$M = 134.4 \pm 0.3$MeV となる。

3.4.3 角度分解能

カロリメータのエネルギーと座標の情報から、入射してきた方向が求められる。この方向と実際に入射してきた方向との角度のずれを評価するために、角度分解能について考える。

5GeV の π^0 を $\theta = 90°$, $\phi = 90°$ (y 軸方向) に入射させ、π^0 から崩壊してできた 2 つの光子のうち、エネルギーの大きい方の光子の軸の周りのエネル

ギーから、入射軸を求め、それが生成点とどれだけ離れているかを計算する。電磁カスケードシャワーにより拡がり、ヒットしたセルのエネルギーと座標により、軸を求める。計算方法は次の通りである。

1. 3.3.2 節の方法 1 と同様に、エネルギーが最大となる軸を見つける。この軸からの距離が 5cm 以内の全てのヒットに対して、エネルギーを重みとして最小二乗法でフィッティングを行い直線を求める。最小二乗法は、まず x-y 平面でフィッティングを行い直線を求める。これを y' 軸とする。次に、y'-z 平面でフィッティングを行い直線を求める。これが求める直線となる。
2. 1 で求められた直線と生成点との距離 r を求める。
3. エネルギーを重みとした電磁シャワーの重心点と、生成点の距離 d を求める。
4. $r/d\,[\mathrm{rad}]$ を求める。

1000 イベント生成したときの r/d の分布を図 13 に示す。$Ax\exp\left(-\dfrac{x^2}{2\sigma^2}\right)$ の形でフィッティングしてある。角度分解能 $\sigma = 0.0697\pm0.0014\mathrm{rad} = 4.0\pm0.1^\circ$ となった。

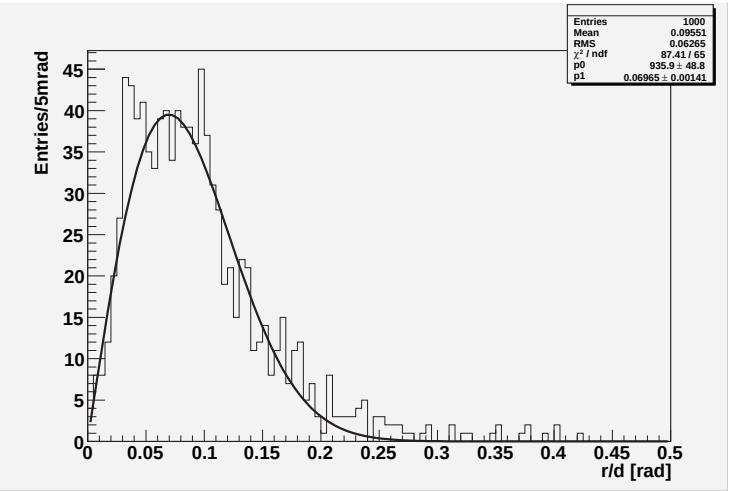

図 13: 角度分解能

計算方法の 1 の別の方法として、カロリメータの層ごとに分ける方法でも計算を行った。エネルギーを重みとして、各層の重心点を求める、それらの

表 5: 入射エネルギーと角度分解能

入射エネルギー [GeV]	σ[rad]
1	0.1248 ± 0.0028
1.5	0.1005 ± 0.0023
2	0.09187 ± 0.00199
3	0.07171 ± 0.00151
5	0.05807 ± 0.00134
10	0.04433 ± 0.00084
50	0.02797 ± 0.00055

重心点に対して最小二乗法でフィッティングするという方法である。重心点に対してフィッティングするときの重みは、各層のエネルギー和である。結果は、$\sigma = 0.0680 \pm 0.0014\text{rad} = 3.9 \pm 0.1^\circ$ となり、層に分けずに計算したときの場合とほぼ同じであった。

ここでは、5GeV 未満の様々なエネルギーの光子についての角度分解能を計算した。4.0° 程度のずれがあることになる。より良い角度分解能を得るためには、セルの大きさや材質を考える必要がある。

3.4.4 入射エネルギーと角度分解能

3.4.3 節では、π^0 から崩壊してできた光子から角度分解能を求めた。π^0 から崩壊してできた光子は、様々なエネルギーをもっている。ここでは、入射エネルギーとの関係を求めたい。そのため、入射粒子を π^0 ではなく光子し、入射エネルギーを決めて行う。

光子の入射エネルギーを、1, 1.5, 2, 3, 5, 10, 50GeV として、それぞれの場合について、3.4.3 節の方法と同様に角度分解能を計算した。光子は、$\theta = 45^\circ$ から 135° の範囲で一様に入射した。

r/d の分布を $Ax \exp\left(-\dfrac{x^2}{2\sigma^2}\right)$ の形でフィッティングし、入射エネルギーと σ の関係を表した関係を、表 5 に示す。これを横軸を $1/\sqrt{\text{入射エネルギー}}$、縦軸を角度分解能としたものを図 14 に示す。

横軸に入射エネルギー、縦軸に角度分解能をとったものを、図 15 に示す。$\sqrt{a^2/E + b^2}$ でフィッティングを行い、$a = 0.122 \pm 0.001,\ b = 0.022 \pm 0.001$ となった。角度分解能は、$(0.122 \pm 0.001)/\sqrt{E[\text{GeV}]} \oplus (0.022 \pm 0.001)\text{rad}$ となる。

入射エネルギーと角度分解能の関係が求められ、それぞれの入射エネルギーにおいて、どの程度の角度のずれがあるのかが求められることになる。

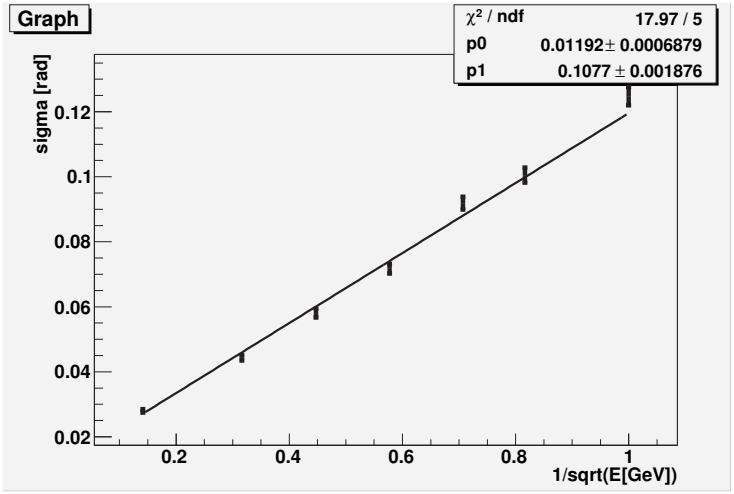

図 14: 入射エネルギーと角度分解能

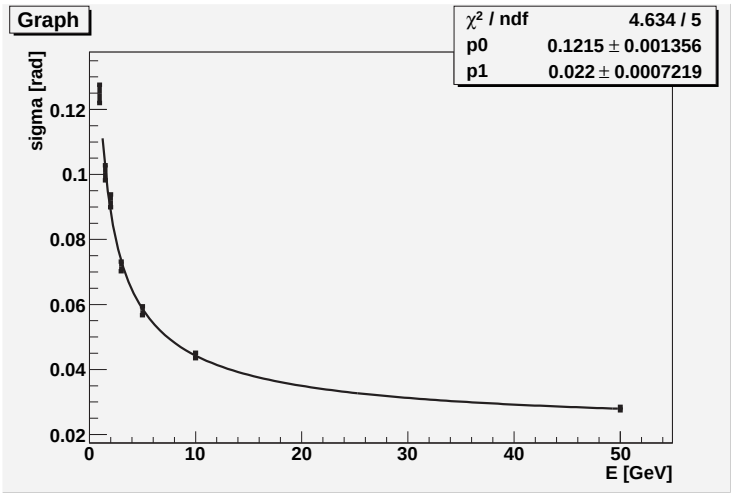

図 15: 入射エネルギーと角度分解能

3.4.5 2 つの軸の距離

3.4.3 節では、エネルギーの大きい方の光子のエネルギーからフィッティングを行い直線を求めた。π^0 から崩壊してできたもう一方の光子からも、同様にしてフィッティングを行い直線を求め、2 つの直線の交わる角度を考え、π^0

の崩壊した角度を考えたい。しかし、空間上では 2 つの直線は交わるとは限らないので、ここでは、2 つの直線の距離を計算した。

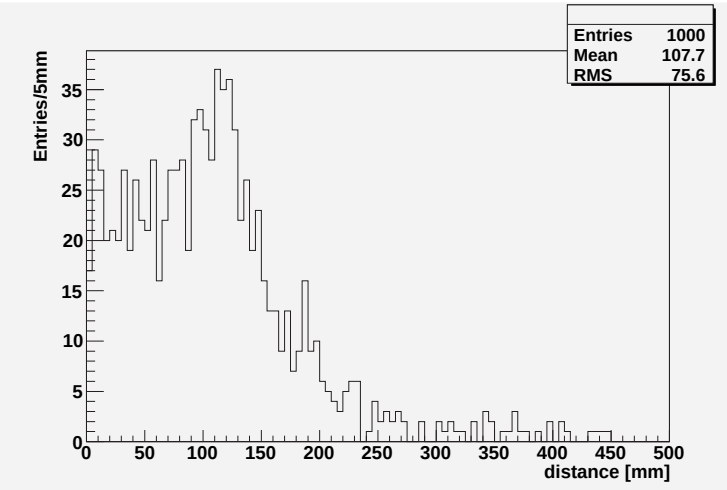

図 16: 2 つの軸の距離

1000 イベントでの結果を図 16 に示す。ヒストグラムの形は、$Ax \exp\left(-\frac{x^2}{2\sigma^2}\right)$ と異なり、2 つの軸の距離が小さい部分 (0mm から 50mm の部分) が多くなっており、ピークの位置は 110mm から 120mm 付近である。平均は 107.7mm である。

カロリメータからの情報から 2 つの軸を計算した場合、平均で 107.7mm の距離がある。2 つの軸のなす角を求める場合、どう処理するかを考える必要がある。

4　まとめ

Jupiter を使って電磁カロリメータのシミュレーションを行った。

入射エネルギーとエネルギー分解能の関係について、電子の場合と光子の場合について求められた。光子エネルギー分解能は、$(0.156\pm0.002)/\sqrt{E[\mathrm{GeV}]}\oplus(0.000\pm0.006)\%$ となり、ビームテストの場合と近い値が得られた。

π^0 の崩壊してできた 2 つの光子から、不変質量を求めた。π^0 の入射エネルギーを変化させて行い、不変質量は入射エネルギーによらず、ほぼ一定であり、入射エネルギーが 1GeV から 5GeV までの平均および誤差は、$M = 134.4 \pm 0.3\mathrm{MeV}$ となった。しかし、π^0 の崩壊した 2 つの光子の角度によっては、不変質量がうまく計算できていない。新たな計算方法を考える必要がある。

角度分解能の計算では、入射エネルギーと角度分解能の関係が求められ、$(0.122 \pm 0.001)/\sqrt{E[\mathrm{GeV}]} \oplus (0.022 \pm 0.001)\mathrm{rad}$ となった。

謝辞

本研究を行うにあたって、多くの方にお世話になりました。御指導していただいた竹下徹先生、長谷川庸司先生に深く感謝します。魚住聖氏にはミーティングでアドバイスをいただきました。また、院生の皆様にもお世話になりました。特に井手康裕氏には、Jupiter について一から教えていただきました。皆様に深く感謝いたします。

参考文献

[1] 長島順清 『素粒子物理学の基礎 I』 (朝倉書店, 1998)

[2] 渡邊靖志 『素粒子物理入門=基本概念から最先端まで』 (培風館, 2002)

[3] http://ppwww.phys.sci.kobe-u.ac.jp/~jeans/scecal_calicenote/ecal_note_Q_and_A.html

[4] 井手康裕 『Jupiter を使ったカロリメータのシミュレーション』 (2007)

A　プログラム

データの処理を行う際に用いたプログラムを載せる。

A.1　モリエール半径

3.2.2 節で使ったモリエール半径を求めるプログラムである。カロリメータのどの部分でどれだけのエネルギーが測定されたのかを、“file1.dat” に用意しておく。“file1.dat” は、

エネルギー　x 座標　y 座標　z 座標

の組のデータが 1000 イベント分書かれている。

```
#include <stdio.h>
#include <math.h>
#define N 26311 // file1.dat の行数

int main()
{
  int i;
  float E_total, E;
  float edep[N], x[N], y[N], z[N], R[N];
  float RM;

  // ファイルからデータを読み込む
  FILE *fp = fopen("file1.dat", "r");
  for(i=0; i<N; i++)
    fscanf(fp, "%f␣%f␣%f␣%f", &edep[i], &x[i], &y[i], &z[i]);
  fclose(fp);

  // R の定義
  for(i=0; i<N; i++)
    R[i]=sqrt(x[i]*x[i]+z[i]*z[i]); // 入射軸 (y 軸)からの距離

  // 初期化
  E_total = 0.0;
  E = 0.0;
  RM = 0.0;

  // 全てのセルのエネルギー和を求める
  for(i=0; i<N; i++)
    E_total += edep[i];

  while(E/E_total < 0.9) // 90%を超えるまで続ける
  {
    for(i=0; i<N; i++)
      if((R[i] >= RM) && (R[i] < RM+0.1))
        E += edep[i]; // 入射軸 (y 軸)からの距離が小さい順にエネルギーを加える
    RM += 0.1;
  }
```

```
38
39   printf("RM␣=␣%f\n", RM);
40   return 0;
41 }
```

A.2 π^0 の不変質量

3.3.2 節の 2 つの光子のエネルギーと、3.4.1 節の π^0 の不変質量を求めるプログラムである。カロリメータのどの部分でどれだけのエネルギーが測定されたのかを、“file2.dat” に用意しておく。“file2.dat” は、

エネルギー　x 座標　y 座標　z 座標

の組のデータが 1000 イベント分書かれており、イベントの境目は、

0.0　0.0　0.0　0.0

となっている。計算した結果の光子のエネルギー、方向などのデータを “file3.dat” に保存しておく。

```
1  #include <stdio.h>
2  #include <math.h>
3  #define EVENT 1000 // イベント数
4
5  // 原点と (x1,y1,z1)を通る直線と、原点と (x2,y2,z2)とを通る直線がなす角度を求める関数
6  float nasukaku(float x1, float y1, float z1, float x2, float y2, float z2)
7  {
8    float theta;
9    theta = acos((x1*x2+y1*y2+z1*z2)/(sqrt((x1*x1+y1*y1+z1*z1)*(x2*x2+y2*y2+z2*z2))));
         // 2つの直線の間の角度
10   return theta;
11 }
12
13 int main()
14 {
15   int i, j, k[EVENT];
16   float x[EVENT][1000], y[EVENT][1000], z[EVENT][1000], edep[EVENT][1000];
17   float x11[EVENT][1000], y11[EVENT][1000], z11[EVENT][1000], y21[EVENT][1000];
18   float x12[EVENT][1000], y12[EVENT][1000], z12[EVENT][1000], y22[EVENT][1000];
19   float theta1, phi1, theta2, phi2;
20   float theta10, phi10, theta20, phi20;
21   float E1, E2, E10, E20;
22   float theta, M2, M;
23   float pi, R, delta;
24
25   pi = 3.141592; // 円周率π
26   R = 50.0; // E1, E2 のエネルギーを収める円柱の半径 5cm
27   delta = 0.005; //  θ、Φを少しずつずらす角度
28
29   // ファイルからデータを読み込む
30   FILE *fp1 = fopen("file2.dat", "r");
```

```
31
32  // 書き込みファイルを用意
33  FILE *fp2 = fopen("file3.dat", "wt");
34
35  i = 0;
36  j = 0;
37
38  // 読み取ったデータをイベントごとに分ける。
39  // j はイベント番号、i はそのイベントでのヒットの番号
40  while(j < EVENT)
41  {
42    fscanf(fp1, "%f␣%f␣%f␣%f", &edep[j][i], &x[j][i], &y[j][i], &z[j][i]);
43    if((edep[j][i]==0.0) && (x[j][i]==0.0) && (y[j][i]==0.0) && (z[j][i]==0.0))
44      {
45        k[j] = i;
46        j++;
47        i = 0;
48      }
49    else
50      i++;
51  }
52  fclose(fp1);
53
54  for(j=0; j<EVENT; j++)
55  {
56    E1 = 0.0;
57
58    // θ,φを少しずつずらして最大となる軸を求める。
59    for(theta10=0.0; theta10<pi; theta10+=delta){
60      for(phi10=0.0; phi10<pi; phi10+=delta){
61
62        // θ,φ方向にy'軸をとる。xy 平面状に x'軸がくるようにして、z'軸も決める。// x',y
              ',z'座標を求める。
63        for(i=0; i<k[j]; i++)
64        {
65          y21[j][i] = x[j][i]*cos(phi10)+y[j][i]*sin(phi10);
66          x11[j][i] = x[j][i]*sin(phi10)−y[j][i]*cos(phi10); //x'座標
67          y11[j][i] = y21[j][i]*sin(theta10)+z[j][i]*cos(theta10); //y'座標
68          z11[j][i] = −y21[j][i]*cos(theta10)+z[j][i]*sin(theta10); //z'座標
69        }
70
71        // y'軸からR 以内の距離にあるエネルギーの和を求める。
72        E10 = 0.0;
73        for(i=0; i<k[j]; i++)
74          if(sqrt(x11[j][i]*x11[j][i]+z11[j][i]*z11[j][i]) <= R)
75            E10 += edep[j][i];
76
77        // エネルギー最大のときを求める。
78        if(E1<E10)
79        {
80          E1 = E10;
81          theta1 = theta10;
82          phi1 = phi10;
83        }
```

```
84
85      } }
86
87      fprintf(fp2, "%f␣%f␣%f␣", E1, theta1, phi1);
88
89      E2 = 0.0;
90
91      for(theta20=0.0; theta20<pi; theta20+=delta){
92        for(phi20=0.780; phi20<pi; phi20+=delta){
93
94          // E1 の円柱と E2 の円柱が重ならないようにする。
95          if(nasukaku(sin(theta1)*cos(phi1), sin(theta1)*sin(phi1), cos(theta1), sin(theta20)*cos(
                phi20), sin(theta20)*sin(phi20), cos(theta20)) >= 2*asin(R/2100.0))
96          {
97            for(i=0; i<k[j]; i++)
98            {
99              y22[j][i] = x[j][i]*cos(phi20)+y[j][i]*sin(phi20);
100             x12[j][i] = x[j][i]*sin(phi20)−y[j][i]*cos(phi20);
101             y12[j][i] = y22[j][i]*sin(theta20)+z[j][i]*cos(theta20);
102             z12[j][i] = −y22[j][i]*cos(theta20)+z[j][i]*sin(theta20);
103           }
104
105           E20 = 0.0;
106           for(i=0; i<k[j]; i++)
107             if(sqrt(x12[j][i]*x12[j][i]+z12[j][i]*z12[j][i]) <= R)
108               E20 += edep[j][i];
109
110           if(E2<E20)
111           {
112             E2 = E20;
113             theta2 = theta20;
114             phi2 = phi20;
115           }
116
117         }
118
119     } }
120
121     fprintf(fp2, "%f␣%f␣%f␣", E2, theta2, phi2);
122
123     // E1 の軸と E2 の軸のなす角を求める。
124     theta = nasukaku(sin(theta1)*cos(phi1), sin(theta1)*sin(phi1), cos(theta1), sin(theta2)*cos
            (phi2), sin(theta2)*sin(phi2), cos(theta2));
125     fprintf(fp2, "%f␣", theta);
126
127     // 不変質量M を求める。
128     M2 = 2*E1*E2*(1−cos(theta));
129     M = sqrt(M2); // これに 24.0を掛けた値が不変質量
130     fprintf(fp2, "%f␣%f\n", M2, M);
131   }
132
133   fclose(fp2);
134   return 0;
135 }
```

A.3 角度分解能

3.4.3 節の角度分解能と 3.4.5 節の 2 つの軸の距離の計算で用いたプログラムである。A.2 で述べた “file2.dat” と A.2 で作成した “file3.dat” を使ってある。

```
#include <stdio.h>
#include <math.h>
#define EVENT 1000 // イベント数

int main()
{
  int i, j, k[EVENT];
  float pi, R;
  float x[EVENT][1000], y[EVENT][1000], z[EVENT][1000], edep[EVENT][1000];
  float ydash1[EVENT][1000], ydash2[EVENT][1000];
  float x11[EVENT][1000], y11[EVENT][1000], z11[EVENT][1000], y21[EVENT][1000];
  float x12[EVENT][1000], y12[EVENT][1000], z12[EVENT][1000], y22[EVENT][1000];
  float E1[EVENT], theta1[EVENT], phi1[EVENT], E2[EVENT], theta2[EVENT], phi2[
      EVENT], theta[EVENT], M2[EVENT], M[EVENT];
  float sum1_E[EVENT], sum1_Ex[EVENT], sum1_Ey[EVENT], sum1_Ey2[EVENT],
      sum1_Exy[EVENT];
  float sum2_E[EVENT], sum2_Ex[EVENT], sum2_Ey[EVENT], sum2_Ey2[EVENT],
      sum2_Exy[EVENT];
  float sum1_Eydash[EVENT], sum1_Ez[EVENT], sum1_Eydash2[EVENT], sum1_Eydashz[
      EVENT];
  float sum2_Eydash[EVENT], sum2_Ez[EVENT], sum2_Eydash2[EVENT], sum2_Eydashz[
      EVENT];
  float a1[EVENT], b1[EVENT], m1[EVENT], n1[EVENT], theta01[EVENT];
  float a2[EVENT], b2[EVENT], m2[EVENT], n2[EVENT], theta02[EVENT];
  float x1[EVENT], y1[EVENT], z1[EVENT], r1[EVENT], d1[EVENT], angle1[EVENT];
  float x2[EVENT], y2[EVENT], z2[EVENT], r2[EVENT], d2[EVENT], angle2[EVENT];
  float gpoint_x10[EVENT], gpoint_y10[EVENT], gpoint_z10[EVENT], sumedep10[EVENT];
  float gpoint_x20[EVENT], gpoint_y20[EVENT], gpoint_z20[EVENT], sumedep20[EVENT];
  float A11[EVENT], A12[EVENT], A21[EVENT], A22[EVENT], B1[EVENT], B2[EVENT];
  float x01[EVENT], x02[EVENT], y01[EVENT], y02[EVENT], z01[EVENT], z02[EVENT],
      r12[EVENT];

  pi = 3.141592; // 円周率π
  R = 50.0; // E1, E2 のエネルギーを収める円柱の半径

  FILE *fp1 = fopen("file2.dat", "r");

  i = 0;
  j = 0;

  while(j < EVENT)
  {
    fscanf(fp1, "%f␣%f␣%f␣%f", &edep[j][i], &x[j][i], &y[j][i], &z[j][i]);
```

```
38      if((edep[j][i]==0.0) && (x[j][i]==0.0) && (y[j][i]==0.0) && (z[j][i]==0.0))
39      {
40        k[j] = i;
41        j++;
42        i = 0;
43      }
44      else
45        i++;
46    }
47    fclose(fp1);
48
49    FILE *fp2 = fopen("file3.dat", "r");
50    for(j=0; j<EVENT; j++)
51    {
52      fscanf(fp2, "%f␣%f␣%f␣%f␣%f␣%f␣%f␣%f␣%f", &E1[j], &theta1[j], &phi1[j], &E2[j], &theta2[j
            ], &phi2[j], &theta[j], &M2[j], &M[j]);
53    }
54    fclose(fp2);
55
56    for(j=0; j<EVENT; j++)
57    {
58
59      // 座標変換
60      for(i=0; i<k[j]; i++)
61      {
62        y21[j][i] = x[j][i]*cos(phi1[j])+y[j][i]*sin(phi1[j]);
63        x11[j][i] = x[j][i]*sin(phi1[j])-y[j][i]*cos(phi1[j]);   // x'座標
64        y11[j][i] = y21[j][i]*cos(pi/2-theta1[j])+z[j][i]*sin(pi/2-theta1[j]); // y'座標
65        z11[j][i] = -y21[j][i]*sin(pi/2-theta1[j])+z[j][i]*cos(pi/2-theta1[j]); // z'座標
66
67        y22[j][i] = x[j][i]*cos(phi2[j])+y[j][i]*sin(phi2[j]);
68        x12[j][i] = x[j][i]*sin(phi2[j])-y[j][i]*cos(phi2[j]);   // x'座標
69        y12[j][i] = y22[j][i]*cos(pi/2-theta2[j])+z[j][i]*sin(pi/2-theta2[j]); // y'座標
70        z12[j][i] = -y22[j][i]*sin(pi/2-theta2[j])+z[j][i]*cos(pi/2-theta2[j]); // z'座標
71      }
72
73      gpoint_x10[EVENT] = 0.0;
74      gpoint_y10[EVENT] = 0.0;
75      gpoint_z10[EVENT] = 0.0;
76      gpoint_x20[EVENT] = 0.0;
77      gpoint_y20[EVENT] = 0.0;
78      gpoint_z20[EVENT] = 0.0;
79      sumedep10[EVENT] = 0.0;
80      sumedep20[EVENT] = 0.0;
81
82      sum1_E[j] = 0.0;
83      sum1_Exy[j] = 0.0;
84      sum1_Ex[j] = 0.0;
85      sum1_Ey[j] = 0.0;
86      sum1_Ey2[j] = 0.0;
87      sum1_Eydashz[j] = 0.0;
88      sum1_Eydash[j] = 0.0;
89      sum1_Ez[j] = 0.0;
90      sum1_Eydash2[j] = 0.0;
```

```
91      sum2_E[j] = 0.0;
92      sum2_Exy[j] = 0.0;
93      sum2_Ex[j] = 0.0;
94      sum2_Ey[j] = 0.0;
95      sum2_Ey2[j] = 0.0;
96      sum2_Eydashz[j] = 0.0;
97      sum2_Eydash[j] = 0.0;
98      sum2_Ez[j] = 0.0;
99      sum2_Eydash2[j] = 0.0;
100   }
101
102   for(j=0; j<EVENT; j++)
103   {
104
105     // クラスターの重心点を求める。
106     for(i=0; i<k[j]; i++)
107       if(sqrt(x11[j][i]*x11[j][i]+z11[j][i]*z11[j][i]) <= R)
108       {
109         gpoint_x10[j] += edep[j][i]*x[j][i];
110         gpoint_y10[j] += edep[j][i]*y[j][i];
111         gpoint_z10[j] += edep[j][i]*z[j][i];
112         sumedep10[j] += edep[j][i];
113       }
114     gpoint_x10[j] /= sumedep10[j]; // クラスターの重心点のx 座標
115     gpoint_y10[j] /= sumedep10[j]; // クラスターの重心点のy 座標
116     gpoint_z10[j] /= sumedep10[j]; // クラスターの重心点のz 座標
117     d1[j] = sqrt(gpoint_x10[j]*gpoint_x10[j]+gpoint_y10[j]*gpoint_y10[j]+gpoint_z10[j]*
            gpoint_z10[j]); // クラスターの重心点までの距離
118
119     for(i=0; i<k[j]; i++)
120       if(sqrt(x11[j][i]*x11[j][i]+z11[j][i]*z11[j][i]) <= R)
121       {
122         sum1_E[j] += edep[j][i]; // ΣE
123         sum1_Exy[j] += edep[j][i]*x[j][i]*y[j][i]; // ΣExy
124         sum1_Ex[j] += edep[j][i]*x[j][i]; // ΣEx
125         sum1_Ey[j] += edep[j][i]*y[j][i]; // ΣEy
126         sum1_Ey2[j] += edep[j][i]*y[j][i]*y[j][i]; // ΣEy^2
127       }
128
129     // x-y 平面でフィッティングを行う。y':y=ax+b
130     a1[j] = ( sum1_E[j] * sum1_Exy[j] − sum1_Ex[j] * sum1_Ey[j] ) / ( sum1_E[j] * sum1_Ey2[j]
              − sum1_Ey[j] * sum1_Ey[j] );
131     b1[j] = ( sum1_Ex[j] * sum1_Ey2[j] − sum1_Ey[j] * sum1_Exy[j]) / ( sum1_E[j] * sum1_Ey2[
            j] − sum1_Ey[j] * sum1_Ey[j] );
132     a1[j] = 1.0/a1[j];
133     b1[j] = −a1[j]*b1[j];
134     theta01[j] = atan(a1[j]);
135
136     // y 軸と y'軸が交わる点をy'=0として、y'座標を求める。
137     for(i=0; i<=k[j]; i++)
138       ydash1[j][i] = x[j][i] * cos(theta01[j]) + (y[j][i]−b1[j]) * sin(theta01[j]);
139
140     for(i=0; i<=k[j]; i++)
141       if(sqrt(x11[j][i]*x11[j][i]+z11[j][i]*z11[j][i]) <= R)
```

```
142         {
143           sum1_Eydashz[j] += edep[j][i]*ydash1[j][i]*z[j][i]; // Σ Ey'z
144           sum1_Eydash[j] += edep[j][i]*ydash1[j][i]; // Σ Ey'
145           sum1_Ez[j] += edep[j][i]*z[j][i]; // Σ Ez
146           sum1_Eydash2[j] += edep[j][i]*ydash1[j][i]*ydash1[j][i]; // Σ Ey'^2
147         }
148
149       // y'-z 平面でフィッティングを行う。z=my'+n
150       m1[j] = ( sum1_E[j] * sum1_Eydashz[j] - sum1_Eydash[j] * sum1_Ez[j] ) / ( sum1_E[j] *
              sum1_Eydash2[j] - sum1_Eydash[j] * sum1_Eydash[j]);
151       n1[j] = ( sum1_Ez[j] * sum1_Eydash2[j] - sum1_Eydash[j] * sum1_Eydashz[j] ) / ( sum1_E[
              j] * sum1_Eydash2[j] - sum1_Eydash[j] * sum1_Eydash[j]);
152
153       // 生成点に最接近する点の座標
154       x1[j] = -1.0 * ( a1[j]*b1[j] + m1[j]*n1[j] *( cos(theta01[j]) + a1[j]*sin(theta01[j]) ) ) / (1 +
                a1[j]*a1[j] +m1[j]*m1[j] * (cos(theta01[j])+a1[j]*sin(theta01[j])) * (cos(theta01[j]) +
              a1[j]*sin(theta01[j])) );
155       y1[j] = a1[j] * x1[j] + b1[j];
156       z1[j] = m1[j]*x1[j]*(cos(theta01[j])+a1[j]*sin(theta01[j]))+n1[j];
157       r1[j] = sqrt( x1[j]*x1[j] + y1[j]*y1[j] + z1[j]*z1[j] ); // フィッティングで求めた直線と生成点
              の距離r
158
159       angle1[j] = r1[j]/d1[j];
160
161       for(i=0; i<k[j]; i++)
162         if(sqrt(x12[j][i]*x12[j][i]+z12[j][i]*z12[j][i]) <= R)
163         {
164           gpoint_x20[j] += edep[j][i]*x[j][i];
165           gpoint_y20[j] += edep[j][i]*y[j][i];
166           gpoint_z20[j] += edep[j][i]*z[j][i];
167           sumedep20[j] += edep[j][i];
168         }
169       gpoint_x20[j] /= sumedep20[j];
170       gpoint_y20[j] /= sumedep20[j];
171       gpoint_z20[j] /= sumedep20[j];
172       d2[j] = sqrt(gpoint_x20[j]*gpoint_x20[j]+gpoint_y20[j]*gpoint_y20[j]+gpoint_z20[j]*
              gpoint_z20[j]);
173
174       for(i=0; i<k[j]; i++)
175         if(sqrt(x12[j][i]*x12[j][i]+z12[j][i]*z12[j][i]) <= R)
176           {
177             sum2_E[j] += edep[j][i];
178             sum2_Exy[j] += edep[j][i]*x[j][i]*y[j][i];
179             sum2_Ex[j] += edep[j][i]*x[j][i];
180             sum2_Ey[j] += edep[j][i]*y[j][i];
181             sum2_Ey2[j] += edep[j][i]*y[j][i]*y[j][i];
182           }
183
184       a2[j] = ( sum2_E[j] * sum2_Exy[j] - sum2_Ex[j] * sum2_Ey[j] ) / ( sum2_E[j] * sum2_Ey2[j]
                - sum2_Ey[j] * sum2_Ey[j] );
185       b2[j] = ( sum2_Ex[j] * sum2_Ey2[j] - sum2_Ey[j] * sum2_Exy[j]) / ( sum2_E[j] * sum2_Ey2[
              j] - sum2_Ey[j] * sum2_Ey[j] );
186       a2[j] = 1.0/a2[j];
187       b2[j] = -a2[j]*b2[j];
```

```
188     theta02[j] = atan(a2[j]);
189
190     for(i=0; i<=k[j]; i++)
191       ydash2[j][i] = x[j][i] * cos(theta02[j]) + (y[j][i]−b2[j]) * sin(theta02[j]);
192
193     for(i=0; i<=k[j]; i++)
194       if(sqrt(x12[j][i]*x12[j][i]+z12[j][i]*z12[j][i]) <= R)
195       {
196         sum2_Eydashz[j] += edep[j][i]*ydash2[j][i]*z[j][i];
197         sum2_Eydash[j] += edep[j][i]*ydash2[j][i];
198         sum2_Ez[j] += edep[j][i]*z[j][i];
199         sum2_Eydash2[j] += edep[j][i]*ydash2[j][i]*ydash2[j][i];
200       }
201
202     m2[j] = ( sum2_E[j] * sum2_Eydashz[j] − sum2_Eydash[j] * sum2_Ez[j] ) / ( sum2_E[j] *
            sum2_Eydash2[j] − sum2_Eydash[j] * sum2_Eydash[j]);
203     n2[j] = ( sum2_Ez[j] * sum2_Eydash2[j] − sum2_Eydash[j] * sum2_Eydashz[j] ) / ( sum2_E[
            j] * sum2_Eydash2[j] − sum2_Eydash[j] * sum2_Eydash[j]);
204
205     x2[j] = −1.0 * ( a2[j]*b2[j] + m2[j]*n2[j] *( cos(theta02[j]) + a2[j]*sin(theta02[j]) ) ) / (1 +
             a2[j]*a2[j] +m2[j]*m2[j] * (cos(theta02[j])+a2[j]*sin(theta02[j])) * (cos(theta02[j]) +
            a2[j]*sin(theta02[j])) );
206     y2[j] = a2[j] * x2[j] + b2[j];
207     z2[j] = m2[j]*x2[j]*(cos(theta02[j])+a2[j]*sin(theta02[j]))+n2[j];
208     r2[j] = sqrt( x2[j]*x2[j] + y2[j]*y2[j] + z2[j]*z2[j] );
209
210     angle2[j] = r2[j]/d2[j];
211   }
212
213   for(j=0; j<EVENT; j++)
214     {
215       A11[j] = 1.0 + a1[j]*a1[j] + m1[j]*m1[j]*(cos(theta01[j])+a1[j]*sin(theta01[j]))*(cos(
              theta01[j])+a1[j]*sin(theta01[j]));
216       A22[j] = 1.0 + a2[j]*a2[j] + m2[j]*m2[j]*(cos(theta02[j])+a2[j]*sin(theta02[j]))*(cos(
              theta02[j])+a2[j]*sin(theta02[j]));
217       A12[j] = − ( 1.0 + a1[j]*a2[j] + m1[j]*m2[j]*(cos(theta01[j])+a1[j]*sin(theta01[j]))*(cos(
              theta02[j])+a2[j]*sin(theta02[j])) );
218       A21[j] = A12[j];
219       B1[j] = a1[j]*(b2[j]−b1[j]) + m1[j]*(n2[j]−n1[j])*(cos(theta01[j])+a1[j]*sin(theta01[j]));
220       B2[j] = − a2[j]*(b2[j]−b1[j]) − m2[j]*(n2[j]−n1[j])*(cos(theta02[j])+a2[j]*sin(theta02[j]));
221
222       // もう一方の軸に最接近するときの座標
223       x01[j] = ( A22[j]*B1[j] − A12[j]*B2[j] ) / ( A11[j]*A22[j] − A12[j]*A21[j] );
224       x02[j] = ( A21[j]*B1[j] − A11[j]*B2[j] ) / ( A12[j]*A21[j] − A11[j]*A22[j] );
225       y01[j] = a1[j]*x01[j] + b1[j];
226       y02[j] = a2[j]*x02[j] + b2[j];
227       z01[j] = m1[j]*x01[j]*(cos(theta01[j])+a1[j]*sin(theta01[j])) + n1[j];
228       z02[j] = m2[j]*x02[j]*(cos(theta02[j])+a2[j]*sin(theta02[j])) + n2[j];
229       r12[j] = sqrt((x02[j]−x01[j])*(x02[j]−x01[j]) + (y02[j]−y01[j])*(y02[j]−y01[j]) + (z02[j]−
              z01[j])*(z02[j]−z01[j])); // 2つの軸の距離
230     }
231
232   for(j=0; j<EVENT; j++)
233     printf("%f\n", angle1[j]);
```

```
234   return 0;
235 }
```