オブジェクト指向技術を使用した
アトラス検出器のシミュレータの研究

京都教育大学総合科学課程物質科学専攻
素粒子物理学研究室
994117 山田洋平

平成 15 年 2 月 7 日

はじめに

2007 年に CERN で TeV エネルギー領域での素粒子の観測を行う Atlas 実験が開始される。この高エネルギー領域において、ヒッグス粒子と呼ばれる素粒子が発見されると考えられている。現在までに素粒子物理学の分野で行われた様々な実験の結果を高い精度で系統的に説明できる理論として、標準模型と呼ばれるモデルがある。この標準模型が予言するヒッグス粒子の発見が理論の実証に不可欠であるとされている。

その準備段階として、計算機によるシミュレーションが行われている。計算機シミュレーションは、測定器の性能の検証、及び実験データを解析するプログラムやデータを前処理する機械の開発におけるテスト用データの生成を目的としている。

Atlas 実験の測定器シミュレーションのためのツールキットとして Geant4 というソフトウェアが使用される。Geant4 は C++ というオブジェクト指向言語で実装されている。

現在の実験装置は大規模かつ複雑であり、プログラムコードは膨大なものとなる。またツールキットは多様なシミュレーションに対応せねばならない。そのために必要な保守性と拡張性を実現するために用いられるのがオブジェクト指向と呼ばれる技術である。

C++ はオブジェクト指向言語であり、同時に低レベル言語 C を拡張したものであるため、大規模プログラムの保守の容易さと大量のデータを扱うために必要不可欠な処理の高速さを両立させることが出来る。

Geant4 はフルシミュレータである。フルシミュレータは検出器の情報を持ち、粒子がどの検出器で検出されたかを与える。フルシミュレータに対してファストシミュレータというものがある。ファストシミュレータは単に粒子の通過した座標を返す。

本研究では Geant4 を使ったフレームワークである FADS/Goofy を用いて現実の Atlas 実験の検出器の情報を持ったシミュレータを構成しシミュレーションを行った。

目 次

1 Atlas 実験とシリコン検出器 **1**

1.1 LHC (加速器) 1

1.2 Atlas, SCT (測定器) 2

2 アトラス実験で期待される物理 **4**

2.1 ヒッグス粒子 4

2.2 超対称性理論 5

3 Geant4 と FADS/Goofy を使った測定器シミュレータ **6**

3.1 PYTHIA を用いた SCT シミュレーション 11

3.1.1 マクロファイル 12

3.1.2 シミュレーションの結果 16

3.2 ディジタイゼイション 18

3.3 座標変換 19

3.4 並進と回転の統一的な取り扱い 21

4 まとめ **23**

1　Atlas 実験とシリコン検出器

1.1　LHC (加速器)

LHC は Large Hadron Collider の略称で、大型陽子陽子衝突型加速器を意味する。スイスのジュネーブ近郊にある CERN の地下トンネル内に建設中である。地下トンネルは周長約 27km で、地下 100m の深さにある。1989 年から 2000 年まで実験が行われていた大型電子陽電子衝突型加速器 (LEP) のトンネルをそのまま使っている。トンネル内部には多数の超伝導磁石を用いたシンクロトロン加速器が設置されている。エネルギーを与え 7TeV まで加速した 2 つの陽子 (重心系で 14TeV) を衝突させて反応を測定する。衝突点では陽子・陽子衝突が毎秒十億回起こる。1 回の衝突あたり約 100 個の二次粒子が発生する。超伝導電磁石はトンネル内に 1,232 台配置される。一つの超伝導電磁石は長さが 14.2m で、超流動ヘリウムにより絶対温度 1.9K に冷やされる。12kA の電流を流すことで 8.4T の磁場をリング全周に作る。LHC は 2007 年の完成に向けて作業が進められている。

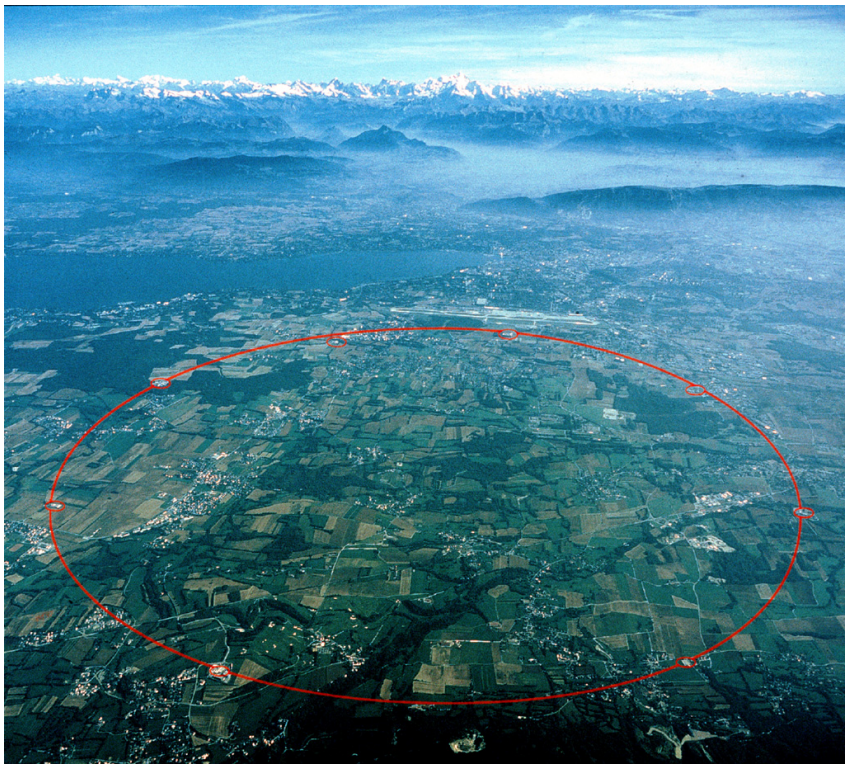

図 1: 赤い円で示された場所の地下にトンネルが存在する

1.2 Atlas, SCT (測定器)

アトラス (Atlas = A Toroidal LHC Apparatus) 実験はヨーロッパを中心に 35ヵ国の参加する国際共同実験である。参加する研究者は 1,500 人を越える。Atlas 測定器は、LHC の高ルミノシティ運転時の高い粒子頻度のもとでも、多くのシグナルをバランス良く取り出し、精度の良い測定を行なうことを目的とした汎用測定器であり、LHC の性能を充分に発揮させることが出来る。測定器は多数の検出器で構成される大型の実験装置で、LHC の中で陽子が衝突する場所にある実験場に配置される。LHC にはアトラスと CMT (Compact Muon Spectrometer) という 2 つの陽子陽子衝突の測定器が設置される。測定器の中心軸に加速器ビームパイプが通り、衝突点を中心として検出器が配置されている。多数の検出器で 2 次粒子のほぼ全てを観測する。測定器は半径約 12m, 全長約 44m の円筒形で、重量は約 7,000 トンに及ぶ。

具体的には以下のような種類の検出器がある。

- インナー・ディテクタ (内部飛跡検出器)
- 電磁カロリメータ
- ハドロン・カロリメータ
- ミューオン・ディテクタ (μ 粒子検出器)

ビームパイプをとりまくように配置されているインナー・ディテクタは荷電粒子の飛跡認識を目的とした検出器で、内側から順にピクセル型検出器、半導体飛跡検出器 (SCT)、遷移輻射飛跡検出器が置かれる。

これらの検出器は、超伝導ソレノイド磁石による磁場中に置かれており、荷電粒子の飛跡の曲率から粒子のエネルギーを求めることが可能である。

電磁カロリメータは、電子・γ 線のエネルギーと位置を測定する。耐放射線に優れ、安定性のよい液体アルゴン・カロリメータが使用される。

ハドロン・カロリメータは、陽子やπ 中間子等のハドロンのエネルギーを測定する。前後方は液体アルゴン・カロリメータ、バレル部はタイル・カロリメータが使用される。

μ 粒子はカロリメータの中で反応することなく外まで突き抜けて来るため、μ 粒子検出器は測定器の一番外側に設置される。μ 粒子検出器は、精密位置測定用のドリフト・チュー

ブと、反応の種類を見極めるためのトリガー・チェンバーから成る。さらに、超伝導トロイド磁石と組み合わせて、μ 粒子の運動量を精密に測定する。

内部飛跡検出器の一つ SCT (Silicon Microstrip Semiconductor Tracker = シリコンマイクロストリップ半導体飛跡検出器) は 4 層からなる円筒状のバレル部と前後方各 9 枚からなる円盤状の前後方部からなる。バレル SCT のうち、約 700 モジュールをアトラス日本グループが量産する。

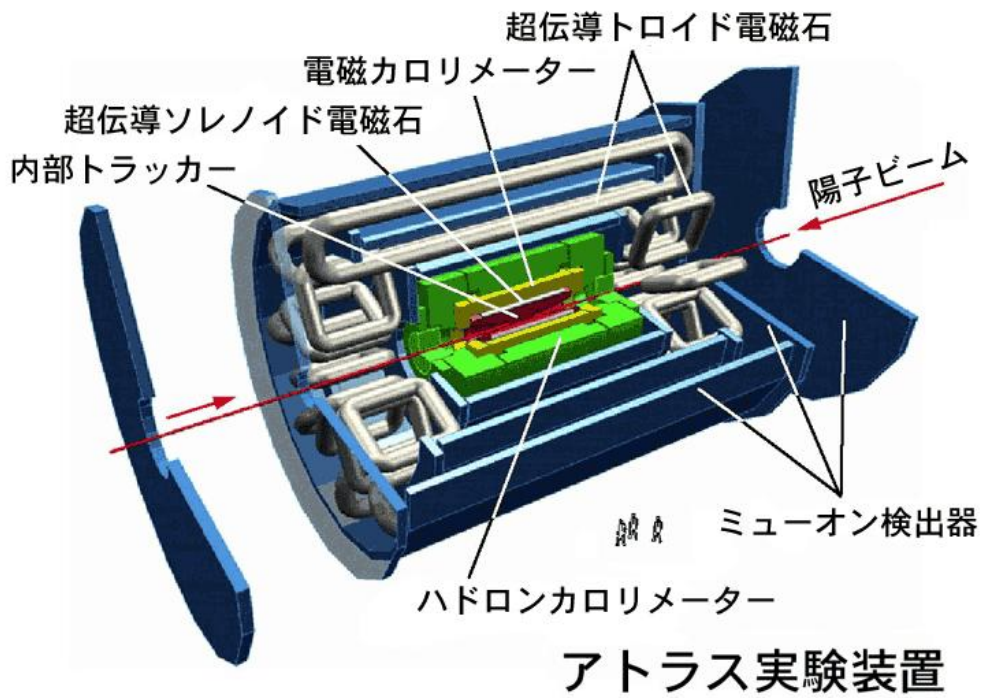

図 2: アトラス測定器の概観図

2　アトラス実験で期待される物理

2.1　ヒッグス粒子

LHC 計画のなかでヒッグス粒子の発見が特に大きな目標とされている。

この宇宙で働く基本的な 4 つの力のうち、電磁気力は距離の自乗に反比例する、無限遠まで届く力である。それに対して粒子の崩壊に関係する「弱い力」は 10^{-16}cm というごく近距離でしか作用しない。これは、電磁気力を媒介する光子が質量を全く持たないのに対し、弱い力を媒介する W / Z ボゾンは陽子の約 100 倍という大きな質量を持つことが原因である。しかし、基本的な力を媒介する粒子 (これらはゲージ粒子と呼ばれる) には、光子の様に質量がゼロでなくてはならないという規則がある。物質を構成する粒子がゲージ粒子を交換する際に、全体の対称性が保たれなくてはならないからである。

このことの説明として、ヒッグス場というものが考え出された。W ボゾンや Z ボゾンは真空中に存在するヒッグス場との相互作用によって抵抗を生じる。そのため、本来質量を持たない筈のゲージ粒子も質量があるものとして観測されうる。

現在最も基本的な粒子と考えられているのは物質をかたちづくるクォーク・レプトンとゲージ粒子、それにヒッグス場の量子化したヒッグス粒子で、その理論は標準模型と呼ばれる。これまでに標準模型と矛盾する実験的事実は見付かっておらず、ヒッグス粒子を除

物質粒子	第 1 世代	第 2 世代	第 3 世代
クォーク	$\begin{pmatrix} u \\ d \end{pmatrix}$	$\begin{pmatrix} c \\ s \end{pmatrix}$	$\begin{pmatrix} t \\ b \end{pmatrix}$
レプトン	$\begin{pmatrix} \nu_e \\ e \end{pmatrix}$	$\begin{pmatrix} \nu_\mu \\ \mu \end{pmatrix}$	$\begin{pmatrix} \nu_\tau \\ \tau \end{pmatrix}$
ゲージ粒子	γ, g, $W^{\pm}$, Z^0		
ヒッグス粒子	ヒッグス粒子		

表 1: 標準模型の基本粒子の分類

く粒子は既に発見されている。

標準模型では、ヒッグス粒子は 1TeV 以上の質量は持たないと考えられている。真空が安定した状態にあるためには、ヒッグス場があまり大きなエネルギーを持つことは出来ない。LHC はこのエネルギーの範囲をカバーしており、ヒッグス粒子を発見できる可能性が高いと言われている。

ヒッグス粒子が LHC で到達可能なエネルギー領域に存在するならば、1 年間の実験により約 1 ~ 10 万個のヒッグス粒子が衝突点で生成されると期待されている。

2.2 超対称性理論

Atlas 実験ではヒッグス粒子の他に超対称粒子や現在知られていない全く新しい相互作用などの発見も期待されている。

超対称性理論は、標準模型を越えた延長線上にある理論である。標準模型の基本粒子は半整数のスピンを持つフェルミオンであるクォーク、レプトンと整数のスピンを持つボゾンであるゲージボゾン、ヒッグス粒子とに分けられる。超対称性はフェルミオンとボゾンを関係づける対称性である。自然界が超対称であれば、全ての基本粒子は等しい内部量子数を持つフェルミオンとボゾンがペアで存在する。標準模型の基本粒子とペアをなすパートナーの粒子を超対称粒子と呼ぶ。

3　Geant4 と FADS/Goofy を使った測定器シミュレータ

Geant4 (GEometry ANd Transport) は高エネルギー物理のシミュレーションのためのツールキットである。Geant4 を土台に作られるアプリケーションであるシミュレータは素粒子と物質との相互作用をシミュレートする。Geant4 の前身の GEANT3 は FORTRAN で書かれていたが、ソフトウェアの大規模化・複雑化に伴い、開発の維持が困難を来すようになった。GEANT3 のこの問題を解決するため、オブジェクト指向技術を用いて書き直したものが Geant4 である。これによって巨大なプログラムの保守性が高められた。

本研究では Geant4 を直接利用するのではなく、FADS/Goofy[1] というフレームワークを用いてより効率的に Geant4 の機能を使ってシミュレーションの研究を行えるようにした。

FADS/Goofy にはあらかじめアプリケーションプログラムのメインモジュールが用意されている。goofy がそれである。Geant4 では必要なモジュールを組み込んだメインプログラムを書くのはアプリケーションプログラマの仕事であった。しかしメインプログラムの行う処理はどのアプリケーションプログラムも大部分共通している。goofy はマクロ言語によってプラグインモジュールを実行時に読み込むという指示を受け取ることが出来る。FADS/Goofy のこの機能により、エンドユーザレベルでの追加機能の組み込みが可能となり、アプリケーションプログラマはプラグインモジュールの開発に専念することが出来る。

FADS/Goofy のプラグインモジュール構造は、共有ライブラリの動的ローディングとオブジェクト指向言語の実行時バインディング機能を用いて実現されている。

[1] Framework for Autonomous Detector Simulation / Geant4-based Object Oriented Folly

load は、 dlopen (3) の man page に記されている動的ローディングの例を書き換えて、プログラム自体がコンパイルされた段階では名前を知らないライブラリの中の関数を呼び出せるということを示したものである。ここでは libm.so から 1 引数関数 cos と sin, 2 引数関数 pow を呼び出している。プログラム本体にはライブラリ及び関数の名前は全く書かれていないが、コマンドラインで指定した共有ライブラリの関数を呼び出すことが出来る。

```
% cat load.c
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>

void* dl_open (char* r);
void* dl_sym (void* handle, char* s);

int
main (int argc, char** argv)
{
        void *handle;
        double (*dl_func) (double);
        double (*dl_func2) (double, double);

        char* r;
        char* s;
        double x, y;

        if (argc < 4) {
                puts ("load: usage: load lib func arg");
                exit (1);
        }

        r = argv[1];
```

```
        s = argv[2];

        handle = dl_open (r);

        if (argc == 4) {
                x = atof (argv[3]);
                printf ("%s (%.2f) = ", s, x);

                dl_func = dl_sym (handle, s);

                printf ("%f\n", (*dl_func) (x));
        }
        else if (argc == 5) {
                x = atof (argv[3]);
                y = atof (argv[4]);
                printf ("%s (%.2f, %.2f) = ", s, x, y);

                dl_func2 = dl_sym (handle, s);

                printf ("%f\n", (*dl_func2) (x, y));
        }

        dlclose (handle);
        return 0;
}

void*
dl_open (char* r)
{
        void* handle;
```

```
        char b[80];

        snprintf (b, sizeof b, "lib%s.so", r);
        handle = dlopen (b, RTLD_LAZY);
        if (!handle) {
                fprintf (stderr, "%s\n", dlerror());
                exit (1);
        }
        return handle;
}

void*
dl_sym (void* handle, char* s)
{
        void* func;
        char* error;

        func = dlsym (handle, s);
        if ((error = dlerror()) != NULL) {
                fprintf (stderr, "%s\n", error);
                exit (1);
        }
        return func;
}
% cat Makefile
load: load.c
        gcc -rdynamic -ldl -o load load.c
% ./load m cos 0
cos (0.00) = 1.000000
% ./load m cos 3.14159265
```

```
cos (3.14) = -1.000000
% ./load m sin 1.57079633
sin (1.57) = 1.000000
% ./load m pow 2 10
pow (2.00, 10.00) = 1024.000000
```

3.1 PYTHIA を用いた SCT シミュレーション

PYTHIA は FORTRAN で書かれたイベントジェネレータである。イベントジェネレータはシミュレータの中で発生する物理プロセスに関する情報を与える。

- PYTHIA Generator
- 磁場
- SCT Geometry

以上のプラグインモジュールを組み合わせることにより物理学的に現実的なシミュレーションを行うことが出来る。これらのプラグインモジュールはあらかじめ用意されたプログラムコードをそれぞれ独立にコンパイルして作ったものである。マクロ命令によって実行中の goofy にプラグインモジュールを組み込む。

setup.sh などの必要な環境設定を読みこんだ後、

```
% goofy
```

で Goofy を起動する。Goofy のプロンプトに対して

```
/control/execute PySCTmag.mac
```

と入力して以下に示すマクロファイルを読みこむ。マクロファイルの実行が完了した後、

```
/run/beamOn 1
```

と入力するとシミュレーションの実行が開始される。

3.1.1 マクロファイル

PySCTmag.mac

これは FADS/Goofy チュートリアル [3] の 'Tutorial - 2: Pythia generator' の pythiaGenerator.mac を書き換えたものである。

PYTHIA の設定はチュートリアルで使用されているマクロファイルをそのままのかたちで再利用した。/control/execute mypythiaSetup.mac という行で外部のマクロファイルから PYTHIA の設定を読みこむ。

磁場の中の SCT をシミュレートするための命令が書き加えられている。

```
/load SCT
/load fadsField:MagField-1
```

の 2 つの行によりプラグインモジュールが組み込まれる。

このマクロファイルではデフォルトで OpenGL により結果を画面出力する。後から視点などのパラメータを与えて可視化を行える様に結果をファイルに保存するためには、/vis/open OGLIX という行を /vis/open DAWNFILE に書き換えるか、マクロファイル実行後に goofy のプロンプトに/vis/open DAWNFILE と入力する。

```
/control/ReadXML material.xml
/load dbORB
/load SCT
/Geometry/GetFacility SCTGeometry SCT
/SCT/SetAsWorld

/control/execute mypythiaSetup.mac

### Set up Geant4 physics list
#/Physics/GetPhysicsList Fast_Physics
#/Physics/GetPhysicsList Atlas_Physics
/Physics/GetPhysicsList EM_Physics
```

```
### Set up visualization
/control/ReadXML color.xml
/vis/Volume/SetInvisible SCT SCTBarrel
/vis/Volume/SetInvisible SCT SCTSki
/vis/Volume/SetInvisible SCT SCTCounter
/vis/Volume/SetInvisible SCT SCTLayer
/vis/Volume/SetColor SCT SCTWafer tomato3
/vis/Volume/SetColor SCT SCTHybrid violet
/vis/Volume/SetColor SCT SCTBaseBoard wheat3
/vis/Volume/SetColor SCT SCTLayerSupport yellow2

/vis/open OGLIX
#/vis/open DAWNFILE
/vis/scene/create
/vis/viewer/zoomTo 1.0
/vis/viewer/set/viewpointThetaPhi 70 30 deg

### Set up run and tracking
/run/initialize
/tracking/verbose 1
/tracking/storeTrajectory 1

/vis/drawVolume

### Add trajectories
/vis/scene/add/trajectories

### Magnetic Field
/load fadsField:MagField-1
/MagneticField/Select AtlasField
```

```
/MagneticField/Initialize
```

mypythiaSetup.mac

これは FADS/Goofy チュートリアルの 'Tutorial - 2: Pythia generator' の pythiaSetup.mac と同じ内容である (PYTHIA のバージョンを示す数字のみが異なる)。

```
###  Set up Pythia generator
/load pythia6210
/load Pythia6
/Generator/Select Pythia-6.1

### Set Pythia parameters
/Generator/Pythia6/SetParameter MSEL=0
/Generator/Pythia6/SetParameter MSUB(102)=1
/Generator/Pythia6/SetParameter MSUB(123)=1
/Generator/Pythia6/SetParameter MSUB(124)=1

### Initialize generator
/Generator/Initialize
```

3.1.2　シミュレーションの結果

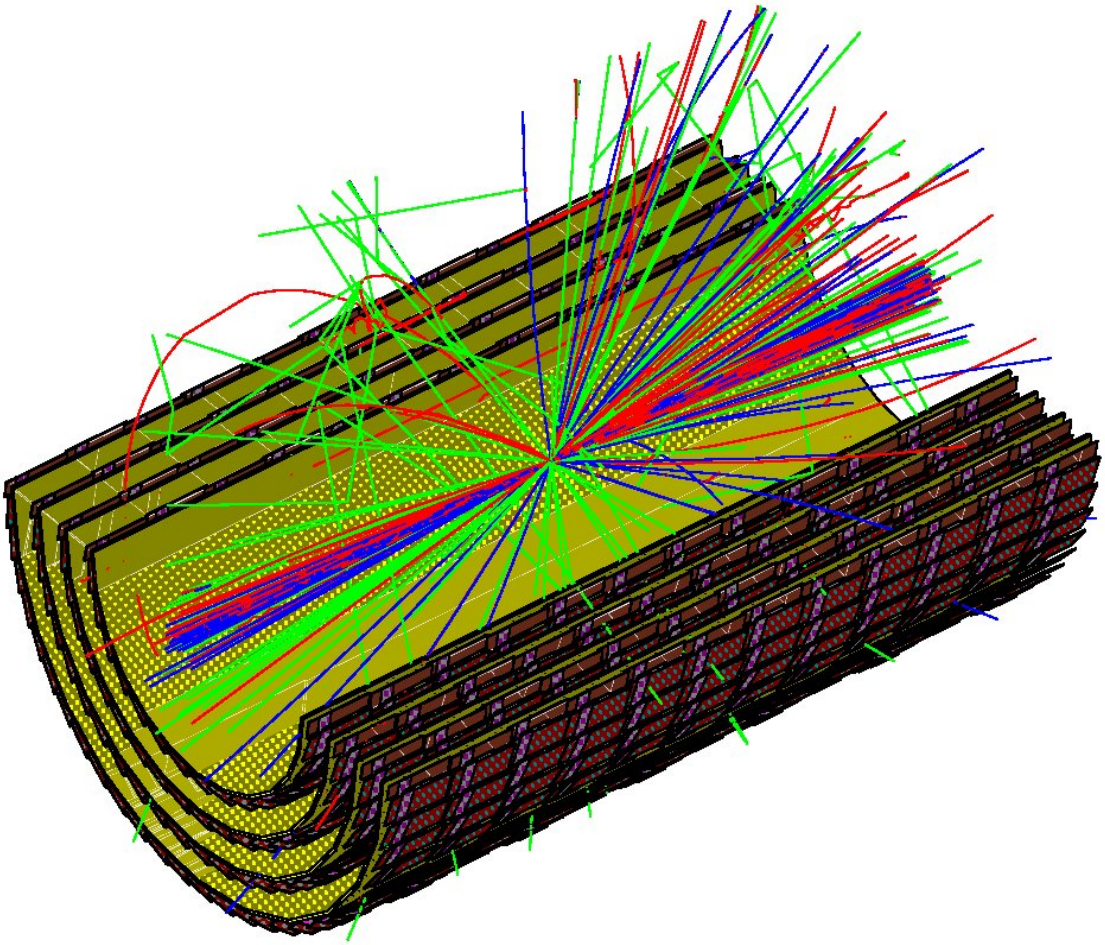

図 3: 測定器と磁場のもとで相互作用した粒子の飛跡

磁場の影響で大きく弧を描く赤い飛跡が確認できる。

今回のシミュレーションでは DAWN というソフトウェアで結果の画像を生成した。DAWN は隠線・隠面処理を行うため、SCT 内部の粒子の飛跡を観察するためには、一部の検出器を非表示にする必要がある。そのためのツールが DAWNCUT である。

DAWNCUT をコンパイルする。DAWNCUT の web page [7] からダウンロードし、展開した後 make する。configure スクリプトファイルはない。g++/egcs コンパイラを使うことが推奨されている。make したディレクトリに dawncut が出来るので PATH の通った

適当なディレクトリに置く。

```
% tar xzf dawncut_1_51a.taz
% cd dawncut_1_51a
% make
% cp dawncut ~/bin
```

```
% dawncut g4_06.prim g4_06-cut.prim
```

'dawncut g4_06.prim g4_06-cut.prim' は

```
dawncut 0 1 0 0 g4_06.prim g4_06-cut.prim
```

と等価で、平面 $\mathrm{y} = 0$ で切り取る。切り取られるのは平面の手前側である。

出力された g4_06-cut.prim を DAWN で処理することで EPS ファイルが得られる。

```
% dawn g4_06-cut.prim
```

3.2 ディジタイゼイション

SCT では粒子の飛跡を割り出すことが大事である。飛跡の曲率から粒子の持つエネルギーが算出されるからである。検出器は何層もの階層構造のなかに配置されており、階層構造の中ではマザーボリュームと呼ばれる一つ上の階層に対して相対的な位置関係が決められている。バレルを例に取ると、まず半径の異なる円筒状の階層が 4 層ある。それぞれの層はスキーと呼ばれる多数の板状の部分からなる。1 枚のスキーには 12 個のモジュールが入り、モジュールには 2 枚のウェファが載るという構造になっている。

粒子の飛跡を求めるには、個々の最小単位の検出器での反応の位置をある一つの座標系に変換しなければならない。この座標系はグローバル座標系と呼ばれ、トップレベルのボリュームがこのグローバル座標系に配置される。それに対してマザーボリュームとの位置関係はマザーボリューム内のローカル座標という。

3.3 座標変換

『素粒子物理学』[1, pp. 54–61] に、空間の座標変換についての節がある。この中で、座標系の回転の行列表現について述べた部分がある。

$x-y$ 平面内の点 (x,y) は縦ベクトルで表される。

$$\vec{x} = \begin{pmatrix} x \\ y \end{pmatrix} = (x,y)^T$$

添字 T は行列としての転置を表す。系の連続的な回転により、点 $\vec{x}$ が別の点 $\vec{x}'$ へ移ることを、回転の演算子 R_θ を用いて、

$$\vec{x} \to \vec{x}' = R_\theta \vec{x}$$

と書く。系を角度 θ だけ回転する代わりに、系を固定して座標系を $-\theta$ だけ回転しても同じ結果が得られる。この座標変換は成分では次のように書かれる。

$$x' = x\cos\theta - y\sin\theta, \qquad y' = x\sin\theta + y\cos\theta$$

ベクトル形式では、座標系の回転は実数の 2×2 行列を用いて、

$$\vec{x}' = A(\theta)\vec{x}, \quad A(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

と表される。

2 次元平面の座標系の回転は 3 次元空間の座標系の回転へ拡張できる。3 次元空間内の点 P は 3 成分ベクトルで表される。

$$\vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

座標系の回転により座標 $\vec{x}$ が別の座標 $\vec{x}'$ へ移る作用は 3×3 の実数行列 A を用いて表される。

$$\vec{x} \to \vec{x}' = A\vec{x} \quad \text{あるいは成分で、} \quad x_i' = \sum_{j=1}^{3} A_{ij}x_j$$

任意の 3 次元回転は x 軸、 y 軸、z 軸の周りの回転を合成して得られることが知られている。z 軸の周りの回転は $x-y$ 面上の回転に等しく、 z 座標は

変わらない。よって、z 軸の周りの角度 θ_3 の回転: $\vec{x}' = A_z(\theta_3)\vec{x}$ は次の形となる。

$$A_z(\theta_3) = \begin{pmatrix} \cos\theta_3 & -\sin\theta_3 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

x 軸の周りの θ_1 回転と y 軸の周りの θ_2 回転は $y-z$ 面上の回転と $z-x$ 面上の回転に等しい。上と同様に、それらは次の形になる。

$$A_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & -\sin\theta_1 \\ 0 & \sin\theta_1 & \cos\theta_1 \end{pmatrix}, \quad A_y = \begin{pmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ 0 & 1 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 \end{pmatrix}$$

2 次元の位置ベクトル $\vec{x}$ を $\vec{x}'$ に移す並進は水平及び垂直方向の単位ベクトル $\vec{\imath}, \vec{\jmath}$ を用いて

$$\vec{x} \to \vec{x}' = \vec{x} + k\vec{\imath} + l\vec{\jmath} = T_{k,l}\vec{x}$$

と表される。$T_{k,l}$ が並進の演算子である。$T_{k,l}$ は 3 次元空間上の線形変換ではないので、行列によって表現することは出来ない。

3.4 並進と回転の統一的な取り扱い

OpenGL Programming Guide [2] には、3 次元座標の変換に 4×4 行列を用いる方法が示されている。

3 次元空間の点 P (x, y, z) に対応するベクトルとして、4 次元のベクトル

$$\vec{v} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

を考える。このとき、点 P を点 Q $(x + \Delta x, y + \Delta y, z + \Delta z)$ に移す並進 T は次の様に行列表現できる。

$$T = \begin{pmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Q に対応するベクトル $\vec{v}'$ は

$$\vec{v}' = T\vec{v} = \begin{pmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + \Delta x \\ y + \Delta y \\ z + \Delta z \\ 1 \end{pmatrix}$$

である。また、2 つの並進 T_1 と T_2 の合成 $T_1 \circ T_2$ は行列の積で表される。

$$T_1 = \begin{pmatrix} 1 & 0 & 0 & x_1 \\ 0 & 1 & 0 & y_1 \\ 0 & 0 & 1 & z_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad T_2 = \begin{pmatrix} 1 & 0 & 0 & x_2 \\ 0 & 1 & 0 & y_2 \\ 0 & 0 & 1 & z_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

とすると

$$T_1 T_2 = \begin{pmatrix} 1 & 0 & 0 & x_1 \\ 0 & 1 & 0 & y_1 \\ 0 & 0 & 1 & z_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & x_2 \\ 0 & 1 & 0 & y_2 \\ 0 & 0 & 1 & z_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & x_1 + x_2 \\ 0 & 1 & 0 & y_1 + y_2 \\ 0 & 0 & 1 & z_1 + z_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

一方、回転も 4 次元ベクトルを扱うために 4×4 の行列で表現される。z 軸の周りの角度 θ_3 の回転 $R_z(\theta_3)$ は

$$R_z(\theta_3) = \begin{pmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

同様に、

$$R_x(\theta_1) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_1 & -\sin\theta_1 & 0 \\ 0 & \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad R_y(\theta_2) = \begin{pmatrix} \cos\theta_2 & 0 & \sin\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

3 次元座標の変換に用いる 4×4 行列は並進と回転に関する限り 4 行目が常に $\begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}$ である。また、3 次元空間上の点を表す 4 次元ベクトルは 4 番目の成分が常に 1 である。従って記憶上では変換行列は 3×4 個、空間上の点は 3 個の実数で表せる。

4 次元ベクトルを導入したことによって 3 次元の線形変換では表すことが出来なかった並進を回転と等しい型の行列で表現することが出来る。これにより、全ての座標変換は表現行列の積のかたちで合成される。一個の行列に対してはその逆行列が機械的な計算によって得られる。このようにして Geant4 の中ではヒストリという機能でローカル–グローバル座標変換の演算子を求めることが出来る。

4　まとめ

今回の研究で、FADS/Goofy 及び Geant4 を利用した様々なプログラムを通じて、大規模なシミュレータの開発に使用されるオブジェクト指向技術に触れた。ツールキットとしての Geant4 の膨大なコードについては細かい点を把握することは出来なかったが、FADS/Goofy を利用して既存のコードを組み合わせ、磁場の中の SCT という現実の測定器の情報を与えてシミュレーションを実行し、結果を視覚的に観察した。

共有ライブラリから関数を実行時に動的に呼び出す短いプログラムを作成し、FADS/Goofy の特徴である動的ローディングに関する理解を深めた。

また、粒子の飛跡を求めるために必要なローカル–グローバル座標変換の演算子を求める際に、4 行 4 列の行列を用いることにより連続する座標変換は行列の積で表現されることがわかった。

謝辞

参考文献

[1] 原 康夫・稲見武夫・青木健一郎『素粒子物理学』朝倉書店, 2000.

[2] OpenGL Architecture Review Board Jackie Neide/Tom Davis/Mason Woo
OpenGL Programming Guide (日本語版). アジソン・ウェスレイ, 1993.

[3] FADS/Goofy Tutorial ~ Basic Course ~
`http://gfarm.kek.jp/FADS/tutorial/basic/index.html`

[4] The Geant4 Japan Group `http://geant4.kek.jp/GEANT4/`

[5] Geant4 `http://wwwinfo.cern.ch/asd/geant4/geant4.html`

[6] DAWN `http://geant4.kek.jp/~tanaka/DAWN/About_DAWN.html`

[7] DAWNCUT `http://geant4.kek.jp/~tanaka/DAWN/About_DAWNCUT.html`

[8] ATLAS Detector Simulation - Official Site -
http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/domains/simulation/