コライダーの物理とツール ROOT 編

KEK　野尻

前回までの復習

- Pythia 8 で素粒子のevent が作れるということがわかった。 example/main14.cc -> 好きなモデルのイベントを作るのは第三回
- Pythia8 のイベントを hepmc file にすることができる (main41.cc)
- hepmc ファイルから Delphes が root file を作ってくれることがわかった。
  - > **DelphesHepMC ***.tcl test.root pythia8.hepmc**
- Delphes の作るファイルにはdetector の効果が入っている。
- root test.root のあと TBrowser b; とすると、画面がでてくる。クリックするとなんと分布図までででてくるようだ。

hepmc file の中身をみてみよう

event の最初

```
N 1 "0"
U GEV MM
C 1.7478938039142924e+00 2.3317098725716918e-02
F 21 21 5.102144553648e-01 2.60781124640e-01 1.17330357828e+03…
```

F - PdfInfo information

- **int**: *flavour code of first parton*
- **int**: *flavour code of second parton*
- **double**: *fraction of beam momentum carried by first parton*
- **double**: *fraction of beam momentum carried by second parton*
- **double**: *Q-scale used in evaluation of PDF's (in GeV)*
- **double**: *x*f(x) for id1, x1, Q*
- **double**: *x*f(x) for id2, x2, Q*
- **int**: *LHAPDF set id of first parton (zero by default)*
- **int**: *LHAPDF set id of second parton (zero by default)*

P - GenParticle information

- **int**: *barcode*
- **int**: *PDG id*
- **double**: *px*
- **double**: *py*
- **double**: *pz*
- **double**: *energy*
- **double**: *generated mass*
- **int**: *status code*
- **double**: *Polarization theta*
- **double**: *Polarization phi*
- **int**: *barcode for vertex that has this particle as an incoming particle*
- **int**: *number of entries in flow list (may be zero)*
- **int, int**: *optional code_index and code for each entry in the flow list*

V - GenVertex information

- **int**: *barcode*
- **int**: *id*
- **double**: *x*
- **double**: *y*
- **double**: *z*
- **double**: *ctau*
- **int**: *number of "orphan" incoming particles*
- **int**: *number of outgoing particles*
- **int**: *number of entries in weight list (may be zero)*
- **double**: *optional list of weights*

PDG id

QUARKS

d	1
u	2
s	3
c	4
b	5
t	6
b'	7
t'	8

LEPTONS

e^-	11
ν_e	12
μ^-	13
ν_μ	14
τ^-	15
ν_τ	16
τ'^-	17
$\nu_{\tau'}$	18

GAUGE AND HIGGS BOSONS

g	(9) 21
γ	22
Z^0	23
W^+	24
h^0/H_1^0	25
Z'/Z_2^0	32
Z''/Z_3^0	33
W'/W_2^+	34
H^0/H_2^0	35
A^0/H_3^0	36
H^+	37

decay を追いかける

gluino の対生成

```
V -230 0 0 0 0 0 0 1 0
P 307 1000021 -1.5620e+02 1.7511e+02 1.1932e+03 1.6798e+03 1.1588e+03 62 0 0 -306 2 1 104 2 101
V -231 0 0 0 0 0 0 1 0
P 308 1000021 2.2237e+02 -1.7124e+01 -3.2024e+02 1.2224e+03 1.1584e+03 62 0 0 -307 2 1 103 2 109
```

gluino(308 番) が stop が top に崩壊

103 109はカラーの流れ

```
V -307 0 2.0458e-15 -1.5754e-16 -2.9462e-15 1.1246e-14 0 2 0
P 390 1000006 -1.5267e+02 -2.2329e+02 -3.8035e+02 7.5667e+02 5.9557e+02 22 0 0 -309 1 1 103
P 391 -6 3.7505e+02 2.061e+02 6.011e+01 4.657e+02 1.7360e+02 22 0 0 -308 1 2 109
```

……..

top 作業中

```
V -311 0 2.0458e-15 -1.5754e-16 -2.946e-15 1.12463e-14 0 1 0
P 397 -6 3.325e+02 2.0283e+02 4.6956e+01 4.2901e+02 1.7360e+02 52 0 0 -313 1 2 202
```

top 作業中

```
V -313 0 2.045e-15 -1.575e-16 -2.946e-15 1.1246e-14 0 1 0
P 400 -6 3.3324e+02 2.0162e+02 4.7680e+01 4.2908e+02 1.736e+02 52 0 0 -331 1 2 202
```

top 壊れた

```
V -331 0 2.0458e-15 -1.575e-16 -2.9462e-15 1.1246e-14 0 2 0
P 428 -24 2.995e+02 1.3153e+02 -1.7870e+00 3.3678e+02 8.003e+01 22 0 0 -357 0
P 429 -5 3.3724e+01 7.0089e+01 4.9467e+01 9.2303e+01 4.7999e+00 23 0 0 -531 1 2 202
```

できた root file で遊ぶ

- **Example directory の下の　Example1.C**

動かし方 $root -l examples/Example1.C'("delphes_output.root")'

(こうすると あたかもC++ のようにコマンドを解釈して計算をしてくれる. cint という)

```
void Example1(const char *inputFile)
{
 gSystem->Load("libDelphes");
  // Create chain of root trees

 TChain chain("Delphes");
 chain.Add(inputFile);

 ExRootTreeReader *treeReader = new ExRootTreeReader(&chain);
 Long64_t numberOfEntries = treeReader->GetEntries();

 // Get pointers to branches used in this analysis
 TClonesArray *branchJet = treeReader->UseBranch("Jet");
 TClonesArray *branchElectron = treeReader->UseBranch("Electron");
```

ライブラリ読み込み

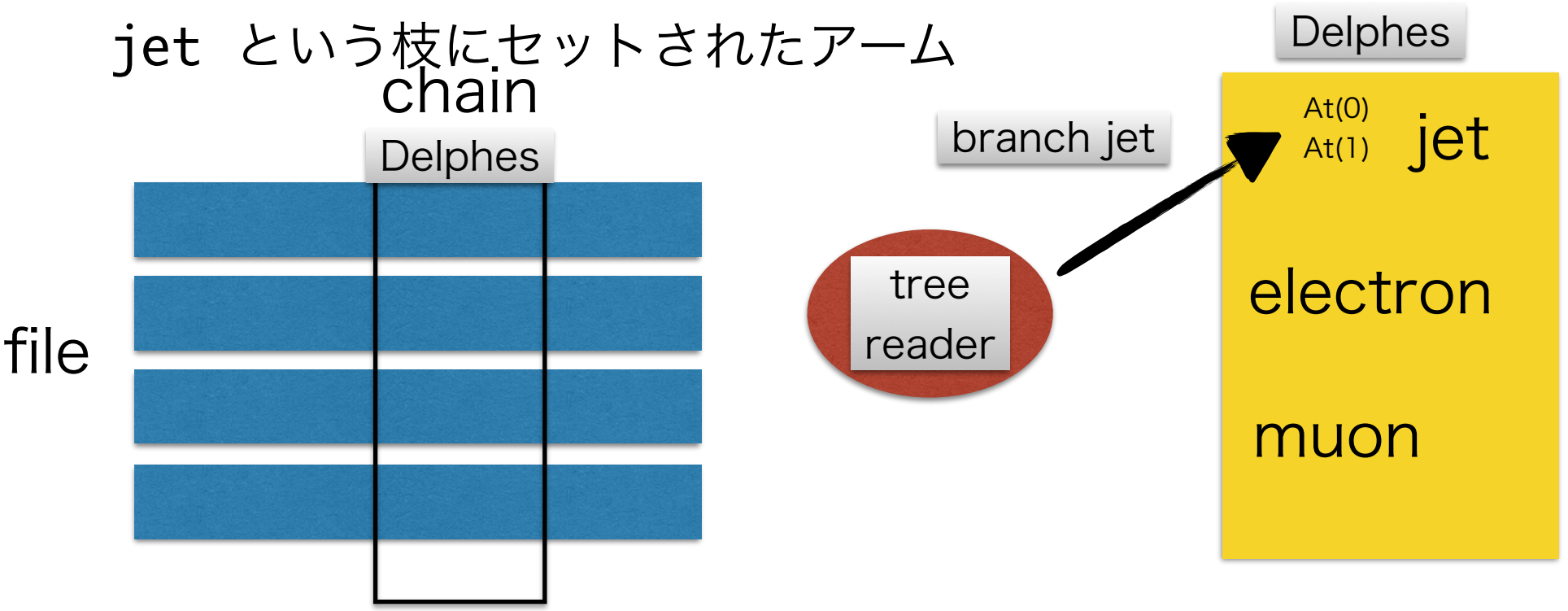
root file の中にある Delphes の tree を利用

使うファイルを指定 (ファイルをいくつも付け足せる)

event 数ゲット

branch の割り付け

- TChain chain :たくさんのファイルを一つとみなして“Delphes” という tree の中身にアクセスするためのアーム
- ExRootTreeReader *treeReader event にアクセスして、tree の中身の枝にアクセスして “jet”, “electron” などを をもらってくるためのアーム
- TClonesArray event の*branchJet tree の中の jet という枝にセットされたアーム



何がおこっているか

- root file の中には、データが隙間なく詰まっている。hepmc-> root でファイルサイズ は 1/3 くらいに
- メモリーのなかで、event -> jet -> jet の運動量など必要な量に的確にアクセスできる機能を rootのライブラリ が提供している。要するにアドレス指定機能。
- そして root のライブラリーは、root file をその処方に従って作ってくれる。

```
// Loop over all events
  for(Int_t entry = 0; entry < numberOfEntries; ++entry)
  {
    // Load selected branches with data from specified event
    treeReader->ReadEntry(entry);

    // If event contains at least 1 jet
    if(branchJet->GetEntries() > 0)
    {
      // Take first jet
      Jet *jet = (Jet*) branchJet->At(0);

      // Plot jet transverse momentum
      histJetPT->Fill(jet->PT);

      // Print jet transverse momentum
      cout << "Jet pt: "<<jet->PT << endl;
}
```

event でloop する

イベント読み込み: ここで branchJet 等の中身が書きかわる

jet の数が０でないとして

highest pT jet　をとる

画面にも数字を出す

図を表示

```
 // Show resulting histograms
  histJetPT->Draw();
  histMass->Draw();
```

グラフをかく

まず何かを始める前に図を定義する。

```
// Book histograms
 TH1 *histJetPT = new TH1F("jet_pt", "jet P_{T}", 100, 0.0,
100.0);
 TH1 *histMass = new TH1F("mass", "M_{inv}(e_{1}, e_{2})",
100, 40.0, 140.0);
```

event loop の中で、情報をヒストグラムに詰める

```
      // Plot jet transverse momentum
      histJetPT->Fill(jet->PT);
```

event loop が終わったら、図を画面に表示

```
 // Show resulting histograms
  histJetPT->Draw();
```

root のclassを利用する

(root -l hogehoge.c で使うときは root の class のheader は読み込まなくてよい)

```
  TLorentzVector EMom0, EMom1;   ベクトル
  TLorentzVector MuMom0, MuMom1;
 for(Int_t i=0; i<nmu; ++i){
        mu=  (Muon*) branchMuon->At(i);
        if(i==0)
          {MuMom0=mu->P4(); Delphes のMuon class からベクトルを読み出す関数 P4
            xptm0=MuMom0.Pt();           Root のベクトルのメンバー関数Pt()
            plots->fPTm0->Fill(mu->PT);  Muon class の  メンバ関数 PT
          }
…
…
    }
    TLorentzVector dummy
      if(nmu>1)
         {
           dummy=MuMom0+MuMom1; ベクトルの足し算も定義されている. Muon は足せない。
           xMmumu=dummy.M();      ベクトルのメンバー関数 M()
           plots->fMmumu->Fill(xMmumu);
         }
```

Root のクラス　TLorenzVector

Public Types

enum	{ kX =0, kY =1, kZ =2, kT =3, kNUM_COORDINATES =4, kSIZE =kNUM_COORDINATES }
typedef Double_t	Scalar

▸ **Public Types inherited from TObject**

Public Member Functions

	TLorentzVector ()
	TLorentzVector (Double_t x, Double_t y, Double_t z, Double_t t)
	TLorentzVector (const Double_t *carray)
	TLorentzVector (const Float_t *carray)
	TLorentzVector (const TVector3 &vector3, Double_t t)
	TLorentzVector (const TLorentzVector &lorentzvector)
virtual	~TLorentzVector ()
Double_t	X () const
Double_t	Y () const
Double_t	Z () const
Double_t	T () const

Double_t	DeltaPhi (const TLorentzVector &
Double_t	DeltaR (const TLorentzVector &) c
Double_t	DrEtaPhi (const TLorentzVector &
TVector2	EtaPhiVector ()
Double_t	Angle (const TVector3 &v) const
Double_t	Mag2 () const

例題

- 配布の SUSY event を使って
  - event にある jet や lepton のpT (横方向運動量) をすべて足したものを　HT , HT に見えない運動量を加えたものを　Meff という。Example file の　Meff と　ETmiss (見えない運動量)/ Meff を計算してグラフを作れ
  - lepton のあるイベントについてvector p(lepton) + p_T(miss)の Mass を計算しなさい。
- Pythia8 で qg-> qW を作って lepton があるイベントについて　W boson 生成からくるbackground を落とすにはどうすればいいか。

disk space を節約する

- 作った event が全部欲しいわけではない
  - lepton のあるものだけ欲しい
  - missing のあるものだけ欲しい
  - jet pt のでかいものだけほしい
  - そもそも、いくつかの量が知りたいだけ。
- なのに ファイルがでかいなあ (; ;)

small2l.cc

- たくさんのroot file を読み込んで、そこから計算した一部の量だけを、別の root file に保存
- 読み込んだデータからいくつかの図を作って、pdf file に書き込む
- 作った図を別のroot file に保存してあとで加工
- 使い方
  - smallroot_input というファイルに読み込みたい Delphes の root file をかく
  - gSystem->Load(“~/delphes_test/delphes2/libDelphes"); を自分のdelphes の library のある場所に変更

小さいroot file を作る

```
TFile *ftan = new TFile("small.root","RECREATE");   作りたい root file の名前
TTree *tw = new TTree("wprim","parameter for wprim");
                              作りたい root file の tree 名
Int_t xnmu, xnele;     入れたい量を定義
Double_t xpte0,xpte1;
…
…
tw->Branch("nele",&xnele);  branch の名前をそこにいれたい量を定義
tw->Branch("nmu",&xnmu);

for(entry = 0; entry < allEntries; ++entry){
          いろいろ計算して、   xnele… に数字を代入
          最後に
           tw->Fill(); event を tree につめる
} //loop over all events

ftan->Write(); // ファイルに書き込む
```

ファイルの構造

- struct TestPlots plot を定義するところ。struct はクラスの一種
- BookHistograms 図を定義するところ。plots はTestPlots の構造を持っている
- AnalyseEvents ExRootTreeReader treereader と TestPlots plots を外部変数としていろいろ計算。とくにここではlepton が複数ある場合に ２つのレプトンの inv mass を計算している。
- small2l main program は上を順番にcall。最後に図を画面にかく。

plot をpdf にかく

plot を計算し終わったらキャンバスを定義

```
TCanvas *c1=new TCanvas("c1","test",841,594);
  c1->Divide(3,3,0.0025,0.0025); 3x3 のキャンバスを定義
  TVirtualPad *cutpad=c1-> cd(1);  第一面（左上に移動）
  plots->fPTe0->Draw();      左上に図をかく
  c1->cd(2);                  真ん中上に移動
  plots->fPTm0->Draw();      別の図をかく
  c1->cd(3);
  plots->fPTe1->Draw();
  c1->cd(4);
  plots->fPTm1->Draw();
  c1->cd(5);
…..

c1->Print("lepton.pdf");
```

c++ のコードとしてroot のライブラリーを使うこともできる (Example1.cpp)

・必要な header

```
#include <iostream>   ファイルの読み書き
#include <utility>
#include <vector>      例えばジェット n まとめて扱うとか

#include “TROOT.h"      ROOT のライブラリのheader
#include "TSystem.h"
#include "TApplication.h"

#include “TString.h"    文字の取り扱い

#include “TH2.h"          図をかく
#include "THStack.h"
#include “TLegend.h"       図のラベルとか
#include “TPaveText.h"
#include “TClonesArray.h"      効率的な配列の取り扱い
#include “TLorentzVector.h"     TLorentzVector を使うのに必要

#include “classes/DelphesClasses.h"       Delphes で作ったデータをその class の機能つきで使いたい
#include “ExRootAnalysis/ExRootTreeReader.h"    ROOT File の読みかき
#include "ExRootAnalysis/ExRootTreeWriter.h"
#include "ExRootAnalysis/ExRootTreeBranch.h"
#include "ExRootAnalysis/ExRootResult.h"
#include "ExRootAnalysis/ExRootUtilities.h"
```