Introdução à Análise de Dados em FAE
(01/11/2024)

Exercícios de Cinemática Relativística em FAE
(aula 5)

Professores:
Eliza Melo, Dilson Damião e Mauricio Thiel

Nome:
Matheus da Costa Geraldes

1 Aviso

os códigos, as imagens e assim como os arquivos usados se encontram no meu Git: `https://github.com/Geraldes-Matheus/cruso-analise-de-dados-2024-2/tree/main/exercicio5/Data`

2 Exercício 1

Neste exercício, foi realizado uma análise de dados utilizando amostras do experimento CMS, da simulação ZZTo4L(meu grupo é o 2). O objetivo foi plotar as distribuições de p_T, η e ϕ dos objetos de estado final (léptons e jatos), calcular a massa invariante dos dois léptons com maior p_T e salvar as figuras geradas no formato PNG.

3 Metodologia

Utilizando a biblioteca ROOT para manipular os arquivos de dados. Os arquivos analisados estavam localizados no diretório:

```
/opendata/eos/opendata/cms/mc/RunIISummer20UL16NanoAODv9/
    ZZTo4L_TuneCP5_13TeV_powheg_pythia8/NANOAODSIM/106
    X_mcRun2_asymptotic_v17-v1/2430000
```

Dentro deste diretório, foi utilizado os arquivos ROOT que continham os dados dos eventos.

3.1 Cálculo da Massa Invariante

A massa invariante foi calculada utilizando a seguinte fórmula:

$$M = \sqrt{2p_{T1}p_{T2}\left(\cosh(\eta_1 - \eta_2) - \cos(\phi_1 - \phi_2)\right)} \tag{1}$$

onde p_{T1}, p_{T2} são os momentos transversais dos dois léptons, e η_1, η_2, ϕ_1, ϕ_2 são as respectivas coordenadas.

3.2 Análise

Os passos principais da análise incluem:

- Leitura dos dados dos léptons (elétrons, muons e tau) e jatos.
- Preenchimento de histogramas para as distribuições de p_T, η e ϕ.
- Cálculo da massa invariante para os dois léptons com maior p_T.
- Geração de gráficos para cada uma das distribuições e a massa invariante.

3.3 Código Implementado

O código foi escrito em C++ utilizando a biblioteca ROOT.

```
#include <TTree.h>
#include <TTreeReader.h>
#include <TTreeReaderArray.h>
#include <TCanvas.h>
#include <TH1F.h>
#include <TMath.h>
#include <iostream>
#include <vector>
#include <filesystem>
#include <algorithm>
#include <string>

double calcular_massa_invariante(const std::vector<float>& pt, const std::
    vector<float>& eta, const std::vector<float>& phi) {
    if (pt.size() >= 2) {
        return sqrt(2 * pt[0] * pt[1] * (TMath::CosH(eta[0] - eta[1]) - TMath
            ::Cos(phi[0] - phi[1])));
    }
    return -1.0;
}

void analise() {
    std::string diretorio = "/opendata/eos/opendata/cms/mc/
        RunIISummer20UL16NanoAODv9/ZZTo4L_TuneCP5_13TeV_powheg_pythia8/
        NANOAODSIM/106X_mcRun2_asymptotic_v17-v1/2430000";

    std::vector<double> e_massas_invariantes;

    // Histogramas
    TH1F* hElectronPt = new TH1F("hElectronPt", "Electron p_{T} Distribution"
        , 50, 0, 200);
    TH1F* hMuonPt = new TH1F("hMuonPt", "Muon p_{T} Distribution", 50, 0,
        200);
    TH1F* hTauPt = new TH1F("hTauPt", "Tau p_{T} Distribution", 50, 0, 200);
    TH1F* hJetPt = new TH1F("hJetPt", "Jet p_{T} Distribution", 50, 0, 200);

    TH1F* hElectronEta = new TH1F("hElectronEta", "Electron #eta Distribution
        ", 50, -5, 5);
    TH1F* hMuonEta = new TH1F("hMuonEta", "Muon #eta Distribution", 50, -5,
        5);
    TH1F* hTauEta = new TH1F("hTauEta", "Tau #eta Distribution", 50, -5, 5);
    TH1F* hJetEta = new TH1F("hJetEta", "Jet #eta Distribution", 50, -5, 5);

    TH1F* hElectronPhi = new TH1F("hElectronPhi", "Electron #phi Distribution
        ", 50, -3.14, 3.14);
    TH1F* hMuonPhi = new TH1F("hMuonPhi", "Muon #phi Distribution", 50,
        -3.14, 3.14);
    TH1F* hTauPhi = new TH1F("hTauPhi", "Tau #phi Distribution", 50, -3.14,
        3.14);
```

```
    TH1F* hJetPhi = new TH1F("hJetPhi", "Jet #phi Distribution", 50, -3.14, 3.14);

    for (const auto& entry : std::filesystem::directory_iterator(diretorio)) {
        std::string file_path = entry.path();
        TFile file(file_path.c_str(), "READ");
        if (!file.IsOpen()) continue;

        TTreeReader reader("Events", &file);
        TTreeReaderArray<float> Electron_pt(reader, "Electron_pt");
        TTreeReaderArray<float> Electron_eta(reader, "Electron_eta");
        TTreeReaderArray<float> Electron_phi(reader, "Electron_phi");
        TTreeReaderArray<float> Muon_pt(reader, "Muon_pt");
        TTreeReaderArray<float> Muon_eta(reader, "Muon_eta");
        TTreeReaderArray<float> Muon_phi(reader, "Muon_phi");
        TTreeReaderArray<float> Jet_pt(reader, "Jet_pt");
        TTreeReaderArray<float> Jet_eta(reader, "Jet_eta");
        TTreeReaderArray<float> Jet_phi(reader, "Jet_phi");
        TTreeReaderArray<float> Tau_pt(reader, "Tau_pt");
        TTreeReaderArray<float> Tau_eta(reader, "Tau_eta");
        TTreeReaderArray<float> Tau_phi(reader, "Tau_phi");

        while (reader.Next()) {
            // Preencher histogramas de pT,    e
            for (size_t i = 0; i < Electron_pt.GetSize(); ++i) {
                hElectronPt->Fill(Electron_pt[i]);
                hElectronEta->Fill(Electron_eta[i]);
                hElectronPhi->Fill(Electron_phi[i]);
            }
            for (size_t i = 0; i < Muon_pt.GetSize(); ++i) {
                hMuonPt->Fill(Muon_pt[i]);
                hMuonEta->Fill(Muon_eta[i]);
                hMuonPhi->Fill(Muon_phi[i]);
            }
            for (size_t i = 0; i < Jet_pt.GetSize(); ++i) {
                hJetPt->Fill(Jet_pt[i]);
                hJetEta->Fill(Jet_eta[i]);
                hJetPhi->Fill(Jet_phi[i]);
            }
            for (size_t i = 0; i < Tau_pt.GetSize(); ++i) {
                hTauPt->Fill(Tau_pt[i]);
                hTauEta->Fill(Tau_eta[i]);
                hTauPhi->Fill(Tau_phi[i]);
            }

            // Calcular a massa invariante dos dois l ptons com maior pT
            std::vector<std::pair<float, int>> leptons; // (pT,  ndice )
            for (size_t i = 0; i < Electron_pt.GetSize(); ++i) {
                leptons.emplace_back(Electron_pt[i], i); // El trons
            }
            for (size_t i = 0; i < Muon_pt.GetSize(); ++i) {
                leptons.emplace_back(Muon_pt[i], i + Electron_pt.GetSize()); // Muons
            }
            for (size_t i = 0; i < Tau_pt.GetSize(); ++i) {
                leptons.emplace_back(Tau_pt[i], i + Electron_pt.GetSize() + Muon_pt.GetSize()); // Taus
            }

            // Ordenar  pT
            std::sort(leptons.rbegin(), leptons.rend());

            if (leptons.size() >= 2) {
                int idx1 = leptons[0].second;
                int idx2 = leptons[1].second;

                float pt1, eta1, phi1, pt2, eta2, phi2;
```

```
103
104                 if (idx1 < Electron_pt.GetSize()) {
105                     pt1 = Electron_pt[idx1];
106                     eta1 = Electron_eta[idx1];
107                     phi1 = Electron_phi[idx1];
108                 } else if (idx1 < Electron_pt.GetSize() + Muon_pt.GetSize())
                        {
109                     pt1 = Muon_pt[idx1 - Electron_pt.GetSize()];
110                     eta1 = Muon_eta[idx1 - Electron_pt.GetSize()];
111                     phi1 = Muon_phi[idx1 - Electron_pt.GetSize()];
112                 } else {
113                     pt1 = Tau_pt[idx1 - Electron_pt.GetSize() - Muon_pt.
                            GetSize()];
114                     eta1 = Tau_eta[idx1 - Electron_pt.GetSize() - Muon_pt.
                            GetSize()];
115                     phi1 = Tau_phi[idx1 - Electron_pt.GetSize() - Muon_pt.
                            GetSize()];
116                 }
117
118                 if (idx2 < Electron_pt.GetSize()) {
119                     pt2 = Electron_pt[idx2];
120                     eta2 = Electron_eta[idx2];
121                     phi2 = Electron_phi[idx2];
122                 } else if (idx2 < Electron_pt.GetSize() + Muon_pt.GetSize())
                        {
123                     pt2 = Muon_pt[idx2 - Electron_pt.GetSize()];
124                     eta2 = Muon_eta[idx2 - Electron_pt.GetSize()];
125                     phi2 = Muon_phi[idx2 - Electron_pt.GetSize()];
126                 } else {
127                     pt2 = Tau_pt[idx2 - Electron_pt.GetSize() - Muon_pt.
                            GetSize()];
128                     eta2 = Tau_eta[idx2 - Electron_pt.GetSize() - Muon_pt.
                            GetSize()];
129                     phi2 = Tau_phi[idx2 - Electron_pt.GetSize() - Muon_pt.
                            GetSize()];
130                 }
131
132                 // Calcular massa invariante
133                 std::vector<float> pt_values = {pt1, pt2};
134                 std::vector<float> eta_values = {eta1, eta2};
135                 std::vector<float> phi_values = {phi1, phi2};
136                 double massa_invariante = calcular_massa_invariante(pt_values
                        , eta_values, phi_values);
137
138                 if (massa_invariante >= 0) {
139                     e_massas_invariantes.push_back(massa_invariante);
140                 }
141             }
142         }
143     }
144
145
146     TCanvas* canvas;
147
148     // Salvar histogramas para pT
149     canvas = new TCanvas("canvasElectronPt", "Electron p_{T} Distribution",
            800, 600);
150     hElectronPt->SetLineColor(kBlue);
151     hElectronPt->Draw();
152     hElectronPt->GetXaxis()->SetTitle("p_{T} (GeV/c)");
153     hElectronPt->GetYaxis()->SetTitle("Events");
154     canvas->SaveAs("electron_pt_distribution.png");
155     delete canvas;
156
157     canvas = new TCanvas("canvasMuonPt", "Muon p_{T} Distribution", 800, 600)
            ;
158     hMuonPt->SetLineColor(kRed);
159     hMuonPt->Draw();
```

```
160       hMuonPt->GetXaxis()->SetTitle("p_{T}␣(GeV/c)");
161       hMuonPt->GetYaxis()->SetTitle("Events");
162       canvas->SaveAs("muon_pt_distribution.png");
163       delete canvas;
164
165       canvas = new TCanvas("canvasTauPt", "Tau␣p_{T}␣Distribution", 800, 600);
166       hTauPt->SetLineColor(kMagenta);
167       hTauPt->Draw();
168       hTauPt->GetXaxis()->SetTitle("p_{T}␣(GeV/c)");
169       hTauPt->GetYaxis()->SetTitle("Events");
170       canvas->SaveAs("tau_pt_distribution.png");
171       delete canvas;
172
173       canvas = new TCanvas("canvasJetPt", "Jet␣p_{T}␣Distribution", 800, 600);
174       hJetPt->SetLineColor(kGreen);
175       hJetPt->Draw();
176       hJetPt->GetXaxis()->SetTitle("p_{T}␣(GeV/c)");
177       hJetPt->GetYaxis()->SetTitle("Events");
178       canvas->SaveAs("jet_pt_distribution.png");
179       delete canvas;
180
181       // Salvar histogramas para
182       canvas = new TCanvas("canvasElectronEta", "Electron␣#eta␣Distribution",
              800, 600);
183       hElectronEta->SetLineColor(kBlue);
184       hElectronEta->Draw();
185       hElectronEta->GetXaxis()->SetTitle("#eta");
186       hElectronEta->GetYaxis()->SetTitle("Events");
187       canvas->SaveAs("electron_eta_distribution.png");
188       delete canvas;
189
190       canvas = new TCanvas("canvasMuonEta", "Muon␣#eta␣Distribution", 800, 600)
              ;
191       hMuonEta->SetLineColor(kRed);
192       hMuonEta->Draw();
193       hMuonEta->GetXaxis()->SetTitle("#eta");
194       hMuonEta->GetYaxis()->SetTitle("Events");
195       canvas->SaveAs("muon_eta_distribution.png");
196       delete canvas;
197
198       canvas = new TCanvas("canvasTauEta", "Tau␣#eta␣Distribution", 800, 600);
199       hTauEta->SetLineColor(kMagenta);
200       hTauEta->Draw();
201       hTauEta->GetXaxis()->SetTitle("#eta");
202       hTauEta->GetYaxis()->SetTitle("Events");
203       canvas->SaveAs("tau_eta_distribution.png");
204       delete canvas;
205
206       canvas = new TCanvas("canvasJetEta", "Jet␣#eta␣Distribution", 800, 600);
207       hJetEta->SetLineColor(kGreen);
208       hJetEta->Draw();
209       hJetEta->GetXaxis()->SetTitle("#eta");
210       hJetEta->GetYaxis()->SetTitle("Events");
211       canvas->SaveAs("jet_eta_distribution.png");
212       delete canvas;
213
214       // Salvar histogramas para
215       canvas = new TCanvas("canvasElectronPhi", "Electron␣#phi␣Distribution",
              800, 600);
216       hElectronPhi->SetLineColor(kBlue);
217       hElectronPhi->Draw();
218       hElectronPhi->GetXaxis()->SetTitle("#phi");
219       hElectronPhi->GetYaxis()->SetTitle("Events");
220       canvas->SaveAs("electron_phi_distribution.png");
221       delete canvas;
222
223       canvas = new TCanvas("canvasMuonPhi", "Muon␣#phi␣Distribution", 800, 600)
              ;
```

```
    hMuonPhi->SetLineColor(kRed);
    hMuonPhi->Draw();
    hMuonPhi->GetXaxis()->SetTitle("#phi");
    hMuonPhi->GetYaxis()->SetTitle("Events");
    canvas->SaveAs("muon_phi_distribution.png");
    delete canvas;

    canvas = new TCanvas("canvasTauPhi", "Tau #phi Distribution", 800, 600);
    hTauPhi->SetLineColor(kMagenta);
    hTauPhi->Draw();
    hTauPhi->GetXaxis()->SetTitle("#phi");
    hTauPhi->GetYaxis()->SetTitle("Events");
    canvas->SaveAs("tau_phi_distribution.png");
    delete canvas;

    canvas = new TCanvas("canvasJetPhi", "Jet #phi Distribution", 800, 600);
    hJetPhi->SetLineColor(kGreen);
    hJetPhi->Draw();
    hJetPhi->GetXaxis()->SetTitle("#phi");
    hJetPhi->GetYaxis()->SetTitle("Events");
    canvas->SaveAs("jet_phi_distribution.png");
    delete canvas;

    // plot da Massa Invariante
    TH1F* hMassaInvariante = new TH1F("hMassaInvariante", "Invariant Mass
        Distribution", 50, 0, 200);
    for (const auto& massa : e_massas_invariantes) {
        if (massa >= 0) hMassaInvariante->Fill(massa);
    }

    canvas = new TCanvas("canvasInvariantMass", "Invariant Mass Distribution"
        , 800, 600);
    hMassaInvariante->SetLineColor(kBlack);
    canvas->SetLogy();
    hMassaInvariante->Draw();
    hMassaInvariante->GetXaxis()->SetTitle("Invariant Mass (GeV/c^{2})");
    hMassaInvariante->GetYaxis()->SetTitle("Events");
    canvas->SaveAs("invariant_mass_distribution.png");
    delete canvas;


    delete hElectronPt;
    delete hMuonPt;
    delete hTauPt;
    delete hJetPt;
    delete hElectronEta;
    delete hMuonEta;
    delete hTauEta;
    delete hJetEta;
    delete hElectronPhi;
    delete hMuonPhi;
    delete hTauPhi;
    delete hJetPhi;
    delete hMassaInvariante;
}
```

4 Resultados

Os histogramas gerados mostram as distribuições de p_T, η, e ϕ para os léptons e jatos. Além disso, a distribuição da massa invariante dos dois léptons com maior p_T foi plotada. Todas as figuras foram salvas no formato PNG.

4.1 plots Pt

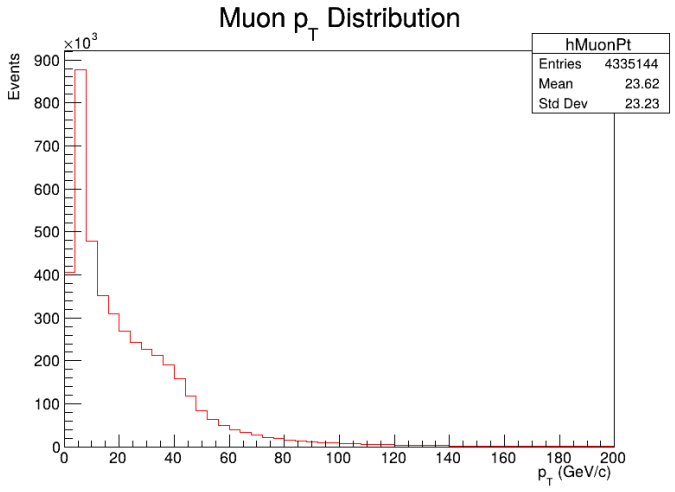

Figure 1: Distribuição de p_T do múon

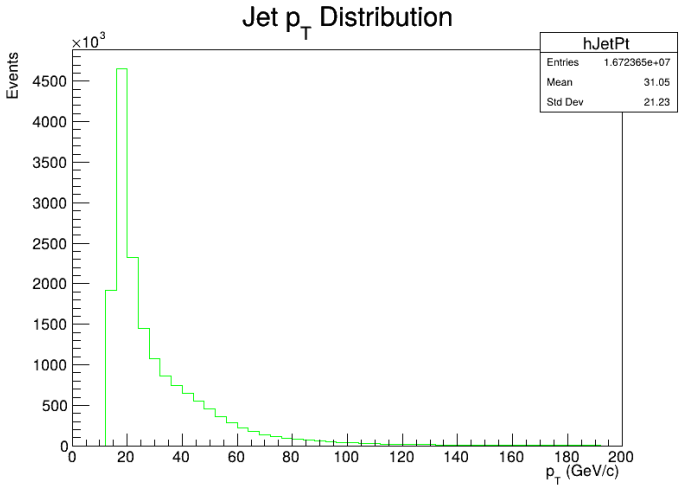

Figure 2: Distribuição de p_T do jato

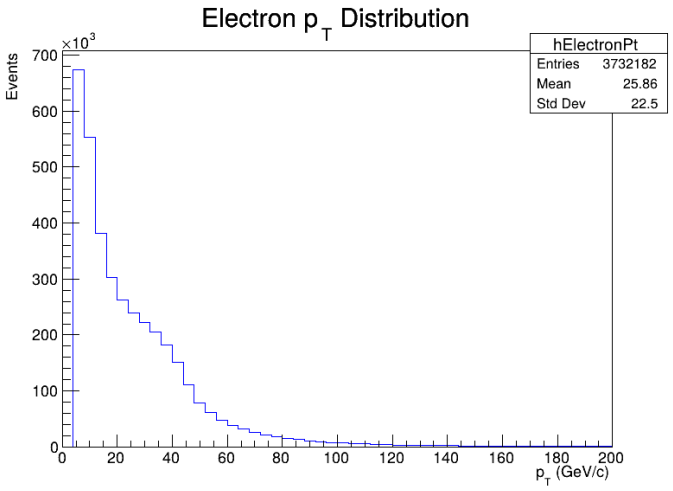

Figure 3: Distribuição de p_T do elétron

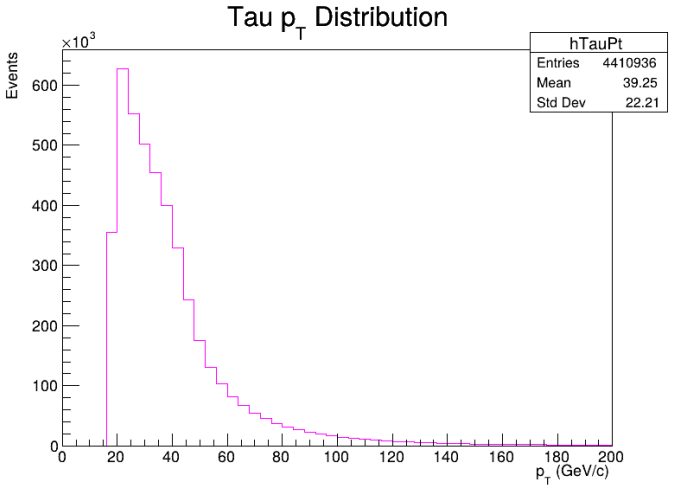

Figure 4: Distribuição de p_T do tau

4.2 plots Eta

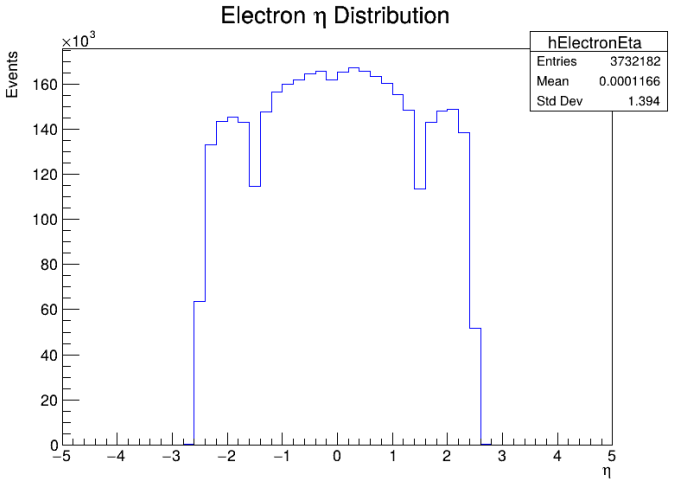

Figure 5: Distribuição de η do elétron

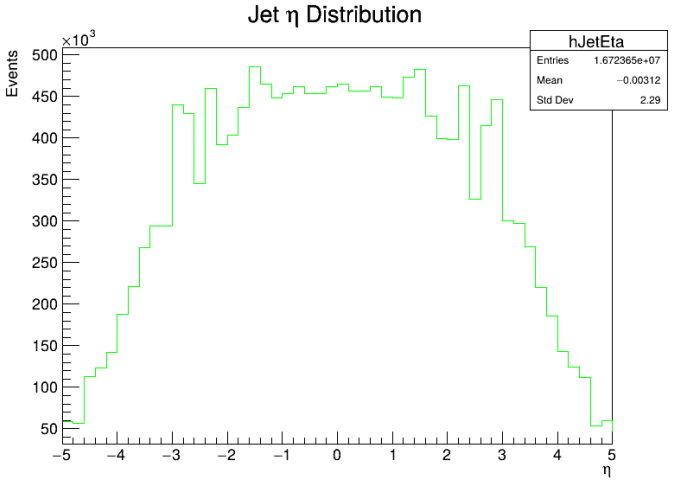

Figure 6: Distribuição de η do jato

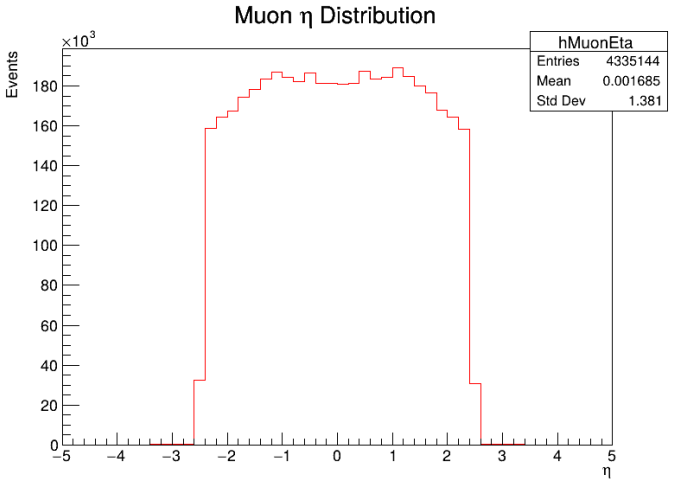

Figure 7: Distribuição de η do múon

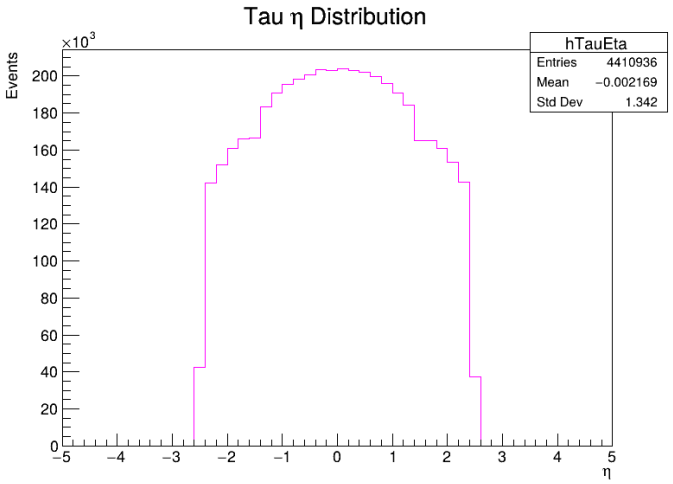

Figure 8: Distribuição de η do tau

4.3 plots Phi

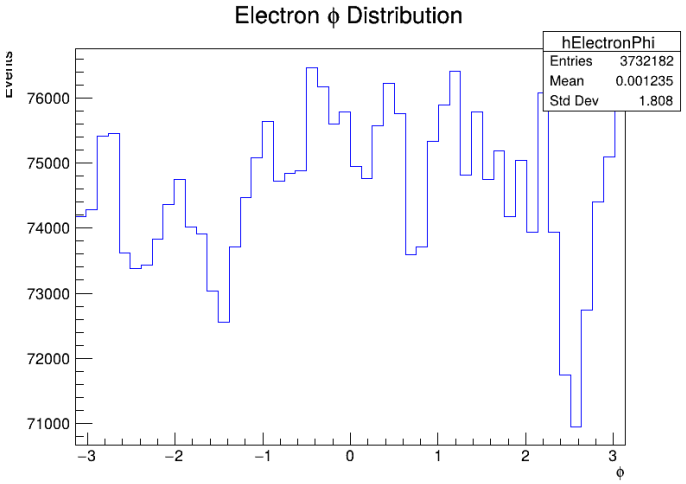

Figure 9: Distribuição de ϕ do elétron

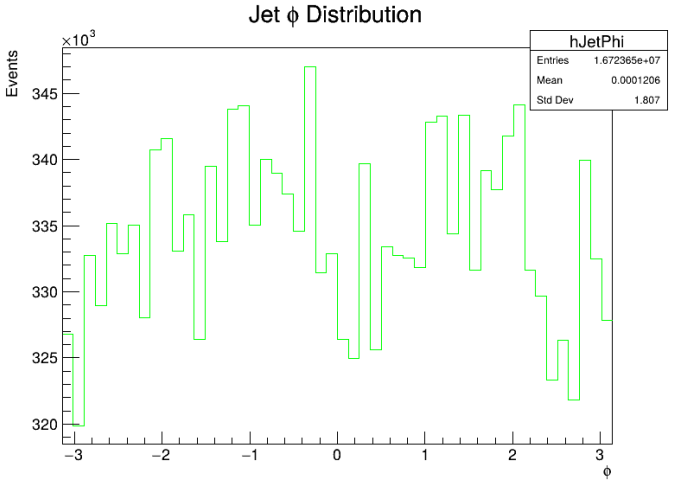

Figure 10: Distribuição de ϕ do jato

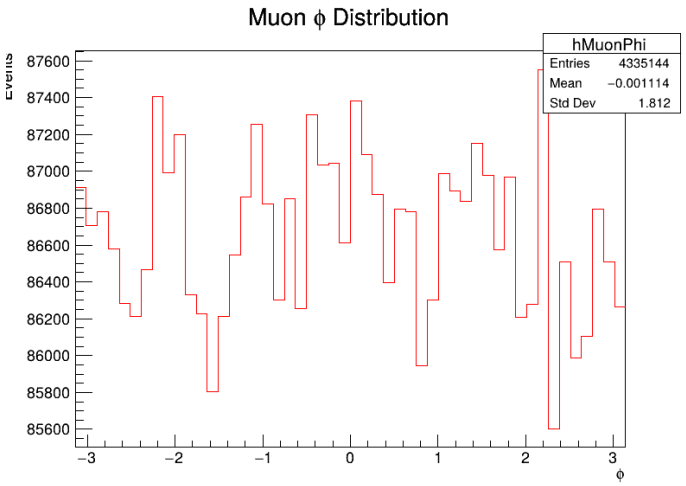

Figure 11: Distribuição de ϕ do múon

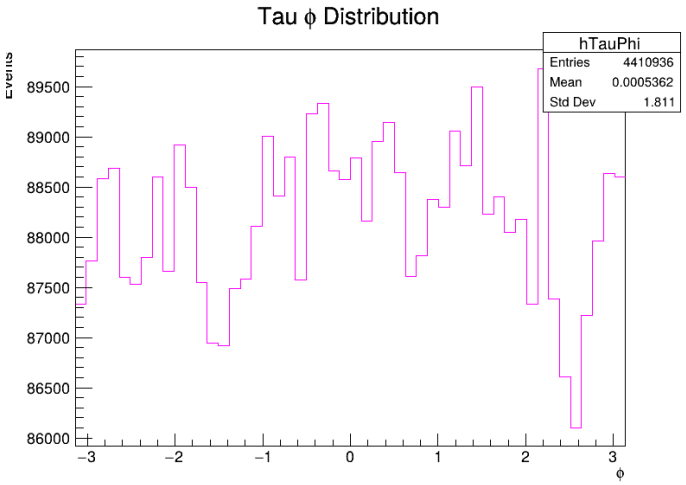

Figure 12: Distribuição de ϕ do tau

4.4 massa invariante dos dois léptons com maior pT

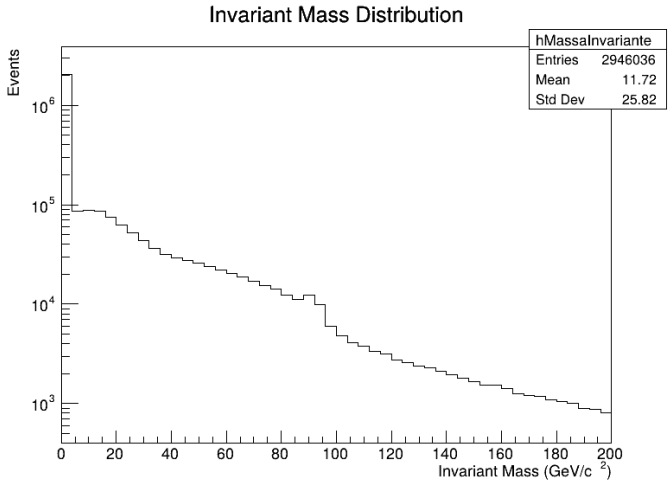

Figure 13: Distribuição da massa invariante