Introdução à Análise de Dados em FAE
(15/10/2024)

Exercícios de ROOT (aula 3)

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/Exerc%C3%ADcio3/Data`

2 Exercício 0

Neste exercício, o objetivo foi analisar amostras de dados experimentais do OpenData, que se encontra Grid no seguinte caminho:

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

Utilizando a ferramenta `MakeClass` para gerar histogramas que ilustram as distribuições de algumas propriedades de partículas, especificamente muons e elétrons.

Definição dos Histogramas

Três histogramas foram definidos para representar as distribuições das seguintes propriedades:

- **Muon PT (Momento Transversal)**: Representa a distribuição do momento transversal dos muons.
- **Muon Mass (Massa dos Muons)**: Representa a distribuição da massa dos muons.
- **Electron Mass (Massa dos Elétrons)**: Representa a distribuição da massa dos elétrons.

Carregamento e Preenchimento dos Dados

Os dados foram carregados a partir de um `TChain`. O código percorreu cada entrada do conjunto de dados, preenchendo os histogramas com os valores correspondentes de momento e massa.

```
for (Long64_t jentry = 0; jentry < nentries; jentry++) {
    Long64_t ientry = LoadTree(jentry);
    if (ientry < 0) break;
    fChain->GetEntry(jentry);

    for (UInt_t i = 0; i < nMuon; i++) {
        histo1->Fill(Muon_pt[i]);
        histo2->Fill(Muon_mass[i]);
    }

    for (UInt_t j = 0; j < nElectron; j++) {
        histo3->Fill(Electron_mass[j]);
    }
}
```

Visualização dos Resultados

Para cada histograma, um `TCanvas` foi criado para permitir a visualização gráfica e posterior salvamento dos histogramas em formato PNG.

```
TCanvas *canvas1 = new TCanvas("canvas1", "Histograma de Muon PT", 800, 600);
histo1->Draw();
canvas1->Print("histograma_muon_pt.png");
```

Os histogramas gerados foram salvos como arquivos PNG, permitindo uma análise visual das distribuições. As imagens resultantes foram:

- `histograma_muon_pt.png`

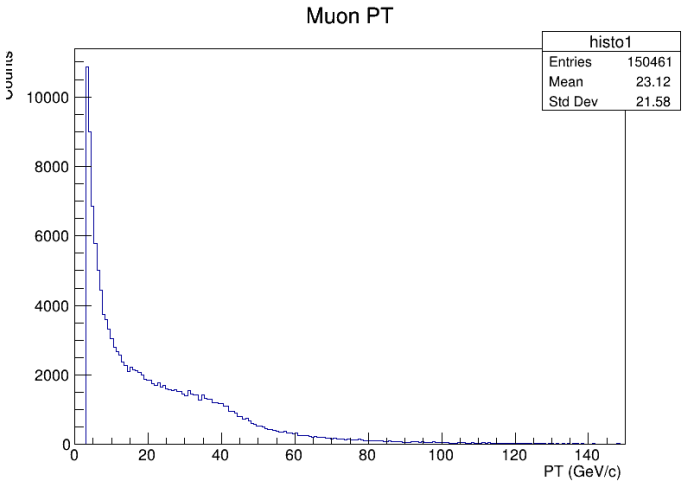

Figure 1: Muon Pt

- `histograma_muon_mass.png`

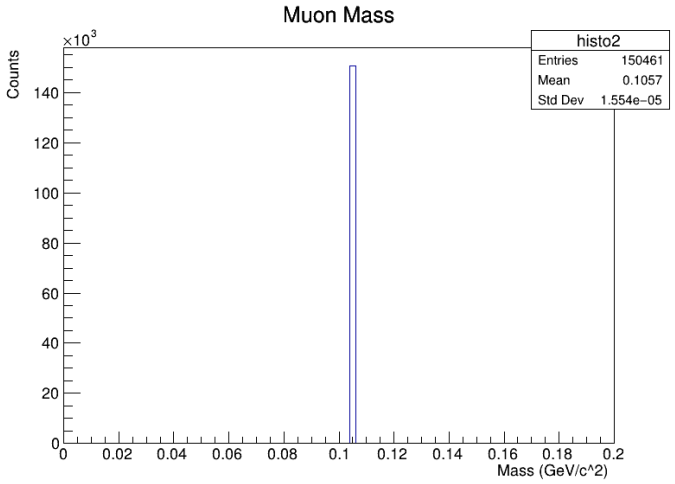

Figure 2: Muon Mass

- `histograma_electron_mass.png`

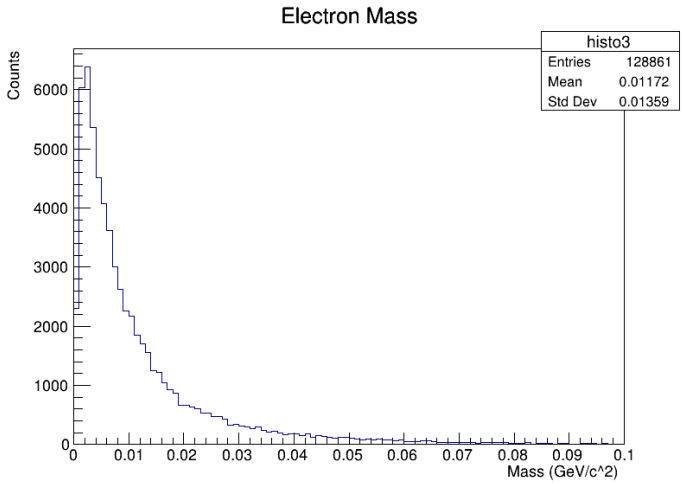

Figure 3: electron Mass

3 Exercício 1

No primeiro exercício, foi criada uma função paramétrica definida como:

$$f(x) = \frac{p_0 \cdot \sin(p_1 \cdot x)}{x}$$

Onde o gráfico da função é:

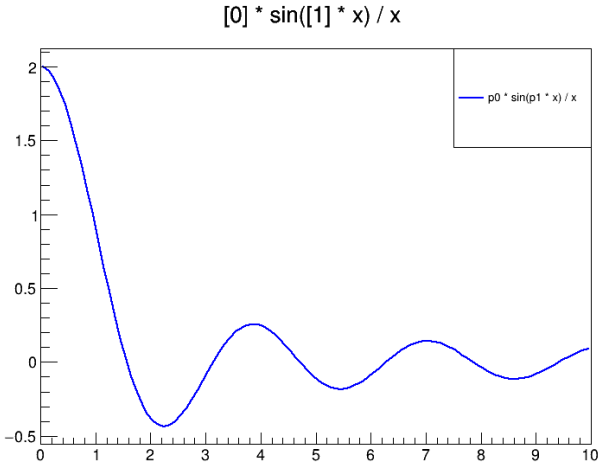

Figure 4: Enter Caption

Foram traçadas curvas da função para diferentes valores dos parâmetros p_0 e p_1, com a cor azul.

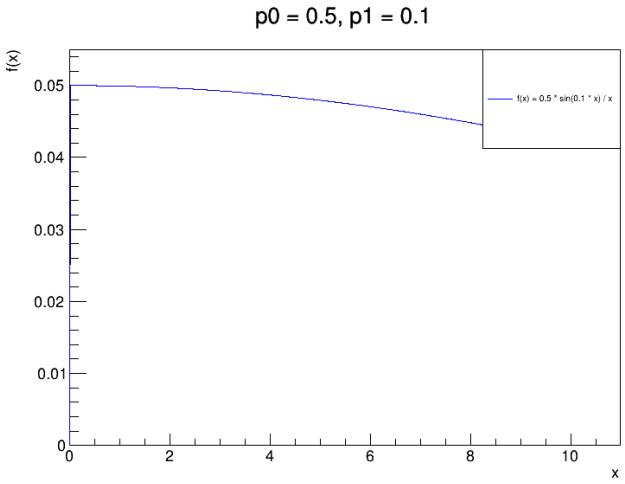

Figure 5: Enter Caption

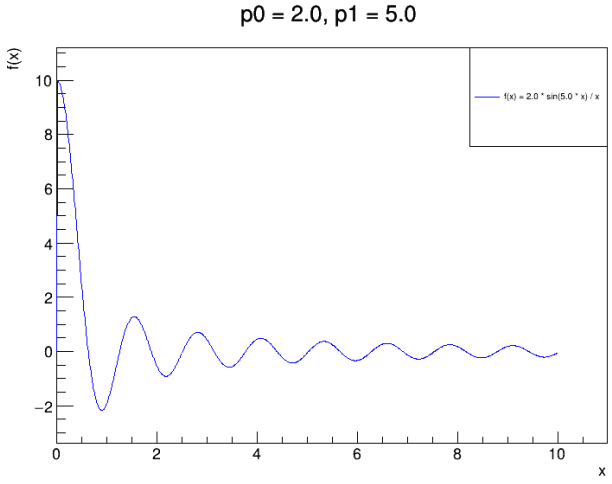

Figure 6: Enter Caption

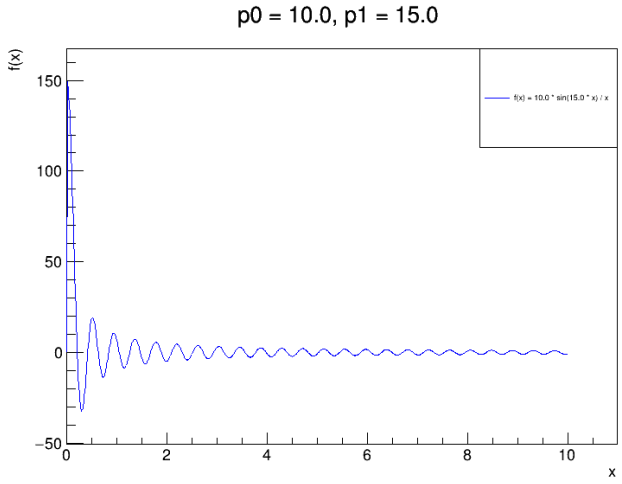

Figure 7: Enter Caption

Após o gráfico, foram computados para $p0 = 1$ e $p1 = 2$:

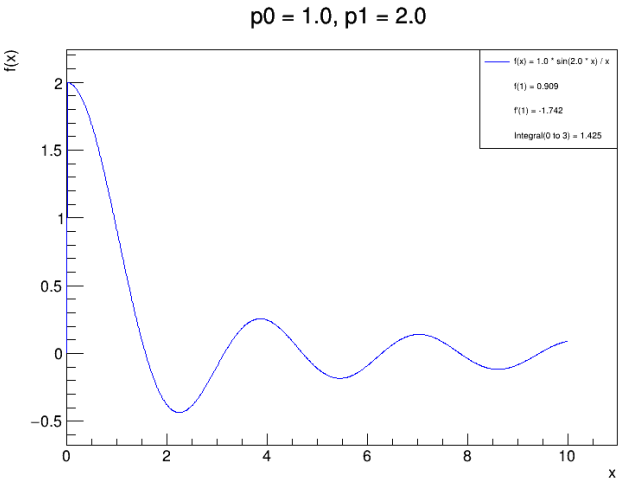

Figure 8: Enter Caption

- **Valor da função para** $x = 1$:

$$f(1) = 0,909297$$

- **Derivada da função para** $x = 1$:

$$f'(1) = -1,74159$$

- **Integral da função entre** 0 e 3:

$$\int_0^3 f(x)\,dx = 1,42469$$

4 Exercício 2

No segundo exercício, um conjunto de pontos, utilizando um conjunto de dados `graphdata.txt`, foi plotado utilizando a classe `TGraph`. Os pontos foram representados por marcadores em forma de quadrado preto e uma linha foi traçada conectando os pontos. Além disso, foi criada uma `TGraphError` utilizando um conjunto de dados com erros nos eixos x e y a partir do arquivo `graphdata_error.txt`.

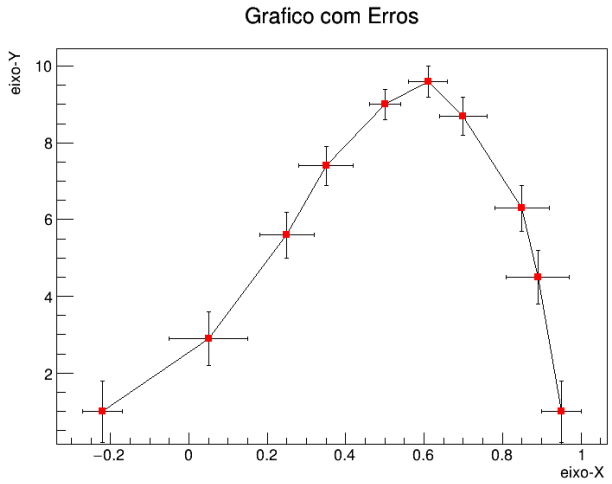

Figure 9: Enter Caption

5 Exercício 3

O terceiro exercício envolveu a criação de um histograma unidimensional com 50 bins entre 0 e 10, preenchido com 10000 números aleatórios distribuídos normalmente, com média 5 e desvio padrão 2. O histograma foi plotado e, utilizando a documentação do `THistPainter`, foi exibida uma caixa de estatísticas mostrando:

- Número de entradas
- Média
- RMS

- Integral do histograma
- Número de underflows
- Número de overflows
- Assimetria (skewness)
- Curtose (kurtosis)

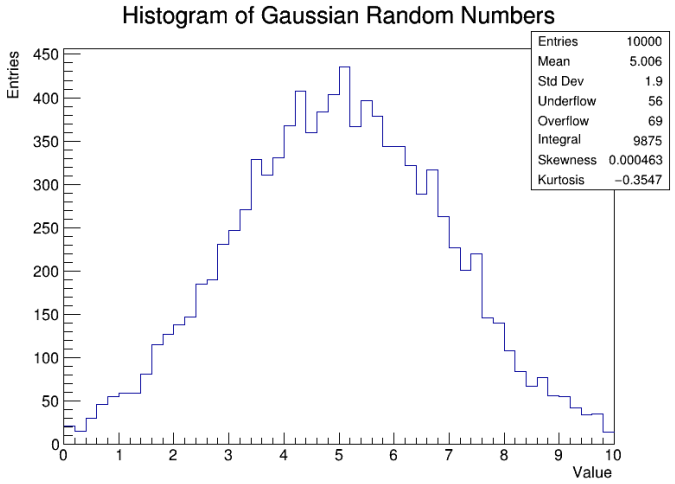

Figure 10: Resultado do exercício 3

6 Exercício 4

O último exercício consistiu em utilizar uma árvore de dados contida no arquivo `tree.root` para criar uma distribuição do momento total das partículas cuja energia de feixe estava fora da média por mais de 0.2. Foram utilizados objetos `TCut` para selecionar os eventos. A distribuição foi projetada em um histograma, que foi então desenhado e salvo em um arquivo.

6.1 Analisando o tree.root

Analisando as Variáveis contidas no TTree.

```
root [2] TFile *file = TFile::Open("tree.root");
root [3] TTree *tree = (TTree*)file->Get("tree1");
root [4] tree->Print();
******************************************************************************
*Tree    :tree1     : Reconstruction ntuple                                  *
*Entries :   100000 : Total =         2810761 bytes  File  Size =    2171135 *
*        :          : Tree compression factor =   1.30                       *
******************************************************************************
*Br    0 :event     : event/I                                                *
*Entries :   100000 : Total  Size=     401519 bytes  File Size  =     134514 *
*Baskets :       12 : Basket Size=      32000 bytes  Compression=   2.85     *
*.............................................................................*
*Br    1 :ebeam     : ebeam/F                                                *
*Entries :   100000 : Total  Size=     401519 bytes  File Size  =     260330 *
*Baskets :       12 : Basket Size=      32000 bytes  Compression=   1.47     *
*.............................................................................*
*Br    2 :px        : px/F                                                   *
*Entries :   100000 : Total  Size=     401465 bytes  File Size  =     359238 *
*Baskets :       12 : Basket Size=      32000 bytes  Compression=   1.07     *
*.............................................................................*
*Br    3 :py        : py/F                                                   *
*Entries :   100000 : Total  Size=     401465 bytes  File Size  =     359138 *
*Baskets :       12 : Basket Size=      32000 bytes  Compression=   1.07     *
*.............................................................................*
*Br    4 :pz        : pz/F                                                   *
*Entries :   100000 : Total  Size=     401465 bytes  File Size  =     292046 *
*Baskets :       12 : Basket Size=      32000 bytes  Compression=   1.31     *
*.............................................................................*
*Br    5 :zv        : zv/F                                                   *
*Entries :   100000 : Total  Size=     401465 bytes  File Size  =     349087 *
*Baskets :       12 : Basket Size=      32000 bytes  Compression=   1.10     *
```

Figure 11: Variáveis contidas no TTree

- `ebeam`: Energia do feixe (float)
- `px`: Componente x do momento (float)
- `py`: Componente y do momento (float)
- `pz`: Componente z do momento (float)

6.2 Etapas do Exercício

1. **Calcular a média da energia do feixe (`ebeam`):**É preciso calcular a média da variável `ebeam`.
2. **Definir os cortes (TCut):** Os cortes devem selecionar os eventos onde a energia do feixe (`ebeam`) está fora da média por mais de 0.2.
3. **Calcular o momento total:** O momento total pode ser calculado usando as componentes do momento:

$$\text{totalMomentum} = \sqrt{p_x^2 + p_y^2 + p_z^2}$$

4. **Criar um histograma:** Criando um histograma para a distribuição do momento total.

5. **Preencher o histograma** com os eventos que atendem ao corte.

6. **Desenhar o histograma e salvar em um arquivo.**

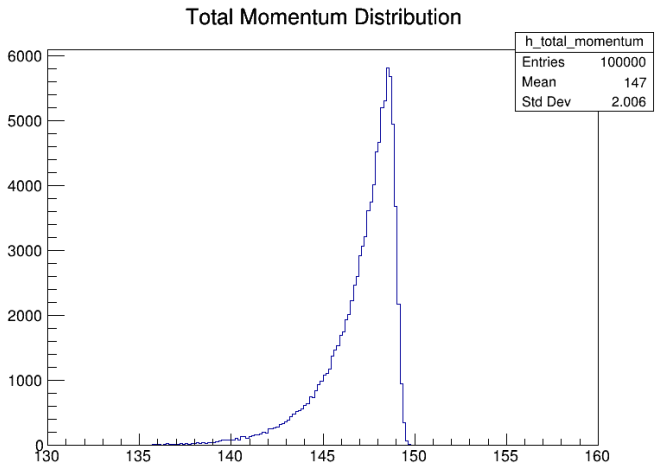

Figure 12: momento total das partículas cuja energia de feixe > 0.2

A Códigos dos exercícios em C

A.1 Código 1: Exercícios 0

Título do Código 1

```
#define ex1_cxx
#include "ex1.h"
#include <TH1F.h>
#include <TCanvas.h>

void ex1::Loop()
{
    if (fChain == 0) return;

    Long64_t nentries = fChain->GetEntriesFast();
    TH1F *histo1 = new TH1F("histo1", "Muon PT;PT (GeV/c);Counts",
        200, 0, 150);
    TH1F *histo2 = new TH1F("histo2", "Muon Mass;Mass (GeV/c^2);
        Counts", 100, 0, 0.2);
    TH1F *histo3 = new TH1F("histo3", "Electron Mass;Mass (GeV/c^2);
        Counts", 100, 0, 0.1);

    for (Long64_t jentry = 0; jentry < nentries; jentry++) {
        Long64_t ientry = LoadTree(jentry);
        if (ientry < 0) break;
        fChain->GetEntry(jentry);

        for (UInt_t i = 0; i < nMuon; i++) {
            histo1->Fill(Muon_pt[i]);
            histo2->Fill(Muon_mass[i]);
        }

        for (UInt_t j = 0; j < nElectron; j++) {
            histo3->Fill(Electron_mass[j]);
        }
    }

    TCanvas *canvas1 = new TCanvas("canvas1", "Histograma de Muon PT"
        , 800, 600);
    histo1->Draw();
    canvas1->Print("histograma_muon_pt.png");

    TCanvas *canvas2 = new TCanvas("canvas2", "Histograma de Muon
        Mass", 800, 600);
    histo2->Draw();
    canvas2->Print("histograma_muon_mass.png");

    TCanvas *canvas3 = new TCanvas("canvas3", "Histograma de Electron
        Mass", 800, 600);
    histo3->Draw();
    canvas3->Print("histograma_electron_mass.png");

    delete canvas1;
    delete canvas2;
    delete canvas3;

    delete histo1;
    delete histo2;
    delete histo3;
}
```

A.2 Código 2: Exercícios 1

Título do Código 2

```
void exer_1() {
    // Definindo a fun  o com par metros p0 e p1
    auto sine_function = [](double x, double p0, double p1) {
        return p0 * sin(p1 * x) / x;
    };

    // Valores dos par metros
    std::vector<std::pair<double, double>> parameters = {
        {0.5, 0.1},
        {2.0, 5.0},
        {10.0, 15.0},
        {1.0, 2.0} // Adicionando o novo par de par metros
    };

    // Intervalo de x
    const int nPoints = 1000;
    double x[nPoints];
    double y[nPoints];

    // Criar e salvar os gr ficos
    for (size_t i = 0; i < parameters.size(); ++i) {
        double p0 = parameters[i].first;
        double p1 = parameters[i].second;

        // Preencher os valores de x e y
        for (int j = 0; j < nPoints; ++j) {
            x[j] = j / 100.0; // de 0 a 10
            if (x[j] != 0) {
                y[j] = sine_function(x[j], p0, p1);
            } else {
                y[j] = 0; // Para evitar divis o por zero
            }
        }

        // Criar o gr fico
        TGraph *graph = new TGraph(nPoints, x, y);
        graph->SetTitle(Form("p0 = %.1f, p1 = %.1f", p0, p1));
        graph->GetXaxis()->SetTitle("x");
        graph->GetYaxis()->SetTitle("f(x)");
        graph->SetLineColor(kBlue); // Definindo a cor do gr fico
            para azul

        // Adicionar legenda com a fun  o para todos os gr ficos
        auto legend = new TLegend(0.7, 0.7, 0.9, 0.9); // Canto
            superior direito
        legend->AddEntry(graph, Form("f(x) = %.1f * sin(%.1f * x) / x
            ", p0, p1), "l");

        // Verificar se p0=1 e p1=2 para c lculos adicionais
        if (p0 == 1.0 && p1 == 2.0) {
            double x_value = 1.0;
            double function_value = sine_function(x_value, p0, p1);

            // Definindo a fun  o para a derivada
            TF1 *f = new TF1("f", [&](double *x, double *par) {
                return sine_function(x[0], par[0], par[1]);
            }, 0, 10, 2);
            f->SetParameters(p0, p1);
            double derivative_value = f->Derivative(x_value);

            // Definindo a fun  o para a integral
            double integral_value = f->Integral(0, 3);

            // Adicionar resultados adicionais    legenda
            legend->AddEntry((TObject*)0, Form("f(1) = %.3f",
                function_value), "");
            legend->AddEntry((TObject*)0, Form("f'(1) = %.3f",
                derivative_value), "");
            legend->AddEntry((TObject*)0, Form("Integral(0 to 3) =
                %.3f", integral_value), "");

            delete f; // Limpar a fun  o
        }

        // Salvar gr fico em PNG
        TCanvas *canvas = new TCanvas("canvas", "Canvas", 800, 600);
        graph->Draw("AL");
        legend->Draw(); // Desenhar a legenda ap s o gr fico
```

A.3 Código 3: Exercícios 2

Código do Exercícios 2

```
#include <TGraph.h>
#include <TGraphErrors.h>
#include <TCanvas.h>
#include <TStyle.h>
#include <iostream>

void plotGraphs() {
    // Cria  o do canvas
    TCanvas *canvas = new TCanvas("canvas", "Graph␣with␣Errors", 800,
         600);

    // Criar TGraph para os dados
    TGraph *graph = new TGraph("graphdata.txt");
    if (!graph) {
        std::cerr << "Erro␣ao␣carregar␣graphdata.txt" << std::endl;
        return;
    }

    // Criar TGraphErrors para os dados de erro
    TGraphErrors *graphErrors = new TGraphErrors("graphdata_error.txt
        ");
    if (!graphErrors) {
        std::cerr << "Erro␣ao␣carregar␣graphdata_error.txt" << std::
            endl;
        return;
    }

    // Configura  es do gr fico
    graph->SetMarkerStyle(22); // Estilo do marcador: caixa preta
    graph->SetMarkerColor(kBlack);
    graph->SetTitle("Grafico␣com␣Erros;eixo-X;eixo-Y");
    graph->Draw("ALP"); // Desenha os pontos do TGraph com linhas

    // Desenha o gr fico com erro
    graphErrors->SetMarkerStyle(21); // Estilo do marcador: c rculo
    graphErrors->SetMarkerColor(kRed);
    graphErrors->Draw("P"); // Desenha os pontos do TGraphErrors

    // Salva o gr fico em PNG
    canvas->Print("exerc cio_2.png");

    // Limpa a mem ria
    delete canvas;
    delete graph;
    delete graphErrors;
}

void exer_2() {
    plotGraphs();
}
```

A.4 Código 4: Exercícios 3

Código do Exercícios 3

```
#include <TH1F.h>
#include <TCanvas.h>
#include <TRandom3.h>
#include <TStyle.h>
#include <TApplication.h>

void exer_3() {
    // Cria o gerador de n meros aleat rios
    TRandom3 random;

    // Cria um histograma com 50 bins entre 0 e 10
    TH1F *hist = new TH1F("hist", "Histogram of Gaussian Random
        Numbers;Value;Entries", 50, 0, 10);

    // Preenche o histograma com 10000 n meros aleat rios com
        distribui  o gaussiana
    for (int i = 0; i < 10000; i++) {
        double value = random.Gaus(5, 2); // M dia 5 e sigma 2
        hist->Fill(value);
    }

    // Cria um canvas para desenhar o histograma
    TCanvas *canvas = new TCanvas("canvas", "Gaussian Histogram",
        800, 600);

    // Define o modo da caixa de estat sticas
    gStyle->SetOptStat("kseiorum"); // k: kurtosis, s: skewness, i:
        integral, o: overflows, u: underflows, r: RMS, m: mean, e:
        number of entries

    // Desenha o histograma
    hist->Draw();

    // Salva o gr fico em PDF
    canvas->Print("exerc cio_3.png");

    // Limpa a mem ria
    delete canvas;
    delete hist;
}
```

A.5 Código 5: Exercícios 4

Código do Exercícios 4

```
void exer_4() {
    // 1. Abrir o arquivo e o TTree
    TFile *file = TFile::Open("tree.root");
    TTree *tree = (TTree*)file->Get("tree1");

    // 2. Definir vari veis
    float ebeam, px, py, pz;
    tree->SetBranchAddress("ebeam", &ebeam);
    tree->SetBranchAddress("px", &px);
    tree->SetBranchAddress("py", &py);
    tree->SetBranchAddress("pz", &pz);

    // 3. Calcular a m dia da energia do feixe
    double totalEnergy = 0;
    Long64_t nEntries = tree->GetEntries();

    // Loop para calcular a m dia
    for (Long64_t i = 0; i < nEntries; i++) {
        tree->GetEntry(i);
        totalEnergy += ebeam; // Soma a energia do feixe
    }

    // Verifique se     maior que 0
    if (nEntries > 0) {
        double meanEnergy = totalEnergy / nEntries; // Calcula a
            m dia
        double lowerCut = meanEnergy - 0.2;
        double upperCut = meanEnergy + 0.2;

        // Criar um objeto TCut para os cortes
        TCut cut = Form("ebeam␣<␣%f␣||␣ebeam␣>␣%f", upperCut,
            lowerCut);

        // 4. Criar o histograma
        TH1F *histogram = new TH1F("h_total_momentum", "Total␣
            Momentum␣Distribution", 200, 130, 160);

        // 5. Preencher o histograma usando TTree::Draw
        tree->Draw("TMath::Sqrt(px*px␣+␣py*py␣+␣pz*pz)>>
            h_total_momentum", cut);

        // 6. Criar um canvas para desenhar o histograma
        TCanvas *c2 = new TCanvas("c2", "Total␣Momentum␣Distribution"
            , 800, 600);
        histogram->Draw();

        // 7. Salvar o histograma em um arquivo PDF
        c2->SaveAs("exerc cio_4.png");
    }

    // 8. Fechar o arquivo original
    file->Close(); // Fecha o arquivo original
}
```