Universidade do Estado do Rio de Janeiro

Jully Moura

Atividade 3

Rio de Janeiro
2024

Exercício 01

```
//Importando bibliotecas
#include <iostream>
#include <vector>
#include <TCanvas.h>
#include <TF1.h>
#include <TLegend.h>
#include <TMath.h>

void atv1() {

    //Definindo funcao de seno
    auto fseno = [](double *x, double *par) {
        double p0 = par[0];
        double p1 = par[1];
        return (x[0] == 0) ? p0 * p1 : p0 * sin(p1 * x[0]) / x[0];
    };

    // Definindo parametros
    std::vector<std::pair<double, double>> parameters = {
        {0.5, 1.5},
        {1.0, 2.0},
        {2.0, 4.0},
        {1.5, 6}
    };

    // Criando loop para cada par de p0 e p1
    for (const auto& param : parameters) {
        double p0 = param.first;
        double p1 = param.second;

        // Criando  canvas para cada grafico
        TCanvas *canvas = new TCanvas(Form("canvas_p0_%.1f_p1_%.1f", p0, p1), "Funcao
            Parametrica", 800, 600);
        TLegend *legend = new TLegend(0.7, 0.7, 0.9, 0.9);

        // Criando a funcao
        TF1 *func = new TF1("func", fseno, 0, 5, 2);

        //Configurando parametros
        func->SetParameters(p0, p1);

        // Desenhando a fun   o no canvas
```

```
        func->SetLineColor(kBlue);
        func->Draw();

        // C lculando os valores
        std::cout << "p0 = " << p0 << ", p1 = " << p1 << ":\n";
        std::cout << "Valor em x=1: " << func->Eval(1.0) << "\n"; // Valor da fun   o
        std::cout << "Derivada em x=1: " << func->Derivative(1.0) << "\n"; // Derivada
        std::cout << "Integral de 0 a 3: " << func->Integral(0, 3) << "\n"; // Integral

        // Adicionando legenda
        legend->AddEntry(func, Form("p0 = %.1f, p1 = %.1f", p0, p1), "l");
        legend->Draw();

        // Salvando a imagem gerada
        canvas->SaveAs(Form("Plot_p0_%.1f_p1_%.1f.png", p0, p1));

    }
}
```

Resultados

- Para p0 = 1 e p1 = 2:

Valor em x=1: 0.909297

Derivada em x=1: -1.74159

Integral de 0 a 3: 1.42469

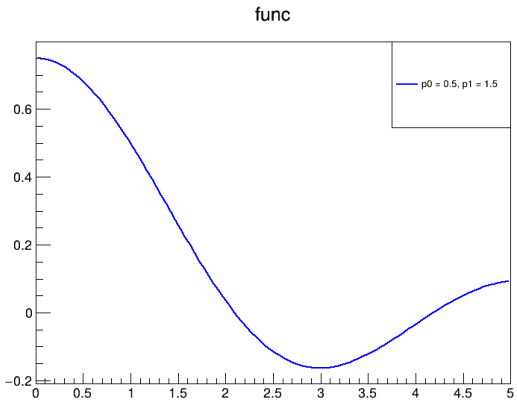

Figure 1: 0.5 a 1.5

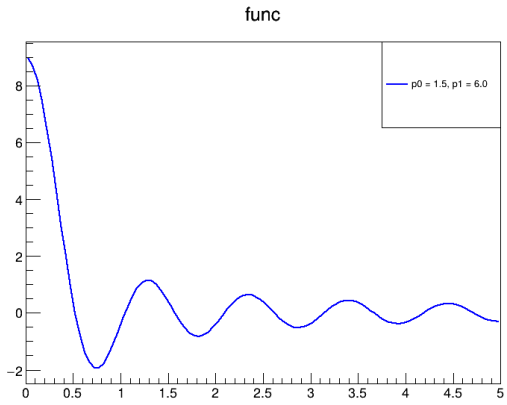

Figure 2: 1.5 a 6.0

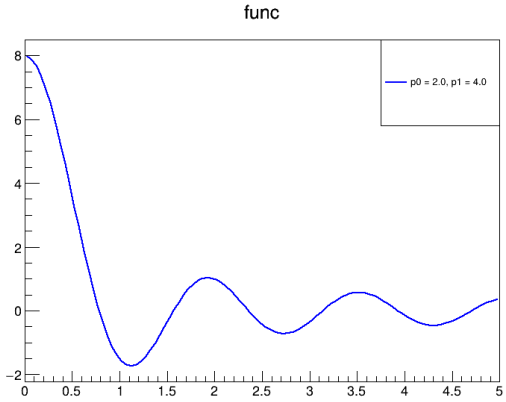

Figure 3: 2.0 a 4.0

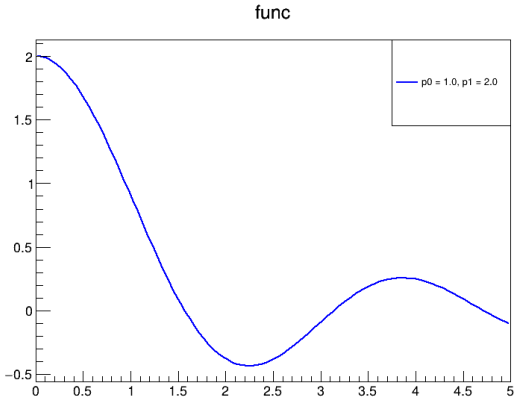

Figure 4: 1.0 a 2.0

Exercício 02

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

void atv2() {
    // Criando canvas
    TCanvas *canvas = new TCanvas("canvas", "Grafico de erros", 800, 600);

    // Criando TGraph para os dados
    TGraph *graph = new TGraph("graphdata.txt");
    TGraphErrors *graphErrors = new TGraphErrors("graphdata_error.txt");

    // Configurando o plot
    graph->SetMarkerStyle(22);
    graph->SetTitle("Grafico com Erros;X;Y");
    graph->Draw("ALP"); // Desenha os pontos no gr fico com linhas

    // Desenhando os erros no gr fico
    graphErrors->SetMarkerStyle(21);
    graphErrors->SetMarkerColor(kGreen);
    graphErrors->Draw("P");

    // Salvando a imagem gerada
    canvas->SaveAs("Grafico.png");

}
```

Resultados

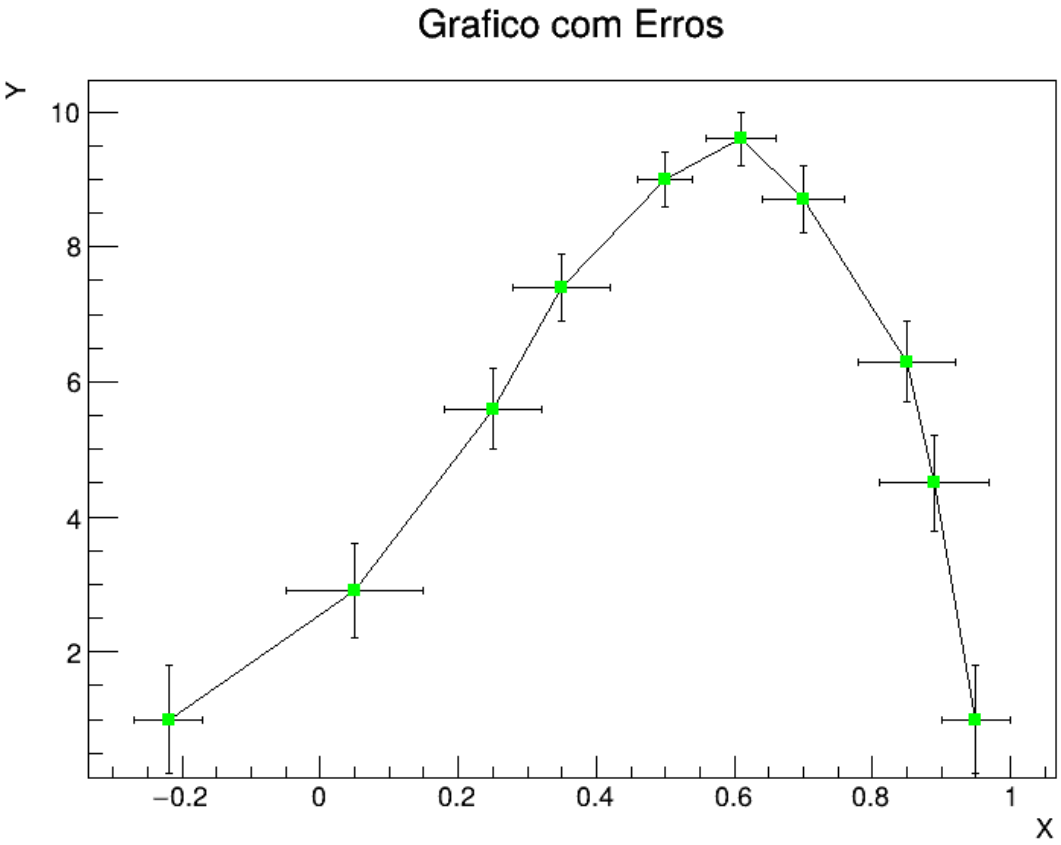

Figure 5: Gráfico com Erros

Exercício 03

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

void atv3() {

    //Criando histograma
    TH1F *histograma = new TH1F("histograma", "Histograma Gaussiana", 50, 0, 10);

    //Criando objeto random
    TRandom *random = new TRandom();

    // Preenchendo 10.000 n meros aleat rios distribu dos com gaussiana de m dia 5 e
        sigma 2.
    for (int i = 0; i < 10000; i++) {
        double value = random->Gaus(5, 2);
        histograma->Fill(value);
    }

    // Criando um canvas
    TCanvas *canvas = new TCanvas("canvas", "Gaussiana como numero aleatorios", 1000,
        600);

    //Adicionando caixa de informa  es
    gStyle->SetOptStat("kseiorum");

    // Desenhando o histograma
    histograma->Draw();

    // Salvando a imagem gerada
    canvas->SaveAs("HistoGaussRandom.png");
}
```

Resultados

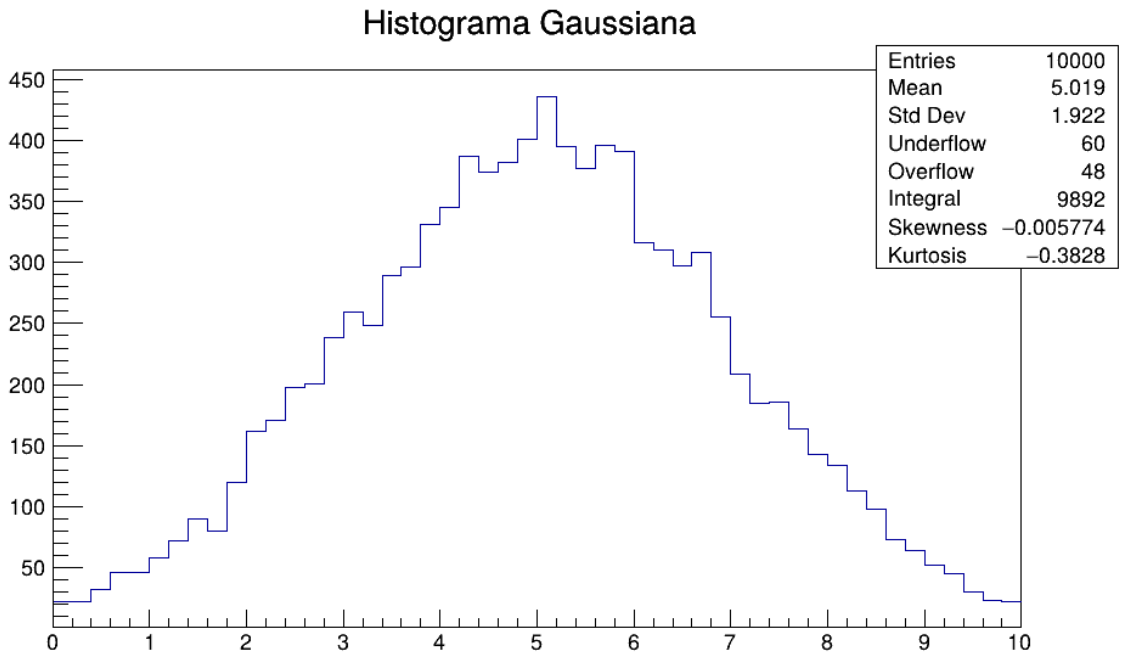

Figure 6: Histograma Gaussiana

Exercício 04

```
void atv4() {

    // Abrindo o arquivo
    TFile *file = TFile::Open("tree.root");
    //Acessando a tree
    TTree *tree = (TTree*)file->Get("tree1");

    // Definindo as Vari veis
    float ebeam, px, py, pz;
    tree->SetBranchAddress("ebeam", &ebeam);
    tree->SetBranchAddress("px", &px);
    tree->SetBranchAddress("py", &py);
    tree->SetBranchAddress("pz", &pz);

    // C lculando a m dia
    double totalEnergy = 0;
    Long64_t nEntries = tree->GetEntries();

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

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

    // Criando os cortes
        TCut cut = Form("ebeam < %f || ebeam > %f", upperCut, lowerCut);

    // Criando o histograma
    TH1F *histograma = new TH1F("Momento_total", "Distribuicao de momento total", 200,
        130, 160);

    //Desenhando a tree no histograma
    tree->Draw("TMath::Sqrt(px*px + py*py + pz*pz)>>Momento_total", cut);

    // Criando um canvas
    TCanvas *canvas = new TCanvas("canvas", "Distribuicao de momento total", 800, 600);
    histograma->Draw();
```

```
42
43      // Salvando arquivo gerado em PDF
44       canvas->SaveAs("Momento_total.pdf");
45
46       }
47
48       file->Close();
49  }
```

Resultados

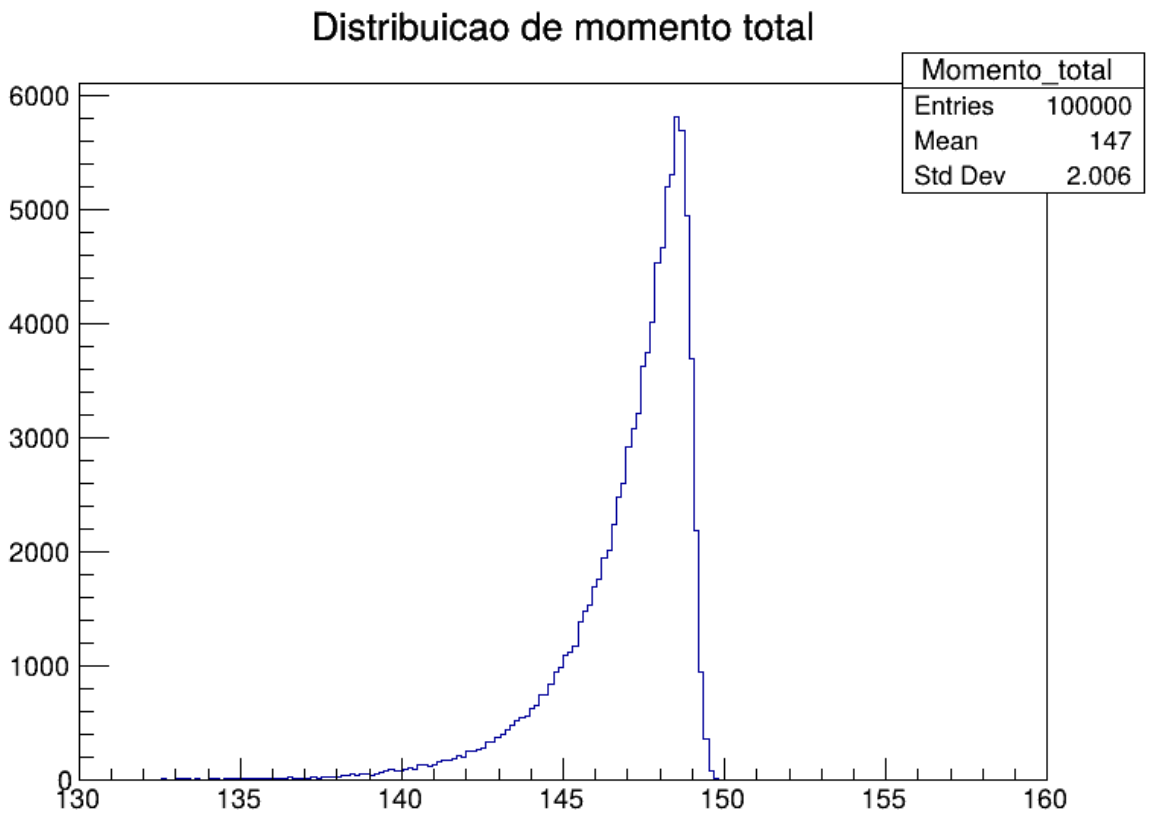

Figure 7: Distribuição momento total