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

Exercícios - Aula ROOT

Professores: Dilson, Eliza & Maurício *Name:* Diogo Gomes Santos Caffonso de Morais

EXERCÍCIO 1

O código para o exercício foi:

```
#include <iostream>
#include <cmath>
#include "TCanvas.h"
#include "TF1.h"
#include "TGraph.h"
#include "TAxis.h"
#include "TMath.h"
#include "Math/Integrator.h"

double myFunc(double *x, double *par) {
    double p0 = par[0];
    double p1 = par[1];
    return p0 * TMath::Sin(p1 * x[0]) / x[0];
}

void exercicio1() {
        double p0 = 1.0;
        double p1 = 2.0;
        double x = 1.0;

        TCanvas *c1 = new TCanvas("c1", "Function Plot", 800, 600);
        TF1 *f1 = new TF1("f1", myFunc, 0.01, 10, 2);
        f1->SetParameters(p0,p1);
        f1->SetLineColor(kBlue);
        f1->Draw();
        c1->SaveAs("function_plot.png");

        double func_value = f1->Eval(x);
        std::cout << "Function value at x = 1: " << func_value << std::endl;

        double dx = 1e-6;
        double derivada = (f1->Eval(x + dx) - f1->Eval(x - dx)) / (2 * dx);
        std::cout << "Derivative at x = 1: " << derivada << std::endl;

        double integral = f1->Integral(0, 3);
        std::cout << "Integral from 0 to 3: " << integral << std::endl;
}
```

Executado o código acima, tive o seguinte output:

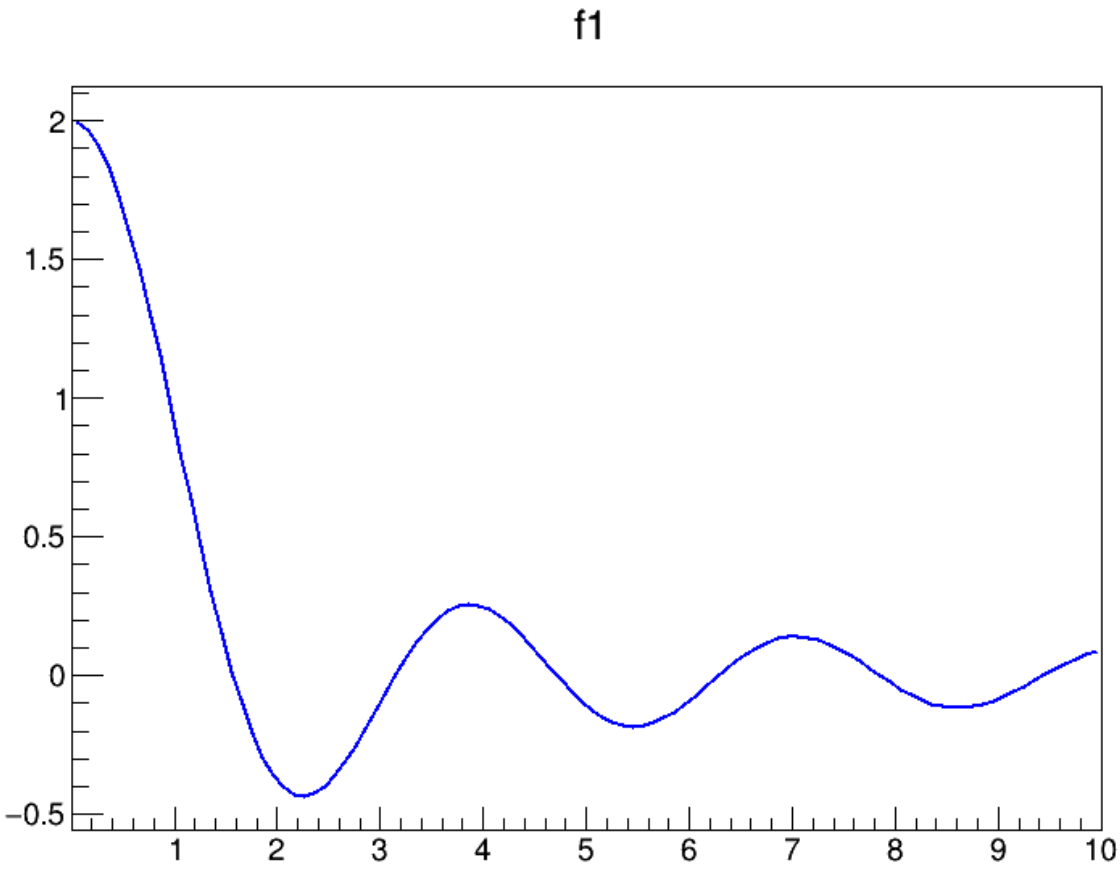

Figura 1: Gráfico da função $f_1(x)$.

Seguindo o comando do exercício:

$$f_1(1) = 0,909297$$

$$f_1^{'}(1) = -1,74159$$

$$\int_0^3 f_1(x)dx = 1,42469$$

EXERCÍCIO 2

O código para o exercício foi:

```
1
2  #include <iostream>
3  #include <cmath>
4  #include "TCanvas.h"
5  #include "TF1.h"
6  #include "TGraph.h"
7  #include "TAxis.h"
8  #include "TMath.h"
9  #include "Math/Integrator.h"
10 #include "TGraphErrors.h"
11 #include "TStyle.h"
12
13 void exercicio2() {
14         std::ifstream dataFile("graphdata.txt");
15         std::ifstream errorFile("graphdata_error.txt");
16
17         const int nPoints = 10;
18         double x[nPoints], y[nPoints];
19         double ex[nPoints], ey[nPoints];
20
21         for (int i = 0; i < nPoints; ++i) {
22                 dataFile >> x[i] >> y[i];
23         }
24
25         for (int i = 0; i < nPoints; ++i) {
26                 errorFile >> x[i] >> y[i] >> ex[i] >> ey[i];
27         }
28
29         dataFile.close();
30         errorFile.close();
31
32         TCanvas *c1 = new TCanvas("c1", "Graph of Points", 800, 600);
33         TGraph *graph1 = new TGraph(nPoints, x, y);
34         graph1->SetMarkerStyle(21);
35         graph1->SetMarkerColor(kBlack);
36         graph1->SetTitle("Graph of Points;X;Y");
37         graph1->Draw("AP");
38         c1->SaveAs("graph_points.png");
39
40         TCanvas *c2 = new TCanvas("c2", "Graph with Line", 800, 600);
41         TGraph *graph2 = new TGraph(nPoints, x, y);
42         graph2->SetMarkerStyle(21);
43         graph2->SetMarkerColor(kBlack);
44         graph2->SetLineColor(kBlue);
45         graph2->SetLineWidth(2);
46         graph2->SetTitle("Graph with Line;X;Y");
47         graph2->Draw("APL");
48         c2->SaveAs("graph_with_line.png");
49
50         TCanvas *c3 = new TCanvas("c3", "Graph with Errors", 800, 600);
51         TGraphErrors *graph3 = new TGraphErrors(nPoints, x, y, ex, ey);
52         graph3->SetMarkerStyle(21);
53         graph3->SetMarkerColor(kBlack);
54         graph3->SetLineColor(kRed);
55         graph3->SetLineWidth(2);
56         graph3->SetTitle("Graph with Errors;X;Y");
57         graph3->Draw("AP");
58         c3->SaveAs("graph_with_errors.png");
59 }
```

Executado o código acima, tive o seguinte output:

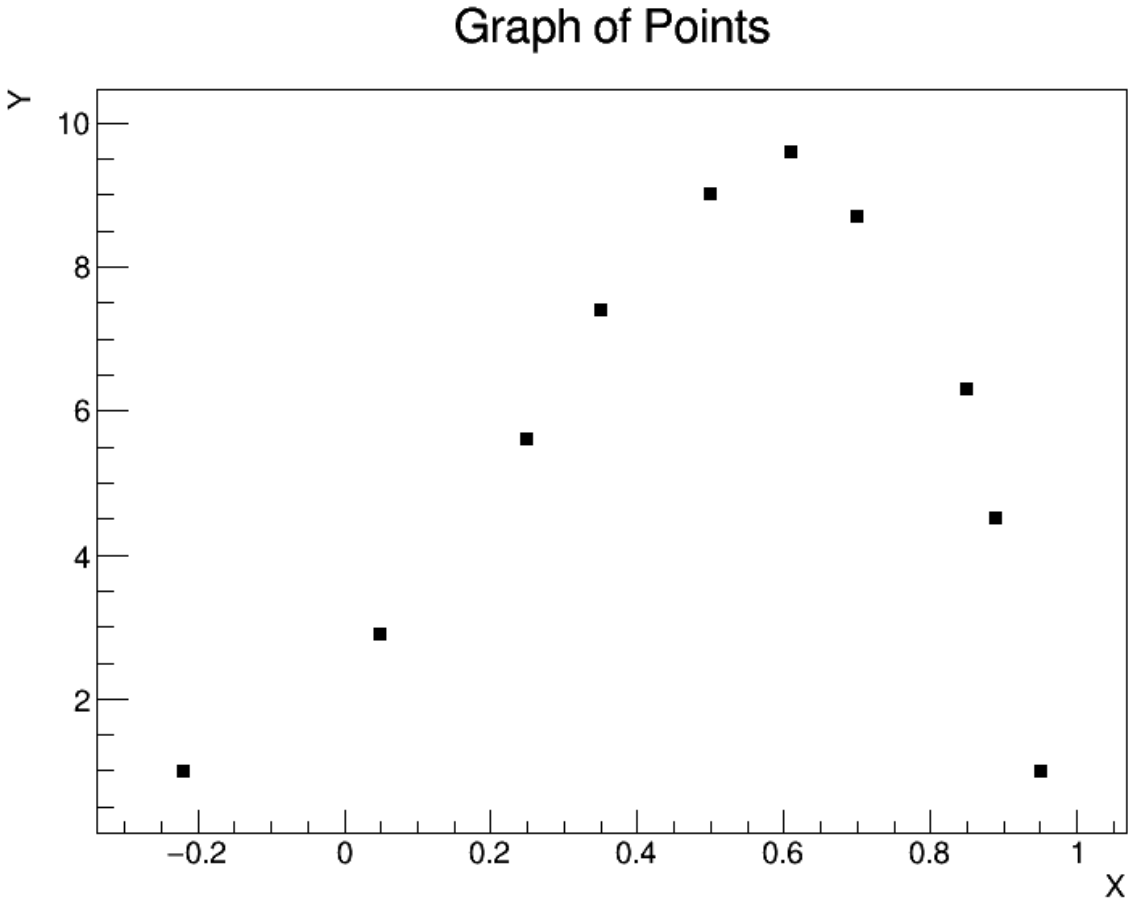

Figura 2: Gráfico de distribuição para os pontos.

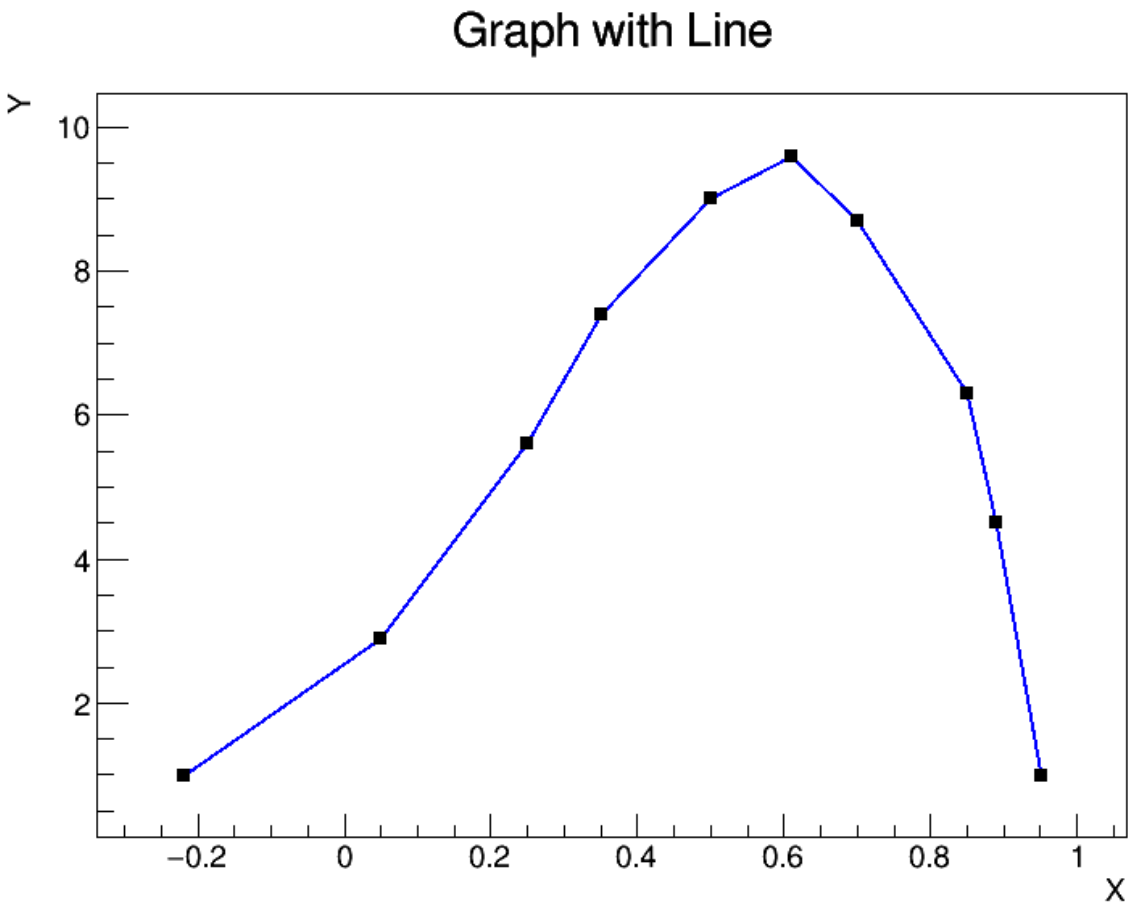

Figura 3: Gráfico de distribuição para os pontos, com linha que liga os pontos.

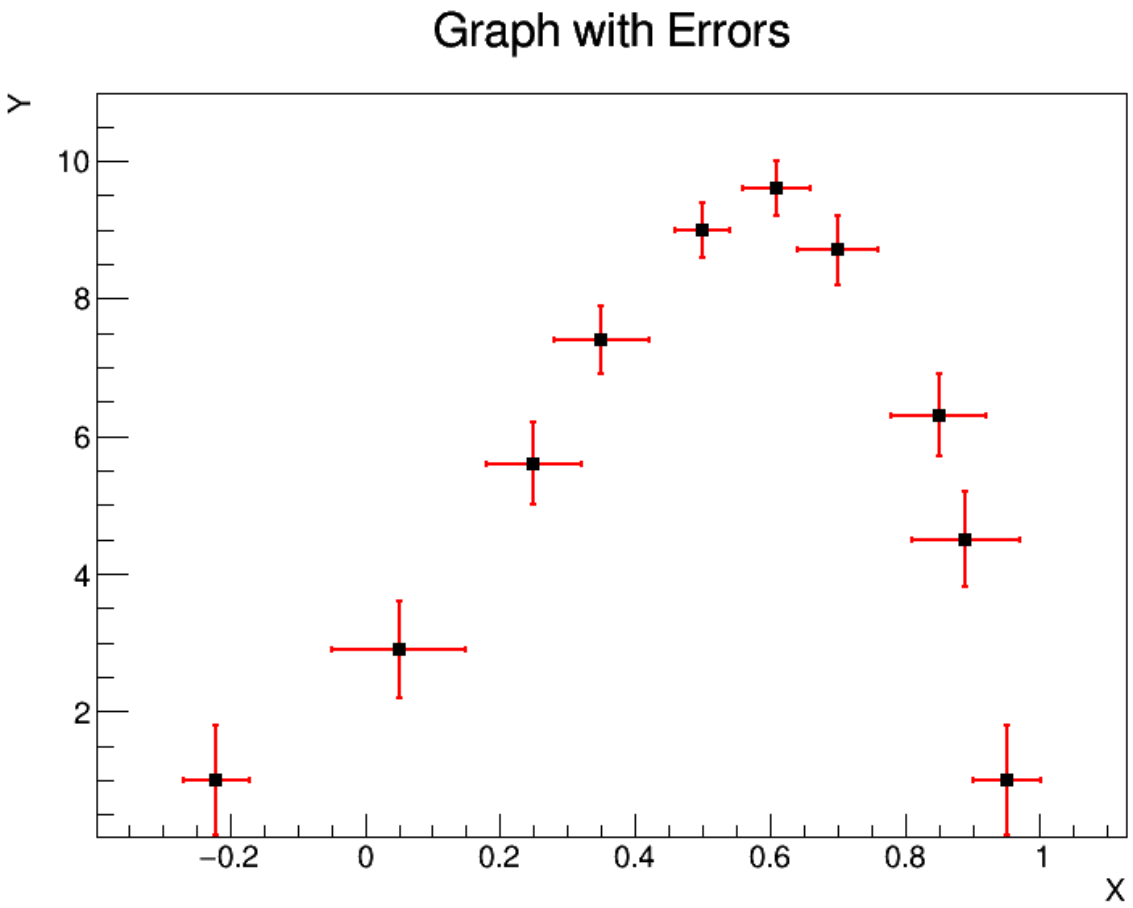

Figura 4: Gráfico de distribuição para os pontos, com barra de erros.

EXERCÍCIO 3

O código para o exercício foi:

```
#include <iostream>
#include <cmath>
#include "TCanvas.h"
#include "TF1.h"
#include "TGraph.h"
#include "TAxis.h"
#include "TMath.h"
#include "Math/Integrator.h"
#include "TRandom.h"

void exercicio3() {

        TH1F *h1 = new TH1F("h1", "Gaussian Distribution", 50, 0, 10);

        TRandom *rand = new TRandom();
        for (int i = 0; i < 10000; i++) {
                h1->Fill(rand->Gaus(5, 2));  // Mean 5, Sigma 2
        }

        gStyle->SetOptStat("nemruoiks");
        gStyle->SetStatX(0.9);
        gStyle->SetStatY(0.9);
        gStyle->SetStatW(0.2);
        gStyle->SetStatH(0.3);

        TCanvas *c1 = new TCanvas("c1", "Gaussian Histogram", 800, 600);

        h1->Draw();

        c1->Update();

        TPaveStats *stats = (TPaveStats*)h1->GetListOfFunctions()->FindObject("stats"
            );

        if (stats) {
                stats->SetName("mystats");

                double skewness = h1->GetSkewness();
                double kurtosis = h1->GetKurtosis();

                stats->AddText(Form("Skewness = %.3f", skewness));
                stats->AddText(Form("Kurtosis = %.3f", kurtosis));

                stats->SetX1NDC(0.7);
                stats->SetY1NDC(0.7);

                c1->Modified();
                c1->Update();
        }

        c1->SaveAs("gaussian_histogram.png");
}
```

Executado o código acima, tive o seguinte output:

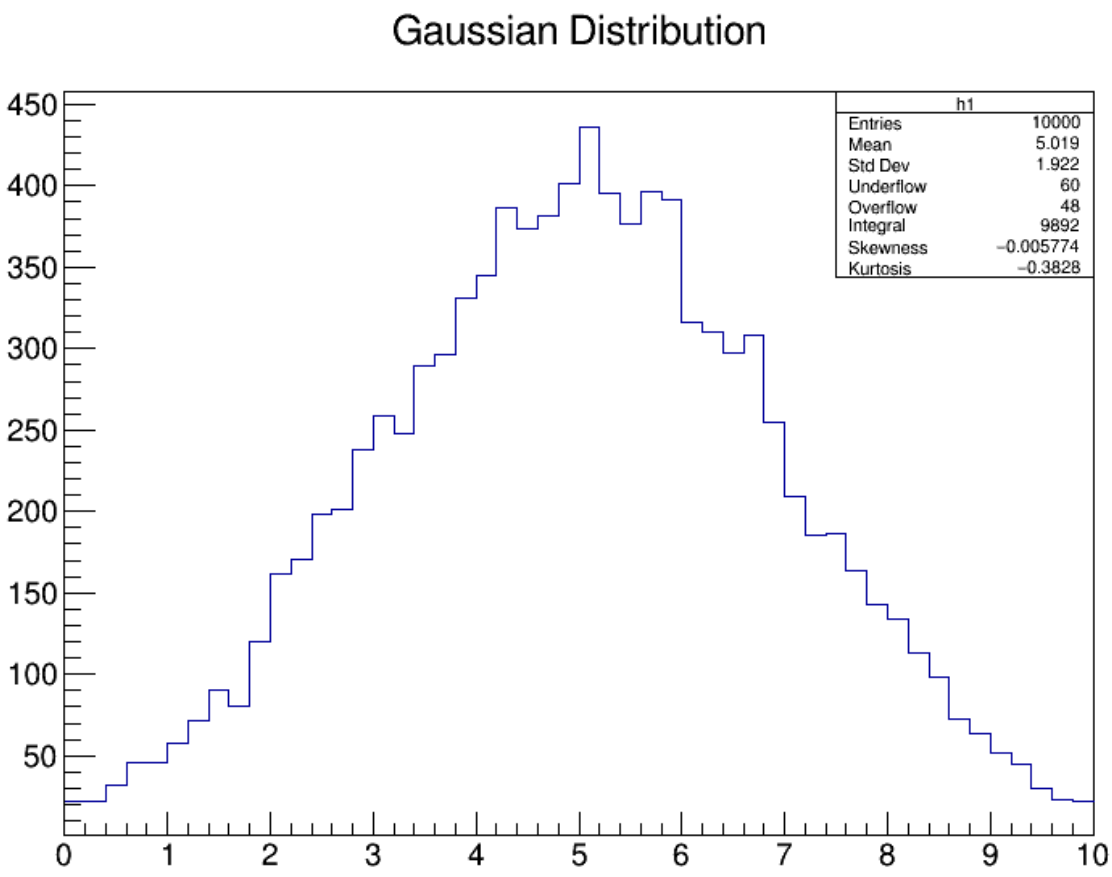

Figura 5: Histograma com os dados no statistic box.

EXERCÍCIO 4

O código para o exercício foi:

```
#include <iostream>
#include <cmath>
#include "TCanvas.h"
#include "TF1.h"
#include "TGraph.h"
#include "TAxis.h"
#include "TMath.h"
#include "Math/Integrator.h"
#include "TRandom.h"

void exercicio4() {

        TCanvas *c1 = new TCanvas("c1", "Total Momentum Distribution", 800, 600);
        TFile *file = TFile::Open("tree.root");
        TTree *tree = (TTree*)file->Get("tree1");
        TH1F *h1 = new TH1F("h1", "Total Momentum Distribution", 50, 0, 1000);
        TH1F *h2 = new TH1F("h2", "Cut Momentum Distribution", 100, 100, 200);
        float px, py, pz, e;
        Int_t noE = 1e+5;

        tree->SetBranchAddress("px", &px);
        tree->SetBranchAddress("py", &py);
        tree->SetBranchAddress("pz", &pz);
        tree->SetBranchAddress("ebeam", &e);

        for (Int_t i=0; i<noE; i++){
                tree->GetEntry(i);
                h1->Fill(e);
        }

        float e_mean = h1->GetMean();
        for (Int_t i=0; i<noE; i++){
                tree->GetEntry(i);
                if (e > e_mean + 0.2){
                        float p = sqrt(px*px + py*py + pz*pz);
                        h2->Fill(p);
                }
        }

        h2->Draw();
        c1->SaveAs("total_momentum_distribution.png");

}
```

Executado o código acima, tive o seguinte output:

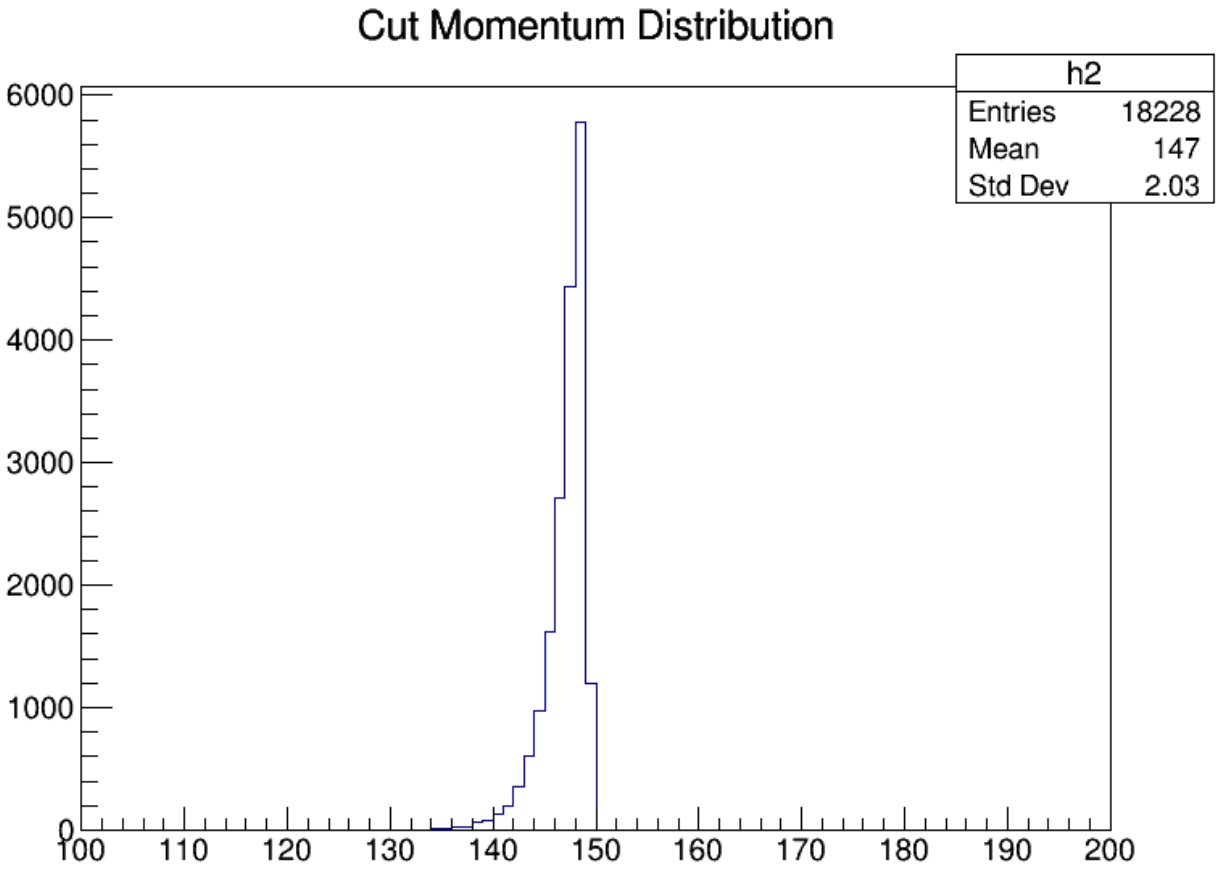

Figura 6: Histograma de momento total para a janela de energia sugerida.