

*Professores:* Dilson de Jesus Damião, Eliza Melo da Costa e Mauricio Thiel

*Name:* Bruno Kron Guandalini e Pedro Oliveira

**EXERCICIO 1**

Crie uma função com parâmetros,  $p_0 \cdot \frac{\sin(p_1 \cdot x)}{x}$ , e desenhe-a para diferentes valores de parâmetros. Defina a cor da função paramétrica como azul. Após desenhar a função, calcule para os valores dos parâmetros ( $p_0 = 1$ ,  $p_1 = 2$ ):

1. Valor da função para  $x = 1$
2. Derivada da função para  $x = 1$
3. Integral da função entre 0 e 3

RESPOSTA:

```
1  #include <TCanvas.h>
2  #include <TMath.h>
3  #include <TF1.h>
4
5  void exercicio_1() {
6
7      double p0 = 1, p1 = 2;
8
9      TF1 *f = new TF1("f", "[0] * sin([1] * x) / x", 0.1, 10);
10     f->SetParameters(p0, p1);
11
12     TCanvas *c1 = new TCanvas("c1", "Fun o", 800, 600);
13     f->SetLineColor(kBlue);
14     f->Draw();
15
16     c1->SaveAs("exercicio_1.png");
17 }
18
19 void calcu_val() {
20
21     double p0 = 1, p1 = 2;
22     TF1 *f = new TF1("f", "[0] * sin([1] * x) / x", 0.1, 10);
23     f->SetParameters(p0, p1);
24
25     double x_value = 1;
26     double func_value = f->Eval(x_value);
27     std::cout << "Valor da fun o para x = " << x_val << ": " << func_val << std::
        endl;
28
29     double derivative_value = f->Derivative(x_value);
30     std::cout << "Derivada da fun o para x = " << x_val << ": " << der_val << std
        ::endl;
31
32     double integral_value = f->Integral(0, 3);
33     std::cout << "Integral da fun o entre 0 e 3: " << integral_val << std::endl;
```

34 }

1. Valor da função para  $x = 1$ : 0.909297
2. Derivada da função para  $x = 1$ : -1.74159
3. Integral da função entre 0 e 3: 1.42469

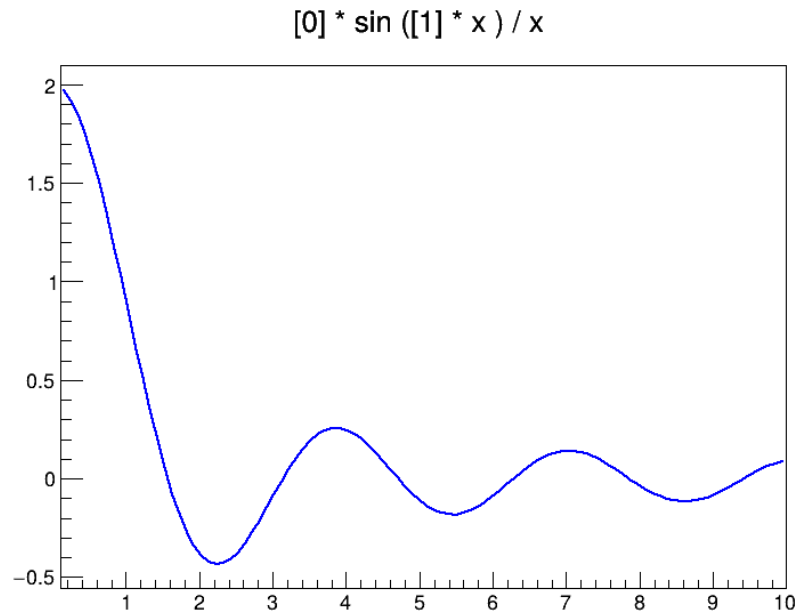


Figura 1: xxxxxx

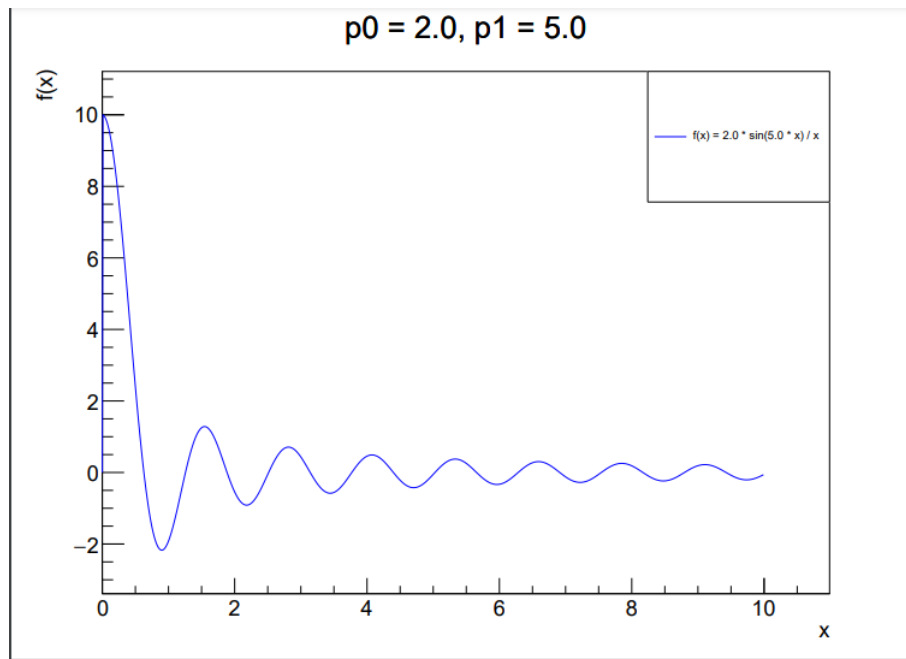


Figura 2: xxxxxx

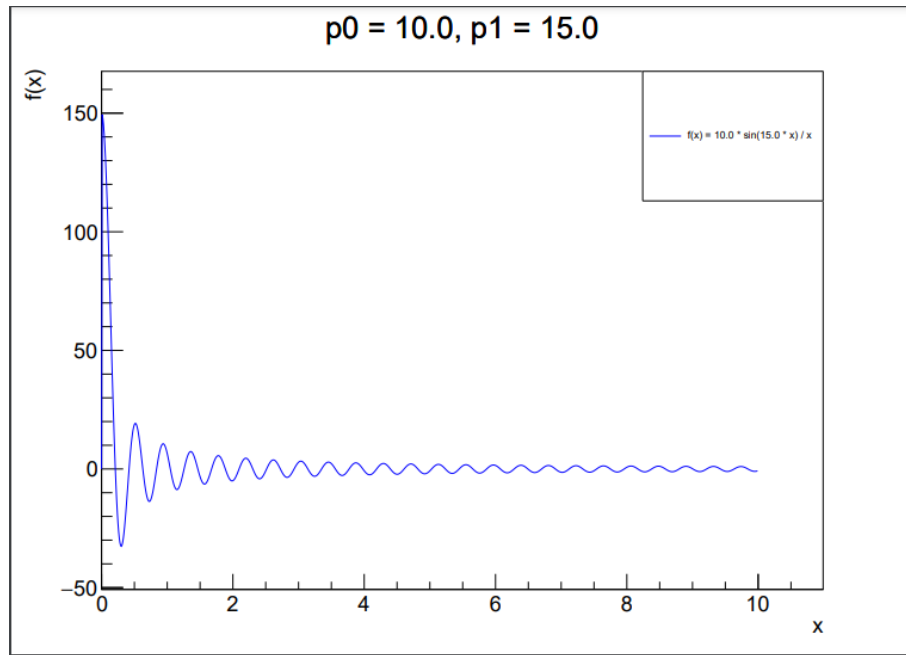


Figura 3: xxxxxx

**EXERCICIO 2**

Suponha que você tenha este conjunto de pontos definido no arquivo anexado `graphdata.txt`. Plote esses pontos utilizando a classe `TGraph`. Use como marcador de ponto uma caixa preta. Observando as opções possíveis para desenhar o `TGraph` no `TGraphPainter`, plote uma linha conectando os pontos. Crie um `TGraphError` e exiba-o utilizando o conjunto de dados anexado, `graphdata_error.txt`, contendo erro em  $x$  e  $y$ .

RESPOSTA:

```

1  #include <TGraph.h>
2  #include <TGraphErrors.h>
3  #include <TCanvas.h>
4  #include <TStyle.h>
5  #include <iostream>
6
7  void exercicio_2() {
8
9      TCanvas *canvas = new TCanvas("canvas", "Graph with Errors", 800, 600);
10
11
12      TGraph *graph = new TGraph("graphdata.txt");
13      if (!graph) {
14          std::cerr << "Erro ao carregar graphdata.txt" << std::endl;
15          return;
16      }
17
18
19      TGraphErrors *graphErrors = new TGraphErrors("graphdata_error.txt");
20      if (!graphErrors) {
21          std::cerr << "Erro ao carregar graphdata_error.txt" << std::endl;
22          return;
23      }
24
25
26      graph->SetMarkerStyle(22);
27      graph->SetMarkerColor(kBlack);
28      graph->SetTitle("Grafico com Erros;eixo-X;eixo-Y");

```

```
29 graph->Draw("ALP");
30
31
32 graphErrors->SetMarkerStyle(21);
33 graphErrors->SetMarkerColor(kRed);
34 graphErrors->Draw("P");
35
36
37 canvas->Print("exerc cio_2.pdf");
38
39
40 }
41
42 void grafico() {
43     graph();
44 }
```

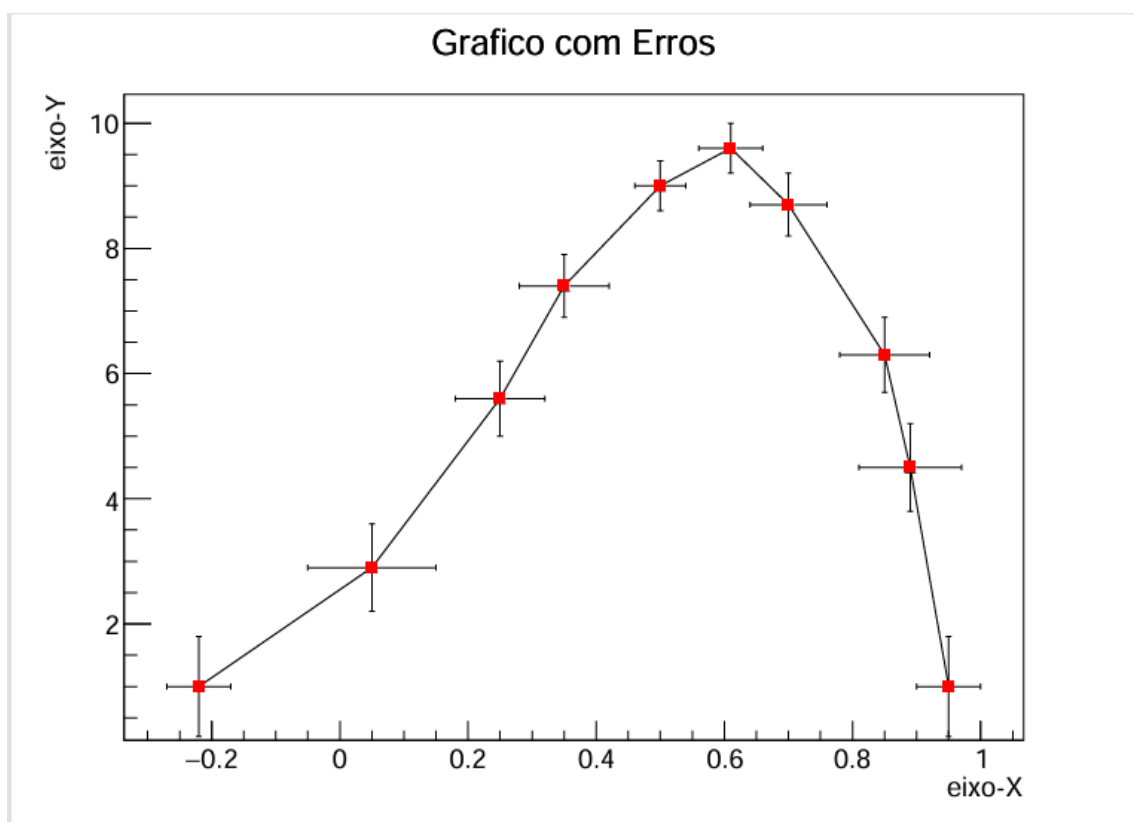


Figura 4: xxxxxx

**EXERCICIO 3**

Crie um histograma unidimensional com 50 intervalos entre 0 e 10, e preencha-o com 10.000 números aleatórios distribuídos de forma gaussiana com média 5 e desvio padrão 2. Plote o histograma e, observando a documentação no `THistPainter`, mostre na caixa de estatísticas o número de entradas, a média, o RMS, a integral do histograma, o número de underflows, o número de overflows, a assimetria e a curtose.

RESPOSTA:

```

1  #include <TH1F.h>
2  #include <TCanvas.h>
3  #include <TRandom3.h>
4  #include <TStyle.h>
5  #include <TApplication.h>
6
7  void exercicio_3() {
8      TRandom3 random;
9
10     TH1F *hist = new TH1F("hist", "Histogram of Gaussian Random Numbers;Value;Entries", 50, 0, 10);
11
12     for (int i = 0; i < 10000; i++) {
13         double value = random.Gaus(5, 2);
14         hist->Fill(value);
15     }
16
17     TCanvas *canvas = new TCanvas("canvas", "Gaussian Histogram", 800, 600);
18
19     gStyle->SetOptStat("kseiorum");
20
21     hist->Draw();
22
23     canvas->Print("exerc cio_3.pdf");
24 }

```

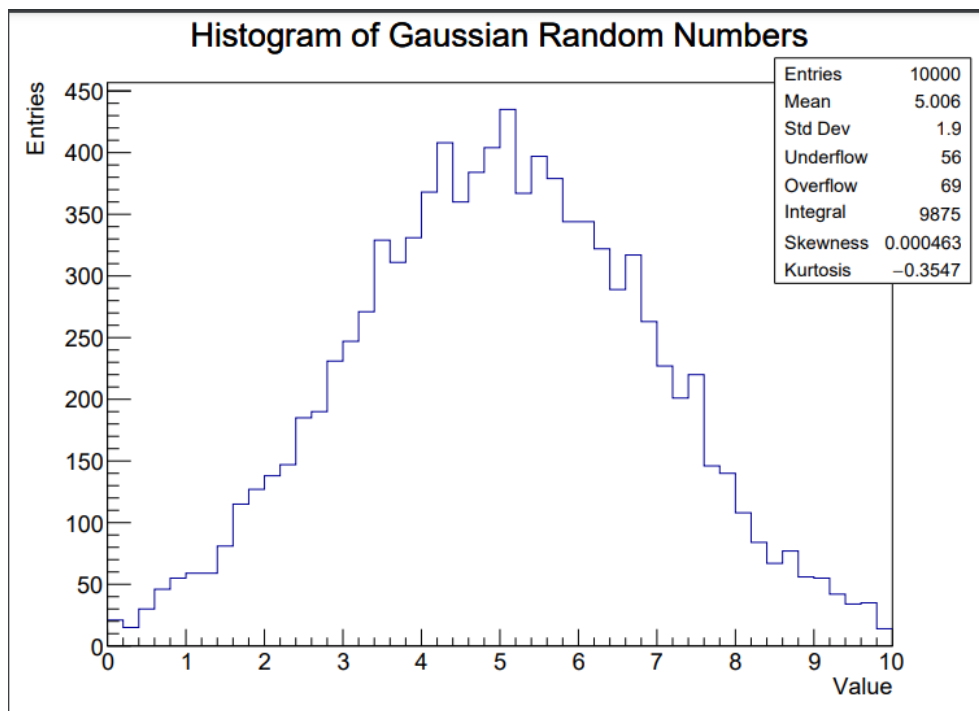


Figura 5: Gaussia

**EXERCICIO 4**

Usando a árvore contida no arquivo `tree.root`, faça uma distribuição do momento total de cada partícula cujo energia do feixe estava fora da média em mais de 0,2. Use objetos `TCut` para fazer a seleção dos eventos. Projete essa distribuição em um histograma, desenhe-o e salve-o em um arquivo.

RESPOSTA:

```

1  #include <TFile.h>
2  #include <TTree.h>
3  #include <TBranch.h>
4  #include <TCut.h>
5  #include <TH1F.h>
6  #include <TMath.h>
7  #include <TCanvas.h>
8  #include <TString.h>
9  #include <iostream>
10
11
12 void exercicio_4() {
13     TFile *file = TFile::Open("tree.root");
14     TTree *tree = (TTree*)file->Get("tree1");
15
16     float ebeam, px, py, pz;
17     tree->SetBranchAddress("ebeam", &ebeam);
18     tree->SetBranchAddress("px", &px);
19     tree->SetBranchAddress("py", &py);
20     tree->SetBranchAddress("pz", &pz);
21
22     double totalEnergy = 0;
23     Long64_t nEntries = tree->GetEntries();
24
25     for (Long64_t i = 0; i < nEntries; i++) {
26         tree->GetEntry(i);
27         totalEnergy += ebeam;
28     }
29
30     if (nEntries > 0) {
31         double meanEnergy = totalEnergy / nEntries;
32         double lowerCut = meanEnergy - 0.2;
33         double upperCut = meanEnergy + 0.2;
34
35         TCut cut = Form("ebeam < %f || ebeam > %f", upperCut, lowerCut);
36
37         TH1F *histogram = new TH1F("h_total_momentum", "Total Momentum Distribution",
38                                     200, 130, 160);
39
40         tree->Draw("TMath::Sqrt(px*px + py*py + pz*pz)>>h_total_momentum", cut);
41
42         TCanvas *c2 = new TCanvas("c2", "Total Momentum Distribution", 800, 600);
43         histogram->Draw();
44
45         c2->SaveAs("exerc cio_4.pdf");
46     }
47     file->Close();
48 }
```

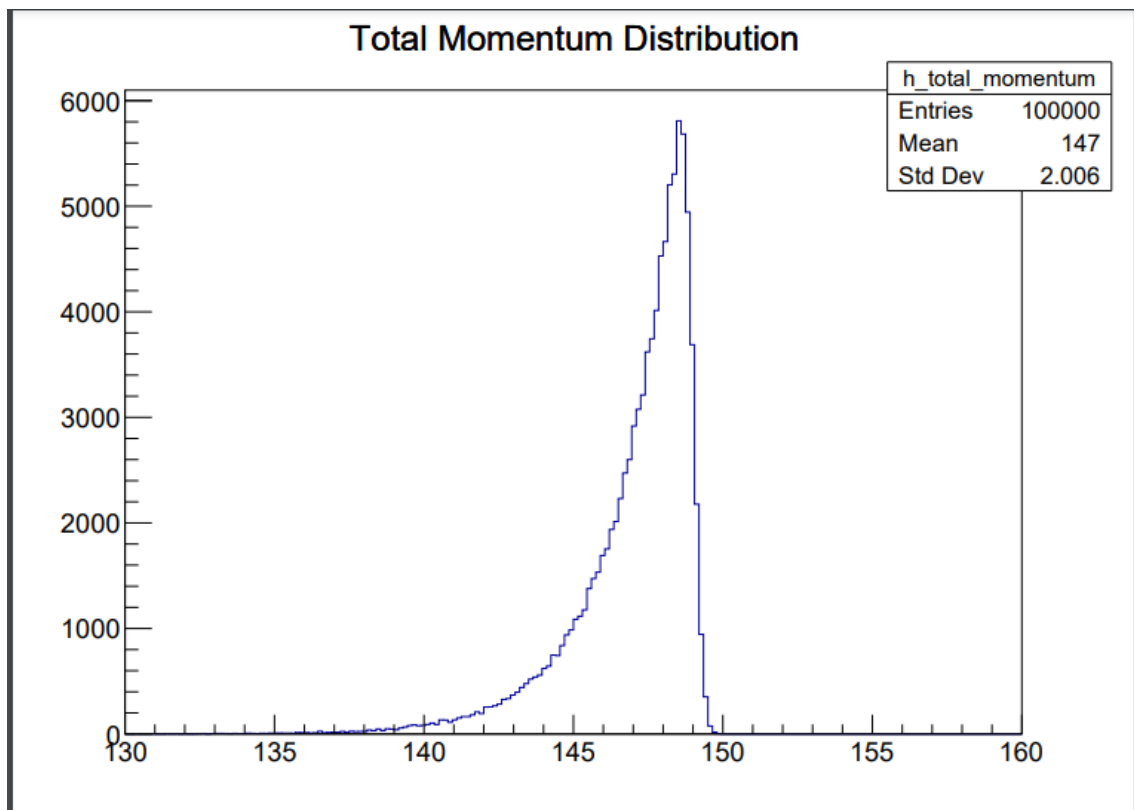


Figura 6: Distribuição do Momentum Total