Introdução à Análise de dados em FAE e tecnologias associadas **(DATA)**

Exercicios dos Slides Introdução ao Root

Professores: Dilson Jesus,Eliza Melo e Mauricio Thiel *Name:* João Marcos Modesto Ribeiro

TEXTO

EXERCICIO 1

Abaixo está o código que define e plota um gráfico de uma função paramétrica usando ROOT e TFunction aprendido nas aulas:

```
#include "TMath.h"
#include "TF1.h"
#include "TCanvas.h"
#include "TLegend.h"
#include "TGraph.h"
#include "TLatex.h"

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

void parfunc() {
    TF1 *f1 = new TF1("f1", parametric_function, 0.1, 3, 2);
    f1->SetParameters(1, 2);
    f1->SetLineColor(kBlue);

    TCanvas *c1 = new TCanvas("c1", "Parametric Function", 1000, 600);
    gPad->SetRightMargin(0.3);
    f1->Draw();

    Double_t x_val = 1;
    Double_t func_value = f1->Eval(x_val);
    Double_t derivative_value = f1->Derivative(x_val);
    Double_t integral_value = f1->Integral(0.1, 3);

    TGraph *graph = new TGraph(1);
    graph->SetPoint(0, x_val, func_value);
    graph->SetMarkerStyle(21);
    graph->SetMarkerSize(1.5);
    graph->SetMarkerColor(kRed);
    graph->Draw("P SAME");

    Double_t arrow_x_end = x_val + 0.5;
    Double_t arrow_y_end = func_value + derivative_value * 0.5;
    TArrow *arrow = new TArrow(x_val, func_value, arrow_x_end, arrow_y_end, 0.02, "|>
        ");
    arrow->SetLineColor(kGreen);
    arrow->Draw();

    f1->SetFillColor(kYellow - 10);
    f1->SetFillStyle(3001);
    f1->Draw("FC SAME");

    TLegend *legend = new TLegend(0.72, 0.7, 0.98, 0.9);
```

```
47      legend->SetHeader("Valores Calculados", "C");
48      legend->AddEntry(f1, Form("f(1) = %.4f", func_value), "l");
49      legend->AddEntry(arrow, Form("f'(1) = %.4f", derivative_value), "l");
50      legend->AddEntry(f1, Form("Integral(0.1, 3) = %.4f", integral_value), "f");
51      legend->Draw();
52
53      TLatex latex;
54      latex.SetTextSize(0.04);
55      latex.DrawLatexNDC(0.15, 0.93, "p0 * sin(p1 * x) / x, com p0 = 1 e p1 = 2");
56
57      c1->Update();
58      c1->SaveAs("parametric_function_plot.png");
59  }
```

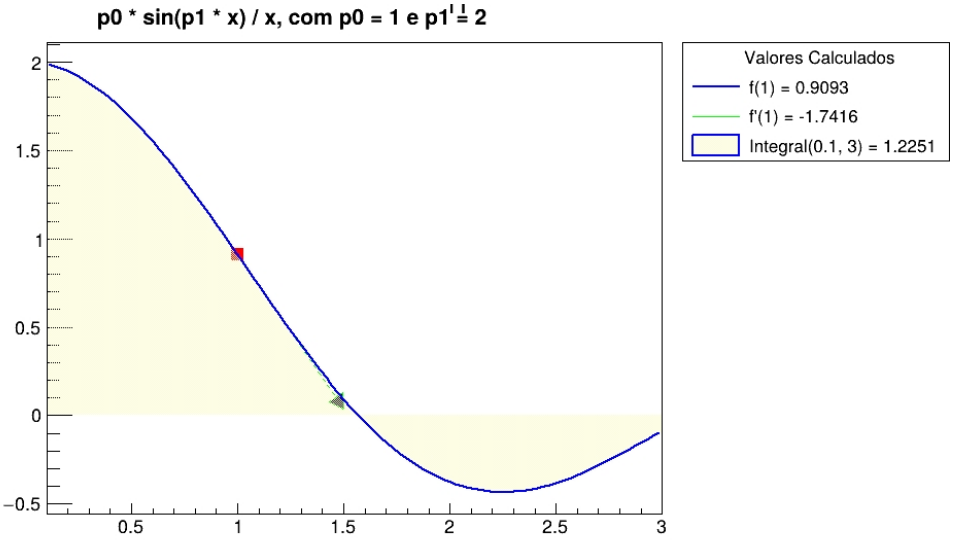


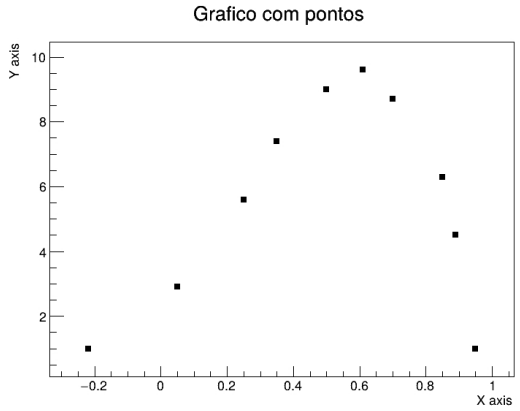
Figura 1: Gráfico da função $p0 \cdot \frac{\sin(p1 \cdot x)}{x}$ com $p0 = 1$ e $p1 = 2$.

EXERCICIO 2

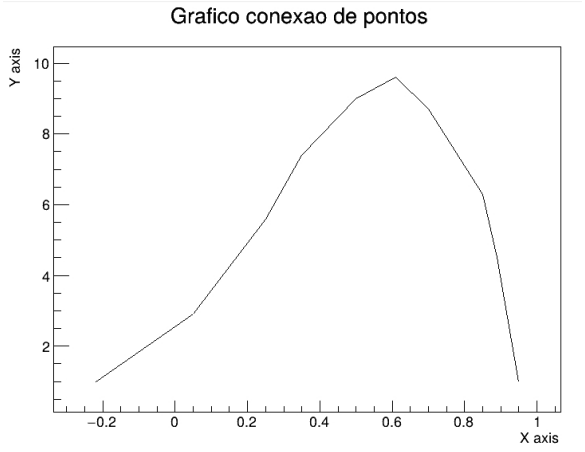
Codigo para o exercicio dos plots:

```
1   #include "TGraph.h"
2   #include "TGraphErrors.h"
3   #include "TCanvas.h"
4   #include "TFile.h"
5   #include "TAxis.h"
6   #include <iostream>
7   #include <fstream>
8   #include <vector>
9
10  void graphconerrors() {
11
12      std::ifstream file("graphdata.txt");
13      std::ifstream file_error("graphdata_error.txt");
14
15
16      std::vector<double> x, y, ex, ey;
17
18      double x_val, y_val, ex_val, ey_val;
19
20
21      while (file >> x_val >> y_val) {
22          x.push_back(x_val);
```

```
23          y.push_back(y_val);
24      }
25
26
27      while (file_error >> ex_val >> ey_val) {
28          ex.push_back(ex_val);
29          ey.push_back(ey_val);
30      }
31
32
33      file.close();
34      file_error.close();
35
36
37      TGraph *graph = new TGraph(x.size(), &x[0], &y[0]);
38      graph->SetMarkerStyle(21);
39      graph->SetMarkerColor(kBlack);
40
41
42      TCanvas *c1 = new TCanvas("c1", "Grafico com pontos sem erros", 800, 600);
43      graph->Draw("AP");
44      graph->SetTitle("Grafico com pontos");
45      graph->GetXaxis()->SetTitle("X axis");
46      graph->GetYaxis()->SetTitle("Y axis");
47
48
49      c1->SaveAs("grafico_dispersao.png");
50
51
52  TCanvas *c2 = new TCanvas("c2", "Grafico conex o sem erros", 800, 600);
53  graph->Draw("AL");
54  graph->SetTitle("Grafico de conex o dos pontos");
55  graph->GetXaxis()->SetTitle("Eixo X");
56  graph->GetYaxis()->SetTitle("Eixo Y");
57  c2->Update();
58  c2->SaveAs("graph_conexao_sem_erros.png");
59
60
61      TCanvas *c3 = new TCanvas("c3", "Graph with Error Bars and Line", 800, 600);
62      TGraphErrors *graphError = new TGraphErrors(x.size(), &x[0], &y[0], &ex[0], &ey
            [0]);
63      graphError->SetMarkerStyle(21);
64      graphError->SetMarkerColor(kRed);
65
66      graph->Draw("APL");
67      graphError->Draw("P same");
68
69
70      graph->SetTitle("Grafico dos dados com erros");
71      graph->GetXaxis()->SetTitle("X axis");
72      graph->GetYaxis()->SetTitle("Y axis");
73
74      c3->Draw();
75      c3->SaveAs("grafico de dados com erros.png");
76  }
```



(a) Dispersão sem erros.



(b) Conexão sem erros.

Figura 2: Gráficos de dispersão e conexão sem erros.

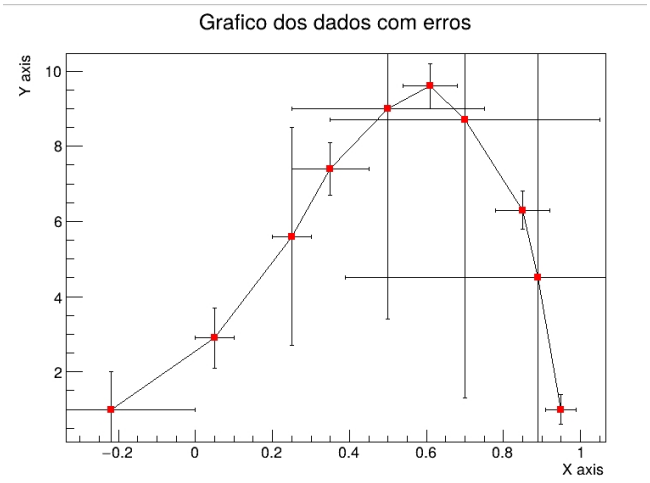


Figura 3: Gráfico de dados com barras de erro.

EXERCICIO 3

```
#include "TCanvas.h"
#include "TH1F.h"
#include "TRandom3.h"
#include "TStyle.h"
#include "TMath.h"
#include "TPaveStats.h"
#include "TText.h"
#include "TF1.h"

void histadapt() {
    TH1F *hist = new TH1F("histograma", "Histograma 1D ajuste", 50, 0, 10);


    TRandom3 rnd;
    for (int i = 0; i < 10000; ++i) {
        double number = rnd.Gaus(5, 2);
        hist->Fill(number);
    }
```

```
19
20
21      TCanvas *c = new TCanvas("canvas", "Histograma ajuste 1D", 800, 600);
22      gStyle->SetOptStat("neMRiou");
23
24      TF1 *gaussFit = new TF1("gaussFit", "gaus", 0, 10);
25      hist->Fit("gaussFit", "R");
26
27      hist->Draw();
28
29      double skewness = hist->GetSkewness();
30      double kurtosis = hist->GetKurtosis();
31
32
33      double mean = hist->GetMean();
34      double stdDev = hist->GetRMS();
35      double meanError = hist->GetMeanError();
36      double stdDevError = hist->GetRMSError();
37
38
39      printf("Mean: %.3f    %.3f\n", mean, meanError);
40      printf("Standard Deviation (RMS): %.3f    %.3f\n", stdDev, stdDevError);
41      printf("Skewness: %.3f\n", skewness);
42      printf("Kurtosis: %.3f\n", kurtosis);
43      printf("Number of Entries: %d\n", (int)hist->GetEntries());
44      printf("Underflows: %d\n", (int)hist->GetBinContent(0));
45      printf("Overflows: %d\n", (int)hist->GetBinContent(hist->GetNbinsX() + 1));
46
47
48      c->Update();
49      TPaveStats *stats = (TPaveStats*)hist->GetListOfFunctions()->FindObject("stats");
50
51      if (stats) {
52          TString meanText = Form("Mean = %.3f    %.3f", mean, meanError);
53          TString stdDevText = Form("Std Dev = %.3f    %.3f", stdDev, stdDevError);
54
55
56          stats->AddText(meanText);
57          stats->AddText(stdDevText);
58          stats->AddText(Form("Skewness = %.3f", skewness));
59          stats->AddText(Form("Kurtosis = %.3f", kurtosis));
60          stats->SetY1NDC(0.55);
61          stats->SetY2NDC(0.85);
62          hist->SetStats(1);
63
64          c->Modified();
65      }
66
67
68      c->Draw();
69      c->SaveAs("Histogram with parameters.png");
70
71  }
```

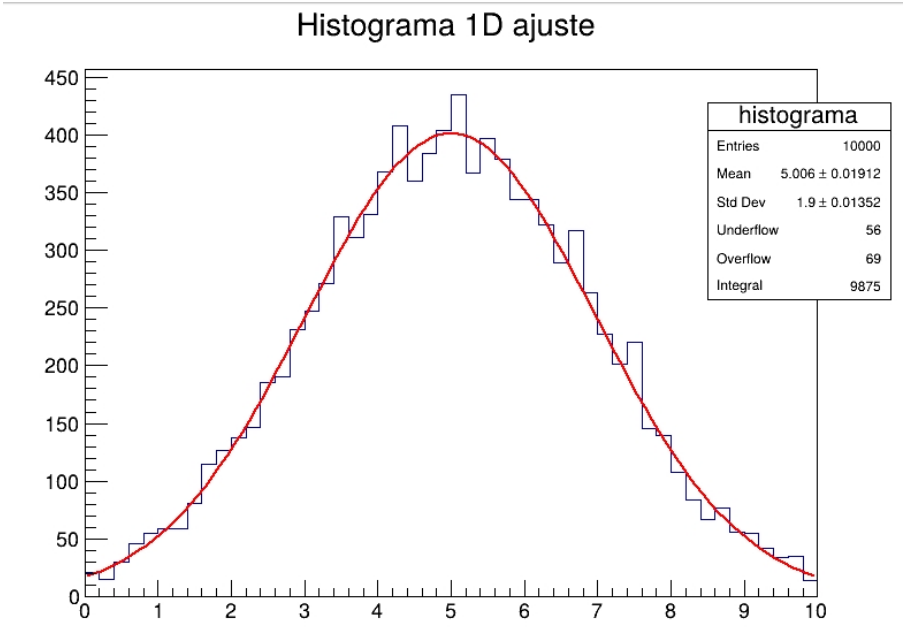


Figura 4: Resultado do histograma com ajuste gaussiano.

EXERCICIO 4

TEXTO

```
void distbeam() {
    TCanvas *c1 = new TCanvas("c1", "Histograma de Momento", 800, 600);

    TFile *file = new TFile("tree.root");
    TTree *tree = (TTree*)file->Get("tree1");

    TH1F *hist = new TH1F("hist", "Distribuicao do Momento Total;Momento [GeV/c];
        Eventos", 100, 0, 1000);
    TH1F *histEbeam = new TH1F("histEbeam", "Distribuicao de Ebeam;Ebeam [GeV];
        Eventos", 100, 0, 1000);

    float px, py, pz, ebeam;

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


    Int_t nEntries = tree->GetEntries();


    for (Int_t i = 0; i < nEntries; i++) {
        tree->GetEntry(i);
        histEbeam->Fill(ebeam);
    }

    // Calcular a m dia da energia do feixe (ebeam)
    float meanEbeam = histEbeam->GetMean();


    for (Int_t i = 0; i < nEntries; i++) {
```

```
31          tree->GetEntry(i);
32          if (ebeam < meanEbeam - 0.2 || ebeam > meanEbeam + 0.2) {
33              float pMagnitude = sqrt(px * px + py * py + pz * pz);
34              hist->Fill(pMagnitude);
35          }
36      }
37
38
39      hist->Draw();
40      c1->SaveAs("histograma_momento.png");
41
42
43      file->Close();
44  }
```

Após a estrutura do codigo e rodá-lo, obtemos o seguinte histograma:

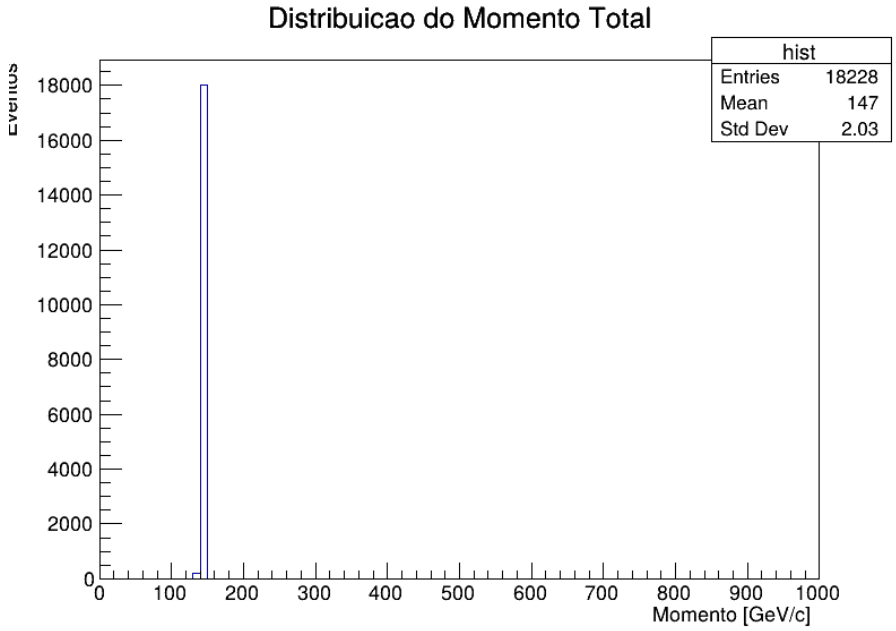


Figura 5: Resultado do histograma do momento total