Introdução à Análise de dados em FAE **(18 de Outubro de 2024)**

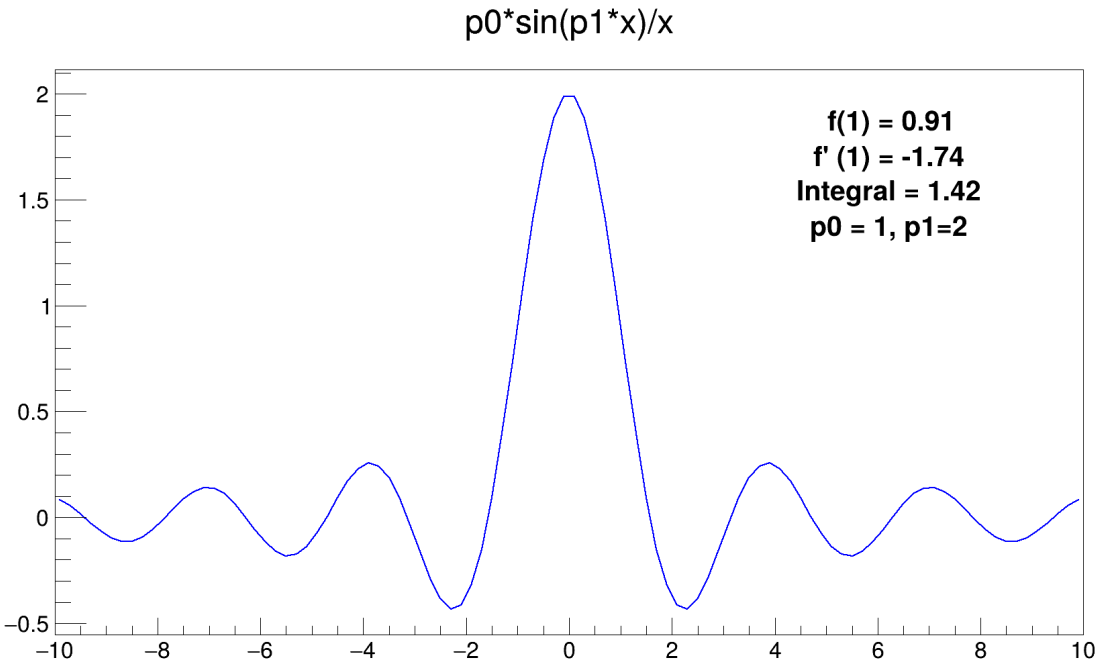
Lista 3

Professores: Dilson Damião, Eliza Melo e Maurício Thiel *Nome:* José Gonçalves Chaves Junior

EXERCICIO 1

Criar o arquivo com a função correspondente:

```
#include "TCanvas.h"
#include "TF1.h"
#include "TMath.h"
#include <iostream>

double f(double *x, double *c_i) {
    return (c_i[0] * TMath::Sin(c_i[1] * x[0])) / x[0];
}

void plotFunction(double p0, double p1) {
    TF1 *func = new TF1("p0*sin(p1*x)/x", f, -10, 10, 5);
    func->SetParameters(p0, p1);

    TCanvas *c1 = new TCanvas("c1", "Function Plot", 1920, 1080);
    func->SetLineColor(kBlue);
    func->Draw();
    // valor para x=1
    double x0 = 1.0;
    double funcValue = func->Eval(x0);
    std::cout << "f(x) = 1: " << funcValue << std::endl;
    // derivada em x = 1
    double funcDerivative = func->Derivative(x0);
    std::cout << "f'(1) = : " << funcDerivative << std::endl;
    // integral de 0 a 3
    double integralValue = func->Integral(0, 3);
    std::cout << "Integral de 0 a 3: " << integralValue << std::endl;
    // Legenda
    TPaveText *pt = new TPaveText(0.65, 0.65, 0.85, 0.85, "NDC");
    pt->SetBorderSize(0);
    pt->SetFillColor(0);
    pt->AddText(Form("f(1) = %.2f", funcValue));
    pt->AddText(Form("f' (1) = %.2f", funcDerivative));
    pt->AddText(Form("Integral = %.2f", integralValue));
    pt->AddText("p0 = 1, p1=2");
    pt->Draw();
    c1->SaveAs("plots/function_plot.png");
}

int main() {
    double p0 = 1.0;
    double p1 = 2.0;
    plotFunction(p0, p1);
}
```



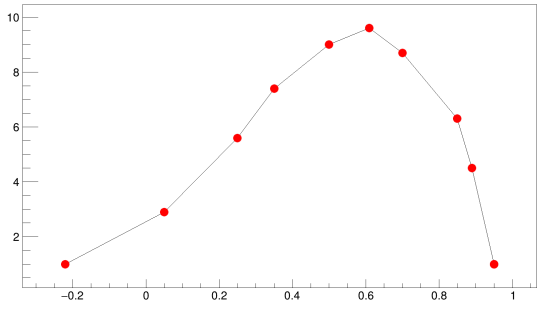
EXERCICIO 2

(a)

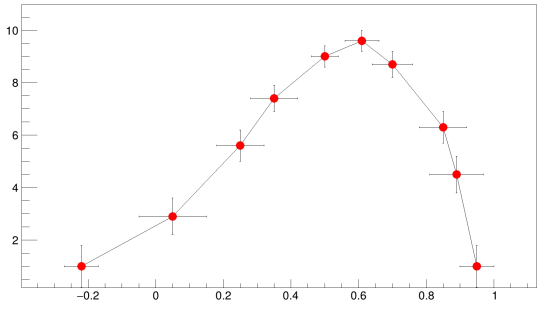
```
#include "TGraph.h"
#include "TGraphErrors.h"
#include "TCanvas.h"
#include "iostream"

void plot_graph() {
    //
   std::ifstream file("graphdata.txt");
    TGraph *plot = new TGraph();
    TCanvas *c1 = new TCanvas("c1","data",1920,1080);
    //
    double x,y;
    while (1) {
        file >> x >> y;
        plot->SetPoint(plot->GetN(),x,y);
        if (file.eof()){break;}
}
    //
    plot->SetMarkerStyle(20);
    plot->SetMarkerSize(3);
    plot->SetMarkerColor(kRed);
    //
    plot->Draw("APL");
    c1->SaveAs("ex_2.png");
}
```

(b)

```
    void plot_err() {
    TCanvas *c1 = new TCanvas ("c1","Plot com erros", 1920,1080);
    TGraphErrors *plot_errors = new TGraphErrors();
    //
    std::ifstream file("graphdata_error.txt");
    if (!file.is_open()) {
        std::cerr << "Error: Could not open the file!" << std::endl;
        return;
    }
    //
    double x,y, e_x, e_y;
    while (1) {
        file >> x >> y >> e_x >> e_y;
        plot_errors->SetPoint(plot_errors->GetN(),x,y);
        plot_errors->SetPointError(plot_errors->GetN()-1,e_x,e_y);
        if (file.eof()){break;}
    }
    //
    plot_errors->SetMarkerStyle(20);
    plot_errors->SetMarkerSize(3);
    plot_errors->SetMarkerColor(kRed);
    //
    plot_errors->Draw("APL");
    c1->SaveAs("ex_2b.png");
}
```



(a)

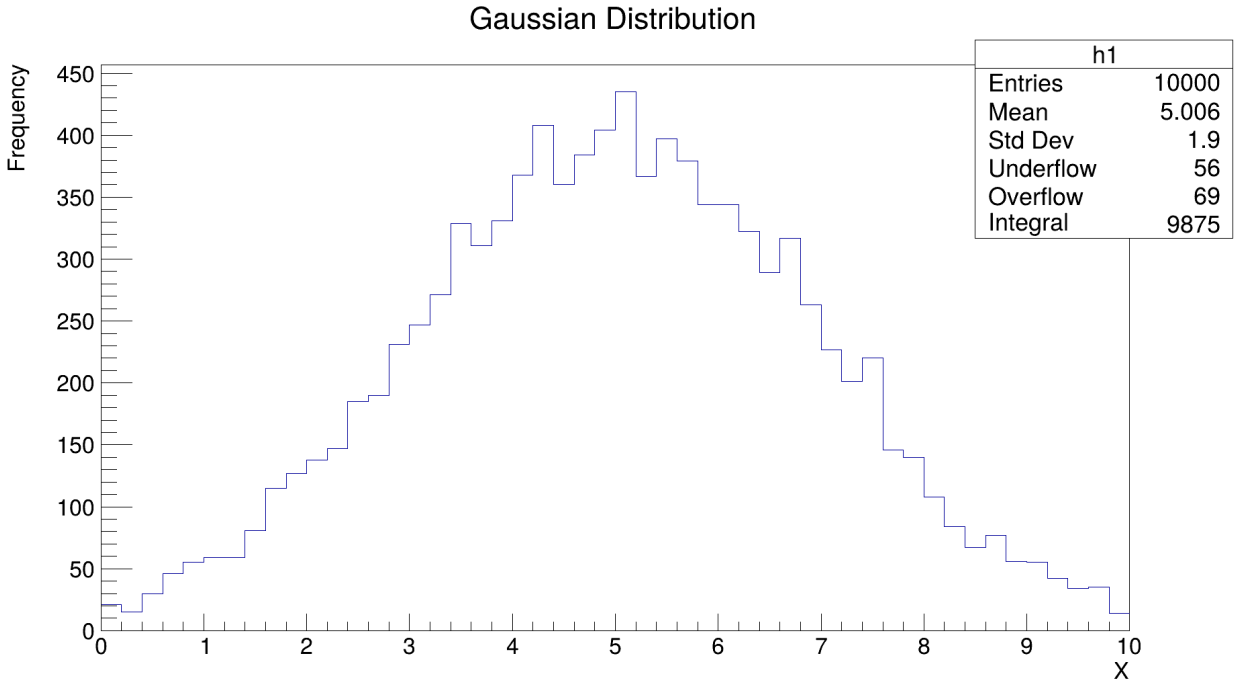


(b)

EXERCICIO 3

```
    #include "TH1F.h"
#include "TCanvas.h"
#include "TRandom.h"
#include "TStyle.h"
#include "TPaveStats.h"

void plot_gaussian_histogram() {
    //
    TCanvas *c1 = new TCanvas("c1", "Gaussian Histogram", 1920, 1080);
    //
    TH1F *h1 = new TH1F("h1", "Gaussian Distribution;X;Frequency", 50, 0, 10);
    for (int i = 0; i < 10000; ++i) {
        double random_value = gRandom->Gaus(5, 2);
        h1->Fill(random_value);
    }
    //
```

```
    gStyle->SetOptStat(1111111);  // Default stats box settings
    h1->Draw();
    //
    double skewness = h1->GetSkewness();
    double kurtosis = h1->GetKurtosis();
    TPaveStats *stats = (TPaveStats*)h1->FindObject("stats");
    if (stats) {
        stats->SetTextColor(kBlack);
    //
double skewness = h1->GetSkewness();
double kurtosis = h1->GetKurtosis();

    // R
TPaveStats *stats = (TPaveStats*)h1->FindObject("stats");
if (stats) {
    //
    stats->AddText(Form("Skewness = %.5f", skewness));
    stats->AddText(Form("Kurtosis = %.5f", kurtosis));
}
    stats->SetY1NDC(0.6);
    }
    c1->Modified();
    c1->Update();
    c1->SaveAs("gaussian_histogram.png");
}
```



EXERCICIO 4

```
#include "TFile.h"
#include "TTree.h"
#include "TCut.h"
#include "TH1F.h"
#include "TCanvas.h"
#include "TString.h"
```

```
#include "TRandom.h"
#include "TSystem.h"

void plot_momentum() {
    TCanvas *c1 = new TCanvas("c1","Momentum Histogram",1920,1080);
    TFile *file = new TFile("tree.root");
    TTree *tree = (TTree*)file->Get("tree1");
    if (!tree) {
        std::cerr << "Error: Tree not found!" << std::endl;
        file->Close();
        return;
    }
    //
    TH1F *h1 = new TH1F("histogram","Total Momentum distribution",132,0,200);

    //
    float px,py,pz,ebeam;
    tree->SetBranchAddress("px",&px);
    tree->SetBranchAddress("py",&py);
    tree->SetBranchAddress("pz",&pz);
    tree->SetBranchAddress("ebeam",&ebeam);
    //
    float sumEbeam = 0;
    Int_t nEntries = tree->GetEntries();
    for (Int_t i =0; i< nEntries; i++) {
        tree->GetEntry(i);
        sumEbeam += ebeam;
    }
    float meanEbeam = sumEbeam/nEntries;
    TCut *cutEbeam = new TCut(Form("ebeam < %f || e beam > %f", meanEbeam - 0.2,
       meanEbeam + 0.2));
    tree->Draw("sqrt(px**2 + py**2 + pz**2)",*cutEbeam);
    c1->SaveAs("momentum_hist.png");

}
```