Introdução à Análise de dados em FAE **(04/05)**

Exercício 6

Professores: Mauricio, Eliza e Sandro *Name:* Dalmo da Silva Dalto

EXERCICIO 1 e EXERCICIO 2

```
{

        int d=2;
        int l=1;
        TCanvas *c1= new TCanvas("c1","c1",500,500);
        c1->Divide(4,3);

        TNtuple *t1= new TNtuple("t1","t1","x:theta");
        TF1 *sineFunc = new TF1("sineFunc", Form("(%.2i*0.5)*TMath::Sin(x)", l), 0,
            M_PI);

        TRandom rand;

        // Gera  o de um n mero aleat rio entre 0 e 1
        double random_number;
        double random_angle;
        //M_PI;
        float x;
        float x_aim;
        int N=1000;
    std::vector<float> valores = {10, 50, 100, 1000,10000,2e6};
        int k=1;

    for (int i = 0; i < valores.size(); i++) {
                N=valores[i];
                //cout<<N << " " <<i<<endl;
                for( int i=0; i<N; i++){

                random_number = rand.Uniform();
                x=l*random_number;
                random_number = rand.Uniform();
                random_angle = 0 + (M_PI-0)*random_number;

                //x_aim=(l/2)*(sin(random_number));


                t1->Fill(x,random_angle);


                }

                c1->cd(k);
                t1->Draw("x:theta","","");
                sineFunc->Draw("same");


                //t1->Draw("theta","","");
                //sineFunc->Draw("");


```

```
50                  std::string message = Form(" N = %d", N);
51
52                  TNtuple *t2 = new TNtuple("message.c_str()","t2" , "x:theta");
53
54                  // Iterar sobre os pontos e adicionar aqueles que est o dentro da
                        rea  do TF1 ao novo TNtuple
55                  for (int i = 0; i < t1->GetEntries(); i++) {
56                          t1->GetEntry(i);
57                          double x = t1->GetArgs()[0]; // Obt m o valor de x do
                                TNtuple
58                          double theta = t1->GetArgs()[1]; // Obt m o valor de theta
                                do TNtuple
59
60                          if (sineFunc->Eval(theta) >= x) {
61                                  t2->Fill(x, theta); // Adiciona o ponto ao TNtuple t2
62                          }
63                  }
64
65                          // Plotar os pontos que est o dentro da  rea  do TF1
66                  k++;
67                  c1->cd(k);
68
69                  t2->Draw("x:theta", "", "");
70                  int m;
71
72                  m= t2->GetEntries();
73
74                  double pi = (2.0 * N / m) * (float(l) / d);
75
76                  float I;
77
78                  I=d*M_PI*(float(m)/N);
79
80                  //cout<<(2.0 * N / m)<< " " << float(l / d)<< " " << l<< " " << d <<
                        endl;
81                  cout<< Form("O valor de pi para N= %.2i: ",N) << pi<< endl;
82                  cout<< "Area efetiva  I:"<< I<< endl;
83                  k++;
84          }
85  }
```

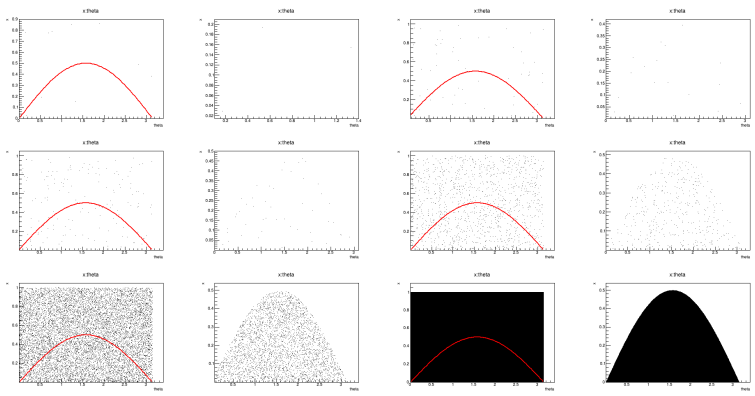

Figura 1: Exercício 1

```
1  O valor de pi para N= 10: 3.33333
2  Area efetiva  I:1.88496
3  O valor de pi para N= 50: 2.77778
4  Area efetiva  I:2.26195
5  O valor de pi para N= 100: 1.96078
6  Area efetiva  I:3.20442
```

```
O valor de pi para N= 1000: 2.57069
Area efetiva  I:2.44416
O valor de pi para N= 10000: 2.78784
Area efetiva  I:2.25378
O valor de pi para N= 2000000: 3.12259
Area efetiva  I:2.01217
```

EXERCICIO 3

```
{


double differentialCrossSection(double E, double theta, double m, double alpha) {
    double r0 = (alpha*alpha) / (2*m*m);
        double E_prime= (E)/(1+ (E/m)*(1-std::cos(theta)));
    double dOmega = std::sin(theta);
        //cout<<E_prime<<endl;
    return (r0 * r0) * std::pow((E_prime / E), 2) * ((E_prime / E) + (E / E_prime) -
       dOmega * dOmega);
}

int N=100000;
double m=0.5;
double alpha= 0.0072992701;

TCanvas *c1= new TCanvas("c1","c1",500,800);
c1->Divide(3,1);
double E = 1.0; // Energia do f ton incidente (1 MeV)
//double E_prime = 9.0e5; // Energia do f ton espalhado (900 keV)
TNtuple *t1= new TNtuple("t1","t1","x");
TNtuple *t2= new TNtuple("t2","t2","x_aim:theta");

double random_number;
double x,x_aim,theta;
TRandom rand;
double max_value=0;
float aux=0;
for(float i=0; i<M_PI;i=i+0.01){
        if(i==0){
                max_value=differentialCrossSection(E,i,m,alpha);
        }
        else if(max_value<differentialCrossSection(E,i,m,alpha)){
                max_value=differentialCrossSection(E,i,m,alpha);
                aux=i;
                //cout<<aux<<endl;
        }
        //cout<< max_value<< " " << differentialCrossSection(E,i,m,alpha) <<endl;
}



//max_value= differentialCrossSection(E,M_PI/2,m,alpha);
        for(int j=0; j<N;j++){
                random_number = rand.Uniform();
                x=max_value*random_number;

                random_number = rand.Uniform();
                theta=M_PI*random_number;
                if(differentialCrossSection(E,theta,m,alpha)>x){
                        x_aim=x;
                        t2->Fill(x_aim,theta);
```

```
52                          //cout<<"aqui"<<endl;
53                          //j=1200;
54                  }
55                  t1->Fill(x);
56                  //cout<< x << "   " << differentialCrossSection(E,theta,m,alpha)<<
                        endl;
57          }
58
59  c1->cd(1);
60  t1->Draw("x>>h2","","");
61  h2->GetXaxis()->SetTitle("d#sigma/d#Omega");
62  h2->SetTitle("Area Total");
63
64
65  c1->cd(2);
66  t2->Draw("theta>>h1","","");
67  h1->SetTitle("Area efetiva");
68  h1->GetXaxis()->SetTitle("#theta");
69
70
71  c1->cd(3);
72
73  t2->Draw("x_aim>>h3","","");
74  h3->SetTitle("Area efetiva");
75  h3->GetXaxis()->SetTitle("d#sigma/d#Omega");
76  }
```

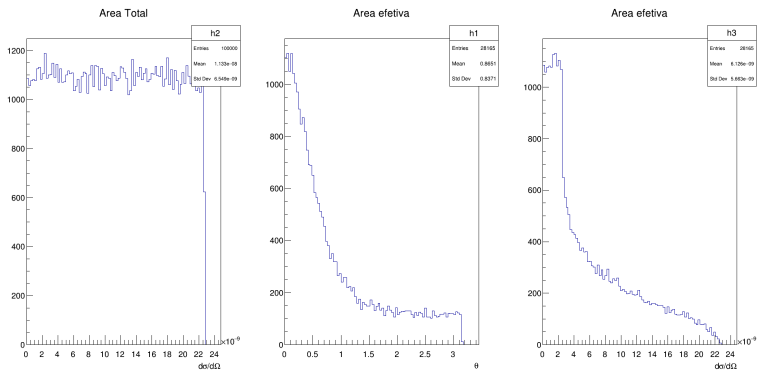

Figura 2: Exercício 3: E= 1MeV

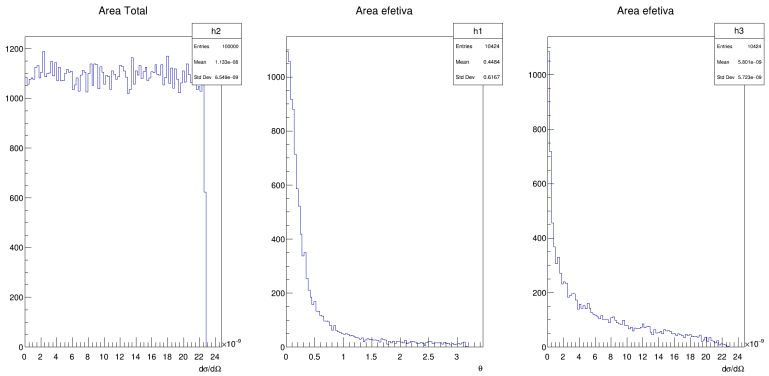

Figura 3: Exercício 3: E= 10MeV

EXERCICIO 4

```
import pythia8
from ROOT import TFile, TTree, vector

# Configura  o do Pythia
pythia = pythia8.Pythia()
pythia.readString("Beams:eCM = 13000")
pythia.readString("SoftQCD:all = on")  # Configura  o do processo f sico

# Abre o arquivo ROOT
f = TFile("output.root", "RECREATE")

# Cria um TTree para armazenar os eventos
tree = TTree("tree_pythia", "Pythia Events")

# Define as vari veis para armazenar as informa  es dos eventos
event_id = vector("int")()
particle_id = vector("int")()
particle_px = vector("float")()
particle_py = vector("float")()
particle_pz = vector("float")()

# Cria os branches do TTree para armazenar as informa  es
tree.Branch("event_id", event_id)
tree.Branch("particle_id", particle_id)
tree.Branch("particle_px", particle_px)
tree.Branch("particle_py", particle_py)
tree.Branch("particle_pz", particle_pz)

# Inicializa o Pythia e gera eventos
pythia.init()
for iEvent in range(1000):
    if not pythia.next():
        continue

    # Preenche as informa  es do evento
    event_id.push_back(iEvent)
    for particle in pythia.event:
        particle_id.push_back(particle.id())
        particle_px.push_back(particle.px())
        particle_py.push_back(particle.py())
        particle_pz.push_back(particle.pz())

    # Preenche o TTree com as informa  es do evento atual
    tree.Fill()

    # Limpa as vari veis do evento atual para o pr ximo evento
    event_id.clear()
    particle_id.clear()
    particle_px.clear()
    particle_py.clear()
    particle_pz.clear()

# Finaliza o Pythia ap s gerar os eventos
pythia.stat()

# Salva o TTree no arquivo ROOT
f.Write()
f.Close()
```

```
62 //Agora irei a Tree em C++ e fitar um plot de convolu  o de 3 gaussianas em Pz
63
64
65 using namespace RooFit;
66
67 TFile file("output.root");
68 TTree* tree = dynamic_cast<TTree*>(file.Get("tree_pythia"));
69 tree->Draw("particle_pz","particle_pz>-10 && particle_pz<10","");
70
71 std::vector<float>* pz_vec = nullptr;
72 std::vector<float>* py_vec = nullptr;
73 std::vector<float>* px_vec = nullptr;
74
75 tree->SetBranchAddress("particle_pz", &pz_vec);
76 tree->SetBranchAddress("particle_py", &py_vec);
77 tree->SetBranchAddress("particle_px", &px_vec);
78
79 if (!pz_vec || !py_vec || !px_vec) {
80         std::cerr << "Erro: N o foi poss vel associar os branches do TTree." << std
              ::endl;
81         return;
82 }
83
84 // Crie vetores para armazenar os valores das vari veis
85 std::vector<float> pz_values;
86 std::vector<float> py_values;
87 std::vector<float> px_values;
88
89 // Preencha os vetores com os valores do TTree
90 for (Long64_t i = 0; i < tree->GetEntries(); ++i) {
91         tree->GetEntry(i);
92
93         for (size_t j = 0; j < pz_vec->size(); ++j) {
94                 pz_values.push_back(pz_vec->at(j));
95                 py_values.push_back(py_vec->at(j));
96                 px_values.push_back(px_vec->at(j));
97         }
98 }
99
100 // Crie RooRealVar para cada vari vel
101 RooRealVar pz_var("pz_var", "pz", -8000., 8000.);
102 RooRealVar py_var("py_var", "py", -10000., 10000.);
103 RooRealVar px_var("px_var", "px", -10000., 10000.);
104
105 // Crie RooDataSet
106 RooDataSet rooData("data", "data", RooArgSet(pz_var, py_var, px_var));
107 for (size_t i = 0; i < pz_values.size(); ++i) {
108         pz_var.setVal(pz_values[i]);
109         py_var.setVal(py_values[i]);
110         px_var.setVal(px_values[i]);
111         rooData.add(RooArgSet(pz_var, py_var, px_var));
112 }
113
114
115
116
117 // Plote o RooDataSet
118 RooPlot* frame3 = pz_var.frame(Title("Pz"));
119 rooData.plotOn(frame3);
120 frame3->Draw();
121
122 RooRealVar mean1("mean1", "mean of first Gaussian", -6500, -6700, -6400);
123 RooRealVar sigma1("sigma1", "width of first Gaussian", 30, 20, 50);
```

```
124 RooRealVar mean2("mean2", "mean of second Gaussian", 0, -1000, 1000);
125 RooRealVar sigma2("sigma2", "width of second Gaussian", 100, 50, 200);
126 RooRealVar mean3("mean3", "mean of third Gaussian", 6500, 6500, 6700);
127 RooRealVar sigma3("sigma3", "width of third Gaussian", 30, 20, 50);
128
129 // Crie as tr s Gaussianas
130 RooGaussian gauss1("gauss1", "first Gaussian PDF", pz_var, mean1, sigma1);
131 RooGaussian gauss2("gauss2", "second Gaussian PDF", pz_var, mean2, sigma2);
132 RooGaussian gauss3("gauss3", "third Gaussian PDF", pz_var, mean3, sigma3);
133
134 // Crie RooRealVar para os coeficientes de cada Gaussiana
135 RooRealVar coef1("coef1", "coefficient of first Gaussian", 0.5, 0., 3);
136 RooRealVar coef2("coef2", "coefficient of second Gaussian", 80, 0., 100);
137 RooRealVar coef3("coef3", "coefficient of third Gaussian", 0.5, 0., 3);
138
139 // Crie a PDF de soma das tr s Gaussianas
140 RooAddPdf sum_pdf("sum_pdf", "Sum of three Gaussians", RooArgList(gauss1, gauss2,
        gauss3), RooArgList(coef1, coef2, coef3));
141
142 // Fa a o ajuste
143 RooFitResult* result = sum_pdf.fitTo(rooData, RooFit::Save());
144
145
146
147 // Criar a canvas para o plot
148 TCanvas *c3 = new TCanvas("exemplo03","exemplo03",800,400);
149 TPaveText *statistics = new TPaveText(0.6, 0.6, 0.9, 0.9, "NDC");
150 statistics->SetFillColor(0);
151 statistics->AddText(Form("Chi2/NdF = %.2f", frame3->chiSquare()));
152 //statistics->AddText(Form("Linear Parameters:"));
153 //statistics->AddText(Form("   Slope = %.3f", slope.getVal()));
154 //statistics->AddText(Form("   Intercept = %.3f", intercept.getVal()));
155 statistics->AddText(Form("Gaussian Parameters:"));
156 statistics->AddText(Form("   Mean_1 = %.3f", mean1.getVal()));
157 statistics->AddText(Form("   Sigma = %.3f", sigma1.getVal()));
158 statistics->AddText(Form("   Mean_2 = %.3f", mean2.getVal()));
159 statistics->AddText(Form("   Sigma_2 = %.3f", sigma2.getVal()));
160 statistics->AddText(Form("   Mean_3 = %.3f", mean3.getVal()));
161 statistics->AddText(Form("   Sigma_3 = %.3f", sigma3.getVal()));
162 statistics->AddText(Form("   cof_1 = %.3f", coef1.getVal()));
163 statistics->AddText(Form("   cof_2 = %.3f", coef2.getVal()));
164 statistics->AddText(Form("   cof_3 = %.3f", coef3.getVal()));
165 //statistics->AddText(Form("   amplitude = %.3f", sinal.getVal()));
166 statistics->Draw();
167 sum_pdf.plotOn(frame3, RooFit::LineColor(kRed));
168 frame3->Draw();
169 sum_pdf.plotOn(frame3, RooFit::LineColor(kRed));
170 c3->Draw();
171 statistics->Draw();
```

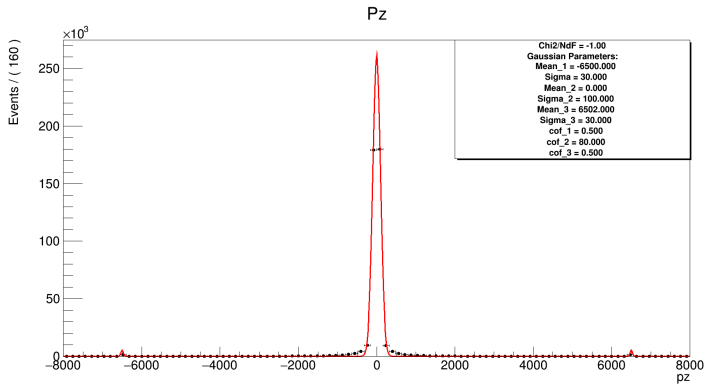

Figura 4: Exercício 4: Pz