Introdução à Análise de dados em FAE e Tecnologias Associadas **(DATA)**

Exercicio dos slides Roofit parte 02

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

EXERCICIO 1

Codigo para rodar o ajusta da Crystal Ball

```
include "TTree.h"
#include "TH1.h"
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooArgSet.h"
#include "RooPlot.h"
#include "RooFit.h"
#include "TCanvas.h"
#include "RooCBShape.h"
#include "RooExponential.h"
#include "RooAddPdf.h"
#include "RooCategory.h"
#include "RooConstVar.h"
#include <iostream>
#include "TLegend.h"
#include "TPad.h"

using namespace RooFit;

void crystal_ball_fit2()
{

    RooRealVar mass("mass", "mass", -10, 10);
    RooRealVar mean("mean", "mean", 2, -10, 10);
    RooRealVar sigma("sigma", "sigma", 1.2, 0.5, 3);
    RooRealVar alpha("alpha", "alpha", 2.0, 0.5, 3.0);
    RooRealVar n("n", "n", 5.0, 0.1, 10.0);


    RooRealVar tau("tau", "tau", -0.5, -2.0, 0.0);
    RooExponential expDecay("expDecay", "Exponential Decay", mass, tau);


    RooCBShape cbShape("cbShape", "Crystall Ball pdf", mass, mean, sigma, alpha, n);


    RooRealVar cbYield("cbYield", "Crystal Ball yield", 800, 0, 10000);
    RooRealVar backgroundYield("backgroundYield", "background yield", 200, 0, 10000);


    RooAddPdf model("model", "Crystall Ball with Background", RooArgList(cbShape,
        expDecay), RooArgList(cbYield, backgroundYield));


    RooWorkspace w("w");
    w.import(model);

    RooDataSet* dado = model.generate(RooArgSet(mass), 1000);

    RooFitResult* fitResult = model.fitTo(*dado, Save());
```

```
50      RooPlot* frame = mass.frame();
51      frame->SetTitle("Ajuste da Crystal Ball");
52
53
54      dado->plotOn(frame);
55      model.plotOn(frame);
56      model.plotOn(frame, Components(cbShape), LineStyle(kDashed), LineColor(kRed));
57      model.plotOn(frame, Components(expDecay), LineStyle(kDotted), LineColor(kBlue));
58
59      double chi2 = frame->chiSquare(fitResult->floatParsFinal().getSize());
60
61
62      TCanvas* c = new TCanvas("c", "c", 800, 800);
63      TPad* pad1 = new TPad("pad1", "Graph", 0, 0.3, 1, 1.0);
64      pad1->SetBottomMargin(0);
65      pad1->Draw();
66      pad1->cd();
67      frame->Draw();
68
69
70      c->cd();
71      TPad* pad2 = new TPad("pad2", "Legend", 0, 0.0, 1, 0.3);
72      pad2->SetTopMargin(0);
73      pad2->SetBottomMargin(0.2);
74      pad2->Draw();
75      pad2->cd();
76
77
78      TLegend* legend = new TLegend(0.1, 0.1, 0.9, 0.9);
79      legend->SetTextSize(0.08);
80      legend->SetBorderSize(0);
81      legend->SetFillStyle(0);
82      legend->AddEntry((TObject*)0, Form("Mean = %.3f    %.3f", mean.getVal(), mean.
            getError()), "");
83      legend->AddEntry((TObject*)0, Form("Sigma = %.3f    %.3f", sigma.getVal(), sigma.
            getError()), "");
84      legend->AddEntry((TObject*)0, Form("Alpha = %.3f    %.3f", alpha.getVal(), alpha.
            getError()), "");
85      legend->AddEntry((TObject*)0, Form("N = %.3f    %.3f", n.getVal(), n.getError()),
             "");
86      legend->AddEntry((TObject*)0, Form("Tau = %.3f    %.3f", tau.getVal(), tau.
            getError()), "");
87      legend->AddEntry((TObject*)0, Form("#chi^{2}/ndf = %.2f", chi2), "");
88      legend->Draw();
89
90
91      c->SaveAs("bDecayCrystallBall3.png");
92      w.writeToFile("wspaceCrystallBall3.root");
93  }
```

Logo após o código rodar teremos o seguinte ajuste:

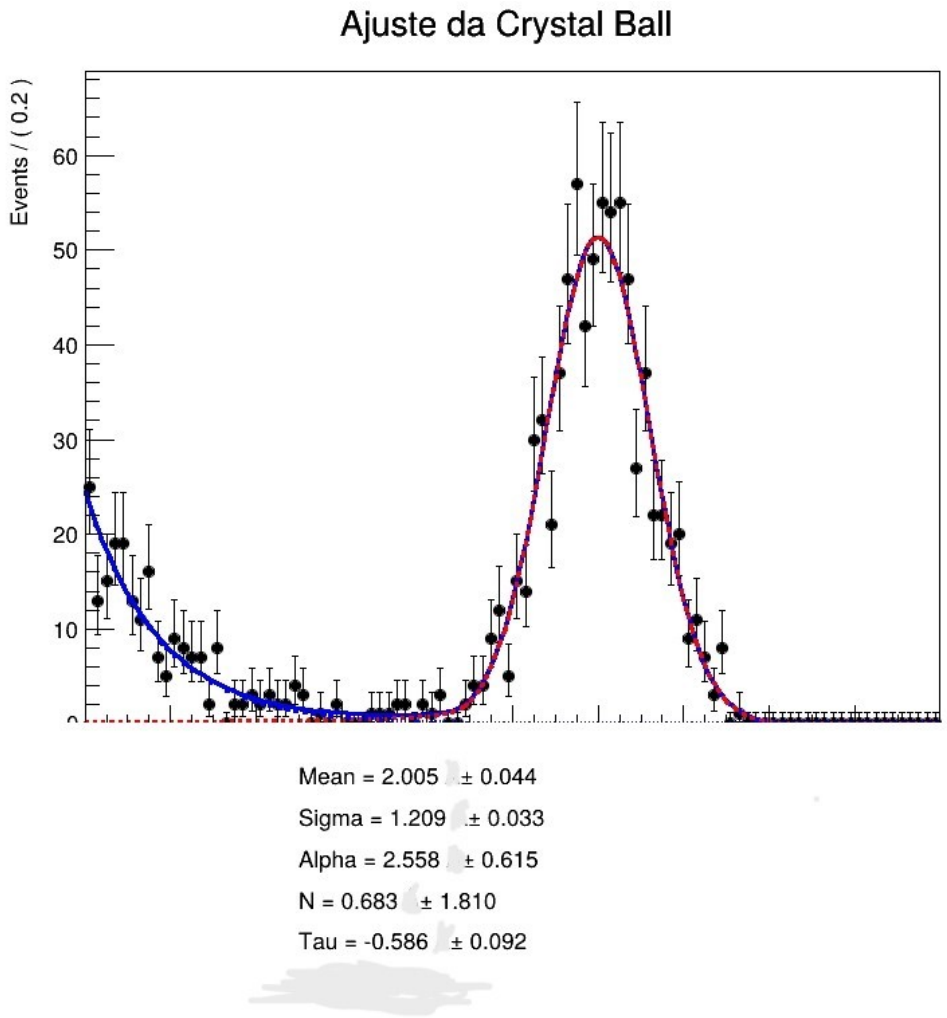

Figura 1: Ajuste da função Crystal Ball com background. A curva vermelha representa a função Crystal Ball, enquanto a curva azul representa o decaimento exponencial do background.

No entanto, como o código ainda precisa ser executado para gerar os resultados específicos, aqui está um exemplo de como esses valores serão exibidos após o ajuste:

- **Mean**: Valor central ajustado da distribuição Crystal Ball.
- **Sigma**: Largura do pico ajustado da distribuição.
- **Alpha**: Parâmetro que controla a assimetria (cauda) da distribuição.
- **N**: Controla o decaimento da cauda da distribuição Crystal Ball.
- **Tau**: Constante de decaimento exponencial que controla o fundo.

EXERCICIO 2

TEXTO

```
#include "RooRealVar.h"
#include "RooExponential.h"
#include "RooDataSet.h"
#include "RooPlot.h"
#include "TCanvas.h"
```

```
6
7  void ajuste_exponencialext() {
8      RooRealVar x("x", "x", 0, 10);
9      RooRealVar lambda("lambda", "decay rate of exponential", 1, 0.1, 2);
10     RooExponential expo("expo", "Exponential PDF", x, lambda);
11
12
13     RooRealVar nexp("nexp", "number of expected events", 1500, 0, 3000);
14
15
16     RooAddPdf model("model", "Extended Exponential Model", RooArgList(expo),
           RooArgList(nexp));
17
18
19     RooDataSet* data = model.generate(x, 1500);
20
21     RooFitResult* fit_result = model.fitTo(*data, RooFit::Save(), RooFit::Extended())
           ;
22     fit_result->Print("v");
23
24     double lambda_val = lambda.getVal();
25     double lambda_err = lambda.getError();
26     double nexp_val = nexp.getVal();
27     double nexp_err = nexp.getError();
28
29     std::cout << "Valor ajustado de lambda: " << lambda_val << "    " << lambda_err
           << std::endl;
30     std::cout << "N mero total de eventos ajustados: " << nexp_val << "    " <<
           nexp_err << std::endl;
31
32
33     RooPlot* frame = x.frame();
34     frame->SetTitle("Ajuste da funcao exponencial estendida");
35     data->plotOn(frame);
36     model.plotOn(frame);
37
38     TCanvas* c = new TCanvas("c", "Ajuste da fun ao exponencial estendida", 900,
           700);
39     frame->Draw();
40     model.paramOn(frame, RooFit::Layout(0.6, 0.9, 0.9));
41     frame->Draw();
42     c->Draw();
43     c->SaveAs("AjusteExponencialext.png");
44 }
```

Logo o ajuste, após a rodagem do codigo:

Valor ajustado para o parâmetro λ:

O valor ajustado de λ é obtido a partir do ajuste aos dados simulados. Ele será exibido no terminal após a execução do código e depende do comportamento estatístico dos eventos simulados.

Número total de eventos ajustados:

O número total de eventos ajustados (n_{exp}) também será impresso no terminal, juntamente com a incerteza associada.

Comparação dos valores ajustados com os valores gerados:

O valor inicial de λ era 1, e o número de eventos gerados foi 1500. Após o ajuste, os valores estimados para λ e o número total de eventos podem variar um pouco devido a flutuações estatísticas inerentes ao processo de simulação.

Se o valor ajustado de λ estiver próximo de 1 e o número de eventos ajustado estiver próximo de 1500, então os resultados são considerados dentro das expectativas. Pequenas variações são esperadas devido à natureza dos ajustes de verossimilhança.

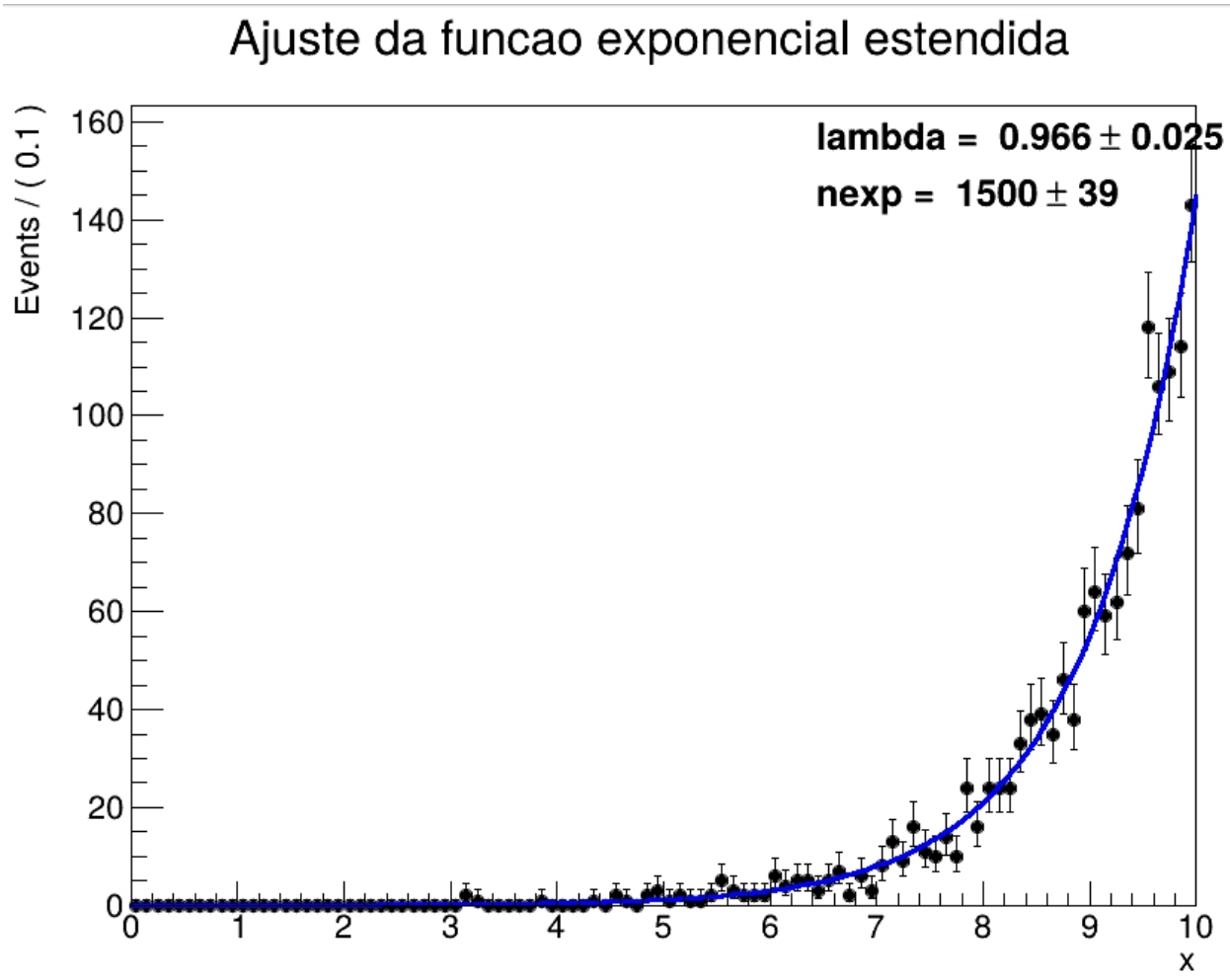

Figura 2: Ajuste da função exponencial estendida

EXERCICIO 3

Com o codigo abaixo para modelo do J/psi:

```
#include <iostream>
#include "TFile.h"
#include "TH1.h"
#include "TCanvas.h"
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooAddPdf.h"
#include "RooCBShape.h"
#include "RooPolynomial.h"
#include "RooPlot.h"
#include "RooFitResult.h"
#include "RooFit.h"
#include "TLegend.h"

void ajustejpsi() {

    TFile *file = TFile::Open("/home/jmarcos/lastexroo/DataSet_lowstat.root");
    if (!file || file->IsZombie()) {
        std::cerr << "Erro: N o foi poss vel abrir o arquivo." << std::endl;
        return;
    }

    RooDataSet *data = (RooDataSet*)file->Get("data");
    RooRealVar mass("mass", "Massa [GeV/c^{2}]", 2.9, 3.3);

    // PDF do sinal no pico do J/       com uma fun   o Crystal Ball
    RooRealVar mean("mean", "M dia", 3.096916, 3.08, 3.12);
    RooRealVar sigma("sigma", "Largura", 0.02, 0.01, 0.05);
    RooRealVar alpha("alpha", "Alpha", 1.5, 0.5, 3.0);
    RooRealVar n("n", "n", 5.0, 0.5, 10.0);
    RooCBShape signal("signal", "Fun   o Crystal Ball (Sinal)", mass, mean, sigma,
        alpha, n);


    RooRealVar a1("a1", "a1", -0.1, -1.0, 1.0);
    RooPolynomial background("background", "Fun   o Polinomial (Background)", mass,
        RooArgList(a1));


    RooRealVar frac("frac", "Fra   o do sinal", 0.5, 0.0, 1.0);
    RooAddPdf model("model", "Modelo Sinal + Fundo", RooArgList(signal, background),
        RooArgList(frac));


    RooFitResult *fitResult = model.fitTo(*data, RooFit::Save());


    RooPlot *frame = mass.frame();
    data->plotOn(frame);
    model.plotOn(frame);
    model.plotOn(frame, RooFit::Components("background"), RooFit::LineStyle(kDashed),
         RooFit::LineColor(kRed));
    model.plotOn(frame, RooFit::Components("signal"), RooFit::LineStyle(kSolid),
        RooFit::LineColor(kBlue));


    double chi2 = frame->chiSquare();
    int ndf = data->numEntries() - fitResult->floatParsFinal().getSize();  //
        Calculando ndf
```

```
54
55      std::cout << "Chi^2 / ndf = " << chi2 << " / " << ndf << std::endl;
56      std::cout << "M dia (mean) = " << mean.getVal() << " +/- " << mean.getError() <<
            std::endl;
57      std::cout << "Sigma = " << sigma.getVal() << " +/- " << sigma.getError() << std::
            endl;
58      std::cout << "Alpha = " << alpha.getVal() << " +/- " << alpha.getError() << std::
            endl;
59      std::cout << "n = " << n.getVal() << " +/- " << n.getError() << std::endl;
60      std::cout << "Fra  o do sinal = " << frac.getVal() << " +/- " << frac.getError
            () << std::endl;
61
62      TCanvas *c = new TCanvas("c", "Ajuste da Massa do J/psi", 1000, 600);
63      c->Divide(2, 1);
64
65      c->cd(1);
66      frame->SetTitle("Ajuste do modelo para o J/psi ");
67      frame->Draw();
68
69      c->cd(2);
70      TLegend *leg = new TLegend(0.1, 0.1, 0.9, 0.9);  // Legenda ocupando a maior
            parte do espa o
71      leg->AddEntry(frame->findObject("data"), "Dados", "PL");
72      leg->AddEntry(frame->findObject("model"), "Modelo Ajustado", "l");
73      leg->AddEntry((TObject*)0, Form("M dia (Pico) = %.3f +/- %.3f GeV/c ", mean.
            getVal(), mean.getError()), "");
74      leg->AddEntry((TObject*)0, Form("Sigma = %.3f +/- %.3f GeV/c ", sigma.getVal(),
            sigma.getError()), "");
75      leg->AddEntry((TObject*)0, Form("Alpha = %.3f +/- %.3f", alpha.getVal(), alpha.
            getError()), "");
76      leg->AddEntry((TObject*)0, Form("n = %.3f +/- %.3f", n.getVal(), n.getError()), "
            ");
77      leg->AddEntry((TObject*)0, Form("Fra  o do sinal = %.3f +/- %.3f", frac.getVal
            (), frac.getError()), "");
78      leg->AddEntry((TObject*)0, Form("Chi^2/ndf = %.3f", chi2 / ndf), "");
79      leg->AddEntry((TObject*)0, "Fundo (Polinomial)", "l");
80      leg->AddEntry((TObject*)0, "Sinal (Crystal Ball)", "l");
81
82      leg->Draw();
83      c->SaveAs("ajuste_jpsi_sinal_background.png");
84
85      file->Close();
86  }
```

Obtemos o seguinte resultado:

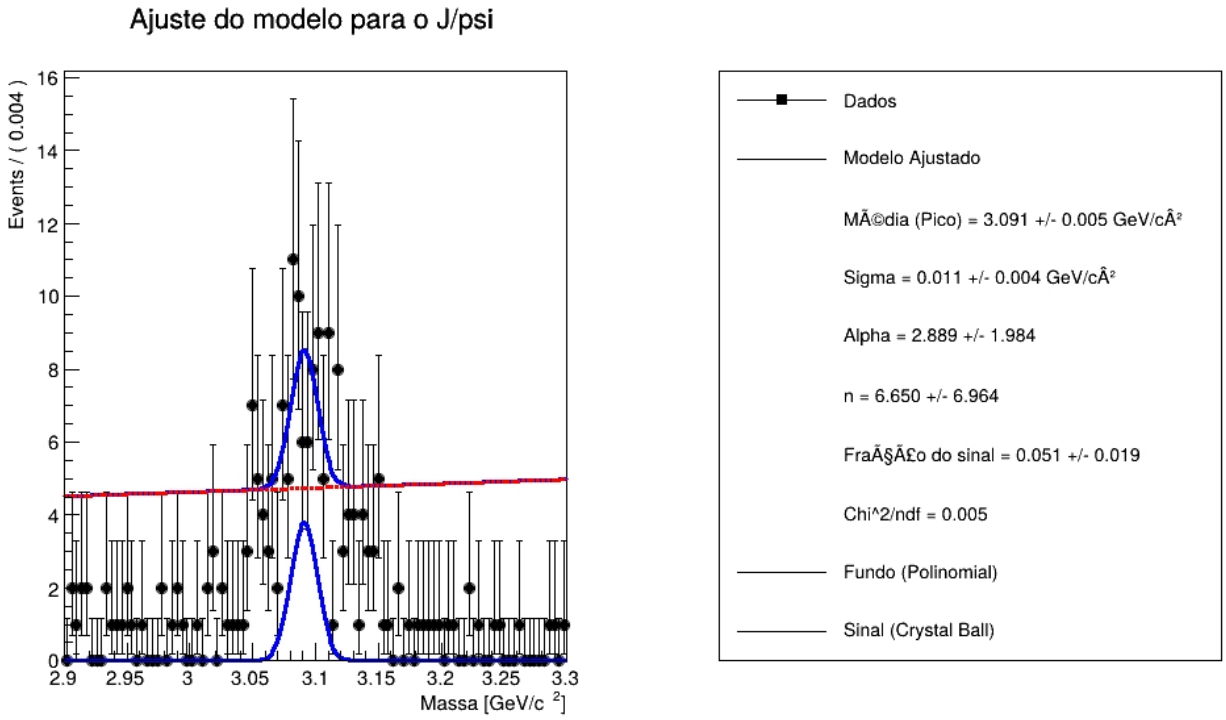

Figura 3: Ajuste do modelo da massa J/psi