Root 2008

www.if.usp.br/suaide

Alexandre Suaide

aula 2

Programa

- Aula 1
  - Introdução ao c++ e ROOT
  - Conceito de classe e objeto
  - Básico de gráficos e funções no ROOT
- Aula 2
  - Mais gráficos e funções
  - Histogramas de 1 e 2D
  - Ajustes de funções, legendas, etc.
  - Escrevendo programas simples: Monte Carlo e simulações
- Aula 3
  - Referências e ponteiros
  - Nomes e memória
  - Programação mais complexa: mais Monte Carlo
- Aula 4
  - I/O no ROOT
  - Mais programação no ROOT
  - Compilando com o ROOT

Algumas variáveis internas do ROOT

- gSystem
  - TSystem – Classe que controla interface com S.O.
    - gSystem->cd("c:\root");
    - gSystem->pwd();
    - gSystem->Exec("emacs meu_programa.C");
- gStyle
  - TStyle – Define vários padrões básicos gráficos, por exemplo, cor padrão de tela, linha, fonte padrão, etc.
- gROOT
  - TROOT – Controla os atributos globais do ROOT
- gRandom
  - TRandom – Gerador de números aleatórios
- gPad
  - Variável da classe TPad, sempre corresponde à tela gráfica ativa atualmente
    - gPad->SetLogy(); // log Y na tela ativa

Gráficos e funções

$$f(x) = \frac{A}{Bx\sqrt{C^2 + \left(Dx - \frac{1}{Bx}\right)^2}}$$

fit.dat - /\

File Edit Search
Preferences Shell
Macro Windows
Help

3500	18.6	0.8
3600	17.3	0.9
3700	19.1	0.9
3800	18.7	1.0
3900	20.1	1.0
4000	21.0	1.1
4100	25.1	1.2
4200	25.0	1.2
4300	28.9	1.4
4400	30.1	1.5
4500	37.4	1.6
4600	35.6	1.8
4700	45.6	2.1
4800	48.9	2.5
4900	59.4	2.9
5000	72.4	3.7
5100	96.0	4.8
5200	146.2	6.6
5300	177.6	8.9
5400	190.2	8.6
5500	124.2	6.2
5600	86.9	4.4
5700	72.5	3.3
5800	53.8	2.6
5900	47.5	2.2
6000	39.7	1.8
6100	32.2	1.6
6200	28.2	1.4
6300	24.9	1.2
6400	22.6	1.1

Variáveis definidas fora de funções são globais e podem ser manipuladas diretamente no prompt

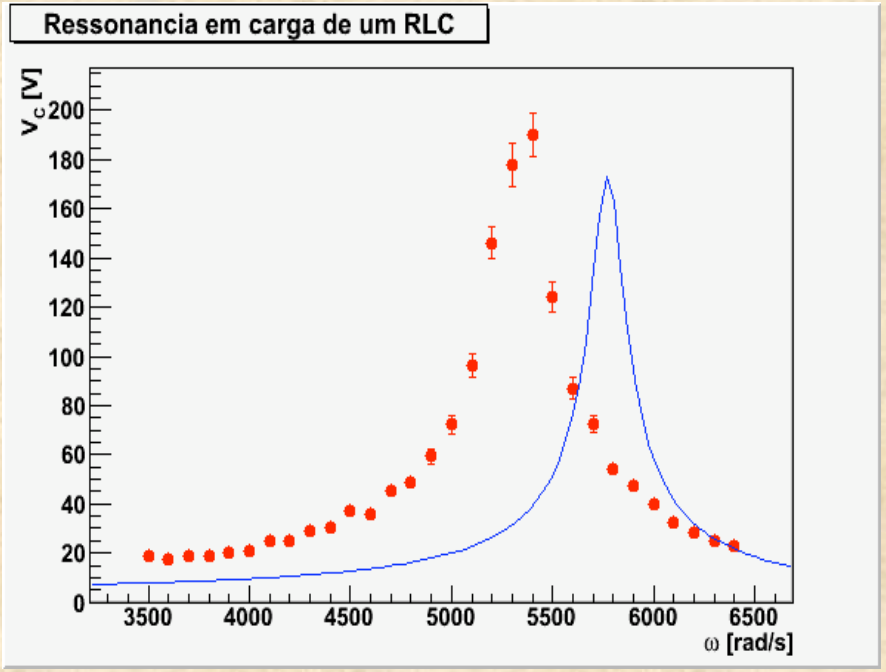
```
// fora das funcoes
TGraphErrors *g;
TF1 *f;

exemplo_fit()
{
  g = new TGraphErrors("fit.dat","%lg %lg %lg");

  g->Draw("a p");
  g->SetMarkerStyle(20);
  g->SetMarkerColor(2);
  g->SetLineColor(2);
  g->SetTitle("Ressonancia em carga de um RLC");
  g->GetXaxis()->SetTitle("#omega [rad/s]");
  g->GetYaxis()->SetTitle("V_(C) [V]");

  f = new TF1("funcao","[0]/(x*[1]*sqrt([2]^2+(x*[3]-1/(x*[1]))^2))",3000,7000);
  f->SetParameters(5,1e-6,5,30e-3);

  f->Draw("same l");
  f->SetLineWidth(1);
  f->SetLineColor(4);

}
```

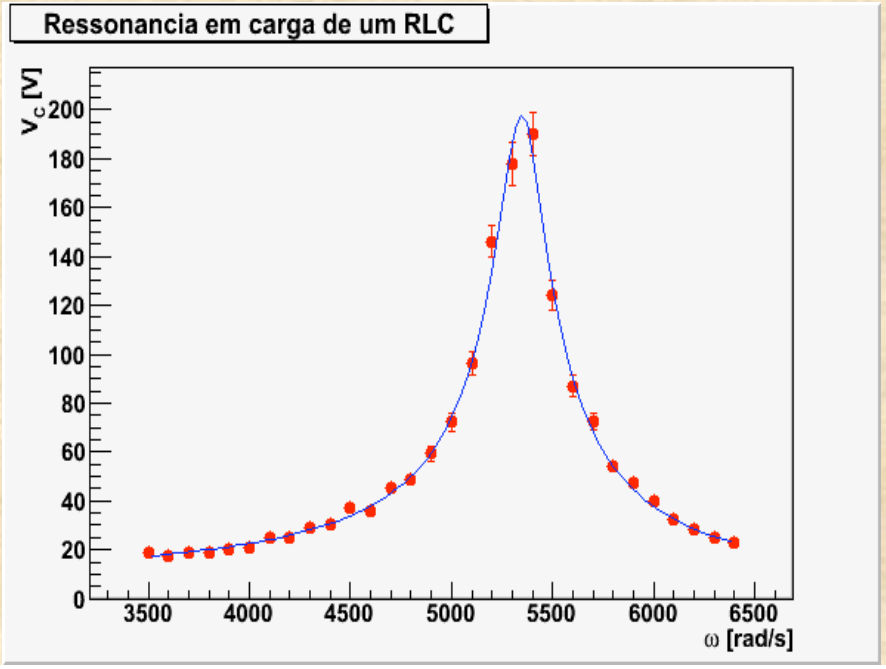
O same faz com que o gráfico seja desenhado em superposição ao anterior. O l desenha linha

Ajustando funções

No prompt do ROOT digite

```
Root [0] .x exemplo_fit.C

Root [1] g->ChiSquare(f)
(const Double_t)6.5397488e+03

Root [2] g->Fit(f)
Aparece um monte de coisa

Root [3] g->ChiSquare(f)
(const Double_t)2.967201e+01

Root [4] g->ChiSquare/f->GetNDF()
(const double)1.1412340e+00

Root [5] f->GetParameter(2)
(const Double_t)8.947880e+00

Root [6] f->GetParError(2)
(const Double_t)1.782507e+00
```

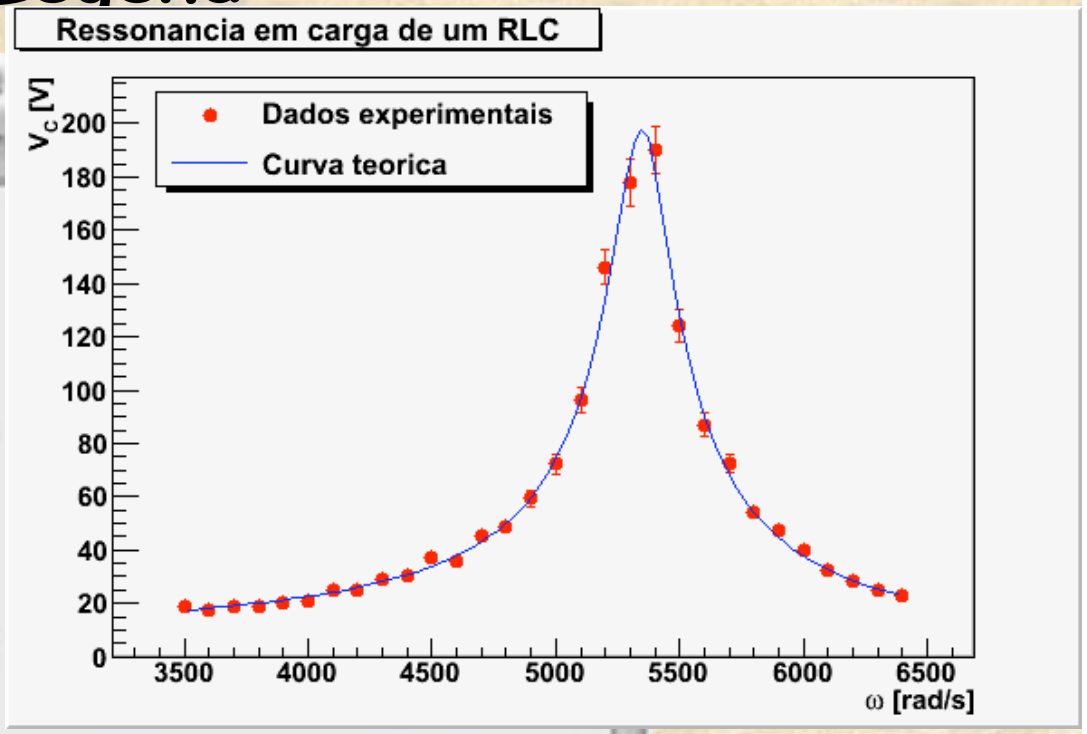




TLegend

- A classe Tlegend cria legenda de gráficos
  - … = new Tlegend(xi, y1, xf, yf);
  - Onde xi, yi, xf, yf são as coordenadas na tela.
    - Números reais (float) entre 0 e 1
  - Ex:
    - TLegend *L = new TLegend(0.25, 0.30, 0.70, 0.50);
- Adicionando ítem na legenda
  - AddEntry(*objeto, "Título", "modo");
  - Onde modo = p (ponto), l (linha), f (fill, caixa)

TLegend



exemplo_fit_2.C - /Volumes/Data/U

File Edit Search Preferences Shell Macr

```
// variaveis globais, podem ser manipuladas
// fora das funcoes ou em outras funcoes
TGraphErrors *g;
TF1 *f;

exemplo_fit_2()
{
  g = new TGraphErrors("fit.dat","%lg %lg %lg");

  g->Draw("a p");
  g->SetMarkerStyle(20);
  g->SetMarkerColor(2);
  g->SetLineColor(2);
  g->SetTitle("Ressonancia em carga de um RLC");
  g->GetXaxis()->SetTitle("#omega [rad/s]");
  g->GetYaxis()->SetTitle("V_{C} [V]");

  f = new TF1("funcao","[0]/(x*[1]*sqrt([2]^2+(x*[3]-1/(x*[1]))^2))",3000,7000);
  f->SetParameters(5.1e-6,5.30e-3);

  f->Draw("same l");
  f->SetLineWidth(1);
  f->SetLineColor(4);

  g->Fit(f);

  TLegend *l = new TLegend(0.14, 0.75, 0.54, 0.89);
  l->AddEntry(g,"Dados experimentais","p");
  l->AddEntry(f,"Curva teorica","l");
  l->Draw();

}
```

Parte nova adicionada ao programa. Faz o ajuste dos dados e desenha a legenda

Monte Carlo no ROOT

- Sorteio de números aleatórios simples
  - TRandom, TRandom1, TRandom2, TRandom3
    - A única diferença é o algorítimo de geração
  - Geradores básicos

    Exp(tau)
    - Integer(imax)
    - Gaus(mean,sigma)
    - Rndm()
    - Uniform(x1)
    - Landau(mpv,sigma)
    - Poisson(mean)
    - Binomial(ntot,prob)
- Ou usar funções (TF1,etc.) para outras distribuições de probabilidade

Gráficos e histogramas no ROOT

- **Gráficos e histogramas**
  - O ROOT possui uma quantidade enorme de classes para tratar objetos gráficos
  - Histogramas
    - TH1 – Histogramas de 1 dimensão
      - TH1I, TH1S, TH1F, TH1D, ... (estabelece a precisão do eixo)
    - TH2 – Histogramas de 2 dimensões
    - TH3 – Histogramas de 3 dimensões

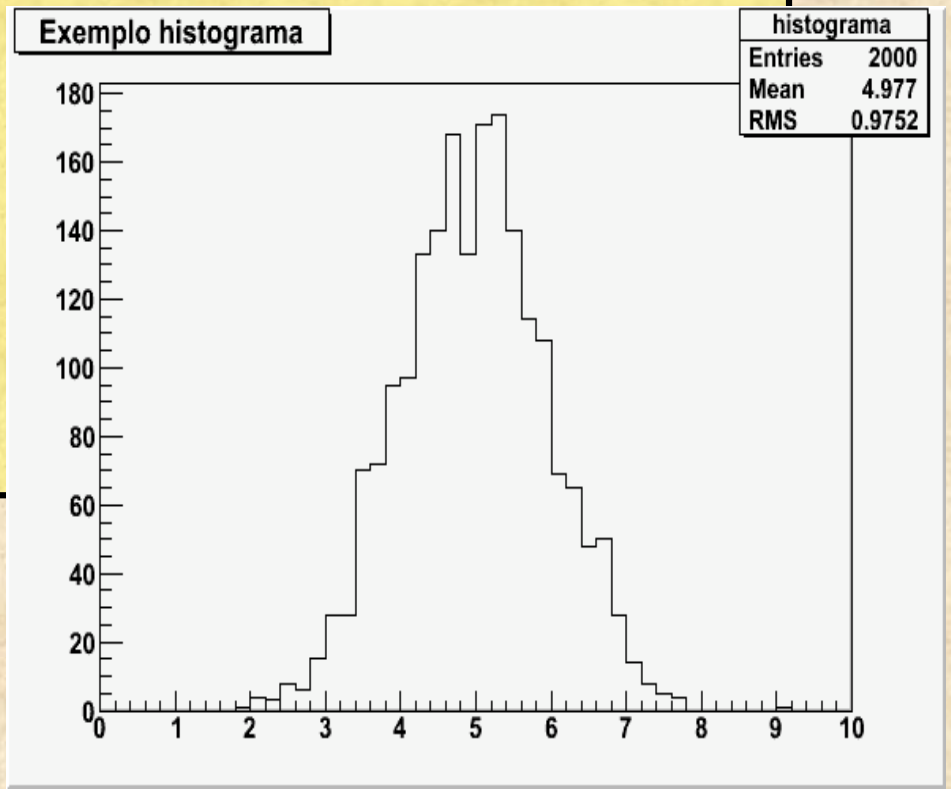
Histogramas de 1 dimensão

- Criando um Histograma de 1 dimensão
  - TH1F *h = new TH1F("nome","título", Nbins, Xmin, Xmax);
  - TH1F h ("nome","título", Nbins, Xmin, Xmax);

```
void exemplo_TH1()
{
  TRandom *r = new TRandom();
  TH1F *h1 = new TH1F("histograma","Exemplo histograma",50,0,10);
  for(int i = 0;i<2000;i++)
  {
    float x = r->Gaus(5,1);
    h1->Fill(x);
  }
  h1->Draw();
}
```

Para rodar esse exemplo, assim como os Seguintes, salve-o em um arquivo, por Exemplo, exemplo_TH1.C e digite, no prompt do ROOT

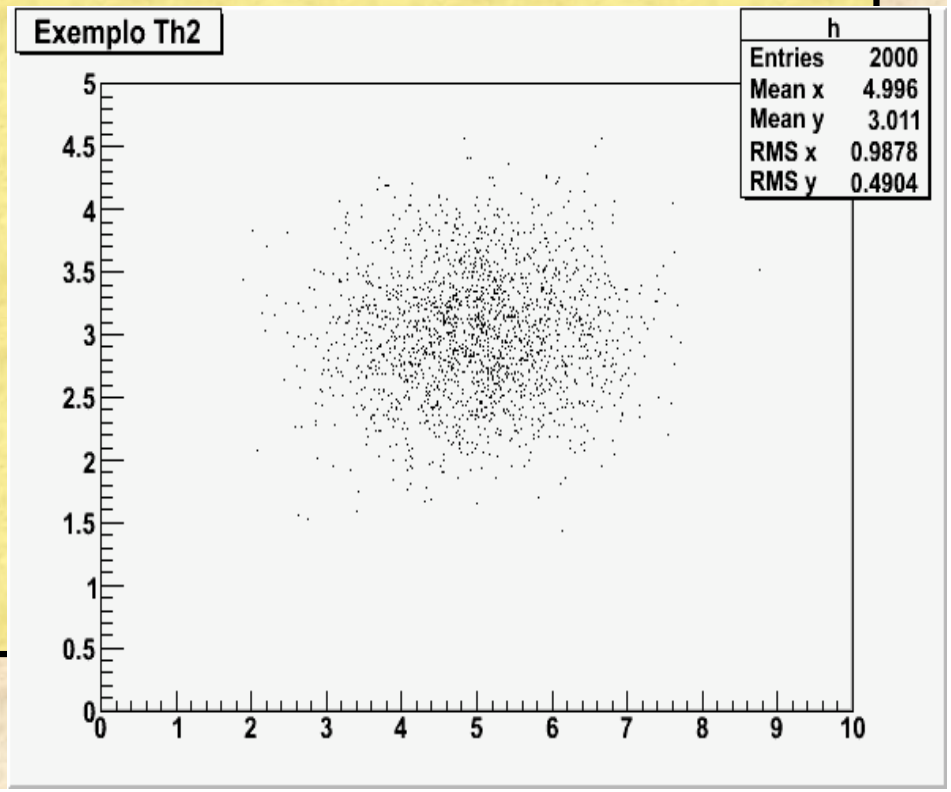
root [0] .x exemplo_TH1.C



Histogramas de 2 dimensões

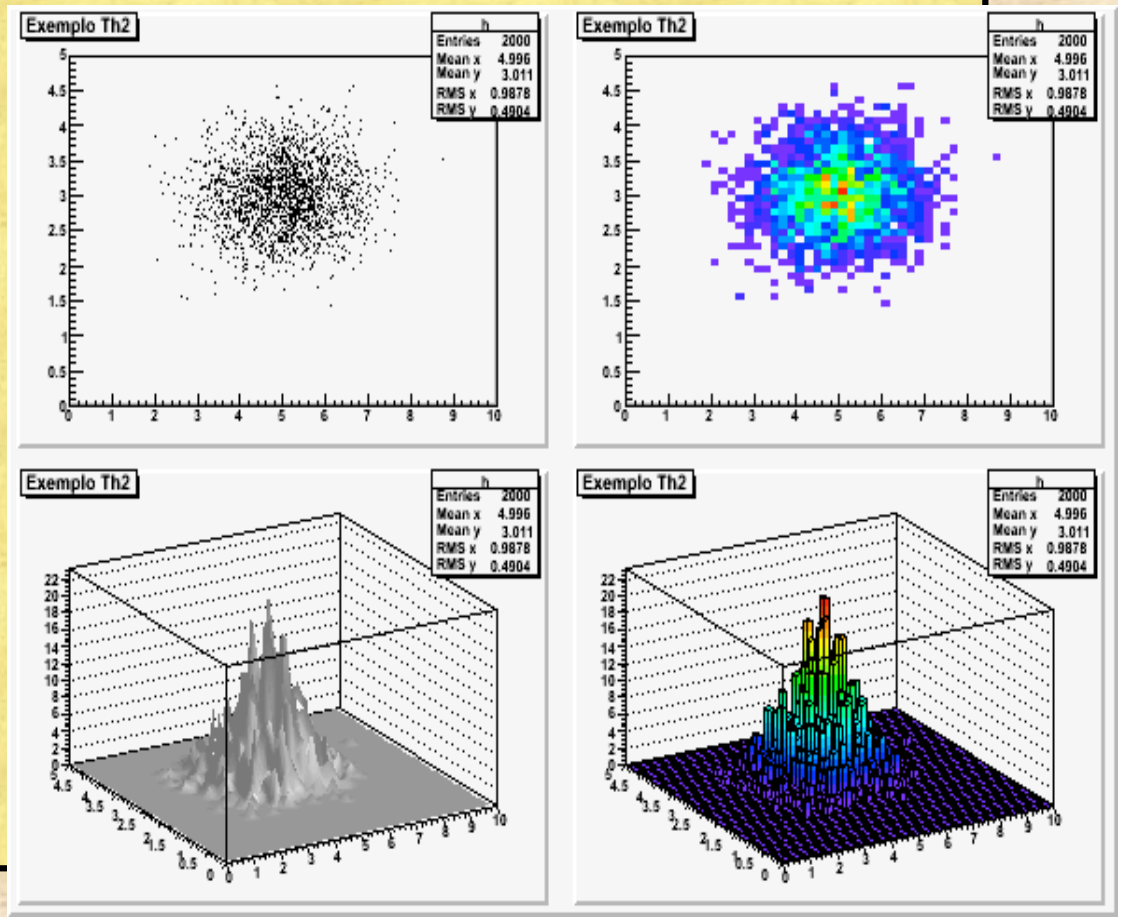
- **Muito similar ao TH1**
  - TH2F *h = new TH2F("nome","título", NbinsX, Xmin, Xmax, NBinsY, Ymin, Ymax);
  - TH2F h ("nome","título", NbinsX, Xmin, Xmax, NbinsY, Ymin,Ymax);

```
void exemplo_TH2()
{
  TRandom *r = new TRandom();
  TH2F *h2 = new TH2F("h","Exemplo Th2",50,0,10,50,0,5);
  for(int i = 0;i<2000;i++)
  {
    float x = r->Gaus(5,1);
    float y = r->Gaus(3,0.5);
    h2->Fill(x,y);
  }
  h2->Draw();
}
```



Dividindo uma tela

```
void exemplo_TH2_2()
 {
      gStyle->SetPalette(1,0);
      TRandom *r = new TRandom();
      TH2F *h2 = new TH2F("h","Exemplo Th2",50,0,10,50,0,5);
      for(int i = 0;i<2000;i++)
      {
        float x = r->Gaus(5,1);
        float y = r->Gaus(3,0.5);
        h2->Fill(x,y);
      }
      TCanvas *c = new TCanvas();
      c->Divide(2,2);
      c->cd(1);
      h2->Draw();
      c->cd(2);
      h2->Draw("col");
      c->cd(3);
      h2->Draw("surf4");
      c->cd(4);
      h2->Draw("lego2");
 }
```



Exemplo: Propagação de incertezas com Monte Carlo

- Distância focal de uma lente convergente

$$\frac{1}{f} = \frac{1}{i} + \frac{1}{o}$$

- i = 10.5 $\pm$ 0.8 cm
- o = 25.3 $\pm$ 0.2 cm

- Incerteza f = ????

MC_lente.C - /Volumes/Data/Users/suaide/cursoRoot/

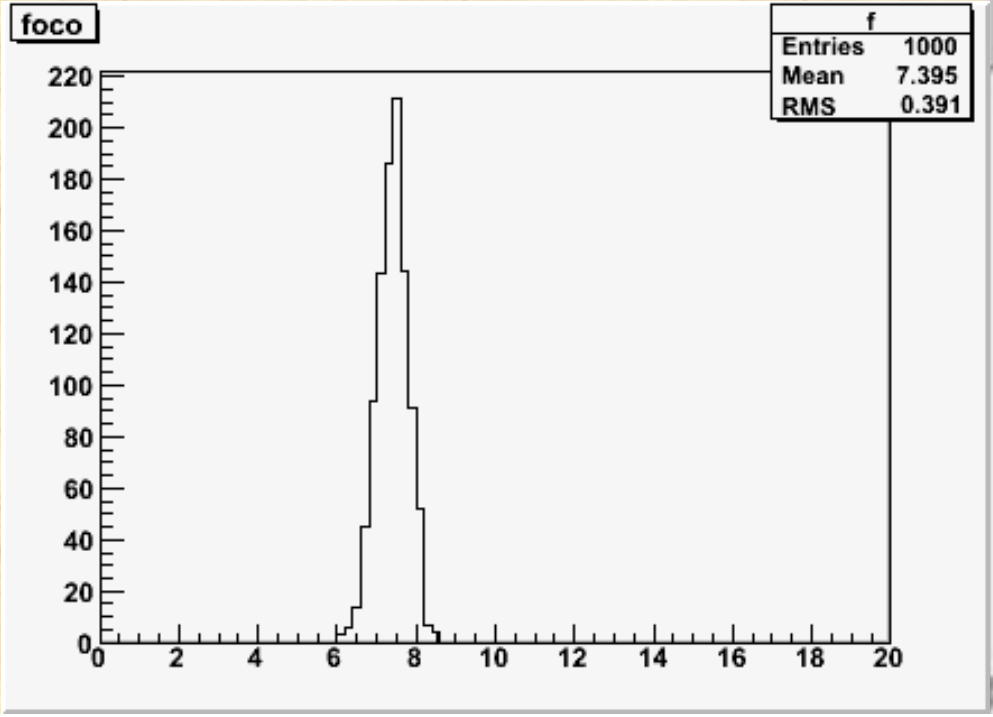
File Edit Search Preferences Shell Macro Windows Help

```
1 TH1F *hist;
2 float MC_lente(float o, float so, float i, float si, int N)
3 {
4   hist = new TH1F("f","foco",100,0,20);
5
6   TRandom *r = new TRandom();
7   for(int k = 0; k<N; k++)
8   {
9     float I = r->Gaus(i,si);
10    float O = r->Gaus(o,so);
11    float F = I*O/(I+O);
12    hist->Fill(F);
13  }
14  float RMS =hist->GetRMS();
15  return RMS;
16 }
17
```

Exemplo: Propagação de incertezas com Monte Carlo

- **No ROOT digite:**

```
root [0] .L MC_lente.C

root [1] MC_lente(25.3, 0.2, 10.5, 0.8, 1000)
(float)3.9100208785171509e-01

root [2] hist->Draw()
```



MC_lente.C - /Volumes/Data/Users/suaide/cursoRoot/

File Edit Search Preferences Shell Macro Windows Help

```
1 TH1F *hist;
2 float MC_lente(float o, float so, float i, float si, int N)
3 {
4   hist = new TH1F("f","foco",100,0,20);
5
6   TRandom *r = new TRandom();
7   for(int k = 0; k<N; k++)
8   {
9     float I = r->Gaus(i,si);
10    float O = r->Gaus(o,so);
11    float F = I*O/(I+O);
12    hist->Fill(F);
13  }
14  float RMS =hist->GetRMS();
15  return RMS;
16 }
17
```

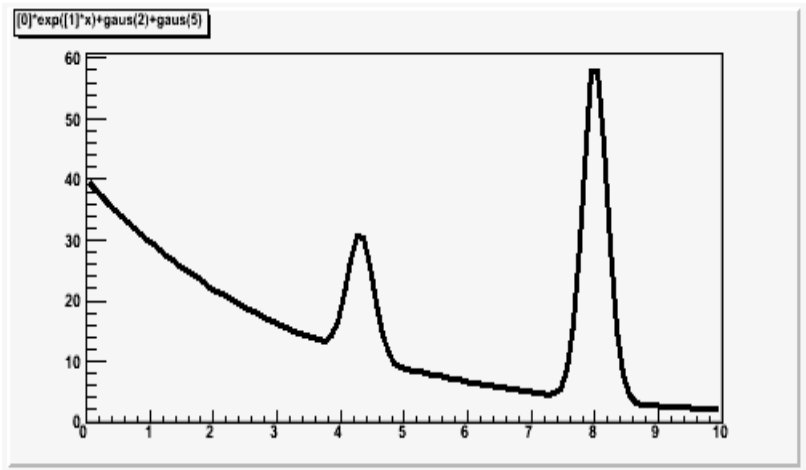
Monte Carlo com TF1

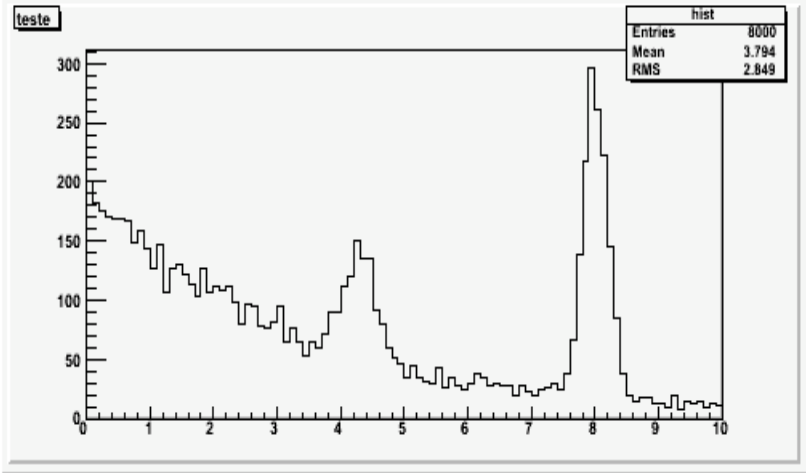
Funções pré-definidas.
O Root possui algumas funções pré-definidas, tais como:

`gaus` – Gaussiana
`polN` – polinômio de grau N (N = 0...9)
`landau` – distribuição de Landau
`expo` – exp([0]+[1]*x)

- Eu sei a distribuição de probabilidad

```
void MC_TF1()
{
  TF1 *f = new TF1("f","[0]*exp([1]*x)+gaus(2)+gaus(5)",0,10);
  f->SetParameter(0,40);
  f->SetParameter(1,-0.3);
  f->SetParameter(2,20);
  f->SetParameter(3,4.3);
  f->SetParameter(4,0.2);
  f->SetParameter(5,56);
  f->SetParameter(6,8);
  f->SetParameter(7,0.2);

  TCanvas *c = new TCanvas();
  c->Divide(1,2);
  c->cd(1);
  f->Draw();

  TH1F* h = new TH1F("hist", "teste",100,0,10);
  for(int i=0; i<8000; i++) h->Fill(f->GetRandom());

  c->cd(2);
  h->Draw();
}
```





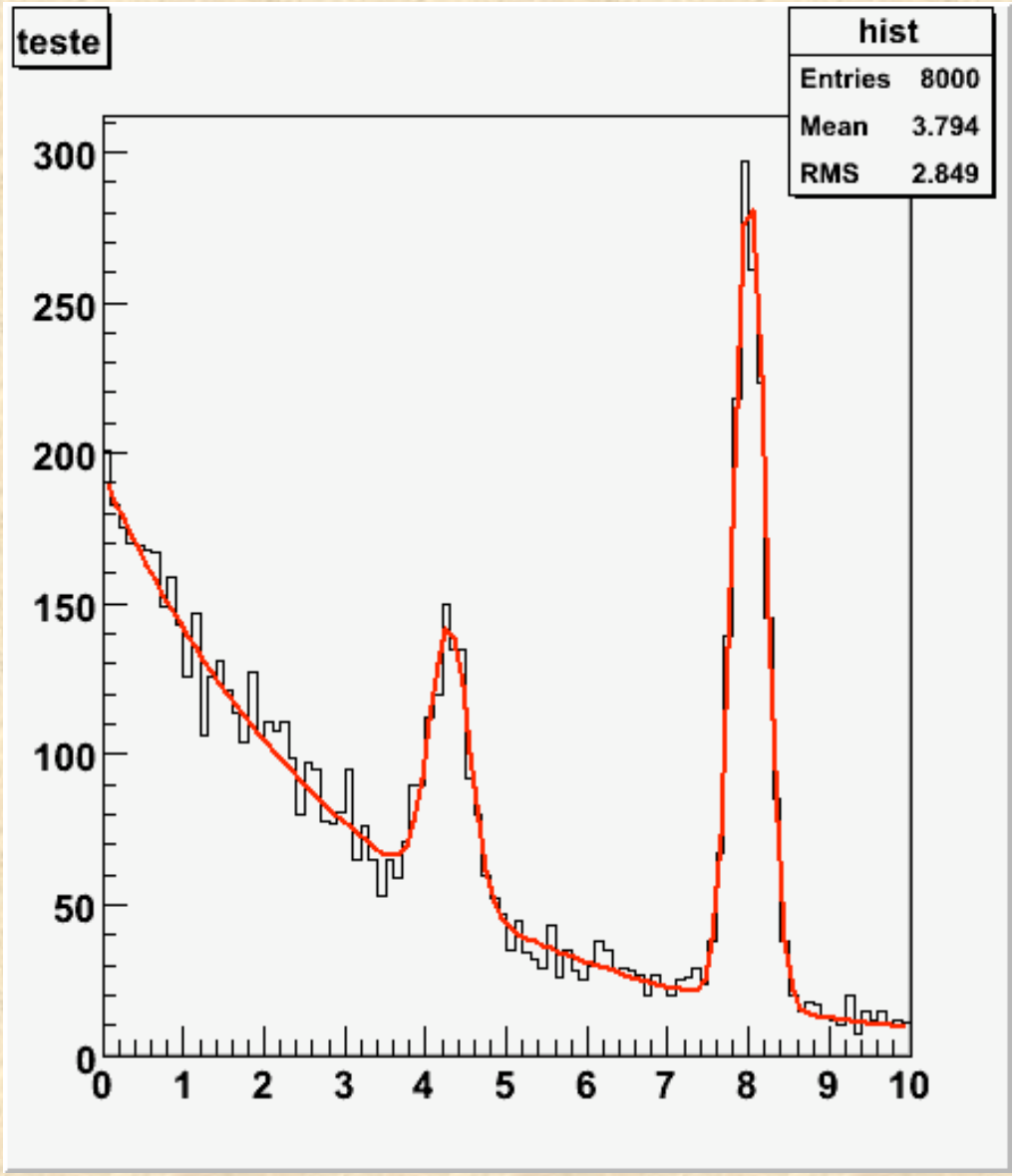
Outro exemplo de fit (com histogramas)

`[0]*exp([1]*x)+gaus(2)+gaus(5)`

- Vamos ajustar o histograma

```
new Tcanvas();
h->Draw();
f->SetLineColor(2);
f->SetLineWidth(2);
h->Fit(f);
```

- Como extrair informações da função ajustada que não seja pela tela?



Outro exemplo de fit (com histogramas)

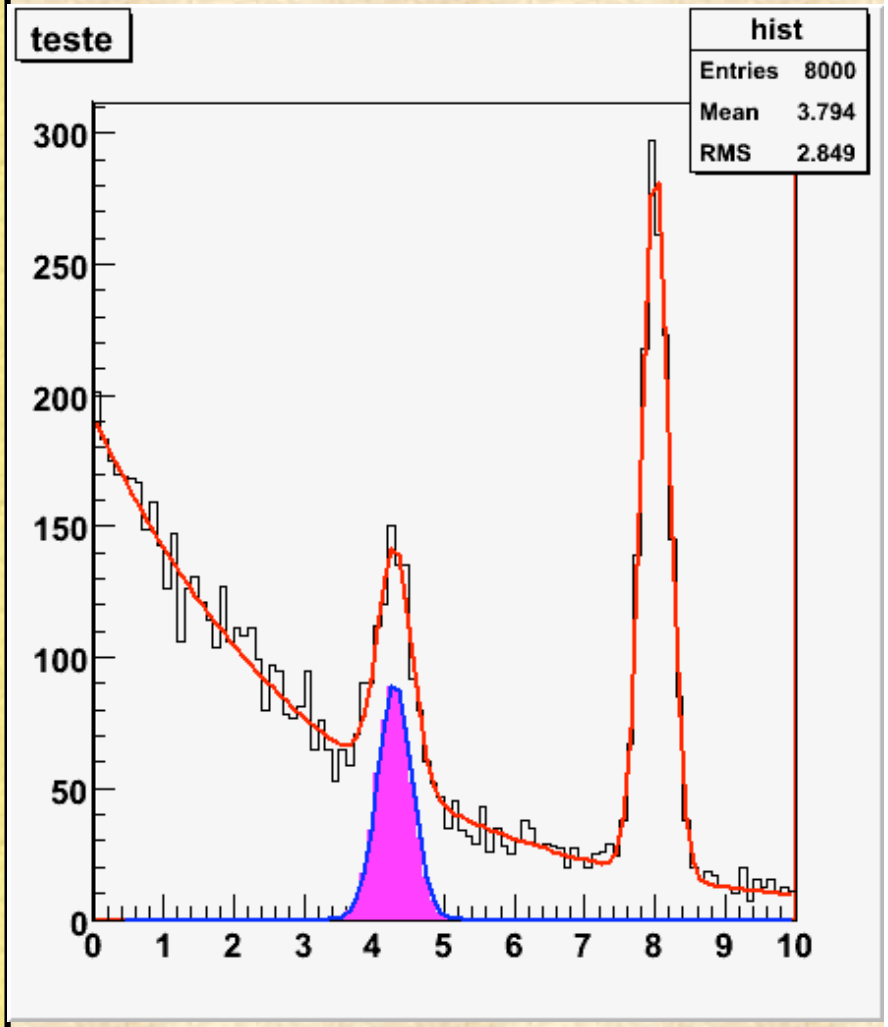
`[0]*exp([1]*x)+gaus(2)+gaus(5)`

- Qual o X^2_{red} do ajuste?

```
f->GetChisquare()/f->GetNDF();
1.36299
```

- Qual e a integral no primeiro pico?

```
TF1 *peak = new TF1("peak","gaus(0)",0,10);
peak->SetParameter(0,f->GetParameter(2));
peak->SetParameter(1,f->GetParameter(3));
peak->SetParameter(2,f->GetParameter(4));
peak->SetLineColor(4);
peak->SetLineWidth(2);
peak->SetFillColor(6);
peak->SetFillStyle(1000);
peak->Draw("same l f");

peak->Integral(0,10);
53.5103
```



Cálculo vetorial no ROOT

- **O Root possui classes para cálculos vetoriais em Física (ver http://root.cern.ch)**

Several documents describing these classes are listed below:

- The main histogram class is documented in class TGeoManager.
- The Chapter about the Physics Vectors classes in the Users Guide

TFeldmanCousins	calculate the CL upper limit using the Feldman-Cousins method
TGenPhaseSpace	Simple Phase Space Generator
TLorentzRotation	Lorentz transformations including boosts and rotations
TLorentzVector	A four vector with (-,-,-,+) metric
TQuaternion	a quaternion class
TRobustEstimator	Minimum Covariance Determinant Estimator
TRolke	calculate confidence limits using the Rolke method
TRotation	Rotations of TVector3 objects
TVector2	A 2D physics vector
TVector3	A 3D physics vector

Alguns exemplos de vetores

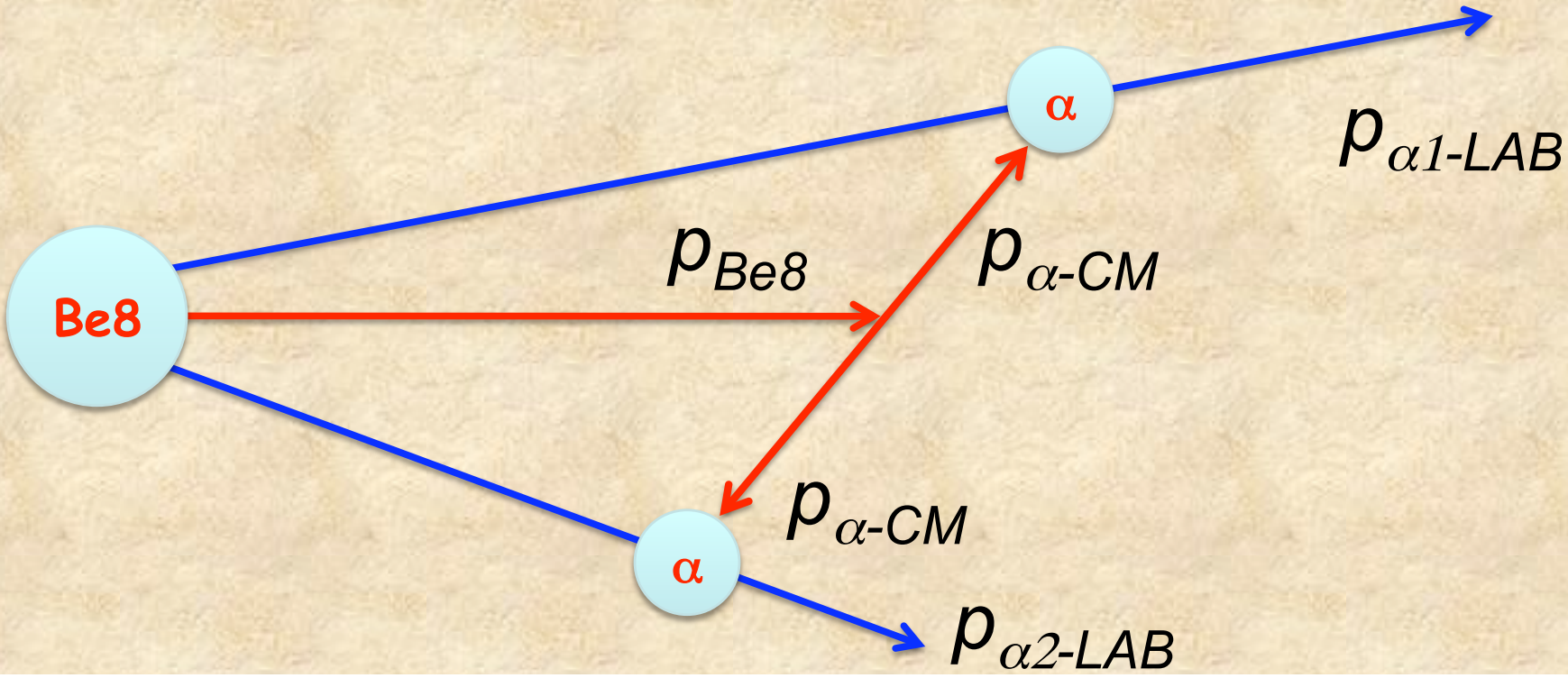
No prompt do ROOT, digite:

```
root [0] TVector3 A(3,2,6)
root [1] A.Print()
TVector3 A 3D physics vector (x,y,z)=(3.000000,2.000000,6.000000)
(rho,theta,phi)=(7.000000,31.002719,33.690068)

root [2] TVector3 B(2,7,8)
root [3] B.Print()
TVector3 A 3D physics vector (x,y,z)=(2.000000,7.000000,8.000000)
(rho,theta,phi)=(10.816654,42.302625,74.054604)

root [4] TVector3 C = A-B
root [5] C.Print()
TVector3 A 3D physics vector (x,y,z)=(1.000000,-5.000000,-2.000000)
(rho,theta,phi)=(5.477226,111.416714,-78.690068)


root [6] A.Dot(B)
(const Double_t)6.8000000000000000e+01

root [7] TVector3 D=A.Cross(B)
root [8] D.Print()
TVector3 A 3D physics vector (x,y,z)=(-26.000000,-12.000000,17.000000)
(rho,theta,phi)=(33.301652,59.303846,-155.224859)
```

Monte Carlo: Decaimento espontâneo do Berílio-8

Em muitas reações nucleares formamos Be-8
Este é instável e decai quase que instantaneamente
Em duas partículas alpha (He-4)

Vamos simular o ângulo e decaimento destas partículas
Alpha sabendo a distribuição de energia do Berílio
Formado na reação.



Vamos resolver o problema teóricamente

- No referencial do C.M. do berílio, o momento relativo entre as duas partículas alpha é constante = 18 MeV/c

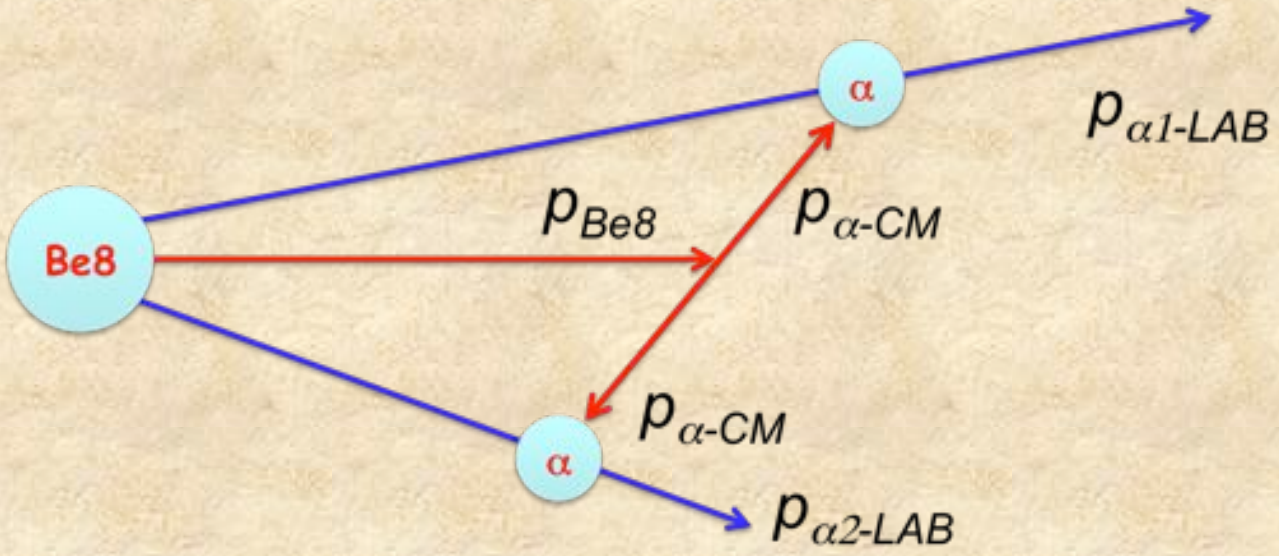
- Neste caso, podemos transformar para o referencial do laboratório e calcular o momento das alphas neste referencial

No C.M. $\vec{p}_{total} = \vec{p}_{\alpha 1}^{CM} + \vec{p}_{\alpha 2}^{CM} = 0 \Rightarrow \vec{p}_{\alpha 1}^{CM} = -\vec{p}_{\alpha 2}^{CM}$

$$\left|\vec{p}_{\alpha 1}^{CM}\right| = 18\,MeV/c$$

$$\vec{p}_{\alpha 1}^{lab} = \vec{p}_{Be} + \vec{p}_{\alpha 1}^{CM}$$

$$\vec{p}_{\alpha 2}^{lab} = \vec{p}_{Be} + \vec{p}_{\alpha 2}^{CM}$$



Vamos resolver o problema teóricamente

- Só que, no Referencial do CM, a partícula alpha pode decair em qualquer direção

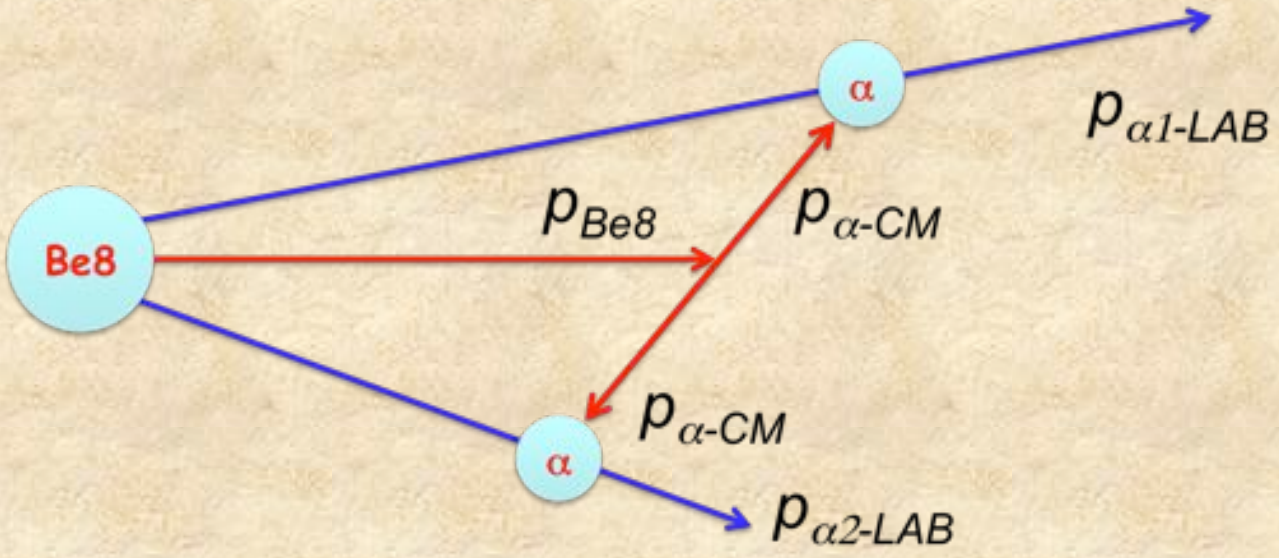
- Além disto, a distribuição de momento (p_{Be8}) do Berílio depende da reação nuclear que ocorreu

No C.M. $\vec{p}_{total} = \vec{p}^{CM}_{\alpha 1} + \vec{p}^{CM}_{\alpha 2} = 0 \Rightarrow \vec{p}^{CM}_{\alpha 1} = -\vec{p}^{CM}_{\alpha 2}$

$\left|\vec{p}^{CM}_{\alpha 1}\right| = 18\,MeV/c$

$\vec{p}^{lab}_{\alpha 1} = \vec{p}_{Be} + \vec{p}^{CM}_{\alpha 1}$

$\vec{p}^{lab}_{\alpha 2} = \vec{p}_{Be} + \vec{p}^{CM}_{\alpha 2}$



Simulação

- Para cada evento simulado
  - No C.M. sortear a direção da alpha1. Calcular o seu momento, sabendo que o módulo é 18 MeV/c
  - Por conservação de momento, obter o momento da alpha 2 no C.M.
  - Sabendo a física da reação, sortear o momento do Be-8 no laboratório
  - Fazer a mudança de referencial das partículas alfa do C.M. para o laboratório
  - Preencher histogramas com energias e ângulos das alphas.
- Repetir o processo acima para quantos eventos desejar.

Iniciando o programa Be8.C

- Primeiramente vamos definir algumas variáveis que vão ser utilizadas no programa

- Variáveis globais
  - fora de funções

```
float pi = 3.1415926;
float pAlpha = 18;  //MeV/c
float mp = 931.5; //MeV/c2
float mn = 934.5; //MeV/c2

float mBe    = 4*mp + 4*mn + 4.942;
float mAlpha = 2*mp + 2*mn + 2.425;
```

Sorteando a particula alpha no C.M.

- **Vamos fazer uma função que sorteia o ângulo da alpha no CM, com módulo de momento fíxo e retorna um vetor com o momento**

```
TVector3 sorteiaAlpha1()
{
  // sorteia direcao de decaimento de uma das alphas
  // o decaimento é isotrópico
  float thetaAlpha1 = gRandom->Uniform(-pi,pi);
  float phiAlpha1   = gRandom->Uniform(0,2*pi);

  // cria os vetores de decaimento das alphas no
  // centro de massa do berÌlio
  TVector3 pAlpha1CM;
  pAlpha1CM.SetMagThetaPhi(pAlpha,thetaAlpha1,phiAlpha1);

  return pAlpha1CM;
}
```

Fazendo o programa

- **Primeira parte**
  - Iniciar a função simula
  - Definir distribuição de velocidades do Be no laboratório
  - Definir alguns histogramas

```
void simula(float VMAX, int N)
{
  gStyle->SetPalette(1,0);

  // A distr. de probabilidades eh dada por uma
  // distr de Maxwell-Boltzmann
  TF1 *prob = new TF1("prob","(x^2)*exp(-[0]*x^2)",0,0.2);
  prob->SetParameter(0,1.0/(VMAX*VMAX));

  TH1F *h1 = new TH1F("Ealpha","Energia das alpha (MeV)", 1000, 0, 20);
  TH1F *h2 = new TH1F("Aalpha","Angulo das alpha (MeV)", 1000, 0, 2*pi);
  TH1F *h3 = new TH1F("EBe","Energia dos berilios (MeV)", 1000, 0, 20);

  TH2F *h4 = new TH2F("EA","EnergiaXangulo", 1000, 0, 2*pi, 1000, 0, 20);
```

Note que não terminamos a função. Ela continua na próxima página

Fazendo o programa

- Agora fazemos o LOOP com a simulação
  - O LOOP não está completo. Ainda falta coisas
  - Neste ponto, sorteamos a velociade do Berílio, calculamos o seu momento.
  - Fazemos o mesmo para alfas e convertemos para Lab.

```
for(int i = 0; i<N; i++)
{
   // cria o vetor de momento do berilio no espaco
   TVector3 pBe;
   float v = prob->GetRandom(); // em unidades de C
   float p = mBe*v;             // em MeV/c
   pBe.SetMagThetaPhi(p,0,0);
   float EBe8 = pBe.Mag2()/(2.0*mBe); // em MeV

   //sorteia particula alpha1 e calcula alpha2
   TVector3 pAlpha1CM = sorteiaAlpha1();
   TVector3 pAlpha2CM = -pAlpha1CM;

   // converte para o referencial do laboratorio
   TVector3 pAlpha1 = pAlpha1CM + pBe;
   TVector3 pAlpha2 = pAlpha2CM + pBe;
```

Note que não terminamos a função. Ela continua na próxima página

Fazendo o programa

- **Ainda dentro do LOOP, calculamos as energias e ângulos das alphas com base nos momentos sorteados. Preenchemos histogramas e fechamos o LOOP**

```
  // calcula energia cinetica das alphas
  // E = p≤/2m
  float EAlpha1 = pAlpha1.Mag2()/(2.0*mAlpha);
  float EAlpha2 = pAlpha2.Mag2()/(2.0*mAlpha);

  // calcula os angulos relativos entre as
  // alphas e o berÌlio (angle1 e angle2)
  float angle1 = pAlpha1.Angle(pBe);
  float angle2 = pAlpha2.Angle(pBe);

  h1->Fill(EAlpha1);
  h1->Fill(EAlpha2);
  h2->Fill(angle1);
  h2->Fill(angle2);
  h3->Fill(EBe8);
  h4->Fill(angle1,EAlpha1);
  h4->Fill(angle2,EAlpha2);
}   // << ----- Fechamos o LOOP
```

Note que não terminamos a função. Ela continua na próxima página

Fazendo o programa

- Agora que a simulação está pronta, desenhamos os gráficos e terminamos o programa

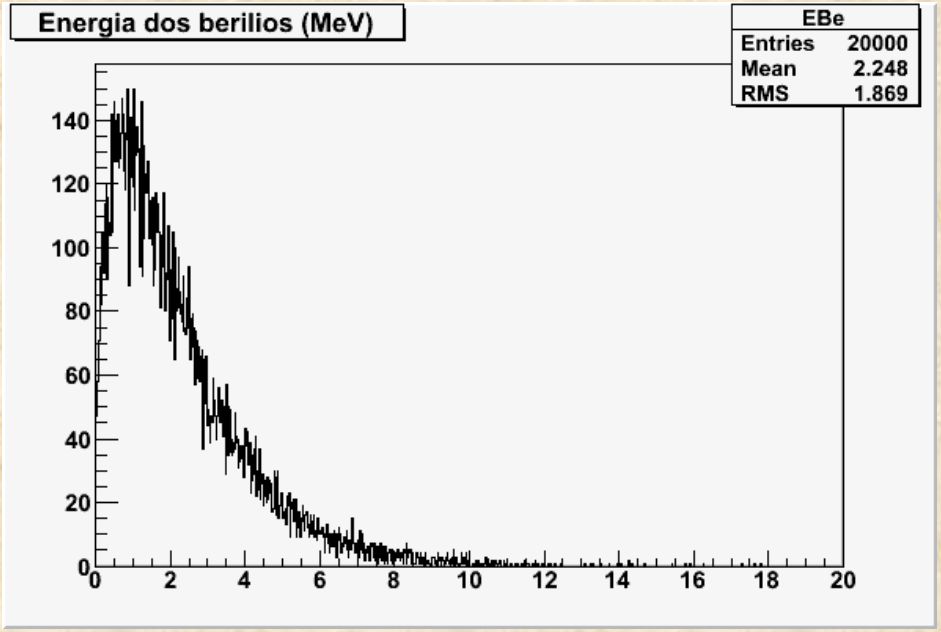
```
TCanvas *c1 = new TCanvas();
h3->Draw();
TCanvas *c2 = new TCanvas();
h1->Draw();
TCanvas *c3 = new TCanvas();
h2->Draw();
TCanvas *c4 = new TCanvas();
h4->Draw("col");

}
```

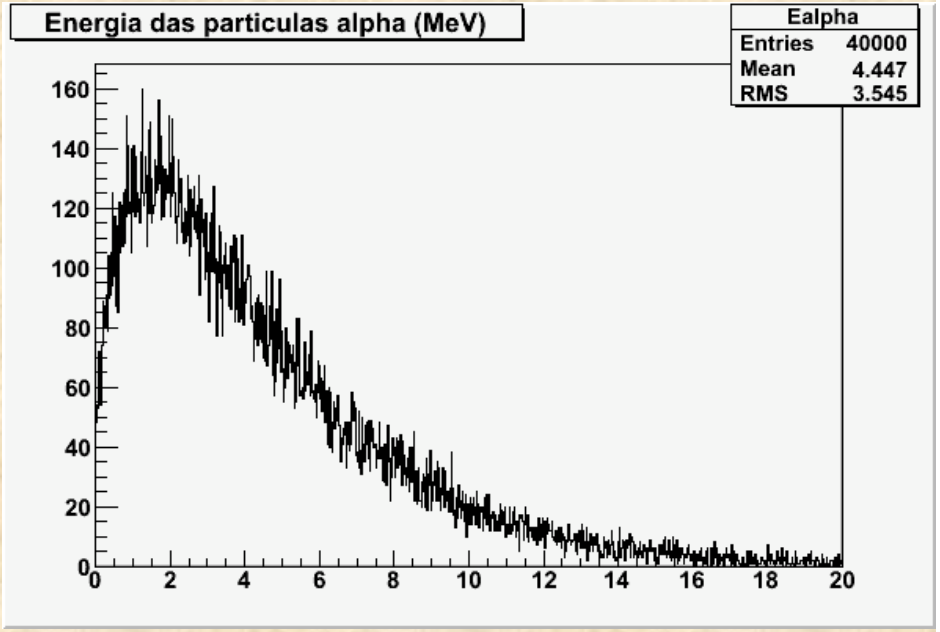
Agora terminamos o programa, que foi salvo como Be8.C

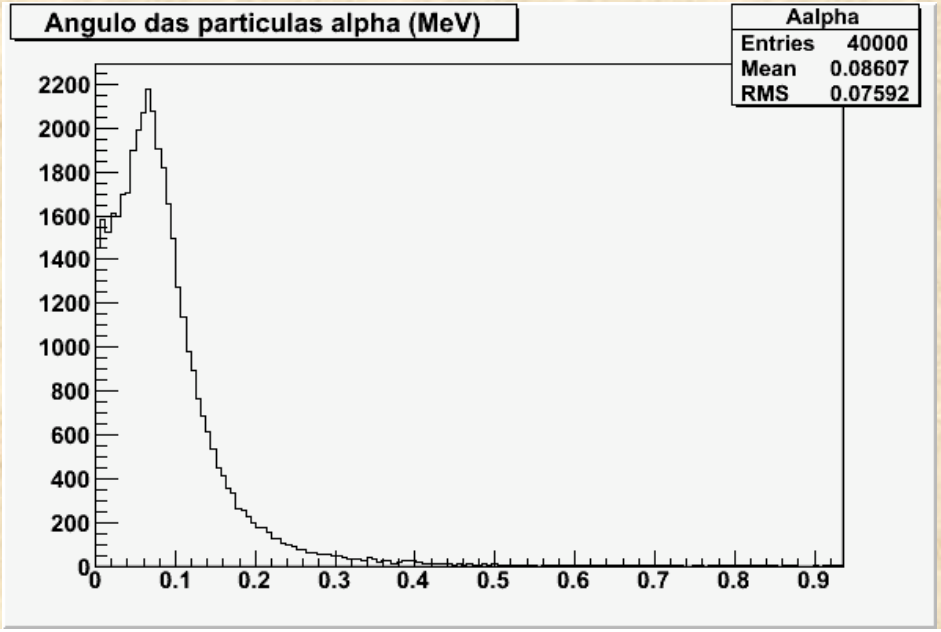
- Para executar, no prompt do ROOT, digite

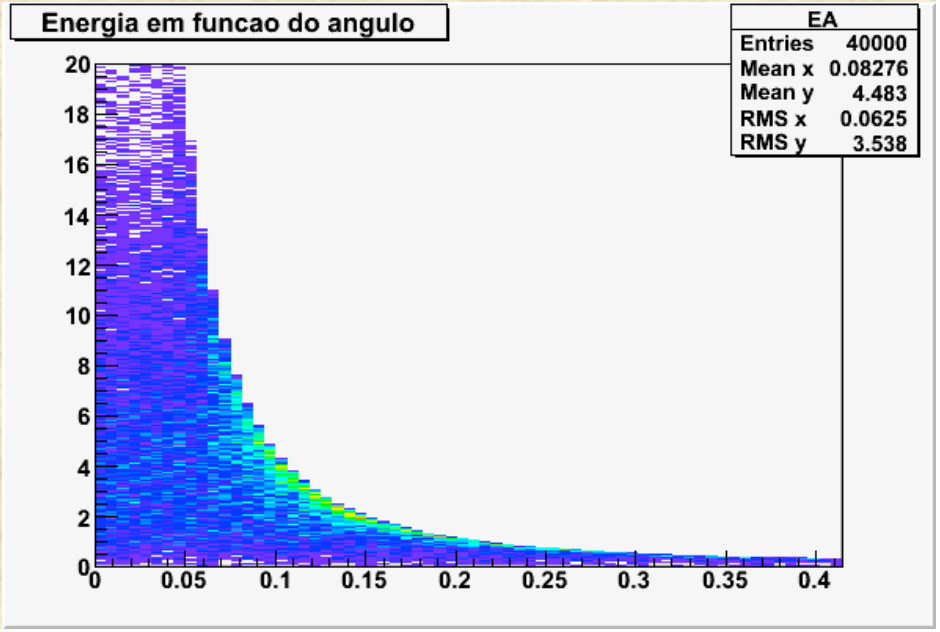
```
.L Be8.C
simula(0.02, 20000)
```

Resultados


Energia dos berilios (MeV)
EBe
Entries 20000
Mean 2.248
RMS 1.869


Energia das particulas alpha (MeV)
Ealpha
Entries 40000
Mean 4.447
RMS 3.545


Angulo das particulas alpha (MeV)
Aalpha
Entries 40000
Mean 0.08607
RMS 0.07592


Energia em funcao do angulo
EA
Entries 40000
Mean x 0.08276
Mean y 4.483
RMS x 0.0625
RMS y 3.538

Resumindo

- Vimos algumas funcionalidades do ROOT
  - Ajuste de funções
  - Histogramas em 1 e 2 dimensões
  - TLegend
  - Vetores (TVector2 e TVector3)
  - Alguns exemplos de Monte Carlo

- O ROOT é muito versátil para análise de dados e simulações. Usem!!!