Exercícios de

Física Computacional

(Parte 3)

Mestrado em Engenharia Física-Tecnológica (MEFT)

Rui Coelho

Departamento de Física do Instituto Superior Técnico

Ano Lectivo: 2019-20

rui.alves.coelho@tecnico.ulisboa.pt

versão: 1 de Outubro de 2019

3. ROOT

Exercício 30. Lance o root em sessão interactiva e utilize o interpretador de ROOT (consulte o manual online das classes envolvidas para aferir qual o método/sintaxe a utilizar) para correr código C++ que realize as seguintes tarefas (*adaptado de Barão 2016*):

a) Faça um array de dois inteiros sem inicializar os valores e verifique os valores existentes em cada posição do array.
b) Crie um array de 2 histogramas TH1F utilizando o default constructor.
c) Crie um array de 2 histogramas TH1F com as seguintes características numa só linha de coman- dos: ***10*** canais ("*number of bins*") e limites inferior e superior respectivamente, 0.5 e 10.5
d) Crie um array de 2 histogramas TH1F com ***5*** canais ("*number of bins*") de largura variável dada por: 0.5, 1.5, 4.5, 2.0, 1.0
e) Crie agora o array de 2 histogramas TH1F utilizando o default constructor e inicializando-os de seguida com as características da alínea c)
f) Crie agora um array de 2 ponteiros que aponte para os histogramas com características da alínea c)
g) Construa uma macro *mHisto.cpp* onde reuna o conjunto de operações da alíena d) e execute-a.

Exercício 31. Elabore uma macro ***histogram_test.cpp*** onde construa um array de três objectos histograma que armazene floats (TH1F) entre os valores -10 e 10, com canais de largura 0.2

a) O primeiro histograma é preenchido com números aleatórios (#=10000) entre -5 e 5 (distribuição uniforme).
b) O segundo e terceiro histograma são preenchidos (#=10000) a partir das funções geradoras dadas por f(x) e g(x) (use σ=2 mas dado como parâmetro da função g(x) !) respectivamente

$$f(x) = 2x^2$$

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

Deverá imprimir os 3 histogramas num ficheiro *histograma_teste.eps* num layout de 3 colunas por 1 linha.

Exercício 32. Neste exercício vamos verificar a validade do teorema do limite central usando : função de distribuição de probabilidade uniforme f(x) e função de distribuição dada por $g(x)=x^2$. Os histogramas a utilizar para "*samplar*" (dimensão da amostra 100.000) as populações devem estar definidos com 500 canais e os números aleatórios gerados devem estar entre -4 e 4. Deverá produzir as figuras necessárias que mostrem:

a) Um histograma da amostra gerada para cada distribuição (com indicação da média e desvio padrão)
b) O histograma da distribuição da média amostral, realizando 10.000 experiências com cada função geradora (com indicação da média e desvio padrão).

O que conclui face aos previsto pelo Teorema do Limite Central ?

Exercício 33. Neste exercício vamos fazer histogramas e ajustes de "*fit*" apenas a parte do histograma. Considere então a função geradora dada por:

```
TF1 *f1 = new TF1("f1","[0] +[1]*x +gaus(2)",0,5);
f1->SetParameters(6,-1,5,3,0.2);
```

a) Construa, usando a função geradora f(x), o histograma usando 100 canais, de x=0 a x=5, com 4000 elementos na amostra.
b) Faça o ajuste com uma função linear do tipo

```
double fline(double *x, double *par)
{
   return par[0] + par[1]*x[0];
}
```

mas não obtenha/adicione imediatamente o ajuste ao histograma (ver opções no método Fit() da classe TH1F).

c) Defina agora 2 novas funções (do tipo "*fline*"), definidas respectivamente em x=[0,2.5] e x=[3.5,5]. Para definir parâmetros de cada uma destas funções, use os métodos GetParameters() e SetParameters().
d) Adicione finalmente ao Histograma estas 2 funções de "fit" usando o método GetListOfFunctions()->Add(...).

Exercício 34*. Neste exercício vamos ler dados a partir de ficheiro de texto, fazer o(s) respectivo(s) gráfico(s) num mesmo sistema de eixos e imprimir para ficheiro.

O ficheiro em questão é constituído por 3 colunas. Em cada linha do ficheiro, temos um tripleto (*x,y,type*) em que (x,y) são as coordenadas dos pontos a serem impressos no gráfico e a variável "type" é um índice que distingue cada curva a ser impressa (a começar em 1).

Pretende-se um código ***plot_continuum.cpp*** que leia o ficheiro n5.csqr e faça o gráfico das várias curvas (x,y) incluídas no ficheiro num mesmo sistema de eixos. No final deve gravar a figura num ficheiro ***n5_continuum.eps***.

Sugestões:

a) Desenvolva uma classe Curves que vai conter cada curva a representar o gráfico

```
class Curves {
              vector<double> x,y;
       public:
              void setX(double a) {x.push_back(a);}
              void setY(double b) {y.push_back(b);}
              vector<double> getX() {return x;}
              vector<double> getY() {return y;}
              void Reset() {x.clear();y.clear();}
};
```

b) Desenvolva uma função `read_spec` que leia o ficheiro e retorne um vector de objectos da classe Curves.

```
vector<Curves> read_spec(string name)
```

c) Use a classe `TMultiGraph` para adicionar cada uma das curvas que leu/identificou ao gráfico.