Exercícios de

Física Computacional

(Parte 8)

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

8. Métodos de Monte-Carlo

Exercício 53. (*adaptado de Barão 2016* Os geradores de números aleatórios usam relações do tipo:

$$I_{i+1} = (aI_i + c)\%m$$

a) Construamos uma classe em C++ ***FCrand*** que implemente o método das congruências lineares para geração de números aleatórios. Use somente um construtor que possua um parâmetro semente e que possa também funcionar como default constructor a partir da função time. O header file ***FCrand.h*** (incompleto) é apresentado abaixo

```
class FCrand {
public:
//seed number (prototype incompleto)
FCrand(int seed);
//generate one random between [0,1]
float GetRandom();
//generate one random between [min,max]
float GetRandom(float min, float max);
//generate N randoms between [0,1]
float* GetRandom(int N);
//generate N randoms between [min,max]
float* GetRandom(int N, float min, float max);
private:
...
};
```

b) Use agora um gerador constituído por: a = 65, c = 319, m = 65537. Obtenha uma amostra com um milhão de elementos entre 0 e 10, faça o histograma (produza a figura ***Random.eps*** respectiva) e determine o valor médio e o desvio padrão amostral. Corresponde ao esperado ?

Exercício 54. Construa um código ***myDistribution.cpp*** onde implemente e teste uma função (código ***ExpFunct.cpp***) que gere números aleatórios seguindo uma distribuição exponencial de decaimento ***ga*** i.e. p(x)~ga*exp(-ga*x) a partir de distribuições U(0,1). Obtenha uma amostra com 10000 números aleatórios entre 0 e 100 e faça o respectivo histograma, comparando com a mesma distribuição e amostragem feita a partir de função TF1 do ROOT e método GetRandom(). Os histogramas, sobrepostos no mesmo plot, devem ser gravados numa figura ***ExpHistogram.eps***.

Exercício 55. (*adaptado de Barão 2016)* Determinemos a superfície de um de círculo de raio 1, isto é, o valor de π. Consideremos um grande número N de pares de números aleatórios (r1,r2), usando para tal distribuições aleatórias entre 0 e 1 (pode usar os geradores de números aleatórios do ROOT bem como funções da classe TF1).

a) Construa um algoritmo que determine o valor de π (estimativo e erro após 10000 experiências aleatórias) em função da dimensão da amostra N (1000, 10000, 100000). Deverá construir para o efeito um programa ***calculatePI.cpp*** e produzir figuras com os histogramas respectivos i.e. **Pie4.eps**, **Pie5.eps** e **Pie6.eps**. Todo o código pode estar apenas no programa principal, colocado na pasta *serie_problemas*.

Exercício 56 Elabore um programa ***test_UniformSampling.cpp*** onde implemente o algoritmo (não precisa de ser como função, pode ser no programa principal) de Uniform Sampling para estimar o valor de $\int_{0.2\pi}^{0.5\pi} \cos(x)\,dx$. Produza uma figura ***Integral_cos.eps*** onde mostre o valor do integral usando amostras com dimensão desde 20 até 100000. Produza outra figura equivalente ***IntegralError_cos.eps*** desta feita para o erro de integração.

Exercício 57 Utilizando o método de Acceptance-Rejection, calcule o valor do integral e erro da função $f(x)=3x^2$ no intervalo [3,7] usando como maximizante (constante) os valores 90, 150 e 200. O que pode concluir dos valores obtidos ? Ajuda : talvez seja útil obter o gráfico com a distribuição obtida nos 3 casos. Código a produzir: ***Accept-test.cpp. Qualquer figura produzida deve ser grava em ficheiro .eps***

Exercício 58 (*adaptado de Barão 2016).* A classe ***Integrator*** pode ser extendida de forma a incluir os métodos de integração de Monte-Carlo, ***simples***, ***importance sampling*** e de ***aceitação-rejeição***. No método de "importance sampling" é usada uma função auxiliar p(x) para geração dos números aleatórios, de forma a minimizar-se o erro da integração. A geração dos aleatórios é feita, recorrendo à função acumulada $y(x)=\int_a^x p(x)\,dx$ e fazendo a sua inversão de forma a obter-se x(y).

As funções p(x) e x(y) devem ser passadas como parâmetros do método ***ImportanceSampling*** da classe.

```
class IntegratorMC: public Integrator {
 public:
   IntegratorMC(TF1* f=NULL, int M=1); //integrand function (1-dim by default)
   …
   ~IntegratorMC();
   void UniformRandom(…);
   void ImportanceSampling(TF1* p, TF1* xofy, …);
   void AcceptanceRejection  (…);
 private:
   int Ndim;
   …   //any other useful members e.g. integration limits ? a Formula ?
};
```

Teste os seus métodos na função f(x)=cos(x) calculando o integral $\int_{0.2\pi}^{0.5\pi} \cos(x)\,dx$

Exercício 59 (*adaptado de Barão 2016). Com a classe IntegratorMC, c*alcule os seguintes integrais e respectivos erros

a) $\int_0^1 \frac{dx}{1+x^2}$ (método trapezoidal, Simpson e UniformRandom)

b) $\int_0^1\int_0^1 \frac{x+y}{x^2+y^2}\,dx\,dy$ (UniformRandom)