Introdução à Análise de dados em FAE **(Data: 24/11/2020)**

AULA 6: Monte Carlo

Professores: Sandro Fonseca, Sheila Mara da Silva, Eliza Melo *Name:* Ana Maria Garcia Trzeciak

Os arquivos usados com a descrição completa do código estão no meu github: https://github.com/AnaTrzeciak/Curso-FAE.git

Exercícios referente à aula sobre Monte Carlo

Problema 1:

Escreva um código que estime a área de um disco unitário usando o método Monte Carlo de acerto ou erro. Sabemos que o raio do disco unitário é 1, portanto, o círculo unitário está inscrito em um quadrado de comprimento 2. Dica: gere amostras dentro desse quadrado e conte o número de pontos que caem dentro do disco. Para testar se o ponto está dentro (hit) ou fora (miss) do disco, basta medir a distância da amostra da origem (o centro do disco unitário) e verificar se essa distância é menor (ou igual) do que o raio do disco (que é igual a 1 para um disco unitário).

A área do círculo pode ser calculada pela seguinte relação:

$$\frac{A_{circulo}}{A_{quadrado}} = \frac{\pi}{2 \times 2} = \frac{\pi}{4} \longrightarrow \pi = \frac{4N_{acertos}}{N_{tentativas}} \tag{0.1}$$

Vamos escrever um código que simula eventos em um quadrado de lado 2 e contar quantos eventos satisfazem a relação:

$$x^2 + y^2 \leq 1 \tag{0.2}$$

Foram gerados 10.000 eventos:

```
#define MAXEVENTS 10000

// generating sample
  for( int ievent = 0 ; ievent < MAXEVENTS ; ievent++ ){
    // generate random points
    Double_t x = gRandom->Uniform(-1.0,1.0);
    Double_t y = gRandom->Uniform(-1.0,1.0);
```

Implementando a seleção e preenchendo o histograma:

```
if (x*x + y*y <= 1){
    MC->Fill(x,y);
```

O resultado é dado na figura 1:
A área do círculo pode ser calculada utilizando a relação 0.1. E substituindo $N_{acertos} = 7856$, sendo assim:

$$A_{circulo} = \frac{4 * 7856}{10000} = 3,1424 \tag{0.3}$$

Problema 2:

Calcule a integral $\int_0^3 (1 - x^2)^2 dx$ utilizando o método da rejeição e o método de Monte Carlo.

- **Metodo da Rejeição**

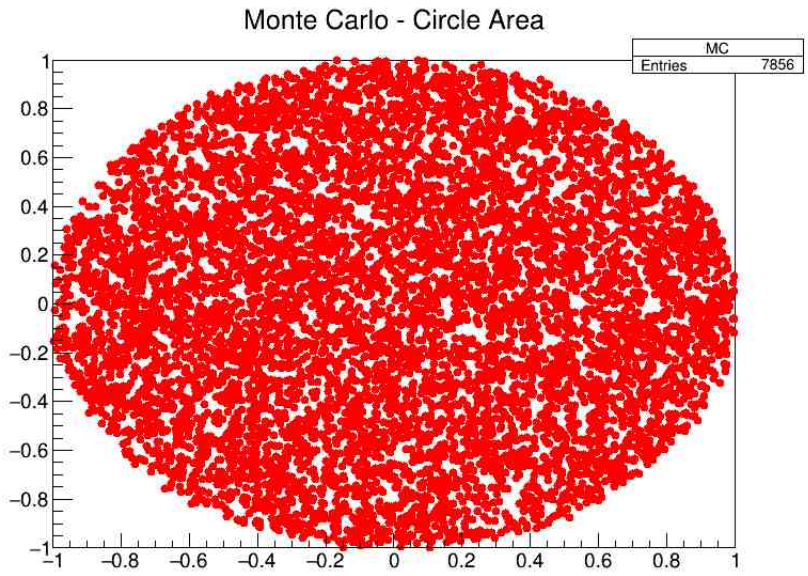

Figura 1: Círculo de raio unitário inscrito em um quadrado de lado 2. Temos 7856 pontos que satisfazem a condição imposta.

Uma integração de uma equação pelo método de rejeiçao consiste em analisar uma amosta de uma distribuição de pontos que probabilisticamente está contido na área abaixo da curva da equação de interesse. A nossa função de interesse é dada por:

$$f(x) = (1 - x^2)^2 \tag{0.4}$$

O que desejamos obter é a integral desta equação nos intervalos $0 \leq x \leq 3$. O método da rejeição nos leva à seguinte relação:

$$I = \int_a^b f(x) = \frac{N_{acertos}}{N_{tentativas}} \times (x_f - x_i) \times (y_f - y_i) \tag{0.5}$$

O que entendemos como número de acertos é a quantidade de pontos que foram sorteados abaixo da curva e as tentativas é o números total de pontos sorteados.

Cada ponto (x, y) sorteado deve estar contido dentro do quadrado definido por: $0 \leq x \leq 3$ e $0 \leq y \leq max(f(x))$. O valor máximo que nossa função pode atingir é $f(3) = 64$.

Para o que chamamos de número de acertos, o ponto sorteado além de satisfazer a condição anterior deve satisfazer também a condição de $y \leq f(x)$.

A nossa área de interesse é dada por:

$$area = (x_f - x_i) \times (y_f - y_i) = (3 - 0) \times (64 - 0) = 192 \tag{0.6}$$

Sendo assim, a equação 0.5 pode ser reescrita como:

$$I = 192 \times \frac{N_{acertos}}{N_{tentativas}} \tag{0.7}$$

Vamos utilizar a equação 0.7 para mostrar o resultado final da nossa simulação e assim calcular o resultado da integral.

Para escrever o código, primeiro começamos definindo as variáveis, as constantes e a função de interesse $f(x)$:

```
#define MAXEVENTS 100000

Double_t Function(Double_t n){
   return (1-n*n)*(1-n*n);
}
.
.
```

```
.
{
//Integration Interval
Int_t x_i = 0;
Int_t x_f = 3;
Int_t y_i = 0;
Double_t area = 192;
Int_t n_acertos = 0;        //Variavel que vai contar quantos eventos satisfazem a
    condicao
Double_t integration = 0; //Variavel que vai receber o resultado final da integracao

  // generating sample
  for( int ievent = 0 ; ievent < MAXEVENTS ; ievent++ ){
    // generate random points
    Double_t x = gRandom->Uniform(x_i,x_f);
    Double_t y = gRandom->Uniform(0,64);
    //Selection Events
    if(y <= Function(x)){
     area_in->Fill(x,y);    //Fill Histogram 2D
     n_acertos++;
    }
  }
}
```

Logo depois a integral é calculada e o resultado é impresso na tela:

```
integration = (n_acertos*area)/MAXEVENTS;

cout << "A area sob a curva e: " << integration << endl;
```

O valor da integral calculado pela simulação é: $I = 33.8419$. O gráfico é mostrado na figura 2.

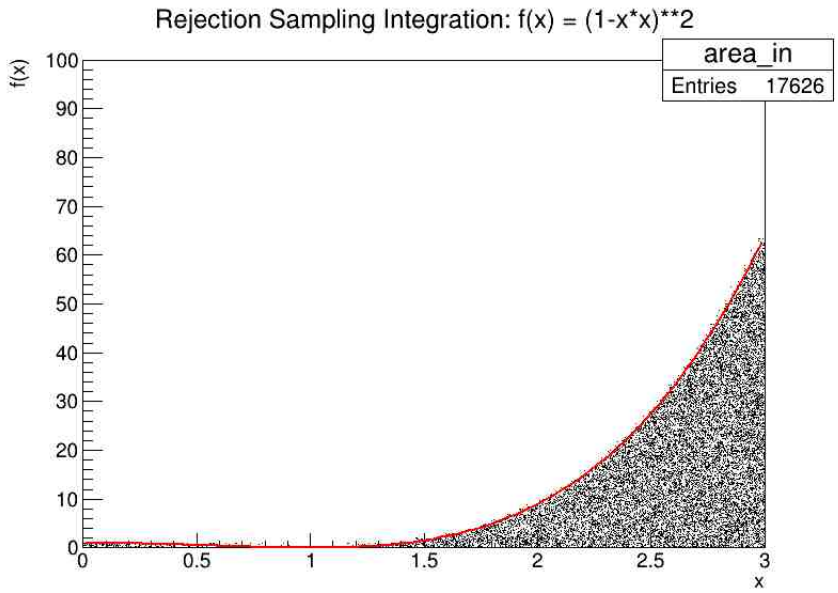

Figura 2: Gráfico da função $f(x) = (1 - x^2)^2$, temos 17626 pontos (x, y) que satisfazem a condição imposta.

Se utilizarmos a equação 0.7, o resultado da integração é:

$$I = 192 \times \frac{N_{acertos}}{N_{tentativas}} = 192 \times \frac{17626}{100000} = 33,84192 \tag{0.8}$$

- **Método de Monte Carlo**

O métode de Monte Carlo para calcular integral consiste em gerar valores aleatórios distribudíos uniformentes num intervalo de $(0, 1)$ a fim de aproximar o resultado de uma integral do tipo:

$$I = \int_a^b f(x)dx \tag{0.9}$$

Sendo assim, a aproximação de Monte Carlo é dada por:

$$I = \frac{(b-a)}{N} \sum_{i=1}^{N} f[x_i(b-a)+a] \tag{0.10}$$

onde x_i são números aleatórios num intervalo (0,1).

Obs: não consegui chegar em um valor próximo ao valor exato dessa integral, que é de $\sim 33,84$. Todas implementações do código davam um resultado muito pequeno. Abaixo o que escrevi:

```
#define MAXEVENTS 100

Double_t Function(Double_t n){
   return (1-n*n)*(1-n*n);
}
...
{
//Integration Interval
Int_t x_i = 0;
Int_t x_f = 3;
Double_t total_function = 0;
Double_t integration = 0;

// generating sample
  for( int ievent = 0 ; ievent < MAXEVENTS ; ievent++ ){
    // generate random points
    Double_t x = gRandom->Uniform(0,1);
    total_function = Function(x*3);
    total_function++;
    }
integration = (3*total_function)/MAXEVENTS;
cout << "A area sob a curva eh: " << integration << endl;
}
```

E os resultados eram sempre da ordem de: 0.180, 0.36... quanto maior o N menor ficava o valor da integral.

Problema 3:

Escreva um código para calcular a seção de choque diferencial do espalhamento Rutherford.

A seção de choque diferencial do espalhamento Rutherford é dada pela equação:

$$\frac{d\sigma}{d\Omega} = \left(\frac{e^2}{8\pi\epsilon_0 m v_0^2}\right)^2 \frac{1}{\sin^4(\theta/2)} \tag{0.11}$$

Onde:

- e é a carga da partícula alfa: $2 \times 1.602 \times 10^{-19}$ C
- m é a massa da partícula alfa: 6.664×10^{-27} kg
- ϵ_0 é a constante de permissividade do vácuo: $8.854 \times 10^{-12} kg^{-1} m^{-3} s^4 A^2$
- v_0 é a velocidade inicial da partícula: 2×10^7 m/s

No código, primeiro definimos todas as constantes e o intervalo do ângulo θ:

```
//Constants
Double_t mass_alpha = 6.644e-27;  //kg
Double_t velocity_alpha = 2e7;    //m/s
Double_t charge_alpha = 2*1.602e-19;   //C
```

```
Double_t permittivity = 8.854e-12; //kg^-1 m^-3 s^4 A^2

//Range theta
Double_t theta_i = 0.1*TMath::Pi();
Double_t theta_f = 0.9*TMath::Pi();
```

Em seguida implementamos a simulação. Foram gerados 3000 eventos, com θ variando no *range* escolhido:

```
Double_t numerical = (pow(charge_alpha*charge_alpha,2))/(pow(8*TMath::Pi()*
    permittivity*mass_alpha*velocity_alpha*velocity_alpha,2)); //Calculo das
    constantes

 cout << numerical << endl;
 // generating sample
  for( int ievent = 0 ; ievent < MAXEVENTS ; ievent++ ){
    // generate random points
  Double_t theta = gRandom->Uniform(theta_i,theta_f);

  //calculo da secao de choque diferencial
  Double_t diff_section = numerical*(1/(pow(TMath::Sin(theta/2),4)));
```

Todos os eventos gerados foram salvos em um arquivo de saída.

```
{...
// open out file
TString file = "aula6_scattering_rutherford-output.txt";
ofstream outfile;
outfile.open(file);
...

outfile << theta << " " << diff_section << endl;
}
```

No terminal do root, plotamos o gráfico de θ versus $d\sigma/d\Omega$ utilizando a classe TGraph. O resultado é dado na figura 3:

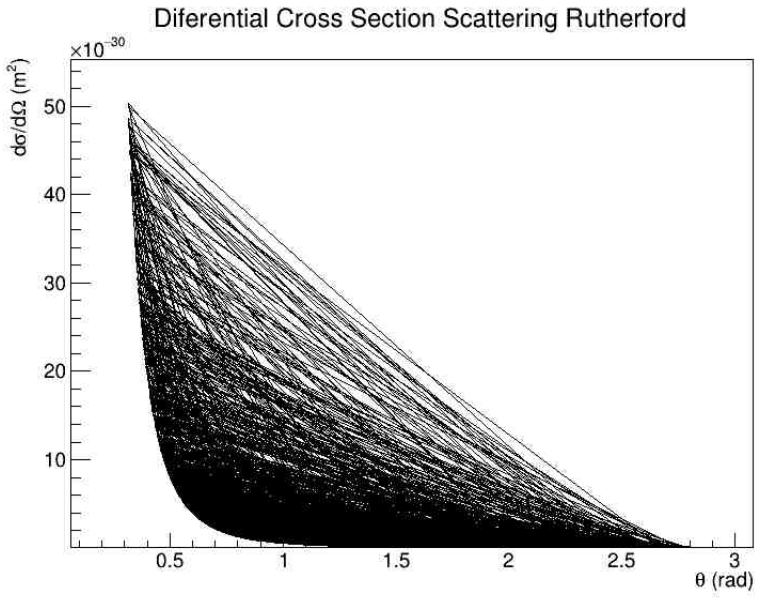

Figura 3: Dependência da seção de choque diferencial pelo ângulo θ espalhado.