13/04/2015

Exercício de Programa 2:

# Método de Monte Carlo

*Instituto de Matemática e Estatística da Universidade de São Paulo*

**Por**

Caio Vinícius Dadauto 7994808

**Professor**

Julio Michael Stern

## 1 Monte Carlo

Seja a função,

$$f(t) = 0.25(2^{1.97994808(1-t)})(1 - \sin(0.88084997\pi t)); \quad [0,1] \tag{1}$$

segue métodos implementados para determinar a integral de $f(t)$.

### 1.1 Número de pontos

Para determinar o número de pontos a serem utilizados nos métodos que se seguem, foram gerados números aleatórios distribuidos uniformemente e assim foi feito o desvio padrão da média, ou seja,

$$\sigma_m = \frac{\sigma}{\sqrt{N}} \tag{2}$$

onde $\sigma = < f(t)^2 > - < f(t) >^2$.

Porém, isso garante apenas que há uma prbabilidade de 68% para que

$$I_{est} \in [I - \sigma_m, I + \sigma_m] \tag{3}$$

onde $I_{est}$ é a integral estimada pelo método de monte carlo implementado e $I$ a integral de $f(t)$ em $[0,1]$.

Assim, buscando um erro de no máximo 0.5%, estimou-se que o número de pontos gerados aleatóriamente deveriam ser em torno de 200000 pontos.

## 1.2 Cru

$$I_{est} = \sum_{i=0}^{N} f(t_i) \tag{4}$$

onde $t_i$ é um número aleatório segundo uma distribuição uniforme e $N$ é o número total de parâmetros aleatórios.

Programa 1: Implementação para o método Cru

```
%Funcao definida no EP
function y = ep(t)
···a = 1.97994808;
···b = 0.88084997;
···for i = 1:length(t)
·····y(i) = 0.25*(2^(a*(1-t(i))))*(1 - sin(b*pi*t(i)));
···endfor
endfunction

%Inicializacao de variaveis
cru          = 0;
t            = 0:0.0001:1;

x  = rand(1, 200000);
y  = rand(1, 200000);
f  = ep(x);
for i = 1:200000
···cru          += f(i);          %Metodo Cru
endfor
cru          *= 0.000005;
```

## 1.3 Hit or Miss

$$I_{est} = \frac{N_0}{N} A \tag{5}$$

onde $N_0$ são os pontos em uma distribuição uniforme tais que se encontram sobre a região abaixo da funcao $f(t)$ (pontos vermelhos em 1) e $A$ é a área limite onde os pontos foram gerados.

Programa 2: Implementação para o método Hit or Miss

```
%Funcao definida no EP
function y = ep(t)
···a = 1.97994808;
···b = 0.88084997;
···for i = 1:length(t)
·····y(i) = 0.25*(2^(a*(1-t(i))))*(1 - sin(b*pi*t(i)));
···endfor
endfunction

%Inicializacao de variaveis
n            = 0;
```

```
t           = 0:0.0001:1;

x  = rand(1, 200000);
y  = rand(1, 200000);
f  = ep(x);
for i = 1:200000
···if(y(i) < f(i))
·········n++;
···endif
endfor
hitmiss      = 0.000005*n;
```

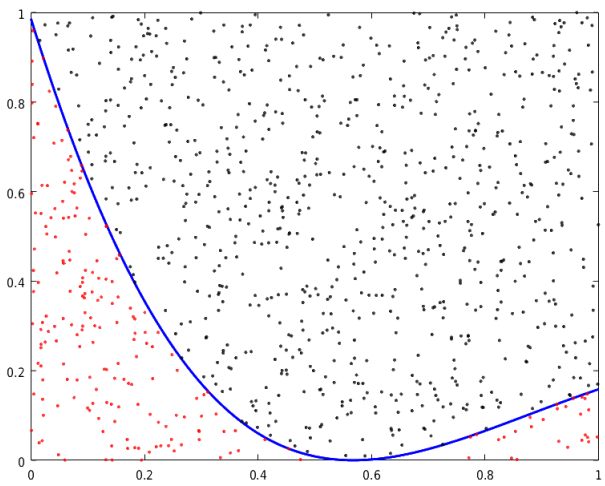



Figura 1: Exemplificação do método Hit or Miss.

## 1.4 Variavel de controle

$$I_{est} = \frac{1}{N}\sum_{i=0}^{N}[f(t_i) - p(t_i] - I_0 \tag{6}$$

onde $p(t)$ é o polinômio de grau dois que melhor aproxima $f(t)$ e $I_0$ é sua integral.

O polinômio foi determinado a partir do método dos mínimos quadrados, pois este se mostrou mais adequado que a expansão por *Taylor*, assim como é apresentado na figura 2.

Programa 3: Implementação para o método com variável de controle

```
%Funcao definida no EP
function y = ep(t)
···a = 1.97994808;
···b = 0.88084997;
···for i = 1:length(t)
```

```
      y(i) = 0.25*(2^(a*(1-t(i))))*(1 - sin(b*pi*t(i)));
   endfor
endfunction

%Inicializacao de variaveis
valcont     = 0;
t           = 0:0.0001:1;

x    = rand(1, 200000);
y    = rand(1, 200000);
f    = ep(x);
p    = polyfit(t, ep(t), 2);
pol  = polyval(p, x);
for i = 1:200000
   valcont     += f(i) - pol(i);
endfor
valcont     += polyint;
```

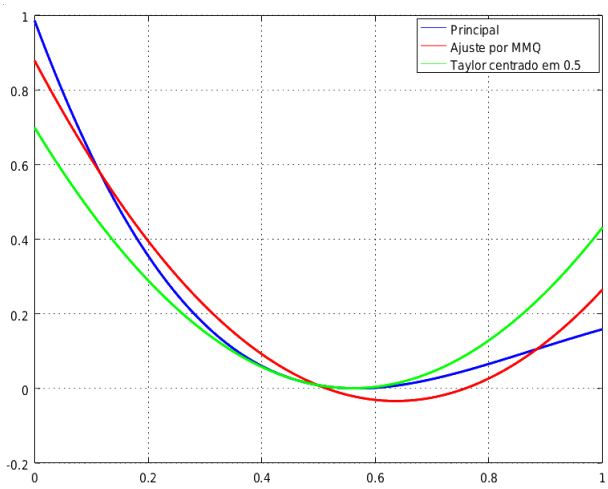



Figura 2: Polinômios ajustados a função $f(t)$.

## 1.5 Importance Sampling

$$I_{est} = \frac{1}{N}\sum_{i=0}^{N}\frac{f(x_i)}{B(x_i)} \tag{7}$$

onde $B$ é a distrinuição beta que melhor se ajusta afunção $f(t)$. Os parâmetros $x_i$ são obtidos segundo a distribuiçaõ beta.

Os parâmetros $\alpha$ e $\beta$ da função beta foram determinados minimizando a disferança da soma dos quadrados entre a função beta e a função $f(t)$, como pode ser observado na implementação a seguir.

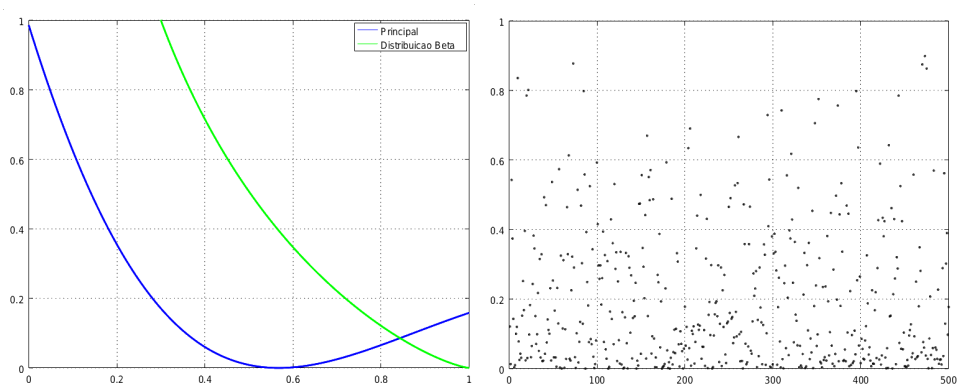
Programa 4: Implementação para o método com variável de controle

```
%Funcao definida no EP
function y = ep(t)
···a = 1.97994808;
···b = 0.88084997;
···for i = 1:length(t)
·····y(i) = 0.25*(2^(a*(1-t(i))))*(1 - sin(b*pi*t(i)));
···endfor
endfunction

%Inicializacao de variaveis
valcont     = 0;
t           = 0:0.0001:1;

x    = rand(1, 200000);
y    = rand(1, 200000);
f    = ep(x);
p    = polyfit(t, ep(t), 2);
pol  = polyval(p, x);
for i = 1:200000
···valcont     += f(i) - pol(i);
endfor
valcont     += polyint;
```

O ajuste e a distribuição dos pontos $x_i$ podem ser observados na seguinte figura:




(a) Função beta ajustada.

(b) Distribuição beta ajsutada.

## 2 Implementação

Segue o prgrama completo que retorna como saida alguns dos gráficos apresentados e a estimatima da $I_{est}$ para todos os métodos abordados seguido de seus respectivos erros.

Programa 5: Implementação final

```
#!/usr/bin/octave -qf

1;
ignore_function_time_stamp("all");
clear all;
%Funcao definida no EP
function y = ep(t)
  a = 1.97994808;
  b = 0.88084997;
  for i = 1:length(t)
    y(i) = 0.25*(2^(a*(1-t(i))))*(1 - sin(b*pi*t(i)));
  endfor
endfunction

%Primeira derivada da funcao ep
function y = d1ep(t)
  a = 1.97994808;
  b = 0.88084997;
  for i = 1:length(t)
    y(i) = exp(-1.3724*t(i))*(1.35345*sin(2.76727*t(i))-2.72908*cos(2.76727*t(i))-1.35345);
  endfor
endfunction

%Segunda derivada da funcao ep
function y = d2ep(t)
  a = 1.97994808;
  b = 0.88084997;
  for i = 1:length(t)
    y(i) = exp(-1.3724*t(i))*(5.69463*sin(2.76727*t(i))+7.49075*cos(2.76727*t(i))+1.85747);
  endfor
endfunction

%Taylor centrado em 0.5
function y = taylor(t)
  a = 1.97994808;
  b = 0.88084997;
  for i = 1:length(t)
    y(i) = ep(0.5) + d1ep(0.5)*(t(i) - 0.5) + d2ep(0.5)*0.5*(t(i) - 0.5)*(t(i) - 0.5);
  endfor
endfunction

%Quadrado da diferenca entre a funcao beta de parametros p(1) e p(2) e a funcao y
function sumSquareErrors = model(p,t,y)
  a=p(1);
  b=p(2);
  B=betapdf(t, a, b);
  for i = 10:length(y)
    difference = y(i) - B(i);
  endfor
  sumSquareErrors = sum(difference.^2);
endfunction

%Inicializacao de variaveis
```

```
54  n           = 0;
55  cru         = 0;
56  valcont     = 0;
57  importance = 0;
58  t           = 0:0.0001:1;
59  inicial     = [0.55, 1.4];
60
61  %Ajusta polinomio e plota os graficos
62  figure;
63  p       = polyfit(t, ep(t), 2);
64  plot(t, ep(t), "linewidth", 2);
65  polyint = p(3) + p(2)*0.5 + p(1)/3;
66  y       = polyval(p, t);
67  hold on;
68  plot(t, y, "color", 'r', "linewidth", 2);
69  plot(t, taylor(t), "color", 'g', "linewidth", 2);
70  legend('Principal', 'Ajuste por MMQ', 'Taylor centrado em 0.5');
71
72  %Ajusta Funcao Beta e plota os graficos
73  figure;
74  hold off;
75  ajustado = fmins('model', inicial, [],[], t, y);
76  y        = betapdf(t, ajustado(1), ajustado(2));
77  plot(t, ep(t), "linewidth", 2);
78  hold on;
79  plot(t, y, "color", 'g', "linewidth", 2);
80  legend('Principal', 'Distribuicao Beta');
81  ylim([0, 1]);
82  a = ajustado(1);
83  b = ajustado(2);
84
85  x  = rand(1, 200000);
86  y  = rand(1, 200000);
87  f  = ep(x);
88  w  = betaincinv(x, a, b);
89  B  = betapdf(w, a, b);
90  fb = ep(w);
91  pol = polyval(p, x);
92  for i = 1:200000
93  ···cru        += f(i);              %Metodo Cru
94  ···valcont    += f(i) - pol(i);     %Metodo com variavel de controle
95  ···importance += fb(i)/B(i);·   %Metodo Importance
96  ···if(y(i) < f(i))                  %Metodo Hit or Miss
97  ·····n++;
98  ···endif
99  endfor
100 cru        *= 0.000005;
101 valcont    *= 0.000005;
102 valcont    += polyint;
103 hitmiss     = 0.000005*n;
104 importance *= 0.000005;
105 printf('Metodo cru:\n\tValor -> %f\n\tErro -> %f%%\n\n',
106 ···cru, 100*abs(cru - 0.196064)/0.196064);
107 printf('Metodo hit or miss:\n\tValor -> %f\n\tErro -> %f%%\n\n',
```

```
108 ···hitmiss, 100*abs(hitmiss - 0.196064)/0.196064);
109  printf('Metodo por variavel de controle:\n\tValor -> %f\n\tErro -> %f%%\n\n',
110 ···valcont, 100*abs(valcont - 0.196064)/0.196064);
111  printf('Metodo por importance sample:\n\tValor -> %f\n\tErro -> %f%%\n\n',
112 ···importance, 100*abs(importance - 0.196064)/0.196064);
```