EP03 - Geradores Quasi-aleatórios
MAP2212 - Laboratório de Comutação e Simulação

Danilo Brito da Silva - Nº USP: 10693250
William Veloso - Nº USP: 10801513

24 de Maio de 2021

1 Introdução

Este exercício consiste em substituir o gerador de números pseudo-aleatórios por um gerador de números quasi-aleatórios para estimar a integral da função $f(x)$, $x \in [0.1]$ utilizando quatro variantes do Método de Monte Carlo (MMC): *Crud*, *Hit or Miss*, *Importance Sampling* e *Control Variate*.

$$f(x) = e^{-ax} \cos bx \tag{1}$$

Utilizaremos $a = 0.442705074$ e $b = 0.10693250$. Para cada método, o erro relativo deverá ser $\mid \hat{g} - g \mid /g \leq 0.0005$, onde $\hat{g}$ é a estimativa numérica do valor da integral, e g é o verdadeiro valor da integral (desconhecido).

Com um gerador de números quasi-aleatórios, verificaremos o que acontece quando utilizamos variáveis mais igualmente distribuídas (mais "espalhadas") que as geradas por geradores pseudo aleatórios ao calcular integrais analiticamente complexas através do Método de Monte Carlo.

Para cada um dos métodos, vamos gerar n variáveis quasi-aleatórias e armazená-las como vetor, calcular variância e erro padrão. Caso o erro seja maior que 0.0005, aumentamos o n em 2 vezes e repetimos o processo. O valor da função $g = \int_0^1 f(x)dx$ será estimado como a média dos valores gerados em cada método.

A implementção das variáveis quasi-aleatórias foi feita atravése da função *halton* de dimensão 1 do R.

2 Método Crud

O valor da integral é calculado como sendo a média dos valores da $f(x)$ em n pontos aleatórios com x_i pertencente ao intervalo $[0, 1]$ e distribuição uniforme, $i = 1, ..., n$. Portanto, temos:

$$\hat{g} = \frac{1}{n} \sum_{1}^{n} f(x_i)$$

Utilizando o software R, rodamos o seguinte código:

```
#----------1.Crud----------
n <- 5    #amostra incial
erro_crud <- 1     #erro inicial
while(erro_crud > 0.0005)
{                                  # n serÃ¡ incrementado atÃ© que erro_crud < 0.0005
  vetorx <- vector("numeric", n)     # criaÃ§Ã£o de vetor de tamanho n
  x <- halton(n,1) #gerando sequencia de variÃ¡vel quasi-aleatÃ³ria em [0,1]
  for (i in 0:n){
    vetorx[i] <- f(x[i]) #vetor recebe f(x)
  }
  erro_crud <- sqrt(var(vetorx)/n) #cÃ¡lculo do erro
  if (erro_crud > 0.0005)
  {
    n <- n*2
  }
}
estimativa_g_crud <- mean(vetorx)
print(n)
print(estimativa_g_crud)
print(erro_crud)
```

Dessa forma, obtivemos os seguintes resutados:
$\hat{g} = 0.8066269; n = 81920; \epsilon = 0.0003640106$

3 Método Hit or Miss

Neste método, geramos pares de pontos (x_i, y_i) e escolhemos uma função auxiliar $h(x, y)$ de distribuição Bernoulli tal que

$$h(x,y) = \begin{cases} 1 & \text{se } f(x) \geq y, \\ 0 & \text{se } f(x) < y. \end{cases}$$

Assim, vamos estimar $g = \int_0^1 \int_0^1 h(x,y)dxdy$ como $\hat{g} = \frac{1}{n}\sum_1^n h(x_i, y_i) = \frac{n*}{n}$ e a estimativa da variância será dada por $s^2 = \dfrac{\hat{g}(1-\hat{g})}{n}$.

Executamos o código abaixo no R e obtivemos os seguintes resultados:

```
#----------2.Hit or Miss----------
n <- 5  #amostra incial
erro_hit  <- 1 #erro inicial
while(erro_hit > 0.01){
  vetor_x <- halton(n, 1, init = FALSE)
  vetor_y <- halton(n, 1, init = FALSE)
  vetor_n <- vector("numeric", n)
  vetor_f <- vector("numeric", n)

  for (i in 1:n){
    vetor_f[i] <- f(vetor_x[i])
  }
  for (i in 1:n){ #armazenando valores de n*
    if (vetor_f[i] >= vetor_y[i]){
      vetor_n[i] = 1
    }
    else{vetor_n[i] = 0}
  }
  erro_hit = sqrt(var(vetor_n)/n) #cÃ¡lculo do erro
  if (erro_hit > 0.0005){
    n <- n*2
  }
}
estimativa_g_hit = mean(vetor_n)

print(n)
print(estimativa_g_hit)
print(erro_hit)
```

$\hat{g} = 0.6972237; n = 1310720; \epsilon = 0.0004013213$

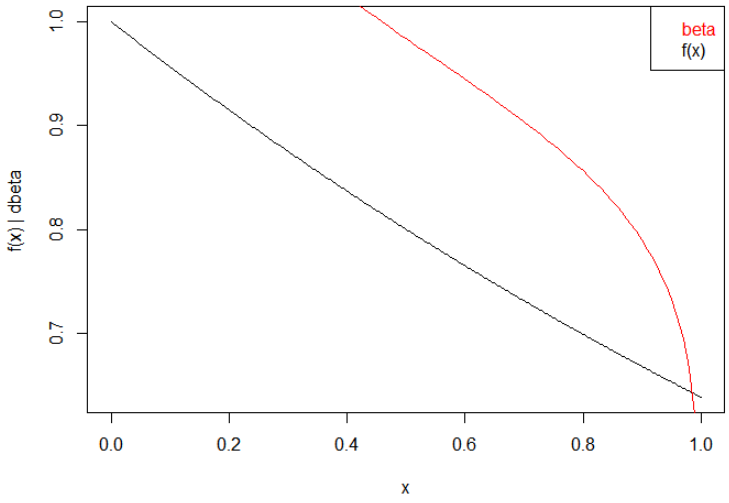
4 Método Importance Sampling

Neste método, escolhemos uma função auxiliar $k(x)$ que seja função de distribuição de probabilidade conhecida e se aproxime da $f(x)$.

$$g = \int f(x)dx = \int \frac{f(x)}{k(x)}k(x)dx = \int \frac{f(x)}{k(x)}dK(x)$$

$$\hat{g} = \frac{1}{n}\sum_{1}^{n}\frac{f(x_i)}{k(x_i)}, x_i \sim k, i = 1:n, s^2 = \frac{1}{n}\int\left(\frac{f(x)}{k(x)} - \hat{g}\right)^2 dK(x)$$

Vamos escolher a função de distribuição Beta(α, β), integrável, razoavelmente proporcional a $f(x)$ e cuja razão $f/Beta$ seja limitada no intervalo $[0, 1]$, com $\alpha = 0.9$ e $\beta = 1.1$, ilustrada no gráfico abaixo, e armazenar os resultados de $f(x_i)/k(x_i)$ em um vetor e calcular sua média para chegar na aproximação $\hat{g}$.




```
#----------3.Importance Sampling----------
n <- 5  #amostra incial
erro_is <- 1 #erro inicial

#v_x <- vector("numeric", n) #vetor de n variÁ¡veis xi uniformes no intervalo [0,1]
v_weight <- vector("numeric",n) #vetor com peso de importancia (importance weight) para cada valor amostrado

while(erro_is > 0.0005){
  v_x <- halton(n,1) #armazena n variÁ¡veis pseudo aletatÁ³rias
  for (i in 0:n){
    v_weight[i] <- (f(v_x[i]))/dbeta(v_x[i],0.9,1.1)
  }

  erro_is <- sqrt(var(v_weight)/n) # calcula o erro do mÃ©todo
  if (erro_is > 0.0005){
    n <- n^2
  }
}

estimativa_g_is <- mean(v_weight)

print(n)
print(estimativa_g_is)
print(erro_is)

# Gera um grÁ¡fico comparando a funÃ§Ã£o densidade da beta (x, 0.9,1.1) e a f(x)
x<- seq(0,1,length.out = 400)
plot(x,f(x),'l', xlab = 'x', ylab = 'f(x) | dbeta ')
curve(dbeta(x,0.9,1.1), add = TRUE, col = 'red')
legend("topright", legend=c("beta", "f(x)"),text.col=c("red", "black"),col =c("red", "black"))
```

Executando os comandos acima no R, obtivemos os seguintes resultados: $\hat{g} = 0.7881357; n = 10240; \epsilon = 0.0004244787$

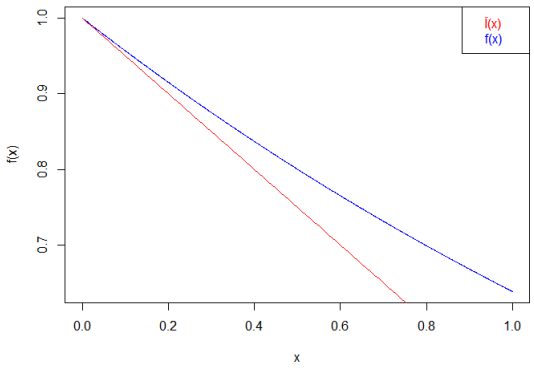
5 Método Control Variate

No método Control Variate, escolhemos uma função $\varphi(x)$ de comportamento parecido com a $f(x)$, porém mais fácil de integrar.

$$g = \int \varphi(x)dx + \int (f(x) - \varphi(x))dx = g' + \int (f(x) - \varphi(x))dx$$

Com os estimadores $\hat{g} = \frac{1}{n}\sum_1^n f(x_i)$ e $\hat{g'} = \frac{1}{n}\sum_1^n f(x_i)$ e a variância dada por $Var(\hat{g} - \hat{g'}) = Var(\hat{g}) + Var(\hat{g'}) - 2Cov(\hat{g}, \hat{g'})$, o método é útil se as funções $f(x)$ e $\varphi(x)$ possuem forte correlação positiva.

Assim sendo, escolhemos a $\varphi(x) = 1 - 0.5x$, conforme imagem a seguir:



No R, armazenamos os valores de $f(x)$ e $\varphi(x)$ e suas diferenças em vetores. Assim, estimamos a integral de $f(x)$ como sendo a média de $\varphi(x)$ mais a média da diferença de $f(x)$ e $\varphi(x)$, conforme abaixo:

```
#----------4.Control Variate----------
n <- 5
erro_cv <- 1

#Criando funÃ§Ã£o phi:
phi = function(x){
  return (1 - (0.5*x))
}

# MÃ©todo iterativo:
while (erro_cv > 0.0005)
{
  v_x   <- halton(n,1) #vetor de n variÃ¡veis quasi-aleatÃ³rias
  v_phi <- vector("numeric",n) #vetor com valores da phi (x1)
  v_f   <- vector("numeric",n) #vetorcom valores da f(x1)

  for (i in 0:n)
  {
    v_phi[i] <- phi(v_x[i]) #vetor que armazena os valores de phi(x1)
    v_f[i]   <- f(v_x[i])   #vetor que armazena os valores da f(x1)
  }
  var <- var(v_phi) + var(v_f) - 2*var(v_phi,v_f) # variÃ¢ncia da diferenÃ§a de f e phi
  erro_cv <- sqrt(var/n)                          #cÃ¡lculo do erro
  if (erro_cv > 0.0005)
  {
    n <- n^2
  }
}
v_dif <- f(v_x) - phi(v_x)
estimativa_g_cv <- mean(v_phi) + mean(v_dif)

print(n)
print(estimativa_g_cv)
print(erro_cv)
```

Executando os comandos acima no R, obtivemos os seguintes resultados:
$\hat{g} = 0.7855624; n = 1280; \epsilon = 0.0004053953$

6 Conclusões

A implementação deste exercício foi muito parecida com o EP anterior, a diferença consistiu na ultilização do gerador quasi-aleatório *halton* e, como era esperada uma convergência mais rápida, iniciamos com um n menor (n=5) para erro padrão $\leq$0.005.

Diferentemente do esperado na teoria, os métodos *Crud* e *Hit or Miss* demoraram mais para convergir em relação ao gerador pseudo-aleatório, porém apresentou pequena melhora na precisão da estimativa, que é proporcional a $\frac{1}{\sqrt{n}}$.

Os métodos *Importance Sampling* e *Control Variate* tiveram melhora na velocidade de convergência, piorando um pouco na precisão da estimativa, pois o erro padrão é inversamente proporcional a $\sqrt{n}$.

Destacamos abaixo uma comparação dos desempenhos dos métodos:

Método	Estimativa	n	Erro
Crud pseudo	0.8064915	64000	0.0004124224
Crud quasi	0.8066269	81920	0.0003640106
Hit or Miss pseudo	0.8062969	1024000	0.0003905405
Hit or Miss quasi	0.6972237	1310720	0.0004013213
Importance Sampling pseudo	0.8160307	16000	0.0004419491
Importance Sampling quasi	0.7881357	10240	0.0004244787
Control Variate pseudo	0.8160307	8000	0.0004566607
Control Variate quasi	0.7855624	1280	0.0004053953