CPE723 Otimização Natural:
Lista de Exercícios #1

Data de entrega: Terça, 19 de Março, 2019

Profs. José Gabriel Rodríguez Carneiro Gomes, Terças e Quintas: 08:00-10:00

Vinicius Mesquita de Pinho

Questão 1

Calcular $\int_0^1 xe^{-x}dx$ de três formas diferentes.

a) Primeiro calculando a integral indefinida.

$$\int xe^{-x}dx = -e^{-x}x - e^{-x} + k, \tag{1}$$

onde k é uma constante, que daqui para frente assumiremos $k = 0$. Aplicando integral por parte com $u = x$ e $v' = e^{-x}$, teremos

$$-e^{-x}x - e^{-x} + k = -e^{-x} - \int -e^{-x}dx. \tag{2}$$

Como $\int -e^{-x}dx = e^{-x}$, teremos

$$-e^{-x} - \int -e^{-x}dx = -e^{-x}x - e^{-x}. \tag{3}$$

Calculando os limites:

$$\int_0^1 xe^{-x}dx = -\frac{2}{e} - (-1) = -\frac{2}{e} + 1 \approx 0.26424\ldots \tag{4}$$

b) Pelo método de Monte Carlo, usando 10 números escolhidos aleatoriamente com densidade uniforme entre 0 e 1.

Os números gerados: $\mathcal{I} = (0.29097317, 0.71138164, 0.01536752, 0.12761764, 0.64961303, 0.187312, 0.61146622, 0.22022406, 0.71521083, 0.10874773)$
O calculo:

$$\int_0^1 xe^{-x}d \approxeq \frac{\sum_{i\in\mathcal{I}} ie^{-i}}{|\mathcal{I}|} \tag{5}$$

onde $|\mathcal{I}|$ é a cardinalidade do conjunto $\mathcal{I}$.
Para o conjunto $\mathcal{I}$ o resultado foi 0.21446048.

c) Pelo método de Monte Carlo, usando 10 números escolhidos aleatoriamente com densidade exponencial (note que as amostras geradas a partir da p.d.f. exponencial devem ser limitadas ao intervalo de 0 a 1).

Os números gerados: $\mathcal{W} = (0.08543949, 2.25812132, 0.3702004, 0.38308359, 1.67093227, 1.31658109, 0.7362191, 0.51912815, 0.09628735, 5.96343813)$
Utilizando a equação (5) para o conjunto $\mathcal{W}$, o resultado encontrado foi 0.2262788.
Discussão: apesar do resultado encontrado para os números gerados neste exemplo mostrarem que a distribuição exponencial chegar mais próximo ao valor da integral, quando aumentamos o número de amostras geradas, a distribuição normal alcança um resultado mais próximo do calculado no primeiro item do exercício. Vendo o gráfico da função xe^{-x}, pode-se visualmente pensar que isso ocorre pois no intervalo a curva tem uma tendência mais normal do que exponencial.

Questão 2

Usando $N = 20$ números aleatórios, escolhidos a partir de uma p.d.f. uniforme entre -1 e $+1$, calcular uma aproximação para o número π pelo método de Monte Carlo. Faça o mesmo no computador, utilizando um valor alto para N (por exemplo, 1.000.000). Comente o resultado.
A fórmula para π utilizada é,

$$\pi = \int_{-1}^{1} \frac{dx}{\sqrt{1-x^2}} \tag{6}$$

Os números gerados: $\mathcal{Q} = (0.69754726, 0.51144544, -0.78323647, -0.66956193, 0.90813157, 0.25154249,$ $0.09702152 - 0.88948143, -0.74180524, 0.65899295, 0.05104066, 0.07932701, -0.55706355, -0.87772446$ $0.89741046, 0.71267101, 0.98467096, -0.63258529, -0.51673907, 0.64780901)$
O calculo:

$$\int_{-1}^{1} \frac{dx}{\sqrt{1-x^2}} \approxeq \frac{\sum_{q\in\mathcal{Q}} \left(\sqrt{1-q^2}\right)^{-1}}{|\mathcal{Q}|} \tag{7}$$

onde $|\mathcal{Q}|$ é a cardinalidade do conjunto $\mathcal{Q}$.
Utilizando o conjunto $\mathcal{Q}$, o resultado obtido foi 1.672111. Para um valor de N alto, foi utilizado $N = 10^6$, onde o valor obtido foi $1,57$

Questão 3

Escrever um algoritmo para gerar números $x(n)$ com energia $J(x) = x^2$, de forma que as probabilidades dos números gerados ejam proporcionais aos fatores de Boltzmann $e^{-J(x)/T}$, com temperatura $T = 0.1$. Começando de um valor $x(0)$ qualquer, aplique sempre perturbações ϵR ao valor $x(n)$ atual. Neste caso, R é uma vriável aleatória uniforme. Considere $\epsilon = 0.1$.

a) Execute o algoritmo proposto no computador, calculando $x(n)$ até $n = 100.000$.

O algoritmo.

```
# Definindo:
x_validos = np.array(0)  # definindo o primeiro ponto, x_0
epsilon = 0.1
T = 0.1
n = 100000

while np.size(x_validos) != n:
    R = np.random.uniform(0,1)
    x_candidato = x_n + epsilon*R
    delta_J = (x_candidato)**2 - x_n**2
    q = exp(-delta_J/T)
    r = np.random.uniform(0,1)
    if r > q:
        a = 0
    else:
        a = 1
    x_proximo = (1-a)*x_n + a*x_candidato
    if x_proximo != x_n:
        x_validos = np.append(x_validos,x_proximo)
```

b) Execute manualmente os 10 primeiros passos do algoritmo.