CPE-723 – Otimização Natural
Lista de Exercícios #2

Amanda Isabela de Campos (DRE 120074842)

1. **Considere um processo de Markov $X(t)$ que tem três estados possíveis: 0, 1, e 2. A evolução temporal deste processo é dada pela matriz de transição a seguir:**

$$M = \begin{bmatrix} 0.50 & 0.25 & 0.25 \\ 0.25 & 0.50 & 0.25 \\ 0.25 & 0.25 & 0.50 \end{bmatrix}$$

   a) **Considerando que a distribuição de probabilidade de $X(0)$ é dada pelo vetor $p_0 = [0.3\ 0.4\ 0.3]^T$, calcule a distribuição de probabilidade de $X(3)$ (ou seja, do processo de Markov no instante $t = 3$).**

$$p_1 = Mp_0$$
$$p_2 = M^2p_0$$
$$p_3 = M^3p_0$$

```
import numpy as np
M = np.array([[0.5,0.25,0.25],[0.25,0.50,0.25], [0.25, 0.25, 0.50]])
p0 = np.array([[0.3],[0.4], [0.3]])
M2 = M.dot(M)
M3 = M2.dot(M)

print(M3.dot(p0))
```

[[0.3328125], [0.334375], [0.3328125]]

   b) **Iniciando em X(0) = 1, e usando um gerador de números aleatórios (são necessários apenas três números aleatórios, sorteados de PDF uniforme entre 0 e 1), calcule manualmente uma amostra do processo $X(t)$ até $t = 3$.**

```
a = np.random.uniform(0.00,0.25,1)
b = np.random.uniform(0.25,0.75,1)
c = np.random.uniform(0.75,1.00,1)
p0 = np.array([a,b,c])
M = np.array([[0.5,0.25,0.25],[0.25,0.50,0.25], [0.25, 0.25, 0.50]])
M2 = M.dot(M)
M3 = M2.dot(M)
print(M3.dot(p0))
```

[[0.5860114], [0.59551058], [0.59795439]]

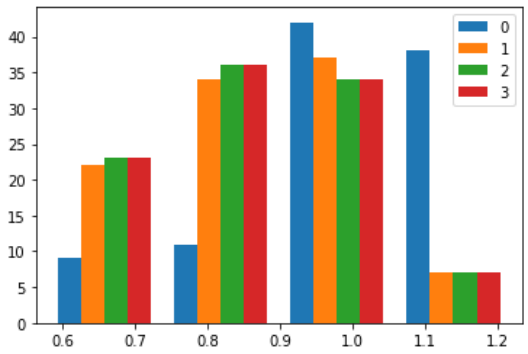
c) **Usando um computador, execute 100 repetições do item (b). Em cada uma das 100 repetições, comece a simulação com um valor diferente de $X(0)$, assumindo que os eventos $X(0) = 0$, $X(0) = 1$, e $X(0) = 2$ são equiprovaveis. Armazene as 100 cadeias obtidas em uma matriz X, com 4 colunas ($t = 0$ até $t = 3$) e 100 linhas.**

```
M = np.array([[0.5,0.25,0.25],[0.25,0.50,0.25], [0.25, 0.25, 0.50]])
M2 = M.dot(M)
M3 = M2.dot(M)
C = np.zeros((100,4))
for i in range(100):
    prob = np.random.choice([0,1,2])

    if prob == 0:
        a = np.random.uniform(0.00,0.50,1)
        b = np.random.uniform(0.50,0.75,1)
        c = np.random.uniform(0.75,1.00,1)
        p0 = np.array([a,b,c])

    if prob == 1:
        a = np.random.uniform(0.00,0.25,1)
        b = np.random.uniform(0.25,0.75,1)
        c = np.random.uniform(0.75,1.00,1)
        p0 = np.array([a,b,c])

    if prob == 2:
        a = np.random.uniform(0.00,0.25,1)
        b = np.random.uniform(0.25,0.50,1)
        c = np.random.uniform(0.50,1.00,1)
        p0 = np.array([a,b,c])

    C[i,0] = np.linalg.norm(p0)
    C[i,1] = np.linalg.norm(M.dot(p0))
    C[i,2] = np.linalg.norm(M2.dot(p0))
    C[i,3] = np.linalg.norm(M3.dot(p0))
```

d) **Fazendo histogramas de cada uma das 4 colunas, calcule as distribuições de probabilidade do processo $X(t)$ em cada um dos 4 instantes: t = 0, 1, 2, 3. Comente os resultados obtidos.**

```
import matplotlib.pyplot as plt
plt.figure(1)
plt.hist(C, 4)
plt.legend('0123')
plt.show()
```

Observa-se que em todos instantes de tempo o histograma indica que a maior probabilidade está perto do estado 1. E com a diminuição da temperatura esse comportamento é acentuado.



2. **Considere um sistema em que só há 5 estados possíveis: $x = 1$, $x = 2$, $x = 3$, $x = 4$, $x = 5$. Os custos $J(x)$ de cada um dos estados são indicados na tabela abaixo:**

x	$J(x)$
1	**0.5**
2	**0.2**
3	**0.3**
4	**0.1**
5	**0.4**

a) **Considere um processo de Markov gerado pela aplicação do algoritmo de Metropolis aos dados da tabela acima, com temperatura fixa $T = 0.1$. Calcule a matriz de transição M que define o processo $X(t)$. Obs.: note que o estado $X(t)$ é unidimensional, e portanto a matriz M é 5×5.**

A montagem da matriz de transição M segue a seguinte regra: ΔJ < 0, o estado é aceito com uma probabilidade de ¼, caso contrário, a probabilidade de ser aceito é calculada.

M_{12} = (prob. de sortear 1 a partir de 2) x (prob. de aceitar 1 a partir de 2) = $\frac{1}{4}e^{-\Delta J/T}$

$$M_{12} = \frac{1}{4}e^{-(0.5-0.2)/0.1} = \frac{1}{4}\,e^{-3}$$

$$M_{32} = \frac{1}{4}e^{-(0.3-0.2)/0.1} = \frac{1}{4}e^{-1}$$

$$M_{52} = \frac{1}{4}e^{-(0.4-0.2)/0.1} = \frac{1}{4}e^{-2}$$

$$M_{22} = 1 - \frac{1}{4}e^{-1} + \; 1 - \frac{1}{4}e^{-2}$$

$$M = \begin{bmatrix} 0 & 0.0124468 & 0.0338338 & 0.00457891 & 0.0919699 \\ 0.25 & 0.61175 & 0.25 & 0.0919699 & 0.25 \\ 0.25 & 0.0919699 & 0.374196 & 0.0338338 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.857171 & 0.25 \\ 0.25 & 0.0338338 & 0.0919699 & 0.0124468 & 0.15803 \end{bmatrix}$$

```
import numpy as np
```

```
M = np.zeros([5,5])
M[0,0] = 0
M[1,0] = 1/4
M[2,0] = 1/4
M[3,0] = 1/4
M[4,0] = 1/4
M[0,1] = 1/4*np.exp(-3)
M[1,1] = 1/4*(1-np.exp(-3)) + 1/4*(1-np.exp(-1)) + 1/4*(1-np.exp(-2))
M[2,1] = 1/4*np.exp(-1)
M[3,1] = 1/4
M[4,1] = 1/4*np.exp(-2)
M[0,2] = 1/4*np.exp(-2)
M[1,2] = 1/4
M[2,2] = 1/4*(1-np.exp(-2)) + 1/4*(1-np.exp(-1))
M[3,2] = 1/4
M[4,2] = 1/4*np.exp(-1)
M[0,3] = 1/4*np.exp(-4)
M[1,3] = 1/4*np.exp(-1)
M[2,3] = 1/4*np.exp(-2)
M[3,3] = 1/4*(1-np.exp(-4)) + 1/4*(1-np.exp(-1)) + 1/4*(1-np.exp(-
2)) + 1/4*(1-np.exp(-3))
M[4,3] = 1/4*np.exp(-3)
M[0,4] = 1/4*np.exp(-1)
M[1,4] = 1/4
M[2,4] = 1/4
M[3,4] = 1/4
M[4,4] = 1/4*(1-np.exp(-1))
```

b) Iniciando em $X(0) = 1$, calcule manualmente 4 amostras do processo $X(t)$.

$$p_1 = Mp_0$$
$$p_2 = M^2p_0$$
$$p_3 = M^3p_0$$
$$p_4 = M^4p_0$$

```
p0 = np.array([[np.exp(0.5)],[np.exp(0.2)], [np.exp(0.3)], [np.exp(0.1)], [np.
exp(0.4)]])

M2 = M.dot(M)
M3 = M2.dot(M)
M4 = M3.dot(M)

print(M4.dot(p0))
```

p0 = [1.64872, 1.2214, 1.34986, 1.10517, 1.49182]

p4 = [[0.09749708] [1.83688161] [0.71882263] [3.89896649][0.26481064]]

c) Qual é o vetor invariante da matriz *M* do item (a) ?
Obs.: para facilitar os cálculos, pode-se usar o computador neste item.

```
autovals, autovecs = np.linalg.eig(M)
print ("Autovetores de A: \n", autovecs[:,0])
print ("Autovalores de A: \n",autovals)

inv = np.zeros(5)
z = np.sum(autovecs[:,0])
for i in range(0,5):
    print(i)
    inv[i] =  autovecs[i,0] / z
print('Vetor inveriante da matriz M: ' + str(inv) )
```

Vetor inveriante da matriz M: [0.01165623 0.23412166 0.08612854 0.63640865 0.03168492]

d) Calcule os fatores de Boltzmann (ou seja, $e^{-(J(x))/T}$) associados aos dados da tabela acima, e compare-os com o resultado do item (c). Use T = 0.1.

x	$J(x)$	$e^{-(J(x))/T}$	$e^{-(J(x))/T} / \sum e^{-(J(x))/T}$
1	0.5	0.00673795	0.011656231
2	0.2	0.13533528	0.234121657
3	0.3	0.04978707	0.086128544
4	0.1	0.367879441	0.636408646
5	0.4	0.018315639	0.031684921

Observa-se que os valores da ultima coluna da tabela anterior correspondem aos mesmo valores da vetor invariante da matriz M calculados no item c.

e) *Simulated Annealing:* Usando um computador, execute 1000 iterações do algoritmo de Metropolis em cada uma das 10 temperaturas a seguir. Na passagem de uma temperatura para a outra, use o estado atual. Comente as distribuições de probabilidade obtidas no final de cada temperatura.

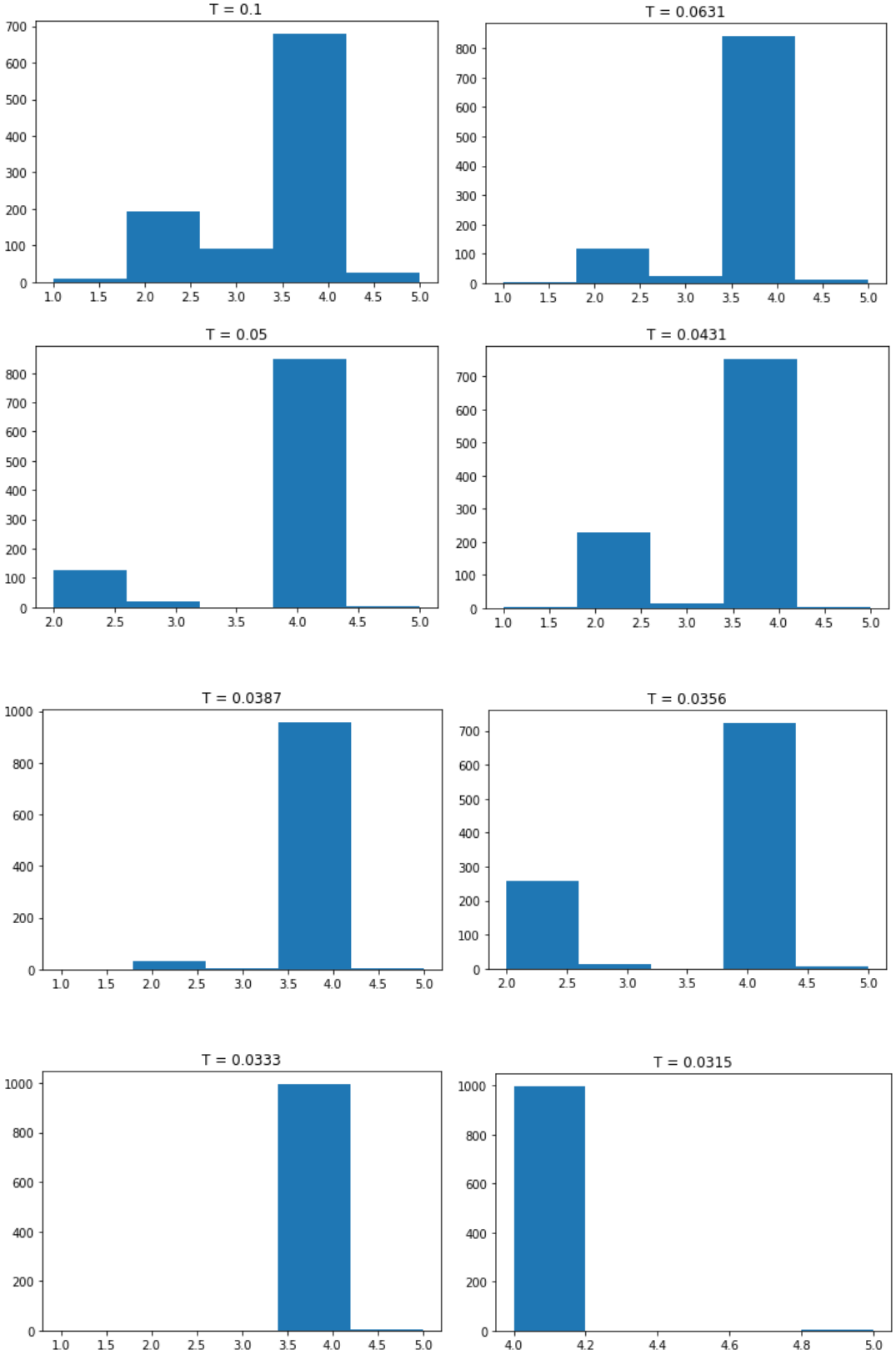
T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
0.1000	**0.0631**	**0.0500**	**0.0431**	**0.0387**	**0.0356**	**0.0333**	**0.0315**	**0.0301**	**0.0289**

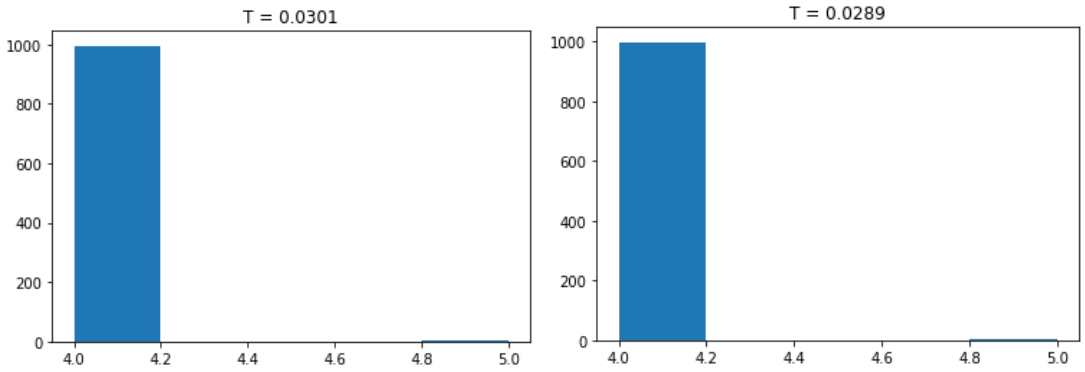
```
import numpy as np
import matplotlib.pyplot as plt
##
def J(x):
    if x == 1:
        return 0.5
    if x == 2:
        return 0.2
    if x == 3:
        return 0.3
```

```
    if x == 4:
        return 0.1
    if x == 5:
        return 0.4

N = 1000
M = 0.1*N
Temp = np.zeros(10)
Temp[0] = 0.1; Temp[1] = 0.0631; Temp[2] = 0.05; Temp[3] = 0.0431;Temp[4] = 0.
0387
Temp[5] = 0.0356; Temp[6] = 0.0333; Temp[7] = 0.0315; Temp[8] = 0.0301; Temp[9
] = 0.0289
x = np.zeros(N)
n = 0
x[n] = np.random.choice([1,2,3,4,5])

i = 0; fim = 0
k = 1; K=11

while not(fim):
    T = Temp[i]
    i += 1
    for n in range(1,N):
        perturbacao = np.random.choice([-1,1])
        x_hat = x[n-1] + perturbacao
        if x_hat == 6:
            x_hat = 1
        elif x_hat == 0:
            x_hat= 5

        if np.random.uniform(0,1)< np.exp((J(x[n-1]) - J(x_hat))/T):
            x[n] = x_hat
        else:
            x[n] = x[n-1]

    y = np.linspace(0,6,1000)
    plt.figure(1)
    plt.hist(x, 5)
    plt.title('T = ' + str(T))
    plt.show()

    k+=1
    if k==K: fim=1
```

Nos histogramas a seguir estão representadas as distribuições de probabilidade em cada temperatura, e com a diminuição da temperatura observa-se que o algoritmo converge para o ponto de menor custo (em x=4) ou seja, esse procedimento de diminuir a temperatura gradativamente em um algoritmo de Metropolis, consiste em criar um método de otimização por Simulated

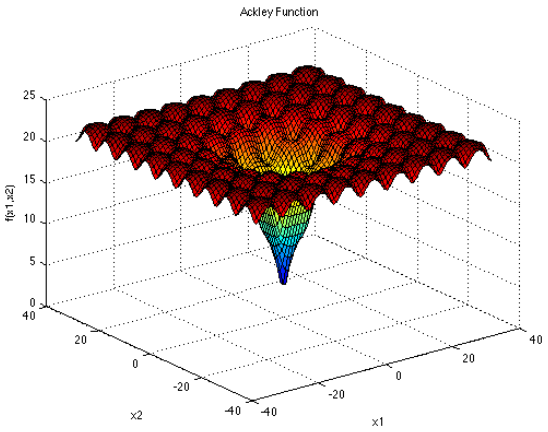
Annealing.





3. **Proponha uma função J(x), sendo x um vetor com 10 dimensões, cujo ponto mínimo você conheça. Evite propor funções que tenham um só ponto mínimo. Encontre o ponto mínimo global utilizando S.A. Obs.: neste exercício, entregue o código utilizado e alguns comentários sobre o resultado obtido.**

A função escolhida para esse problema foi a ACKLEY FUNCTION (Referência: https://www.sfu.ca/~ssurjano/ackley.html) que segue a expressão a seguir e nesse caso, será adotado d = 10 para representar uma função de 10 dimensões.



$$f(\mathbf{x}) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

Com valores recomendados para as constantes como: a = 20, b = 0.2 e c = 2π. Tem-se que o ponto de mínimo global é: f(x) = 0 em x = (0,...,0). A seguir está o código adotando o Simulated Annealing para a busca desse ponto de mínimo global.

```
import numpy as np
numero_variaveis = 10
b = 32.768  ## Limite superior
a = -32.768 ## Limite inferior para os limites = [a,b)

def Custo(X): # ACKLEY FUNCTION
    numero_variaveis = 10
    # transformar matriz 2*10 em vetor de 20 dimensões
    A = np.zeros(2*numero_variaveis)
```

```
    for i in range(0,numero_variaveis):
        A[i] = X[0,i]
    for i in range(0,numero_variaveis):
        A[i+numero_variaveis] = X[1,i]

    c = 2*np.pi
    b = 0.2
    a = 20
    d = 2*numero_variaveis
    sum1 = 0;
    sum2 = 0;
    for ii in range(0,d):
        xi = A[ii]
        sum1 = sum1 + xi**2
        sum2 = sum2 + np.cos(c*xi)

    term1 = -a * np.exp(-b*np.sqrt(sum1/d))
    term2 = -np.exp(sum2/d)

    y = term1 + term2 + a + np.exp(1)

    return y

# Simulated Annealing
X0 = (b-a)*np.random.rand(2,numero_variaveis) + a
N=int(1e5);  K=100; T0=5e-1; e=1e-1
X = X0
Xmin = X0; np.random.seed(0);
fim=0; n=0; k=0; Jmin=Custo(X); Xmin=X; T=T0;

while not(fim):
    T = T0/np.log2(2+k)
    for n in range(N):
        X_hat = X + e*(np.random.normal(0,1,np.shape(X)))
        if np.random.uniform()<np.exp((Custo(X)-Custo(X_hat))/T):
            X = X_hat
            if Custo(X) < Jmin:
                Jmin = Custo(X)
                Xmin = X
    print([k,Jmin])
    k=k+1
    if k == K: fim =1
print(Jmin)
```

4. **Prova de 2009 - Questão 2, itens (a) e (c).**

(*Simulated Annealing*) Considere um problema de otimização representado pela função custo a seguir:

x	J(x)
1	**0.3**
2	**0.1**
3	**0.1**
4	**0.2**

a) Calcule os fatores de Boltzmann $e^{-J(x)/T}$, para T = 1.0 e para T = 0.1.

x	J(x)	$e^{-J(x)/T}$ $T=1.0$	$e^{-J(x)/T}$ $T=0.1$
1	0.3	0.7408182	0.049787
2	0.1	0.9048374	0.367879
3	0.1	0.9048374	0.367879
4	0.2	0.8187307	0.135335

c) Calcule as matrizes de transição M para T = 1.0 e para T = 0.1. Calcule os vetores invariantes destas matrizes e compare-os com os resultados do item (a).

- Para T = 1.0

$$M = \begin{bmatrix} 0 & 0.27291025 & 0.27291025 & 0.30161247 \\ 0.33333333 & 0.09214394 & 0.33333333 & 0.33333333 \\ 0.33333333 & 0.33333333 & 0.09214394 & 0.33333333 \\ 0.33333333 & 0.30161247 & 0.30161247 & 0.03172086 \end{bmatrix}$$

O vetor invariante de M é ([[0.21987801, 0.26855961, 0.26855961, 0.24300278]])

- Para T = 0.1

$$M = \begin{bmatrix} 0 & 0.04511176 & 0.04511176 & 0.12262648 \\ 0.33333333 & 0.49892843 & 0.33333333 & 0.33333333 \\ 0.33333333 & 0.33333333 & 0.49892843 & 0.33333333 \\ 0.33333333 & 0.12262648 & 0.12262648 & 0.21070685 \end{bmatrix}$$

O vetor invariante de M neste caso é ([[0.05406459, 0.3994863 , 0.3994863 , 0.1469628]])

A comparação entre os resultados do item *a* normalizados ($e^{-J(x)/T}/\sum$ **)** com os valores do vetor invariante estão a seguir, onde observa-se que correspondem ao mesmo resultado.

x	J(x)	$e^{-J(x)/T}$ $T=1.0$	$e^{-J(x)/T}/\sum$ $T=1.0$	$e^{-J(x)/T}$ $T=0.1$	$e^{-J(x)/T}/\sum$ $T=0.1$
1	0.3	0.7408182	0.219878	0.049787	0.054064
2	0.1	0.9048374	0.268560	0.367879	0.399486
3	0.1	0.9048374	0.268560	0.367879	0.399486
4	0.2	0.8187307	0.243003	0.135335	0.146963
∑		3.3692238		0.920881	

Os vetores invariantes das matrizes foram calculados com o código a seguir:

```
import numpy as np
T= 0.1

M[0,0] = 0
M[1,0] = 1/3
M[2,0] = 1/3
M[3,0] = 1/3

M[0,1] = 1/3*np.exp(-0.2/T)
M[1,1] = 1/3*(1-np.exp(-0.2/T)) + 1/3*(1-np.exp(-0.1/T))
M[2,1] = 1/3
M[3,1] = 1/3*np.exp(-0.1/T)

M[0,2] = 1/3*np.exp(-0.2/T)
M[1,2] = 1/3
M[2,2] = 1/3*(1-np.exp(-0.2/T)) + 1/3*(1-np.exp(-0.1/T))
M[3,2] = 1/3*np.exp(-0.1/T)

M[0,3] = 1/3*np.exp(-0.1/T)
M[1,3] = 1/3
M[2,3] = 1/3
M[3,3] = 1/3*(1-np.exp(-0.1/T))

autovals, autovecs = np.linalg.eig(M)
print ("Autovetores de A: \n", autovecs[:,0])
print ("Autovalores de A: \n",autovals)

inv = np.zeros(4)
z = np.sum(autovecs[:,0])
for i in range(0,4):
    print(i)
    inv[i] =  autovecs[i,0] / z
print('Vetor inveriante da matriz M: ' + str(inv) )
```

5. Prova de 2011 - Questão 2, itens (a), (b), e (e).
(*Simulated Annealing*) Considere a função custo $J(x_1, x_2)$ definida pela tabela a seguir:

$x1$	$x2$	$J(x)$
0	**0**	**0.2**
0	**1**	**0.3**
1	**0**	**0.3**
1	**1**	**0.1**

a) A aplicação do Algoritmo de Metropolis a um vetor inicial x(0) qualquer, alterando uma componente ($x1$ ou $x2$) de cada vez, define um processo de Markov com duas matrizes de transição: M1 e M2. Calcule estas matrizes de transição, considerando T = 0.5. Note que o número de estados possíveis é 4.

$$M_1 = \begin{bmatrix} 1 - e^{-0.1/T} & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-0.2/T} \\ e^{-0.1/T} & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 - e^{-0.2/T} \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 1 - e^{-0.1/T} & 1 & 0 & 0 \\ e^{-0.1/T} & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{-0.2/T} \\ 0 & 0 & 1 & 1 - e^{-0.2/T} \end{bmatrix}$$

b) Calcule, para temperatura T = 0.5, a distribuição de Boltzmann/Gibbs do vetor aleatório X. Verifique que esta distribuição de probabilidades define um vetor invariante para ambas as matrizes de transição calculadas no item (a).

x	J(x)	$e^{-J(x)/T}$ $T = 0.5$	$e^{-J(x)/T}/\sum$
00	0.2	0.6703200	0.2591436
01	0.3	0.5488116	0.2121688
10	0.3	0.5488116	0.2121688
11	0.1	0.8187307	0.3165187
$\sum$		2.5866741	

Existem dois autovalores unitários, dessa forma dois vetores invariantes para as matrizes M1 e M2 são calculados, de modo que:

Vetor invariante da matriz M1: ([[0.549834 , 0. , 0.450166 , 0.],
[0. , 0.40131234, 0. , 0.59868766]])

Vetor invariante da matriz M2: ([[0.549834 , 0.450166 , 0. , 0.],
[0. , 0. , 0.40131234, 0.59868766]])

Ocorreu que neste caso a distribuição de probabilidades não definiu um vetor invariante para as matrizes de transição.

e) Quando um número suficientemente grande de iterações do algoritmo do item (c) tiver sido calculado à temperatura T = 0.1, com que probabilidade teremos a ocorrência do evento J = 0.3?

$$P(J) = N_J \frac{e^{-J/T}}{z} = 2\,\frac{e^{-0.3/0.5}}{e^{-0.2/0.5} + 2e^{-0.3/0.5} + e^{-0.1/0.5}} = 0.4243377$$

6. **(Opcional/Desafio) Prova de 2012 - Questão 3.**

7. **Prova de 2016 - Questão 2.**

(*Algoritmo de Metropolis*) Considere uma execução do algoritmo de Metropolis à temperatura fixa T = 1, com estados [$X1X2$] ($X1$ e $X2$ são variáveis aleatórias binárias) e as duas matrizes de transição dadas a seguir. A matriz M1, à esquerda, modela as probabilidades de transição entre estados no caso em que a perturbação, sempre diferente de zero, é feita sobre $X1$. A matriz M2, à direita, é para o caso em que a perturbação, sempre diferente de zero, é feita sobre $X2$.

M1	00	01	11	10
00	2/3	0	0	1
01	0	2/3	1	0
11	0	1/3	0	0
10	1/3	0	0	0

M2	00	01	11	10
00	2/3	1	0	0
01	1/3	0	0	0
11	0	0	0	1/3
10	0	0	1	2/3

a) Considerando J (00) = 1, calcule os valores de J (01), J (11) e J (10) de forma que M1 e M2 tenham os valores dados acima.

M1) $00 \rightarrow 10$:

$$00 \rightarrow 10 : e^{-(J_{10}-J_{00})/T} = \frac{1}{3}$$

$$e^{-J_{10}+1} = \frac{1}{3}$$

$$\ln(e^{-J_{10}+1}) = ln\frac{1}{3}$$

$$J_{10} = 1 - ln\frac{1}{3}$$

$$J_{10} = 2.0986$$

M1) $10 \rightarrow 00$: 1
$10 \rightarrow 10$: 0

M2) $11 \rightarrow 10$: 1

M2) $00 \rightarrow 01$:

$$00 \rightarrow 01 : e^{-(J_{01}-J_{00})/T} = \frac{1}{3}$$

$$e^{-J_{01}+1} = \frac{1}{3}$$

$$\ln(e^{-J_{01}+1}) = ln\frac{1}{3}$$

$$J_{01} = 1 - ln\frac{1}{3}$$

$$J_{01} = 2.0986$$

$00 \rightarrow 01 :$ 2/3

M1) $01 \rightarrow 11 :$

$$01 \rightarrow 11 : e^{-(J_{11}-J_{01})/T} = \frac{1}{3}$$

$$e^{-J_{11}+2.0986} = \frac{1}{3}$$

$$\ln(e^{-J_{11}+2.0986}) = ln\frac{1}{3}$$

$$J_{11} = 2.0986 - ln\frac{1}{3}$$

$$J_{10} = 3.1972$$

x1	x2	J(x)
0	0	1
0	1	2.0986
1	1	3.1972
1	0	2.0986

b) Calcule uma matriz de transição M que modele transições de qualquer um dos quatro estados para qualquer um dos quatro estados.

$$M = \begin{bmatrix} 1/3\,(3 - 2e^{-1.0986} - e^{-2.1972}) & 1/3 & 1/3 & 1/3 \\ 1/3\,e^{-1.0986} & 1/3\,(1 - e^{-1.0986}) & 1/3 & 1/3 \\ 1/3\,e^{-2.1972} & 1/3\,e^{-1.0986} & 0 & 1/3\,e^{-1.0986} \\ 1/3\,e^{-1.0986} & 1/3 & 1/3 & 1/3\,(1 - e^{-1.0986}) \end{bmatrix}$$

c) Calcule o vetor invariante da matriz M do item (b). Verifique que ele é um vetor invariante também de M1 e M2, apesar de estas matrizes terem diferentes autovetores correspondentes aos autovalores que têm valor igual a 1.

Vetor inveriante da matriz M (item b): [0.56249654 0.18750115 0.06250115 0.18750115]

Vetor invariante da matriz M1: ([[0.75, 0. , 0. , 0.25],
[0. , 0.75, 0.25, 0.]])

Vetor invariante da matriz M2: ([[0.75, 0.25, 0. , 0.],
[0. , 0. , 0.25, 0.75]])

8. **Prova de 2016 - Questão 3.**

(*Simulated Annealing*) Considere uma função custo dada pela tabela a seguir:

x	1	2	3	4
$J(x)$	7	1	10	4

a) Descreva, usando pseudo-código, a implementação do algoritmo S.A. básico aplicado à minimização da função custo acima. Na sua descrição, leve em consideração os seguintes parâmetros: temperatura inicial $T0$, temperatura mínima T_{min}, e o número de iterações N a serem executadas em temperatura fixa.

```
import math
import numpy as np
def perturbacao(x): ### +1, +2 ou +3
    e = math.ceil(np.remainder(2+(np.random.uniform()*3),3))

    print(e)
    x_hat = x + e
    if x_hat == 5:
        return 1
    if x_hat == 6:
        return 2
    if x_hat == 7:
        return 3
    else:
        return x_hat

def Custo(x):
    if x==1:
        return 7
    if x==2:
        return 1
    if x==3:
        return 10
    if x==4:
        return 4

X0 = 1
X = X0
J0 = Custo(X0)
```

```
Jatual = J0

N = 100
K = 8
T0 = 1
fim = 0
n = 0
k = 1
Jmin = Jatual
Xmin = X


while not(fim):
    T=T0/np.log2(2+k)
    for n in range(0,N):
        X_hat = perturbacao(X)
        JX = Custo(X)
        JX_hat = Custo(X_hat)
        if np.random.uniform() < np.exp((JX-JX_hat)/T):
            X = X_hat
            JX = JX_hat
            if JX<Jmin:
                Jmin = JX
                Xmin = X
        if np.remainder(n+1,100)==0:
            print([k,n+1,Xmin,Jmin])
    k+=1
    if k==K: fim=1

print(Jmin)
```

Resultado: Jmin = 1 e Xmin=2, como esperado.

b) Calcule as matrizes de transição do processo de Markov que corresponde ao S.A. à temperatura T = 10 e à temperatura $T = 5$ (chamadas de M_{10} e M_5) e os seus respectivos vetores invariantes.

$$M_{10} = \begin{bmatrix} 0.0863939 & 0.182937 & 0.333333 & 0.246939 \\ 0.333333 & 0.4346 & 0.333333 & 0.333333 \\ 0.246939 & 0.135523 & 0 & 0.182937 \\ 0.333333 & 0.246939 & 0.333333 & 0.23679 \end{bmatrix}$$

$$M_5 = \begin{bmatrix} 0.15039612 & 0.100398 & 0.333333 & 0.182937 \\ 0.333333 & 0.661565 & 0.333333 & 0.333333 \\ 0.182937 & 0.0550996 & 0 & 0.100398 \\ 0.333333 & 0.182937 & 0.333333 & 0.383331 \end{bmatrix}$$

Vetor invariante de M_{10}: ([[0.20355008, 0.37089243, 0.15079361, 0.27476387]])

Vetor invariante de M_5: ([[0.14945343, 0.49620287, 0.08202178, 0.27232191]])

c) (0.25 ponto extra) Observe o menor dos números em M_{10} e o menor dos números em M_5. Qual é a relação entre estes números e T, J_{max}, J_{min} e o número de estados possíveis?

T = 10 , Jmax = 10, Jmin =1, min(M_{10} = 0.0863939), n = 4

T = 5 , Jmax = 10, Jmin =1, min(M_5 = 0.0550996), n = 4

Com a diminuição da temperatura a probabilidade do estado de menor probabilidade diminuiu, o que indica uma convergência para o ponto de mínimo, como proposto pelo Simulated Annealing.

9. **Prova de 2017 - Questão 3, itens (b) e (c).**

(*Simulated Annealing*) Considere uma função custo definida sobre cinco estados discretos, chamados de estados 1, 2, ... 5, com os seguintes valores: $J(1) = J(5) = 4$, $J(2) = 1$, $J(3) = 3$ e $J(4) = 2$.

b) Calcule uma matriz de transição entre estados à temperatura $T1 = 1/\ln 2$ e uma matriz de transição entre estados à temperatura $T2 = 1/\ln 3$.

$$M = \begin{bmatrix} 0 & 1/4e^{-3/T} & 1/4e^{-1/T} & 1/4e^{-2/T} & 1/4 \\ 1/4 & (1/4)(4-2e^{-\frac{3}{T}}-e^{-\frac{2}{T}}-e^{-\frac{1}{T}}) & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4e^{-2/T} & (1/4)(2-2e^{-\frac{1}{T}}) & 1/4e^{-1/T} & 1/4 \\ 1/4 & 1/4e^{-1/T} & 1/4 & (1/4)(3-2e^{-\frac{2}{T}}-e^{-\frac{1}{T}}) & 1/4 \\ 1/4 & 1/4e^{-3/T} & 1/4e^{-1/T} & 1/4e^{-2/T} & 0 \end{bmatrix}$$

Com T1 = 1/ln2:

$$M_1 = \begin{bmatrix} 0 & 0.03125 & 0.125 & 0.0625 & 0.25 \\ 0.25 & 0.75 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.0625 & 0.25 & 0.125 & 0.25 \\ 0.25 & 0.125 & 0.25 & 0.5 & 0.25 \\ 0.25 & 0.03125 & 0.125 & 0.0625 & 0 \end{bmatrix}$$

Com T2 = 1/ln3:

$$M_2 = \begin{bmatrix} 0 & 0.00925926 & 0.08333333 & 0.02777778 & 0.25 \\ 0.25 & 0.87037037 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.02777778 & 0.33333333 & 0.08333333 & 0.25 \\ 0.25 & 0.08333333 & 0.25 & 0.61111111 & 0.25 \\ 0.25 & 0.00925926 & 0.08333333 & 0.02777778 & 0 \end{bmatrix}$$

c) Calcule os vetores invariantes das matrizes encontradas no item (b).

O vetor invariante da matriz M_1 é: ([[0.0625, 0.5 , 0.125 , 0.25 , 0.0625]])

E o vetor invariante da matriz M_2 é: ([[0.02439024, 0.65853659, 0.07317073, 0.2195122 , 0.02439024]])

10. **Prova de 2018 - Questão 3.**

(*Simulated Annealing*) Considere a matriz de transição dada a seguir (calculada usando $T = T_0 = 0.1$):

$$\mathbf{M} = \begin{bmatrix} (1/2)(1-e^{-2}) & 1/2 & 0 & 0 & 1/2 \\ (1/2)e^{-2} & 0 & (1/2)e^{-1} & 0 & 0 \\ 0 & 1/2 & (1/2)(1-e^{-1}) & (1/2)e^{-2} & 0 \\ 0 & 0 & 1/2 & (1/2)(2-e^{-1}-e^{-2}) & 1/2 \\ 1/2 & 0 & 0 & (1/2)e^{-1} & 0 \end{bmatrix}$$

a) Considerando um grafo com cinco estados (estado 1 conectado aos estados 2 e 5; 2 conectado a 1 e 3; 3 conectado a 2 e 4; 4 conectado a 3 e 5; 5 conectado a 4 e 1) e considerando que o custo associado ao estado 1 é igual a 0.2, calcule os custos associados aos estados 2, 3, 4 e 5.

$$1 \rightarrow 2 : \frac{1}{2}e^{-(J_2-J_1)/T} = \frac{1}{2}e^{-2}$$
$$-J_2 + 0.2 = -0.2$$
$$J_2 = 0.4$$

$$3 \rightarrow 2 : \frac{1}{2}e^{-(J_2-J_3)/T} = \frac{1}{2}e^{-1}$$
$$-0.4 + J_3 = -0.1$$
$$J_3 = 0.3$$

$$4 \rightarrow 3 : \frac{1}{2}e^{-(J_3-J_4)/T} = \frac{1}{2}e^{-2}$$
$$-0.3 + J_4 = -0.2$$
$$J_4 = 0.1$$

$$4 \rightarrow 5 : \frac{1}{2}e^{-(J_5-J_4)/T} = \frac{1}{2}e^{-1}$$
$$-J_5 + 0.1 = -0.1$$
$$J_5 = 0.2$$

x	J(x)
1	0.2
2	0.4
3	0.3
4	0.1
5	0.2

b) Calcule o vetor invariante da matriz M.

```
import numpy as np
M = np.zeros([5,5])
M[0,0] = 1/2*(1-np.exp(-2))
M[1,0] = 1/2*np.exp(-2)
M[2,0] = 0
M[3,0] = 0
M[4,0] = 1/2
M[0,1] = 1/2
M[1,1] = 0
M[2,1] = 1/2
M[3,1] = 0
M[4,1] = 0
M[0,2] = 0
M[1,2] = 1/2*np.exp(-1)
M[2,2] = 1/2*(1-np.exp(-1))
M[3,2] = 1/2
```

```
M[4,2] = 0
M[0,3] = 0
M[1,3] = 0
M[2,3] = 1/2*np.exp(-2)
M[3,3] = 1/2*(2-np.exp(-1)-np.exp(-2))
M[4,3] = 1/2*np.exp(-1)
M[0,4] = 1/2
M[1,4] = 0
M[2,4] = 0
M[3,4] = 1/2
M[4,4] = 0

autovals, autovecs = np.linalg.eig(M)

inv = np.zeros(5)

z = np.sum(autovecs[:,0])
for i in range(0,5):
    inv[i] =  autovecs[i,0] / z
print('Vetor inveriante da matriz M: ' + str(inv) )
```

Vetor inveriante da matriz M: [0.19151597 0.02591887 0.07045479 0.52059439 0.19151597]

c) Escreva a menor das probabilidades da matriz M em função dos valores máximo e mínimo de $J(x)$, da temperatura T e do número N das transições possíveis para cada estado. Recalcule esta probabilidade para $T = T_0/\log_2 4$.

x	J(x)	$e^{-J(x)/T}$ $T_0 = 0.1$	$e^{-J(x)/T}/\sum$ $T=0.1$	$e^{-J(x)/T}$ $T=T_0/\log 2$	$e^{-J(x)/T}/\sum$ $T=T_0/\log 2$
1	0.2	0.13533	0.19151	0.54768	0.21556
2	0.4	0.01831	0.02592	0.29995	0.11806
3	0.3	0.04979	0.07045	0.40531	0.15953
4	0.1	0.36788	0.52059	0.74005	0.29128
5	0.2	0.13533	0.19152	0.54768	0.21556
∑		0.70665		2.54069	