Universidade Federal do Rio de Janeiro
COPPE
Programa de Engenharia Elétrica - PEE

Disciplina: Otimização Natural

Aluno: Gustavo Martins da Silva Nunes
Professor: José Gabriel

Data: 24/03/2016

Lista 2 - Resolução

Questão 1

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

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

a) Considerando que a distribuição de probabilidade de $X(0)$ é dada pelo vetor $\mathbf{p}_0 = [0.3 \quad 0.4 \quad 0.3]^T$, calcule a distribuição de probabilidade de $X(3)$ (ou seja, do processo de Markov no instante $t = 3$).

A distribuição de probabilidade dos estados em cada instante t é dada por:

$$\mathbf{p}_t = \mathbf{M} \times \mathbf{p}_{t-1}, \quad t = 1, 2, ... \tag{1}$$

Seguindo a Equação (1), temos, para $t = 1$:

$$\mathbf{p}_1 = \mathbf{M} \times \mathbf{p}_0 = \begin{bmatrix} 0.50 & 0.25 & 0.25 \\ 0.25 & 0.50 & 0.25 \\ 0.25 & 0.25 & 0.50 \end{bmatrix} \times \begin{bmatrix} 0.3 \\ 0.4 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 0.3250 \\ 0.3500 \\ 0.3250 \end{bmatrix}$$

Da mesma forma, calcula-se o vetor $\mathbf{p}_2$, substituindo, somente, o estado inicial $\mathbf{p}_0$ por $\mathbf{p}_1$:

$$\mathbf{p}_1 = \mathbf{M} \times \mathbf{p}_0 = \begin{bmatrix} 0.50 & 0.25 & 0.25 \\ 0.25 & 0.50 & 0.25 \\ 0.25 & 0.25 & 0.50 \end{bmatrix} \times \begin{bmatrix} 0.3250, \\ 0.3500 \\ 0.3250 \end{bmatrix} = \begin{bmatrix} 0.3312 \\ 0.3375 \\ 0.3313 \end{bmatrix}$$

Finalmente, em $t = 3$:

$$\mathbf{p}_1 = \mathbf{M} \times \mathbf{p}_0 = \begin{bmatrix} 0.50 & 0.25 & 0.25 \\ 0.25 & 0.50 & 0.25 \\ 0.25 & 0.25 & 0.50 \end{bmatrix} \times \begin{bmatrix} 0.3312, \\ 0.3375 \\ 0.3313 \end{bmatrix} = \begin{bmatrix} 0.3328 \\ 0.3344 \\ 0.3328 \end{bmatrix}$$

O código, em MATLAB, que implementa a solução deste item encontra-se abaixo:

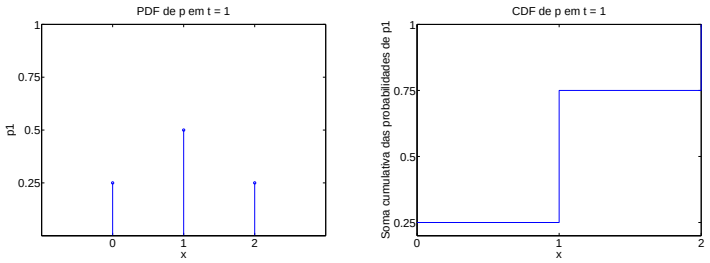
```
M = [0.50 0.25 0.25; 0.25 0.50 0.25; 0.25 0.25 0.50];
p0 = [0.3 0.4 0.3]';

p = p0;

for i = 1:3
    p = M * p;
end

p

% Resultado:
%
% p =
%
%    0.3328
%    0.3344
%    0.3328
```

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

Dado que o estado inicial é 1, temos que o vetor de distribuição de probabilidades inicial $\mathbf{p}_0 = [0 \quad 1 \quad 0]^T$. Aplicando esse vetor na Equação (1), encontra-se $\mathbf{p}_1$.

$$\mathbf{p}_1 = \mathbf{M} \times \mathbf{p}_0 = \begin{bmatrix} 0.50 & 0.25 & 0.25 \\ 0.25 & 0.50 & 0.25 \\ 0.25 & 0.25 & 0.50 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.50 \\ 0.25 \end{bmatrix}$$

Com isso, as amostras do processo $X(t)$, no instante $t = 1$ seguem a distribuição de probabilidade dada por $\mathbf{p}_1$. A PDF e a CDF dessa distribuição estão exibidas na Figura 1.



(a) PDF de $\mathbf{p}_1$ (b) CDF de $\mathbf{p}_1$

Figura 1: Informações sobre a distribuição de probabilidade $\mathbf{p}_1$

Observando a CDF na Figura 1b, conclui-se que $P(X(1) \leq 0) = P(X(1) = 0) = 0,25$, $P((X(1) \leq 1) \cap (X(1) > 0)) = P(X(1) = 1) = 0,50$ e $P(X(1) > 1) = P(X(1) = 2) = 0,25$. Sendo assim, sorteando-se um número aleatório da distribuição uniforme entre [0, 1] e mapeando esse número em um estado segundo a CDF apresentada, obtém-se um estado $X(1)$ segundo a distribuição de probabilidade dada por $\mathbf{p}_1$.

$$X(1) = \begin{cases} 0, & r \leq 0.25 \\ 1, & 0 < r \leq 0.75 \\ 2, & r > 0.75 \end{cases} \quad , \quad r \in (0, 1)$$

Sorteando, então, $r = 0,1$, chega-se a $X(1) = 0$. O mesmo procedimento é adotado para as iterações subsequentes, calculando-se a nova distribuição $\mathbf{p}$ e sorteando uma amostra segundo essa distribuição, usando a mesma abordagem (alterando, somente, os intervalos de r que mapeiam o estado, de forma a respeitar a distribuição em questão).

$$\mathbf{p}_2 = \mathbf{M} \times \mathbf{p}_1 = \begin{bmatrix} 0.50 & 0.25 & 0.25 \\ 0.25 & 0.50 & 0.25 \\ 0.25 & 0.25 & 0.50 \end{bmatrix} \times \begin{bmatrix} 0.25 \\ 0.50 \\ 0.25 \end{bmatrix} = \begin{bmatrix} 0.3125 \\ 0.3750 \\ 0.3125 \end{bmatrix}$$

Sorteando $r = 0,2$, temos que $X(2) = 0$. Executando a última iteração:

$$\mathbf{p}_2 = \mathbf{M} \times \mathbf{p}_1 = \begin{bmatrix} 0.50 & 0.25 & 0.25 \\ 0.25 & 0.50 & 0.25 \\ 0.25 & 0.25 & 0.50 \end{bmatrix} \times \begin{bmatrix} 0.3125 \\ 0.3750 \\ 0.3125 \end{bmatrix} = \begin{bmatrix} 0.3281 \\ 0.3438 \\ 0.3281 \end{bmatrix}$$

Sorteando $r = 0,8$, temos que $X(3) = 2$. Com isso, os estados sorteados em cada um dos 4 instantes foram: $\mathbf{x} = [X(0) \quad X(1) \quad X(2) \quad X(3)] = [1 \quad 0 \quad 0 \quad 2]$.

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 equiprováveis. Armazene as 100 cadeias obtidas em uma matriz X, com 4 colunas ($t = 0$ até $t = 3$) e 100 linhas.

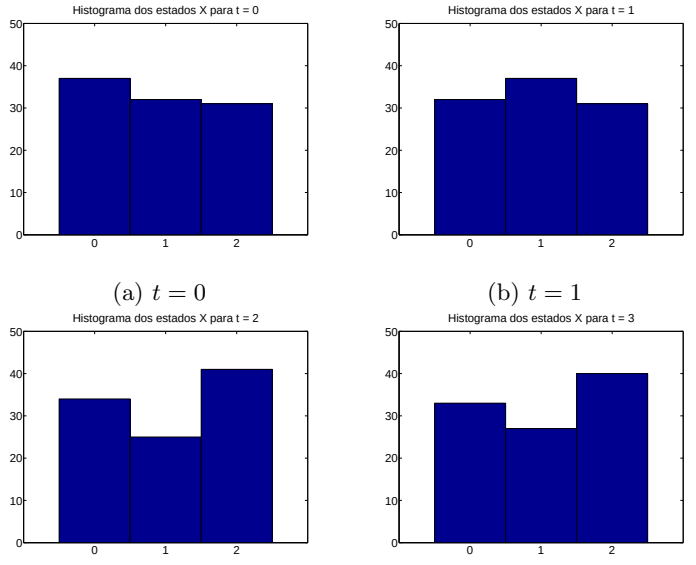
O código, em MATLAB, que implementa a solução deste item é apresentado a seguir:

```
N = 100;
T = 4;

X = zeros(N,T);
M = [0.50 0.25 0.25; 0.25 0.50 0.25; 0.25 0.25 0.50];
p0 = [0.33 0.33 0.33]';

for n = 1:100

    p = p0;

    for i = 0:(T-1)

        if (i ~= 0) % Não é o estado inicial
            p = M * p;
            r = rand();

            p_cdf = cumsum(p); % CDF da distribuição p

            for j = 1:length(p_cdf)
                if r <= p_cdf(j)
                    switch j
                        case 1
                            x = 0;
                        case 2
                            x = 1;
                        case 3
```

```
                                x = 2;
                            end
                        break;
                        end
                    end

                    X(n, i+1) = x;

                else % Estado inicial; apenas inicializa X(n, 1)
                    X(n, i+1) = random('unid', 3) - 1; % Estado Inicial
                end
            end
        end
```

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.

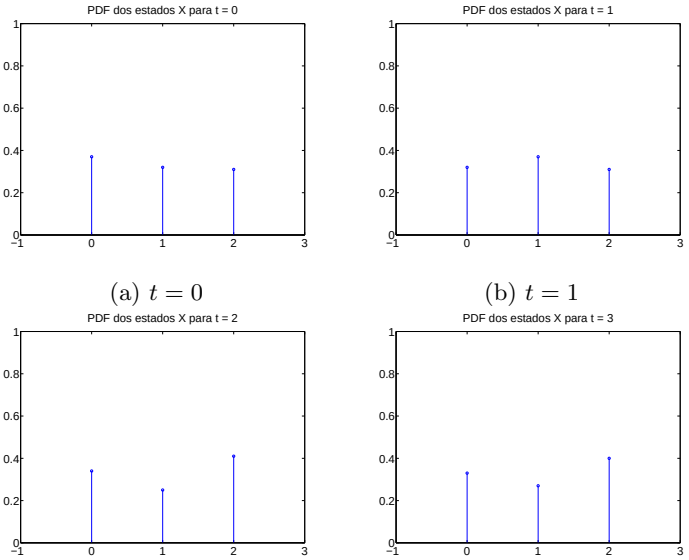
Os histogramas dos estados em cada um dos 4 instantes encontram-se na Figura 2. Uma aproximação das distribuições representadas por cada histogramas estão exibidas na Figura 3.



(a) $t = 0$ (b) $t = 1$

(c) $t = 2$ (d) $t = 3$

Figura 2: Histogramas de $X(t)$



(a) $t = 0$ (b) $t = 1$

(c) $t = 2$ (d) $t = 3$

Figura 3: Distribuições de $X(t)$

Como a distribuição inicial dos estados é equiprovável, segue que o histograma do estado $X(0)$ assemelha-se, de fato, a uma distribuição uniforme entre 3 estados. O que se nota, no entanto, nas distribuições dos estados nos instantes seguintes é que eles também se assemelham à distribuição inicial equiprovável. Isso, no entanto, já era esperado, porque o vetor de distribuições inicial $\mathbf{p}_0 = [0.33 \quad 0.33 \quad 0.33]^T$ é, na verdade, o vetor invariante $\boldsymbol{\pi}$ desse processo de Markov (ou seja, ele é o autovetor da matriz de transição $\mathbf{M}$ associado ao autovalor 1; com isso, a transformação $\mathbf{M}$ aplicada a esse autovetor resulta no mesmo vetor). De fato, é possível verificar que o vetor $\boldsymbol{\pi} = [0.33 \quad 0.33 \quad 0.33]^T$ é o vetor invariante, substituindo-o na equação $p_t = \mathbf{M} \times p_{t-1}$:

$$p_t = \begin{bmatrix} 0.50 & 0.25 & 0.25 \\ 0.25 & 0.50 & 0.25 \\ 0.25 & 0.25 & 0.50 \end{bmatrix} \times \begin{bmatrix} 0.33 \\ 0.33 \\ 0.33 \end{bmatrix} = \begin{bmatrix} 0.33 \\ 0.33 \\ 0.33 \end{bmatrix} = \boldsymbol{\pi}$$

Questão 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 x 5.

Primeiramente, adota-se a seguinte convenção: as colunas da matriz **M** correspondem aos estados de origem do processo e que as linhas correspondem aos estados de destino. Dispondo da tabela de estados e $J(x)$, sorteia-se um estado, que não seja o de origem, e calcula-se, para cada estado de origem, o ΔJ. No caso do algoritmo de Metropolis, se $\Delta J < 0$, então a transição é aceita imediatamente. Caso contrário, a aceitação segue a seguinte fórmula:

$$\hat{X}(t+1) = \begin{cases} X(t+1), & \text{se } e^{-\frac{(\Delta J)}{T}} > r \\ X(t), & \text{caso contrário} \end{cases}, \quad \text{onde } r \in (0,1)$$

Calculando-se, então, os ΔJ referentes ao estado $x = 1$, temos:

$$\Delta J_{12} = J(2) - J(1) = 0.2 - 0.5 = -0.3$$
$$\Delta J_{13} = J(2) - J(1) = 0.3 - 0.5 = -0.2$$
$$\Delta J_{14} = J(2) - J(1) = 0.1 - 0.5 = -0.4$$
$$\Delta J_{15} = J(2) - J(1) = 0.4 - 0.5 = -0.1$$

Por $x = 1$ ser o estado de maior valor da função J, a transição para qualquer outro estado resultará em um $\Delta J < 0$, conforme se observa acima. Portanto, todas as transições serão aceitas. Chamando $\mathbf{c}_1$ de o vetor de probabilidades de transição do estado $X(t) = 1$ para $X(t) = j$, chega-se ao seguinte formato para ele:

$$\mathbf{c}_1 = \begin{bmatrix} (4 - \sum_{j \neq i} P(X(t+1) = j \mid X(t) = 1) \\ P(X(t+1) = 2) \\ P(X(t+1) = 3) \\ P(X(t+1) = 4) \\ P(X(t+1) = 5) \end{bmatrix}$$

Aqui, temos que $P(X(t+1) = j) = P(X(t+1) = j \mid X(t) = 1) \times P(sortear\ X(t+1) = 2)$. A probabilidade $P(X(t+1) = j \mid X(t) = 1)$ é a probabilidade de transição do estado $X(t) = 1$ para $X(t+1) = j$, o qual, por sua vez, é a probabilidade de se aceitar uma transição no algoritmo de Metropolis, dada por $e^{-\frac{(\Delta J)}{T}}$, se $\Delta J > 0$, ou 1, se $\Delta J < 0$. Sendo assim:

$$P(X(t+1) = j \mid X(t) = 1) = \begin{cases} e^{-\frac{(\Delta J)}{T}}, & \text{se } \Delta J > 0 \\ 1, & \text{se } \Delta J < 0 \end{cases}$$

Já $P(sortear\ X(t+1) = 2) = \frac{1}{4}$, porque assume-se que não se pode sortear o estado de origem. Com isso, restam 4 estados, os quais têm probabilidade igual de serem sorteados. A probabilidade de permanecer no estado $X(t+1) = 1$ é o complemento da soma das probabilidades de transição, descritas nas outras posições do vetor. Os demais vetores $\mathbf{c}_i, i = 2, 3, 4, 5$, são análogos ao vetor $\mathbf{c}_1$, mudando, somente, no vetor, a posição das probabilidades. Monta-se, então, a matriz de transição $\mathbf{M} = [\mathbf{c}_1 \quad \mathbf{c}_2 \quad \mathbf{c}_3 \quad \mathbf{c}_4 \quad \mathbf{c}_5]$. Após calcular cada uma dessas probabilidades, chega-se na matriz $\mathbf{M}$ final, exibida abaixo:

$$\mathbf{M} = \begin{bmatrix} 0 & 0.0124 & 0.0338 & 0.0046 & 0.0920 \\ 0.2500 & 0.6117 & 0.2500 & 0.0920 & 0.2500 \\ 0.2500 & 0.0920 & 0.3742 & 0.0338 & 0.2500 \\ 0.2500 & 0.2500 & 0.2500 & 0.8572 & 0.2500 \\ 0.2500 & 0.0338 & 0.0920 & 0.0124 & 0.1580 \end{bmatrix}$$

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

Como $X(0) = 1$, segue que $\mathbf{p}_0 = [1 \quad 0 \quad 0 \quad 0 \quad 0]^T$. A distribuição de probabilidades $\mathbf{p}_1$ é, então, calculada, utilizando a matriz de transição $\mathbf{M}$ obtida no item anterior.

$$\mathbf{p}_1 = \mathbf{M}\times\mathbf{p}_0 = \begin{bmatrix} 0 & 0.0124 & 0.0338 & 0.0046 & 0.0920 \\ 0.2500 & 0.6117 & 0.2500 & 0.0920 & 0.2500 \\ 0.2500 & 0.0920 & 0.3742 & 0.0338 & 0.2500 \\ 0.2500 & 0.2500 & 0.2500 & 0.8572 & 0.2500 \\ 0.2500 & 0.0338 & 0.0920 & 0.0124 & 0.1580 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.2500 \\ 0.2500 \\ 0.2500 \\ 0.2500 \end{bmatrix}$$

Sorteia-se, então, uma amostra da distribuição $\mathbf{p}_1$. Para tal, adota-se a mesma abordagem utilizada no item (b) da questão 1: sorteia-se um número da distribuição uniforme (0,1) e ele é mapeado em algum estado possível, seguindo a CDF obtida do vetor $\mathbf{p}_1$. Para essa iteração, o mapeamento está exibido abaixo. A extensão para as iterações subsequentes é direta, alterando, somente, os intervalos em que cada estado é mapeado, respeitando a CDF em questão, e, portanto, ela será omitida.

$$X(1) = \begin{cases} 2, & r \leq 0.25 \\ 3, & 0.25 < r \leq 0.5 \\ 4, & 0.5 < r \leq 0.75 \\ 5, & r > 0.75 \end{cases} \quad , \quad r \in (0,1)$$

Sorteando $r = 0,8$, chega-se no estado $X(1) = 5$. Iniciando a segunda iteração, calcula-se $\mathbf{p}_2$:

$$\mathbf{p}_2 = \begin{bmatrix} 0 & 0.0124 & 0.0338 & 0.0046 & 0.0920 \\ 0.2500 & 0.6117 & 0.2500 & 0.0920 & 0.2500 \\ 0.2500 & 0.0920 & 0.3742 & 0.0338 & 0.2500 \\ 0.2500 & 0.2500 & 0.2500 & 0.8572 & 0.2500 \\ 0.2500 & 0.0338 & 0.0920 & 0.0124 & 0.1580 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0.2500 \\ 0.2500 \\ 0.2500 \\ 0.2500 \end{bmatrix} = \begin{bmatrix} 0.0357 \\ 0.3009 \\ 0.1875 \\ 0.4018 \\ 0.0741 \end{bmatrix}$$

Sorteando $r = 0,6$, obtém-se $X(2) = 4$. Começa-se, então, a terceira iteração:

$$\mathbf{p}_3 = \begin{bmatrix} 0 & 0.0124 & 0.0338 & 0.0046 & 0.0920 \\ 0.2500 & 0.6117 & 0.2500 & 0.0920 & 0.2500 \\ 0.2500 & 0.0920 & 0.3742 & 0.0338 & 0.2500 \\ 0.2500 & 0.2500 & 0.2500 & 0.8572 & 0.2500 \\ 0.2500 & 0.0338 & 0.0920 & 0.0124 & 0.1580 \end{bmatrix} \times \begin{bmatrix} 0.0357 \\ 0.3009 \\ 0.1875 \\ 0.4018 \\ 0.0741 \end{bmatrix} = \begin{bmatrix} 0.0187 \\ 0.2954 \\ 0.1389 \\ 0.4940 \\ 0.0531 \end{bmatrix}$$

Sorteando $r = 0,8$, resulta em $X(3) = 4$. As 4 primeiras amostras do processo são, portanto, $\mathbf{x} = [X(0) \quad X(1) \quad X(2) \quad X(3)] = [1 \quad 5 \quad 4 \quad 4]$.

c) Qual é o vetor invariante da matriz M do item (a)?

Obs.: para facilitar os cálculos, pode-se usar o computador neste item.

O vetor invariante $\boldsymbol{\pi}$ é aquele associado ao autovalor 1 da matriz de transição $\mathbf{M}$. Isso significa que quando a matriz $\mathbf{M}$ é aplicada nesse vetor, o vetor resultante é ele próprio, ou seja, a distribuição dos estados, para instantes t subsequentes não se altera mais. O código que calcula o vetor invariante é exibido a seguir:

```
%% Montagem da matriz de transição M

C1 = [0 1/4 1/4 1/4 1/4]';
C2 = [exp(-3)/4 (3-exp(-3)-exp(-2)-exp(-1))/4 exp(-1)/4 1/4 exp(-2)
    /4]';
C3 = [exp(-2)/4 1/4 (2-exp(-2)-exp(-1))/4 1/4 exp(-1)/4]';
C4 = [exp(-4)/4 exp(-1)/4 exp(-2)/4 (4-exp(-4)-exp(-3)-exp(-2)-exp
    (-1))/4 exp(-3)/4]';
C5 = [exp(-1)/4 1/4 1/4 1/4 (1-exp(-1))/4]';

M = [C1 C2 C3 C4 C5];

%% Inicialização p0

p0 = zeros(1,5)';

% Geração aleatória de p0

r = rand();
b = 1;

for i = 1:(length(p0)-1)
    p0(i) = r;
    c = b - r;
    p0(i+1) = c;
    r = random('unif', 0, c);
    b = c;
end

%% Cálculo do vetor invariante

p_atual = p0;
contador = 1;

while(true)
```

```
    p_seguinte = M * p_atual;
    if norm((p_seguinte - p_atual), 2) == 0
        break
    end
    p_atual = p_seguinte;
    contador = contador + 1;
end

p_inv = p_atual
contador

% Resultados
%
% p_inv =
%
%     0.0117
%     0.2341
%     0.0861
%     0.6364
%     0.0317
%
%
% contador =
%
%     84
```

O vetor invariante, para o qual o processo de Markov em questão converge, é, portanto, $\boldsymbol{\pi} = [0.0117 \quad 0.2341 \quad 0.0861 \quad 0.6364 \quad 0.0317]^T$. A velocidade com a qual a convergência é alcançada depende do estado inicial $\mathbf{p}_0$. Rodando esse código algumas vezes, notou-se que, em média, são necessárias 83 iterações ($t = 83$) para se alcançar o vetor invariante.

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 $T = 0.1$.

A Tabela 1 exibe os fatores de Boltzmann calculados para cada valor da função custo $J(x)$:

x	$J(x)$	$e^{-\frac{J(x)}{T}}$
1	0.5	0.0067
2	0.2	0.1353
3	0.3	0.0498
4	0.1	0.3679
5	0.4	0.0183

Tabela 1: Tabela com os fatores de Boltzmann calculados para cada $J(x)$

$e^{-\frac{J(x)}{T}}$	$p = \frac{e^{-\frac{J(x)}{T}}}{\sum_{i=1}^{5} e^{-\frac{J(x_i)}{T}}}$	$\boldsymbol{\pi}$
0.0067	0.0117	0.0117
0.1353	0.2341	0.2341
0.0498	0.0861	0.0861
0.3679	0.6364	0.6364
0.0183	0.0317	0.0317

Tabela 2: Tabela com os fatores de Boltzmann transformados em uma distribuição

Nota-se que os maiores fatores de Boltzmann estão associados aos menores valores da função $J(x)$, enquanto que os menores fatores encontram-se associados aos maiores valores dessa função. Isso é esperado, já que os estados mais prováveis são aqueles que resultam nos menores valores de $J(x)$. Isso é confirmado através da transformação desses fatores de Boltzmann em uma distribuição, conforme exibido na Tabela 2.

O que se observa, também, é que o vetor correspondente aos fatores de Boltzmann transformados em uma distribuição é igual ao vetor invariante $\boldsymbol{\pi}$, calculado no item (c). De fato, o vetor invariante é a distribuição de estados para a qual o processo de Markov converge. Lembrando que ao executar o algoritmo de Metropolis um número suficiente de vezes, a distribuição de estados também converge para uma distribuição, que é, justamente, a distribuição de Boltzmann. Sendo assim, o vetor invariante $\boldsymbol{\pi}$ corresponde, na verdade, à distribuição de Boltzmann, o que explica a igualdade entre ambos.

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.

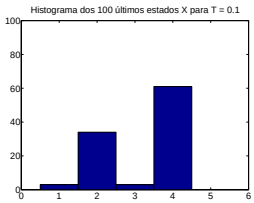
T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
0.1000	0.0631	0.0500	0.0431	0.0387	0.0356	0.0333	0.0315	0.0301	0.0289

O código, em MATLAB, que implementa o Simulated Annealing para esse problema é exibido a seguir.

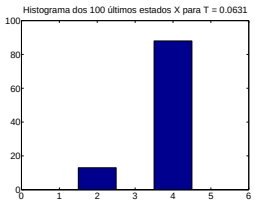
```
T = [0.100 0.0631 0.0500 0.0431 0.0387 0.0356 0.0333 0.0315 0.0301
    0.0289];
J = [0.5 0.2 0.3 0.1 0.4];
x_atual = random('unid',5); % Estado inicial aleatório
```

```
J_atual = J(x_atual);
J_min = Inf;
N = 1000;
X = zeros(length(T), N);
custos = zeros(length(T), N);

for k = 1:length(T)
    for n = 1:N
        x_futuro = random('unid',5);
        J_futuro = J(x_futuro);
        delta_J = J_futuro - J_atual;

        if delta_J < 0
            x_atual = x_futuro;
            J_atual = J_futuro;
        else
            r = rand();
            if r < exp(-(delta_J)/T(k))
                x_atual = x_futuro;
                J_atual = J_futuro;
            end
        end

        if J_atual < J_min
            J_min = J_atual;
            x_min = x_atual;
        end

        X(k, n) = x_atual;
        custos(k, n) = J_atual;
    end
end

for k = 1:length(T)
    figure(k)
    hist(X(k, (N-100):end), [0 1 2 3 4 5 6])
    axis([0 6 0 100])
    set(gca, 'FontSize', 30)
    title(['Histograma dos 100 últimos estados X para T = ' num2str
        (T(k))], 'FontSize', 30)
end
```

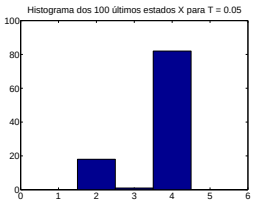
Os histogramas dos estados obtidos ao final de cada temperatura encontram-se na Figura 4. Como era de se esperar, os estados mais prováveis são os dois de menor energia ($x = 2$ e $x = 4$) e, ao longo das temperaturas, eles vão ficando cada vez mais prováveis, indicando que o processo está convergindo para o ponto de mínimo. Tal convergência se verifica, especialmente, nas últimas temperaturas, nas quais sorteou-se, somente, os estados de menor energia.



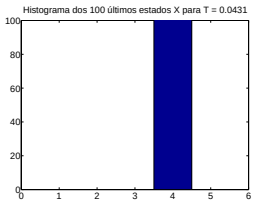
(a) $T = 0.1$



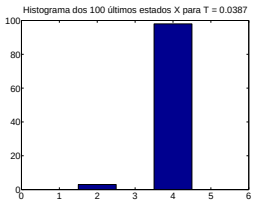
(b) $T = 0.0631$



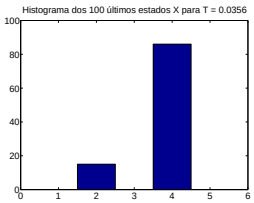
(c) $T = 0.05$



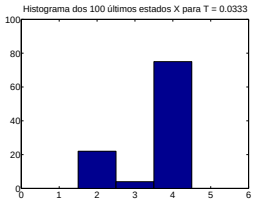
(d) $T = 0.0431$



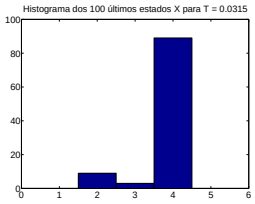
(e) $T = 0.0387$



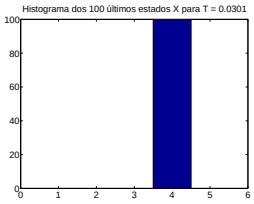
(f) $T = 0.0356$



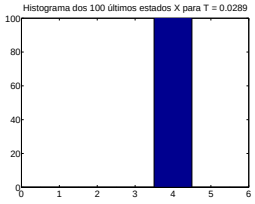
(g) $T = 0.0333$



(h) $T = 0.0315$



(i) $T = 0.0301$



(j) $T = 0.0289$

Figura 4: Histogramas dos últimos 100 estados de cada temperatura T

Questão 3

Proponha uma função $J(\mathbf{x})$, 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 usando S.A.

Obs.: neste exercício, entregue o código utilizado e alguns comentários sobre o resultado obtido.

A função de custo escolhida para se minimizar é a função $sync(\mathbf{x})$ de 10 dimensões. Assumindo que $\mathbf{x} = [x_1 \quad x_2 \quad ... \quad x_{10}]^T$, a fórmula da $sync(\mathbf{x})$ é dada por:

$$J(\mathbf{x}) = \sum_{i=1}^{10} \sin(x_i)/x_i$$

Por ser uma função simétrica, o mínimo global ocorre em diversos pontos. O valor mínimo dessa função é $argmin(J(\mathbf{x})) \approx -2,17$. Essa função, no entanto, tende a uma superfície plana quando $x_i \to \infty,\ i = 1, 2, ..., 10$, e isso pode fazer com que o vetor $\mathbf{x}$ se distancie cada vez mais do mínimo. De modo a evitar isso, restringe-se a região a ser considerada através da colocação de uma função $(a \times sync(\mathbf{x} - \mathbf{b}))^p$, onde $\mathbf{b}$ é um ponto no $\mathbb{R}^{10}$ sobre o qual essa função está centralizada, e $a, r > 0$ tal que eles atribuem um ganho elevado à função. Dessa forma, os estados dificilmente passarão além dessa função (assumindo que o estado inicial fique dentro da região "delimitada"por essa função). A função custo $J(\mathbf{x})$ é, portanto, dada por:

$$J(\mathbf{x}) = \sum_{i=1}^{10} \left(\sin(x_i)/x_i + (2 \times \sin(x_i - 10)/(x_i - 10))^{10}\right)$$

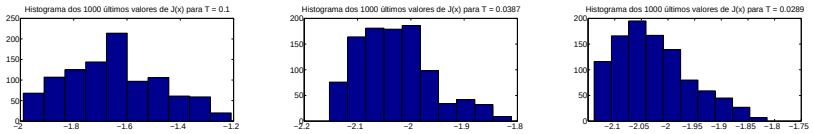
O código, em MATLAB, que implementa a solução, é exibido a seguir.

```
T = zeros(1,10);
T0 = 0.1;
for i = 1:10
    T(i) = T0/log2(1+i);
end

N = 10000;
epsilon = 0.1;

x_atual = random('unif', -5,5,10,1);

J_aux = sin(x_atual)./x_atual + (2*sin(x_atual - 10)./(x_atual -
    10)).^10; % 'Limita' a sync(x) para valores de x entre -10 e 10
```

```
J_aux(isnan(J_aux)) = 1;
J_atual = sum(J_aux);

J_min = J_atual;

J = zeros(length(T), N);
X = zeros(size(x_atual,1),N,length(T));

for k = 1:length(T)
    for n = 1:N

        r = random('unif', -1,1, size(x_atual));

        x_futuro = x_atual + epsilon * r;
        J_aux = sin(x_futuro)./x_futuro + (2*sin(x_futuro - 10)./(
            x_futuro - 10)).^10;
        J_aux(isnan(J_aux)) = 1;
        J_futuro = sum(J_aux);

        delta_J = J_futuro - J_atual;

        if delta_J < 0
            x_atual = x_futuro;
            J_atual = J_futuro;
        else
            a = rand();

            if a < exp(-(delta_J)/T(k))
                x_atual = x_futuro;
                J_atual = J_futuro;
            end
        end

        if J_atual < J_min
            J_min = J_atual;
            X_min = x_atual;
        end

        J(k,n) = J_atual;
        X(:,n,k) = x_atual;

    end
end

% Resultados
%
% J_min =
%
%    -2.1542
%
% X_min =
%
%    -4.3440
%     4.3660
%     4.2649
%    -4.5078
%    -4.5456
```

```
%     4.5105
%     4.6312
%    -4.3770
%     4.5351
%    -4.6876
```

O valor mínimo encontrado é bem próximo do mínimo global da função, apontado anteriormente. Como se observa pelos histogramas dos valores de J para diferentes temperaturas, exibidos na Figura 5, conforme se diminui a temperatura, mais o histograma vai se concentrando próximo ao valor mínimo da função.



(a) $T = 0.1$ (b) $T = 0.0387$ (c) $T = 0.0289$

Figura 5: Histogramas de valores da função $J(\mathbf{x})$ para diferentes temperaturas.