ECE 438 Homework 9

1. To calculate y_0,

$$\int_0^{x_1} x f_X(x)dx = \int_0^{x_1} \frac{1}{9}x^3 dx = \frac{1}{36}x_1^4$$
$$\int_0^{x_1} f_X(x)dx = \int_0^{x_1} \frac{1}{9}x^2 dx = \frac{1}{27}x_1^3$$
$$y_0 = \varphi_0(x_1) = \frac{1/36 \cdot x_1^4}{1/27 \cdot x_1^3} = \frac{3}{4}x_1$$

To calculate y_1,

$$\int_{x_1}^3 x f_X(x)dx = \int_{x_1}^3 \frac{1}{9}x^3 dx = \frac{9}{4} - \frac{1}{36}x_1^4$$
$$\int_{x_1}^3 f_X(x)dx = \int_{x_1}^3 \frac{1}{9}x^2 dx = 1 - \frac{1}{27}x_1^3$$
$$y_1 = \varphi_1(x_1) = \frac{9/4 - 1/36 \cdot x_1^4}{1 - 1/27 \cdot x_1^3} = \frac{243 - 3x_1^4}{108 - 4x_1^3} \text{ , } x_1 \neq 3$$

Then,

$$x_1 = \phi(y_0, y_1) = \frac{y_0 + y_1}{2} = \frac{1}{2}(\frac{3x_1}{4} + \frac{243 - 3x_1^4}{4(27 - x_1^3)}) = \frac{-6x_1^4 + 81x_1 + 243}{-8x_1^3 + 216} \text{ , } x_1 \neq 3$$

To solve x_1 ($x_1 \neq 3$),

$$\begin{aligned} x_1 &= \frac{-6x_1^4 + 81x_1 + 243}{-8x_1^3 + 216} \\ x_1(-8x_1^3 + 216) &= -6x_1^4 + 81x_1 + 243 \\ 2x_1^4 - 135x_1 + 243 &= 0 \\ (x_1 - 3)(2x_1^3 + 6x_1^2 + 18x_1 - 81) &= 0 \\ 2x_1^3 + 6x_1^2 + 18x_1 - 81 &= 0 \end{aligned}$$

Using Newton's method to find the zero of the objective function $2x_1^3 + 6x_1^2 + 18x_1 - 81$, the zero is at $x_1^* = 2.074242$. (The MATLAB code is attached by the end of this problem.)
For finding the other two imaginary roots, let $2x_1^3 + 6x_1^2 + 18x_1 - 81 = 2(x_1 - x_1^*)(x_1 - a)(x - b)$.

$$\begin{aligned} 2x_1^3 + 6x_1^2 + 18x_1 - 81 &= 2(x_1 - x_1^*)(x_1 - a)(x_1 - b) \\ &= 2x_1^3 + 2(-a - b - x_1^*)x_1^2 + 2(ab + x_1^*(a + b))x_1 - 2x_1^* ab \end{aligned}$$

Based on the parameters,

$$\begin{cases} 6 = 2(-a-b-x_1^*) \Rightarrow a+b = -3-x_1^* & (1) \\ 18 = 2(ab + x_1^*(a+b)) \Rightarrow ab + x_1^*(a+b) = 9 & (2) \\ -81 = -2x_1^* ab \Rightarrow ab = \frac{81}{2x_1^*} & (3) \end{cases}$$

With (1) and (3), a and b are the solution to the quadratic equation

$$(z-a)(z-b) = z^2 - (a+b)z + ab = 0$$

Hence,

$$z = \frac{(a+b) \pm \sqrt{(a+b)^2 - 4ab}}{2} = \frac{-3 - x_1^* \pm \sqrt{(3+x_1^*)^2 - 4\frac{81}{2x_1^*}}}{2} = -2.5371 \pm 3.6178j$$

So, the roots of $2x_1^3 + 6x_1^2 + 18x_1 - 81 = 0$ are

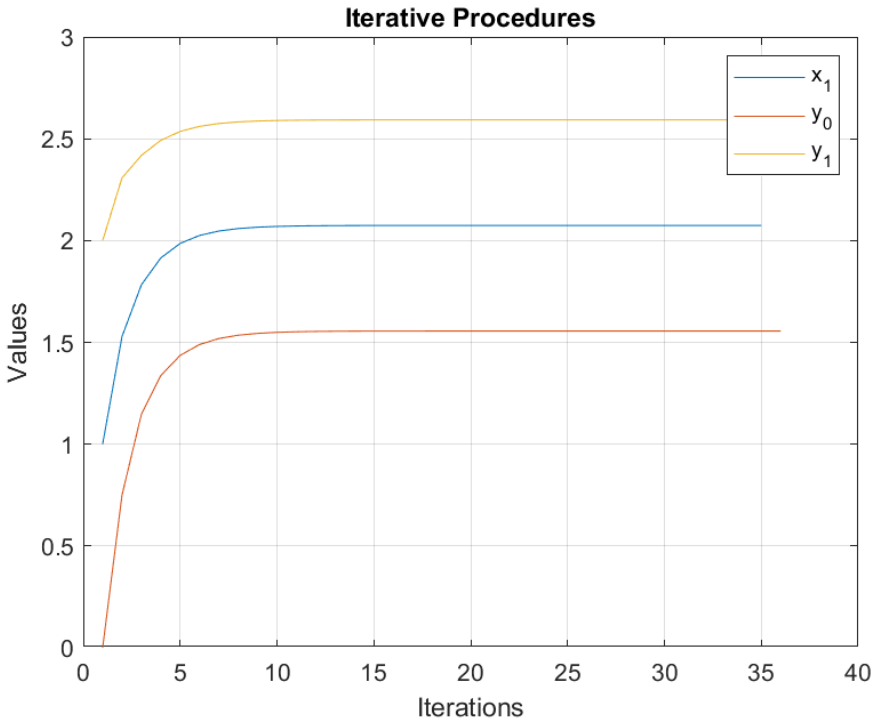
$$x_1 = 2.074242, \ -2.5371 \pm 3.6178j$$

However, in our case, the real root $x_1 = 2.074242$ is the only valid root.

$$\begin{aligned} y_0 &= \varphi_0(x_1) = \frac{3}{4}x_1 = 1.5557 \\ y_1 &= \varphi_1(x_1) = \frac{243 - 3x_1^4}{108 - 4x_1^3} = 2.5928 \\ \phi(y_0, y_1) &= x_1 = 2.0742 \end{aligned}$$

```
% Homework 9 P1
clear;clc;close all;

x0 = 3;              % initial guess
n_iter = 10;        % number of iterations
precision = 1e-5;    % f(x)-0 < precision
epsilon = 1e-9;      % gradient value > epsilon

x = x0;
for i=1:n_iter
    f = objective(x);
    df = gradient(x);
    fprintf("iter %d, x = %.6f, f = %.6f\n", i, x, f);
    if(abs(f)<precision)
        fprintf("x* = %.6f\n", x);
        fprintf("Obejctive Function Value = %.6f\n", f);
        break;
    end
    if(abs(df)<epsilon)
        fprintf("Error: The gradient value is too small.\n
           ");
        break;
    end

    x = x - f/df;
end

if(abs(f)>precision)
    fprintf("Error: Need larger number of iterations.\n");
end


function f = objective(x)
    f = 2*x^(3)+6*x^(2)+18*x-81;
end
function df = gradient(x)
    df = 6*x^(2)+12*x+18;
end
```

2. With the iterative procedures, the program saturated at iteration 35.
   The resulting values are $x_1 = 2.074242$, $y_0 = 1.555681$, and $y_1 = 2.592802$.
   The plot and the MATLAB code are attached below.



```
% Homework 9 P2
clear;clc;close all;

y0 = 0;
y1 = 2;              % initial values
epsilon = 1e-9;      % gradient value > epsilon
y0s = [y0];
y1s = [y1];
xs = [];

i = 1;
diff = 1;
while diff>epsilon
    x = (y0+y1)/2;
    y0 = phi1(x);
    y1 = phi2(x);
    fprintf("iter %d, x = %.6f, y0 = %.6f, y1 = %0.6f\n", i
       , x, y0, y1);

    xs = [xs,x];
    y0s = [y0s,y0];
    y1s = [y1s,y1];

    diff = abs(y1-y1s(i));

    i = i + 1;
end

figure;
plot(xs);
hold on;
plot(y0s);
hold on;
plot(y1s);
xlabel('Iterations'); ylabel('Values'); title('Iterative
   Procedures')
legend('x_{1}','y_{0}','y_{1}'); grid on;
saveas(gcf, 'hw9_p2.png');

function y0 = phi1(x)
    y0 = 3*x/4;
end
function y1 = phi2(x)
    y1 = (243-3*x^(4))/(108-4*x^(3));
end
```

3. (a) The given PDF $f_X(x)$,

$$f_X(x) = \begin{cases} 0 & x < 0 \\ \frac{1}{9}x^2 & 0 \leq x \leq 3 \\ 0 & x > 3 \end{cases} \tag{1}$$

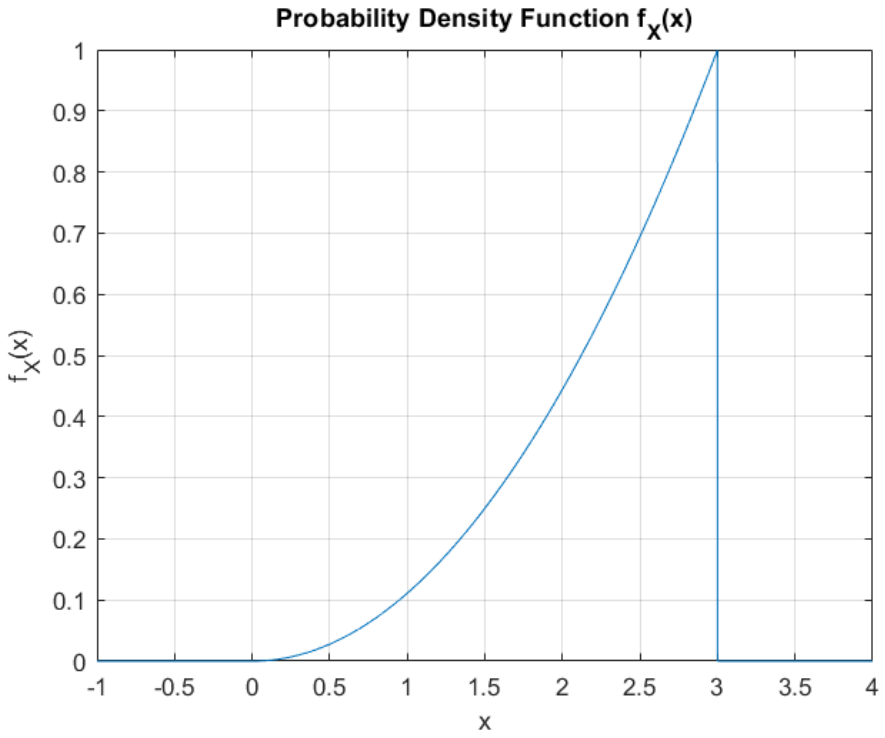
The CDF $F_X(x)$,

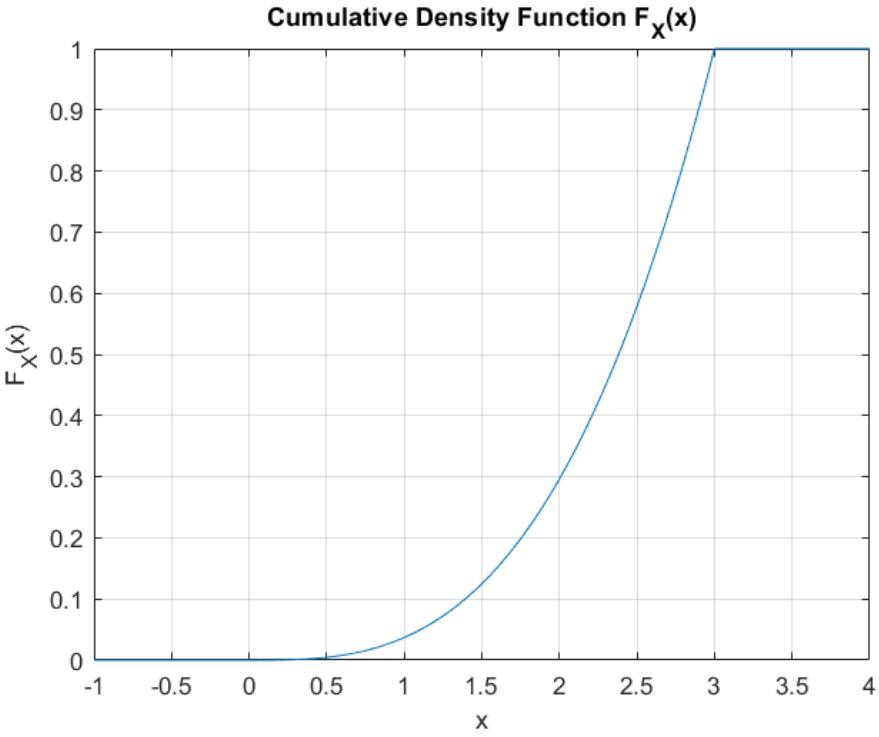
$$F_X(x) = \begin{cases} 0 & x < 0 \\ \frac{1}{27}x^3 & 0 \leq x \leq 3 \\ 1 & x > 3 \end{cases} \tag{2}$$

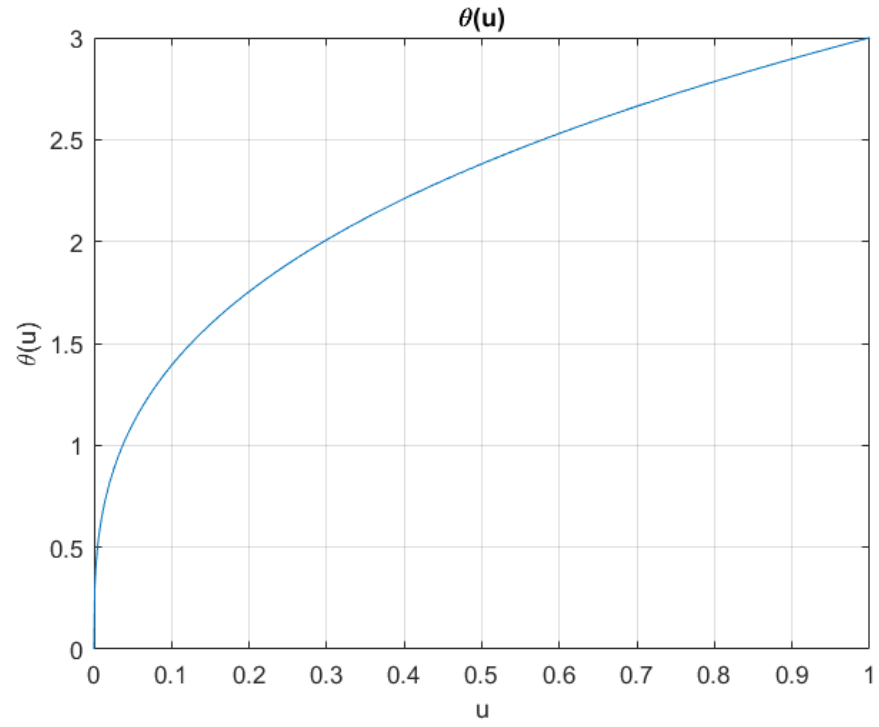
Then, invert $F_X(x)$,

$$\theta(u) = 3u^{1/3} \ , \ 0 \leq u \leq 1$$

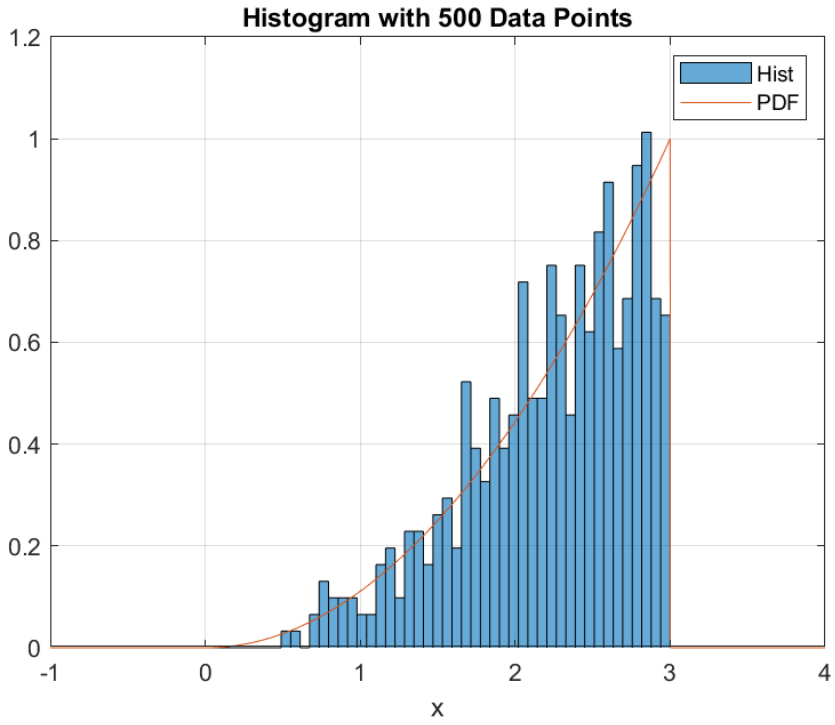
The plots are attached below.

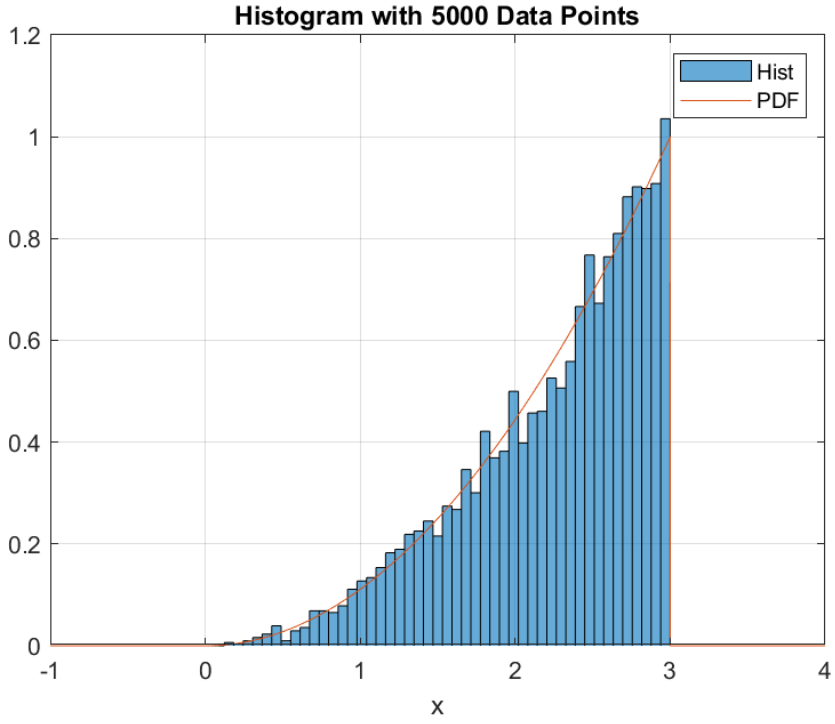


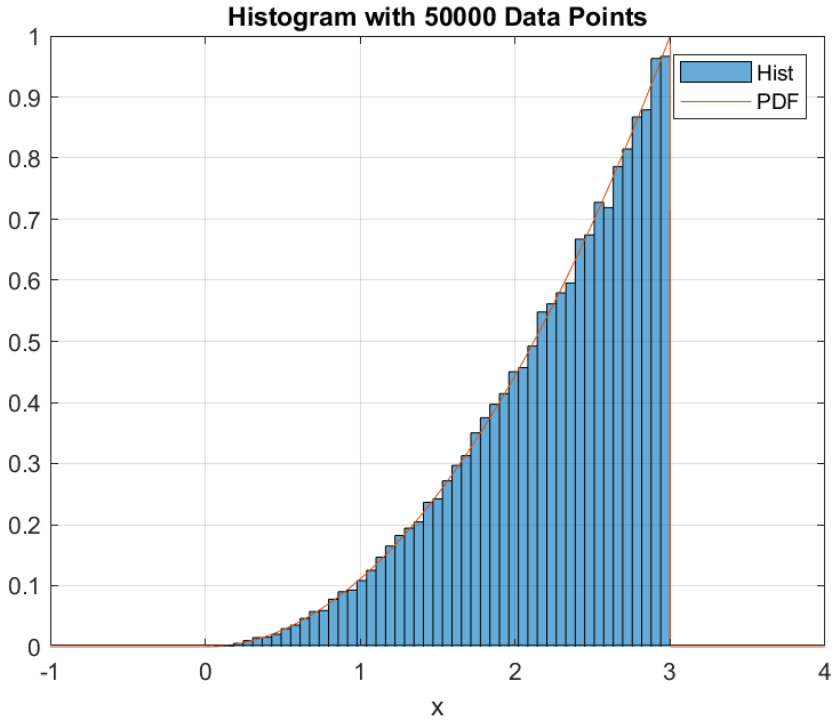

Cumulative Density Function $F_X(x)$
$F_X(x)$
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4


θ(u)
θ(u)
u
3
2.5
2
1.5
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

(b) The histogram plots are attached below. The MATLAB code is attached following the plots. The number of bins is fixed to 50. The bigger the number of data points, the closer the histogram looks to the probability density function plot.




Histogram with 5000 Data Points
Hist
PDF
1.2
1
0.8
0.6
0.4
0.2
0
-1
0
1
2
3
4
x


Histogram with 50000 Data Points
Hist
PDF
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
0
1
2
3
4
x

```
% Homework 9 P3 (b)
clear;clc;close all;

num_points = [5e2,5e3,5e4];
for j=1:length(num_points)
    hist_points(num_points(j), 50, j);
end

function hist_points(n_points, n_bins, cnt)
    x = linspace(-1,4,5001);
    fx = pdf(x);

    u = rand(n_points,1);
    theta = 3.*(u.^(1/3));

    edges = linspace(0,3,n_bins);
    figure;
    histogram(theta,edges,'Normalization','pdf');
    hold on;
    plot(x,fx);
    xlabel('x'); title(['Histogram with ',num2str(
       n_points),' Data Points']);
    legend('Hist','PDF');
    grid on;
    saveas(gcf,['hw9_p3_b_',num2str(cnt),'.png']);
    close all;
end

function fx = pdf(x)
    n = length(x);
    fx = zeros(size(x));
    for i=1:n
        if(x(i)>=0 && x(i)<=3) fx(i) = x(i)^(2)/9;
        end
    end
end
```

(c) Given

$$Q(x) = \begin{cases} 0.75 & 0 \leq x < 1.5 \\ 2.25 & 1.5 \leq x \leq 3 \end{cases} \tag{3}$$

The root-mean square error computed using the given quantizer is 0.444273. MATLAB code is attached below.

```
% Homework 9 P3 (c)
clear;clc;close all;

n_points = 5e4;
u = rand(n_points,1);
theta = 3.*(u.^(1/3));
qx = Quan(theta);
rmse = sqrt(sum((theta-qx).^(2))/n_points);
fprintf("RMSE: %.6f\n", rmse);

function qx = Quan(x)
    n = length(x);
    qx = zeros(size(x));
    for i=1:n
        if(x(i)>=0 && x(i)<1.5) qx(i) = 0.75;
        end
        if(x(i)>=1.5 && x(i)<=3) qx(i) = 2.25;
        end
    end
end
```

(d) Given

$$Q(x) = \begin{cases} 1.5557 & 0 \leq x < 2.0742 \\ 2.5928 & 2.0742 \leq x \leq 3 \end{cases} \tag{4}$$

The root-mean square error computed using the given quantizer is 0.316419. Compared to the uniform quantizer given in part (c), the quantizer calculated in problem 1 has lower root-mean-square error with better performance. MATLAB code is attached below.

```
% Homework 9 P3 (d)
clear;clc;close all;

n_points = 5e4;
u = rand(n_points,1);
theta = 3.*(u.^(1/3));
qx = Quan(theta);
rmse = sqrt(sum((theta-qx).^(2))/n_points);
fprintf("RMSE: %.6f\n", rmse);

function qx = Quan(x)
    x_star = 2.074241939418233;
    y_0 = 0.75*x_star;
    y_1 = (243-3.*x_star.^(4))./(108-4.*x_star.^(3));

    n = length(x);
    qx = zeros(size(x));
    for i=1:n
        if(x(i)>=0 && x(i)<x_star) qx(i) = y_0;
        end
        if(x(i)>=x_star && x(i)<=3) qx(i) = y_1;
        end
    end
end
```