ECON90055 Computational Economics: Assignment 5

Zelin Chen - 797036

June 8, 2017

Contents

Problem 1: Lagrange Polynomial

To find each corresponding value y of x on the Lagrange polynomial that interpolated by given pairs of (x_j, y_j), its algorithm basically involves two steps. First, calculate each product $\ell_{j,n}(x)$ in a loop. n is the size of known interpolations which equals to 10, and let j and k be a sequence of $[1:1:10]$. So, for each value of j, it first calculates each Lagrange basis polynomial:

$$\ell_{j,10}(x) = \prod_{k=1,k\neq j}^{10} \frac{x - x_k}{x_j - x_k}$$

th result for each j is stored in a $m * n$ matrix, m is the length of unknowns that are waiting to be solved (=8), and n=10 as mentioned above.
Next, multiplying each $\ell_{j,10}(x)$ by its corresponding value $f(x_j)$, and then sum all these products up to obtain the value of y for each given x:

$$y = \sum_{j=1}^{10} \ell_{j,10}(x) f(x_j)$$

The result is given below in table 1:

x	0.2000	0.5000	0.6000	0.8000	1.0000	1.4000	1.8000	2.2000
y	0.9652	0.8230	0.7628	0.6354	0.5083	0.2850	0.1234	0.0244

Table 1: (x, y) in Lagrange Polynomial

Listing 1: Q1 Main Calculation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project - "Q1 - Main"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;

x = [0.1 0.3 0.7 0.9 1.2 1.5 1.7 2.0 2.1 2.3];
y = [0.9907 0.9267 0.6997 0.5712 0.3899 0.2384 0.1576
    0.0667 0.0439 0.0080];
z = [0.2 0.5 0.6 0.8 1.0 1.4 1.8 2.2];

fz = Q1_lagrange_interpolation(x, y, z);

% create latex table
q1data = zeros(2,length(z));
q1data(1,:) = z;
q1data(2,:) = fz';
```

```

table.data = q1data;
table.makeCompleteLatexDocument = 1;
table.tableColumnAlignment = 'c';
table.dataFormat = {'%.4f'};
latex = latexTable(table);
```

Listing 2: Q1 Computing Lagrange Polynomials

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project - "Lagrange interpolation"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% function of Lagrange interpolation
function pn = lagrange_interpolation(x, y, z)
    % x: given points of x  (a vector)
    % y: give corresponding value of x (a vector)
    % z: points that we want to find value in the
       polynomial (a vector)

    % know the length of x and z:
    nx = length(x);
    nz = length(z);
    % create space to store temporary values.
    tem = ones(nz,nx);

 % first step: compute each product elements
    for i = 1:nz % solve for 'nz' amount of pn
        for j = 1:nx
            for k = 1:nx
                if k ~= j
                    % conducting product for each
                       element of Lagrange polynomial
                    tem(i,j) = tem(i,j) * (z(i) - x(k)
                       )/(x(j) - x(k));
                end
            end
        end
    end
% second step: multiply product 'tem' by its
   corresponding y, and then sum them up.
    pn = tem * y';
end
```

Problem 2: Solving Linear Equation

(a) diagonal dominance

To check the system to establish $(\mathbf{I}-\mathbf{A})\mathbf{X}=\mathbf{C}$ whether or not is diagonally dominant, that is to check that:

$$|c_{jj}| \geq \sum_{i=1, i\neq j}^{n} |c_{ji}|$$

where n is the total number of column. First I form the coefficient matrix: $(\mathbf{I}-\mathbf{A})$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0.4 & 0.3 \\ 0.2 & 0.12 & 0.14 \\ 0.5 & 0.2 & 0.05 \end{bmatrix} = \begin{bmatrix} 1 & -0.4 & -0.3 \\ -0.2 & 0.88 & -0.14 \\ -0.5 & -0.2 & 0.95 \end{bmatrix}$$

Then from $j=1$ to $j=3$ we check its diagonal dominance:
$|c_{11}|=1, |c_{12}|+|c_{13}|=0.4+0.3=0.7$, implies $|c_{11}|>|c_{12}|+|c_{13}|$
$|c_{22}|=0.88, |c_{21}|+|c_{23}|=0.2+0.14=0.34$, implies $|c_{22}|>|c_{21}|+|c_{23}|$
$|c_{33}|=0.95, |c_{31}|+|c_{32}|=0.5+0.2=0.7$, implies $|c_{33}|>|c_{31}|+|c_{32}|$
All diagonal elements satisfy the diagonal dominance condition, therefore the coefficient matrix $(\mathbf{I}-\mathbf{A})$ is diagonally dominant.

(b) Gaussian elimination

The Gaussian Elimination function is almost the same as the function I created in Assignment 2. Here is again, the explanation on my code:
I created a function called **Q2_gausselim(C,b)** in Matlab. With two inputs, coefficients **C** and results **b**, the function can return the solution **X** by using Guassian Elimination by steps:

1. Take the input and create augmented coefficient matrix. Record its size.
2. Start the loop for each column j.

2-1. First thing is to make sure diagonal element in each column is non-zero. I make it by switching that row (R_j) (the row where zero-value diagonal element exists) with another row $(R_{k,k>j})$ that has largest value in the j_{th} column. I also record road switches, so it won't mix up solutions in its order. Furthermore, if all elements below are zeros, it will skip to the next loop.

2-2. Make the diagonal element value to one through dividing the whole row by the value of that diagonal element. Row_j/c_{jj}

2-3. Remove other rows $(R_{k,k\neq j})$'s value in the j_{th} column (i.e.$c_{kj,k\neq j}$) to zeros. Since diagonal element has become 1, multiply R_j by the value of $R_{k,k\neq j}$ in the j_{th} column that I want to remove $(c_{kj,k\neq j})$ will produce the same value as $c_{kj,k\neq j}$. By subtraction $(R_{k,k\neq j}=R_{k,k\neq j}-R_j*c_{kj,k\neq j}$), it eliminates the value of $c_{kj,k\neq j}$ to 0, and it will repeat this elimination for each $R_{k,k\neq j}$ under each column.

3. After the loop, we will have eliminated augmented matrix as a combination of identity matrix and a column vector for solutions. The value of each x is simply corresponding to each value in the right-hand side of augmented matrix. (Noting that, whole process might involve row switch if there exists zero value for diagonal element, the order recorder that I wrote in the code will help to identify the exact corresponding solution for each x.)

Listing 3: Q2 part(b) Gaussian Elimination

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Assignment 5 - "Gaussian Elimination"
% Computational Economics
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function sol = gausselim(C,b)

% create the argument A
A = [C b];

% row of A: column is assumed to be (n+1)
n = size(A,1);

% predefine storage space for row switch
colbelow = zeros(1,n);
labels = [1:1:n]';

for j = 1:n
    % check for each diagnol element to be non-zero.
       if A(j,j)=0, replace row j by the row below
       with largest number in the same column j
    if A(j,j) == 0
        % gather each element in column j below row j,
            then locate the row of that largest number
           . Switch two rows.
        for k = j+1:n
            colbelow(k) = A(k,j);
        end
        % if all zeros, we skip
        if abs(max(colbelow)) ~=0
            maxlocation = find(colbelow==max(colbelow)
               );
            temrow = A(j,:); %copy jth row out
            A(j,:) = A(maxlocation,:);% replace jth
               row by row we found
            A(maxlocation,:) = temrow; % replace that
               row by jth row
```

```
            % memorise row switch
            labels(j) = maxlocation;
            labels(maxlocation) = j;
        else
            continue % pass to next iteration j+1
        end
    end

    % make diagnol element into 1
    A(j,:) = A(j,:) / A(j,j);

    % make each value other than diagnol element in
       column j become zero
    for k = 1:n
        if (k ~= j) && (A(k,j)~=0)
            A(k,:) = A(k,:) - A(j,:)*A(k,j);
        end
    end
end
% collect values for x:
x = zeros(n,1);
for j = 1:n
    x(j) = A(j,n+1);
end
% indicating which value for which x
sol = [labels x];
end
```

(c) results

Using the Gaussian Elimination function that I wrote in part (b), input the coefficient matrix $(\mathbf{I} - \mathbf{A})$ and exogenous demand vector $\mathbf{C}$, the function will return the solution of commodity output $\mathbf{X}$:

$$X_1 = 300, X_2 = 200, X_3 = 300$$

Listing 4: Q2 part(c) application of Gaussian Elimination function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project - "Q2 - Main"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;

% taking inputs
```

```
% I-A
IA = [1, -0.4, -0.3; -0.2, 0.88, -0.14; -0.5, -0.2,
    0.95];
% exogenous demand
C = [130; 74; 95];

% calling gauss elimination function
X = Q2_gausselim(IA,C);
disp(X)
```

Problem 3: Solving Non-Linear Equation

(a) FOC of log-likelihood function

Set up likelihood function as $\mathbf{L}(\theta|x) = \prod f(x_i;\theta)$, where $\theta = [\theta_1\ \theta_2]^T$:

$$\mathbf{L}(\theta|x) = \frac{\theta_2^{\theta_1} x_1^{\theta_1-1} e^{-\theta_2 x_1}}{\Gamma(\theta_1)} * ... * \frac{\theta_2^{\theta_1} x_n^{\theta_1-1} e^{-\theta_2 x_n}}{\Gamma(\theta_1)}$$

$$= (\frac{\theta_2^{\theta_1}}{\Gamma(\theta_1)})^n (x_1 x_2 ... x_n)^{\theta_1-1} e^{-\theta_2(x_1+x_2+...+x_n)}$$

Then write likelihood function into log-likelihood form:

$$\ln \mathbf{L}(\theta|x) = n(\theta_1 \ln \theta_2 - \ln \Gamma(\theta_1)) + (\theta_1 - 1)\sum_{i=1}^{n} \ln x_i - \theta_2 \sum_{i=1}^{n} x_i$$

Noting that maximum likelihood estimator $\hat{\theta}(x)$ that solves $\arg\max_\theta \ln \mathbf{L}(\theta|x)$, also solves $\arg\max_\theta \mathbf{L}(\theta|x)$. Take FOCs:

$$\frac{\partial \ln \mathbf{L}(\hat{\theta}|x)}{\partial \theta_1} = n \ln \hat{\theta}_2 - n\frac{\partial \ln \Gamma(\hat{\theta}_1)}{\partial \theta_1} + \sum_{i=1}^{n} \ln x_i = 0 \tag{1}$$

$$\frac{\partial \ln \mathbf{L}(\hat{\theta}|x)}{\partial \theta_2} = n\frac{\hat{\theta}_1}{\hat{\theta}_2} - \sum_{i=1}^{n} x_i = 0 \tag{2}$$

(b) optimality condition

Let's assume $\sum_{i=1}^{n} x_i$ has a mean $\bar{x}$ such that $n\bar{x} = \sum_{i=1}^{n} x_i$. Then we can rearrange FOC equation (2) to obtain $\hat{\theta}_2$ in terms of $\hat{\theta}_1$.

$$n\frac{\hat{\theta}_1}{\hat{\theta}_2} - n\bar{x} = 0 \longrightarrow \hat{\theta}_2 = \frac{\hat{\theta}_1}{\bar{x}}$$

Then substitute $\hat{\theta}_2$ into FOC equation (1) to solve for the optimality condition of $\hat{\theta}_1$:

$$n(\ln \hat{\theta}_1 - \ln \bar{x}) - n\frac{\partial \ln \Gamma(\hat{\theta}_1)}{\partial \theta_1} + \sum_{i=1}^{n} \ln x_i = 0$$

Let $\psi(\hat{\theta}_1)$ to denote $\frac{\partial \ln \Gamma(\hat{\theta}_1)}{\partial \theta_1}$, and clean up the above equation:

$$n \ln(\hat{\theta}_1) - n\psi(\hat{\theta}_1) - n \ln \bar{x} + \sum_{i=1}^{n} \ln x_i = 0$$

Rearrange:

$$\ln(\hat{\theta}_1) - \psi(\hat{\theta}_1) - \ln \bar{x} = -\frac{1}{n} \sum_{i=1}^{n} \ln x_i$$

Then take exponential on both side:

$$e^{\ln(\hat{\theta}_1) - \psi(\hat{\theta}_1) - \ln \bar{x}} = e^{-\frac{1}{n} \sum_{i=1}^{n} \ln x_i}$$

Simplify left-hand side:

$$e^{\ln(\hat{\theta}_1)} e^{-\psi(\hat{\theta}_1)} e^{-\ln \bar{x}} = e^{-\frac{1}{n} \sum_{i=1}^{n} \ln x_i}$$

Noting that $e^{-\ln \bar{x}} = \bar{x}^{-1}$, and move $\bar{x}^{-1}$ to the right-hand side:

$$e^{\ln(\hat{\theta}_1)} e^{-\psi(\hat{\theta}_1)} = \frac{\bar{x}}{\exp(\frac{1}{n} \sum_{i=1}^{n} \ln x_i)}$$

Question defines that $Y_1 = \frac{1}{n} \sum_{i=1}^{n} x_i = \bar{x}$, $Y_2 = \exp(\frac{1}{n} \sum_{i=1}^{n} \ln x_i)$. Therefore, we can write optimality condition of $\hat{\theta}_1$ in terms of Y_1 and Y_2:

$$e^{\ln(\hat{\theta}_1) - \psi(\hat{\theta}_1)} = \frac{Y_1}{Y_2}$$

Take log on both side:

$$\ln(\hat{\theta}_1) - \psi(\hat{\theta}_1) = \ln Y_1 - \ln Y_2$$

Now let $\frac{\partial \ln \mathbf{L}(\hat{\theta}|x)}{\partial \theta_1}$ be a function $g(\theta_1, x)$ such that:

$$g(\theta_1; x) = \ln(\hat{\theta}_1) - \psi(\hat{\theta}_1) - \ln Y_1 + \ln Y_2$$

where Y_1 and Y_2 are getting from observations x. So we want to find $\hat{\theta}_1$ such that $g(\hat{\theta}_1, x) = 0$ that satisfies the FOC.

(c) Newton's Method

Find solution of $\hat{\theta}_1$ by Newton's Method. The iteration scheme is set as following:

$$\theta_{1,k} = \theta_{1,k-1} - \frac{g(\theta_{1,k-1}, x)}{g'(\theta_{1,k-1}, x)}$$

where

$$g(\theta_{1,k-1}; x) = \ln(\theta_1) - \psi(\theta_1) - \ln Y_1 + \ln Y_2$$
$$g'(\theta_{1,k-1}; x) = \frac{1}{\theta_1} - \psi'(\theta_1)$$

[1]Now we can write Matlab function to solve for $\hat{\theta}_1$ and $\hat{\theta}_2$. Let's create a series of random numbers x_i from range(0,1), size n equals to 1000000, and has the property that $Y_1 \approx 0.5000$ and $Y_2 \approx 0.3678$.[2] My written Matlab function **Q3_MLE_fn** solves $\hat{\theta}_1$, and then $\hat{\theta}_2$ can be computed by $\frac{\hat{\theta}_1}{Y_1}$:

$$\begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \end{bmatrix} = \begin{bmatrix} 1.7779 \\ 3.5559 \end{bmatrix}$$

Next, solve for covariance matrix. First, obtain the Fisher Information Matrix.

$$I(\theta_1, \theta_2) = \begin{bmatrix} I(\theta_1, \theta_2)_{11} & I(\theta_1, \theta_2)_{12} \\ I(\theta_1, \theta_2)_{21} & I(\theta_1, \theta_2)_{22} \end{bmatrix}$$

where:

$$I(\theta_1, \theta_2)_{11} = -\frac{\partial^2}{\partial \theta_1^2} \ln \mathbf{L}(\theta|x) = n\frac{\partial^2 \ln \Gamma(\theta_1)}{\partial \theta_1^2}$$
$$I(\theta_1, \theta_2)_{22} = -\frac{\partial^2}{\partial \theta_2^2} \ln \mathbf{L}(\theta|x) = n\frac{\theta_1}{\theta_2^2}$$
$$I(\theta_1, \theta_2)_{12} = I(\theta_1, \theta_2)_{21} = -\frac{\partial^2}{\partial \theta_1 \partial \theta_2} \ln \mathbf{L}(\theta|x) = -\frac{n}{\theta_2}$$

So, we get Fisher Information Matrix:

$$I(\theta_1, \theta_2) = \begin{bmatrix} n\frac{\partial^2 \ln \Gamma(\theta_1)}{\partial \theta_1^2} & -\frac{n}{\theta_2} \\ -\frac{n}{\theta_2} & n\frac{\theta_1}{\theta_2^2} \end{bmatrix}$$

Knowing that covariance matrix of Maximum Likelihood Estimation equals to the inverse of Fisher Information matrix: $V(\theta) = I(\theta)^{-1}$. Let Matlab solves the covariance matrix: (by **Q3_cov($\hat{\theta}_1, \hat{\theta}_2, n$)**)

$$Cov(\hat{\theta}_1, \hat{\theta}_2) = \begin{bmatrix} 0.0534 & 0.1068 \\ 0.1068 & 0.2842 \end{bmatrix} * 10^{-4}$$

[1] $g(\theta_{1,k-1}; x)$ corresponds to $Q3\_g$ function, $g'(\theta_{1,k-1}; x)$ corresponds to $Q3\_g\_d$.

[2] There exists truncation errors, as we cannot get the exact expected value. Means are obtained via $\frac{1}{n}\sum_{i=1}^{n} x_i$ and $\frac{1}{n}\sum_{i=1}^{n} \ln x_i$. Therefore, answers may differ every time depends on the value of arithmetic mean and geometric mean.

Listing 5: Q3 part(b) main script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project -"Q3 - Main"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;

% define observations
n = 1000000;
x = rand([1,n]);
y1 = mean(x);
y2 = exp(mean(log(x)));
% calculation y1,y2 given observation x
y1y2 = log(y1)-log(y2);

% solve for ML estimators and covariance matrix
theta1 = Q3_MLE_fn(y1y2);
theta2 = theta1/y1;
V = Q3_cov(theta1,theta2,n);

disp(theta1);
disp(theta2);
disp(V);
```

Listing 6: Q3 part(b) main function that finds $\hat{\theta}_1$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project -"Q3 - MLE_fn"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%tailored for part(d), otherwise will include soluton
    for theta2
function theta_1 = Q3_MLE_fn(y1y2)
    % set tolerance
    tol = 0.0000001;
    % initial starting value
    theta_1_old = 1;
    theta_err = 1;

    % Newton's method find theta_1 that solves FOC
    while theta_err > tol
        theta_1_new = theta_1_old - Q3_g(theta_1_old,
            y1y2)/Q3_g_d(theta_1_old);
```

```
        theta_err = theta_1_new - theta_1_old;
        theta_1_old = theta_1_new;
    end
    % report results:
    theta_1 = theta_1_new;
end
```

Listing 7: Q3 part(b) main function that finds covariance matrix

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project -"Q3 - Covariance_Matrix"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function V = Q3_cov(theta_1,theta_2,n)
    % Covariance matrix

    % find Fisher Information Matrix
    I = [n*psi(1,theta_1), -n/theta_2; -n/theta_2, n*
        theta_1/(theta_2^2)];
    % inv of I = covariance matrix of maximum
        likelihood estiamtion
    V = inv(I);

end
```

Listing 8: Q3 part(b) $g(\theta_1)$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project -"Q3 - g"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function z = Q3_g(theta_1, y1y2)
    % psi(0,x) is psi function
    z = log(theta_1)-psi(theta_1) - y1y2;
end
```

Listing 9: Q3 part(b) $g'(\theta_1)$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project -"Q3 - g_d"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

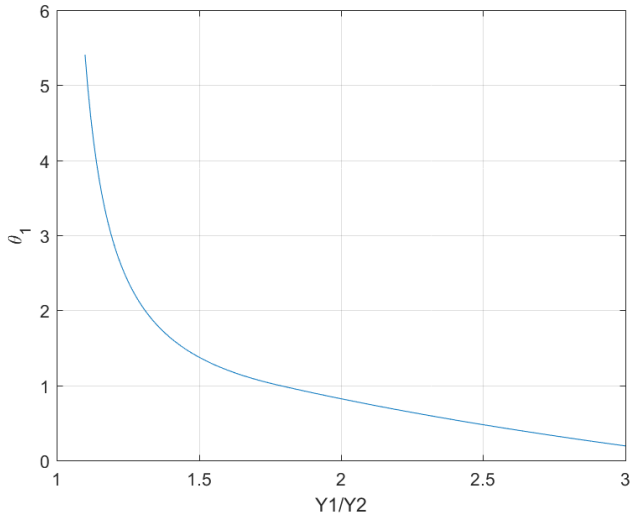


Figure 1: change of $\hat{\theta}_1$ by ratio $\frac{Y_1}{Y_2}$

```
function z = Q3_g_d(theta_1)
    % psi(1,x) is trigamma function
    z = 1/theta_1 - psi(1,theta_1);
end
```

(d) plot

Noting that the FOC of $\hat{\theta}_1$ is written as $\ln(\theta_1) - \psi(\theta_1) - (\ln Y_1 - \ln Y_2) = 0$. Therefore, given a sequence of $\frac{Y_1}{Y_2}$, it first needs to take log to obtain $\ln Y_1 - \ln Y_2$. Then I can use Newton's Method to solve for each corresponding $\hat{\theta}_1$ as illustrated in part(c). From Figure 1 we see that $\hat{\theta}_1$ is high when ratio of $\frac{Y_1}{Y_2}$ is relatively low. But $\hat{\theta}_1$ starts to plunge quickly when the ratio of $\frac{Y_1}{Y_2}$ rises. After a certain level, (around $\frac{Y_1}{Y_2} = 1.5$) the marginal reduction on $\hat{\theta}_1$ given increase of $\frac{Y_1}{Y_2}$ becomes small and relatively constant.

Listing 10: Q3 part(d) code for plot

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project -"Q3 - plot"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot theta1 for Y1/Y2 over range [1.1,3]

```

```
% slice the range of Y1/Y2 and get its length
y1y2_range = [1.1:0.01:3];
y1y2_length = length(y1y2_range);

% tranform Y1/Y2 into lnY1-lnY2
y1y2_log = log(y1y2_range);

% store value of theta1 in each loop
theta1 = zeros(1,y1y2_length);

% for each value of Y1/Y2
for i = 1:y1y2_length
    y1y2 = y1y2_log(i);
    % store solution theta1 for each value of Y1/Y2
    theta1(i) = Q3_MLE_fn(y1y2);
end

plot(y1y2_range, theta1)
grid on
xlabel('Y1/Y2')
ylabel('\theta_1')
saveas(gcf,'Q3plot.png')
```

Problem 4: Quadratic Programming

(a) graphic solution

I first drew a 3-dimensional graph (Figure 2) to indicate all feasible combinations of x_1 and x_2, and their corresponding value of z, noting that $z = x_1^2 + x_2^2 - 8x_1 - 10x_2$. The graphical minimal is difficult to capture by eyes in Figure 2. Therefore I drew another two figures (Figure 3) to present the minimal value of z in a clear manner. Figure 3a represents all feasible values of z given x_1, and Figure 3b represents all feasible values of z given x_2. Now minimal value of z and its corresponding value of x_1 and x_2 becomes more obvious. Figure 3a shows that when $x_1 = 0.308$, z reaches minimal. Figure 3b shows that the corresponding value of x_2 when z equals to its minimal value is 2.538. Therefore the solution is:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.308 \\ 2.538 \end{bmatrix}$$

Noting that the accuracy of this answer is only up to three decimal places, since 0.001 is set as the interval to slice x_1 and x_2 when I construct graphical solution.

Listing 11: Q4 part(a) Graphic Solution

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

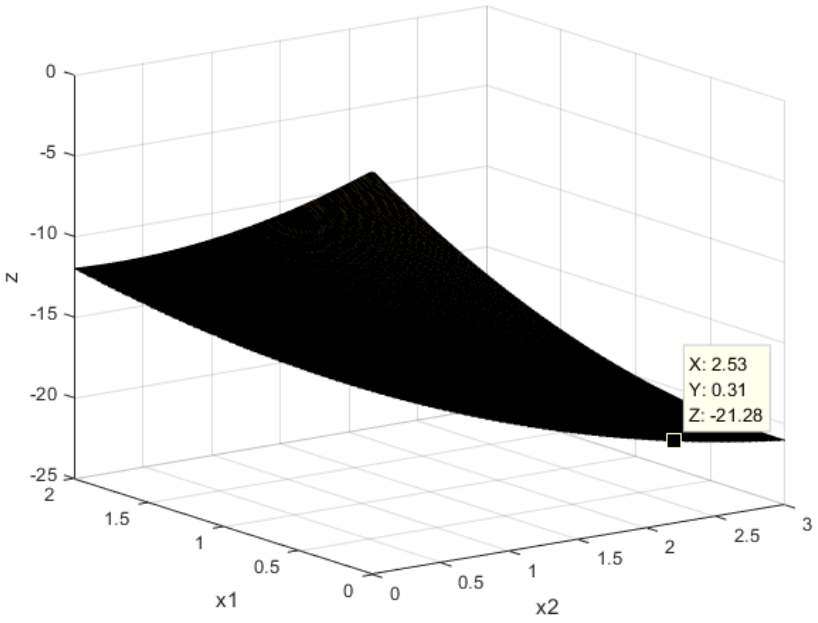
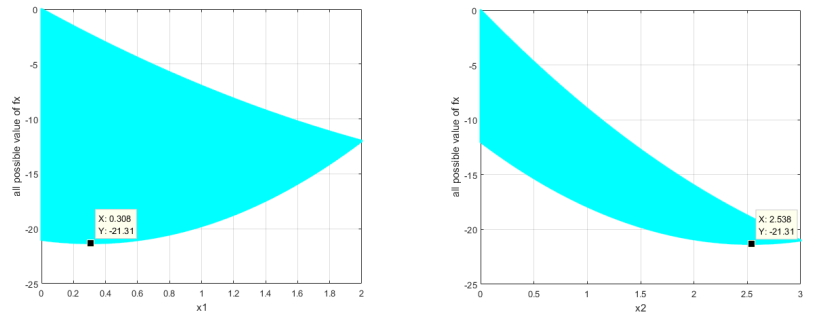


Figure 2: All feasible values of quadratic function



(a) all possible values of z given $x1$ (b) all possible values of z given $x2$

Figure 3: 2-D solutions of qudratic function

```
% ECON90055 Final Project - "Q4-Graphic Solution"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% graph in 3-D figure
clear all;

% accuracy
a = 0.001;
% slice x1
x1 = (0:a:2);
n = length(x1); % number of rows

for i = 1:n
   x2_max = 3 - x1(i)*1.5;
   % slice x2
   x2 = (0:a:x2_max);
   m = length(x2);
   for j = 1:m
      % generate value of z for each feasible
          combination of x1,x2
      z(i,j) = Q4_fx(x1(i),x2(j));
   end
end
z(z == 0) = NaN;
x2 = (0:a:3);
m = length(x2);

% find the minimum value of z. and its corresponding
   x1, x2
z_min = min(min(z));
index_min = find(z == min(min(z)));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the reminder indicates rows of the min - x1.
x1_min_index = rem(index_min,n);
x1_min = x1(x1_min_index);
% find x2 by dividing index_min by number of rows.
x2_min_index = fix(index_min/n)+1;
x2_min = x2(x2_min_index);

figure
% 3-D surface plot, all possible value of fx, given
   each x1,x2 that
% satisfies constraints.
surf(x2,x1,z)
xlabel('x2')
```

```
ylabel('x1')
zlabel('z');

figure
% 2-D plot feasiable value of fx given each value of x1.
plot(x1,z,'c.',x1_min,z_min,'ro')
xlabel('x1')
ylabel('all possible value of z')
grid;

figure
% 2-D plot feasiable value of fx given each value of x2.
plot(x2,z,'c.',x2_min,z_min,'ro')
xlabel('x2')
ylabel('all possible value of z')
grid;
```

(b) Kuhn-Tucker problem

Define the problem with Kuhn Tucker Condition: First rewrite the problem into following structure: $f(x) = x^T Q x + c^T x$ and with constraint $g(x) = b - Ax$.

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},\ Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ c = \begin{bmatrix} -8 \\ -10 \end{bmatrix},\ A = [3\ 2],\ b = 6$$

Then we can get first derivative of the problem and the constraint:

$$\nabla f(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = 2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -8 \\ -10 \end{bmatrix}$$

$$\nabla g(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = - \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

Then there exists a non-negative Lagrange multiplier λ such that:

$$2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} + \begin{bmatrix} -8 \\ -10 \end{bmatrix} + \begin{bmatrix} 3 \\ 2 \end{bmatrix} \lambda = 0$$

$$[3\ 2] \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} = 6$$

$$\lambda \geq 0$$

We can write this system of equations into the following format and let Matlab compute solutions:

$$\begin{bmatrix} 3 & 2 & 0 \\ 2 & 0 & 3 \\ 0 & 2 & 2 \end{bmatrix} \begin{bmatrix} x_1^* \\ x_2^* \\ \lambda \end{bmatrix} = \begin{bmatrix} 6 \\ -8 \\ -10 \end{bmatrix}$$

Matlab provides the solution:

$$\begin{bmatrix} x_1^* \\ x_2^* \\ \lambda \end{bmatrix} = \begin{bmatrix} 0.3077 \\ 2.5385 \\ 2.4615 \end{bmatrix}$$

Listing 12: Q4 part(b) Solving KKT problem

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project - "Q4-Kuhn-Tucker Problem"
%----------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;

% set up parameters for Quadprog
Q = [1,0;0,1];
c = [-8;-10];
A = [3,2];
b = 6;

% solve KKT problem:
x = [ A, 0; 2*Q, transpose(A) ]\[b; -c] ;
disp(x);
```

(c) quadprog function

quadprog defines the quadratic programming problem as:

$$\min_x \frac{1}{2} x^T H x + c^T x \ s.t. \ Ax \leq b$$

State input values for *quadprog* function:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \ H = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \ c = \begin{bmatrix} -8 \\ -10 \end{bmatrix}, \ A = [3 \ 2], \ b = 6$$

Then with code $x = quadprog(H, c, A, b)$ Matlab provides the solution exactly the same as part (b):

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.3077 \\ 2.5385 \end{bmatrix}$$

Listing 13: Q4 part(c) *quadprog*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project - "Q4-Quadprog"
%----------------------------
```

```
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;

% set up parameters for Quadprog
H = [2,0;0,2];
f = [-8;-10];
A = [3,2];
b = 6;

% call function
x = quadprog(H,f,A,b);
disp(x);
```

(d) Monte Carlo simulation

Define solution algorithm for Monte Carlo simulation.
1. Set initial value: $x_1^* = 1$, $x_2^* = 1$ and $z^* = -16$. Also store those values in a copy: $\hat{x_1}$, $\hat{x_2}$ and $\hat{z}$.
2.1 Outer loop (repeat N times): randomly draw a value for x_3. x_3 indicates the slackness of the constraint: $3x_1 + 2x_2 + x_3 \leq 6$.
2.2 Outer loop Start inner loop: (repeat N times):

Inner loop-1: randomly draw a feasible value of x_1 given x_3, where $x_1 \in [0, \frac{6-x_3}{3}]$.

Inner loop-2: Compute corresponding value of x_2 and z given x_1 and x_3: $x_2 = \frac{6-x_3-3x_1}{2}$ and $z = x_1^2 + x_2^2 - 8x_1 - 10x_2$

Inner loop-3: Compare value of z to z^*. If $z < z^*$, overwrite z^* by z, x_1^* by x_1 x_2^* by x_2.
2.3 Outer loop after N times repeats of inner loop, it provides the minimal value of z^* given a randomly chosen slackness level x_3.
2.4 Outer loop compare z^* to $\hat{z}$. If $z^* \leq \hat{z}$, overwrite $\hat{z}$ by z^*. We update global minimal value for $\hat{z}$ given a new value of x_3.
3. After repeating outer loop N times, we should be able to find a value of $\hat{z}$ that is global minimal among all possible value of x_3, and therefore, its corresponding value of x_1 and x_2 should be the solution of the quadratic problem.

Setting repeats $N = 20000$, Matlab provides the solution: (solutions under Monte Carlo Simulation have a little variations every time on its 4th decimal place when I run the function, because the exact solution may not get drawn every time by simulation.)

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.3078 \\ 2.5383 \end{bmatrix}$$

Listing 14: Q4 part(d) Monte Carlo Simulation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project - "Q4-Monte Carlo"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
% intial values
x1_min = 1;
x2_min = 1;
z_min = Q4_fx(x1_min,x2_min);
% create a temporary value
x1_tem_min = x1_min;
x2_tem_min = x2_min;
z_tem_min = z_min;

n = 20000; % how many times repeats

% recreate random draws of x3
x3_draw = rand([1, n]) * 6;

for i=1:n
    % generate a random sequence of x_1 for every x3
     x3 = x3_draw(i);
     x1_picks = rand([1,n]) * (6-x3)/3;

    % find minimal z for each given x3
for j = 1:n
    x1 = x1_picks(j);
    x2 = (6-x3-x1*3)/2;
    z = Q4_fx(x1,x2);
    % update minimal z for the given x3
    if z <z_tem_min
        x1_tem_min = x1;
        x2_tem_min = x2;
        z_tem_min = z;
    end
end
    % update minimal z among all possible value of x3
    if z_tem_min < z_min
        x1_min = x1_tem_min;
        x2_min = x2_tem_min;
        z_min = z_tem_min;
    end
end

```

```
% print solution
disp(x1_min);
disp(x2_min);
```

Problem 5: Numerical Integration

Problem set up: we want to estimate the expected value of X, such that X is a random variable from $B(a, b)$, and has Beta density function $f(x)$. Noting that computing the expectation of a function $k(x)$ of a random variable X with probability density function $f(\cdot)$ has the formula:

$$E[k(X)] = \int k(x)f(x)\ dx$$

The expected value of X in this case can be written in integral expression as following:

$$E[X] = \int_0^1 xf(x)\ dx$$

Substituting Beta density function:

$$E[X] = \int_0^1 x\frac{1}{B(a,b)}x^{a-1}(1-x)^{b-1}\ dx$$

Simplify and obtain the integration problem that we want to solve:

$$E[X] = \int_0^1 \frac{1}{B(a,b)}x^{a}(1-x)^{b-1}\ dx = \int_0^1 g(x)dx$$

(a) stochastic integration

From the theory of Weak Law of Large Numbers(WLLN), we know that the mean of sampling distribution by simple random sampling is consistent with its population mean. Set up the algorithm that follows this theorem:

Inner loop: (iterate over 1000 times)
1. randomly draw a point(observation) of x between the interval $[0, 1]$ by Monte Carlo simulation. (Drawing from interval $[0, 1]$ because the integral of function $g(x)$ we want to solve is defined between $[0, 1]$.)
2. compute the value of $g(x)$ for the randomly drawn x.

Outter loop: (iterate over 2000 times)
3. after the completion of inner loop, should have obtained a row vector that is a 'sample' which contains 1000 different values of $g(x)$.
4. take the mean of the sample to obtain the sample mean, and then store the sample mean.

After completion of outter loop:
5. should have obtained a distribution of sample means. (i.e. 2000 sample means)
6. (applying WLLN) take the mean of 2000 stored sample means to get approximated consistent population mean.

The stochastic integration function **q5_monte_carlo**$(2, 3)$ produces the solution that $E(X) = 0.4000$. However, due to its stochastic nature, the result sometimes varies a little on its fourth decimal place. Errors can be reduced by increasing the scale of sample size, which is a trade-off between accuracy and time consumption.

(b) 2-point Gaussian quadrature

To apply Gauss-Legendre quadrature rules, it is needed to, firstly, transform the interval $[0, 1]$ into $[-1, 1]$, which has the form:

$$\int_j^k g(x)dx = \frac{k-j}{2}\int_{-1}^{1} g(\frac{k-j}{2}x + \frac{k+j}{2})dx$$

since the value of j and k are fixed in the integral problem $\int_0^1 g(x)dx$, so substitute $j = 0$ and $k = 1$ to obtain the transformation:

$$\int_0^1 g(x)dx = \frac{1}{2}\int_{-1}^{1} g(\frac{1}{2}x + \frac{1}{2})dx$$

then the problem can be written into the two-point Gaussian quadrature:

$$E(X) = \frac{1}{2}\int_{-1}^{1} g(\frac{1}{2}x + \frac{1}{2})dx \approx \frac{1}{2}\sum_{i=1}^{2} w_i g(\frac{1}{2}x_i + \frac{1}{2})$$

We are looking for two nodes x_1 and x_2 and two weights w_1 and w_2 to approximate the integral, fortunately, there are formulas that help to find those values. x are the root of Legendre polynomial equation $P_2(x) = 0$:

$$P_2(x) = \frac{3}{2}x^2 - \frac{1}{2} = 0$$

solve for $P_2(x) = 0$, obtain:

$$x_1 = \sqrt{\frac{1}{3}}, x_2 = -\sqrt{\frac{1}{3}}$$

w_j can be solved by the following formula:

$$w_i = \frac{2}{P_2'(x_i)^2(1-x_i^2)} = \frac{2}{(3x_i)^2(1-x_i^2)} \quad \text{where } i = 1, 2$$

therefore get:

$$w_1 = w_2 = 1$$

Now, we can set up a function that given any value of a and b, return the expected value of X:

1. set up the nodes and weights under Gauss-Legendre quadrature $x_1 = \sqrt{\frac{1}{3}}$, $x_2 = -\sqrt{\frac{1}{3}}$, $w_1 = w_2 = 1$.
2. find values of $g(\frac{1}{2}x_i + \frac{1}{2})$ for $i = 1, 2$, where $g(\cdot) = \frac{1}{B(a,b)}x^a(1-x)^{b-1}$.
3. find $E(X)$ by calculating $\frac{1}{2}\sum_{i=1}^{2} w_i g(\frac{1}{2}x_i + \frac{1}{2})$.

Under the case when $a = 2$, $b = 3$, using function **q5_GLQ_2pt**$(2, 3)$, it provides the solution that $E(X) = 0.3333$. The result has considerably large error towards the correct solution $\frac{a}{a+b} = \frac{2}{5}$, which is possibly due to the measurement error. Fortunately, the function provides good approximation for other values of a and b.

(c) 3-point Gaussian quadrature

3-point Gaussian quadrature follows the same logic as 2-point Gaussian quadrature, while it has different nodes and weights from the two-point case.
Firstly, find the x_i that solves $P_3(x) = 0$:

$$P_3(x) = \frac{5}{2}x^3 - \frac{3}{2}x = 0$$

Rearrange to obtain:

$$x(\frac{5}{2}x^2 - \frac{3}{2}) = 0$$

Solve the equation, we get three solutions of x:

$$x_1 = 0,\ x_2 = \sqrt{\frac{3}{5}} \text{ and } x_3 = -\sqrt{\frac{3}{5}}$$

Next to find the weight of each point:

$$w_j = \frac{2}{P_3'(x_j)^2(1-x_j^2)} \text{ where } j = 1, 2, 3$$

Substitute each x_j into w_j:

$$w_j = \frac{2}{(\frac{15}{2}x^2 - \frac{3}{2})^2(1-x_j^2)}$$

obtain:

$$w_1 = \frac{8}{9}, w_2 = \frac{5}{9} \text{ and } w_3 = \frac{5}{9}$$

Given any value of a and b, the function **q5_GLQ_3pt**(a, b) follows exactly the same algorithm as 2-point case, but now has three nodes and three weights to

compute the approximation. Entering $a = 2$ and $b = 3$ into the function, it produces the exact solution $E(X) = 0.4$. This function also works well for other values of a and b.

The code for stochastic integration, 2-point Gaussian quadrature and 3-point Gaussian quadrature are given below, it also includes a main script that calls all three functions to compute expected value of X, and also the function $g(x)$.

Listing 15: Q5 main script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project -"Q5- main script"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;

% decide values for a and b for beta density function
a = 2;
b = 3;

% find expected X using three different methods:
int2 = q5_GLQ_2pt(a,b);
int3 = q5_GLQ_3pt(a,b);
int_mci = q5_monte_carlo(a,b);
% print results
disp(int2);
disp(int3);
disp(int_mci);
```

Listing 16: Q5 stochastic integration

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project -"Q5-MonteCarlo Integration"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function exp_x=q5_monte_carlo(a,b)

% number of samples
repli = 2000;
exp_x_dis = zeros(1,repli);

for i = 1:repli

    % number of obversations in sample
    n = 1000;
    % randomly draw x from (0,1)
```

```
    y_val = zeros(1,n);

    for j = 1: n

        % draw random x between (0,1)
        x = rand(1);
        % find its corresponding value
        y_val(j) = q5_gx(x,a,b);

    end

    % store each sample mean
    exp_x_dis(i) = mean(y_val);

end
    % mean of sample mean
    exp_x = mean(exp_x_dis);
end
```

Listing 17: Q5 two-point Gaussian Quadrature

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project -"Q5- 2-pt Gauss-quadrature"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function area = q5_GLQ_2pt(a,b)

% integral lower and upper bound, in this case, always
    0,1
    low = 0;
    up = 1;

% predefine space
    n = 2;
    u = zeros(1,n);
    x = zeros(1,n);
    w = zeros(1,n);
    fu = zeros(1,n);

% value of x and weight already determined under
   GaussCLegendre quadrature
% given point number N=2
    x(1) = sqrt(1/3);
    x(2) = -x(1);
    w(1) = 1;
```

```
    w(2) = 1;

% calculate approximated integration using two point GLQ
    for i=1:2
        % finding f(u) for each x
        u(i) = ((up-low)/2)*x(i)+(low+up)/2;
        fu(i) = q5_gx(u(i),a,b);
    end

% final result of two point approximation
    area = ( sum(w.*fu)  )*(up-low)/2;
end
```

Listing 18: Q5 three-point Gaussian Quadrature

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project -"Q5- 3-pt Gauss-quadrature"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function area = q5_GLQ_3pt(a,b)

% integral lower and upper bound, in this case, always 0,1
    low = 0;
    up = 1;

% predefine space
    n = 3;
    u = zeros(1,n);
    x = zeros(1,n);
    fu = zeros(1,n);

% value of x and weight already determined under GaussCLegendre quadrature
% given point number N=2
    x(1) = 0;
    x(2) = sqrt(3/5);
    x(3) = -x(2);
    w(1) = 8/9;
    w(2) = 5/9;
    w(3) = 5/9;

% calculate approximated integration using two point GLQ
```

```
    for i=1:3
        % finding f(u) for each x
        u(i) = ((up-low)/2)*x(i)+(low+up)/2;
        fu(i) = q5_gx(u(i),a,b);
    end

% final result of two point approximation
    area = ( sum(w.*fu) )*(up-low)/2;
end
```

Listing 19: Q5 $g(x)$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ECON90055 Final Project -"Q5- g(x)"
%---------------------------
% Coder: Zelin Chen (797036)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=q5_gx(x,a,b)

    y = (1/beta(a,b))*(x^a)*(1-x)^(b-1);

end
```