**XJTLU Entrepreneur College (Taicang) Cover Sheet**

|  |  |  |
| --- | --- | --- |
| Module code and Title | **DTS104TC Numerical Methods** | |
| School Title | **School of Artificial Intelligence and Advanced Computing** | |
| Assignment Title | **Assignment 1** | |
| Submission Deadline | **June 10, 2021. 5pm (GMT+8)** | |
| Final Word Count | **-** | |
| If you agree to let the university use your work anonymously for teaching and learning purposes, please type **“yes”** here. | | **yes** |

I certify that I have read and understood the University’s Policy for dealing with Plagiarism, Collusion and the Fabrication of Data (available on Learning Mall Online). With reference to this policy I certify that:

* My work does not contain any instances of plagiarism and/or collusion.  
  My work does not contain any fabricated data.

**By uploading my assignment onto Learning Mall Online, I formally declare that all of the above information is true to the best of my knowledge and belief.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Scoring – For Tutor Use** | | | | | | |
| **Student ID** | | | | **1930080** | | |
|  | | | | | | |
| **Stage of Marking** | | **Marker**  **Code** | **Learning Outcomes Achieved （F/P/M/D）**  **(please modify as appropriate)** | | | **Final**  **Score** |
| **A** | **B** | **C** |
| 1st Marker – red pen | |  |  |  |  |  |
| Moderation  – green pen | | **IM**  **Initials** | The original mark has been accepted by the moderator (please circle as appropriate): | | | Y / N |
|  | Data entry and score calculation have been checked by another tutor (please circle): | | | Y |
| 2nd Marker if needed – green pen | |  |  |  |  |  |
| **For Academic Office Use** | | | **Possible Academic Infringement (please tick as appropriate)** | | | |
| **Date**  **Received** | **Days late** | **Late Penalty** | **☐ Category A** | | Total Academic Infringement Penalty (A,B, C, D, E, Please modify where necessary) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
|  |  |  | **☐ Category B** | |
| **☐ Category C** | |
| **☐ Category D** | |
| **☐ Category E** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | **EXAMINER** | **DEPARTMENT** | **Email** |
| DTS104TC | Long Huang | School of Intelligent Manufacturing Ecosystem | Long.Huang@xjtlu.edu.cn |

**2nd SEMESTER 2020/21 Assignment**

**Undergraduate – Year 2**

**DTS104TC Numerical Methods**

**Submission Deadline: June 10, 2021. 5pm (GMT+8)**

|  |  |
| --- | --- |
| **Student Name** | **Student ID** |
| Yaqi Yu | 1930080 |
| **DEPARTMENT** | **Email** |
| School of AI and Advanced Computing | Yaqi.Yu19@student.xjtlu.edu.cn |

# Assignment 1

## Question - 1

### (a)

The solution of the function on the interval is solved by bisection method, which is accurate to two places after the decimal point. First of all, determine how many bisection are needed to achieve the desired accuracy.

According to the formula:

The error after n times is:

If the error is less than: , the solution is accurate to 2 places after the decimal point

The function needs to satisfy:

Therefore, the iteration of steps is required.

The following table will verify this.

### (b)

|  |  |  |  |
| --- | --- | --- | --- |
| y | 8 | 10 | 13 |
| x (4 decimal places) | 299.9371 | 288.5309 | 277.4566 |

------------------------------------------------------------------------------------------

%start of matlab code for Question 1(b)

function AnswerOne

% This is question1 main program

f1= @(x) (-139.34411) + (1.575701\*10^5)/x-(6.642308\*10^7)/x^2+(1.2438\*10^10)/x^3-8.621949\*10^11/x^4 - log(8);

f2= @(x) (-139.34411) + (1.575701\*10^5)/x-(6.642308\*10^7)/x^2+(1.2438\*10^10)/x^3-8.621949\*10^11/x^4- log(10);

f3= @(x) (-139.34411) + (1.575701\*10^5)/x-(6.642308\*10^7)/x^2+(1.2438\*10^10)/x^3-8.621949\*10^11/x^4- log(13);

disp('Case1: y=8');

AnswerOne\_function(273.15,313.15,0.01,f1); %solve x for the y = 8, and absolute error = 0.01

disp('Case2: y=10');

AnswerOne\_function(273.15,313.15,0.01,f2); %solve x for the y = 10, and absolute error = 0.01

disp('Case3: y=13');

AnswerOne\_function(273.15,313.15,0.01,f3); %solve x for the y = 13, and absolute error = 0.01

%assignment2-1.1 bisection method

%compute f(x) = 0's each approximation of root, Iteration, errorlative eor

%input: initial guesses: a,b;

% tolerance:tol;

% Function handle: fun

%output: ap\_root: approximation of root matrix:

% gap: Interval length:

% fx: each approximation of root function value: f(ap\_root)

% count: number of iterations

% error: relative of error

function [ap\_root,gap,fx,count,error]=AnswerOne\_function(a,b,tol,fun)

e=tol+1; %set initial errorlative error

count=-1; %set initial number of iterations

ap\_root=[]; %ap\_root vector store approximation of root

gap=[]; %gap vector store interval length

error=[]; %error vector store relative error

fx=[]; %fx vector store function value

while(e>tol)

count=count+1;

c=(a+b)/2;

x=c;

ap\_root=[ap\_root;x]; %Store the iterative value in the ap\_root matrix

fc=feval(fun,x); %compute function value of ap\_root

fx=[fx;fc]; %store the function value in the fx matrix

x=a;

e=abs(b-a); %compute relative error

%Determine which area the root is in

if(fc\*feval(fun,x)<0)

b=c;

else

a=c;

end

dis=abs(a-b); %compute length of interval

gap=[gap;dis];

error=[error;e];

end

%display result in matrix

disp(' Iteration Approximation of root length of interval f(root) relative error ')

for i=1:count

fprintf('%2d %20.4f %25.4f %20.4f %20.4f \n ',i,ap\_root(i),gap(i),fx(i),error(i))

end

end

end

%end of matlab code for Question 1(b)

------------------------------------------------------------------------------------------

## Question - 2

|  |  |
| --- | --- |
|  | Value (4 decimal places) |
| x1 | 0.9997 |
| x2 | 0.9998 |
| x3 | 1.0000 |
| x4 | 1.0001 |
| x5 | 1.0002 |
| x6 | 1.0001 |
| Number of Iterations | 9 |

------------------------------------------------------------------------------------------

%start of matlab code for Question 2

function AnswerTwo

% This is question 2 main program

A = [3 -1 0 0 0 1/2;-1 3 -1 0 1/2 0;0 -1 3 -1 0 0; 0 0 -1 3 -1 0; 0 1/2 0 -1 3 -1;1/2 0 0 0 -1 3]; %the coefficient matrix

B=[5/2;3/2;1;1;3/2;5/2]; %the constant term on the right side matrix

X0=[0;0;0;0;0;0]; %initial vector of iteration

eps = 1e-3; %tolerance

AnswerTwo\_Function(A, B, X0, 500, 1e-3);% call answer two function

function [X\_reality,n\_reality ] = AnswerTwo\_Function( A,b,X\_start,n\_limit,tol)

%%The function input variables:

% A is the iterative coefficient matrix.

% b is the constant term on the right side of the equation group (column vector)

% X\_start is the initial vector of iteration

% n\_limit is the maximum number of iterations allowed

% tol is the limit of tolerance.

%%The function output variables:

% X\_reality is function running result

% n\_reality is final iteration numbers

[m,n] = size(A); % A's row number = m£¬column number = n

D\_L = tril(A); % The lower triangular matrix of A is obtained.

B = inv(D\_L) \* (D\_L - A); % B is the Gauss-Seidel iterative matrix

% the coefficient matrix of the simplified equivalent system of equations

% which is convenient for iteration.

f = inv(D\_L) \* b; % f is the constant vector of the simplified

n\_reality = 0;

%Check whether the coefficient matrix is a square matrix

if m ~= n

error('A is not a square matrix');

end

%Check the coefficient matrix is diagonally dominant

check = abs(A);

diaLine = diag(check);

res=1;

for i = 1:m

if diaLine(i)<sum(check(i,:))-diaLine(i)

disp('The coefficient matrix is not diagonally dominant')

break;

end

end

disp('The coefficient matrix A is diagonally dominant')

while 1

if(n\_reality > n\_limit)

disp('The number of iterations exceeds the given maximum number of times');

break;

end

X\_reality = B \* X\_start + f; % Gauss-Seidel iteration formula

n\_reality = n\_reality + 1; % Actual number of iterations

if(norm(X\_reality - X\_start) <= tol) % if meet||X(k+1) - X(k)|| 2 norm <= tol

break; % exit the function

else

X\_start = X\_reality; % iteration

end

end

disp('The number of Gauss Seidel iterations is:');

disp(n\_reality);

disp('The iteration x value is:');

disp(X\_reality);

end

end

%end of matlab code for Question 2

------------------------------------------------------------------------------------------

### Question -- 3

#### Instrucation

Power Iteration Method for approximating the dominant Eigenvalues and Eigenvectors of a Matrix

Dominant eigenvalues and eigenvectors

* is the dominant eigenvalue of if for all
* The corresponding eigenvector is also called dominant

So dominant is the highest eigenvalue of matrix A

using power iteration method finding the eigenvalue:

suppose is an eigenvector of , then its eigenvalue is

s called Rayleigh Quotent

Given the eigenvector approximation, the Rayleigh quotient is the optimal approximation of the eigenvalue. In the power iteration, the eigenvalue approximation can be obtained by using the Rayleigh quotient for the normalized eigenvector.

|  |  |
| --- | --- |
| Iteration 1 | |
| Estimated Eigenvector (normalized) | Estimated Eigenvalue |
| 0.5698 | 20.0000 |
| 0.4558 |
| 0.6838 |
| Iteration 2 | |
| Estimated Eigenvector (normalized) | Estimated Eigenvalue |
| 0.5658 | 20.5455 |
| 0.4548 |
| 0.6878 |
| Iteration 3 | |
| Estimated Eigenvector (normalized) | Estimated Eigenvalue |
| 0.5669 | 20.5460 |
| 0.4541 |
| 0.6873 |
| Iteration 4 | |
| Estimated Eigenvector (normalized) | Estimated Eigenvalue |
| 0.5665 | 20.5461 |
| 0.4543 |
| 0.6875 |

b)

In the matlab program, the result of using eig () function is compared with the final result obtained by power method, and it is found that if all the four decimal places are equal, so the solution of power method is convergent.

------------------------------------------------------------------------------------------

%start of matlab code for Question 3

function AnswerThree

%This is question1 main function

A=[2 8 10;8 3 5;10 5 9]; %Needed solved matrix

u= [1;1;1]; % initial eigenvector

it=4;%iteration numbers;

[l,~]=AnswerThree\_Function(A,u,it);

%Check power method answer and eig()function eigenvalue;

c=eig(A); %eig function obtain a column eigenvalue vector

l=roundn(l,-4); %keep 4 decimal places

c=roundn(max(c),-4); %keep 4 decimal places

if l==c

fprintf('\n power method obtain eigenvalue equal to using eig() function obtained');

else

fprintf('\n power method obtain eigenvalue do not equal to using eig() function obtained');

end

%Power method iteration function

%Calculate the dominant(highest) eigenvalues

%and corresponding eigenvectors of each iteration

%Input:A: Matrix A

% x: Initial(non-zero) vector

% k: Iteration numbers

%Output: lam: the dominant(highest) eignvalues

% u: corresponding eigenvectors

%Other variables:u\_norm: normalized Eigenvectors

function [lam,u]=AnswerThree\_Function(A,x,k)

for j = 1:k

u=x/norm(x); %vector normalization

x=A\*u; %Power Steps

lam=u'\*x; %Rayleigh Quotient

%Raleigh Quotient is optimal approximation of eigenvalues

fprintf('\n\nThe iteration %d:',j);

fprintf('\nThe Estimated Eigenvalue(dominant/highest) is %5.4f\n',lam);

disp('The corresponding Estimated Eigenvector(normalized) is:');

u\_norm = x/norm(x); %normalized eigenvectors

fprintf('\n%5.4f',u\_norm);

end

end

end

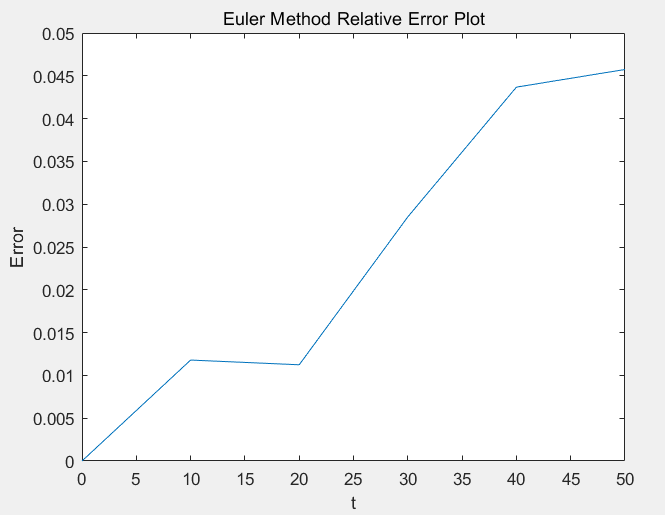
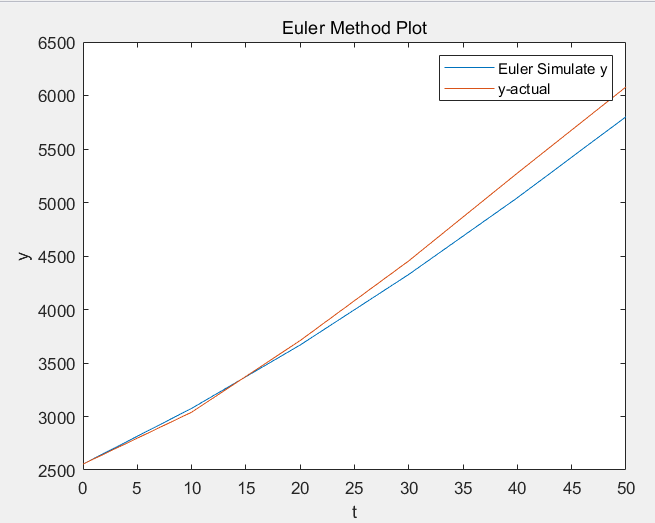
%end of matlab code for Question 3

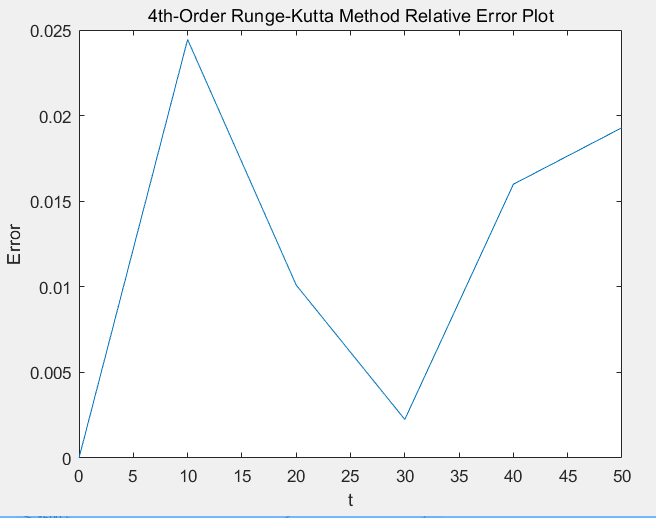
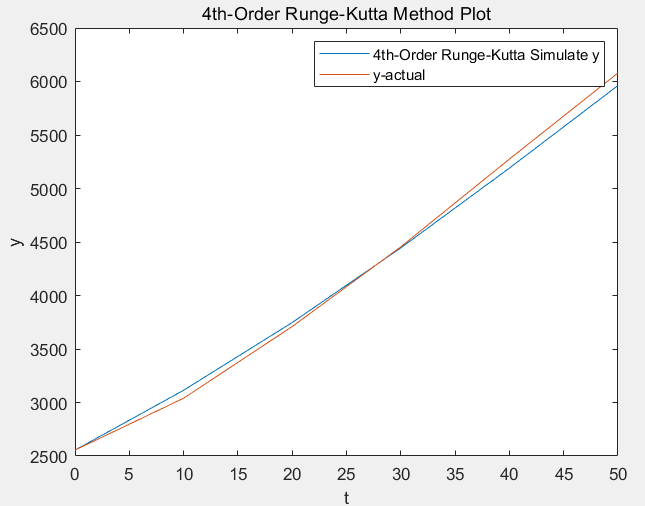
------------------------------------------------------------------------------------------

### Question -- 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data | | Euler | | 4th-Order Runge-Kutta | |
| t | y-actual | Estimation (4 significant figures) | True Error (%) with 2 decimal places | Estimation (4 significant figures) | True Error (%) with 2 decimal places |
| 0 | 2555 | 2555 | 0.00% | 2555 | 0.00% |
| 10 | 3040 | 3076 | 1.18% | 3114 | 2.44% |
| 20 | 3710 | 3668 | 1.12% | 3747 | 1.01% |
| 30 | 4455 | 4328 | 2.85% | 4445 | 0.23% |
| 40 | 5275 | 5045 | 4.37% | 5191 | 1.60% |
| 50 | 6080 | 5802 | 4.57% | 5963 | 1.93% |

The following is Euler and 4th-Order Runge-Kutta Method based on Table-4a Fiugre and its relative error figure using matlab.





------------------------------------------------------------------------------------------

%start of matlab code for Question 4

function AnswerFour

%Main Program

t0=0; %initial t value

y0=2555; %initial y value

st=10; %step size = 10

it=5; %number of iterations

y1=[2555 3040 3710 4455 5275 6080]; %Actual y values

Euler(t0,y0,st,it,y1); %using Euler Method simulate

Runge\_Kutta(t0,y0,st,it,y1); %using 4th-Order Runge-Kutta Method simulate

function Euler(t,y,h,n,y\_actual)

function z=f(t,y)

%Needed to solve the growth of y over time(t)

z=0.0259\*(1-y/(12000))\*y;

end

%Euler Method simulation

%Input: t:Initial condition time t;

% y:Initial condition y e.g:y\_0(t=0)=2555

% h:Step size e.g. 0.1,10,20

% n:Number of iterations

% y\_actual: actual y values

%Euler Method

t\_array1=[t]; %initial time t array to store t

y\_array1=[y]; %initial y array to store y

e\_array1=[0];

fprintf('Euler Method: t=%d, y=%4.0f\n',t,y);

for i=1:n

t=(i-1)\*h;

K=f(t,y);

y=y+h\*K; %using Euler Method to simulate y

e1=abs(y-y\_actual(i+1))/y\_actual(i+1); %Compute relative error

e\_array1=[e\_array1;e1];

fprintf('Euler Method: t=%d, y=%4.0f, t=%5.2f%%\n ',t+h,y,e1\*100);

t\_array1=[t\_array1;t+h]; %store each t

y\_array1=[y\_array1;y]; %store each y

end

%Plot y actual value and using euler method y

figure;

plot(t\_array1,y\_array1,t\_array1,y\_actual); %Plot y actual value and using euler method y

title('Euler Method Plot');

xlabel('t');

ylabel('y');

legend('Euler Simulate y','y-actual');

figure;

plot(t\_array1,e\_array1);

title('Euler Method Relative Error Plot'); %Plot relative error

xlabel('t');

ylabel('Error');

end

function Runge\_Kutta(t,y,h,n,y\_actual)

%Needed to solve the growth of y over time(t)

function z=f(t,y)

z=0.0259\*(1-y/(12000))\*y;

end

%4th-Order Runge-Kutta simulation

%Input: t:Initial condition time t;

% y:Initial condition y e.g:y\_0(t=0)=2555

% h:Step size e.g. 0.1,10,20

% n:Number of iterations

% y\_actual: actual y values

t\_array2=[t]; %initial time t array to store t

y\_array2=[y]; %initial y array to store y

e\_array2=[0]; %initial e array to store error

fprintf('4th-Order Runge-Kutta Method: t=%d, y=%4.0f\n',0,2555);

for i=1:n

t=(i-1)\*h;

K1=f(t,y);

K2=f(t+h/2,y+(h/2)\*K1);

K3=f(t+h/2,y+(h/2)\*K2);

K4=f(t+h,y+h\*K3);

y=y+(h/6)\*(K1+2\*K2+2\*K3+K4); %using 4th-Order Runge-Kutta method to simulate y

e2=abs(y-y\_actual(i+1))/y\_actual(i+1); %compute relative error

fprintf('4th-Order Runge-Kutta Method: t=%d, y=%4.0f,error=%5.2f%%\n',t+h,y,e2\*100);

e\_array2=[e\_array2;e2];

t\_array2=[t\_array2;t+h];

y\_array2=[y\_array2;y];

end

%Plot y actual value and using 4th-Order Runge-Kutta method y

figure;

plot(t\_array2,y\_array2,t\_array2,y\_actual);

title('4th-Order Runge-Kutta Method Plot');

xlabel('t');

ylabel('y');

legend('4th-Order Runge-Kutta Simulate y','y-actual');

figure;

plot(t\_array2,e\_array2); %Plot relative error

title('4th-Order Runge-Kutta Method Relative Error Plot');

xlabel('t');

ylabel('Error');

end

end

%end of matlab code for Question 4

------------------------------------------------------------------------------------------

### Question -- 5

|  |  |
| --- | --- |
| n | Approximation (9 decimal places) |
| 3 | 0.400020482 |
| 4 | 0.397441959 |
| 5 | 0.397613249 |

b) The actual integral is 0.397610357, require 397 by the Trapezoidal rule to attain better than using Gaussian Quadrature(n=5)

------------------------------------------------------------------------------------------

%start of matlab code for Question 5

function AnswerFive

%Question5 (a)

f=@(x)exp((-0.5)\*x^2);

Gaussian\_3(f,1,4);

Gaussian\_4(f,1,4);

Gaussian\_5(f,1,4);

%Question5(b)

CompareGT(f,1,4,0.397610357,0.397613249);

%Gaussian\_3/4/5 are almost same

%input:f:given function

% lower:Lower limit of integral

% upper:upper limit of integral

% Final output is calculate the approximate solution of equation f

% using Gaussian Quadrature Method

function Gaussian\_3(f,lower,upper)

c=[0.5555556 0.8888889 0.5555556];%weighting factors

x=[-0.774596669 0.0 0.774596669];%Funtion Arguments

I=0;

a=lower;

b=upper;

z=[];

%using Gaussian Quadrature calculate given function

for i = 1:3

zo=(b-a)/(2.0)\*x(i)+(a+b)/(2.0);

z=[z;zo];

end

for i = 1:3

I=I+(b-a)/(2.0)\*c(i)\*f(z(i));

end

%Formatted output to retain nine decimal places

fprintf('using Gaussian Quadrature Method with n=3 result is:%5.9f',I);

fprintf('\n');

end

function Gaussian\_4(f,lower,upper)

c=[0.3478548 0.6521452 0.6521452 0.3478548];

x=[-0.861136312 -0.339981044 0.339981044 0.861136312];

I=0;

a=lower;

b=upper;

co=[];

for i = 1:4

zo=(b-a)/(2.0)\*x(i)+(a+b)/(2.0);

co=[co;zo];

end

for i = 1:4

I=I+(b-a)/(2.0)\*c(i)\*f(co(i));

end

fprintf('using Gaussian Quadrature Method with n=4 result is:%5.9f',I);

fprintf('\n');

end

function Gaussian\_5(f,lower,upper)

c=[0.2369269 0.4786287 0.5688889 0.4786287 0.2369269];

x=[-0.906179846 -0.538469310 0.0 0.538469310 0.906179846];

I=0;

a=lower;

b=upper;

co=[];

for i = 1:5

zo=(b-a)/(2.0)\*x(i)+(a+b)/(2.0);

co=[co;zo];

end

for i = 1:5

I=I+(b-a)/(2.0)\*c(i)\*f(co(i));

end

fprintf('Using Gaussian Quadrature Method with n=5 result is:%5.9f',I);

fprintf('\n');

end

%Question5 b function compare Gaussian Quadrature(N=5) and Trapezoidal rule

%input:f:given function

% lower:Lower limit of integral

% upper:upper limit of integral

% trueval:Given equation actual integral

% Final output is the number of iteration of compare

% Trapezoidal rule better than Gaussian Quadrature(n=5)

function CompareGT(f,lower,upper,trueval,Gval)

Gerror=abs(Gval-trueval); %Trapezoidal rule accuracy

n=1;

Terror=1;

while(Terror>=Gerror)

h=(upper-lower)/n;

x=lower;

y=f(x);

for i = 1 : n-1

x=x+h; %Trapezoidal rule iteration

y=y+2\*f(x);

end

y=y+f(upper);

I=(upper-lower)\*y/(2\*n);

Terror=abs(I-trueval);

n=n+1;

end

fprintf('The actual integral is %5.9f, require %d by the Trapezoidal rule to attain better than using Gaussian Quadrature(n=5)', trueval,n-1);

end

end

%end of matlab code for Question 5

------------------------------------------------------------------------------------------