

UWTSD

MSc

Task 1

Module code ACHA7002

Student number 2204780

Date

Contents

1	Plotting and differentiation.....	1
2	Trapezoidal Method.....	5
3	Composite	5
4	Simpsons.....	5
5	Integration results tables.....	7
5.1	Composite convergence iterations	7
5.2	Simpsons Convergence iterations.....	8
6	Appendices.....	10
6.1	Your code script and functions.....	10

1 Plotting and differentiation

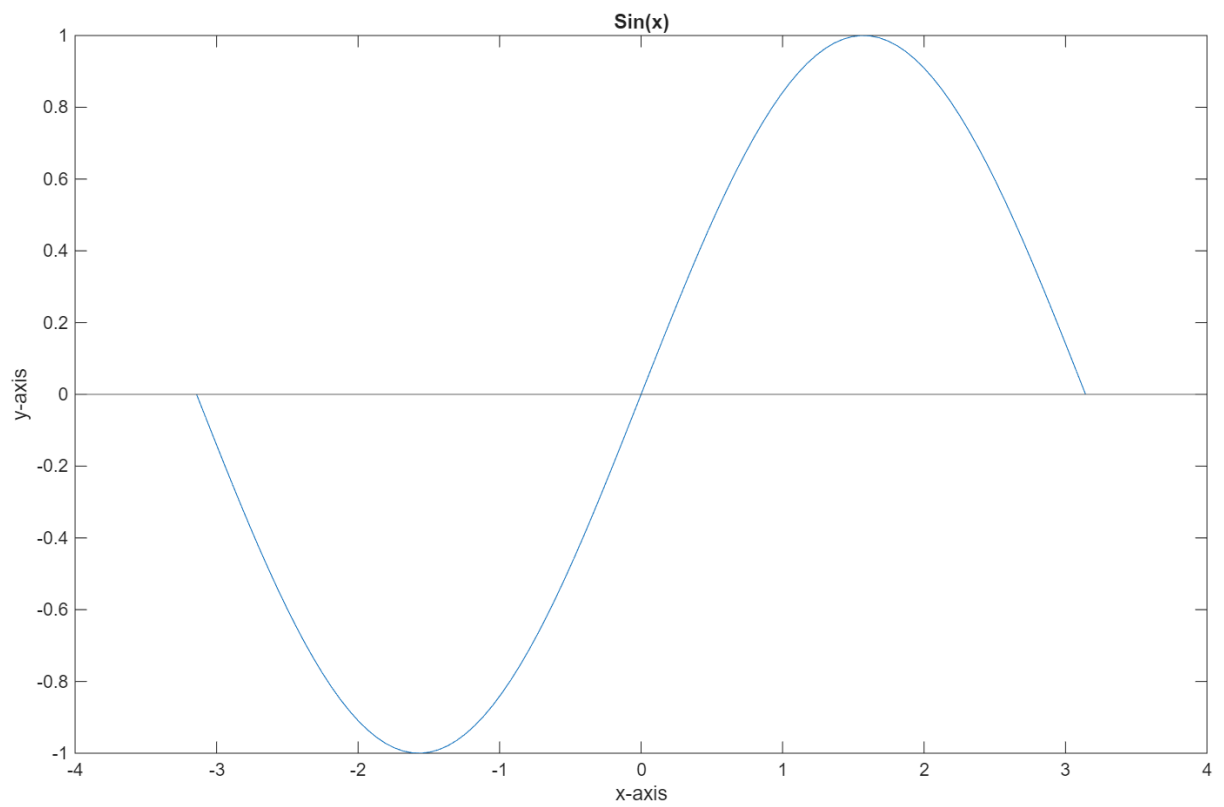


Figure 1 $y = \sin(x)$ plot

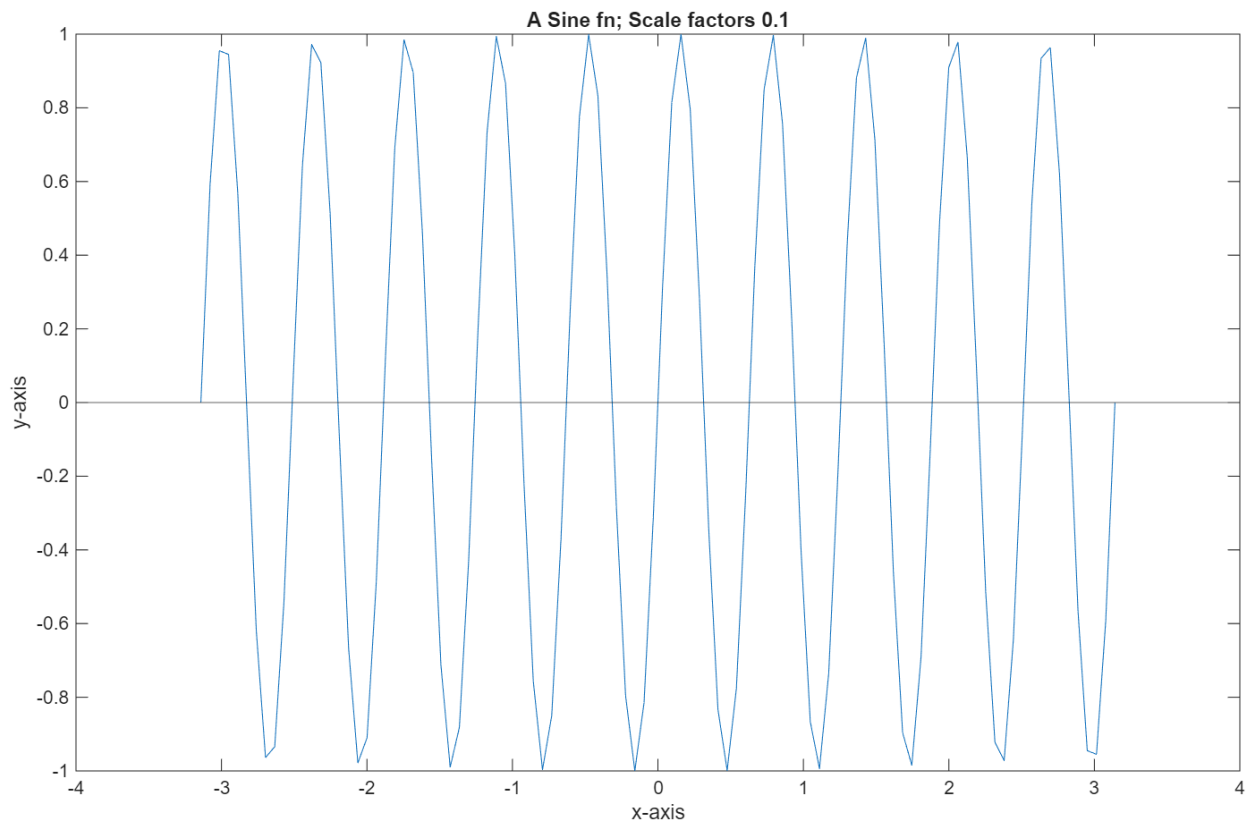


Figure 2 $y = \sin(x/0.1)$ plot

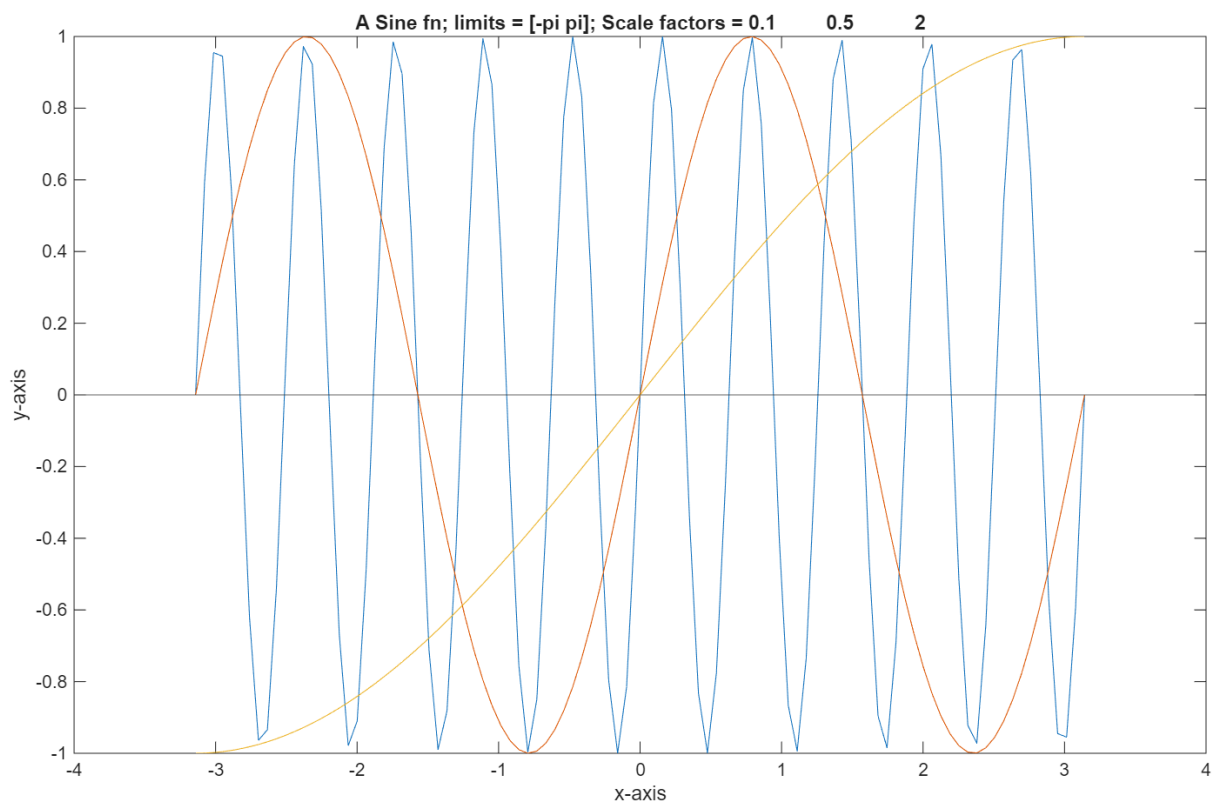


Figure 2 $y = \sin_scale(x, sf)$ plot ([0.1 0.5 2] steps)

Blue line is $y=\sin(x/0.1)$

Orange Line is $y=\sin(x/0.5)$

Yellow Line is $y=\sin(x/2)$

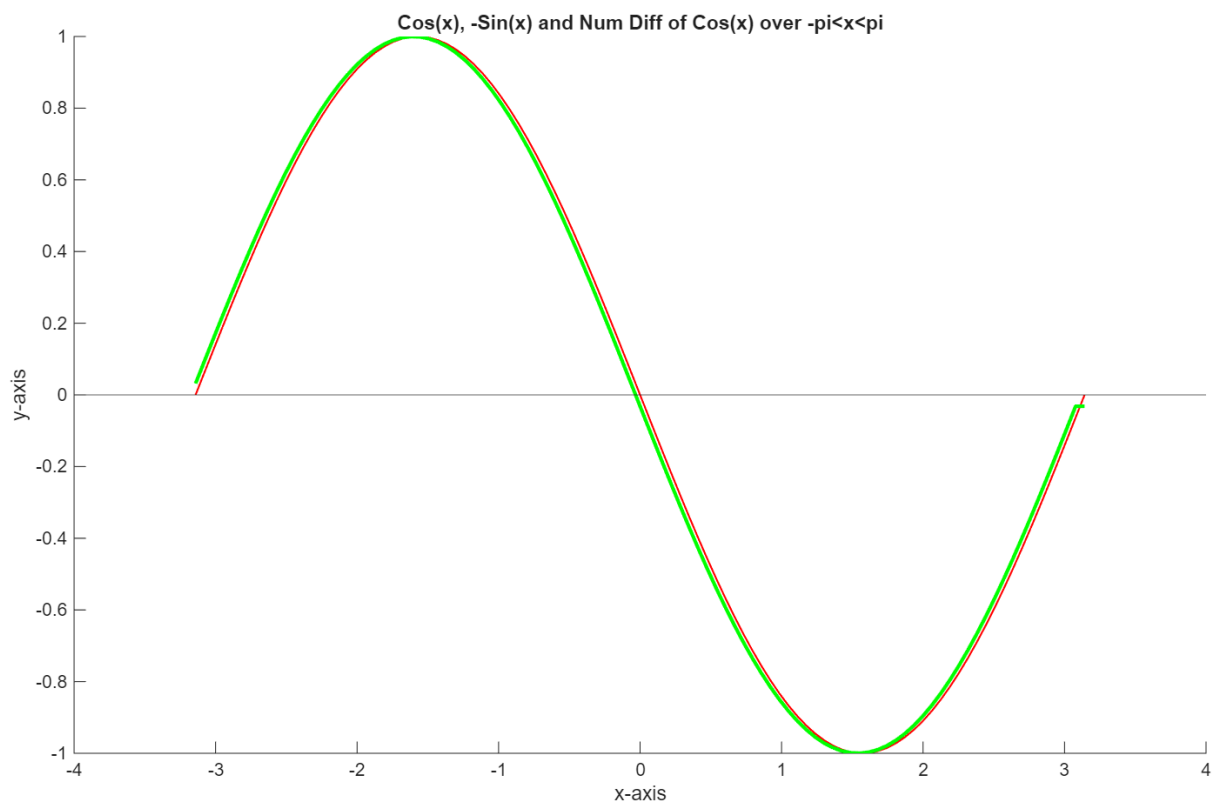


Figure 3 diff cos (x) to -sin(x)

Green line is $y=\cos(x)$

Red line is $y=-\sin(x)$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h_i}$$

Equation 1-1 numerical differentiation

2 Trapezoidal Method

$$I \approx \frac{1}{2}(f_a + f_b)(b - a)$$

Equation 2-1 trapezoidal method

A function was created to integrate 'x sin(x)' between 0 and pi/2 the code is in appendix

$$I = \int_0^{\pi/2} x \sin(x) dx ;$$

IntTrap (your code result)	Analytical (result)	percentage_error_Trap
1.233700550136170	1.0000	23.370055013616977%

3 Composite

A function was created to integrate further functions in a discretised way from equation 3.1.

$$I \approx \frac{1}{2} \Delta x_n (f_a + 2 \sum_{i=1}^{n-1} f_i + f_b)$$

Equation 3-1 composite

Table 3-1 results for integration of $f(x) = 1/x$ using composite method a while loop was used to find when the result was within $1e-6$

	Analytical (6d.p)	Composite Trapezium (6d.p)			Converge 1e-6 (6d.p)
'Result'	0.693147	0.695634	0.693172	0.693153	0.693148
'n iterations'	Na	5	50	100	250
'percentage difference' (%)	Na	0.358905	0.003606	0.000901	0.000144

4 Simpsons

A function was created to integrate further functions in a discretised separating odd and even crossing over in a quadratic manner way from equation 4.1

$$I \approx \frac{1}{3} \Delta x_n (f_a + 4 \sum_{i=1, \text{ odd only}}^{n-1} f_i + 2 \sum_{i=1, \text{ even only}}^{n-1} f_i + f_b)$$

Equation 4-1 Simpsons

Table 4-1 results for integration of $f(x) = 1/x$ using Simpson's method a while loop was used to find when the result was within $1e-6$ of the analytical

	Analytical	Simpsons Trapezium			Converge 1e-6
	0.693147180559	0.693169793169	0.693147185554	0.693147180872	0.693147983875
'Result'	945	793	954	367	089
'n iterations'	NA	5	50	100	12
'percentage difference'	NA	0.003262310008 91617	7.206274658681 18e-07	4.507297973103 68e-08	0.000115893877 462248

5 Integration results tables

5.1 Composite convergence iterations

True value of integral= 0.693147180559

When n=5

Number of Iterations (n)	Integral Value	Difference from true value (Absolute value) (9 d.p)
1	0.316666666666667	0.376480514
2	0.459523809523810	0.233623371
3	0.584523809523810	0.108623371
4	0.695634920634921	0.002487740
5	0.695634920634921	0.002487740

When n=50

Number of Iterations (n)	Integral Value	Difference from true value
1	0.034607843137255	0.658539337
5	0.109406873596663	0.583740307
10	0.195665074622313	0.497482106
15	0.275070180760959	0.418077
20	0.348631419308647	0.344515761
25	0.417150292223622	0.275996888
30	0.481273940616091	0.21187324
35	0.541532402137856	0.151614778
40	0.598365264519091	0.094781916
45	0.652141142559350	0.041006038
50	0.693172179310195	2.499875024997156e-05

When n=100

Iterations Number (n)	Integral Value	Difference from true vaule
1	0.017400990099010	0.67574619
10	0.111366089613366	0.581781091
20	0.197255232523692	0.495891948

30	0.276347408412615	0.416799772
40	0.349639945345217	0.343507235
50	0.417925587560550	0.275221593
60	0.481844887424345	0.211302293
70	0.541922830502770	0.15122435
80	0.598595065799831	0.094552115
90	0.652227092067817	0.040920088
100	0.693153430481824	6.249921878809239e-06

When finding n value when tolerance equals 1×10^{-6} (n=250)

Iterations Number (n)	Integral Value (6d.p)	Difference from true value
1	0.750000	0.056852819440055
20	0.693320250888511	1.730703285652835e-04
40	0.693188268571296	4.108801135038753e-05
60	0.693165134526046	1.795396610082545e-05
80	0.693157194780150	1.001422020452392e-05
100	0.693153557378936	6.376818990849742e-06
120	0.693151594051106	4.413491161092686e-06
140	0.693150415361232	3.234801286722266e-06
160	0.693149652760055	2.472200109737699e-06
180	0.693149131178096	1.950618150892147e-06
200	0.693148758797937	1.578237991850528e-06
220	0.693148483698685	1.303138740138543e-06
240	0.693148274726070	1.094166124371299e-06
250	0.693148188606171	1.008046225270398e-06
251	0.693148180557945	9.999979997399322e-07

5.2 Simpsons Convergence iterations

True value of integral= 0.693147180559

When n=5

Number of Iterations (n)	Integral Value (15 d.p)	Difference from true value (Absolute value) (9 d.p)
1		
2	0.357142857142857	0.336004323
3		
4	0.571957671957672	0.121189509
5 (same answer as n=4)	0.571957671957672	0.121189509

When n=50

Number of Iterations (n)	Integral Value (15 d.p)	Difference from true value
1		
2	0.048964303670186	0.644182877
5		
10	0.191210448440789	0.501936732
15		
20	0.344567478657078	0.348579702
25		
30	0.477503633760499	0.215643547
35		
40	0.594823706759516	0.098323474
45		
50	0.679679172086941	0.013468008

When n=100

Iterations Number (n)	Integral Value	Difference from true value
1		
2	0.024737267844432	0.668409913
10	0.100007149606922	0.593140031
20	0.186766001410929	0.506381179
30	0.266595033914822	0.426552147
40	0.340519855915327	0.352627325
50	0.409353997264471	0.283793183

60	0.473753629528132	0.219393551
70	0.534255702335910	0.158891478
80	0.59130518372214	0.101841997
90	0.645274939111652	0.047872241
100	0.686447013368180	0.006700167

When finding n value when tolerance equals $1 \cdot 10^{-6}$ (n=12)

Iterations Number (n)	Integral Value (6d.p)	Difference from true value
1		
2	0.6944444444444444	0.001297264
3		
4	0.693253968253968	0.000106788
5		
6	0.693169793169793	2.26126E-05
7		
8	0.693154530654531	7.3501E-06
9		
10	0.693150230688930	3.05013E-06
11		
12	0.693148662209101	1.48165E-06
13		
14	0.693147983875089	8.03316E-07

6 Appendices

6.1 Your code script and functions

Numerical Dif

```
function [y] = Cos_eqution(x)

%UNTITLED3 Summary of this function goes here

% Detailed explanation goes here

y=cos(x);

end
```

```
function [fdx] = Numerical_Differentiation(fx,x,h)

%Numerical Differentiation

% fx is set to the name of the fuction to be differenctiated

% x is a vector vaules to be determined

% h is the difference between x1 and x2 it is a constant

fdx = zeros(size(fx)); % makes a empty vector that is the same size of fx

    fdx(1:end-1) = (fx(2:end) - fx(1:end-1)) / h; % calcs the derivative for all x vaules but the
last one

    fdx(end) = (fx(end) - fx(end-1)) / h; % works backwards for final vaule

end
```

```
function [y] = sin_scale( x,sf )

% This function multiplies the first input argument by 10 and the second by 100 then
return

%their values in y1 and y2 respectively.

y=sin(x./sf);

end
```

```
clc, clear ,close all
```

```
%% Work for the Sin Scale equation
```

```
N = 100;
```

```
x = linspace(-pi,pi,N)';
```

```
sf = [0.1 0.5 2]; % Scale factor values in an array named sf
```

```
N_sf = numel(sf); % Number of elements within sf (ML intrinsic function)
```

```
sin_array = zeros(N,N_sf); %Initialise an array of zeros to store data. % Notice the size  
of the array is predicted by
```

```
                %the number of values, N and the number of scale factors to be  
analysed, N_sf.
```

```
%Plot for sin scale equation
```

```
for ndx = 1: N_sf
```

```
sin_array(:,ndx) = sin_scale(x,sf(ndx));
```

```
end
```

```
figure(1)
```

```
plot(x,sin(x))
```

```
title(['Sin(x)'])
```

```
xlabel('x-axis')
```

```
ylabel('y-axis')
```

```
ylines(0) % y axis black line
```

```
figure(2)
```

```
plot(x,sin(x./0.1))

title(['A Sine fn; Scale factors 0.1'])

xlabel('x-axis')

ylabel('y-axis')

yline(0) % y axis black line


figure(3)

plot(x,sin_array);

title(['A Sine fn; limits = [-pi pi]; Scale factors = ',num2str(sf)])

xlabel('x-axis')

ylabel('y-axis')

yline(0) % y axis black line


%% Work for the Cos Equation

N = 100;

x = linspace(-pi,pi,N);

fx=Cos_equation(x); % Wrote like this to make fx a vector


h=x(2)-x(1); %h pulls vaules from the x matrix made in the function

fdx=zeros(size(x));

fdx = Numerical_Differentiation(fx,x,h);


%% Plots for cos(x) -sin(x) and fdx of cos(x)

x = linspace(-pi,pi,N); % range for x-axis
```

```
figure(4)

hold on

yline(0) % y axis black line

% Plot Cos(x)

%fx=Cos_eqution(x);

%plot(x,fx,'r','LineWidth',2) % cos(x) is red

% Plot -Sin(x) ->real derivative of cos(x)

z=-sin(x);

plot(x,z,'r','LineWidth',1)

fdx=Numerical_Differentiation(fx,x,h); % approx derivative of cos(x)

plot(x,fdx,'g','LineWidth',2)

title('Cos(x), -Sin(x) and Num Diff of Cos(x) over  $-\pi < x < \pi$ ')

xlabel('x-axis')

ylabel('y-axis')

hold off
```

Trap1

```
function int=trap1(f,a,b)

%Input - f is the integrand input as a string 'f'

% - a and b are upper and lower limits of integration

%h is the difference between a and b
```

%Output - s is the trapezoidal rule sum, equation 3

% evaluate the equation f for vaule a and b times by h then halfed

if a>b

error('b must be greater than a')

end

h=(b-a);

int=h*(feval(f,a)+feval(f,b))/2;

end

function [f] = Function_x_sinx(x)

%Equation to be intergrated for trapezoidal method

% Detailed explanation goes here

f=x*sin(x)

end

clc, clear ,close all

%% Working out Trapezoidal Method

a=0;

b=pi/2;

x=linspace(a,b,100);

f=@(x)x.*sin(x); % Written with a @ to make it a fuinction handle

int=trap1(f,a,b); % calls function

```
%% Area below the line

format long

int=0.5*(f(a)+f(b)*(b-a)) % calcs area under the graph


%% comparision to real vaule

I = integral(f,a,b); % Real vaule when integrated =1

Delta=I-int;

Delta_Percentage=(int-I/I)*100


%% Plot Graph

f=x.*sin(x);

fi=-x.*cos(x)+sin(x);

point1=a.*sin(a);

point2=b.*sin(b);


figure(1)

grid on

hold on

plot([a,b],[point1,point2],'b','LineWidth',1)

plot(x,f,'r')

plot(x,fi,'g--') % real intergral of x*sin(x)

plot(a,point1,'bo','MarkerSize',6)% circle at (0,0)

plot(b,point2,'bo','MarkerSize',6)% circle at (1.5708,1.5708)
```

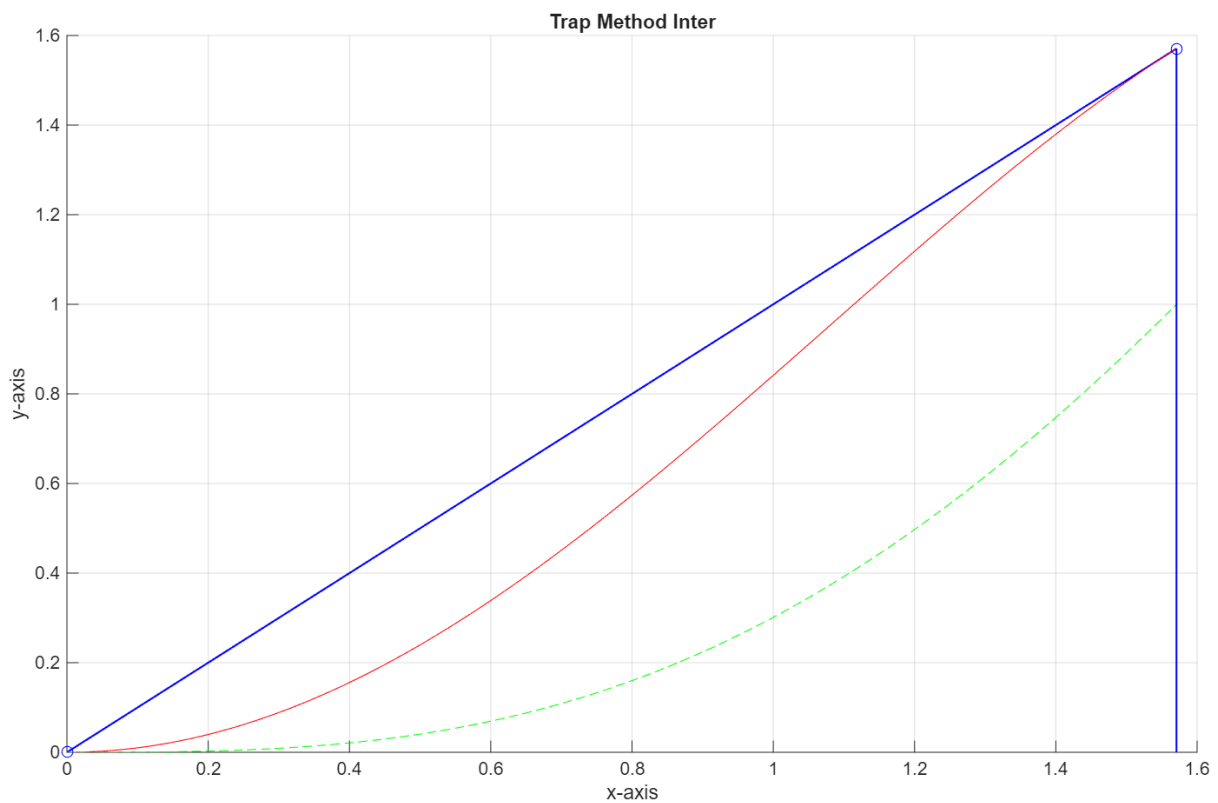
```
plot([b,b],[a,b],'b','LineWidth',1)% vertical line to point2
```

```
xlabel('x-axis');
```

```
ylabel('y-axis');
```

```
title('Trap Method Inter');
```

```
hold off
```



Area of the triangle in this case

Comp Trap

```
function fx = Function_1_x_fortask1_pt3(x)
```

```
%Function to be evaluated from instructions
```

```
% Detailed explanation goes here
```

```
fx=1/x
```

```
end
```

```
function integral_3=Comp_trapexium_rule(f,a,b,n)
```

```
%Input - f is the integrand input as a string 'f'
```

```
% - a and b are upper and lower limits of integration
```

```
% n is the number of intervals (see linspace in base page)
```

```
% deltax=(b-a/n)
```

```
% fxi is the function evaluated at each of the interior points i.e...
```

```
% xi=a+i*deltax
```

```
%Output - integral is the Composite trapezium rule sum, equation 4
```

```
% evaluate the equation f for x vaule between a and b
```

```
if a>b
```

```
error('b must be greater than a')
```

```
end
```

```
deltax=(b-a)/n;
```

```
Sum_for_fxi=0;
```

```
    for i=1:n-1 % i stats at 1. loops to sum for all f xi vaules increases i vaules for each  
loop
```

```
    xi=a+i*deltax;
```

```
Sum_for_fxi=Sum_for_fxi+feval(f,xi); % keeps adding f xi vaules until loop is finished
```

```
%integral_3=0.5*deltax.*(feval(f,a)+2*(Sum_for_fxi)+feval(f,b))

    end

    integral_3=0.5*deltax.*(feval(f,a)+2*(Sum_for_fxi)+feval(f,b));

end

clc, clear ,close all

%% Working out Composite Trapezium Rule n=5

a=1;

b=2;

n=5;

deltax=(b-a)/n;

x=linspace(a,b,n);

f = @(x) 1./x; % Written with a @ to make it a function handle

integral_3=Comp_trapezium_rule(f,a,b,n) % calls function and calcs integral using the
Composite trapezium Rule

Ans1=integral_3;

%% Working out Composite Trapezium Rule n=50

a=1;

b=2;

n=50;

deltax=(b-a)/n;

x=linspace(a,b,n);

f = @(x) 1./x; % Written with a @ to make it a function handle
```

```
integral_3=Comp_trapexium_rule(f,a,b,n); % calls function and calcs integral using the  
Composite trapium Rule
```

```
Ans2=integral_3;
```

```
%% Working out Composite Trapezium Rule n=100
```

```
a=1;
```

```
b=2;
```

```
n=100;
```

```
deltax=(b-a)/n;
```

```
x=linspace(a,b,n);
```

```
f = @(x) 1./x; % Written with a @ to make it a function handle
```

```
integral_3=Comp_trapexium_rule(f,a,b,n) % calls function and calcs integral using the  
Composite trapium Rule
```

```
Ans3=integral_3;
```

```
%% Working out Composite Trapezium Rule n= ? for tol 6 d.p part e
```

```
True_Ans=log(2)-log(1); % Real vaule when integrated for =0.6931
```

```
a=1;
```

```
b=2;
```

```
n=1; %starting vaule of n
```

```
difference=inf ;
```

```
tol=1e-6;
```

```
deltax=(b-a)/n;
```

```
x=linspace(a,b,n);
```

```
f = @(x) 1./x; % Written with a @ to make it a fuinction handle
```

```

while difference>tol % the loop will continue until this statement changes

integral_3=Comp_trapexium_rule(f,a,b,n); % calls function and calcs integral using the
Composite trapium Rule

    difference=abs(integral_3-True_Ans);

    n=n+1; % increase n vaules for the loop to start again

    %fprintf('Comp_trapezium for part e): integral3  $\approx$  %.6f after %n iterations (tol =
    %g)\n',n, tol);

end

n=n-1; % the last vaule of n to be calculated will be to second answer in the tolerance to
one step back is taken (n-1)

Ans4=integral_3;

%% Comparision of Answers to the True Vaule

True_Ans=log(2)-log(1); % Real vaule when integrated for =0.6931

format long % increase number of decimal places

Delta1=True_Ans-Ans1;

Delta2=True_Ans-Ans2;

Delta3=True_Ans-Ans3;

Delta4=True_Ans-Ans4;

Delta_Percentage1=((Ans1-True_Ans)/True_Ans)*100;

Delta_Percentage2=((Ans2-True_Ans)/True_Ans)*100;

Delta_Percentage3=((Ans3-True_Ans)/True_Ans)*100;

Delta_Percentage4=((Ans4-True_Ans)/True_Ans)*100;

% Put Answers in a table

```

```
column1=[5;50;100;NaN;n];

column2=[Ans1;Ans2;Ans3;True_Ans;Ans4];

column3=[Delta_Percentage1;Delta_Percentage2;Delta_Percentage3;0;Delta_Percentage
4];

Table=table(column1,column2,column3,'VariableNames',{'N vaule','Calculated
Answer','Percentage Difference (%)'})

%% Plot Graph

x=linspace(a,b,n); % vaues for x-axis

f=1./x; % Function for task1 pt3

fi= log(x); % real intergral 1/x

%point1=a.*sin(a);

%point2=b.*sin(b);

figure(1)

grid on

hold on

%plot([a,b],[point1,point2],'b','LineWidth',1)

plot(x,f,'r')

plot(x,fi,'g--') % real intergral of x*sin(x)

%plot(a,point1,'bo','MarkerSize',6)% circle at (0,0)

%plot(b,point2,'bo','MarkerSize',6)% circle at (1.5708,1.5708)

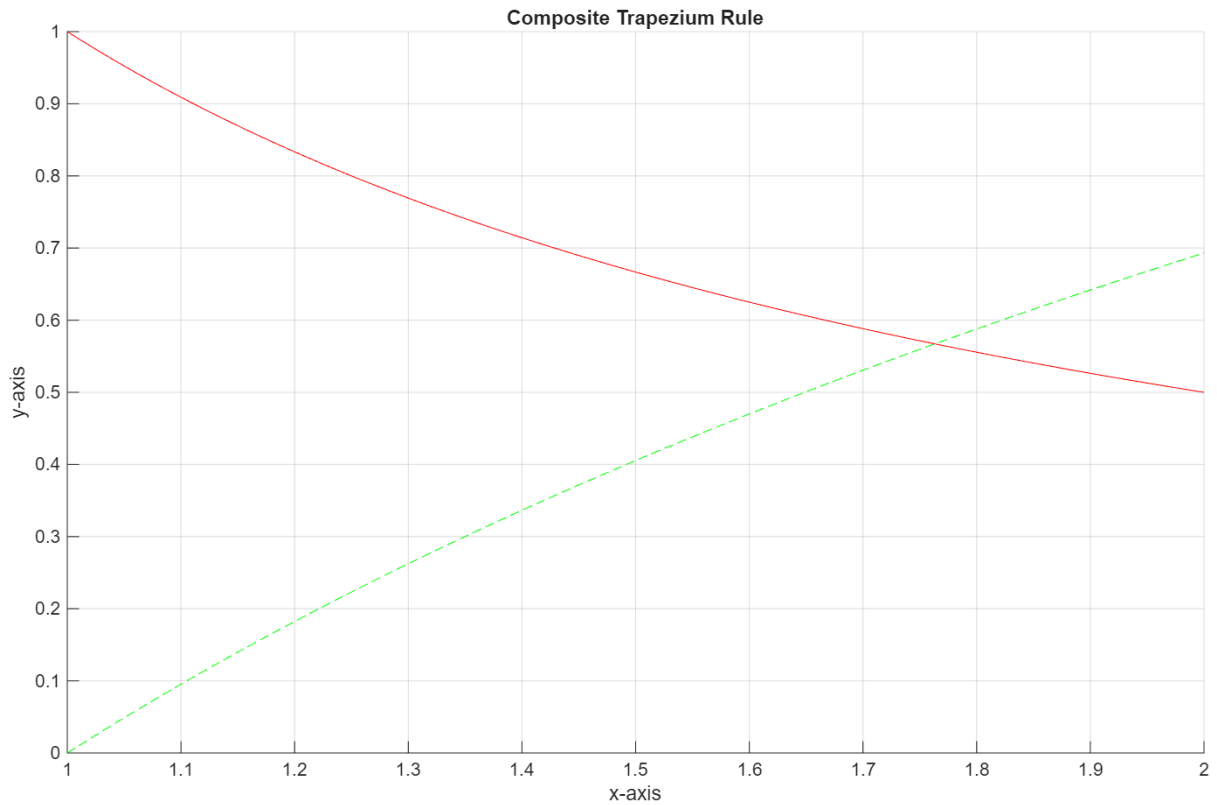
%plot([b,b],[a,b],'b','LineWidth',1)% vertical line to point2

xlabel('x-axis');
```

```
ylabel('y-axis');
```

```
title('Composite Trapezium Rule');
```

```
hold off
```



Simpson Rule

```
function [integral_4]=Simpson_Rule(f,a,b,n)
```

```
%Input - f is the integrand input as a string 'f'
```

```
% - a and b are upper and lower limits of integration
```

```
% n is the number of intervals (see linspace in base page)
```

```
% deltax=(b-a/n)
```

```
% fxi is the function evaluated at each of the interior points i.e...
```

```
% xi=a+i*deltax
```

```
%Output - integral is the Simpson's rule calc, equation 5
```

```
% evaluate the equation f for x vaule between a and b
```

```
if a>b
```

```
error('b must be greater than a')
```

```
end
```

```
% Simpson's rule for even vaules of n
```

```
if mod(n,2) ~= 0
```

```
    warning('n must be even for Simpson"s Rule. Increasing n by 1.');
```

```
    n = n + 1;
```

```
end
```

```
deltax=(b-a)/n;
```

```
Sum_for_fxi_odd=0; % for odd number sum
```

```
Sum_for_fxi_even=0; % for even number sum
```

```
% fxi sum for odd numbers only
```

```
    for i=1:n-1; % i stats at 1. loops to sum for all f xi vaules increases i vaules for each  
loop
```

```
        xi=a+i*deltax;
```

```
        if mod(i,2)==1 % Simpson's rule for odd index
```

```
            % fxi sum for odd numbers only
```

```
            Sum_for_fxi_odd=Sum_for_fxi_odd+feval(f,xi); % keeps adding f xi vaules until loop  
is finished
```

```
        else
```

```

% fxi sum for even numbers only

Sum_for_fxi_even=Sum_for_fxi_even+feval(f,xi); % keeps adding f xi vaules until
loop is finished

%integral_4=(deltax/3).*(feval(f,a)+4*Sum_for_fxi_odd+2*Sum_for_fxi_even+feval(f,b));

    fprintf('integral_4=%.15f, i=%d', integral_4,i);

end

end

integral_4=(deltax/3).*(feval(f,a)+4*Sum_for_fxi_odd+2*Sum_for_fxi_even+feval(f,b));

end

```

```

clc, clear ,close all

```

```

%% Working out Simpson's Rule n=5

```

```

a=1;

```

```

b=2;

```

```

n=5;

```

```

deltax=(b-a)/n;

```

```

f = @(x) 1./x; % Written with a @ to make it a fuinction handle

```

```

format long

```

```

integral_4=Simpson_Rule(f,a,b,n); % calls function and calcs integral using the
Composite trapium Rule

```

```

Ans1=integral_4;

```

```

%% Working out Composite Trapezium Rule n=50

```

```

a=1;

```

```
b=2;

n=50;

deltax=(b-a)/n;

f = @(x) 1./x; % Written with a @ to make it a function handle

integral_4=Simpson_Rule(f,a,b,n); % calls function and calcs integral using the
Composite trapezium Rule

Ans2=integral_4;

%% Working out Simpson's Rule n=100

a=1;

b=2;

n=100;

deltax=(b-a)/n;

f = @(x) 1./x; % Written with a @ to make it a function handle

integral_4=Simpson_Rule(f,a,b,n); % calls function and calcs integral using the
Composite trapezium Rule

Ans3=integral_4;

%% Working out Simpson's Rule n= ? for tol 6 d.p part d

True_Ans=log(2)-log(1); % Real value when integrated for =0.6931

a=1;

b=2;

n=0; %starting value of n

difference=inf;

tol=1e-6;

deltax=(b-a)/n;
```

```
f = @(x) 1./x; % Written with a @ to make it a function handle
```

```
while difference>tol % the loop will continue until this statement changes
```

```
integral_4=Simpson_Rule(f,a,b,n); % calls function and calcs integral using the
Simpson's Rule
```

```
    difference=abs(integral_4-True_Ans);
```

```
    n=n+1; % increase by 1 to keep n an even number for the loop to start again
```

```
    fprintf('integral_4=%.15f, n=%d', integral_4,n);
```

```
end
```

```
n=n-2; % the last value of n to be calculated will be to second answer in the tolerance to
one step back is taken (n-2)
```

```
Ans4 = integral_4
```

```
    %N is odd check if this is correct
```

```
%% Comparision of Answers to the True Value
```

```
True_Ans=log(2)-log(1); % Real value when integrated for =0.6931
```

```
format long % increase number of decimal places
```

```
Delta1=True_Ans-Ans1;
```

```
Delta2=True_Ans-Ans2;
```

```
Delta3=True_Ans-Ans3;
```

```
Delta4=True_Ans-Ans4;
```

```
Delta_Percentage1=((Ans1-True_Ans)/True_Ans)*100;
```

```
Delta_Percentage2=((Ans2-True_Ans)/True_Ans)*100;
```

```
Delta_Percentage3=((Ans3-True_Ans)/True_Ans)*100;
```

```
Delta_Percentage4=((Ans4-True_Ans)/True_Ans)*100;
```

```
% Put Answers in a table
```

```
column1=[5;50;100;NaN;n];
```

```
column2=[Ans1;Ans2;Ans3;True_Ans;Ans4];
```

```
column3=[Delta_Percentage1;Delta_Percentage2;Delta_Percentage3;0;Delta_Percentage  
4];
```

```
Table=table(column1,column2,column3,'VariableNames',{'N vaule','Calculated  
Answer','Percentage Difference (%)'})
```

```
%% Plot Graph
```

```
x=linspace(a,b,n); % vaues for x-axis
```

```
f=1./x; % Function for task1 pt3
```

```
fi= log(x); % real intergral 1/x
```

```
%point1=a.*sin(a);
```

```
%point2=b.*sin(b);
```

```
figure(1)
```

```
grid on
```

```
hold on
```

```
%plot([a,b],[point1,point2],'b','LineWidth',1)
```

```
plot(x,f,'r')
```

```
plot(x,fi,'g--') % real intergral of x*sin(x)
```

```
%plot(a,point1,'bo','MarkerSize',6)% circle at (0,0)
```

```
%plot(b,point2,'bo','MarkerSize',6)% circle at (1.5708,1.5708)
```

```
%plot([b,b],[a,b],'b','LineWidth',1)% vertical line to point2
```

```
xlabel('x-axis');
```

```
ylabel('y-axis');
```

```
title('Composite Trapezium Rule');
```

```
hold off
```

