

```
clear, clc, close all,
format short g;
```

Task 1

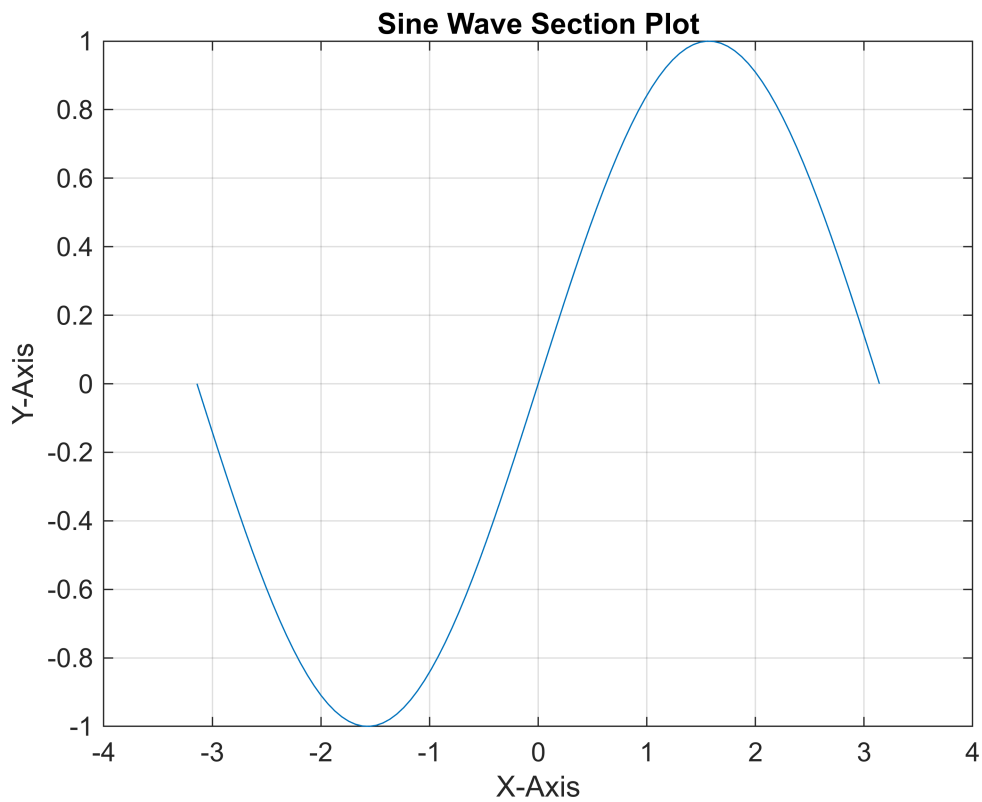
Table of Contents

(i) (e) $y = \sin(x)$ plot [1].....	1
(ii) (e) $y = \sin_scale(x,sf)$ plot [1].....	2
(ii) (e) $y = \sin$ array plot [3].....	3
Numerical Differentiation $y=\cos(x)$ plot [3].....	4
Numerical Intergration (a) $\int \sin(x) dx$ [3].....	6
Numerical Intergration (b) compare [3].....	6
Comp Trap (c) $1/x$ for 5,50,100 [4].....	7
Comp Trap (d) compare [1].....	7
Comp Trap (e) to tolerance [3].....	8
Simpsons (b) $1/x$ for 5,50,100 [4].....	8
Simpsons (c) compare [1].....	9
Simpsons (d) to tolerance [3].....	9

(i) (e) $y = \sin(x)$ plot [1]

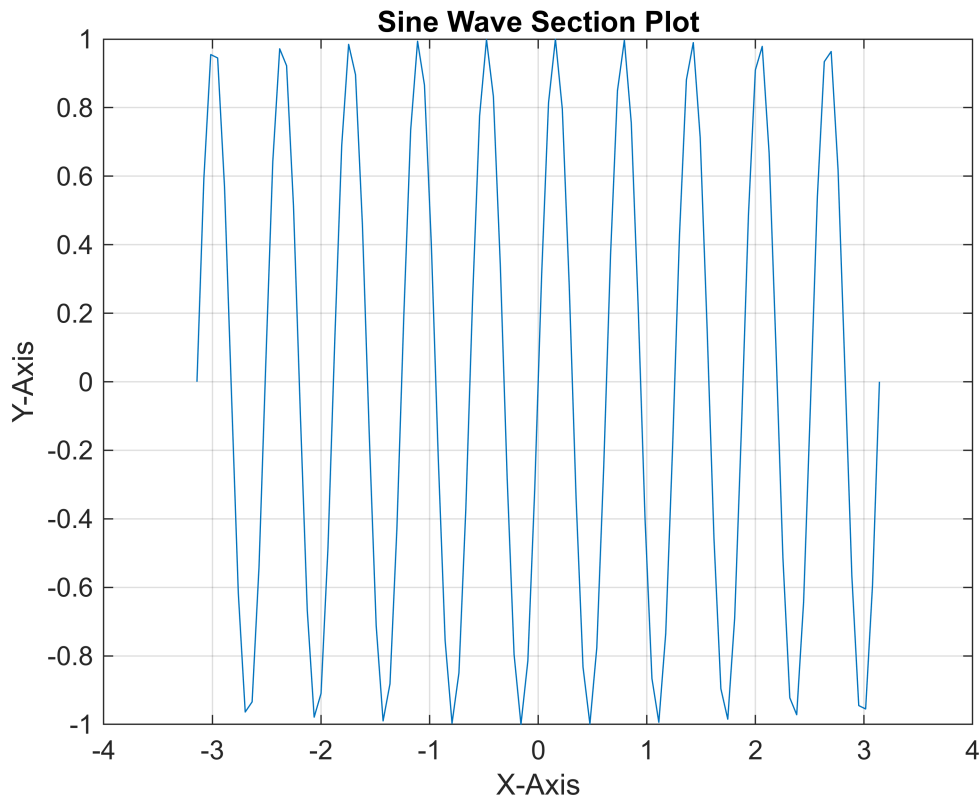
```
N = 100;
x = linspace(-pi,pi,N)';
y = sin(x);

figure(1),
plot(x,y);
xlabel 'X-Axis'; ylabel 'Y-Axis';
grid on; title 'Sine Wave Section Plot';
```



(ii) (e) $y = \sin_scale(x, sf)$ plot [1]

```
sf = 0.1;  
y = sin_scale(x,sf);  
figure(2); plot(x,y);  
xlabel 'X-Axis'; ylabel 'Y-Axis';  
grid on; title 'Sine Wave Section Plot';
```

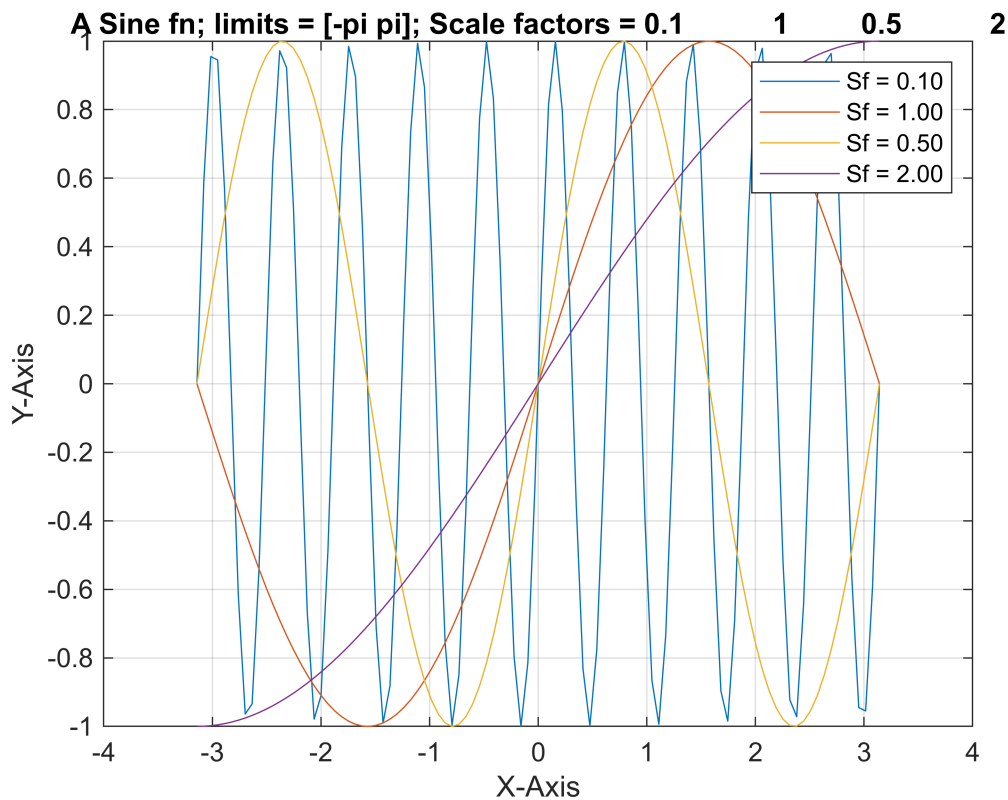


(ii) (e) $y = \sin$ array plot [3]

```
sf=[0.1 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);

for ndx = 1: N_sf
    sin_array(:,ndx) = sin_scale(x,sf(ndx));
end

figure(3); plot(x,sin_array); xlabel 'X-Axis'; ylabel 'Y-Axis'; grid on;
title(['A Sine fn; limits = [-pi pi]; Scale factors = ',num2str(sf)]);
legend(num2str(sf),'Sf = %.2f');
```



Numerical Differentiation $y=\cos(x)$ plot [3]

III.(a)

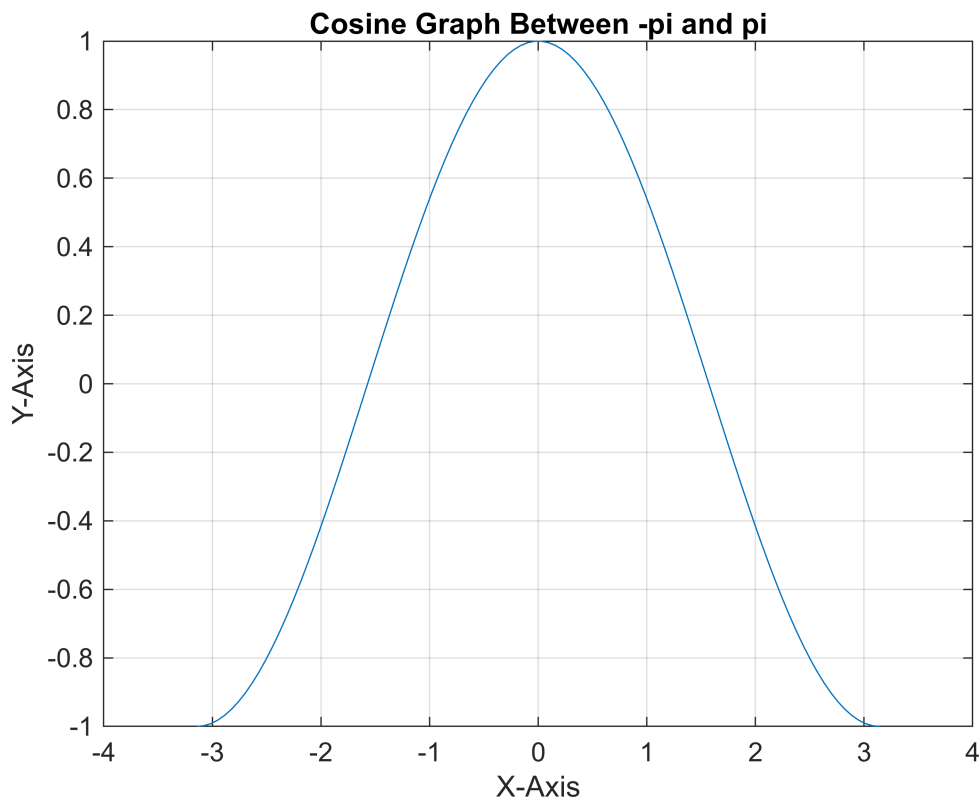
```
xLimit_1 = -pi;
xLimit_2 = pi;
N = 100;
x = linspace(xLimit_1,xLimit_2,N)';
```

III.(b)

```
y = cos(x)';
```

III.(c)

```
[x,y] = MycosFunc(xLimit_1,xLimit_2,N);
figure(); plot(x,y);
title('Cosine Graph Between -pi and pi');
xlabel 'X-Axis'; ylabel 'Y-Axis'; grid on;
```

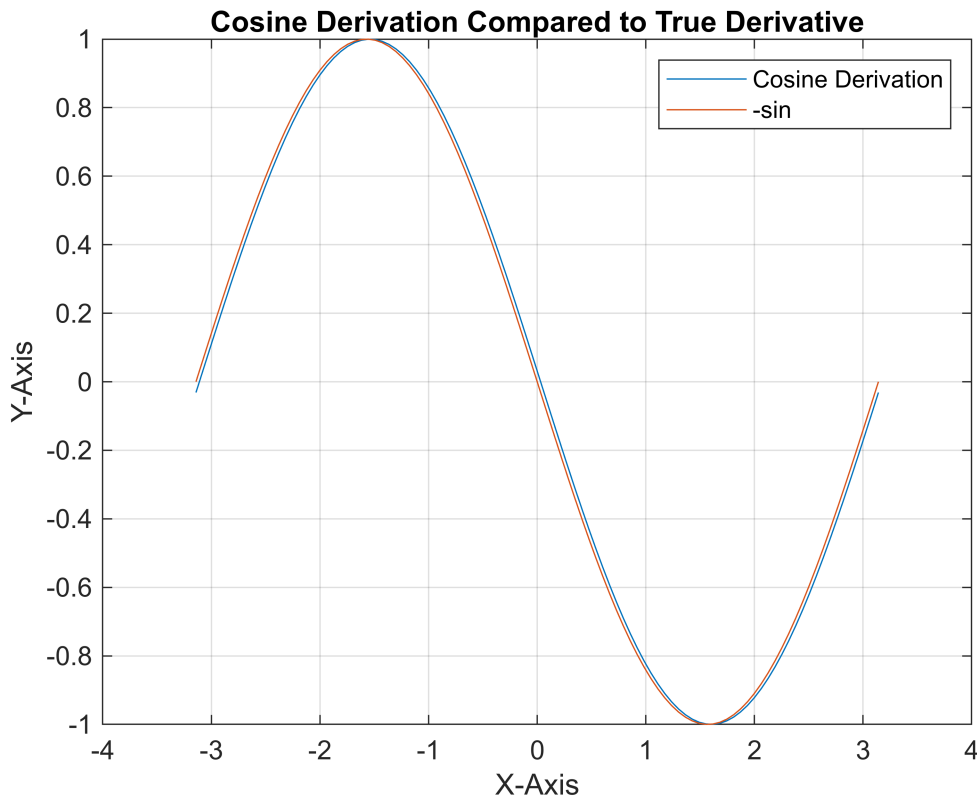


III.(d)

```
xLimit_1 = -pi;
xLimit_2 = pi;
N = 100;
[x,dydx] = DerivFunct(xLimit_1,xLimit_2,N);
```

III.(e)

```
figure(); plot(x,dydx); hold on; grid on;
x = linspace(xLimit_1,xLimit_2,N);
y = -sin(x);
plot(x,y); hold off; legend ('Cosine Derivation','-sin');
title('Cosine Derivation Compared to True Derivative');
xlabel 'X-Axis'; ylabel 'Y-Axis';
```



Numerical Intergration (a) $\int \sin(x) dx$ [3]

```
f = @(x)x.*sin(x);
Integration = trap1(f,0,pi/2);
fprintf('Integration Approx = %.8f \r \r' , Integration)
```

Integration Approx = 1.23370055

Numerical Intergration (b) compare [3]

```
syms q
p = q*sin(q);
awns = int(p,q,[0 pi/2]);

% Creating Table
Difference(1) = Integration-double(awns);
True_Area(:) = double(awns);
Number_of_Trapeziums(1) = 1;
Percentage_Error(1) = ((Integration-double(awns))/double(awns))*100;
fprintf(' Analytical Integration = %.8f \r \r' , True_Area(1))
```

Analytical Integration = 1.00000000

```
fprintf(' Percentage Error = %.8f \r \r' , Percentage_Error(1))
```

Percentage Error = 23.37005501

2. Composite Trapezium Rule

2.(a,b)

```
a = 0;           % Lower x Limit
b = pi/2;        % Upper x Limit
n = 100;         % Number of trapiziums
f = @(x)x.*sin(x); % Function handle

Integration2 = Trap2(f,a,b,n);
PercentageError2 = ((Integration2-double(awns))/double(awns))*100;

% Creating Table
Method = ["Trapezoidal";"Composite"];
Difference(2,1) = Integration2-double(awns);
True_Area(2,1) = double(awns);
Traps(2,1) = n;
Perc_Err(2,1) = PercentageError2;
Integration(2,1) = Integration2;
Trapezium_Integration = table(Method,Traps,True_Area,...
    Integration,Difference,Perc_Err);
display(Trapezium_Integration)
```

Trapezium_Integration = 2x6 table

	Method	Traps	True_Area	Integration	Difference	Perc_Err
1	"Trapezoidal"	0	1	1.2337	0.2337	0
2	"Composite"	100	1	1	2.0562e-05	0.0020562

Comp Trap (c) 1/x for 5,50,100 [4]

```
f = @(x)1/x;
Integration = Trap2(f,1,2,[5 50 100]);
fprintf(' Estimated area (to 8d.p.) = %.8f \r',Integration)
```

Estimated area (to 8d.p.) = 0.69563492
Estimated area (to 8d.p.) = 0.69317218
Estimated area (to 8d.p.) = 0.69315343

```
p = 1/q;
awns = int(p,q,[1 2]);
fprintf(' True area (to 8d.p.) = %.8f \r',awns)
```

True area (to 8d.p.) = 0.69314718

Comp Trap (d) compare [1]

```
Difference = (Integration-double(awns))';
Integration = Integration';
```

```

True_Area(1:3,1) = double(awns);
Numb_Traps = [5;50;100];
Perc_Err = (((Integration-double(awns))./double(awns)).*100);
Composite_Trapeziums = table(Numb_Traps,True_Area,...
    Integration,Difference,Perc_Err);
display(Composite_Trapeziums)

```

Composite_Trapeziums = 3x5 table

	Numb_Traps	True_Area	Integration	Difference	Perc_Err
1	5	0.69315	0.69563	0.0024877	0.35891
2	50	0.69315	0.69317	2.4999e-05	0.0036066
3	100	0.69315	0.69315	6.2499e-06	0.00090167

Comp Trap (e) to tolerance [3]

```

max_count = 500;
tol = 1e-6;
n = 0;
Diff = inf;
counter = 0;
while (Diff>=tol && counter<max_count)
    for n=n+1
        Area = Trap2(f,1,2,n);
    end
    counter = counter +1;
    Diff = norm(double(awns)-Area);
end

fprintf(' Counter = %i \r' , counter)

```

Counter = 250

```
fprintf(' True Area (to 6d.p.) = %.6f \r' , double(awns))
```

True Area (to 6d.p.) = 0.693147

```
fprintf(' Area (to 6d.p.) = %.6f \r' , Area)
```

Area (to 6d.p.) = 0.693148

```
fprintf(' Diff (to 8d.p.) = %.8f \r' , Diff)
```

Diff (to 8d.p.) = 0.00000100

```
fprintf(' Amount of Trapeziums = %i \r\r' , n)
```

Amount of Trapeziums = 250

3. Simpsons Rule

Simpsons (b) 1/x for 5,50,100 [4]

```
Integration = SimpsonRule(f,1,2,[5 50 100]);
fprintf('Estimated area (to 8d.p.) = %.8f \r',Integration)
```

```
Estimated area (to 8d.p.) = 0.65820106
Estimated area (to 8d.p.) = 0.69314719
Estimated area (to 8d.p.) = 0.69314718
```

Simpsons (c) compare [1]

```
Area = Integration';
Numb_Traps = [5;50;100];
True_Area(1:3,1) = double(awns);
Diff_off_True = double(Integration'-awns);
Perc_Err = double(((Integration'-awns)./awns).*100);
Simpson_Rule = table(Numb_Traps,True_Area,Area,...
    Diff_off_True,Perc_Err);
display(Simpson_Rule)
```

Simpson_Rule = 3x5 table

	Numb_Traps	True_Area	Area	Diff_off_True	Perc_Err
1	5	0.69315	0.6582	-0.034946	-5.0417
2	50	0.69315	0.69315	4.995e-09	7.2063e-07
3	100	0.69315	0.69315	3.1242e-10	4.5073e-08

Simpsons (d) to tolerance [3]

```
max_count = 200;
tol = 1e-6;
n = 0;
counter = 0; Diff = inf;
while (Diff>tol && counter<max_count)
    for n=n+1
        Integration = SimpsonRule(f,1,2,n);
        counter = counter +1;
    end
    Diff = norm(double(awns)-Integration);
end

fprintf('Iteration = %i \r' , counter)
```

Iteration = 14

```
fprintf('True Area (to 8d.p.) = %.8f \r' , double(awns))
```

True Area (to 8d.p.) = 0.69314718

```
fprintf('Area Estimate (to 8d.p.) = %.8f \r' , Integration)
```

Area Estimate (to 8d.p.) = 0.69314798

```
fprintf('Amount of Sections = %i \r \r' , n)
```

Amount of Sections = 14