

Task 1

please convert this to pdf for submission

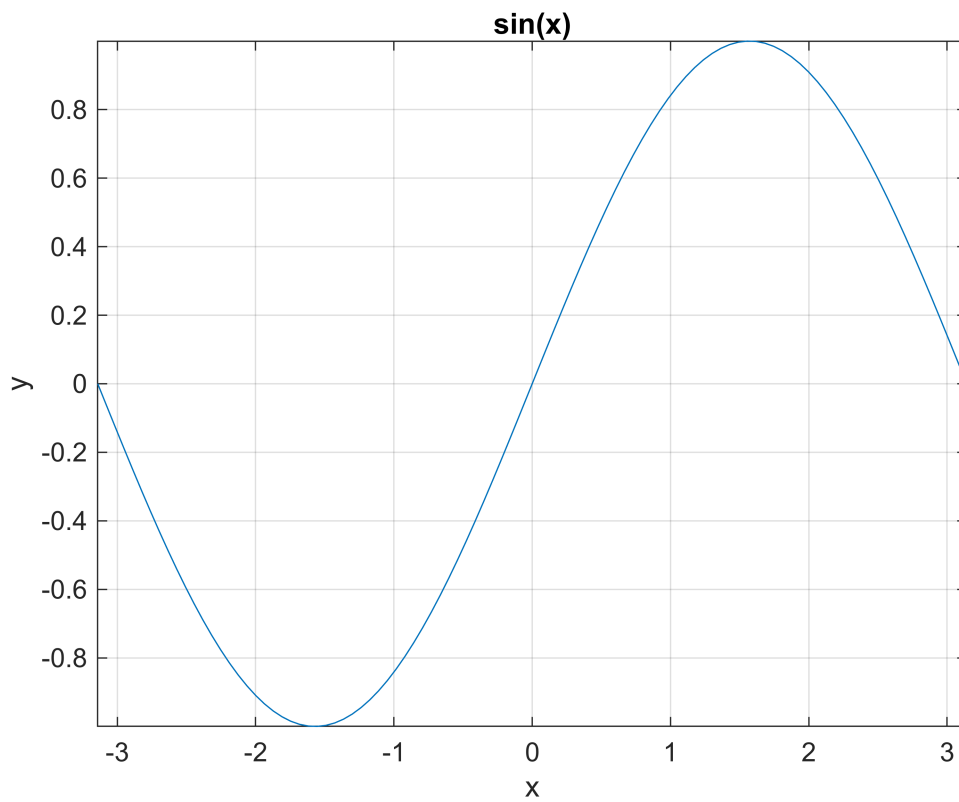
Table of Contents

(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].....	5
Numerical Intergration (b) compare [3].....	5
Comp Trap (c) $1/x$ for 5,50,100 [4].....	6
Comp Trap (d) compare [1].....	6
Comp Trap (e) to tolerance [3].....	6
Simpsons (b) $1/x$ for 5,50,100 [4].....	7
Simpsons (c) compare [1].....	7
Simpsons (d) to tolerance [3].....	7

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

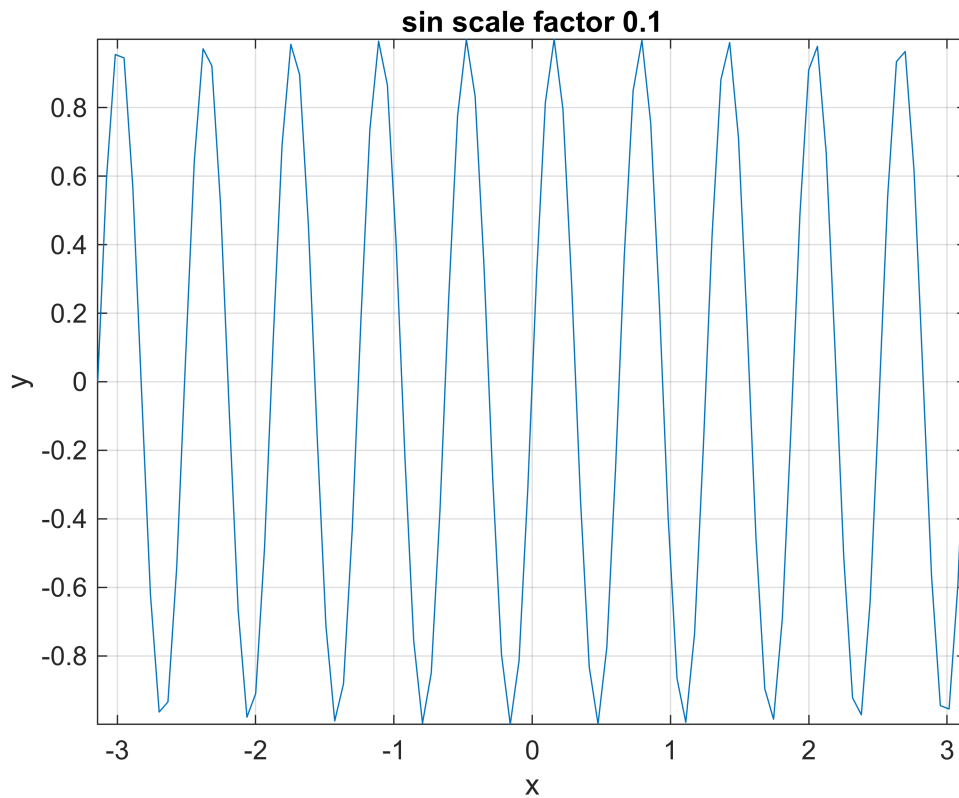
```
clc, clear, close all
format short g

N = 100;
x = linspace(-pi,pi,N)';
y = sin(x);
figure(1)
plot(x,y); axis tight; grid on; xlabel('x'); ylabel('y'); title('sin(x'))
```



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

```
sf = 0.1 ;  
y = sin_scale(x, sf);  
figure(2)  
plot(x, y); axis tight; grid on; xlabel('x'); ylabel('y'); title('sin scale factor  
0.1')
```

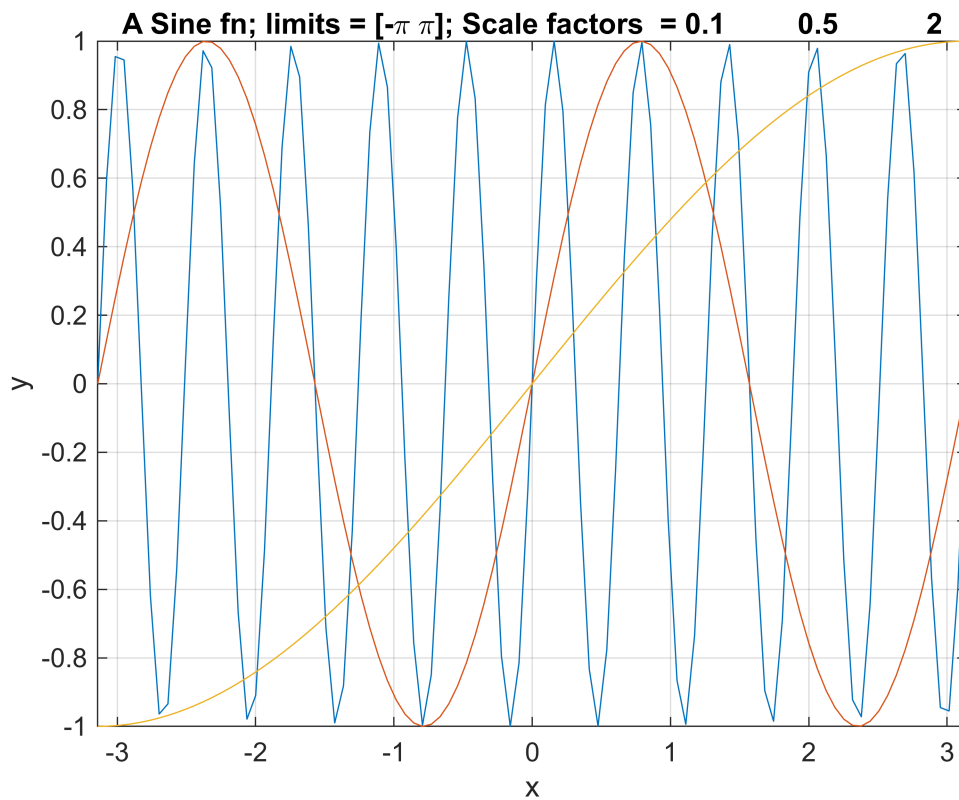


(ii) (e) $y = \sin$ array plot [3]

```
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)

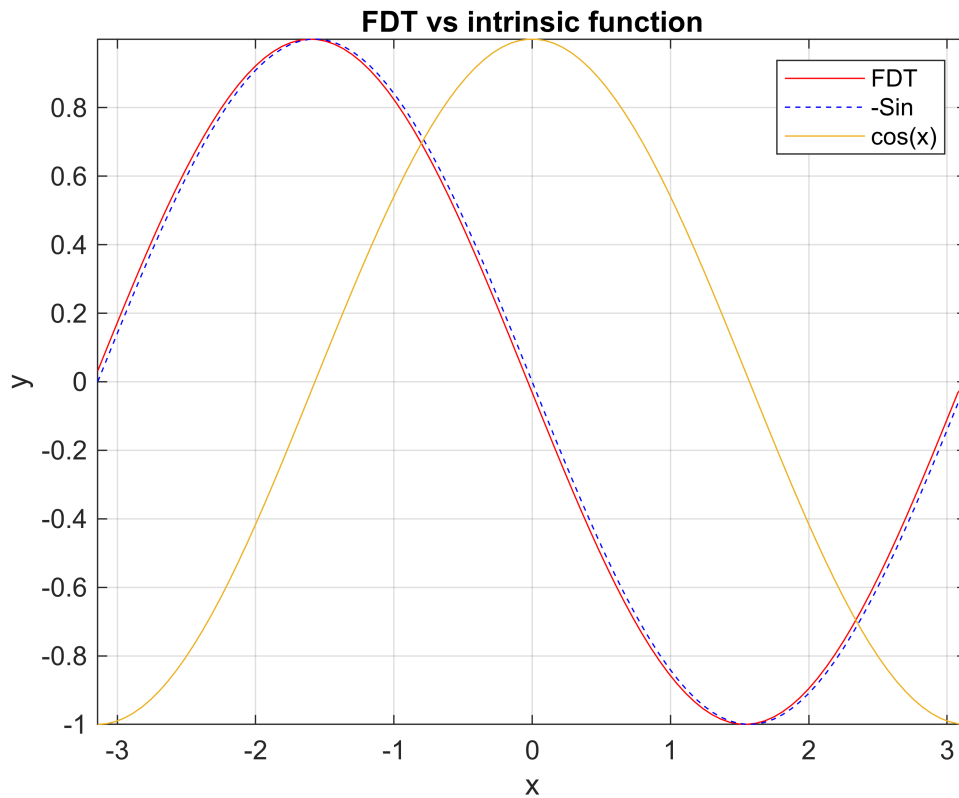
sin_array = zeros(N,N_sf); %Initialise an array of zeros to store data.

for ndx = 1: N_sf
    sin_array(:,ndx) = sin_scale(x,sf(ndx));
end
figure(3),plot(x,sin_array)
title(['A Sine fn; limits = [-\pi \pi]; Scale factors = ',num2str(sf)]); axis
tight; grid on; xlabel('x'); ylabel('y');
```



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

```
N=100;
y=cos(x);
dyest=forwarddiff(x,y);
dyreal=-sin(x);
figure;
plot(x,dyest,"r")
hold on
plot(x,dyreal,"--b")
plot(x,y)
title("FDT vs intrinsic function"); xlabel("x"); ylabel("y"); legend("FDT", "-Sin", "cos(x)"); axis tight, grid on;
hold off
```



Numerical Intergration (a) $\int_0^{\pi/2} \sin(x) dx$ [3]

```
a=0;
b=pi/2;

xsinx=t1(x);
trapez=trap1('t1',a,b);
```

Numerical Intergration (b) compare [3]

```
syms x
I_analytic = int(x * sin(x), x, 0, pi/2);
an_value_trap = double(I_analytic); % Convert symbolic result to a numeric value
pct_err1=((trapez-an_value_trap)/an_value_trap)*100;
Trapez_table=table(["Trapezoidal method"; ...
    "Analytical Integration"; "Percentage Error"], ...
    [trapez; an_value_trap; pct_err1], ...
    'VariableNames', {'Description', 'Value'});
disp(Trapez_table);
```

Description	Value
"Trapezoidal method"	1.2337
"Analytical Integration"	1
"Percentage Error"	23.37

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

```
% f(x)=x*sin(x)
f = @(x) x .* sin(x);
comp_trap=composite_trapezium(f,a,b,100);
pct_err2=((comp_trap-an_value_trap)/an_value_trap)*100;
comp_trap_table=table(["Composite trapezium rule"; ...
    "Analytical Integration"; "Percentage Error"], ...
    [comp_trap; an_value_trap; pct_err2], ...
    'VariableNames', {'Description', 'Value'});
disp(comp_trap_table);
```

Description	Value
"Composite trapezium rule"	1
"Analytical Integration"	1
"Percentage Error"	0.0020562

```
% f(x)=1/x
f1=@(x) 1/x;
a = 1;
b = 2;
comp_trap_5=composite_trapezium(f1,a,b,5);
comp_trap_50=composite_trapezium(f1,a,b,50);
comp_trap_100=composite_trapezium(f1,a,b,100);
```

Comp Trap (d) compare [1]

```
an_val=log(2);
comp_trap_table2 = table( ...
    [5; 50; 100], ... % Number of intervals
    [comp_trap_5; comp_trap_50; comp_trap_100], ... % Composite Trapezium values
    repmat(an_val, 3, 1), ... % Analytical value (repeated for each row)
    'VariableNames', {'Intervals', 'Composite trapezium rule', 'Analytical Value'} );
disp(comp_trap_table2);
```

Intervals	Composite trapezium rule	Analytical Value
5	0.69563	0.69315
50	0.69317	0.69315
100	0.69315	0.69315

Comp Trap (e) to tolerance [3]

```
n1=Accuracy_check(an_val,1e-6,@composite_trapezium,'Composite trapezium',f1,a,b);
```

Using Composite trapezium:
 Number of subintervals needed to match analytical value to 1.000000e-06 decimal places: 250
 Analytical Value: 0.6931471806

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

```
simp_5=simpsons(f1,a,b,5); simp_50=simpsons(f1,a,b,50);
simp_100=simpsons(f1,a,b,100);
```

Simpsons (c) compare [1]

```
simp_table = table( ...
    [5; 50; 100], ... % Number of intervals
    [simp_5; simp_50; simp_100], ... % Composite Trapezium values
    repmat(an_val, 3, 1), ... % Analytical value (repeated for each row)
    'VariableNames', {'Intervals', 'Simpsons rule', 'Analytical Value'} );

disp(simp_table);
```

Intervals	Simpsons rule	Analytical Value
5	0.6582	0.69315
50	0.69315	0.69315
100	0.69315	0.69315

Simpsons (d) to tolerance [3]

```
n2=Accuracy_check(an_val,1e-6,@simpsons,'Simpsons rule',f1,a,b);
```

Using Simpsons rule:
 Number of subintervals needed to match analytical value to 1.000000e-06 decimal places: 14
 Analytical Value: 0.6931471806