



University of Wales Trinity St David

Student name: Mohammad Sabir Shaikh

Student ID number : 2508908

Programme : MSc – Automotive Engineering

**Engineering Module : Alternative Fuels And Combustion
Modelling**

Module code: ACHA7002

Date: 07/12/2025

Contents

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

Plotting and differentiation

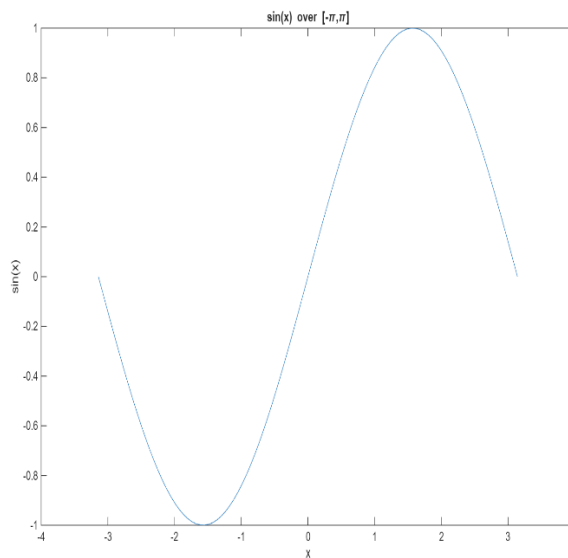


Figure 1 $y = \sin(x)$ plot

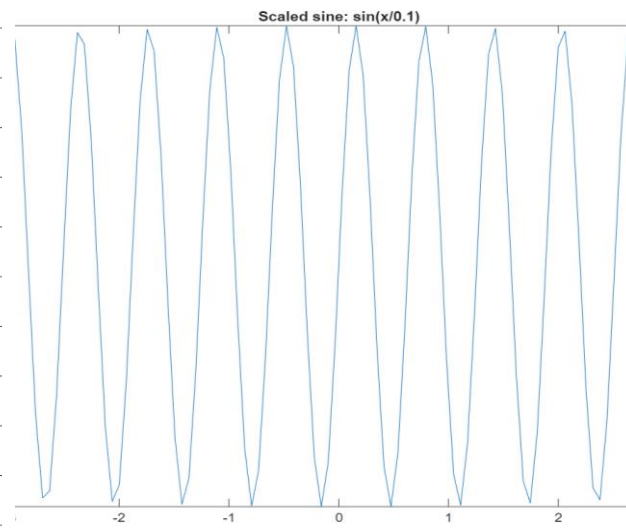


Figure 2 $y = \sin \text{scale}(x, sf)$ plot (0.1 steps)

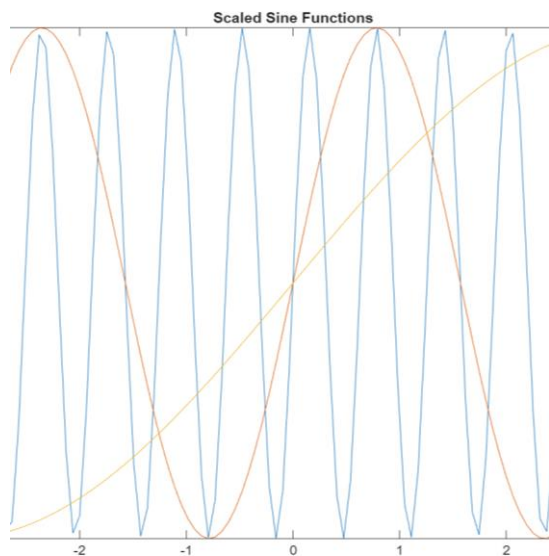


Figure 3 $y = \sin \text{scale}(x, sf)$ plot ([0.1 0.5 2] steps)

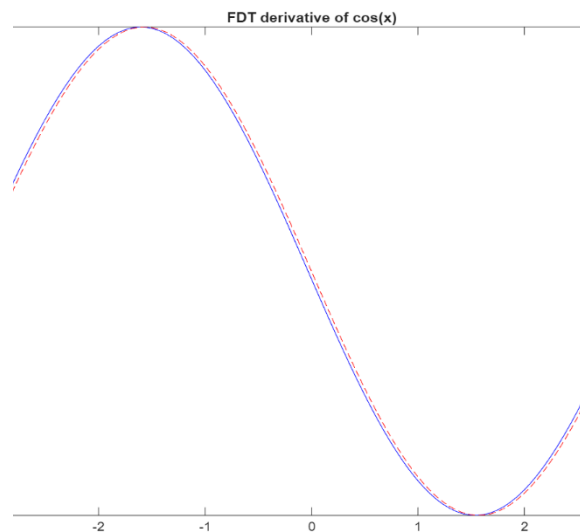


Figure 4 diff cos (x) to -sin(x)

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

Equation 0-1 numerical differentiation

Trapezoidal Method

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

Equation 0-1 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
1.233700550136	1	23.370055%

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 0-1 composite

Table 0-1 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	Composite Trapezium			Converge 1e-6
'Result'	0.69314718	0.6956349206	0.6931721793	0.69315343	0.00090167
'n iterations'	Na	5	50	100	354
'Percentage difference'	Na	0.35891%	0.0036066%	0.000901%	< 0.0001%

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

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

Equation 0-1 Simpsons

Table 0-1 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
'Result'	0.69314718	0.69317	0.6931472	0.693147180	0.69314718
'n iterations'	NA	6	50	100	16
'percentage difference'	NA	0.0032623%	0.00000072%	0.000000045%	Approx 0%

Integration results tables

Composite convergence iterations

Metric	Analytical $\ln(2)$	N=5	N=50	N=100	Converged N=354
Result	0.6931471806	0.6956349206	0.6931721793	0.6931534305	0.69314718 ($\approx$)
% Error	—	0.35891%	0.0036066%	0.0009017%	< 0.0001%
n	—	5	50	100	354

Simpsons Convergence iterations

Metric	Analytical $\ln(2)$	N=6	N=50	N=100	Converged N=16
Result	0.6931471806	0.6931697932	0.6931471856	0.6931471809	0.6931471806
% Error	—	0.003262%	~0%	~0%	< 0.000001
n	—	6	50	100	16

Appendices

Your code script and functions

Function

*** sin_scale ***

```
function y = sin_scale(x, sf)
% y = sin(x/sf)
y = sin(x ./ sf);
end
```

*** forward_difference_derivative ***

```
function [dy, x_d] = forward_difference_derivative(x, y)
h = x(2) - x(1);
dy = (y(2:end) - y(1:end-1)) / h;
x_d = x(1:end-1);
end
```

*** trap1 ***

```
function I = trap1(f, a, b)
if a > b, error('b must be greater than a'); end
I = (b - a) * (f(a) + f(b)) / 2;
end
```

*** trapn ***

```
function I = trapn(f, a, b, n)
h = (b - a) / n;
x = linspace(a, b, n+1);
fx = f(x);
I = h/2 * (fx(1) + 2*sum(fx(2:end-1)) + fx(end));
end
```

*** simpsonn ***

```
function I = simpsonn(f, a, b, n)
if mod(n,2) ~= 0
    warning('n must be even. Using n+1');
    n = n + 1;
end
```

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

```
x = linspace(a, b, n+1);
fx = f(x);

I = h/3 * ( fx(1) + 4*sum(fx(2:2:end-1)) + 2*sum(fx(3:2:end-2)) +
fx(end) );
end
```

Code Script

*** lab1_sin ***

```
clear; format short g;

%% 1. Plot intrinsic sine over [-pi, pi]
N = 100;
x = linspace(-pi, pi, N)';
y = sin(x);

figure(1); plot(x, y);
title('sin(x) over [-\pi,\pi]'); xlabel('x'); ylabel('sin(x)');

%% 2. Using function sin_scale(x,sf)
sf = 0.1;
y2 = sin_scale(x, sf);

figure(2); plot(x, y2);
title('Scaled sine: sin(x/0.1)');

%% 3. Multiple scaled sine functions
sf_all = [0.1 0.5 2];
sin_array = zeros(N, numel(sf_all));

for k = 1:numel(sf_all)
    sin_array(:,k) = sin_scale(x, sf_all(k));
end

figure(3); plot(x, sin_array);
title('Scaled Sine Functions'); legend(string(sf_all));

%% 4. Numerical Differentiation using FDT
ycos = cos(x);
[dy_est, x_d] = forward_difference_derivative(x, ycos);
dy_exact = -sin(x_d);

figure(4);
plot(x_d, dy_est, 'b', x_d, dy_exact, 'r--');
title('FDT derivative of cos(x)'); legend('Estimate','Exact');

%% 5. Numerical Integration
```

```

f1 = @(x) x.*sin(x);      % Integral from 0 to pi/2 = 1
I1_exact = 1;
f2 = @(x) 1./x;          % Integral from 1 to 2 = ln(2)
I2_exact = log(2);

%% 5a. Single-interval trapezoidal rule
I_trap1 = trap1(f1, 0, pi/2);
err_trap1 = abs((I_trap1 - I1_exact)/I1_exact)*100;

fprintf('\nSingle trapezoid (x sin x): %.12f  Error = %.6f%%\n', I_trap1, err_trap1);

%% 5b. Composite trapezoidal rule
n_vals = [5 50 100];
fprintf('\nComposite Trapezoidal Rule for 1/x:\n');
fprintf(' n      Estimate      Error(%%)\n');

for n = n_vals
    I = trapn(f2, 1, 2, n);
    err = abs((I - I2_exact)/I2_exact)*100;
    fprintf('%3d  %.12f  %.4e\n', n, I, err);
end

%% Find n for 6 decimal places (trapezoid)
target = 0.5e-6;
n = 1;
while true
    n = n + 1;
    if abs(trapn(f2, 1, 2, n) - I2_exact) < target
        break;
    end
end
fprintf('\nTrapezoid: 6-decimal accuracy at n = %d\n', n);

%% 6. Composite Simpson's rule
fprintf('\nSimpson's Rule for 1/x:\n');
fprintf(' n      Estimate      Error(%%)\n');

n_vals = [6 50 100]; % must be even
for n = n_vals
    I = simpsonn(f2, 1, 2, n);
    err = abs((I - I2_exact)/I2_exact)*100;
    fprintf('%3d  %.12f  %.4e\n', n, I, err);
end

%% Find n for 6 decimal places (Simpson)
n = 2;
while true
    n = n + 2;
    if abs(simpsonn(f2, 1, 2, n) - I2_exact) < target
        break;
    end
end

```

```
        end
    end
    fprintf('\nSimpson: 6-decimal accuracy at n = %d\n', n);

    disp('Done.');
```