

UWTSD

Alternative fuels and Combustion modelling

Task 1

ACHA 7002

2204298

07/12/2025

Contents

1	Plotting and differentiation.....	1
2	Trapezoidal Method.....	2
3	Composite	2
4	Simpsons.....	2
5	Integration results tables.....	4
5.1	Composite convergence iterations	4
5.2	Simpsons Convergence iterations.....	14
6	Appendices.....	14
6.1	Your code script and functions.....	17

1 Plotting and differentiation

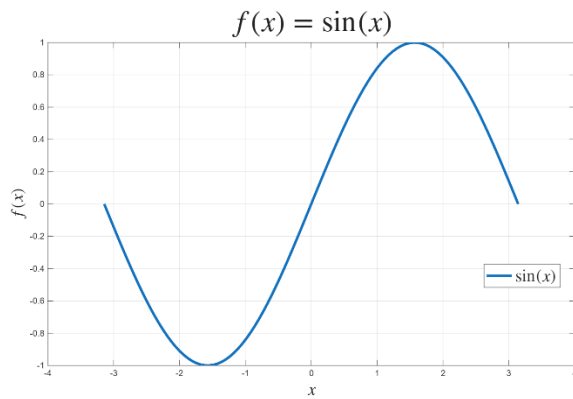


Figure 1 $y = \sin(x)$ plot

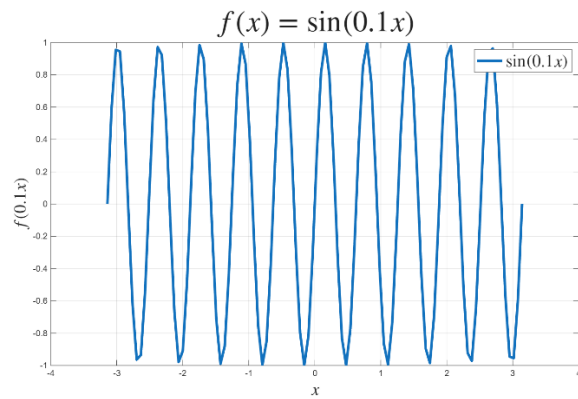


Figure 2 $y = \sin_scale(x, sf)$ plot (0.1 steps)

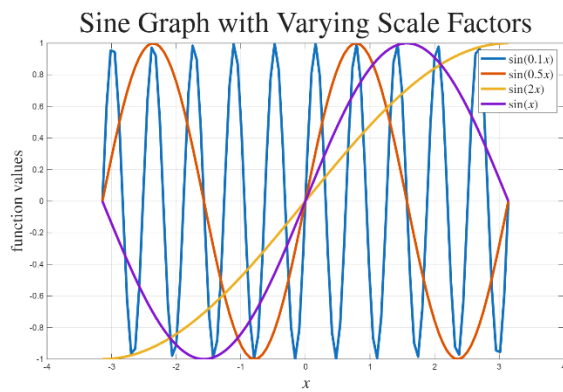


Figure 3 $y = \sin_scale(x, sf)$ plot ([0.1 0.5 2] steps)

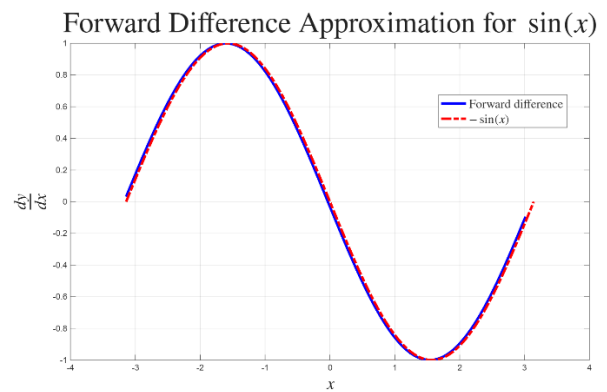


Figure 4 diff cos (x) to $-\sin(x)$

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

Equation 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
1.233701	1.000000	23.37%

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	Composite Trapezium			Converge 1e-6
'Result'	0.693147	0.695635	0.693172	0.693153	0.693148
'n iterations'	Na	5	50	100	250
'percentage difference'	Na	0.358905%	0.003607%	0.000902%	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

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

Equation 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
'Result'	0.693147	0.693386	0.693147	0.693147	0.693148
'n iterations'	NA	5	50	100	14
'percentage difference'	NA	0.034489%	0.000001%	0.000000%	0.000116

5 Integration results tables

5.1 Composite convergence iterations

Convergence Table:

Iteration	Calculated	Exact	Error
1	0.750000	0.693147	0.05685282
2	0.708333	0.693147	0.01518615
3	0.700000	0.693147	0.00685282
4	0.697024	0.693147	0.00387663
5	0.695635	0.693147	0.00248774
6	0.694877	0.693147	0.00173016
7	0.694419	0.693147	0.00127229
8	0.694122	0.693147	0.00097467
9	0.693918	0.693147	0.00077042
10	0.693771	0.693147	0.00062422
11	0.693663	0.693147	0.00051600
12	0.693581	0.693147	0.00043365
13	0.693517	0.693147	0.00036955
14	0.693466	0.693147	0.00031867
15	0.693425	0.693147	0.00027762
16	0.693391	0.693147	0.00024402
17	0.693363	0.693147	0.00021617
18	0.693340	0.693147	0.00019283
19	0.693320	0.693147	0.00017307
20	0.693303	0.693147	0.00015620
21	0.693289	0.693147	0.00014168

22	0.693276	0.693147	0.00012910
23	0.693265	0.693147	0.00011812
24	0.693256	0.693147	0.00010848
25	0.693247	0.693147	0.00009998
26	0.693240	0.693147	0.00009244
27	0.693233	0.693147	0.00008572
28	0.693227	0.693147	0.00007971
29	0.693221	0.693147	0.00007431
30	0.693217	0.693147	0.00006943
31	0.693212	0.693147	0.00006503
32	0.693208	0.693147	0.00006103
33	0.693205	0.693147	0.00005739
34	0.693201	0.693147	0.00005406
35	0.693198	0.693147	0.00005102
36	0.693195	0.693147	0.00004822
37	0.693193	0.693147	0.00004565
38	0.693190	0.693147	0.00004328
39	0.693188	0.693147	0.00004109
40	0.693186	0.693147	0.00003906
41	0.693184	0.693147	0.00003718
42	0.693183	0.693147	0.00003543
43	0.693181	0.693147	0.00003380
44	0.693179	0.693147	0.00003228
45	0.693178	0.693147	0.00003086
46	0.693177	0.693147	0.00002954

47	0.693175	0.693147	0.00002829
48	0.693174	0.693147	0.00002713
49	0.693173	0.693147	0.00002603
50	0.693172	0.693147	0.00002500
51	0.693171	0.693147	0.00002403
52	0.693170	0.693147	0.00002311
53	0.693169	0.693147	0.00002225
54	0.693169	0.693147	0.00002143
55	0.693168	0.693147	0.00002066
56	0.693167	0.693147	0.00001993
57	0.693166	0.693147	0.00001924
58	0.693166	0.693147	0.00001858
59	0.693165	0.693147	0.00001795
60	0.693165	0.693147	0.00001736
61	0.693164	0.693147	0.00001680
62	0.693163	0.693147	0.00001626
63	0.693163	0.693147	0.00001575
64	0.693162	0.693147	0.00001526
65	0.693162	0.693147	0.00001479
66	0.693162	0.693147	0.00001435
67	0.693161	0.693147	0.00001392
68	0.693161	0.693147	0.00001352
69	0.693160	0.693147	0.00001313
70	0.693160	0.693147	0.00001275
71	0.693160	0.693147	0.00001240

72	0.693159	0.693147	0.00001206
73	0.693159	0.693147	0.00001173
74	0.693159	0.693147	0.00001141
75	0.693158	0.693147	0.00001111
76	0.693158	0.693147	0.00001082
77	0.693158	0.693147	0.00001054
78	0.693157	0.693147	0.00001027
79	0.693157	0.693147	0.00001001
80	0.693157	0.693147	0.00000977
81	0.693157	0.693147	0.00000953
82	0.693156	0.693147	0.00000929
83	0.693156	0.693147	0.00000907
84	0.693156	0.693147	0.00000886
85	0.693156	0.693147	0.00000865
86	0.693156	0.693147	0.00000845
87	0.693155	0.693147	0.00000826
88	0.693155	0.693147	0.00000807
89	0.693155	0.693147	0.00000789
90	0.693155	0.693147	0.00000772
91	0.693155	0.693147	0.00000755
92	0.693155	0.693147	0.00000738
93	0.693154	0.693147	0.00000723
94	0.693154	0.693147	0.00000707
95	0.693154	0.693147	0.00000693
96	0.693154	0.693147	0.00000678

97	0.693154	0.693147	0.00000664
98	0.693154	0.693147	0.00000651
99	0.693154	0.693147	0.00000638
100	0.693153	0.693147	0.00000625
101	0.693153	0.693147	0.00000613
102	0.693153	0.693147	0.00000601
103	0.693153	0.693147	0.00000589
104	0.693153	0.693147	0.00000578
105	0.693153	0.693147	0.00000567
106	0.693153	0.693147	0.00000556
107	0.693153	0.693147	0.00000546
108	0.693153	0.693147	0.00000536
109	0.693152	0.693147	0.00000526
110	0.693152	0.693147	0.00000517
111	0.693152	0.693147	0.00000507
112	0.693152	0.693147	0.00000498
113	0.693152	0.693147	0.00000489
114	0.693152	0.693147	0.00000481
115	0.693152	0.693147	0.00000473
116	0.693152	0.693147	0.00000464
117	0.693152	0.693147	0.00000457
118	0.693152	0.693147	0.00000449
119	0.693152	0.693147	0.00000441
120	0.693152	0.693147	0.00000434
121	0.693151	0.693147	0.00000427

122	0.693151	0.693147	0.00000420
123	0.693151	0.693147	0.00000413
124	0.693151	0.693147	0.00000406
125	0.693151	0.693147	0.00000400
126	0.693151	0.693147	0.00000394
127	0.693151	0.693147	0.00000387
128	0.693151	0.693147	0.00000381
129	0.693151	0.693147	0.00000376
130	0.693151	0.693147	0.00000370
131	0.693151	0.693147	0.00000364
132	0.693151	0.693147	0.00000359
133	0.693151	0.693147	0.00000353
134	0.693151	0.693147	0.00000348
135	0.693151	0.693147	0.00000343
136	0.693151	0.693147	0.00000338
137	0.693151	0.693147	0.00000333
138	0.693150	0.693147	0.00000328
139	0.693150	0.693147	0.00000323
140	0.693150	0.693147	0.00000319
141	0.693150	0.693147	0.00000314
142	0.693150	0.693147	0.00000310
143	0.693150	0.693147	0.00000306
144	0.693150	0.693147	0.00000301
145	0.693150	0.693147	0.00000297
146	0.693150	0.693147	0.00000293

147	0.693150	0.693147	0.00000289
148	0.693150	0.693147	0.00000285
149	0.693150	0.693147	0.00000282
150	0.693150	0.693147	0.00000278
151	0.693150	0.693147	0.00000274
152	0.693150	0.693147	0.00000271
153	0.693150	0.693147	0.00000267
154	0.693150	0.693147	0.00000264
155	0.693150	0.693147	0.00000260
156	0.693150	0.693147	0.00000257
157	0.693150	0.693147	0.00000254
158	0.693150	0.693147	0.00000250
159	0.693150	0.693147	0.00000247
160	0.693150	0.693147	0.00000244
161	0.693150	0.693147	0.00000241
162	0.693150	0.693147	0.00000238
163	0.693150	0.693147	0.00000235
164	0.693150	0.693147	0.00000232
165	0.693149	0.693147	0.00000230
166	0.693149	0.693147	0.00000227
167	0.693149	0.693147	0.00000224
168	0.693149	0.693147	0.00000221
169	0.693149	0.693147	0.00000219
170	0.693149	0.693147	0.00000216
171	0.693149	0.693147	0.00000214

172	0.693149	0.693147	0.00000211
173	0.693149	0.693147	0.00000209
174	0.693149	0.693147	0.00000206
175	0.693149	0.693147	0.00000204
176	0.693149	0.693147	0.00000202
177	0.693149	0.693147	0.00000199
178	0.693149	0.693147	0.00000197
179	0.693149	0.693147	0.00000195
180	0.693149	0.693147	0.00000193
181	0.693149	0.693147	0.00000191
182	0.693149	0.693147	0.00000189
183	0.693149	0.693147	0.00000187
184	0.693149	0.693147	0.00000185
185	0.693149	0.693147	0.00000183
186	0.693149	0.693147	0.00000181
187	0.693149	0.693147	0.00000179
188	0.693149	0.693147	0.00000177
189	0.693149	0.693147	0.00000175
190	0.693149	0.693147	0.00000173
191	0.693149	0.693147	0.00000171
192	0.693149	0.693147	0.00000170
193	0.693149	0.693147	0.00000168
194	0.693149	0.693147	0.00000166
195	0.693149	0.693147	0.00000164
196	0.693149	0.693147	0.00000163

197	0.693149	0.693147	0.00000161
198	0.693149	0.693147	0.00000159
199	0.693149	0.693147	0.00000158
200	0.693149	0.693147	0.00000156
201	0.693149	0.693147	0.00000155
202	0.693149	0.693147	0.00000153
203	0.693149	0.693147	0.00000152
204	0.693149	0.693147	0.00000150
205	0.693149	0.693147	0.00000149
206	0.693149	0.693147	0.00000147
207	0.693149	0.693147	0.00000146
208	0.693149	0.693147	0.00000144
209	0.693149	0.693147	0.00000143
210	0.693149	0.693147	0.00000142
211	0.693149	0.693147	0.00000140
212	0.693149	0.693147	0.00000139
213	0.693149	0.693147	0.00000138
214	0.693149	0.693147	0.00000136
215	0.693149	0.693147	0.00000135
216	0.693149	0.693147	0.00000134
217	0.693149	0.693147	0.00000133
218	0.693148	0.693147	0.00000132
219	0.693148	0.693147	0.00000130
220	0.693148	0.693147	0.00000129
221	0.693148	0.693147	0.00000128

222	0.693148	0.693147	0.00000127
223	0.693148	0.693147	0.00000126
224	0.693148	0.693147	0.00000125
225	0.693148	0.693147	0.00000123
226	0.693148	0.693147	0.00000122
227	0.693148	0.693147	0.00000121
228	0.693148	0.693147	0.00000120
229	0.693148	0.693147	0.00000119
230	0.693148	0.693147	0.00000118
231	0.693148	0.693147	0.00000117
232	0.693148	0.693147	0.00000116
233	0.693148	0.693147	0.00000115
234	0.693148	0.693147	0.00000114
235	0.693148	0.693147	0.00000113
236	0.693148	0.693147	0.00000112
237	0.693148	0.693147	0.00000111
238	0.693148	0.693147	0.00000110
239	0.693148	0.693147	0.00000109
240	0.693148	0.693147	0.00000109
241	0.693148	0.693147	0.00000108
242	0.693148	0.693147	0.00000107
243	0.693148	0.693147	0.00000106
244	0.693148	0.693147	0.00000105
245	0.693148	0.693147	0.00000104
246	0.693148	0.693147	0.00000103

247	0.693148	0.693147	0.00000102
248	0.693148	0.693147	0.00000102
249	0.693148	0.693147	0.00000101
250	0.693148	0.693147	0.00000100

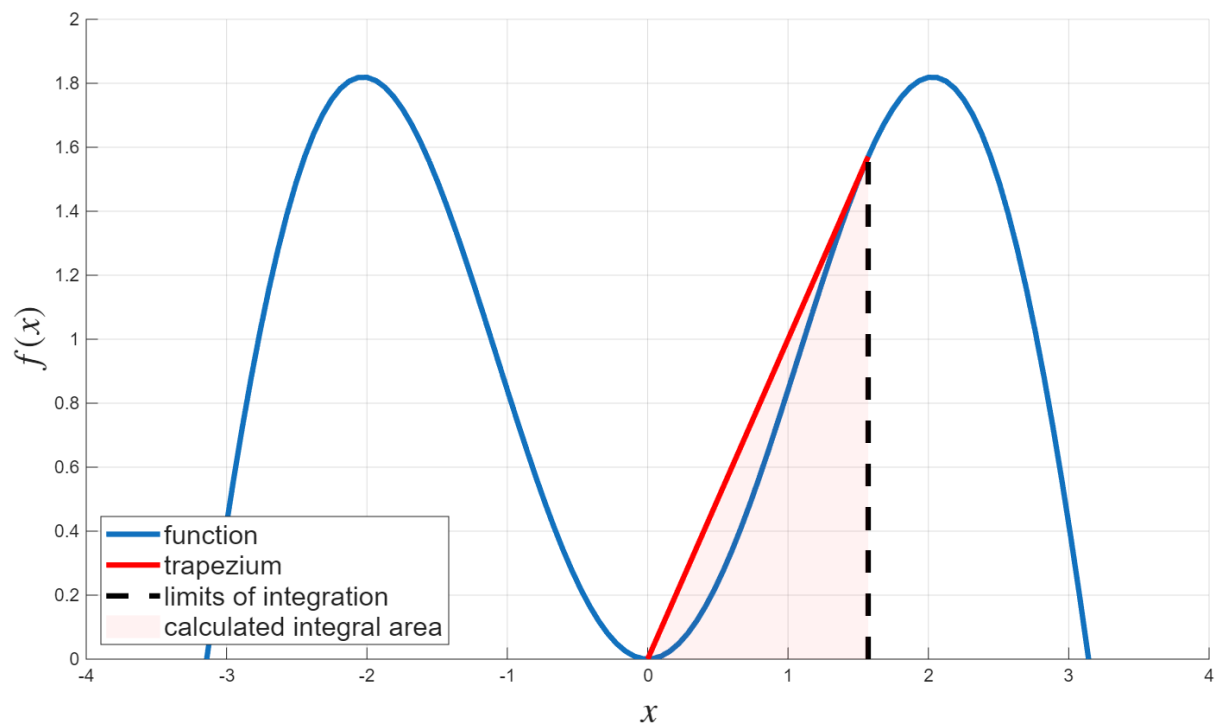
5.2 Simpsons Convergence iterations

Convergence Table (Simpson):

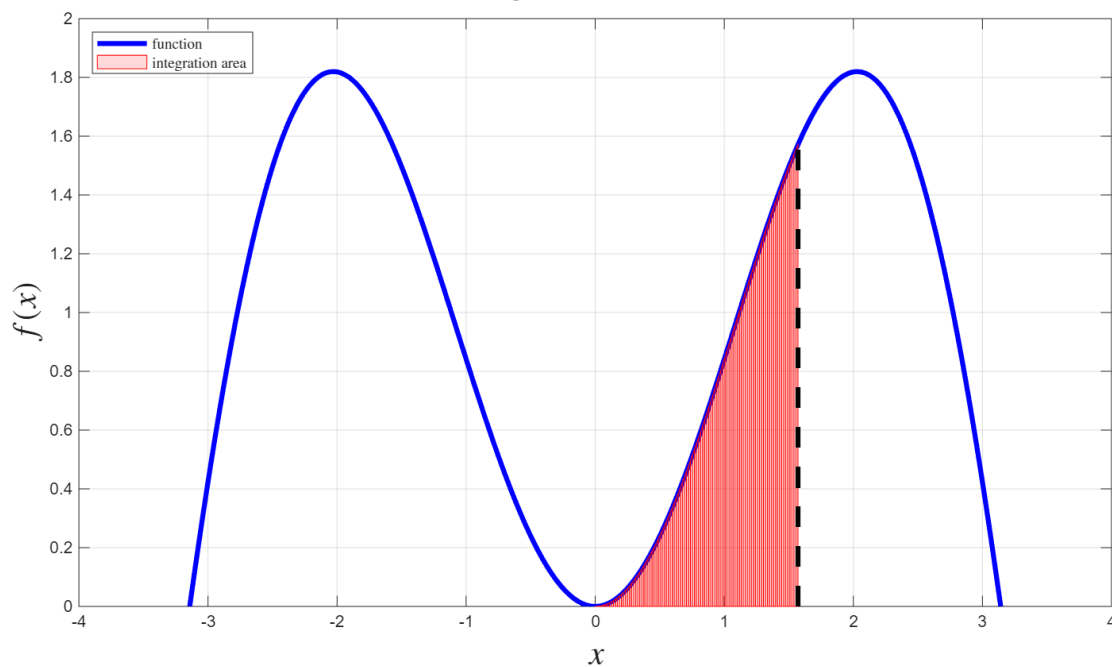
Iter	N	Calculated	Exact	Error
1	2	0.694444	0.693147	0.00129726
2	4	0.693254	0.693147	0.00010679
3	6	0.693170	0.693147	0.00002261
4	8	0.693155	0.693147	0.00000735
5	10	0.693150	0.693147	0.00000305
6	12	0.693149	0.693147	0.00000148
7	14	0.693148	0.693147	0.00000080

6 Appendices

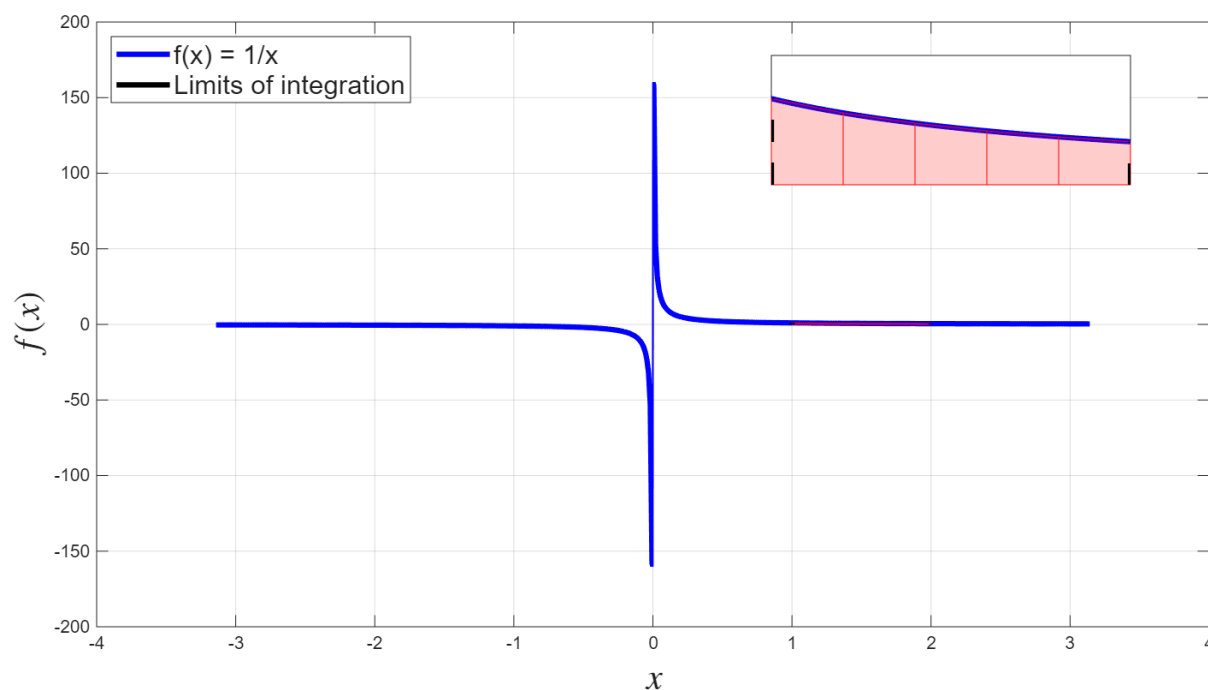
Single Trapezium for $\int_0^{\pi/2} x \sin(x) dx$



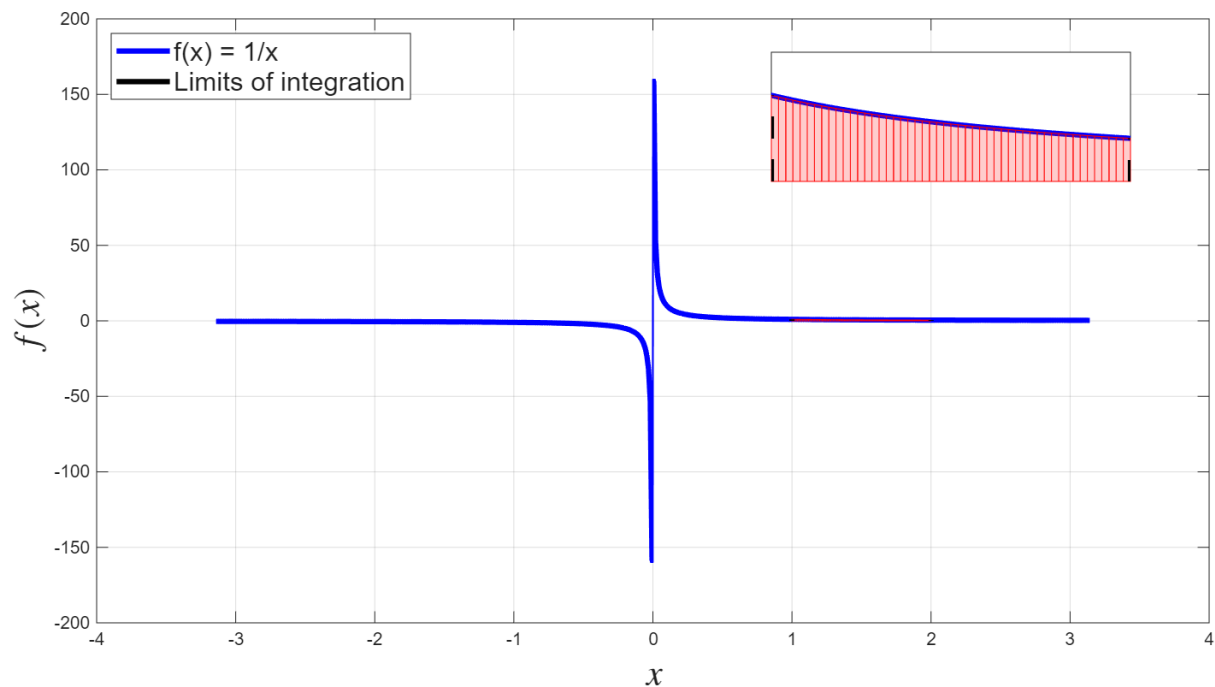
Trapezium Rule for $\int_0^{\pi/2} x \sin(x) dx$ (100 trapezia)



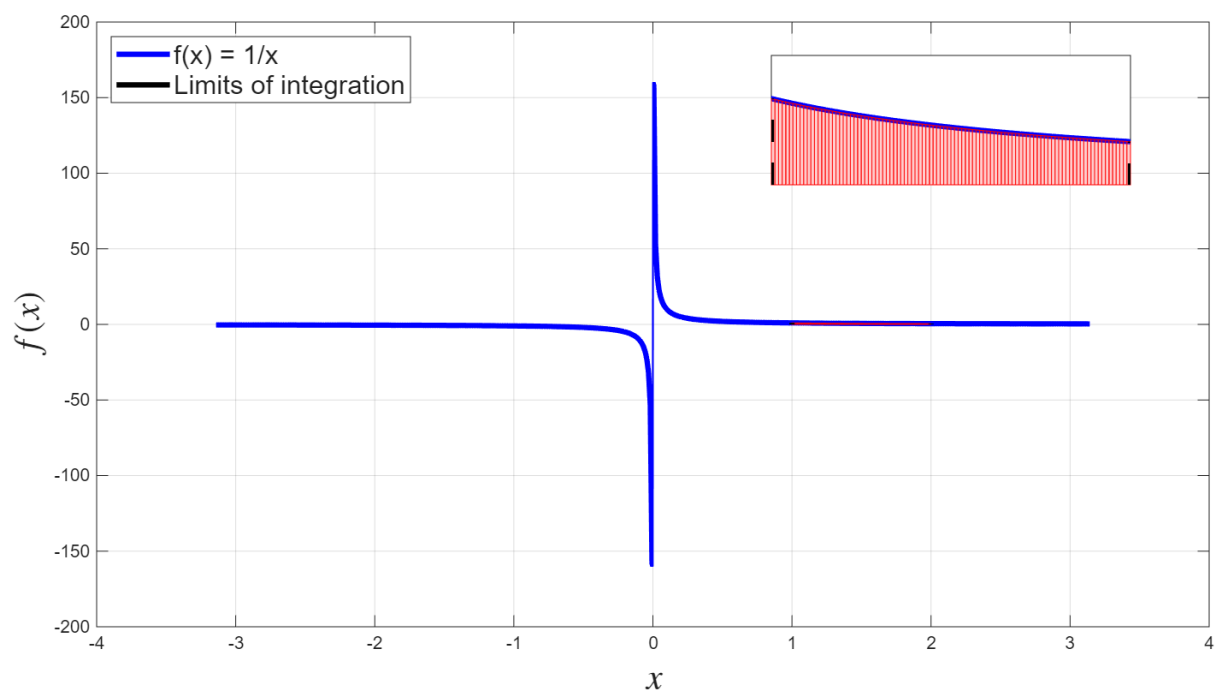
Trapezium Rule for $f(x) = \frac{1}{x}$ with 5



Trapezium Rule for $f(x) = \frac{1}{x}$ with 50



Trapezium Rule for $f(x) = \frac{1}{x}$ with 100



6.1 Your code script and functions

```

clc; %clears all the text from the Command Window
clear; %clears all variables from the workspace
close all; %closes the current figures

N = 100; % no. x values
x = linspace(-pi,pi,N); % creating the x-axis values

y = sin(x); % creating a sin wave

% plotting the sin wave
figure ();
plot(x,y,'linewidth',3)
title('$f(x)=\sin(x)$','Interpreter','latex','FontSize', 35)
xlabel('$x$','Interpreter','latex','FontSize', 20)
ylabel('$f(x)$','Interpreter','latex','FontSize', 20)
legend({'$\sin(x)$'}, 'Location','best','Interpreter','latex','FontSize', 20)
grid on;

sf = 0.1; % creating a scale factor
y = sin_scale(x,sf); % calling sinscale function

% plotting sin wave with a scale factor applied
figure()
plot(x,y,'linewidth',3)
title('$f(x)=\sin(0.1x)$','Interpreter','latex','FontSize', 35)
xlabel('$x$','Interpreter','latex','FontSize', 20)
ylabel('$f(0.1x)$','Interpreter','latex','FontSize', 20)
legend({'$\sin(0.1x)$'}, 'Location','northeast','Interpreter','latex','FontSize', 20)
grid on;

sf=[0.1 0.5 2];
N_sf = numel(sf); % Number of elements within sf (3)

% calling the function sinscale for all scale factors using a for loop
for ndx = 1: N_sf
    sin_array(:,ndx) = sin_scale(x,sf(ndx));
end

% plotting 3 sin curves with varying scale factors (with original sin curve)
figure(),
plot(x,sin_array,'linewidth',3); hold on
plot(x,sin(x),'linewidth',3); hold off
title('Sine Graph with Varying Scale Factors', 'Interpreter','latex','FontSize', 35);
legend( {'$\sin(0.1x)$', '$\sin(0.5x)$', '$\sin(2x)$', '$\sin(x)$'}, ...
        'Location','northeast', 'Interpreter','latex','FontSize', 15);
xlabel('$x$','Interpreter','latex','FontSize', 20);
ylabel('function values', 'Interpreter','latex','FontSize', 20);
grid on;

```

```

%% Diferentaing using forward difference

y = cos(x); % defining the sin curve

h = 2*pi / (N-1); % Step size

% Forward difference
fd = forward_diff (y,h);
fd(end)=-1;

% Plot for -sin(x) and the calculated diffreential of cos(x)
figure ()
plot(x(1:end-2), fd(1:end-2),'linewidth',3,'Color','blue'); hold on
plot(x, -sin(x), 'r-.', 'linewidth', 3); hold off
title('Forward Difference Approximation for  $\sin(x)$ ',
'Interpreter','latex','FontSize', 35);
legend({'Forward difference', ' $-\sin(x)$ '}, 'Interpreter','latex',...
'Location','best','FontSize', 15);
xlabel('$x$', 'Interpreter','latex','FontSize', 20);
ylabel('$\frac{dy}{dx}$', 'Interpreter','latex','FontSize', 20);
grid on;

%% numerical intergration
%trapizium method

fprintf('\n----- single Trapezium Rule for f(x) = x sin(x) over [0, pi/2] -----
\n');

x = linspace(-pi,pi,N); % creating the x-axis range

f = @(x) x.*sin(x);
y = f(x);
a = 0;
b = pi/2;

x = linspace(-pi,pi,N); % creating the x-axis range

fa = f(a); % y value of the function when x=a
fb = f(b); % y value of the function when x=b

% calling function trap1
I_calc = trap1(fa,fb,a,b);
fprintf('\ncalculated integral value = %.6f\n', I_calc);

x_fill = [a b b a]; % x-coordinates of the trapeziums vertices
y_fill = [fa fb 0 0]; % y-coordinates of the trapeziums vertices

% Ploting single trapezium with clear visulisation
figure();
hold on; grid on;
plot(x, y,'linewidth',3);
plot([a b], [fa fb],'linewidth',3,'Color', 'red'); % straight line between f(a)
and f(b)
plot([b b], [0 fb],'k--','linewidth',3); % vertical dotted line between x=b and
f(b)
fill(x_fill, y_fill, 'r', 'FaceAlpha', 0.05, 'EdgeColor','none');
title('Single Trapezium for  $\int_0^{\pi/2} x\sin(x)\,dx$ ', ...
'Interpreter','latex','FontSize', 35);
legend('function','trapezium', 'limits of integration',...

```

```

    'calculated integral area','latex','Location','southwest','FontSize', 15);
xlabel('$x$', 'Interpreter','latex','FontSize', 20);
ylabel('$f(x)$', 'Interpreter','latex','FontSize', 20);
grid on;
hold off;

syms x
expr = x*sin(x);
Iexact = double(int(expr, x, 0, pi/2)); % exact value for the integration
fprintf('analytical integral value = %.6f\n', Iexact);

[error, pct_diff] = errors(I_calc, Iexact);
fprintf('error between my calculated and the analytical integral values = %.2f\n',
error);
fprintf('percentage difference between my calculated and the analytical integral
values = %.2f\n', pct_diff);

%% composite trapezium method

fprintf('\n----- composite Trapezium Rule for f(x) = x sin(x) over [0, pi/2] -----
\n');

x = linspace(-pi,pi,N);
y = @(x) x.*sin(x);
Itrapcomp = trapcomp(y, a, b, N);
fprintf('\ncalculated integral value - composite trapesium - x sin(x)
= %.6f\n\n', Itrapcomp);

% Create points only between 0 and pi/2
x = linspace(a, b, N+1);
y = f(x);

% Plot the function over a larger range for context
x_full = linspace(-pi, pi, 500);
y_full = f(x_full);
figure ();
plot(x_full, y_full, 'b', 'LineWidth', 3); hold on
title('$Trapezium\ Rule for\ \int_{0}^{\pi/2} x\sin(x)\,dx\ (100\ trapezia)$', ...
'Interpreter','latex','FontSize',35);
xlabel('$x$', 'Interpreter','latex','FontSize', 20);
ylabel('$f(x)$', 'Interpreter','latex','FontSize', 20);
grid on

% Plot trapezia between 0 and pi/2
for i = 1:N
    % Vertices of the i-th trapezium
    x_trap = [x(i), x(i), x(i+1), x(i+1)];
    y_trap = [0, y(i), y(i+1), 0];
    fill(x_trap, y_trap, 'r', 'FaceAlpha', 0.15, 'EdgeColor', 'r');
end

% Draw a dashed line at the upper limit (pi/2)
plot([b b], [0 f(b)], 'k--', 'LineWidth', 3);
legend('function', 'integration area', 'Location', 'northwest', 'Interpreter',
'latex');
hold off

%% Composite Trapezium Rule for f(x) = 1/x over [1, 2]

```

```

fprintf('\n----- Composite Trapezium Rule for f(x) = 1/x over [1, 2] -----\n');

% Define the function and limits
f = @(x) 1 ./ x;
a = 1;
b = 2;

% Define the number of trapezia
N_values = [5, 50, 100];

% Exact value of the integral (ln(2))
I_exact = log(2);
fprintf('\n analytical integral value (ln(2)) = %.7f\n\n', I_exact);

% Loop through all N values to calculate the integral and plot all trapezia
for n = N_values
    I_calc = trapcomp(f, a, b, n);
    [error, pct_diff] = errors(I_calc, I_exact);
    fprintf('N = %3d --> Calculated = %.7f, Error = %.2e, Percentage diff
= %.7f %%\n', n, I_calc, error, pct_diff);
    % Plot the function between -pi and pi
    figure ();
    x_full = linspace(-pi, pi, 500);
    y_full = f(x_full);
    plot(x_full, y_full, 'b', 'LineWidth', 3); hold on;
    plot(x_full, 0, 'k', 'linewidth', 3);
    grid on;
    % Create trapezia points only between [1, 2]
    x = linspace(a, b, n+1);
    y = f(x);

    % Plot the trapezia
    for i = 1:n
        x_trap = [x(i), x(i), x(i+1), x(i+1)];
        y_trap = [0, y(i), y(i+1), 0];
        fill(x_trap, y_trap, 'r', 'FaceAlpha', 0.2, 'EdgeColor', 'r');
    end

    % Highlight integration region
    plot([a a], [0 f(a)], 'k--', 'linewidth', 3);
    plot([b b], [0 f(b)], 'k--', 'linewidth', 3);

    title(['Trapezium Rule for $f(x) = \frac{1}{x}$ with ', num2str(n)],
'Interpreter','latex','FontSize', 35);
    xlabel('$x$', 'Interpreter','latex','FontSize', 20);
    ylabel('$f(x)$', 'Interpreter','latex','FontSize', 20);
    legend('f(x) = 1/x', 'Limits of integration', 'Location',
'northwest','FontSize', 15);
    hold off;

    % Define zoom region
    mainAx = gca;
    xZoomRange = [0.75 2.25];
    yZoomRange = [-0.5 1.5];

    % Create inset axes
    insetPos = [0.6 0.62 0.25 0.15]; % [x y width height] in normalized units
    axInset = axes('Position', insetPos);

```

```

copyobj(get(mainAx, 'Children'), axInset);

xlim(axInset, [1 2]);
ylim(axInset, [0 1.5]);
grid(axInset, 'on');

% Optional: clean up inset appearance
set(axInset, 'Box', 'on', 'XTick', [], 'YTick', []);
end

%% Finding the number of sub-intervals (trapezium) needed to achieve an accuracy
to 6 decimal

fprintf('\n----- Finding the number of trapezium needed to achieve an accuracy to
6 decimal when integrating f(x) = 1/x between 1 and 2 -----\n' );

n = 1;
tol = 1e-6; % 6 decimal place tolerance
err = 1;

table_data = [];

while err > tol
    I_calc = trapcomp(f, a, b, n);
    [err, ~] = errors(I_calc, I_exact);
    table_data(n,:) = [n, I_calc, I_exact, err];

    n = n + 1;
end

[~, pct_diff] = errors(I_calc, I_exact);

fprintf('\nRequired number of trapezia for 6-decimal accuracy = %d\n',n);
fprintf('Result = %.7f\n',I_calc);
fprintf('analytical integral value (ln(2)) = %.7f\n', I_exact);
fprintf('Percentage difference = %d\n',pct_diff);

fprintf('\nConvergence Table:\n');
fprintf('%10s %15s %15s %15s\n', 'Iteration', 'Calculated', 'Exact', 'Error');

for i = 1:size(table_data,1)
    fprintf('%10d %15.6f %15.6f %15.8f\n', ...
        table_data(i,1), ... % Iteration
        table_data(i,2), ... % Calculated
        table_data(i,3), ... % Exact
        table_data(i,4)); % Error
end
%% Simpsons rule

fprintf('\n----- Simpsons Rule -----\n' );

a = 1; % initial A value
b = 2; % initial b value

Iexact = log(2);
fprintf('analytical integral value = %.6f\n\n', Iexact);

```

```

% loop for x values where initial guesses dont chnage

for i = 1:length(N_values) % looping through all N_Values (5, 50 and 100)
    N = N_values(i); % Current number of subintervals

    % Generate x values
    x_values(1) = a; % First value is the A value
    Isim = simp13eo(f,a,b,N); % using simpsons rule function
    [err, pct_diff] = errors(Isim, Iexact);
    fprintf('N = %3d --> Calculated = %.6f, Error = %.2e, Percentage diff
= %.6f %%\n', N, Isim, err, pct_diff);

end

%% Finding the number of sub-intervals (simpsons rule) needed to achieve an
accuracy to 6 decimal

fprintf('\n----- Finding the number of sub-intervals needed to achieve an accuracy
to 6 decimal when integrating f(x) = 1/x between 1 and 2 -----\n' );

Iexact = log(2);

N = 2;
tol = 1e-6; % 6 decimal place tolerance
err = 1;

iteration = 1;

while err > tol
    I_calc = simp13eo(f,a,b,N);
    [err, ~] = errors(I_calc, I_exact);
    % Store: iteration, N, calculated, exact, error
    simpson_data(iteration, :) = [iteration, N, I_calc, Iexact, err];

    N = N + 2;
    iteration = iteration + 1;
end

N_required = N - 2;

[~, pct_diff] = errors(I_calc, I_exact);

fprintf('\nRequired number of sub-intervals for 6-decimal accuracy = %d\n',
N_required);
fprintf('Result = %.6f\n', I_calc);
fprintf('analytical value = %.6f\n', Iexact);
fprintf('Percentage difference = %.6f %%\n\n', pct_diff);

% Print formatted convergence table
fprintf('\nConvergence Table (Simpson):\n');
fprintf('%10s %10s %15s %15s %15s\n', 'Iter', 'N', 'Calculated', 'Exact',
'Error');

for i = 1:size(simpson_data,1)
    fprintf('%10d %10d %15.6f %15.6f %15.8f\n', ...
        simpson_data(i,1), ... % iteration
        simpson_data(i,2), ... % N
        simpson_data(i,3), ... % calculated
        simpson_data(i,4), ... % exact

```

```
        simpson_data(i,5));    % error (8 dp)
end
```

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

```
y = sin(x/sf);
```

```
end
```

```
function fd = forward_diff(y,h)
```

```
    N = 100;
```

```
    fd = zeros(1, N);
```

```
    for i = 1:N-1
```

```
        fd(i) = (y(i+1) - y(i)) / h;
```

```
    end
```

```
    fd(N) = fd(N-1);
```

```
end
```

```
function I = trap1(fa,fb,a,b)
```

```
    if a>b
```

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

```
end
```

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

```
% I = h * (sum(y) - 0.5*(y(1) + y(end)));
```

```
I = 0.5*(fa+fb) * (b-a);
```

```
end
```

```
function I = trapcomp(f,a,b,n)
```

```
    if a>b
```

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

```
    end
```

```
    h = (b - a) / n;          % width of each trapezium
```

```
    x = a:h:b;                % create n+1 points
```

```
    y = f(x);                 % evaluate function
```

```
% Trapezium rule formula
```

```
I = h * ( sum(y) - 0.5*(y(1) + y(end)) );
```

```
end
```

```
function [error,pct_diff] = errors(I_calc, I_exact)
```

```
error = abs(I_calc - I_exact);

pct_diff = (I_calc - I_exact) / I_exact * 100;

end

function I = simp13eo(f,a,b,N)

    h = (b - a) / N;

    x = a:h:b;

    y = f(x);

    if mod(N,2) == 0

        % Even number of subintervals use full Simpson's 1/3

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

    else

        % Odd number use Simpson's 1/3 to first n-1 subintervals,

        I1 = h/3 * (y(1) + 4*sum(y(2:2:end-2)) + 2*sum(y(3:2:end-3)) + y(end-1));

        I2 = h/2 * (y(end-1) + y(end));

        I = I1 + I2;

    end

end
```