

UWTSD

Alternative fuels and Combustion modelling

Task 2

ACHA 7002

2204298

Date

Contents

1	Bisection	1
1.1	$fx = \sin x \sinh(x)$	1
1.2	$fx = R^2 - x^{21/2} - x \tan(x)$	2
2	Newton's method	2
2.1	Newton b $fx = ex - (1 + x + (x^2/2))$	3
2.2	Newton c $fx = 10 \log x - x$	4
3	Intersect plot $fx = 100e - x - 5 \sin(\pi x/2)$	4
4	Bracketing	5
4.1	Bisect $fx = 100e - x - 5 \sin(\pi x/2)$	5
4.2	Newton $fx = 100e - x - 5 \sin(\pi x/2)$	5
5	Appendices	6
5.1	Convergence table Bisection $fx = \sin x \sinh(x)$	7
5.2	Convergence table Bisection $fx = R^2 - x^{21/2} - x \tan(x)$	7
5.3	Convergence table Newton b $fx = ex - (1 + x + (x^2/2))$	8
5.4	Convergence table Newton c $fx = 10 \log x - x$	8
5.5	Convergence table Bisect $fx = 100e - x - 5 \sin(\pi x/2)$	9
5.6	Convergence table Newton $fx = 100e - x - 5 \sin(\pi x/2)$	10
5.7	Your code and functions	11

1 Bisection

A function was created based on equation 1.1 to solve roots using the bisection method

$$x_2 = (x_1 + x_3) / 2 \quad \text{Equation 1-1}$$

1.1 $f(x) = \sin(x) \sinh(x)$

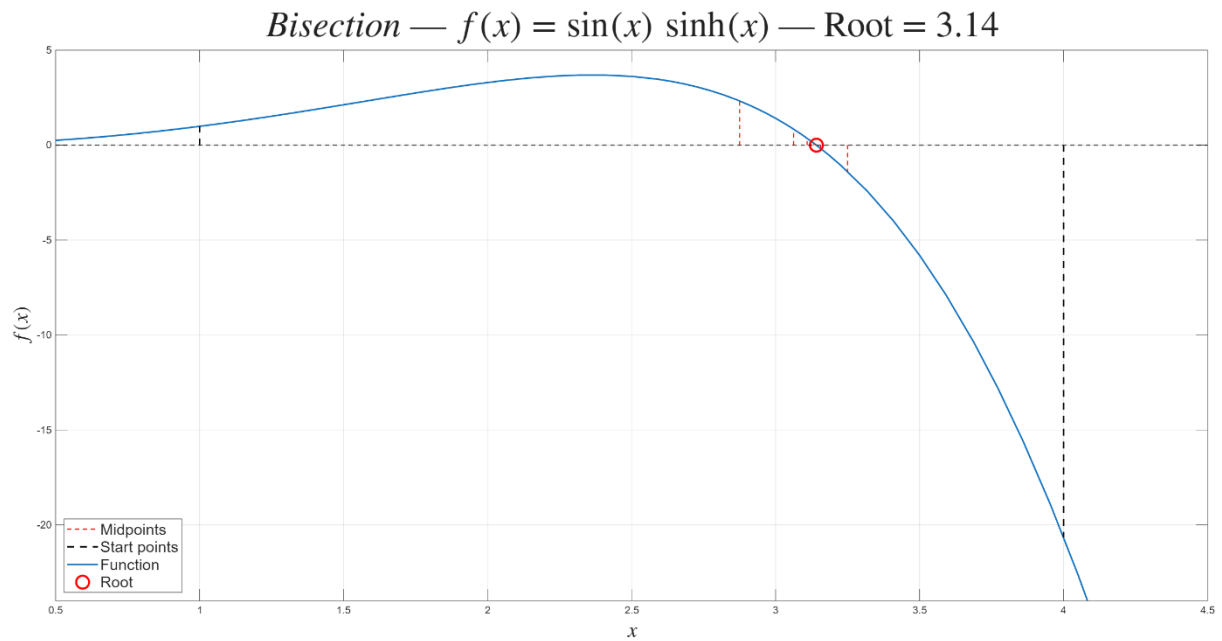


Figure 1-1 this shows the function labelled in the legend and the root

$$1.2 \quad f(x) = (R^2 - x^2)^{1/2} - x \tan(x)$$

Bisection — $f(x) = \sqrt{R^2 - x^2} - x \tan(x)$ — Root = 4.27

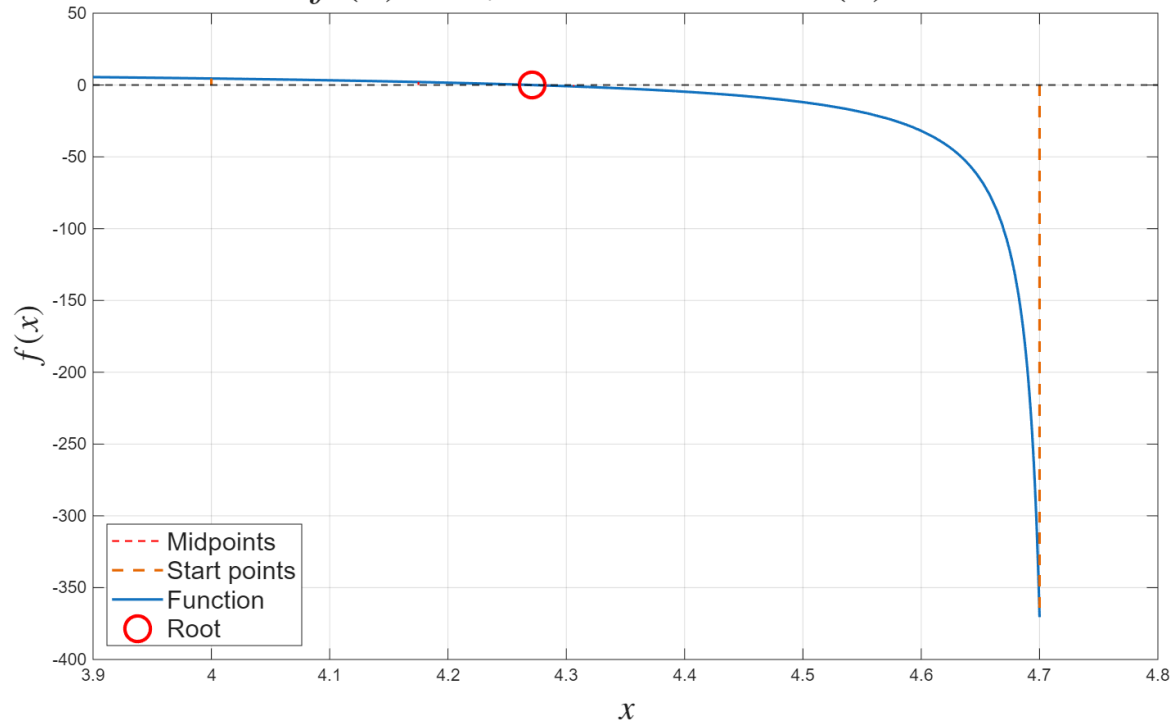


Figure 1-2 this shows the function labelled in the legend and the root

2 Newton's method

A function was made from equation 2.1 to evaluate roots with the newton method to use this method, functions and their derivative functions were created.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad \text{Equation 2-1}$$

2.1 Newton b $f(x) = e^x - (1 + x + (x^2/2))$

NewtonRaphson — $f(x) = e^x - (1 + x + \frac{1}{2}x^2)$ — Root = 0.00

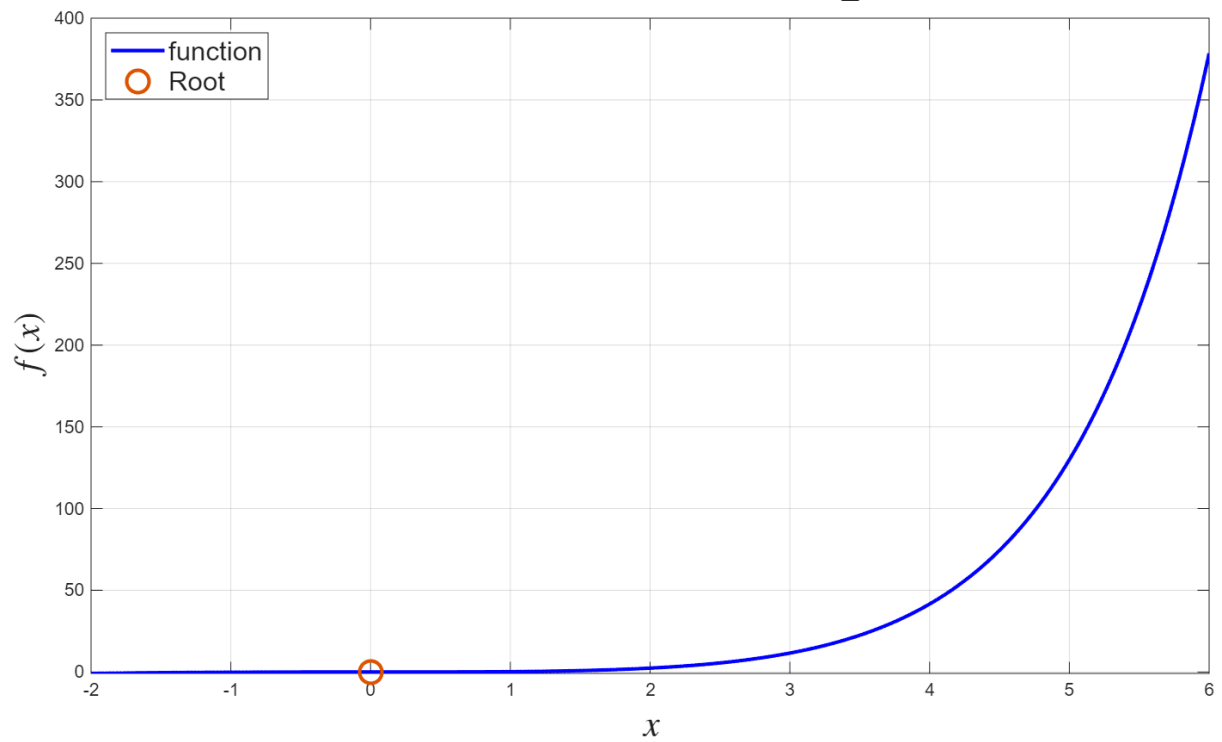


Figure 2-1 this shows the function labelled in the legend and the root

2.2 Newton c $f(x) = 10 \log(x) - x$

NewtonRaphson — $f(x) = 10 \log(x) - x$ — Root = 1.12

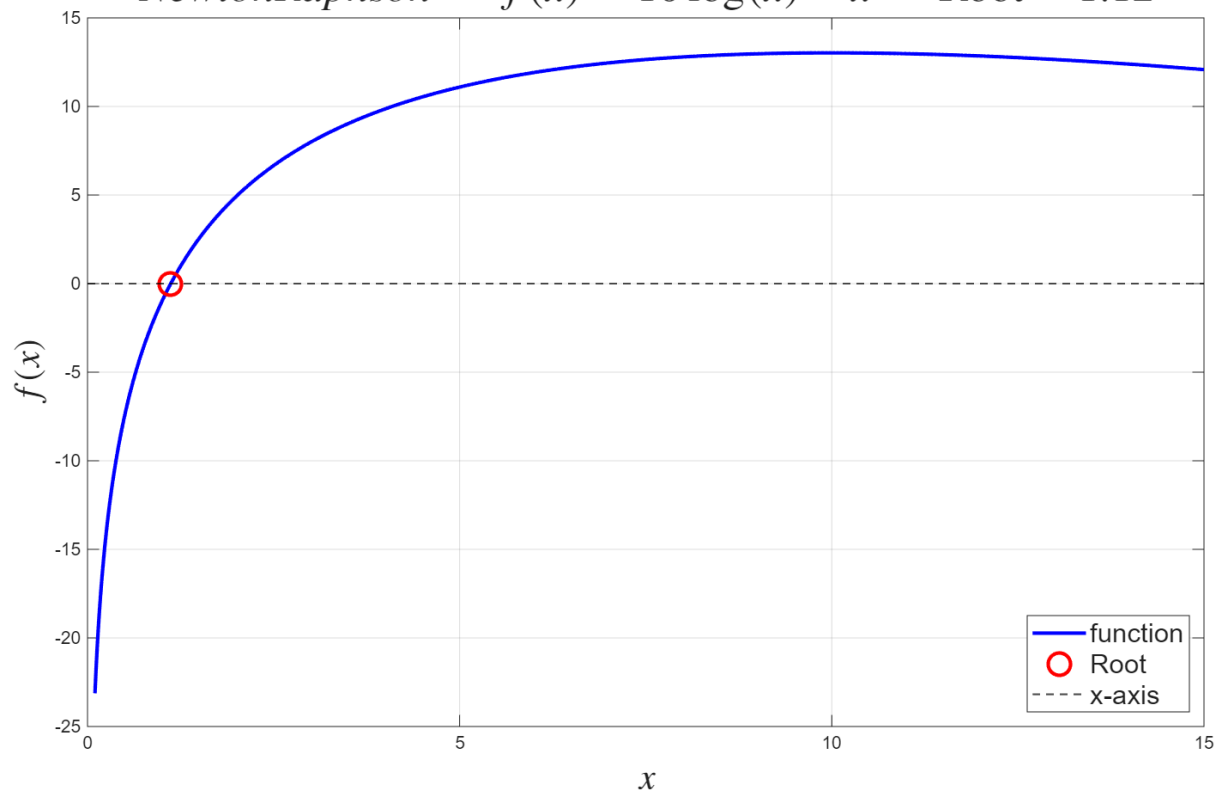


Figure 2-2 this shows the function labelled in the legend and the root

3 Intersect plot $f(x) = 100e^{-x} - 5 \sin(\pi x/2)$

Intersection of $f_1(x)$ and $f_2(x)$

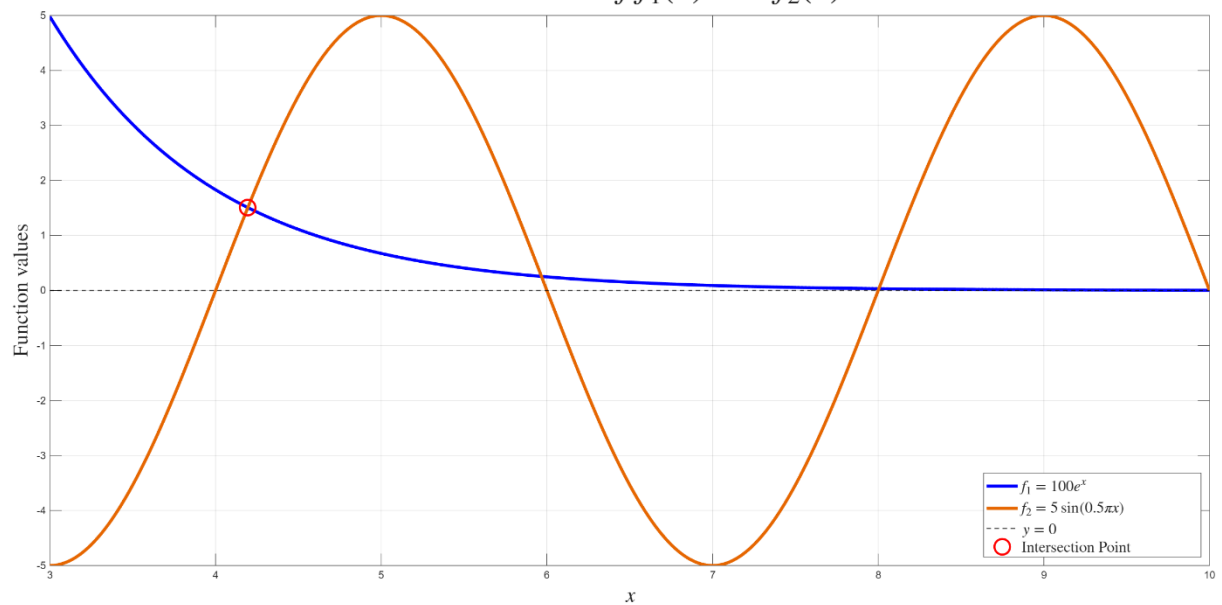


Figure 3-1 plot of function intersection

4 Bracketing

4.1 Bisect $f(x) = 100e^{-x} - 5\sin(\pi x/2)$

Bisection Method with all Roots marked

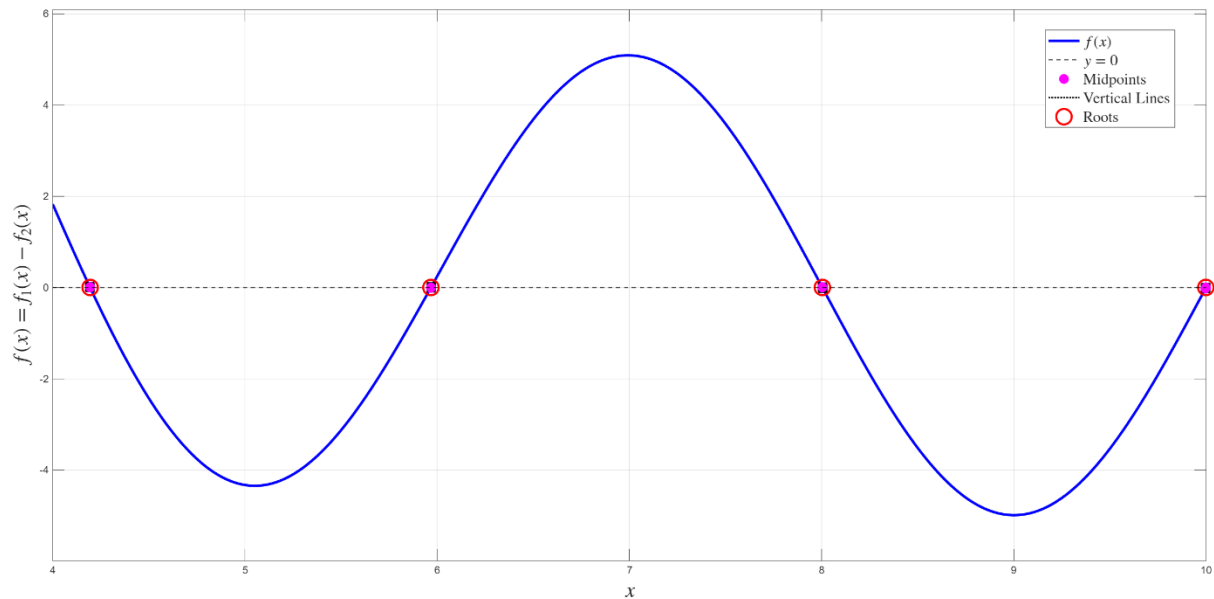


Figure 4-1 multiple roots Bisect for functions shown in legend

4.2 Newton $f(x) = 100e^{-x} - 5\sin(\pi x/2)$

Newton Raphson Method with all Roots marks

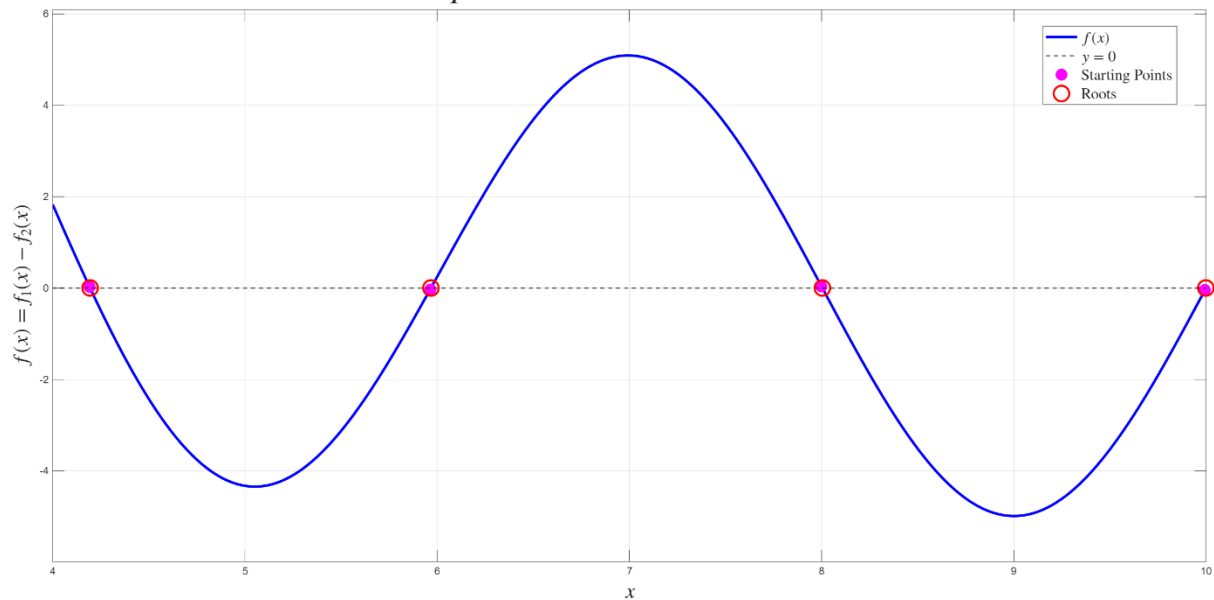
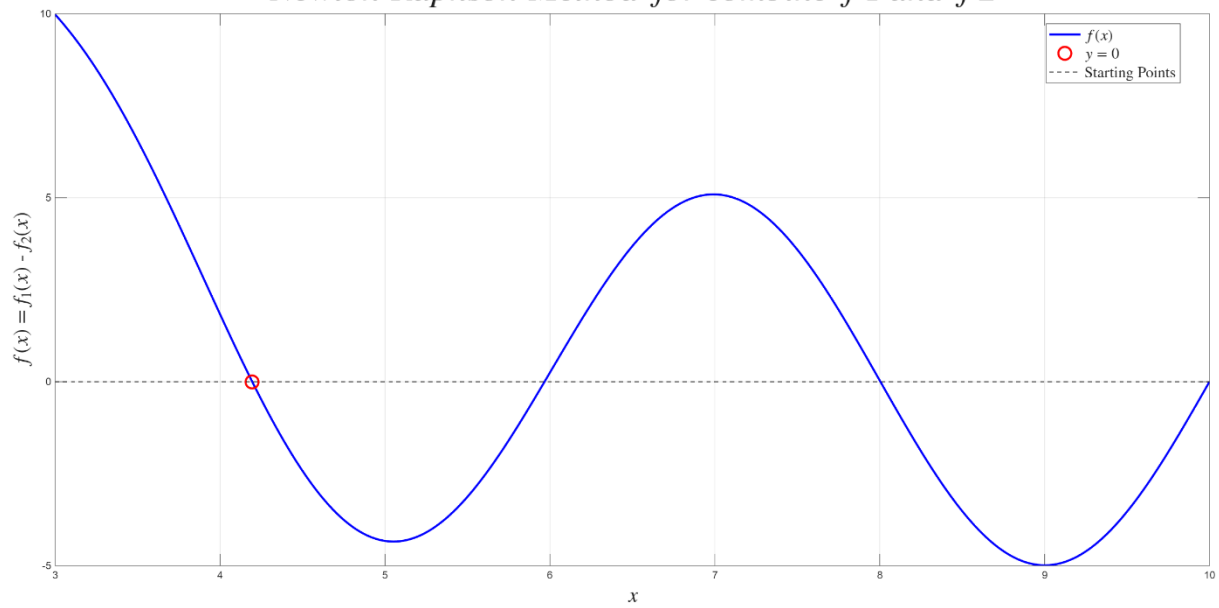


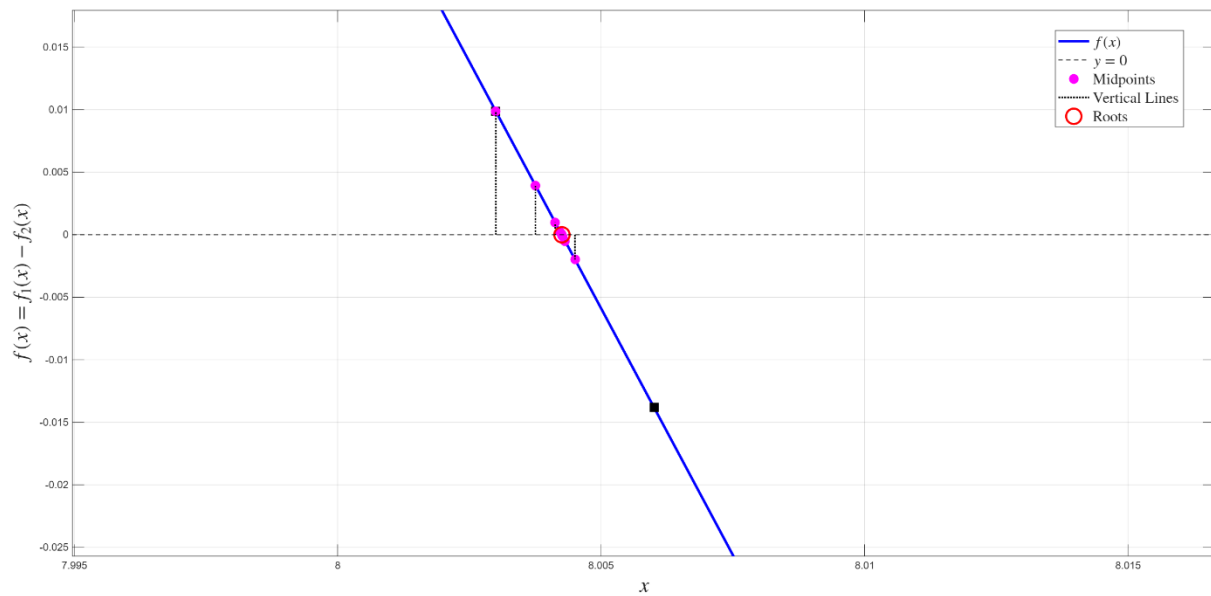
Figure 4-2 multiple roots Newton for functions shown in legend

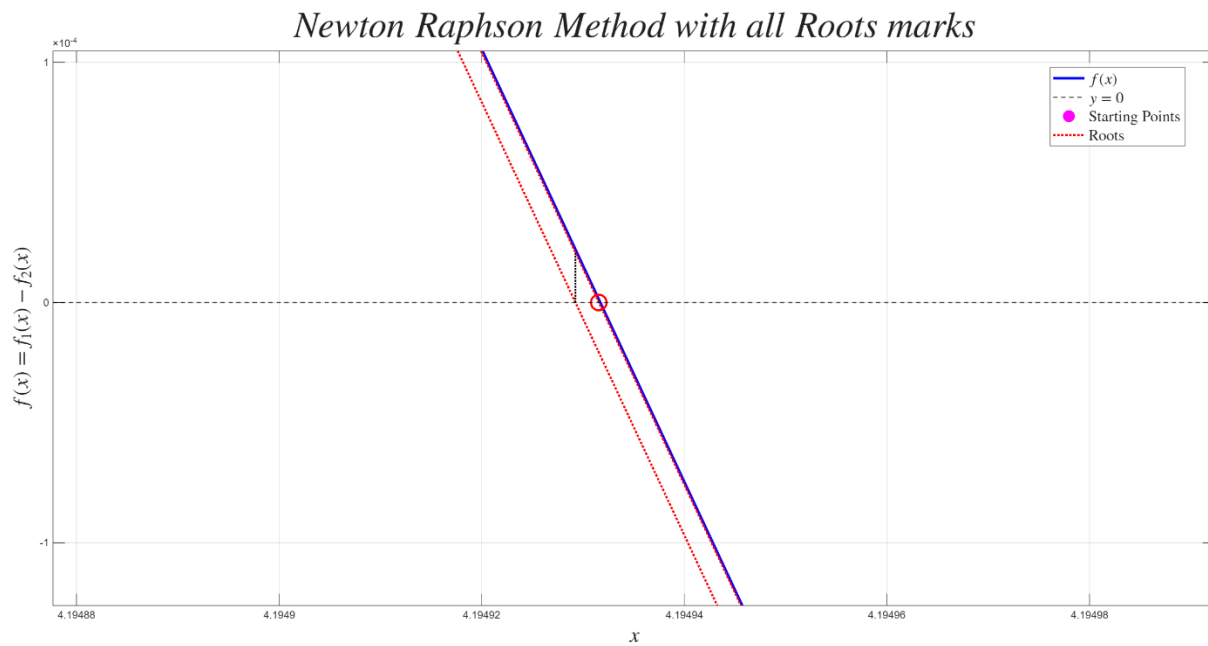
5 Appendices

Newton Raphson Method for combine f_1 and f_2



Bisection Method with all Roots marked





5.1 Convergence table Bisection $f(x) = \sin(x) \sinh(x)$

Iter	Exact	Calc	Error
1	3.141593	3.250000	0.108407
2	3.141593	2.875000	0.266593
3	3.141593	3.062500	0.079093
4	3.141593	3.156250	0.014657
5	3.141593	3.109375	0.032218
6	3.141593	3.132813	0.008780
7	3.141593	3.144531	0.002939
8	3.141593	3.138672	0.002921
9	3.141593	3.141602	0.000009
10	3.141593	3.140137	0.001456
11	3.141593	3.140869	0.000724
12	3.141593	3.141235	0.000357
13	3.141593	3.141418	0.000174
14	3.141593	3.141510	0.000083
15	3.141593	3.141556	0.000037
16	3.141593	3.141579	0.000014
17	3.141593	3.141590	0.000003
18	3.141593	3.141596	0.000003
19	3.141593	3.141593	0.000000

5.2 Convergence table Bisection $f(x) = (R^2 - x^2)^{1/2} - x \tan(x)$

Iter	Exact	Calc	Error
1	4.271096	4.175000	0.096096

2	4.271096	4.262500	0.008596
3	4.271096	4.306250	0.035154
4	4.271096	4.284375	0.013279
5	4.271096	4.273438	0.002342
6	4.271096	4.267969	0.003127
7	4.271096	4.270703	0.000393
8	4.271096	4.272070	0.000975
9	4.271096	4.271387	0.000291
10	4.271096	4.271045	0.000051
11	4.271096	4.271216	0.000120
12	4.271096	4.271130	0.000035
13	4.271096	4.271088	0.000008
14	4.271096	4.271109	0.000013
15	4.271096	4.271098	0.000003
16	4.271096	4.271093	0.000003
17	4.271096	4.271096	0.000000

5.3 Convergence table Newton b $f(x) = e^x - (1 + x + (x^2/2))$

Iter	Exact	Calc	Error
-----	-----	-----	-----
0	-0.000006	5.000000	-
1	-0.000006	4.087773	4.087779
2	-0.000006	3.241020	3.241026
3	-0.000006	2.487381	2.487387
4	-0.000006	1.849522	1.849528
5	-0.000006	1.337187	1.337192
6	-0.000006	0.944907	0.944913
7	-0.000006	0.656151	0.656157
8	-0.000006	0.449883	0.449889
9	-0.000006	0.305706	0.305712
10	-0.000006	0.206451	0.206457
11	-0.000006	0.138834	0.138840
12	-0.000006	0.093096	0.093102
13	-0.000006	0.062307	0.062312
14	-0.000006	0.041646	0.041652
15	-0.000006	0.027812	0.027818
16	-0.000006	0.018563	0.018569
17	-0.000006	0.012385	0.012391
18	-0.000006	0.008261	0.008267
19	-0.000006	0.005509	0.005515
20	-0.000006	0.003674	0.003679

5.4 Convergence table Newton c $f(x) = 10 \log(x) - x$

Iter	Exact	Calc	Error
-----	-----	-----	-----
0	1.118326	1.000000	-
1	1.118326	1.111111	0.007214
2	1.118326	1.118299	0.000026
3	1.118326	1.118326	0.000000

5.5 Convergence table Bisect $f(x) = 100e^{-x} - 5\sin(\pi x/2)$

Tolerance = 1e-6 for my bracketing section.

----- Bisect Convergence Tables -----

Iter	Exact	calc	error

0	4.194932	4.193597	0.001335
1	4.194932	4.194347	0.000585
2	4.194932	4.194722	0.000210
3	4.194932	4.194910	0.000022
4	4.194932	4.195004	0.000072
5	4.194932	4.194957	0.000025
6	4.194932	4.194933	0.000001
7	4.194932	4.194922	0.000010
8	4.194932	4.194928	0.000004
9	4.194932	4.194930	0.000001
10	4.194932	4.194932	0.000000

Root 2 (Bracket [5.965983 , 5.968984])

Iter	Exact	calc	error

0	5.967378	5.967484	0.000106
1	5.967378	5.966733	0.000645
2	5.967378	5.967109	0.000270
3	5.967378	5.967296	0.000082
4	5.967378	5.967390	0.000012
5	5.967378	5.967343	0.000035
6	5.967378	5.967366	0.000012
7	5.967378	5.967378	0.000000

Root 3 (Bracket [8.004002 , 8.007004])

Iter	Exact	calc	error

0	8.004254	8.005503	0.001249
1	8.004254	8.004752	0.000498
2	8.004254	8.004377	0.000123
3	8.004254	8.004190	0.000064
4	8.004254	8.004283	0.000029
5	8.004254	8.004236	0.000018

6	8.004254	8.004260	0.000006
7	8.004254	8.004248	0.000006
8	8.004254	8.004254	0.000000

Root 4 (Bracket [9.996998 , 10.000000])

Iter	Exact	calc	error

0	9.999423	9.998499	0.000923
1	9.999423	9.999250	0.000173
2	9.999423	9.999625	0.000202
3	9.999423	9.999437	0.000015
4	9.999423	9.999343	0.000079
5	9.999423	9.999390	0.000032
6	9.999423	9.999414	0.000009
7	9.999423	9.999425	0.000003
8	9.999423	9.999420	0.000003
9	9.999423	9.999423	0.000000

5.6 Convergence table Newton $f(x) = 100e^{-x} - 5\sin(\pi x/2)$

----- Newtons Convergence Tables -----

Iter	Exact	calc	error

0	4.194932	4.192096	0.002836
1	4.194932	4.194929	0.000002
2	4.194932	4.194932	0.000000

Root 2 (Bracket [5.965983 , 5.968984])

Iter	Exact	calc	error

0	5.967379	5.965983	0.001396
1	5.967379	5.967379	0.000000

Root 3 (Bracket [8.004002 , 8.007004])

Iter	Exact	calc	error

0	8.004253	8.004002	0.000251
1	8.004253	8.004253	0.000000

Root 4 (Bracket [9.996998 , 10.000000])

Iter	Exact	calc	error

0	9.999422	9.996998	0.002423
1	9.999422	9.999422	0.000000

5.7 Your code and functions

```
clc; clear; close all;
```

```
%% Bisection method
```

```
fprintf('\n----- Bisection method for f(x) = sin(x) sinh(x) -----\n');
```

```
f = @(x) sin(x).*sinh(x);
```

```
x1 = 1;
```

```
x3 = 4;
```

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

```
imax = 50;
```

```
root_exact = fzero(f, 3.1);    % Analytical result
```

```
[root, iterations, mid_memory, ~, ~] = bisect(f, x1, x3, tol, root_exact, imax);    %  
calculated result
```

```

fprintf('Analytical root  $\approx$  %.6f\n', root_exact);

fprintf('calculated root = %.6f\n', root);

fprintf('Number of bisections used = %d\n', iterations);


% plot

figure ();

h_func = fplot(f, [0.5 4.5], 'LineWidth', 1.5); hold on; % storing h plot using a handle
(h_...)

xlim([x1-0.5, x3+0.5]);

ylim([-24, 5]);

yline(0, 'k--', 'LineWidth', 1);


for k = 1:length(mid_memory)

    mk = mid_memory(k); % midpoint at iteration k

    h_mid = plot([mk mk], [0 f(mk)], 'r--', 'LineWidth', 1); % plot dotted line at midpoint k
end


% Add dotted lines at x1 and x3

h_Start=plot([x1 x1], [0 f(x1)], 'k--', 'LineWidth', 1.5);

plot([x3 x3], [0 f(x3)], 'k--', 'LineWidth', 1.5);


% Root marker

h_root = plot(root, f(root), 'ro', 'MarkerSize', 12.5, 'LineWidth', 2);


title('$Bisection$ --- $f(x) = \sin(x) \backslash, \sinh(x)$ --- Root = 3.14', 'Interpreter',
'latex', 'FontSize', 35);

```

```

xlabel('$x$', 'Interpreter', 'latex', 'FontSize', 20);

ylabel('$f(x)$', 'Interpreter', 'latex', 'FontSize', 20);

grid on

legend([h_mid, h_Start, h_func, h_root], 'Midpoints', 'Start points', 'Function', 'Root',
'Location', 'southwest', 'FontSize', 15);

% --- Iteration table ---

ExactValue_all = root_exact * ones(iterations,1);

Calc_all = mid_memory(:);

Error_all = abs(ExactValue_all - Calc_all);

% Round to 6 dp

ExactValue_all = round(ExactValue_all, 6);

Calc_all = round(Calc_all, 6);

Error_all = round(Error_all, 6);

fprintf('\nConvergence table Bisection f(x) sin(x).*sinh(x)\n');

fprintf('%10s %12s %12s %12s\n', 'Iter', 'Exact', 'Calc', 'Error');

fprintf('%10s %12s %12s %12s\n', '-----', '-----', '-----', '-----');

for k = 1:(iterations)

    fprintf('%10d %12.6f %12.6f %12.6f\n', ...

        k, ExactValue_all(k), Calc_all(k), Error_all(k));

```

end

```
%% Bisection method for  $f(x) = \sqrt{R^2 - x^2} - (x \cdot \tan(x))$ 
```

```
fprintf('\n----- Bisection method for  $f(x) = \sqrt{R^2 - x^2} - (x \cdot \tan(x))$  ----- \n');
```

```
R = 10;
```

```
f = @(x) sqrt(R^2 - x.^2) - (x.*tan(x));
```

```
x1 = 4;
```

```
x3 = 4.7;
```

```
root_exact = fzero(f, 4.4);
```

```
[root, iterations, mid_memory, ~, ~] = bisect(f, x1, x3, tol, root_exact, imax);
```

```
fprintf('Analytical Root = %.6f\n', root_exact);
```

```
fprintf('calculated root = %.6f\n', root);
```

```
fprintf('Number of bisections used = %d\n', iterations);
```

```
% plot
```

```
figure 0;
```

```
h_func = fplot(f, [3.9 4.7], 'LineWidth', 1.5); hold on;
```

```
xlim([x1-0.1, x3+0.1]);
```

```
ylim([-400, 50]);
```

```
h_root = plot(root, f(root), 'ro', 'MarkerSize', 15, 'LineWidth', 2);
```

```

yline(0, 'k--', 'LineWidth', 1);

% Midpoints

for k = 1:length(mid_memory)

    mk = mid_memory(k);

    h_mid = plot([mk mk], [0 f(mk)], 'r--', 'LineWidth', 1);

end

% Starting points

h_start1 = plot([x1 x1], [0 f(x1)], '--', 'Color', [0.9 0.4 0], 'LineWidth', 1.5);
h_start2=plot([x3 x3], [0 f(x3)], '--', 'Color', [0.9 0.4 0], 'LineWidth', 1.5);

title('$Bisection$ --- $f(x)=\sqrt{R^2 - x^2} - x\tan(x)$ --- Root = 4.27', 'Interpreter',
'latex','FontSize',30);

xlabel('$x$', 'Interpreter', 'latex', 'FontSize', 20);

ylabel('$f(x)$', 'Interpreter', 'latex', 'FontSize', 20);

legend([h_mid, h_start1, h_func, h_root], 'Midpoints', 'Start points', 'Function', 'Root',
'Location', 'southwest','fontsize',15);

grid on;

% -----Iteration Table -----

N = length(mid_memory); % number of stored midpoints

Iteration = (1:N)';

```

```

Calc_all = mid_memory(:);

Exact_all = root * ones(N,1);      % use final root as reference

Error_all = abs(Calc_all - root);   % standard bisection error


% Round to 6 dp

Calc_all = round(Calc_all, 6);

Exact_all = round(Exact_all, 6);

Error_all = round(Error_all, 6);


fprintf('\nConvergence table Bisection f(x)=(R^2-x^2)^(1/2)-xtan(x)\n');

fprintf('%10s %12s %12s %12s\n', 'Iter', 'Exact', 'Calc', 'Error');

fprintf('%10s %12s %12s %12s\n', ...

        '-----', '-----', '-----', '-----');


for k = 1:N

    fprintf('%10d %12.6f %12.6f %12.6f\n', ...

        k, Exact_all(k), Calc_all(k), Error_all(k));

end


%% Newton Raphson method


fprintf('\n----- Newton for f(x) = exp(x) - (1 + x + (0.5*x.^2)) -----\n');


imax = 20;

```

```
tol = 1e-6;

f1 = @(x) exp(x) - (1 + x + (0.5*x.^2)); % f(x)

df1 = @(x) exp(x) - 1 - x; % f'(x)

x0 = 5; % initial guess

root_exact1 = fzero(f1, x0);

[root1, ~, ~, ~, ~] = newton(x0, f1, df1, imax, tol, root_exact1);

fprintf('Analitical root %.6f\n', root_exact1);

fprintf('Calculated root%.6f\n', root1);

x = linspace(-2, 6, 1000);

y = f1(x);

y_root = f1(root1);

figure();

plot(x, y, 'b', root1, y_root, 'o', 'MarkerSize',12.5, 'LineWidth',2);

xlabel('x'); ylabel('f(x)');

title('$Newton Raphson$ --- $f(x) = e^x - (1 + x + \frac{1}{2}x^2)$ --- Root = 0.00',
'Interpreter', 'latex','FontSize',25);

xlabel('$x$', 'Interpreter', 'latex', 'FontSize', 20);

ylabel('$f(x)$', 'Interpreter', 'latex', 'FontSize', 20);

legend('function', 'Root', 'Location', 'northwest','fontsize', 15);

grid on;
```

```
% % ---- Newton Raphson convergence Table for f1 ----
```

```
[root1, xits, x_hist1, fx_hist1, err_hist1] = newton(x0, f1, df1, imax, tol, root_exact1);
```

```
fprintf('\nNewton Iteration Table for f1(x):\n');
```

```
fprintf('%8s %12s %12s %12s\n', 'Iter', 'Exact', 'Calc', 'Error');
```

```
fprintf('%8s %12s %12s %12s\n', '-----', '-----', '-----', '-----');
```

```
N1 = length(x_hist1);
```

```
for i = 1:N1
```

```
    calc = x_hist1(i);
```

```
    exact = root_exact1;
```

```
    err = abs(calc - exact);
```

```
    if i == 1 % initial guess (no Newton update yet)
```

```
        fprintf('%8d %12.6f %12.6f %12s\n', ...
```

```
            i-1, exact, calc, ' - ');
```

```
    else
```

```
        fprintf('%8d %12.6f %12.6f %12.6f\n', ...
```

```
            i-1, exact, calc, err);
```

```
    end
```

```
end
```

```

%% Newton for  $f(x) = 10 \cdot \log(x) - x$ 

fprintf('\n----- Newton for  $f(x) = 10 \cdot \log(x) - x$  -----\n');

f2 = @(x) 10*log(x) - x; % f(x)
df2 = @(x) (10./x) - 1; % f'(x)
x0 = 1; % initial guess

root_exact2 = fzero(f2, x0);

[root2, ~, ~] = newton(x0, f2, df2, imax, tol, root_exact2);

fprintf('Analytical root %.6f\n', root_exact2);
fprintf('Calculated root %.6f\n', root2);

x = linspace(0.1, 15, 1000); % avoid log(0)
y = f2(x);
y_root = f2(root2);

figure();

plot(x, y, 'b', root2, y_root, 'ro', 'MarkerSize',12.5, 'LineWidth',2);
yline(y_root, 'k--', 'LineWidth', 1);
xlabel('x'); ylabel('f(x)');

title('$Newton Raphson$ ---  $f(x) = 10 \cdot \log(x) - x$  --- Root = 1.12', 'Interpreter',
'latex', 'FontSize', 25);

xlabel('$x$', 'Interpreter', 'latex', 'FontSize', 20);
ylabel('$f(x)$', 'Interpreter', 'latex', 'FontSize', 20);

legend('function', 'Root', 'x-axis', 'Location', 'southeast', 'fontsize', 15);

```

grid on;

% ---- Newton Table for f2 ----

```
[root2, xits2, x_hist2, fx_hist2, err_hist2] = newton(x0, f2, df2, imax, tol, root_exact2);
```

```
fprintf('\nNewton Iteration Table for f2(x):\n');
```

```
fprintf('%8s %12s %12s %12s\n', 'Iter', 'Exact', 'Calc', 'Error');
```

```
fprintf('%8s %12s %12s %12s\n', '-----', '-----', '-----', '-----');
```

```
N2 = length(x_hist2);
```

```
for i = 1:N2
```

```
    calc = x_hist2(i);
```

```
    exact = root_exact2;
```

```
    err = abs(calc - exact);
```

```
    if i == 1 % initial guess
```

```
        fprintf('%8d %12.6f %12.6f %12s\n', ...
```

```
            i-1, exact, calc, ' - ');
```

```
    else
```

```
        fprintf('%8d %12.6f %12.6f %12.6f\n', ...
```

```
            i-1, exact, calc, err);
```

```
    end
```

```
end
```

%% Finding the point where two functions intercept

```
fprintf('\n----- Newton for two functions ----- \n');
```

```
f1 = @(x) 100*exp(-x); % f(x)
```

```
df1 = @(x) -100*exp(-x); % f'(x)
```

```
f2 = @(x) 5 * sin(0.5*(pi*x)); % f(x)
```

```
df2 = @(x) 5 * 0.5 * pi * cos(0.5*(pi*x)); % f'(x)
```

% Define the difference function and its derivative

```
f = @(x) f1(x) - f2(x);
```

```
df = @(x) df1(x) - df2(x);
```

```
a = 3;
```

```
b = 10;
```

```
x0 = 4;
```

```
imax = 20;
```

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

```
root_exact = fzero(f, x0);
```

```
[root, k, ~, ~] = newton(x0, f, df, imax, tol, root_exact);
```

```
x = linspace(a, b, 1000);
```

```

figure ();

plot(x, f(x), 'b', 'DisplayName', 'f1 = 100*exp(x)', 'LineWidth', 2);

hold on;

grid on;

title('$Newton$ $Raphson$ $Method$ $for$ $combine$ $f1$ $and$ $f2$', ...
      'Interpreter', 'latex', 'FontSize', 35);

xlabel('$x$', 'Interpreter', 'latex', 'FontSize', 20);

ylabel('$f(x)$ = $f_1(x)$ - $f_2(x)$', 'Interpreter', 'latex', 'FontSize', 20);

plot(root, f(root), 'ro', 'MarkerSize', 12.5, 'linewidth', 2, 'DisplayName', 'Intersection');

yline(0, 'k--', 'LineWidth', 1);

legend('$f(x)$', '$y = 0$', 'Starting Points', 'Tangent Lines', ...
      'Iteration Steps', 'Roots', 'Location', 'best', 'Interpreter', 'latex', 'FontSize', 15);

hold off;

% Plot the functions over the interval

figure();

plot(x, f1(x), 'b', 'DisplayName', 'f1 = 100*exp(x)', 'LineWidth', 3);

hold on;

plot(x, f2(x), 'Color', [0.9 0.4 0], 'DisplayName', 'f2 = 5*sin(0.5*pi*x)', 'LineWidth', 3);

yline(0, 'k--', 'LineWidth', 1);

title('$Intersection$ $of$ $f_1(x)$ $and$ $f_2(x)$', 'Interpreter', 'latex', 'FontSize', 35);

xlabel('$x$', 'Interpreter', 'latex', 'FontSize', 20);

ylabel('Function values', 'Interpreter', 'latex', 'FontSize', 20);

grid on;

% Plot the intersection point

```

```
plot(root, f1(root), 'ro', 'MarkerSize', 15, 'linewidth',2, 'DisplayName', 'Intersection');
```

```
legend show;
```

```
legend({'$f_1 = 100 e^{\text{x}}$', '$f_2 = 5\sin(0.5\pi x)$', '$y = 0$', '$\text{Intersection Point}$'}, ...
```

```
    'Location', 'southeast', 'Interpreter', 'latex', 'FontSize', 15);
```

```
hold off;
```

```
fprintf('Intersection point at x = %.6f\n', root);
```

```
%% higher degree of accuracy
```

```
fprintf('\n----- Higher degree of accuracy -----\n');
```

```
tol = 1e-10;
```

```
[root, ~, ~,~,~] = newton(x0, f, df, imax, tol, root_exact);
```

```
fprintf('Intersection point at x = %.10f\n', root);
```

```
%% Bracketing multiple roots
```

```
fprintf('\n----- Bracketing multiple roots -----\n');
```

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

```
x = linspace(4, 10, 2000);
```

```
y = f(x);
```

```
% Bracketing routine

brackets = [];

for i = 1:length(x)-1

    if y(i)*y(i+1) < 0 % opposite signs means there is a root in the interval

        brackets = [brackets; x(i), x(i+1)];

    end

end

% bisection method - finding roots

fprintf('\nFinding roots using bisection method:\n');

roots_bisect = zeros(size(brackets,1),1);

all_midpoints = cell(size(brackets,1),1);

all_left_bounds = cell(size(brackets,1),1);

all_right_bounds = cell(size(brackets,1),1);

for i = 1:size(brackets,1)

    a = brackets(i,1);

    b = brackets(i,2);

    % Analytical root

    root_exact = fzero(f, (a+b)/2);
```

```
[root_bisect, iter_bisect, mid_memory, left_memory, right_memory] = bisect(f, a, b, tol,
root_exact,imax);
```

```
roots_bisect(i) = root_bisect;
```

```
all_midpoints{i} = mid_memory;
```

```
all_left_bounds{i} = left_memory;
```

```
all_right_bounds{i} = right_memory;
```

```
end
```

```
%% ---- Bisection Convergence Table ----
```

```
fprintf('\n----- Bisection Convergence Table -----\n');
```

```
roots_exact_values = roots_bisect;
```

```
for i = 1:size(brackets,1)
```

```
    mids = all_midpoints{i};
```

```
    exact = roots_exact_values(i); % bisection root = exact
```

```
    fprintf('\nRoot %d (Bracket [%.6f , %.6f])\n', i, brackets(i,1), brackets(i,2));
```

```
    fprintf('%6s %14s %14s %14s\n', 'Iter', 'Exact', 'calc', 'error');
```

```
    fprintf('%6s %14s %14s %14s\n', '----', '-----', '-----', '-----');
```

```
% Iteration 0: Initial guess (average of bracket endpoints)

initial_calc = (brackets(i,1) + brackets(i,2)) / 2;

initial_err = abs(initial_calc - exact);

fprintf('%6d %14.6f %14.6f %14.6f\n', 0, exact, initial_calc, initial_err);


% Iterations 1 through N: Show each midpoint calculation

for j = 1:length(mids)

    calc = mids(j);

    err = abs(calc - exact);

    fprintf('%6d %14.6f %14.6f %14.6f\n', ...

        j, exact, calc, err);

end


fprintf('\nFinal converged root: %.10f\n', exact);

fprintf('Total iterations: %d\n', length(mids));

end


%% PLOT Bisection Method with Convergence Visualization

figure();

y_min = min(y);

y_max = max(y);

y_range = y_max - y_min;


plot(x, y, 'b', 'LineWidth', 2.5); % plot function
```

```
hold on;

yline(0,'k--', 'LineWidth', 1);    % Plot y=0 line

% Flags for legend
midpoint_plotted = false;
vertical_plotted = false;

% Plot bisection convergence for each root
for i = 1:size(brackets,1)

    midpoints = all_midpoints{i};

    left_bounds = all_left_bounds{i};

    right_bounds = all_right_bounds{i};

    % Plot initial bracket as black squares

    plot(left_bounds(1), f(left_bounds(1)), 'ks', 'MarkerSize', 10, 'MarkerFaceColor', 'k',
'LineWidth', 1.5, 'HandleVisibility', 'off');

    plot(right_bounds(1), f(right_bounds(1)), 'ks', 'MarkerSize', 10, 'MarkerFaceColor', 'k',
'LineWidth', 1.5, 'HandleVisibility', 'off');

    for j = 1:length(midpoints)

        mid = midpoints(j); % current midpoint

        % Plot midpoint as a marker on the curve

        if ~midpoint_plotted

            plot(mid, f(mid), 'mo', 'MarkerSize', 8, 'MarkerFaceColor', 'm', 'LineWidth', 1.5);

            midpoint_plotted = true;

        end
    end
end
```

```

else

    plot(mid, f(mid), 'mo', 'MarkerSize', 8, 'MarkerFaceColor', 'm', 'LineWidth', 1.5,
'HandleVisibility', 'off');

end

% Plot vertical line from midpoint on curve to x-axis
if ~vertical_plotted

    plot([mid, mid], [0, f(mid)], 'k:', 'LineWidth', 1.5);

    vertical_plotted = true;

else

    plot([mid, mid], [0, f(mid)], 'k:', 'LineWidth', 1.5, 'HandleVisibility', 'off');

end

end

end

% Plot roots

plot(roots_bisect, f(roots_bisect), 'ro', 'MarkerSize', 15, 'LineWidth', 2);

title('$Bisection$ $Method$ $with$ $all$ $Roots$ $marked$', 'Interpreter', 'latex',
'FontSize', 35);

xlabel('$x$', 'Interpreter', 'latex', 'FontSize', 20);

ylabel('$f(x) = f_1(x) - f_2(x)$', 'Interpreter', 'latex', 'FontSize', 20);

grid on;

legend('$f(x)$', '$y = 0$', 'Midpoints', 'Vertical Lines', 'Roots', ...

'Location', 'best', 'Interpreter', 'latex', 'FontSize', 15);

xlim([min(x) max(x)]);

ylim([y_min - 1, y_max + 1]);

```

hold off;

%% Newton-Raphson method

fprintf('\nFinding roots using Newton-Raphson method:\n');

roots_newton = zeros(size(brackets,1),1);

all_newton_iterations = cell(size(brackets,1),1); % store all iterations

for i = 1:size(brackets,1)

 a = brackets(i,1);

 b = brackets(i,2);

 x0 = a;

 root_exact_i = roots_bisect(i);

 [root_newton, iter_newton, x_history] = newton(x0, f, df, imax, tol, root_exact_i);

 roots_newton(i) = root_newton;

 all_newton_iterations{i} = x_history; % save iteration history

 fprintf(' Root %d: x = %.6f found in %d iterations (Newton-Raphson), starting from x0
= %.6f\n', ...

 i, root_newton, iter_newton, x0);

end

% Plot Newton-Raphson results with visulisation

```
figure();

y_min = min(y);

y_max = max(y);

y_range = y_max - y_min;

plot(x, y, 'b', 'LineWidth', 2.5);

hold on;

yline(0,'k--', 'LineWidth', 1);


% Plot tangent lines for all roots

tangent_plotted = false; % flag for legend

starting_plotted = false;

vertical_plotted = false; % flag for vertical lines legend


for i = 1:size(brackets,1)

    x_history = all_newton_iterations{i};


    % Plot starting guess

    x0 = x_history(1);

    if ~starting_plotted

        plot(x0, f(x0), 'mo', 'MarkerSize', 10, 'MarkerFaceColor', 'm', 'LineWidth', 1.5);

        starting_plotted = true;

    else

        plot(x0, f(x0), 'mo', 'MarkerSize', 10, 'MarkerFaceColor', 'm', 'LineWidth', 1.5,
'HandleVisibility', 'off');

    end

end
```

```
% Plot tangent lines at each iteration

for j = 1:length(x_history)-1

    x_now = x_history(j);

    y_now = f(x_now);

    slope = df(x_now);

    if abs(slope) < 1e-10

        continue; % skip if derivative is too small

    end

    % Calculate where tangent line crosses x-axis

    x_next = x_now - y_now/slope;

    y_next = f(x_next);

    % Create extended tangent line

    tangent_width = 2.0;

    x_tangent = linspace(min(x_now,x_next)-tangent_width,
max(x_now,x_next)+tangent_width, 300);

    y_tangent = y_now + slope*(x_tangent - x_now);

    % Plot tangent line

    if ~tangent_plotted

        plot(x_tangent, y_tangent, 'r:', 'LineWidth', 2);

        tangent_plotted = true;

    end

end
```

```

else

    plot(x_tangent, y_tangent, 'r:', 'LineWidth', 2, 'HandleVisibility', 'off');

end

% Plot BLACK DOTTED vertical line from x-axis to function curve at x_next

if j < length(x_history) % Don't plot for the final iteration

    y_next = f(x_next);

    if ~vertical_plotted

        plot([x_next, x_next], [0, y_next], 'k:', 'LineWidth', 1.5);

        vertical_plotted = true;

    else

        plot([x_next, x_next], [0, y_next], 'k:', 'LineWidth', 1.5, 'HandleVisibility', 'off');

    end

end

end

end

% Plot all roots at the end (so they appear on top)

plot(roots_newton, f(roots_newton), 'ro', 'MarkerSize', 15, 'LineWidth', 2);

title('$Newton$ $Raphson$ $Method$ $with$ $all$ $Roots$ $marks$', 'Interpreter',
'latex', 'FontSize', 35);

xlabel('$x$', 'Interpreter', 'latex', 'FontSize', 20);

ylabel('$f(x) = f_1(x) - f_2(x)$', 'Interpreter', 'latex', 'FontSize', 20);

```

```

grid on;

legend('$f(x)$', '$y = 0$', 'Starting Points', 'Roots', 'Location', 'best', 'Interpreter', 'latex',
'FontSize', 15);

xlim([min(x) max(x)]);

ylim([y_min - 1, y_max + 1]);

hold off;

%% ---- Newton Raphson Convergence Table ----

fprintf('\n----- Newtons Convergence Table ----- \n');

for i = 1:size(brackets,1)

    exact = roots_newton(i);

    x_history = all_newton_iterations{i};

    fprintf('\nRoot %d (Bracket [%.6f , %.6f])\n', i, brackets(i,1), brackets(i,2));

    fprintf('%6s %14s %14s %14s\n', 'Iter', 'Exact', 'calc', 'error');

    fprintf('%6s %14s %14s %14s\n', '----', '-----', '-----', '-----');

    % All Newton iterations

    for k = 1:length(x_history)

        calc = x_history(k);

        err = abs(calc - exact);

```

```
    if k == 1

        iter_num = 0;

    else

        iter_num = k-1;

    end

    fprintf('%6d %14.6f %14.6f %14.6f\n', iter_num, exact, calc, err);

end

fprintf('\nFinal converged root: %.10f\n', exact);

fprintf('Total iterations: %d\n', length(x_history)-1);

end

function [root, iterations, mid_memory, left_memory, right_memory] = bisect(f, x1, x3,
tol, root_exact,imax)

mid_memory = [];

left_memory = [];

right_memory = [];

f1 = f(x1);

f3 = f(x3);
```

```
x2 = (x3 + x1)/2;

difference = abs(x2-root_exact);

iterations = 0;

while difference > tol && iterations < imax

    if f1 * f3 > 0

        error('No root found - need new guesses');

    end

    % midpoint calculation

    f2 = f(x2);

    if f1 * f2 < 0

        x3 = x2;

        f3 = f2;

    else

        x1 = x2;

        f1 = f2;

    end

    x2 = (x1 + x3) / 2;

    mid_memory = [mid_memory; x2];

    left_memory = [left_memory; x1];
```

```
right_memory= [right_memory; x3];

difference = abs(x2-root_exact);

iterations = iterations + 1;

end

% Final root estimate

root = x2;

end

function [root, k, x_history, fx_history, err_history] = newton(x0,f,df,imax,tol,root_exact)

x = x0;

k = 0;

x_history = x0; % Initialize history with starting guess

fx_history = f(x0);

err_history = [];

difference = abs(root_exact - x);

while difference > tol && k < imax
```

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

dfdx = df(x); % evaluate derivative


% Check derivative

if dfdx == 0 % cannot divide by zero

    error('Derivative is zero. Newton method fails.');
```



```
end


% Newton-Raphson update

dx = -fx / dfdx;

xnew = x + dx;


k = k + 1;

difference = abs(xnew-root_exact);


x_history = [x_history; xnew];

fx_history = [fx_history; f(xnew)];

err_history = [err_history; difference];


% Check convergence

if difference < tol

    root = xnew;

    return

end
```

```
    x = xnew;  
end
```

```
% If loop ends without hitting tolerance:
```

```
root = xnew;  
end
```