



University of Wales Trinity St David

Task - 2

Student name: Aangan Nitinkumar Patel

Student ID number: 2511974

Program: MSc – Automotive Engineering

Engineering Module: Alternative fuels and Combustion Modelling

Module code: ACHA7002

Date: 07/12/2025

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 $fx = 100e - x - 5 \sin(\pi x/2)$	5
4	Bracketing	6
4.1	Bisect $fx = 100e - x - 5 \sin(\pi x/2)$	6
4.2	Newton $fx = 100e - x - 5 \sin(\pi x/2)$	7
5	Appendices	7
5.1	Convergence table Bisection $fx = \sin x \sinh(x)$	7
5.2	Convergence table Bisection $fx = R^2 - x^{21/2} - x \tan(x)$	8
5.3	Convergence table Newton b $fx = ex - (1 + x + (x^2/2))$	9
5.4	Convergence table Newton c $fx = 10 \log x - x$	10
5.5	Convergence table Bisect $fx = 100e - x - 5 \sin(\pi x/2)$	10
5.6	Convergence table Newton $fx = 100e - x - 5 \sin(\pi x/2)$	11
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 $f(x) = \sin(x) \sinh(x)$

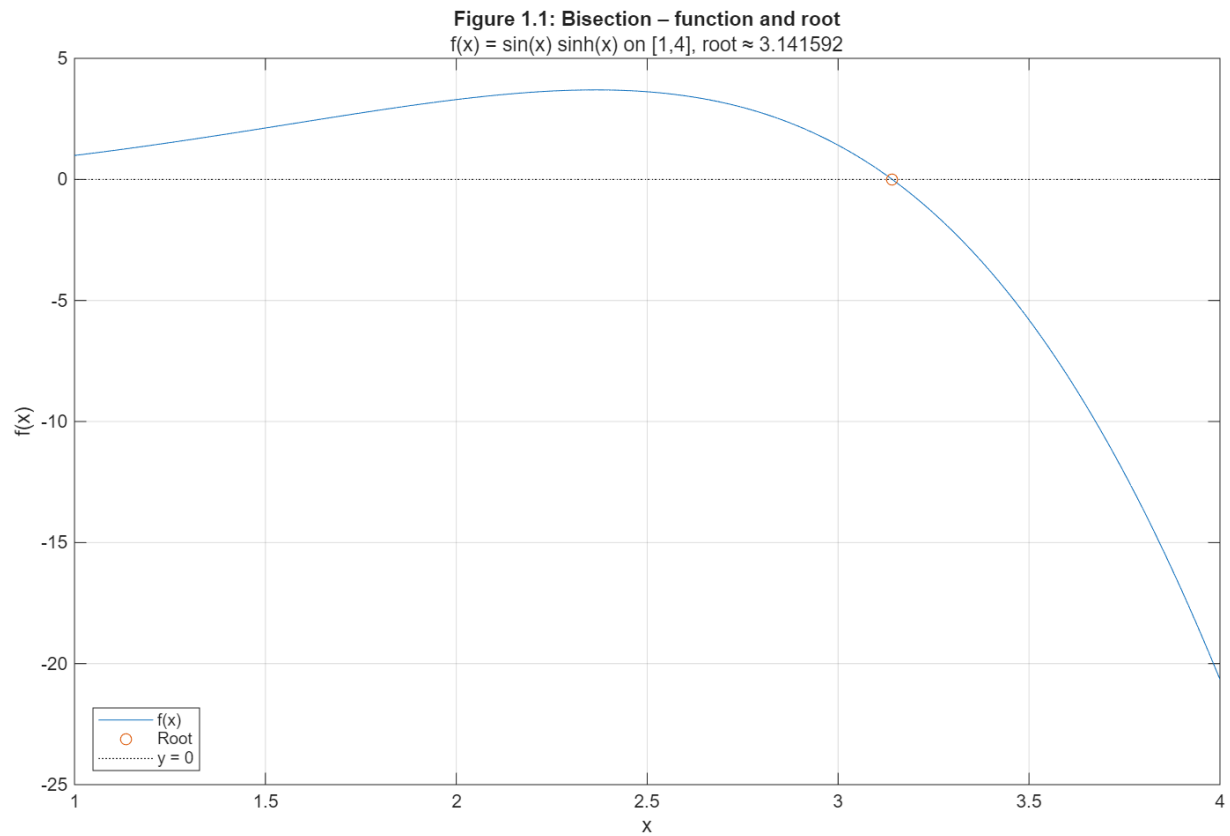


Figure 1-1 this shows the function labelled in the legend and the root

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

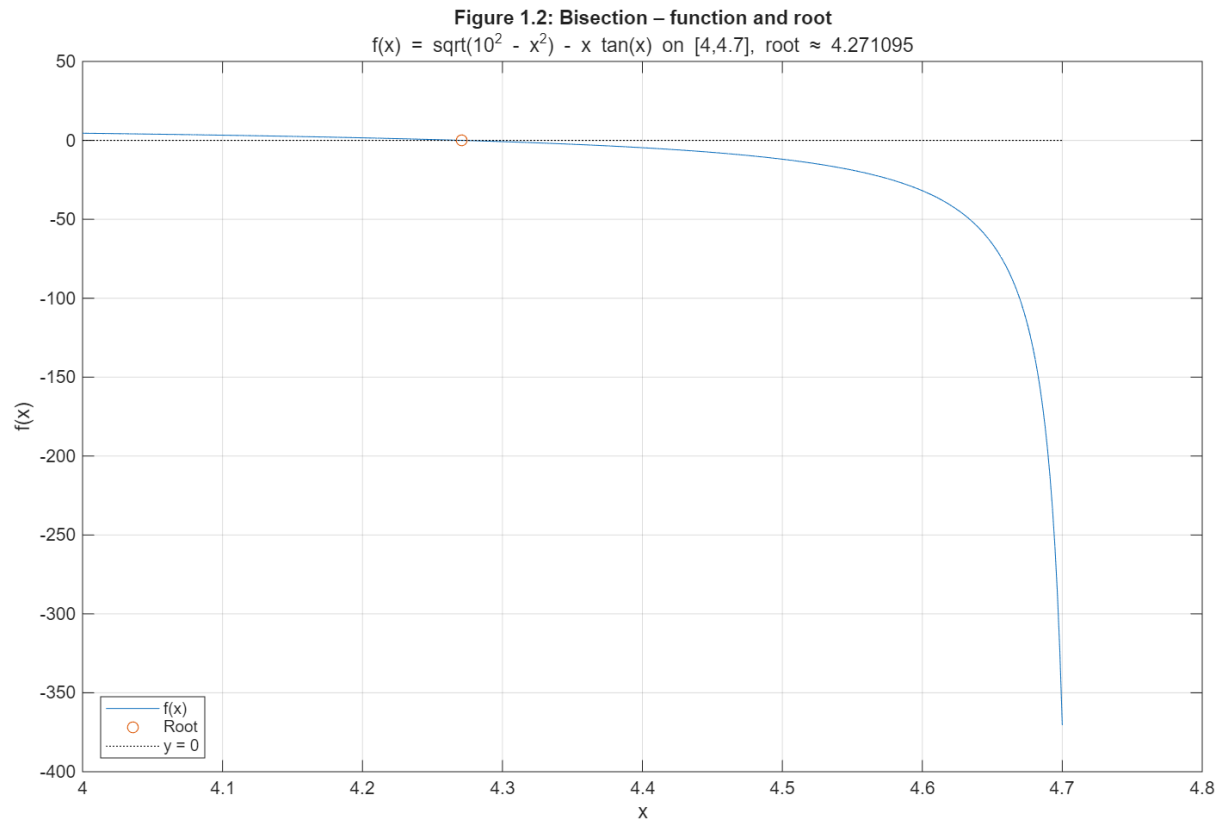


Figure 1-2 this shows the function labelled in the legend and the root

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

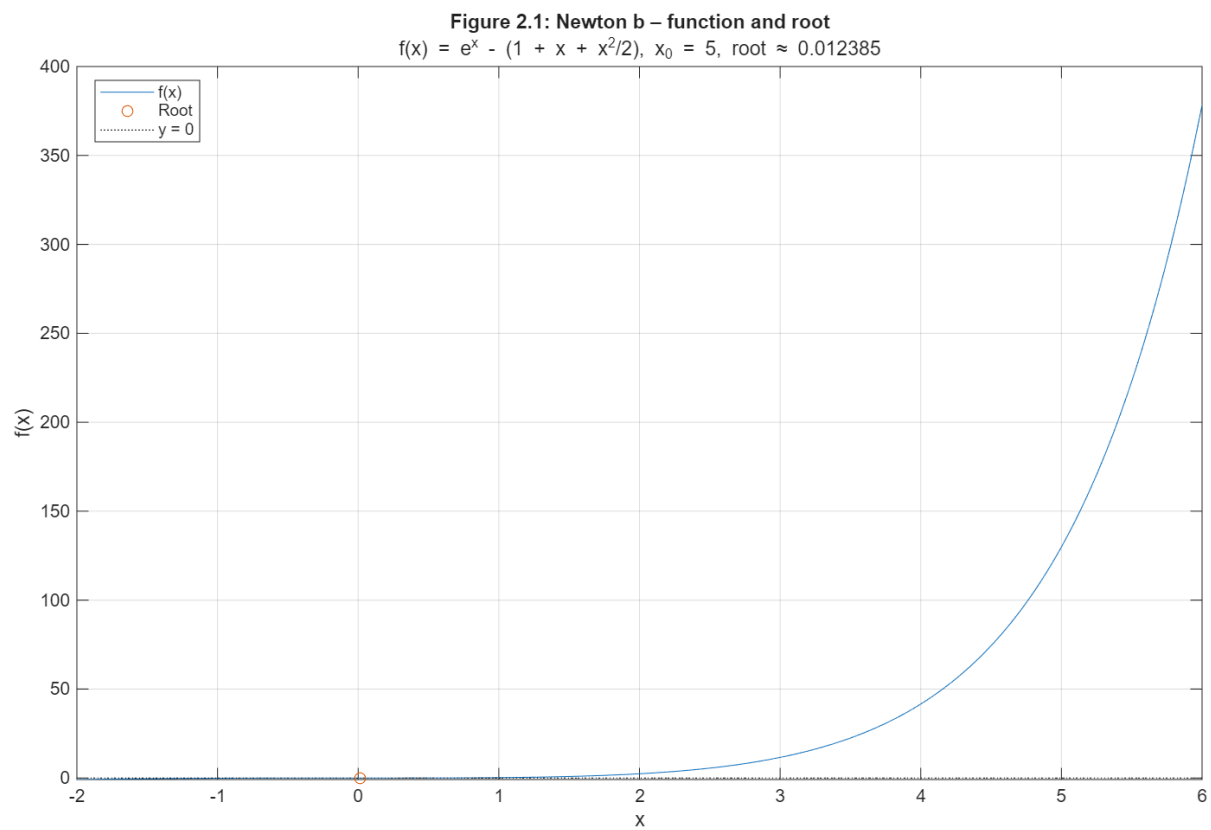


Figure 2-1 this shows the function labelled in the legend and the root

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

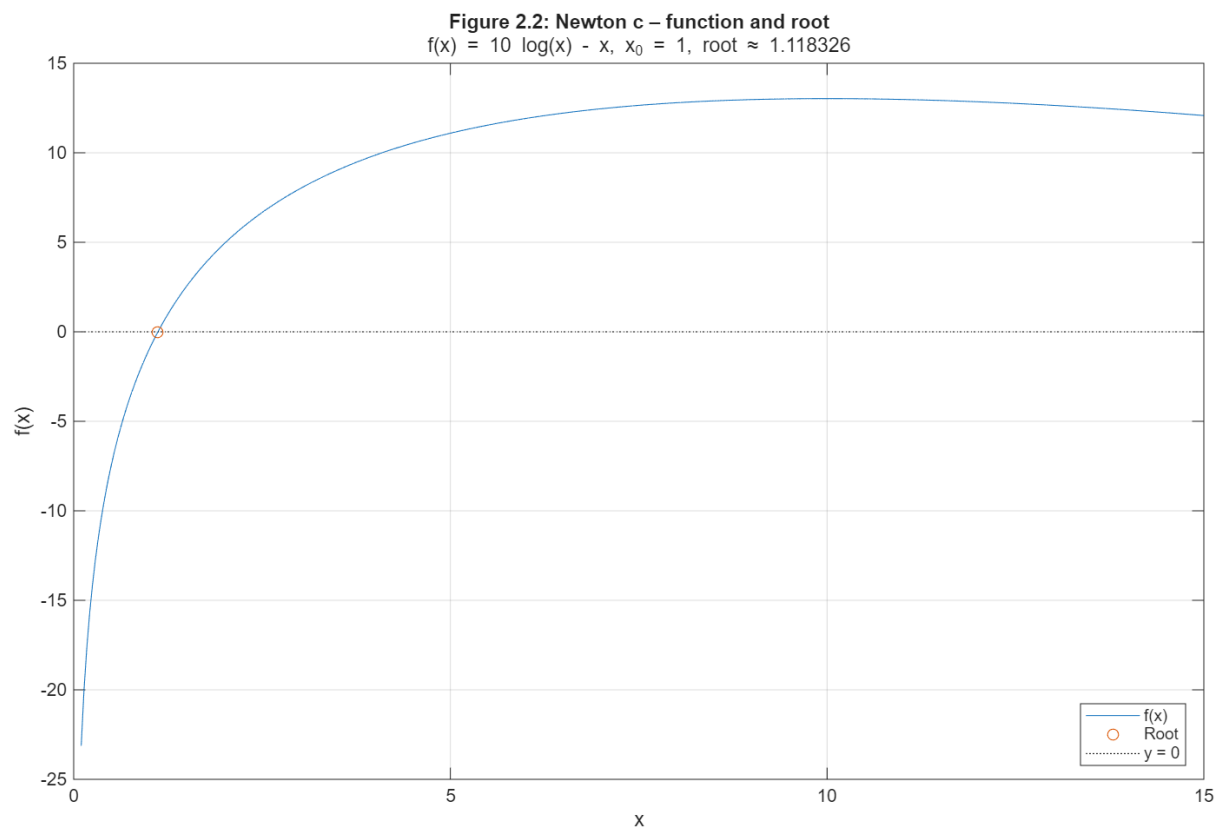


Figure 2-2 this shows the function labelled in the legend and the root

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

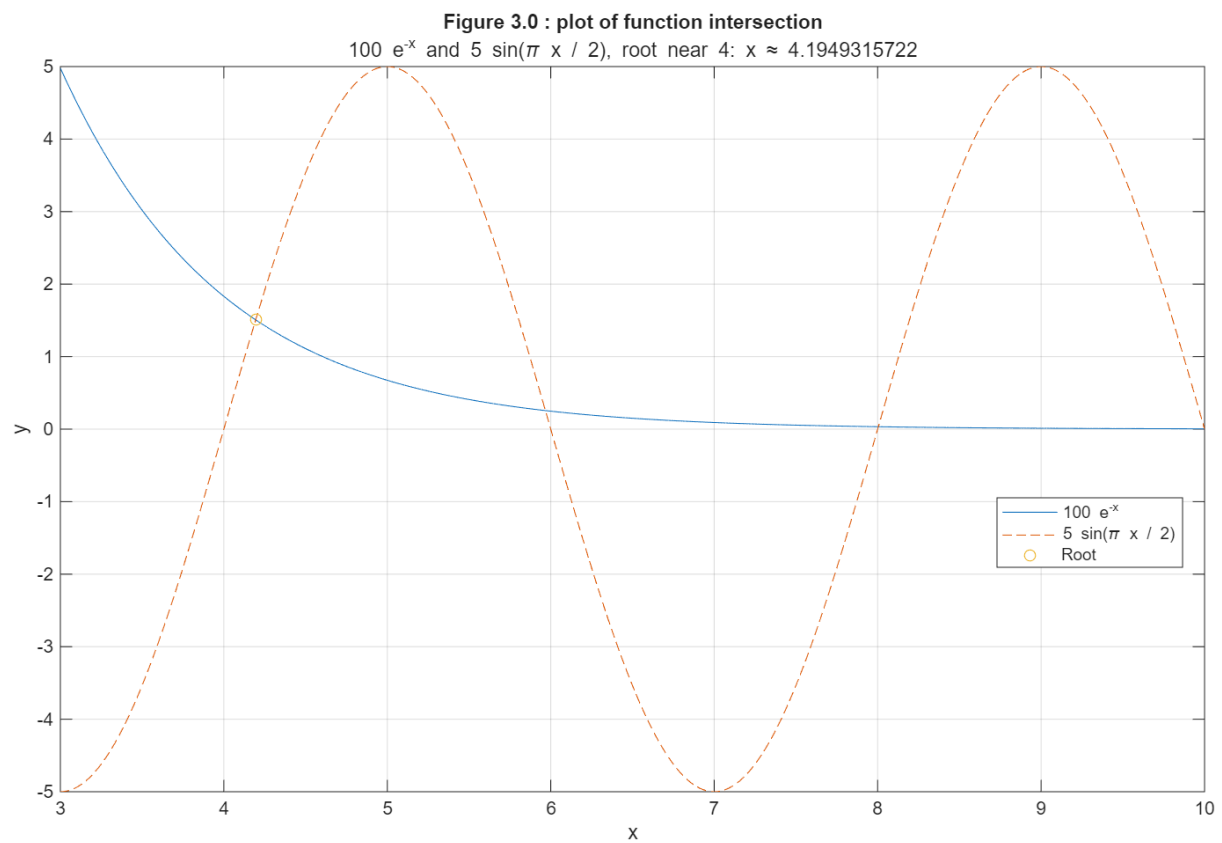


Figure 3-1 plot of function intersection

4 Bracketing

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

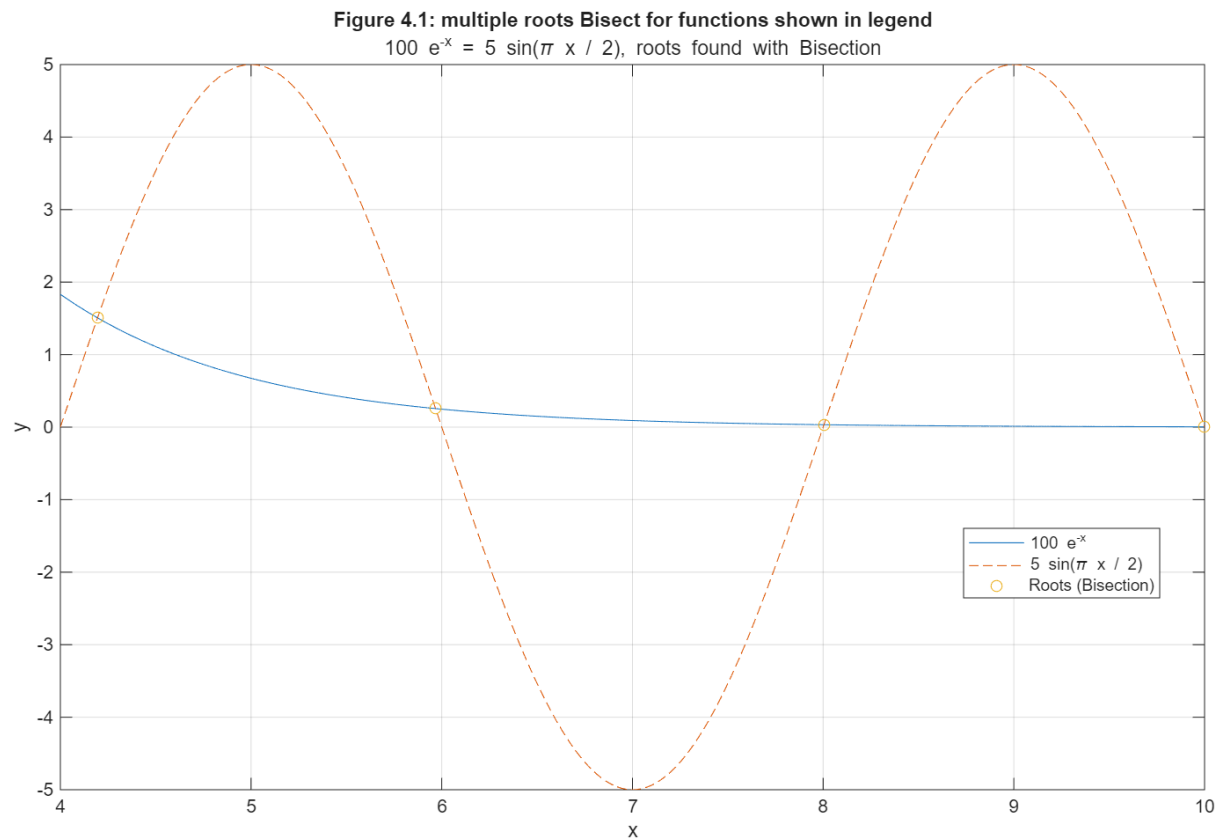


Figure 4-1 multiple roots Bisect for functions shown in legend

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

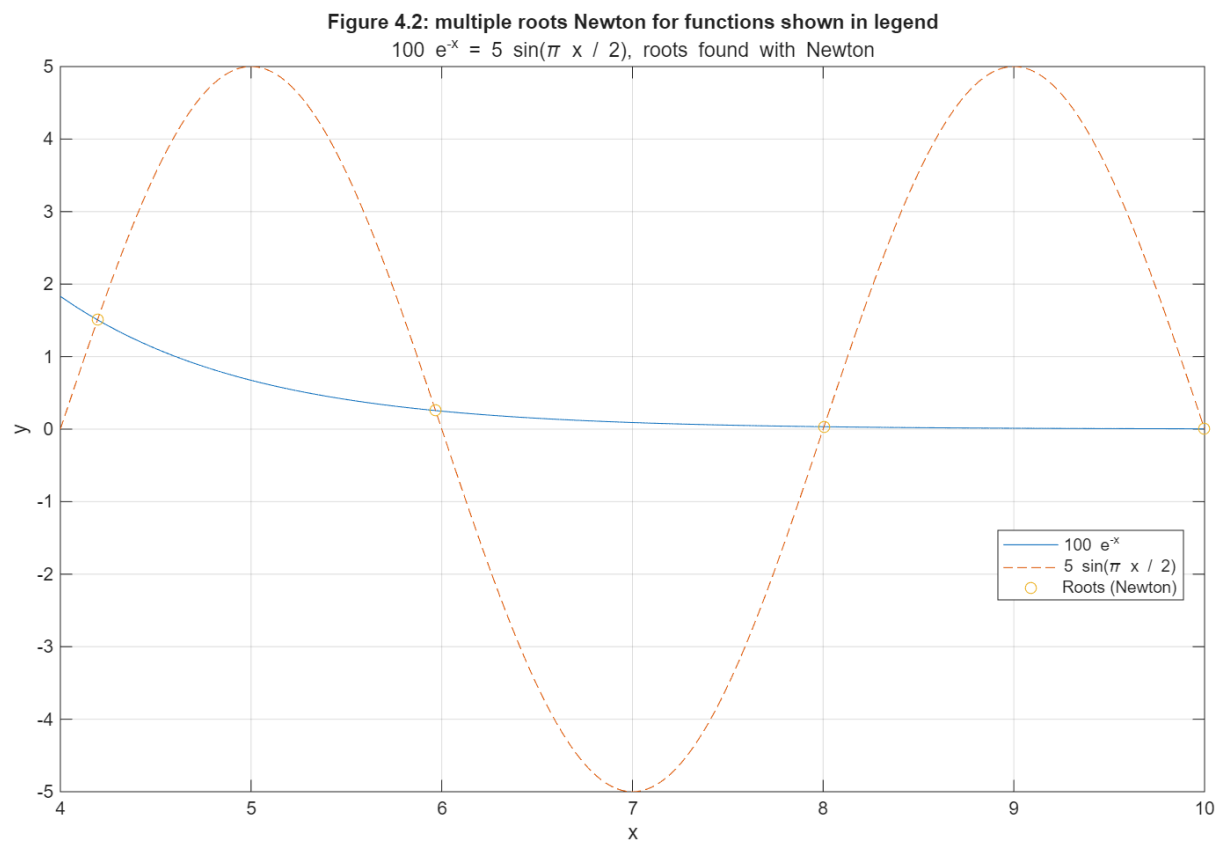


Figure 4-2 multiple roots Newton for functions shown in legend

5 Appendices

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

iter	X mid	F (x mid)	half interval
1	2.5	3.6209	1.5
2	3.25	-1.3931	0.75
3	2.875	2.3274	0.375
4	3.0625	0.8428	0.1875
5	3.1562	-0.1718	0.0938
6	3.1094	0.3602	0.0469
7	3.1328	0.1005	0.0234
8	3.1445	-0.034	0.0117
9	3.1387	0.0336	0.0059
10	3.1416	-0.0001	0.0029
11	3.1401	0.0168	0.0015

12	3.1409	0.0083	0.0007
13	3.1412	0.0041	0.0004
14	3.1414	0.002	0.0002
15	3.1415	0.001	0.0001
16	3.1416	0.0004	0
17	3.1416	0.0002	0
18	3.1416	0	0
19	3.1416	0	0
20	3.1416	0	0
21	3.1416	0	0
22	3.1416	0	0

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

iter	x_mid	f(x_mid)	half_interval
1	4.35	-2.4692	0.35
2	4.175	2.0804	0.175
3	4.2625	0.2195	0.0875
4	4.3062	-0.9881	0.0438
5	4.2844	-0.3553	0.0219
6	4.2734	-0.0612	0.0109
7	4.268	0.0808	0.0055
8	4.2707	0.0102	0.0027
9	4.2721	-0.0254	0.0014
10	4.2714	-0.0076	0.0007
11	4.271	0.0013	0.0003
12	4.2712	-0.0031	0.0002
13	4.2711	-0.0009	0.0001
14	4.2711	0.0002	0
15	4.2711	-0.0004	0
16	4.2711	-0.0001	0
17	4.2711	0.0001	0
18	4.2711	0	0

19	4.2711	0	0
20	4.2711	0	0

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

iter	x	f(x)
1	5	129.9132
2	4.0878	46.1643
3	3.241	16.0667
4	2.4874	5.4488
5	1.8495	1.7969
6	1.3372	0.5771
7	0.9449	0.1812
8	0.6562	0.0559
9	0.4499	0.017
10	0.3057	0.0051
11	0.2065	0.0015
12	0.1388	0.0005
13	0.0931	0.0001
14	0.0623	0
15	0.0416	0
16	0.0278	0
17	0.0186	0

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

iter	x	f(x)
1	1	-1
2	1.1111	-0.0575
3	1.1183	-0.0002

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

iter	x_mid	f(x_mid)	half_interval
1	4.2121	-0.1537	0.0303
2	4.1969	-0.0181	0.0151
3	4.1894	0.0501	0.0076
4	4.1932	0.0159	0.0038
5	4.1951	-0.0011	0.0019
6	4.1941	0.0074	0.0009
7	4.1946	0.0031	0.0005
8	4.1948	0.001	0.0002
9	4.1949	-0.0001	0.0001
10	4.1949	0.0005	0.0001
11	4.1949	0.0002	0
12	4.1949	0.0001	0
13	4.1949	0	0
14	4.1949	0	0
15	4.1949	0	0
16	4.1949	0	0

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

iter	x	f(x)
1	4	1.8316
2	4.1891	0.0525
3	4.1949	0.0001
4	4.1949	0

5.7 Your code and functions

5.7.1 Function

5.7.1.1 *bisect*

```
function [root, conv] = bisect(f, a, b, imax, tol)
%BISECT Find a root of f(x)=0 using the bisection method.
% [root, conv] = bisect(f, a, b, imax, tol)
% Input:
% f - function to be evaluated (string with file name)
% a,b - interval limits
% imax - iteration maximum
% tol - convergence criterion
% Output:
% root - position of the root
% conv - convergence table: [iter, x_mid, f(x_mid), half_interval]

% initialise parameters, i.e.
x1 = a;
f1 = feval(f, x1);
x3 = b;
f3 = feval(f, x3);

% CHECK: verify that there is indeed a root in the interval [a b].
if f1 * f3 > 0
    error('No root in the specified interval')
end

% preallocate convergence table
conv = zeros(imax, 4);

% Begin iteration loop i.e. for ndx = 1:imax
for ndx = 1:imax
    % mid-point
    x2 = (x3 + x1) / 2;
    f2 = feval(f, x2);

    % half interval length d
    d = (x3 - x1) / 2;

    % store convergence info: [iteration, x_mid, f(x_mid), half_interval]
    conv(ndx, :) = [ndx, x2, f2, d];

    % test for convergence i.e. |d| < tol (or |f2| < tol)
```

```

if abs(d) < tol || abs(f2) < tol
    root = x2;
    conv = conv(1:ndx, :); % trim unused rows
    return
end

% Find which half interval contains root – i.e. if-else construction
if f1 * f2 < 0
    % Root is in left half, so
    x3 = x2;
    f3 = f2;
else
    % Root is in right half, so
    x1 = x2;
    f1 = f2;
end
end

% CHECK: Excessive iterations
warning('Excessive iterations in bisect: imax reached without convergence.')
root = x2; % last midpoint
conv = conv(1:imax, :); % return full table up to imax

end

```

5.7.1.2 *f_bisect1*

```

function y = f_bisect1(x)
% f_bisect1: calculates f(x) = sin(x) * sinh(x)

y = sin(x) .* sinh(x);

end

```

5.7.1.3 *f_bisect2*

```

function y = f_bisect2(x)
% f(x) = sqrt(R^2 - x^2) - x * tan(x), with R = 10

R = 10;
y = sqrt(R^2 - x.^2) - x .* tan(x);

end

```

5.7.1.4 *newton*

```

function [root, conv] = newton(x, f, df, imax, tol)
% function root = newton(x,f,df,imax,tol)
% Input: x initial guess at the root
%       f function to be evaluated
%       df derivative of f with respect to x
%       imax iteration maximum
%       tol convergence criterion
% Output: root position of the root
%
% Newton's algorithm
% xnew = xold + dx
% dx = -fold(xold)/dfold(xold)
% NB: both f and df are user coded functions – you will need to write functions for both!!!

```

```

% initialise parameters, i.e.
xold = x;
fold = feval(f, xold); % f(xold)
count = 0;

% preallocate convergence table: [iteration, xold, f(xold)]
conv = zeros(imax, 3);

% CHECK: verify that the derivative is not zero.
while abs(fold) > tol
    dfold = feval(df, xold); % df(xold)
    if dfold == 0
        error('Problem') % derivative is zero – cannot continue
    end
    count = count + 1;

    % calculate equation 4 (Newton step)
    dx = -fold / dfold;
    xnew = xold + dx;

    % store convergence info
    conv(count, :) = [count, xold, fold];

    % Print iteration and current error (optional, matches pseudo-code style)
    fprintf('Iteration - %i,\t Error - %g\n', count, abs(fold));

    % update for next iteration
    xold = xnew;
    fold = feval(f, xold);

    % CHECK: Excessive iterations
    if count >= imax
        warning('Newton: maximum iterations reached without convergence. ');
        break;
    end
end

root = xold;
conv = conv(1:count, :); % trim unused rows

end

```

5.7.1.5 *f_newton_b*

```

function y = f_newton_b(x)
% f_newton_b:  $f(x) = e^x - (1 + x + x^2/2)$ 

y = exp(x) - (1 + x + 0.5 * x.^2);

end

```

5.7.1.6 *df_newton_b*

```

function dy = df_newton_b(x)
% df_newton_b: derivative of  $f(x) = e^x - (1 + x + x^2/2)$ 
% So  $f'(x) = e^x - (1 + x)$ 

dy = exp(x) - (1 + x);

```

end

5.7.1.7 *f_newton_c*

```
function y = f_newton_c(x)
% f_newton_c:  $f(x) = 10 \ln(x) - x$ 

y = 10 * log(x) - x; % log(x) is natural log in MATLAB

end
```

5.7.1.8 *df_newton_c*

```
function dy = df_newton_c(x)
% df_newton_c: derivative of  $f(x) = 10 \ln(x) - x$ 
% so  $f'(x) = 10/x - 1$ 

dy = 10 ./ x - 1;

end
```

5.7.1.9 *f_intersection*

```
function y = f_intersection(x)
% f_intersection:  $f(x) = 100 \exp(-x) - 5 \sin(\pi x / 2)$ 

y = 100 * exp(-x) - 5 * sin(pi * x / 2);

end
```

5.7.1.10 *df_intersection*

```
function dy = df_intersection(x)
% df_intersection: derivative of  $f(x) = 100 \exp(-x) - 5 \sin(\pi x / 2)$ 
% so  $f'(x) = -100 \exp(-x) - 5(\pi/2) \cos(\pi x / 2)$ 

dy = -100 * exp(-x) - 5 * (pi/2) * cos(pi * x / 2);

end
```

5.7.1.11 *bracket_roots*

```
function intervals = bracket_roots(f, xvec)
% bracket_roots: find intervals  $[x_i, x_{i+1}]$  where  $f(x)$  changes sign
%
% f - function name (string), e.g. 'f_intersection'
% xvec - vector of x values, e.g. linspace(4, 10, 100)
% intervals - Nx2 matrix of [a, b] pairs that bracket roots

y = feval(f, xvec); % evaluate function at all x values
n = length(xvec) - 1; % maximum possible number of intervals

% preallocate with zeros (fast)
intervals = zeros(n, 2);
count = 0; % number of intervals we actually find

for i = 1:n
    ya = y(i);
    yb = y(i + 1);
```

```

% if sign changes between x(i) and x(i+1), we have a root bracket
if ya == 0
    count = count + 1;
    intervals(count, :) = [xvec(i), xvec(i)];
elseif ya * yb < 0
    count = count + 1;
    intervals(count, :) = [xvec(i), xvec(i + 1)];
end
end

% keep only the rows we used
if count == 0
    intervals = []; % no roots found
else
    intervals = intervals(1:count, :);
end

end

```

5.7.2 script

5.7.2.1 task2_main

```

%% Task 2 – Root finding
clear; clc; close all;

%% 1. BISECTION METHOD

% For Bisection
tol_bis = 1e-6;
imax_bis = 50;

% 1(b)  $f(x) = \sin(x) \sinh(x)$ ,  $x$  in  $[1, 4]$ 
a1 = 1;
b1 = 4;

[root_b1, conv_b1] = bisect('f_bisect1', a1, b1, imax_bis, tol_bis);

x1 = linspace(a1, b1, 400);
y1 = f_bisect1(x1);
y1_root = f_bisect1(root_b1);

figure(1);
plot(x1, y1, '-'); hold on;
plot(root_b1, y1_root, 'o'); % root marker
plot([a1 b1], [0 0], 'k:'); % x-axis
grid on;
xlabel('x'); ylabel('f(x)');

% TEMPLATE: "Figure 1.1 this shows the function labelled in the legend and the root"
title('Figure 1.1: Bisection – function and root');
subtitle(sprintf('f(x) = sin(x) sinh(x) on [1,4], root ≈ %.6f', root_b1));
legend('f(x)', 'Root', 'y = 0', 'Location', 'best');
disp('1.Bisection (b) - Convergence table: [iter, x_mid, f(x_mid), half_interval]');
disp(conv_b1);

% 1(c)  $f(x) = \sqrt{R^2 - x^2} - x \tan(x)$ ,  $R = 10$ ,  $x$  in  $[4, 4.7]$ 
a2 = 4;

```

```

b2 = 4.7;

[root_b2, conv_b2] = bisection('f_bisect2', a2, b2, imax_bis, tol_bis);

x2 = linspace(a2, b2, 400);
y2 = f_bisect2(x2);
y2_root = f_bisect2(root_b2);

figure(2);
plot(x2, y2, '-'); hold on;
plot(root_b2, y2_root, 'o'); % root marker
plot([a2 b2], [0 0], 'k:');
grid on;
xlabel('x'); ylabel('f(x)');

% TEMPLATE: "Figure 1.2 this shows the function labelled in the legend and the root"
title('Figure 1.2: Bisection – function and root');
subtitle(sprintf('f(x) = sqrt(10^2 - x^2) - x tan(x) on [4,4.7], root ≈ %.6f', root_b2));
legend('f(x)', 'Root', 'y = 0', 'Location','best');
disp('1.Bisection (c) - Convergence table: [iter, x_mid, f(x_mid), half_interval]');
disp(conv_b2);

```

%% 2. NEWTON–RAPHSON METHOD

% For Newton

```

tol_new = 1e-6;
imax_new = 20;

```

```

% 2(b) f(x) = e^x - (1 + x + x^2/2), x0 = 5

```

```

x0_1 = 5;
[root_n1, conv_n1] = newton(x0_1, 'f_newton_b', 'df_newton_b', imax_new, tol_new);

```

```

xn1 = linspace(-2, 6, 400);
yn1 = f_newton_b(xn1);

```

```

figure(3);
plot(xn1, yn1, '-'); hold on;
plot(root_n1, f_newton_b(root_n1), 'o');
plot([min(xn1) max(xn1)], [0 0], 'k:');
grid on;
xlabel('x'); ylabel('f(x)');

```

```

% TEMPLATE: "Figure 2.1 this shows the function labelled in the legend and the root"
title('Figure 2.1: Newton b – function and root');
subtitle(sprintf('f(x) = e^x - (1 + x + x^2/2), x_0 = 5, root ≈ %.6f', root_n1));
legend('f(x)', 'Root', 'y = 0', 'Location','best');
disp('2.Newton (b) - Convergence table : [iter, x, f(x)]');
disp(conv_n1);

```

```

% 2(c) f(x) = 10 log(x) - x, x0 = 1

```

```

x0_2 = 1;
[root_n2, conv_n2] = newton(x0_2, 'f_newton_c', 'df_newton_c', imax_new, tol_new);

```

```

xn2 = linspace(0.1, 15, 400);
yn2 = f_newton_c(xn2);

```

```

figure(4);
plot(xn2, yn2, '-'); hold on;
plot(root_n2, f_newton_c(root_n2), 'o');

```

```

plot([min(xn2) max(xn2)], [0 0], 'k:');
grid on;
xlabel('x'); ylabel('f(x)');

% TEMPLATE: "Figure 2.2 this shows the function labelled in the legend and the root"
title('Figure 2.2: Newton c – function and root');
subtitle(sprintf('f(x) = 10 log(x) - x, x_0 = 1, root ≈ %.6f', root_n2));
legend('f(x)', 'Root', 'y = 0', 'Location','best');
disp('2.Newton (c) - Convergence table : [iter, x, f(x)]');
disp(conv_n2);

%% 3. INTERSECTION (single root near x ≈ 4)

% For intersection / multiple roots (Newton)
tol_int = 1e-10;
imax_int = 40;

[root_int, conv_int] = newton(4, 'f_intersection', 'df_intersection', imax_int, tol_int);

X = linspace(3, 10, 400);
yL = 100 * exp(-X);
yR = 5 * sin(pi * X / 2);

figure(5);
plot(X, yL, '-', X, yR, '--'); hold on;
plot(root_int, 100 * exp(-root_int), 'o');
grid on;
xlabel('x'); ylabel('y');

% TEMPLATE: "Figure 3.0 plot of function intersection"
title('Figure 3.0 : plot of function intersection');
subtitle(sprintf('100 e^{-x} and 5 sin(\pi x / 2), root near 4: x ≈ %.10f', root_int));
legend('100 e^{-x}', '5 sin(\pi x / 2)', 'Root', 'Location','best');
disp('3.Newton (intersection near 4): [iter, x, f(x)]');
disp(conv_int);

%% 4. MULTIPLE ROOTS (BRACKETING + BISECTION + NEWTON)
% 4. Bracketing (Multiple Roots)

% 4.1 Bracket roots in [4, 10]
xgrid = linspace(4, 10, 100);
intervals = bracket_roots('f_intersection', xgrid);

disp('4.Bracketing intervals [a, b] for f_intersection:');
disp(intervals);

n_roots = size(intervals, 1);
roots_bis = zeros(n_roots, 1);
roots_new = zeros(n_roots, 1);

% 4.2 Refine each bracketed root with Bisection and Newton
for k = 1:n_roots
    a = intervals(k, 1);
    b = intervals(k, 2);

    % Bisection
    [roots_bis(k), ~] = bisection('f_intersection', a, b, imax_bis, tol_bis);

```

```

    % Newton, starting from mid-point of the bracket
    x0k = (a + b) / 2;
    [roots_new(k), ~] = newton(x0k, 'f_intersection', 'df_intersection', imax_new, tol_new);
end

% 4.3 Plot equation (**) with all roots – Bisection
X = linspace(4, 10, 400);
yL = 100 * exp(-X);
yR = 5 * sin(pi * X / 2);

figure(6);
plot(X, yL, '-', X, yR, '--'); hold on;
plot(roots_bis, 100 * exp(-roots_bis), 'o');
grid on;
xlabel('x'); ylabel('y');

% TEMPLATE: "Figure 4.1 multiple roots Bisect for functions shown in legend"
title('Figure 4.1: multiple roots Bisect for functions shown in legend');
subtitle('100 e^{-x} = 5 sin(\pi x / 2), roots found with Bisection');
legend('100 e^{-x}', '5 sin(\pi x / 2)', 'Roots (Bisection)', 'Location','best');

% 4.4 Plot equation (**) with all roots – Newton
figure(7);
plot(X, yL, '-', X, yR, '--'); hold on;
plot(roots_new, 100 * exp(-roots_new), 'o');
grid on;
xlabel('x'); ylabel('y');

% TEMPLATE: "Figure 4.2 multiple roots Newton for functions shown in legend"
title('Figure 4.2: multiple roots Newton for functions shown in legend');
subtitle('100 e^{-x} = 5 sin(\pi x / 2), roots found with Newton');
legend('100 e^{-x}', '5 sin(\pi x / 2)', 'Roots (Newton)', 'Location','best');

```