

UWTSD

Module

Task 2

Module code: ACHA7002

Anthony Yin Wai Hau

Student Number:1550559

5-12-2025

Contents

1	Bisection	1
1.1	$fx = \sin x \sinh(x)$	1
1.2	$fx = R2 - x^{21/2} - x \tan(x)$	1
2	Newton's method	2
2.1	Newton b $fx = ex - (1 + x + (x^2/2))$	2
2.2	Newton c $fx = 10 \log x - x$	3
3	Intersect plot $fx = 100e - x - 5 \sin(\pi x/2)$	3
4	Bracketing	4
4.1	Bisect $fx = 100e - x - 5 \sin(\pi x/2)$	4
4.2	Newton $fx = 100e - x - 5 \sin(\pi x/2)$	4
5	Appendices	5
5.1	Convergence table Bisection $fx = \sin x \sinh(x)$ Error! Bookmark not defined.	
5.2	Convergence table Bisection $fx = R2 - x^{21/2} - x \tan(x)$ Error! Bookmark not defined.	
5.3	Convergence table Newton b $fx = ex - (1 + x + (x^2/2))$ Error! Bookmark not defined.	
5.4	Convergence table Newton c $fx = 10 \log x - x$ Error! Bookmark not defined.	
5.5	Convergence table Bisect $fx = 100e - x - 5 \sin(\pi x/2)$ Error! Bookmark not defined.	
5.6	Convergence table Newton $fx = 100e - x - 5 \sin(\pi x/2)$ Error! Bookmark not defined.	
5.7	Your code and functions	Error! Bookmark not defined.

1 Bisection

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

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

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

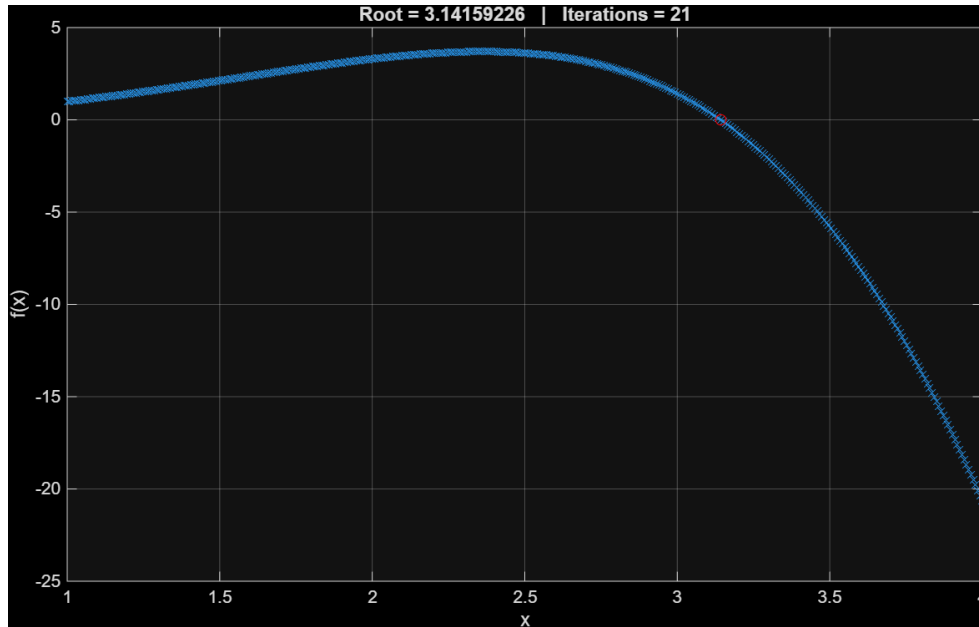


Figure 1-1 this shows the function labelled in the legend and the root

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

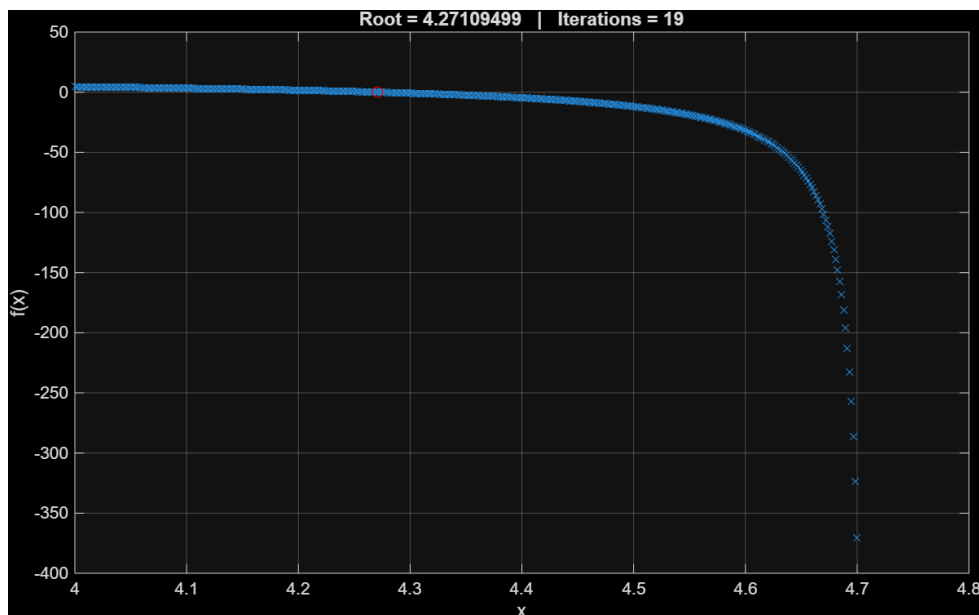


Figure 1-2 this shows the function labelled in the legend and the root

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

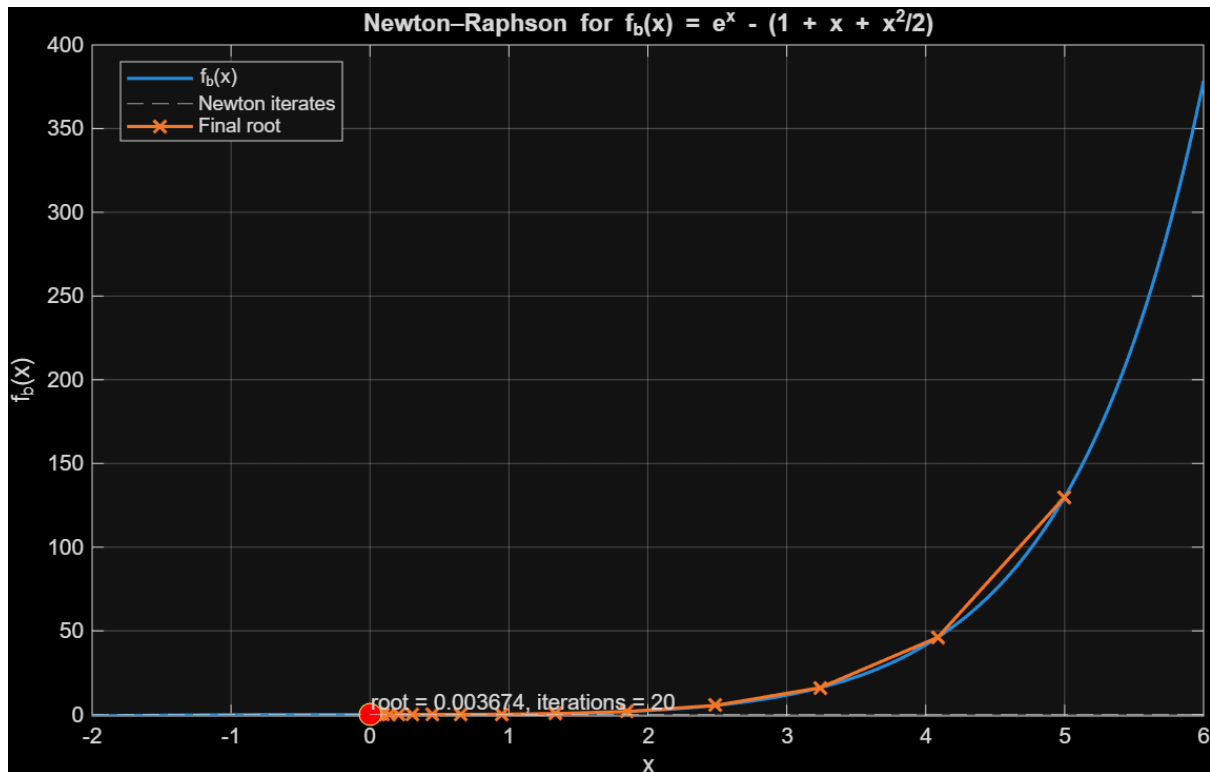


Figure 2-1 this shows the function labelled in the legend and the root

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

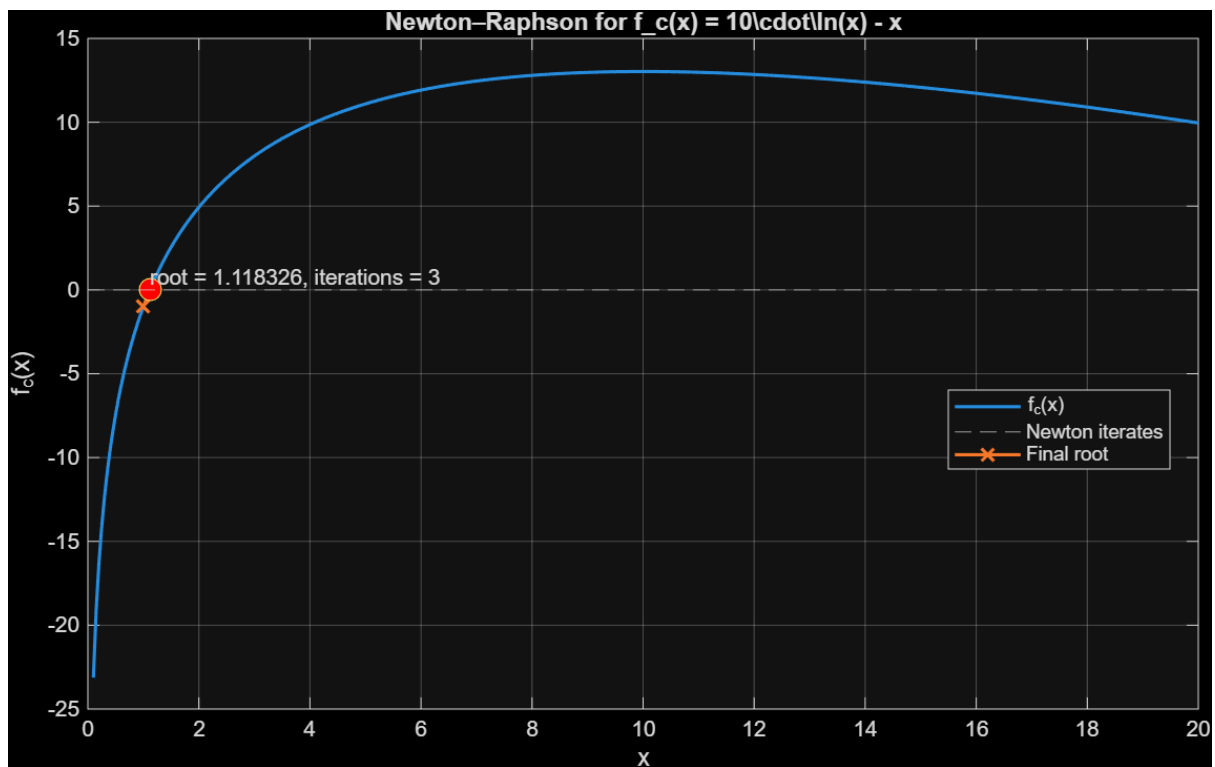


Figure 2-2 this shows the function labelled in the legend and the root

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

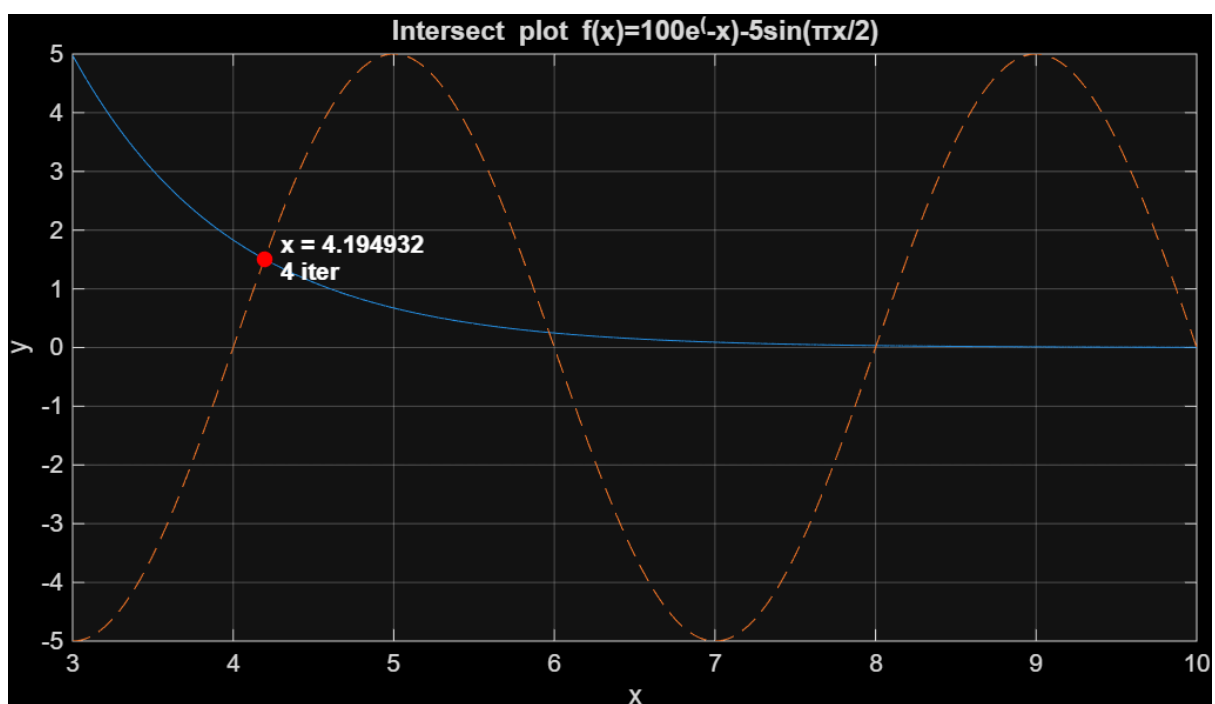


Figure 3-1 plot of function intersection

4 Bracketing

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

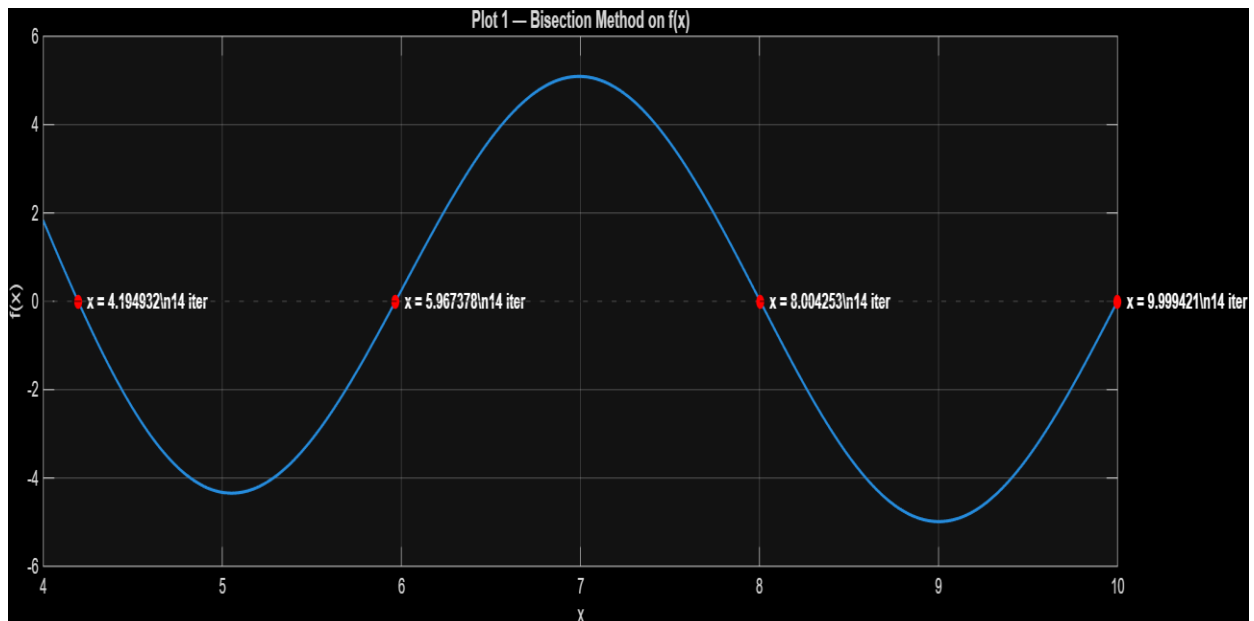


Figure 4-1 multiple roots Bisect for functions shown in legend

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

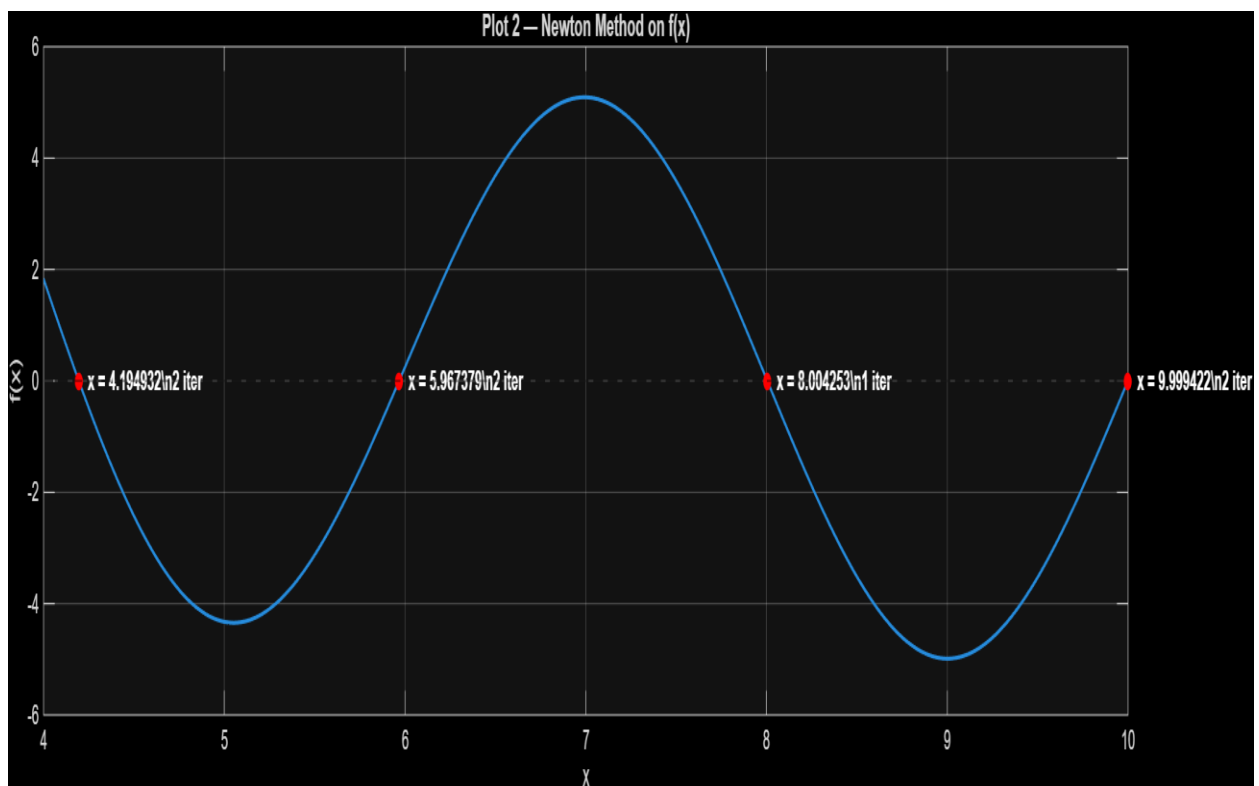


Figure 4-2 multiple roots Newton for functions shown in legend

5 Appendices

5.1.1 Lab2

%% 1. Bisection Method

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

```
tol = 1e-6;    % accuracy
imax = 50;     % maximum iterations
```

% Part (b): $f(x) = \sin(x) \cdot \sinh(x)$ on interval $[1,4]$

% lower bound

```
a = 1;
```

% upper bound

```
b = 4;
```

% function name

```
f = 'f_sinsinh';
```

% call bisection method

```
[root1, iter1] = bisect(f, a, b, imax, tol);
```

% root of the function

```
y_root1 = feval(f, root1);
```

% prepare x-y data for plotting

```
x = linspace(a, b, 400);
```

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

% plot

```
disp(['Root = ', num2str(root1, '%.8f')])
```

```
disp(['Iterations = ', num2str(iter1)])
```

```
figure(1)
```

```
plot(x, y, 'x'), hold on
```

```
plot(root1, y_root1, 'ro')
```

```
xlabel('x')
```

```
ylabel('f(x)')
```

```
grid on
```

```
title(['Root = ', num2str(root1, '%.8f'), ...  
      ' | Iterations = ', num2str(iter1)])
```

```
hold off
```

% Part (c): $f(x) = \sqrt{R^2 - x^2} - x \tan(x)$ on interval $[4,4.7]$

% lower bound

```
a = 4;
```

% upper bound

```

b = 4.7;

% function name
f = 'f_circle_tan';

% call bisection method
[root2, iter2] = bisection(f, a, b, imax, tol);

% root of the function
y_root2 = feval(f, root2);

% prepare x-y data
x2 = linspace(a, b, 400);
y2 = feval(f, x2);

% plot
disp(['Root = ', num2str(root2, '%.8f')])
disp(['Iterations = ', num2str(iter2)])

figure(2)
plot(x2, y2, 'x'), hold on
plot(root2, y_root2, 'ro')
xlabel('x')
ylabel('f(x)')

grid on
title(['Root = ', num2str(root2, '%.8f'), ...
      ' | Iterations = ', num2str(iter2)])
hold off

%% 2. Newton's Method
clc; clear;
x0 = 5;          % initial guess
imax = 50;
tol = 1e-8;

[root_b, it_b, xhist_b] = newton(x0, 'fb', 'dfb', imax, tol);

% Plot the function
figure(3);
xp = linspace(-2, 6, 400);
plot(xp, feval('fb', xp), 'LineWidth', 1.5); hold on;
yline(0, '--'); % x-axis

% Plot Newton iterates as points
plot(xhist_b, feval('fb', xhist_b), 'x-', 'LineWidth', 1.5, 'MarkerSize', 8);

% Mark the final root clearly
plot(root_b, feval('fb', root_b), 'o', 'MarkerSize', 10, 'MarkerFaceColor', 'r');

% Add text with root and iteration number
txt = sprintf('root = %.6f, iterations = %d', root_b, it_b);
text(root_b, feval('fb', root_b), txt, 'VerticalAlignment', 'bottom', ...
     'HorizontalAlignment', 'left', 'Interpreter', 'none');

xlabel('x');
ylabel('f_b(x)');

```

```

title('Newton-Raphson for  $f_b(x) = e^x - (1 + x + x^2/2)$ ');
grid on;
legend('f_b(x)', 'Newton iterates', 'Final root', 'Location', 'best');
hold off;

x0 = 1;           % initial guess
imax = 50;
tol = 1e-8;

[root_c, it_c, xhist_c] = newton(x0, 'fc', 'dfc', imax, tol);

% Plot the function (x must be > 0)
figure(4);
xp = linspace(0.1, 20, 400);
plot(xp, feval('fc', xp), 'LineWidth', 1.5); hold on;
yline(0, '--');

% Plot Newton iterates as points
plot(xhist_c, feval('fc', xhist_c), 'x-', 'LineWidth', 1.5, 'MarkerSize', 8);

% Mark the final root
plot(root_c, feval('fc', root_c), 'o', 'MarkerSize', 10, 'MarkerFaceColor', 'r');

% Add text with root and iteration number
txt = sprintf('root = %.6f, iterations = %d', root_c, it_c);
text(root_c, feval('fc', root_c), txt, 'VerticalAlignment', 'bottom', ...
     'HorizontalAlignment', 'left', 'Interpreter', 'none');

xlabel('x');
ylabel('f_c(x)');
title('Newton-Raphson for  $f_c(x) = 10 \cdot \ln(x) - x$ ');
grid on;
legend('f_c(x)', 'Newton iterates', 'Final root', 'Location', 'best');
hold off;

%% 3. Newton's Method - Intersection of two functions
clc; clear;

tol = 1e-10;
imax = 20;

% Solve using Newton
x0 = 4;
[root3, iter3] = newton(x0, 'f_int', 'df_int', imax, tol);

% Print results
fprintf('Q3 ROOT = %.12f | Newton iterations = %d\n', root3, iter3);

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

figure;
plot(x, yL, x, yR, '--'); hold on;

yroot = 100*exp(-root3);
plot(root3, yroot, 'ro', 'MarkerFaceColor', 'r');

```

```

% Add root label
text(root3 + 0.1, yroot, ...
    sprintf('x = %.6f\n%d iter', root3, iter3), ...
    'FontSize',10,'Color','w','FontWeight','bold');

title('Intersect plot f(x)=100e^(-x)-5sin(πx/2)');
xlabel('x'); ylabel('y');
grid on;

%% 4. Bracketing

clc; clear;

xmin = 4;
xmax = 10;
n = 200;
tol = 1e-6;
imax = 50;

% (a) BRACKETING

x = linspace(xmin, xmax, n);
y = f_int(x);

brackets = [];

for k = 1:length(x)-1
    fa = y(k);
    fb = y(k+1);

    if fa * fb < 0
        brackets = [brackets; x(k) x(k+1)];
    end
end

fprintf('Valid bracketing intervals:\n');
disp(brackets);

nroots = size(brackets,1);

% (b) Bisection + Newton

root_bis = zeros(nroots,1);
iter_bis = zeros(nroots,1);

root_new = zeros(nroots,1);
iter_new = zeros(nroots,1);

for i = 1:nroots
    a = brackets(i,1);
    b = brackets(i,2);

    [root_bis(i), iter_bis(i)] = bisect('f_int', a, b, imax, tol);

```

```

    x0 = (a + b)/2;
    [root_new(i), iter_new(i)] = newton(x0, 'f_int', 'df_int', imax, tol);
end

% PRINT RESULTS

fprintf('\nROOT RESULTS (Q4):\n');
fprintf('-----\n');
fprintf('Root#      BisectionRoot      Iter      NewtonRoot      Iter\n');

for i = 1:nroots
    fprintf('%2d      %12.6f      %3d      %12.6f      %3d\n', ...
        i, root_bis(i), iter_bis(i), root_new(i), iter_new(i));
end

% (c) PLOT 1 – BISECTION on f(x)

xplot = linspace(xmin, xmax, 400);
fplot = f_int(xplot);

figure(6);
plot(xplot, fplot, 'LineWidth',1.5); hold on;
yline(0,'k--');
title('Plot 1 – Bisection Method on f(x)');
xlabel('x'); ylabel('f(x)');
grid on;

% Plot bisection roots
plot(root_bis, zeros(size(root_bis)), 'ro', 'MarkerFaceColor','r');

% Labels
for i = 1:length(root_bis)
    xk = root_bis(i);
    yk = feval('f_int', xk); % or whatever y-value you use

    % Two-line label WITHOUT \n
    txt = { ...
        sprintf('x = %.6f', xk), ...
        sprintf('%d iter', iter_bis(i)) ...
    };

    text(xk, yk, txt, ...
        'VerticalAlignment', 'bottom', ...
        'HorizontalAlignment', 'left');
end

% (d) PLOT 2 – NEWTON on f(x)

figure(7);
plot(xplot, fplot, 'LineWidth',1.5); hold on;
yline(0,'k--');
title('Plot 2 – Newton Method on f(x)');
xlabel('x'); ylabel('f(x)');
grid on;

% Plot Newton roots
plot(root_new, zeros(size(root_new)), 'ro', 'MarkerFaceColor','r');

% Labels

```

```

for i = 1:length(root_bis)
    xk = root_bis(i);
    yk = feval('f_int', xk); % or whatever y-value you use

    % Two-line label WITHOUT \n
    txt = { ...
        sprintf('x = %.6f', xk), ...
        sprintf('%d iter', iter_bis(i)) ...
    };

    text(xk, yk, txt, ...
        'VerticalAlignment', 'bottom', ...
        'HorizontalAlignment', 'left');
end
end

```

5.1.2 Bisection

```

%% 1(a)
function [root, iter] = bisection(f, a, b, imax, tol)
%BISECTION Bisection method for solving  $f(x) = 0$  on  $[a,b]$ 

% f      function to be evaluated (string with function name)
% a, b    interval limits
% imax    maximum number of iterations
% tol     convergence criterion (on interval length)
%
% Output:
% root     position of the root
% iter     number of iterations used

% 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 [a,b].');
end

% Begin iteration loop i.e. for ndx =1:imax
for ndx = 1:imax

    % Midpoint
    x2 = (x3 + x1) / 2;
    f2 = feval(f, x2);

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

```

```

        end

        % test for convergence i.e. d < tol
        d = abs(x3 - x1); % current interval length
        if d < tol
            root = x2;
            iter = ndx;
            return
        end
    end
end

% CHECK: Excessive iterations
warning('Maximum number of iterations reached without satisfying tolerance.');
```

root = x2; % last midpoint as best estimate

iter = imax;

end

5.1.3 F_sinsinh

```

function y = f_sinsinh(x)
% f(x) = sin(x) * sinh(x)
y = sin(x) .* sinh(x);
end
```

5.1.4 f_circle_tan

```

function y = f_circle_tan(x)
% f(x) = sqrt(R^2 - x^2) - x * tan(x), R = 10
R = 10;
y = sqrt(R^2 - x.^2) - x .* tan(x);
end
```

5.1.5 newton

```

function [root, iters, xhist] = newton(x, f, df, imax, tol)

% initialise parameters
xold = x;
fold = feval(f, xold);

% store iteration history
xhist = xold; % first entry is initial guess

% iteration counter
count = 0;

% loop until converged or max iterations reached
while (abs(fold) > tol) && (count < imax)

    % derivative at current point
    dfold = feval(df, xold);

    % CHECK: verify that the derivative is not zero
    if dfold == 0
        error('Problem: derivative is zero, Newton method fails');
    end
```

```

    % Newton step:  $x_{n+1} = x_n - f(x_n)/f'(x_n)$ 
    dx = -fold / dfold;
    xnew = xold + dx;

    % function value at new point
    fr = feval(f, xnew);

    % update iteration counter
    count = count + 1;

    % store new iterate
    xhist(end+1) = xnew; %#ok<AGROW>

    % print iteration info
    fprintf('Iteration - %i,\t Error - %g\n', count, abs(dx));

    % update for next loop
    xold = xnew;
    fold = fr;
end

if count == imax
    warning('Maximum number of iterations reached without full convergence');
end

root = xold;
iters = count;
end

```

5.1.6 fb

```

function y = fb(x)
%  $f(x) = e^x - (1 + x + x^2/2)$ 
y = exp(x) - (1 + x + x.^2/2);
end

```

5.1.7 dfb

```

function dy = dfb(x)
%  $f'(x) = e^x - (1 + x)$ 
dy = exp(x) - (1 + x);
end

```

5.1.8 fc

```

function y = fc(x)
%  $f(x) = 10*\ln(x) - x$ 
y = 10*log(x) - x;
end

```

5.1.9 dfc

```

function dy = dfc(x)
%  $f'(x) = 10/x - 1$ 
dy = 10./x - 1;
end

```

5.1.10 f_int

```
function y = f_int(x)
% f(x) = 100 e^{-x} - 5 sin(pi x / 2)
    y = 100*exp(-x) - 5*sin(pi*x/2);
end
```

5.1.11 df_int

```
function dy = df_int(x)
% f'(x) = -100 e^{-x} - (5*pi/2) cos(pi x / 2)
    dy = -100*exp(-x) - (5*pi/2)*cos(pi*x/2);
end
```