

UWTSD

Module

Task 2

Name :- ARTH PANDYA

**Module Name:- Alternative fuels and Combustion Modelling
(ACHA7002)**

Student number :- 2412514

Date :- 7/12/25

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 $fx = ex - (1 + x + (x^2/2))$	3
2.2	Newton c $fx = 10 \log x - x$	3
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 $fx = 100e - x - 5 \sin(\pi x/2)$	6
5	Appendices	6
5.1	Convergence table Bisection $fx = \sin x \sinh(x)$	6
5.2	Convergence table Bisection $fx = R2 - x^{21/2} - x \tan(x)$	6
5.3	Convergence table Newton b $fx = ex - (1 + x + (x^2/2))$	7
5.4	Convergence table Newton c $fx = 10 \log x - x$	7
5.5	Convergence table Bisect $fx = 100e - x - 5 \sin(\pi x/2)$	7
5.6	Convergence table Newton $fx = 100e - x - 5 \sin(\pi x/2)$	7
5.7	Your code and functions	8

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

BISECTION METHOD - Function 1

Function: $\sin(x)*\sinh(x)$, Interval: [1, 4]

Root found: 3.141593

Iterations: 22

$f(\text{root}) = 3.670941\text{e-}07$

BISECTION METHOD - Function 2

Function: $(1+x-10^{(2/3)})/x^2$, Interval: [4, 4.7]

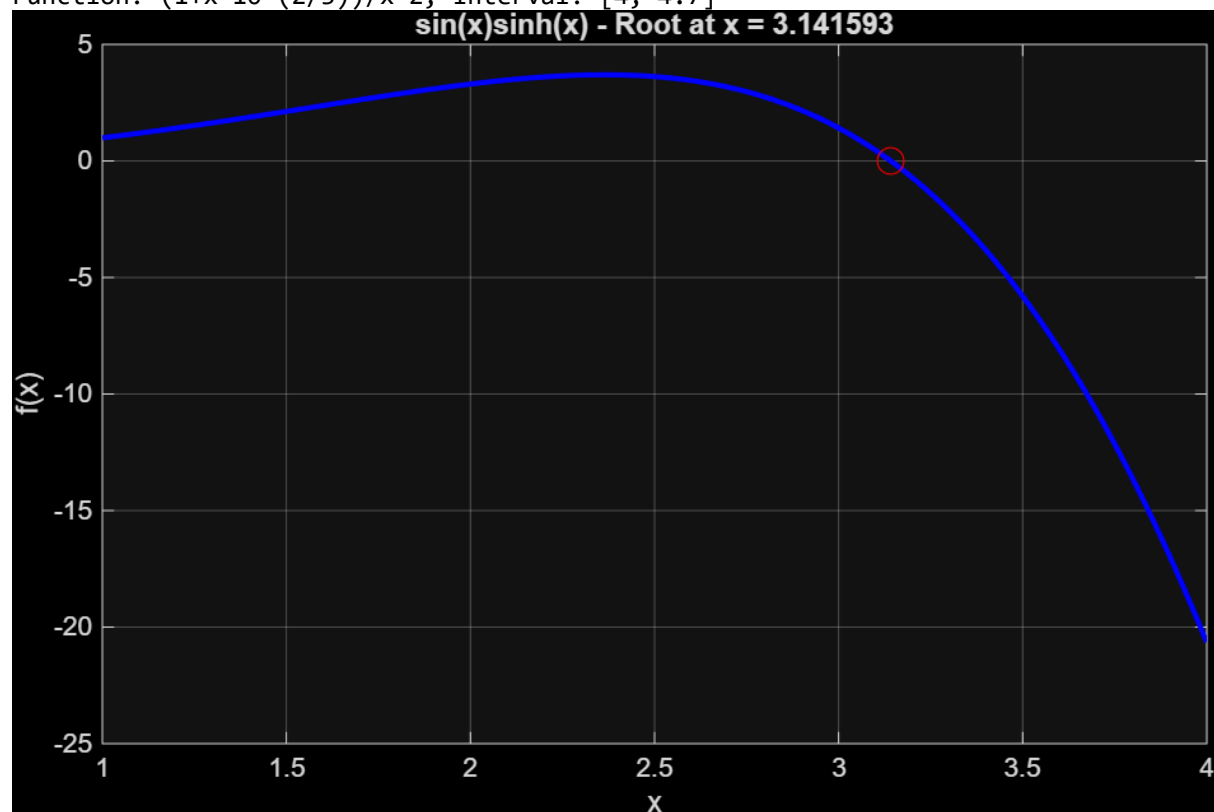


Figure 1-1 this shows the function labelled in the legend and the root

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

BISECTION METHOD - Function 2

Function: $(1+x-10^{(2/3)})/x^2$, Interval: [2, 5]

Root found: 3.641589

Iterations: 22

$f(\text{root}) = -1.998825\text{e-}09$

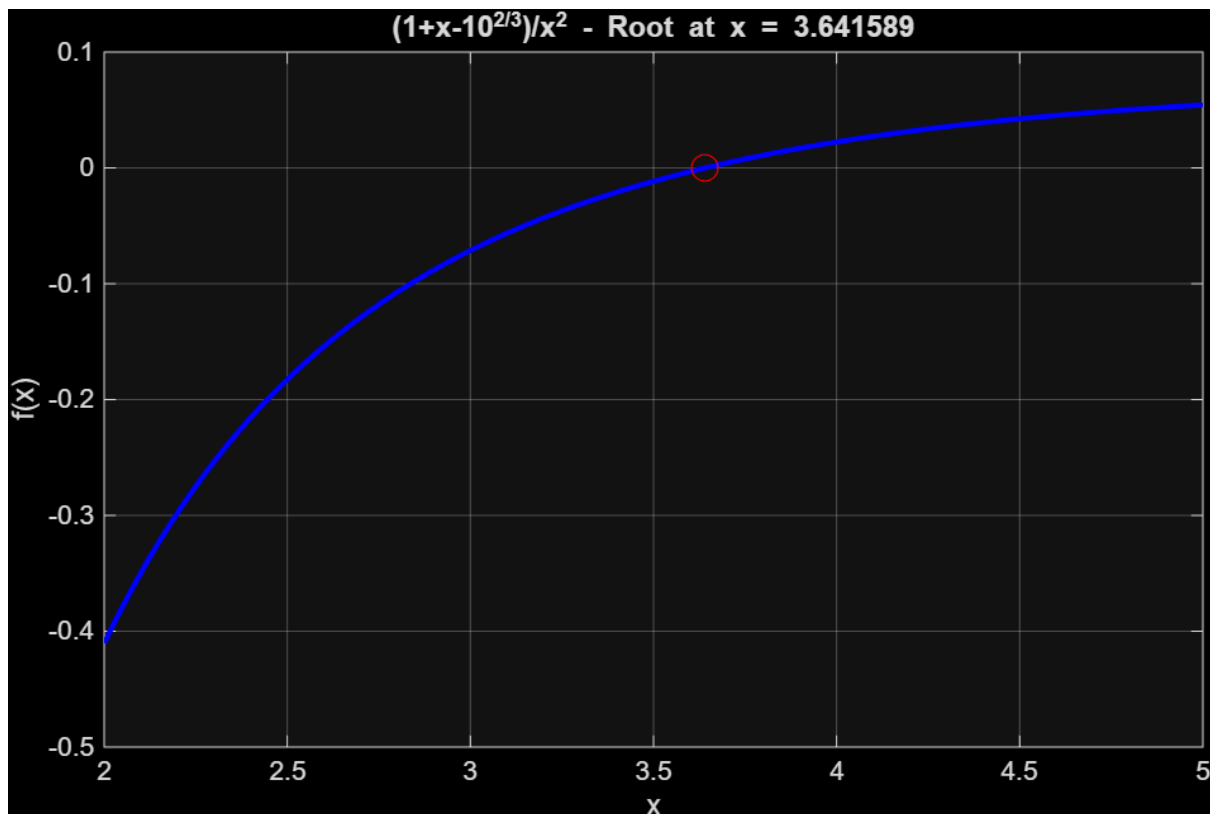


Figure 1-2 this shows the function labelled in the legend and the root

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

NEWTON-RAPHSON METHOD - Function 1

Function: $e^x - (1 + x + x^2/2)$, $x_0 = 5$

Root found: 0.003674

Iterations: 20

$f(\text{root}) = 8.270474e-09$

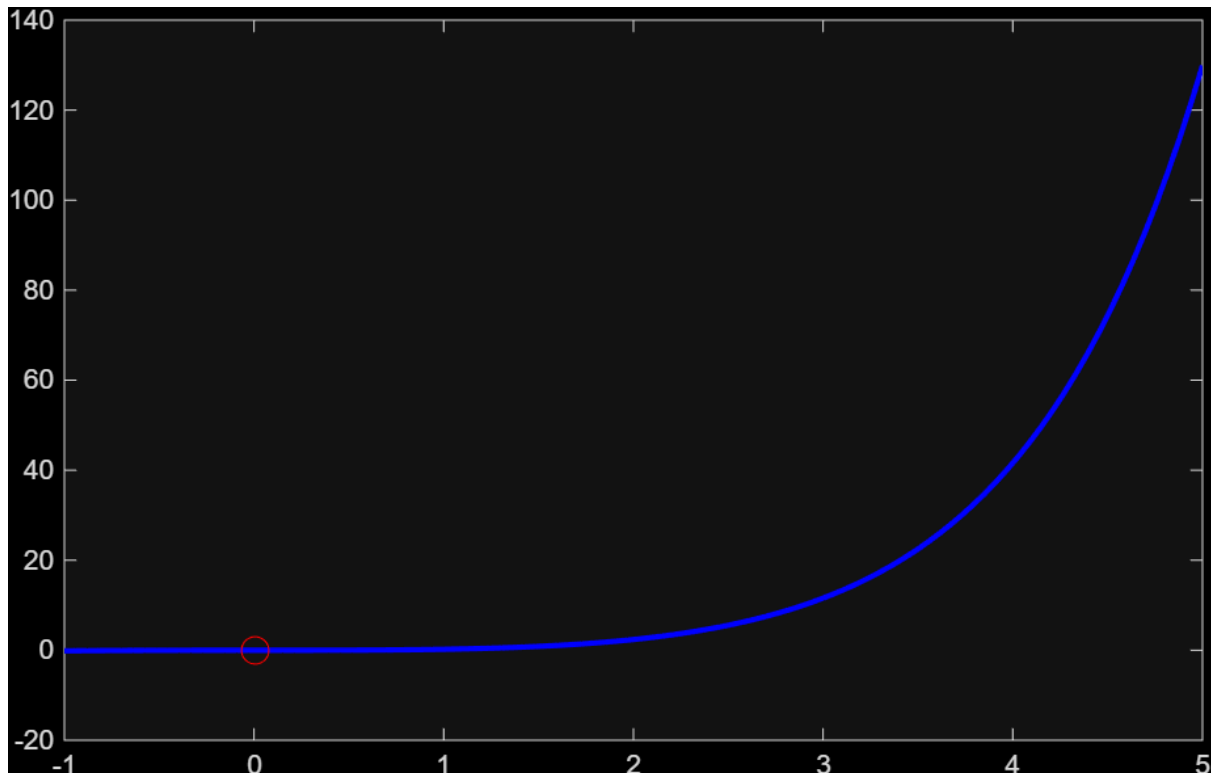


Figure 2-1 this shows the function labelled in the legend and the root

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

NEWTON-RAPHSON METHOD - Function 2

Function: $10 \cdot \log(x) - x$, $x_0 = 1$

Root found: 1.118326

Iterations: 4

$f(\text{root}) = -2.751923e-09$

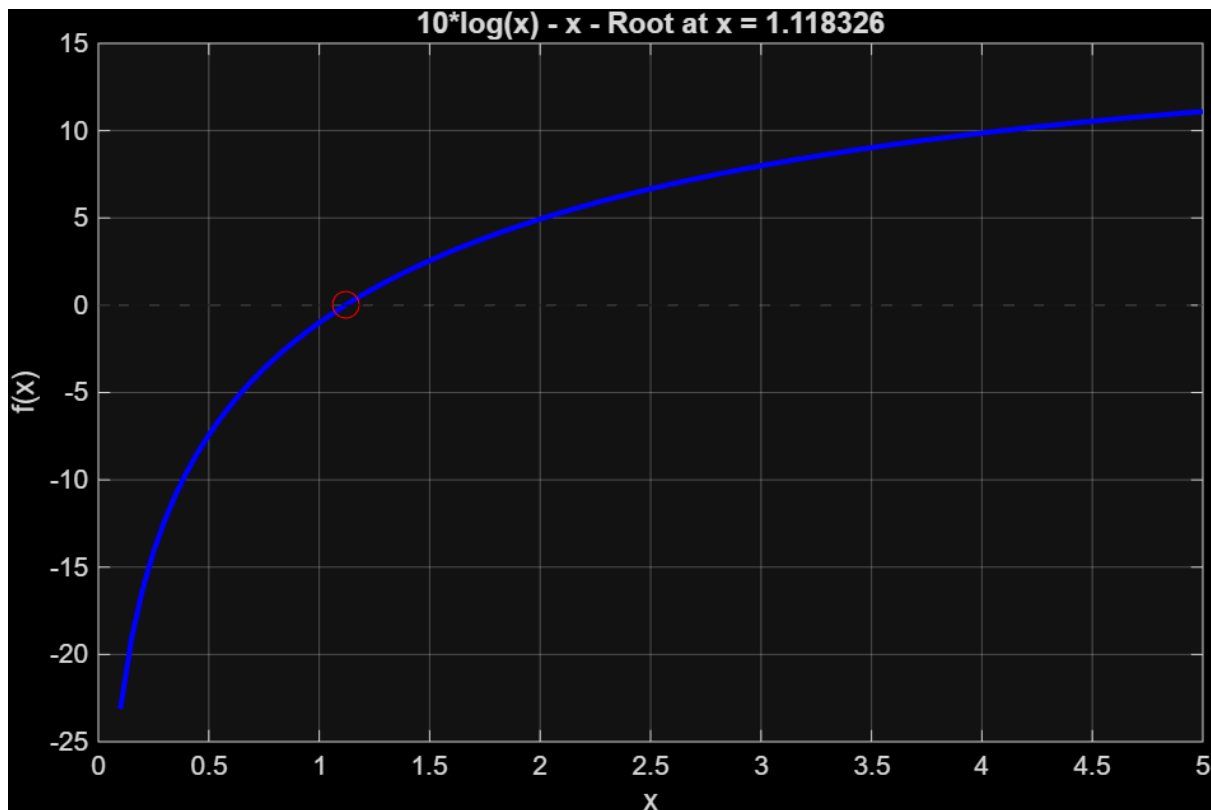


Figure 2-2 this shows the function labelled in the legend and the root

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

Finding root near $x_0 = 4$ with $\text{tol} = 1e-10$

Root: 4.194932

Iterations: 4

$f(\text{root})$: $2.464180e-10$

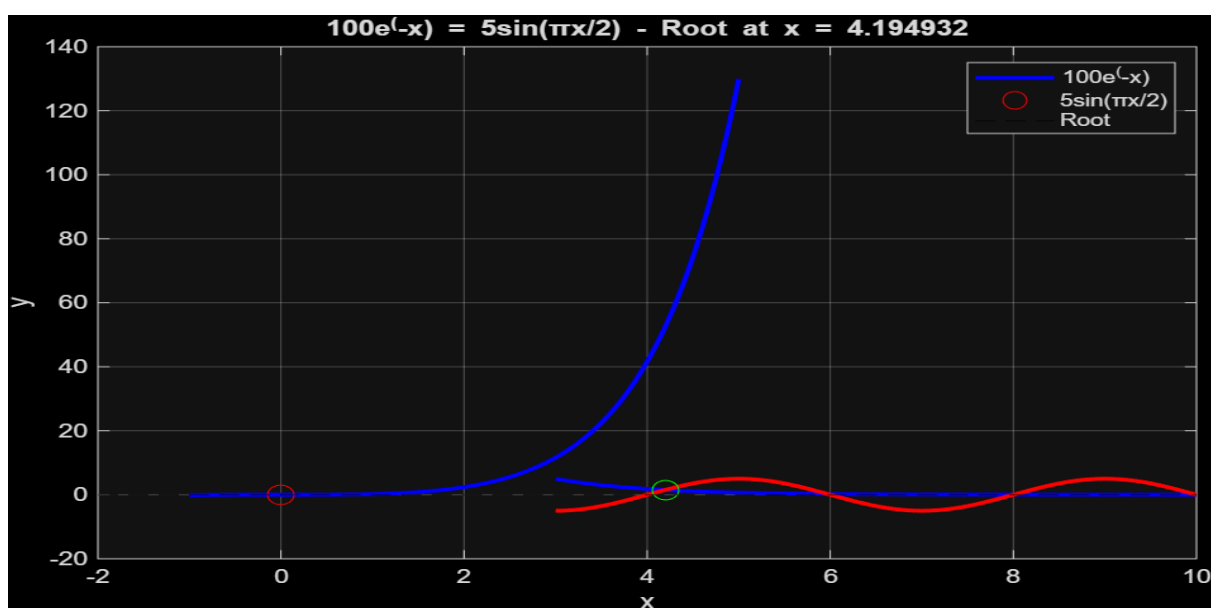


Figure 3-1 plot of function intersection

4 Bracketing

BRACKETING AND MULTIPLE ROOTS

Finding roots of: $100e^{-x} - 5\sin(\pi x/2) = 0$

Found 4 potential roots in $[4, 10]$

Root 1: Bisect = 4.19493196, Newton = 4.19493157

Root 2: Bisect = 5.96737902, Newton = 5.96737906

Root 3: Bisect = 8.00425350, Newton = 8.00425315

Root 4: Bisect = 9.99942155, Newton = 9.99942162

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

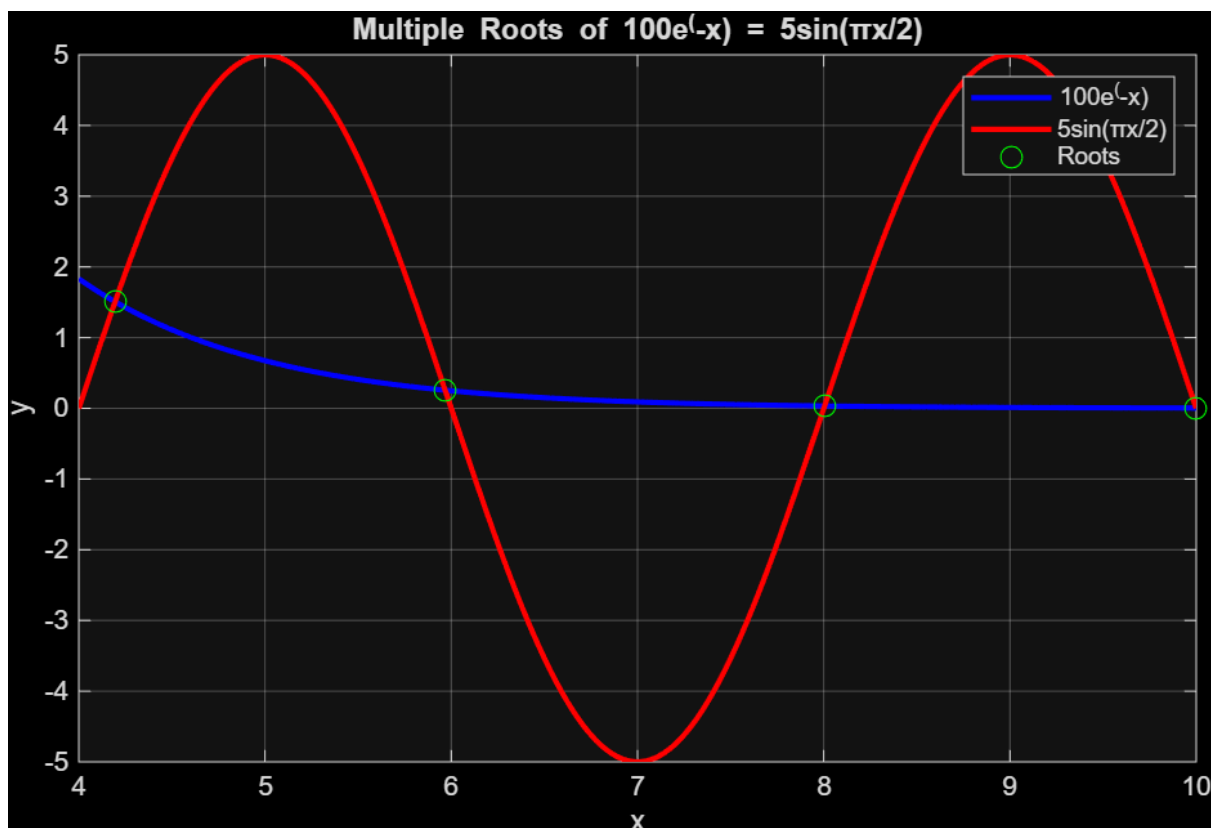


Figure 4-1 multiple roots Bisect for functions shown in legend

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

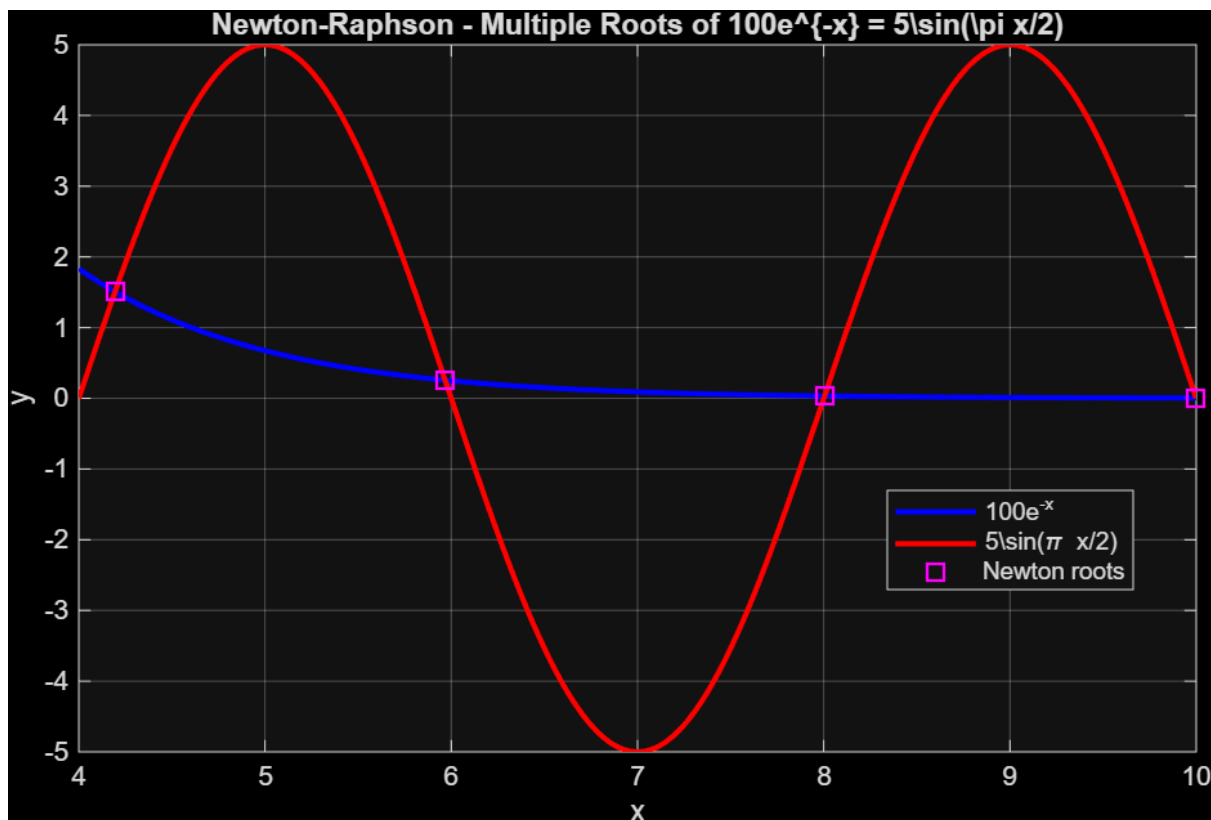


Figure 4-2 multiple roots Newton for functions shown in legend

5 Appendices

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

BISECTION METHOD - Function 1

Function: $\sin(x) \sinh(x)$, Interval: [1, 4]

Root found: 3.141593

Iterations: 22

$f(\text{root}) = 3.670941e-07$

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

BISECTION METHOD - Function 2

Function: $(1+x-10^{(2/3)})/x^2$, Interval: [2, 5]

Root found: 3.641589

Iterations: 22

$f(\text{root}) = -1.998825e-09$

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

NEWTON-RAPHSON METHOD - Function 1

Function: $e^x - (1 + x + x^2/2)$, $x_0 = 5$

Root found: 0.003674

Iterations: 20

$f(\text{root}) = 8.270474e-09$

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

NEWTON-RAPHSON METHOD - Function 2

Function: $10 \log(x) - x$, $x_0 = 1$

Root found: 1.118326

Iterations: 4

$f(\text{root}) = -2.751923e-09$

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

BRACKETING AND MULTIPLE ROOTS

Finding roots of: $100e^{-x} - 5\sin(\pi x/2) = 0$

Found 4 potential roots in [4, 10]

Root 1: Bisect = 4.19493196

Root 2: Bisect = 5.96737902

Root 3: Bisect = 8.00425350

Root 4: Bisect = 9.99942155

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

BRACKETING AND MULTIPLE ROOTS

Finding roots of: $100e^{-x} - 5\sin(\pi x/2) = 0$

Found 4 potential roots in [4, 10]

Root 1: Newton = 4.19493157

Root 2: Newton = 5.96737906

Root 3: Newton = 8.00425315

Root 4: Newton = 9.99942162

5.7 Your code and functions

5.7.1 bisect.m (function)

```
1 function [root, iterations] = bisect(f, a, b, imax, tol)
2 % Bisection method for root finding
3 % Find root of f(x) = 0 in interval [a, b]
4
5 % Check if root exists in bracket
6 f_a = feval(f, a);
7 f_b = feval(f, b);
8
9 if f_a * f_b > 0
10     error('No root in the specified interval [a, b]');
11 end
12
13 iterations = 0;
14
15 while (b - a) > tol && iterations < imax
16     iterations = iterations + 1;
17
```

5.7.2 bracketing.m (function)

```
1 function brackets = bracketing(f, a, b, n_eval)
2 % Find approximate root locations by sign changes
3 % Returns array of bracket pairs [a1 b1; a2 b2; ...]
4
5 x = linspace(a, b, n_eval);
6 y = feval(f, x);
7
8 brackets = [];
9 for i = 1:length(y)-1
10     if y(i) * y(i+1) < 0 % Sign change
11         brackets = [brackets; x(i), x(i+1)];
12     end
13
```

5.7.3 deriv_multiple.m (function)

```
2 function dy = deriv_multiple(x)
3 % f'(x) = -100*e^(-x) - 5*(pi/2)*cos(pi*x/2)
4 dy = -100 * exp(-x) - 5 * (pi/2) * cos(pi * x / 2);
5 end
```

5.7.4 deriv_newton_1.m (function)

```
2 function dy = deriv_newton_1(x)
3 % f'(x) = e^x - (1 + x)
4 dy = exp(x) - (1 + x);
5 end
```

5.7.5 deriv_newton_2.m (function)

```
2 function dy = deriv_newton_2(x)
3 % f'(x) = 10/x - 1
4 dy = 10 ./ x - 1;
5 end
```

5.7.6 func_bisect_1.m (function)

```
1 function y = func_bisect_1(x)
2 % f(x) = sin(x) * sinh(x)
3 y = sin(x) .* sinh(x);
4 end
```

5.7.7 func_bisect_2.m (function)

```
1 function y = func_bisect_2(x)
2 % f(x) = (1 + x - R^(2/3)) / x^2, where R = 10
3 R = 10;
4 y = (1 + x - R^(2/3)) ./ (x.^2);
5 end
```

5.7.8 func_multiple.m (function)

```
1 % func_multiple.m
2 function y = func_multiple(x)
3 % f(x) = 100*e^(-x) - 5*sin(pi*x/2)
4 y = 100 * exp(-x) - 5 * sin(pi * x / 2);
5 end
```

5.7.9 func_newton_1.m (function)

```
2 function y = func_newton_1(x)
3 % f(x) = e^x - (1 + x + x^2/2)
4 y = exp(x) - (1 + x + x.^2/2);
5 end
```

5.7.10 func_newton_2.m (function)

```
1 % func_newton_2.m
2 function y = func_newton_2(x)
3 % f(x) = 10*log(x) - x
4 y = 10 * log(x) - x;
5 end
```

5.7.11 newton.m (function)

```
1 function [root, iterations] = newton(f, df, x0, imax, tol)
2 % Newton-Raphson method
3 % Input: f - function, df - derivative, x0 - initial guess
4 % Output: root, iterations
5
6 x = x0;
7 iterations = 0;
8
9 while iterations < imax
10     f_val = feval(f, x);
11     df_val = feval(df, x);
12
13     if abs(df_val) < 1e-12
14         error('Derivative too small - Newton method failed');
15     end
16
17     x_new = x - f_val / df_val;
18
19     if abs(x_new - x) < tol
20         iterations = iterations + 1;
21         break;
22     end
```

5.7.12 task2_bisection.m (script)

```
1 clear; close all;
2
3 % Function 1: sin(x)*sinh(x) on [1, 4]
4 fprintf('BISECTION METHOD - Function 1\n');
5 fprintf('Function: sin(x)*sinh(x), Interval: [1, 4]\n\n');
6
7 [root1, iter1] = bisect('func_bisect_1', 1, 4, 50, 1e-6);
8
9 fprintf('Root found: %f\n', root1);
10 fprintf('Iterations: %d\n', iter1);
11 fprintf('f(root) = %e\n\n', func_bisect_1(root1));
12
13 % Plot
14 x = linspace(1, 4, 100);
15 y = func_bisect_1(x);
16 figure(1);
17 plot(x, y, 'b-', 'LineWidth', 2);
18 hold on;
19 plot(root1, func_bisect_1(root1), 'ro', 'MarkerSize', 10);
20 xlabel('x');
21 ylabel('f(x)');
22 title(sprintf('sin(x)sinh(x) - Root at x = %.6f', root1));
23 grid on;
```

5.7.13 task2_bracketing.m (script)

```

1  clear; close all;
2
3  fprintf('BRACKETING AND MULTIPLE ROOTS\n');
4  fprintf('Finding roots of: 100*e^(-x) - 5*sin(pi*x/2) = 0\n\n');
5
6  % 4.1 Bracketing in [4,10]
7  brackets = bracketing('func_multiple', 4, 10, 100);
8  fprintf('Found %d potential roots in [4, 10]\n\n', size(brackets, 1));
9
10 % Arrays for storing roots
11 bisect_roots = [];
12 newton_roots = [];
13
14 for i = 1:size(brackets, 1)
15     a = brackets(i, 1);
16     b = brackets(i, 2);
17
18     % --- Bisection ---
19     [r_bisect, ~] = bisect('func_multiple', a, b, 50, 1e-6);
20     bisect_roots = [bisect_roots; r_bisect]; %#ok<AGROW>
21
22     % --- Newton (midpoint as x0) ---
23     x0 = (a + b) / 2;

```

5.7.14 task2_multiple.m(script)

```

1  % Find root near x0 = 4 with high tolerance
2  fprintf('Finding root near x0 = 4 with tol = 1e-10\n');
3
4  [root, iter] = newton('func_multiple', 'deriv_multiple', 4, 50,
5
6  fprintf('Root: %f\n', root);
7  fprintf('Iterations: %d\n', iter);
8  fprintf('f(root): %e\n', feval('func_multiple', root));
9
10 % Plot both functions
11 x = linspace(3, 10, 200);
12 y1 = 100 * exp(-x);
13 y2 = 5 * sin(pi * x / 2);
14
15 figure(1);
16 plot(x, y1, 'b-', 'LineWidth', 2);
17 hold on;
18 plot(x, y2, 'r-', 'LineWidth', 2);
19 plot(root, 100*exp(-root), 'go', 'MarkerSize', 10);
20 xlabel('x');
21 ylabel('y');
22 title(sprintf('100e^(-x) = 5sin(πx/2) - Root at x = %.6f', root));
23 legend('100e^(-x)', '5sin(πx/2)', 'Root');

```

5.7.15 task2_newton.m (script)

```
1 clear; close all;
2
3 % Function 1
4 fprintf('NEWTON-RAPHSON METHOD - Function 1\n');
5 fprintf('Function: e^x - (1 + x + x^2/2), x0 = 5\n\n');
6
7 [root1, iter1] = newton('func_newton_1', 'deriv_newton_1', 5,
8
9 fprintf('Root found: %f\n', root1);
10 fprintf('Iterations: %d\n', iter1);
11 fprintf('f(root) = %e\n\n', feval('func_newton_1', root1));
12
13 % Plot for Function 1
14 x = linspace(-1, 5, 100);
15 y = feval('func_newton_1', x);
16 figure(1);
17 plot(x, y, 'b-', 'LineWidth', 2);
18 hold on;
19 plot(root1, feval('func_newton_1', root1), 'ro', 'MarkerSize',
20 yline(0, 'k--'); % horizontal axis line at y = 0
21 xlabel('x');
22 ylabel('f(x)');
23 title(sprintf('e^x - (1 + x + x^2/2) - Root at x = %6f', root1));
```