

UWTSD

Module

Task 2

Module code: ACHA7002

Student number: 2200899

7/12/2025

Contents

1	Bisection	1
1.1	$fx = \sin x \sinh(x)$	1
1.2	$fx = R^2 - x^{21/2} - x \tan(x)$	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 $fx = \sin x \sinh(x)$	5
5.2	Convergence table Bisection $fx = R^2 - x^{21/2} - x \tan(x)$	5
5.3	Convergence table Newton b $fx = ex - (1 + x + (x^2/2))$	6
5.4	Convergence table Newton c $fx = 10 \log x - x$	6
5.5	Convergence table Bisect $fx = 100e - x - 5 \sin(\pi x/2)$	6
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 $f(x) = \sin(x) \sinh(x)$

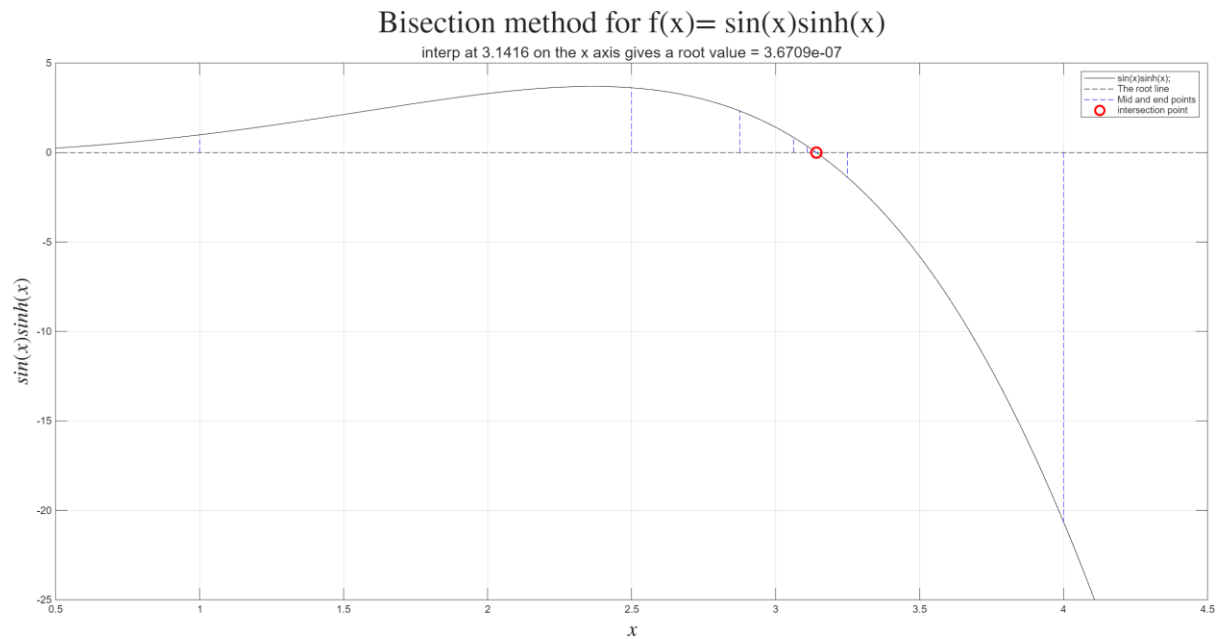


Figure 1-1 this shows the function labelled in the legend and the root

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

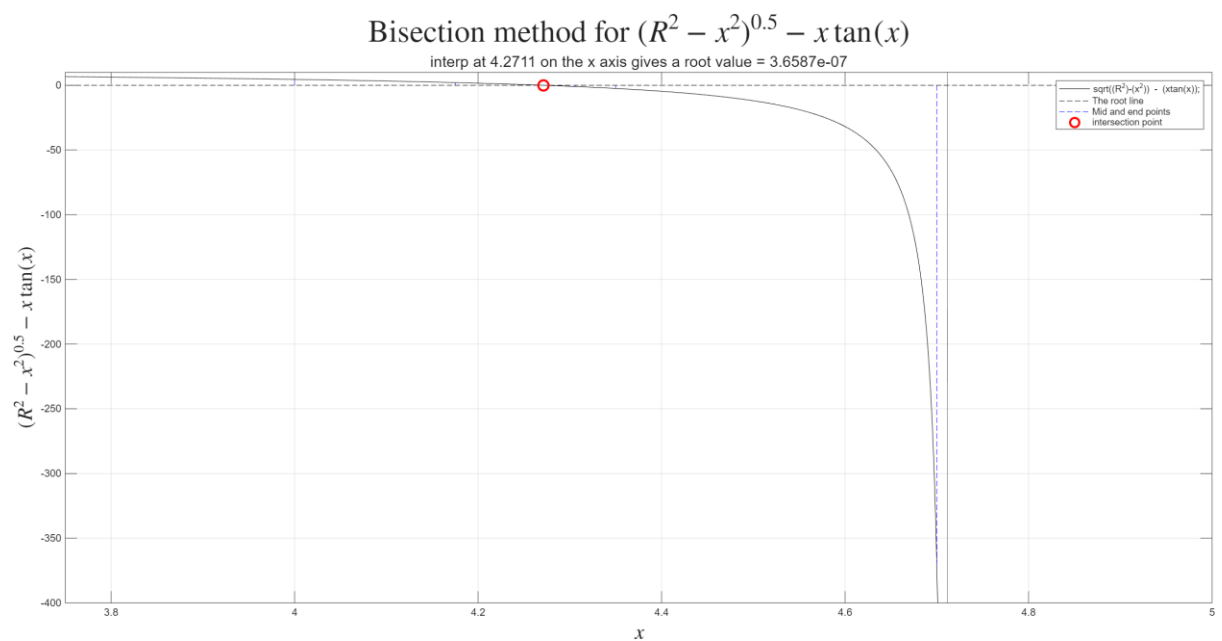


Figure 1-2 this shows the function labelled in the legend and the root

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

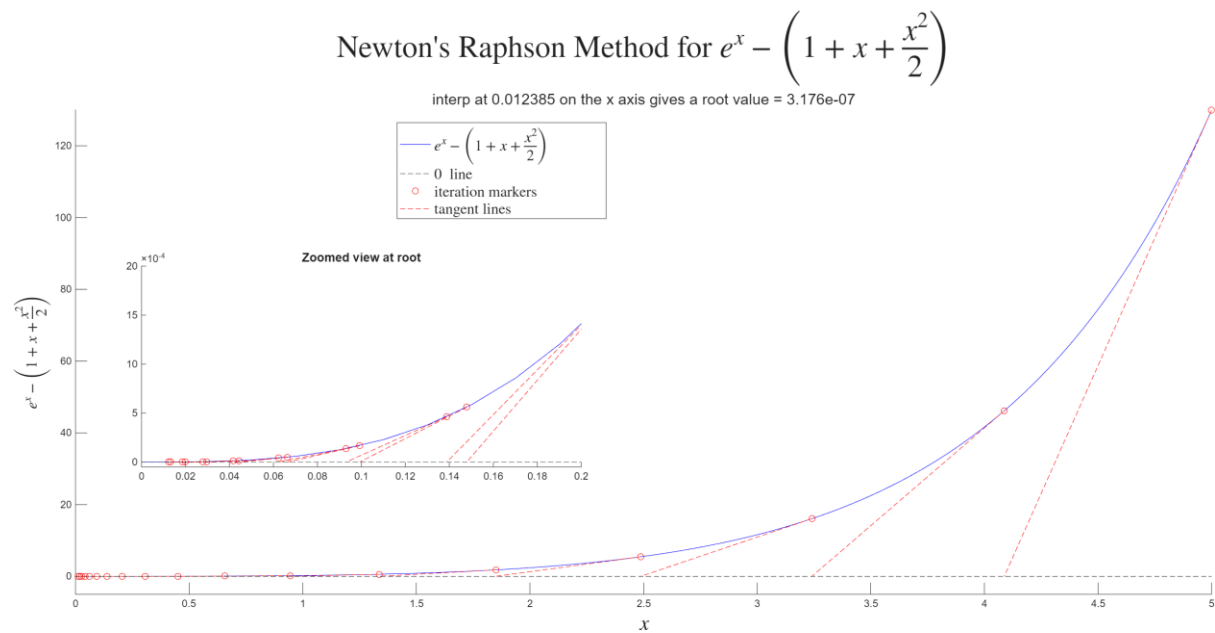


Figure 2-1 this shows the function labelled in the legend and the root

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

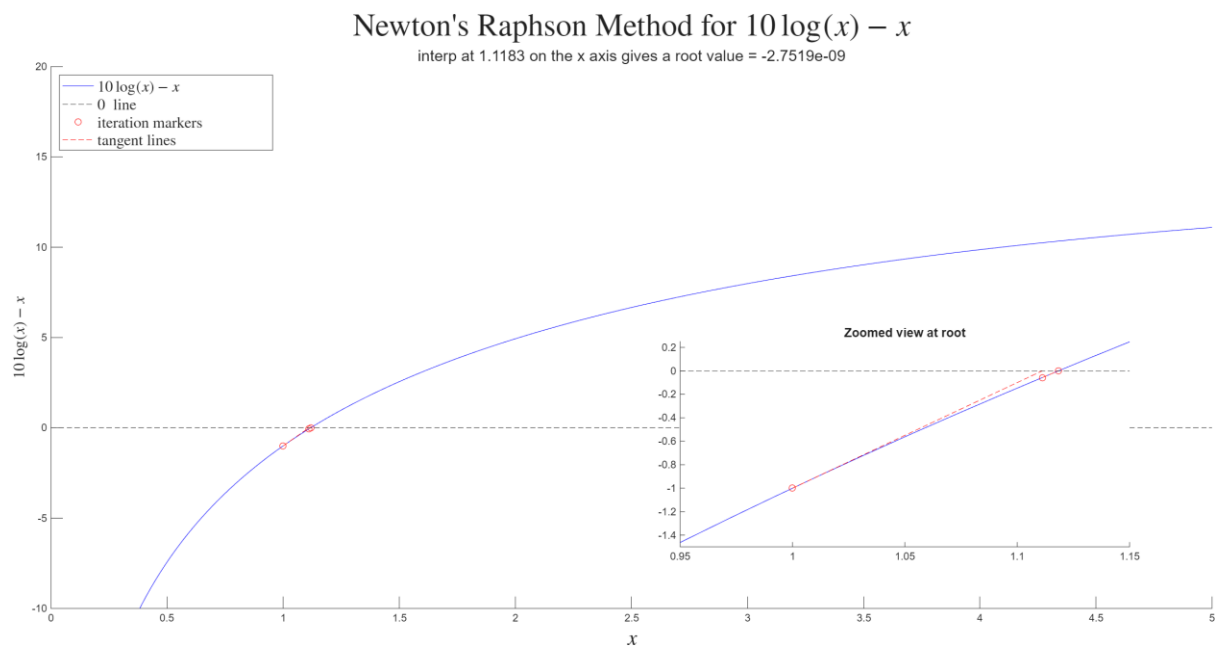


Figure 2-2 this shows the function labelled in the legend and the root

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

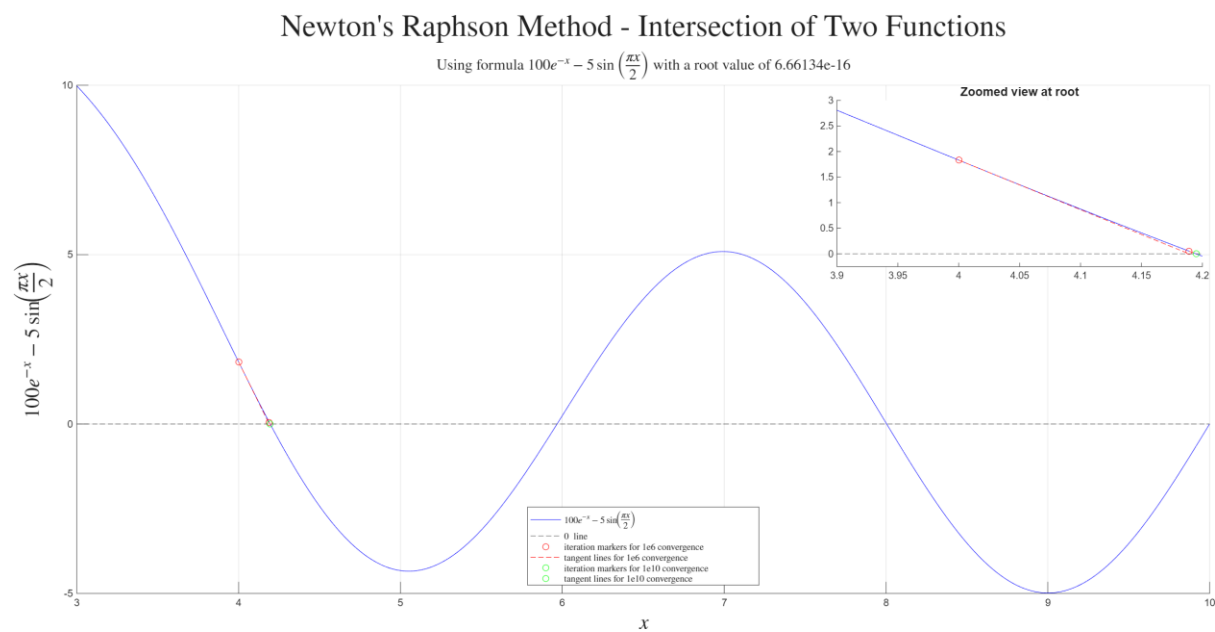


Figure 3-1 plot of function intersection

4 Bracketing

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

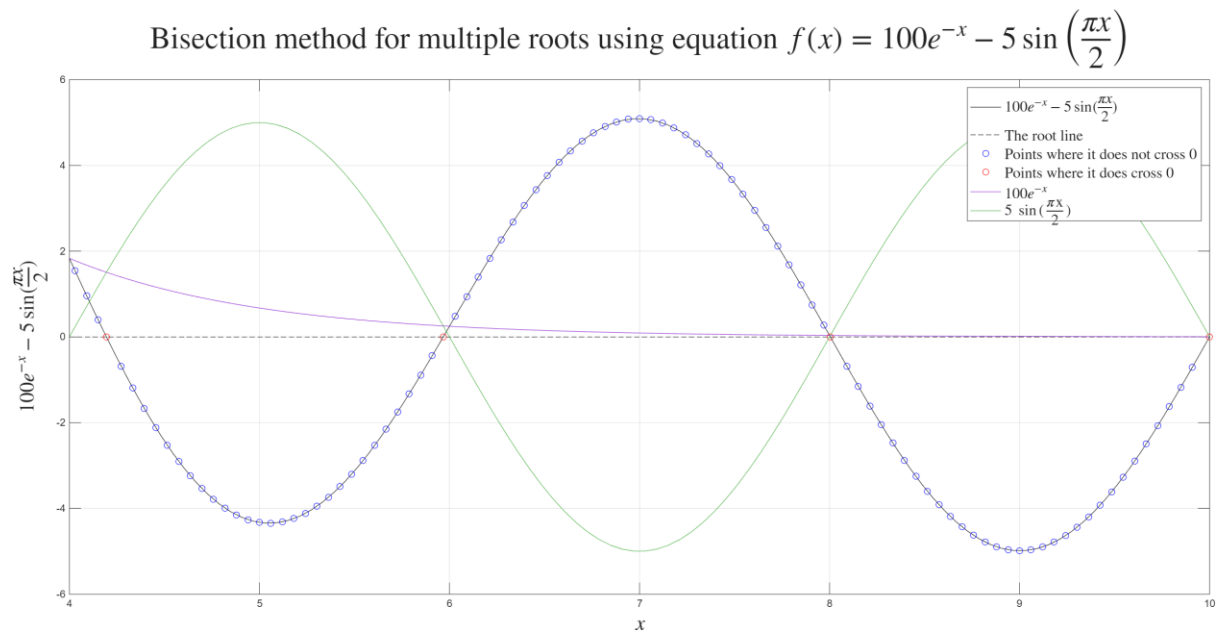


Figure 4-1 multiple roots Bisect for functions shown in legend

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

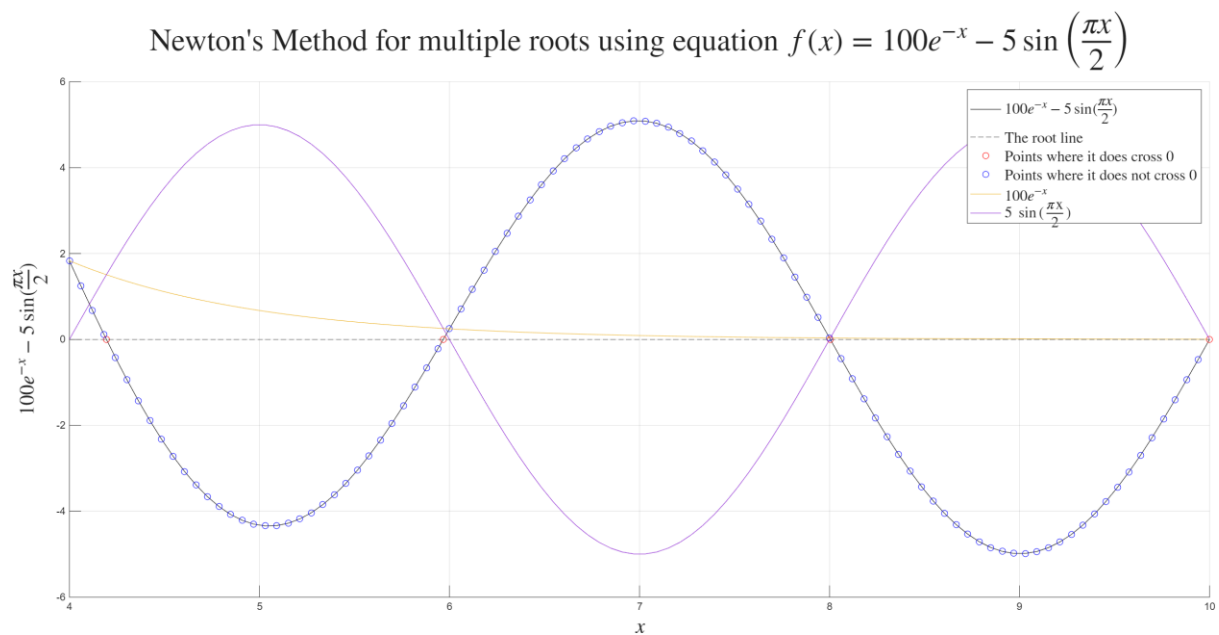


Figure 4-2 multiple roots Newton for functions shown in legend

5 Appendices

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

Iteration	Convergence	X Value
1	1.3931	3.25
2	2.3274	2.875
3	0.8428	3.0625
4	0.1718	3.1562
5	0.3602	3.1094
6	0.1005	3.1328
7	0.034	3.1445
8	0.0336	3.1387
9	1.03E-04	3.1416
10	0.0168	3.1401
11	0.0083	3.1409
12	0.0041	3.1412
13	0.002	3.1414
14	9.54E-04	3.1415
15	4.26E-04	3.1416
16	1.61E-04	3.1416
17	2.93E-05	3.1416
18	3.68E-05	3.1416
19	3.76E-06	3.1416
20	1.28E-05	3.1416
21	4.50E-06	3.1416
22	3.67E-07	3.1416

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

Iteration	Convergence	Value
1	2.0804	4.175
2	0.2195	4.2625
3	0.9881	4.3062
4	0.3553	4.2844
5	0.0612	4.2734
6	0.0808	4.268
7	0.0102	4.2707
8	0.0254	4.2721
9	7.60E-03	4.2714
10	0.0013	4.271
11	0.0031	4.2712
12	9.11E-04	4.2711
13	2.00E-04	4.2711
14	3.56E-04	4.2711

15	7.77E-05	4.2711
16	6.11E-05	4.2711
17	8.31E-06	4.2711
18	2.64E-05	4.2711
19	9.05E-06	4.2711
20	3.66E-07	4.2711

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

Iteration	Convergence	Value
1	46.1643	4.0878
2	16.0667	3.241
3	5.4488	2.4874
4	1.7969	1.8495
5	0.5771	1.3372
6	0.1812	0.9449
7	0.0559	0.6562
8	0.017	0.4499
9	5.10E-03	0.3057
10	0.0015	0.2065
11	4.62E-04	0.1388
12	1.38E-04	0.0931
13	4.09E-05	0.0623
14	1.22E-05	0.0416
15	3.61E-06	0.0278
16	1.07E-06	0.0186
17	3.18E-07	0.0124

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

Iteration	Convergence	X Value
1	0.0575	1.1111
2	2.08E-04	1.1183
3	2.75E-09	1.1183

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

Root 1			Root 2		
Iteration	Convergence	X Value	Iteration	Convergence	X Value
1	0.0183	4.197	1	0.0973	5.9545
2	4.99E-02	4.1894	2	0.0399	5.9621
3	1.57E-02	4.1932	3	0.0112	5.9659
4	0.0013	4.1951	4	0.0032	5.9678

5	0.0072	4.1941	5	0.004	5.9669
6	0.003	4.1946	6	3.73E-04	5.9673
7	8.33E-04	4.1948	7	0.0014	5.9676
8	2.32E-04	4.195	8	5.25E-04	5.9674
9	3.00E-04	4.1949	9	7.57E-05	5.9674
10	3.40E-05	4.1949	10	1.49E-04	5.9674
11	9.91E-05	4.1949	11	3.66E-05	5.9674
12	3.26E-05	4.1949	12	1.96E-05	5.9674
13	7.16E-07	4.1949	13	8.51E-06	5.9674
			14	5.52E-06	5.9674
			15	1.50E-06	5.9674
			16	2.01E-06	5.9674
			17	2.58E-07	5.9674

Root 3			Root 4		
Iteration	Convergence	X Value	Iteration	Convergence	X Value
1	0.0859	8.0152	1	0.1144	9.9848
2	0.0262	8.0076	2	0.0549	9.9924
3	0.0037	8.0038	3	0.0252	9.9962
4	0.0113	8.0057	4	0.0103	9.9981
5	0.0038	8.0047	5	0.0029	9.9991
6	6.48E-05	8.0043	6	8.23E-04	9.9995
7	0.0018	8.004	7	0.001	9.9993
8	8.69E-04	8.0041	8	1.06E-04	9.9994
9	4.02E-04	8.0042	9	3.59E-04	9.9995
10	1.69E-04	8.0042	10	1.27E-04	9.9994
11	5.19E-05	8.0042	11	1.04E-05	9.9994
12	6.46E-06	8.0043	12	4.77E-05	9.9994
13	2.27E-05	8.0043	13	1.86E-05	9.9994
14	8.13E-06	8.0043	14	4.12E-06	9.9994
15	8.37E-07	8.0043	15	3.14E-06	9.9994
			16	4.90E-07	9.9994

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

root 1			root 2		
Iteration	Convergence	Value	Iteration	Convergence	Value
1	8.09E-04	4.1948	1	2.30E-06	5.9674
2	2.11E-08	4.1949	2	3.20E-14	5.9674

root 3			root 4		
Iteration	Convergence	Value	Iteration	Convergence	Value

1	1.53E-04	8.0042	1	1.77E-04	9.9994
2	2.19E-11	8.0043	2	3.92E-12	9.9994

5.7 Your code and functions

```
f='Bis_Meth2_1';
x1 = 1;
x3 = 4;
x2 = (x3 - x1)/2 + x1;

y1 = Bis_Meth2_1(x1);
y2 = Bis_Meth2_1(x2);
y3 = Bis_Meth2_1(x3);

figure;
p1= plot(x, sin(x).*sinh(x),'k');
hold on
p2= plot([x(1) x(end)],[0 0],'k--');
p3 = plot([x1 x1], [0,y1],'b--');
p4 = plot([x3 x3], [0,y3],'b--');
p5 = plot([x2 x2], [0,y2],'b --');

Bisect_Table=table([], [], [], 'VariableNames', {'Iteration','Convergence','Value'});

while abs(y2)>convergence && iteration<50
    % Update x1, x2, x3 each iteration
    [y1, y2, y3, x1, x2, x3] = Bis_Meth_Main(f, R, x1, x2, x3, y1,y2,y3);

    % Difference for next iteration
    difference = abs(feval(f, x2));

    p3 = plot([x1 x1], [0,y1],'b--');
    p4 = plot([x3 x3], [0,y3],'b--');
    p5 = plot([x2 x2], [0,y2],'b --');

    Bisect_Table.Iteration(iteration)=iteration;
    Bisect_Table.Convergence(iteration)=difference;
    Bisect_Table.Value(iteration)=x2;

    iteration = iteration+1;
end

p6 = plot(x2,y2, 'ro', 'MarkerSize', 10,'LineWidth',2);

fprintf('\nConverged root = %.8f\n', x2);
fprintf('f(root) = %.8e\n', y2);
fprintf('iteration number = %.8e\n', iteration);

xlabel('$x$', 'Interpreter', 'latex', FontSize=20);
ylabel('$sin(x) sinh(x)$', 'Interpreter', 'latex', FontSize=20);
title('Bisection method for f(x)= sin(x)sinh(x)', 'Interpreter', 'latex',FontSize=30);
subtitle(['interp at ', num2str(x2),' on the x axis gives a root value = ', num2str(y2)], FontSize=15)
grid on;
ylim([-25 5])
xlim([0.5 4.5])
legend([p1,p2,p3,p6],'sin(x)sinh(x);','The root line', 'Mid and end points','intersection point')

else
```

Figure 3 - Bisection 1 method main script code (sinxsinh(x))

```

f='Bis_Meth2_2';
x1 = 4;
x3 = 4.7;
x2 = (x3 - x1)/2 + x1;

y1 = Bis_Meth2_2(x1, R);
y2 = Bis_Meth2_2(x2, R);
y3 = Bis_Meth2_2(x3, R);
|
figure;

% sqrt((R.^2)-(x.^2)) - (x.*tan(x)) is the same as (R^2-x^2)^(1/2)-xtan(x)
p1 = plot(x, sqrt((R.^2)-(x.^2)) - (x.*tan(x)), 'k');
hold on
p2 = plot([x(1) x(end)], [0 0], 'k--');
p3 = plot([x1 x1], [0,y1], 'b--');
p4 = plot([x3 x3], [0,y3], 'b--');
p5 = plot([x2 x2], [0,y2], 'b --');

Bisection2_Table=table([], [], [], 'VariableNames', {'Iteration','Convergence','Value'});

while abs(y2)>convergence && iteration<50
    % Update x1, x2, x3 each iteration
    [y1, y2, y3, x1, x2, x3] = Bis_Meth_Main(f, R, x1, x2, x3, y1,y2,y3);

    % Difference for next iteration
    difference = abs(feval(f, x2, R));

    p3 = plot([x1 x1], [0,y1], 'b--');
    p4 = plot([x3 x3], [0,y3], 'b--');
    p5 = plot([x2 x2], [0,y2], 'b --');

    Bisection2_Table.Iteration(iteration)=iteration;
    Bisection2_Table.Convergence(iteration)=difference;
    Bisection2_Table.Value(iteration)=x2;

    iteration = iteration+1;

end

p6 = plot(x2,y2, 'ro', 'MarkerSize', 10,'LineWidth',2);

fprintf('\nConverged root = %.8f\n', x2);
fprintf('f(root) = %.8e\n', y2);
fprintf('iteration number = %.8e\n', iteration);

xlabel('$x$', 'Interpreter', 'latex',FontSize=20);
ylabel('$ (R^2 - x^2)^{0.5} - x\tan(x) $', 'Interpreter', 'latex',FontSize=20);
title('Bisection method for $(R^2 - x^2)^{0.5} - x\tan(x)$', ...
      'Interpreter','latex', 'FontSize', 30);
subtitle(['interp at ', num2str(x2), ' on the x axis gives a root value = ', num2str(y2)], FontSize=15)
grid on;
ylim([-400 10])
xlim([3.75 5])
legend([p1,p2,p3,p6], 'sqrt((R^2)-(x^2)) - (xtan(x));', 'The root line', 'Mid and end points','intersection point')

    end
end

```

Figure 4 - Bisection method2 main script code ($R^2 - x^2(1/2) - x \tan(x)$)

```

function [y1,y2,y3,x1,x2,x3] = Bis_Meth_Main(f,R,x1,x2,x3,y1,y2,y3)

fhandle = str2func(f);
nArgs = nargin(fhandle); % 1 or 2

if y1*y3>0 % if there is not a root between y1 and y3
    error(' Need new guesses')

elseif y1*y2<0 % if said root is between y1 and only y2 is positive
    % Root is in left half, so
    d=(x2-x1)/2;
    x1=x1;
    x3=x2;
    x2=x3-d;

else
    % Root is in right half, so
    d=(x3-x2)/2;
    x1=x2;
    x2=x1+d;
    x3=x3;

end

if nArgs == 1
    y1 = fhandle(x1);
    y2 = fhandle(x2);
    y3 = fhandle(x3);
else
    y1 = fhandle(x1, R);
    y2 = fhandle(x2, R);
    y3 = fhandle(x3, R);
end
end

```

Figure 5 - Bisection method function used in 1.1 and 1.2

```

% exp(x) - (1 + x + (x.^2)/2);

figure;
hold on

for i = 1:length(x1_Range)

    f = @Newton_f1;
    a = plot(x, f(x), 'b');
    df = @Newton_df1;
    x1 = x1_Range(1);

c = yline(0, 'k--');
iteration = 0;
difference = 1;

Newton_B_Table=table([], [], [], 'VariableNames', {'Iteration','Convergence','Value'});

    while difference > tolerance && iteration < 20

        [y1, y2, x2] = Newton_Meth_1(f, x1, df);

        difference = abs(y2);

        iteration = iteration + 1;
        d = plot(x1, y1, 'ro');
        e = plot(x2, y2, 'ro');
        g = plot([x1, x2], [y1, 0], 'r--');

        Newton_B_Table.Iteration(iteration)=iteration;
        Newton_B_Table.Convergence(iteration)=difference;
        Newton_B_Table.Value(iteration)=x2;

        x1 = x2;
    end

    y2_stored(i)=y2;
    fprintf('\nNumber of iterations = %.5e\n', iteration);
    fprintf('\ntolerance = %.5e\n', tolerance);
    fprintf('\ny value at x = %.5e\n', y2);
    fprintf('\nroot value = %.5e\n', x2);
end

xlim([0 5])
ylim([-5 130])

title('Newton's Raphson Method for  $e^x - \left(1 + x + \frac{x^2}{2}\right)$ ', ...
      'Interpreter','latex', 'FontSize', 30)

subtitle(['interp at ', num2str(x2), ' on the x axis gives a root value = ', num2str(y2)], FontSize=15)

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

ylabel('$e^x - \left(1 + x + \frac{x^2}{2}\right)$', ...
      'Interpreter','latex', FontSize=15);

```

```

inset_pos = [0.175 0.3 0.3 0.3]; % [x y width height] in normalized units
inset_axes = axes('Position', inset_pos);
hold(inset_axes,'on')

plot(inset_axes, x, f(x), 'b');
yline(inset_axes, 0, 'k--');

% repeat Newton plotting only inside inset
for i = 1:length(x1_Range)

    x1 = x1_Range(i);
    iteration = 0;
    difference = 1;

    while difference > tolerance && iteration < 20

        [y1, y2, x2] = Newton_Meth_1(f, x1, df);

        difference = abs(y2);
        iteration = iteration + 1;

        plot(inset_axes,x1, y1, 'ro');
        plot(inset_axes,x2, y2, 'ro');
        plot(inset_axes,[x1, x2], [y1, 0], 'r--');

        x1 = x2;
    end
end

% zoomed limits
xlim(inset_axes, [0 0.2])
ylim(inset_axes, [-0.00005 0.002])

title(inset_axes, 'Zoomed view at root','FontSize',12)

legend([a,c,d,g], ...
    '$e^x - \left(1 + x + \frac{x^2}{2}\right)$', ...
    '$0 \text{ \text{line}}$', ...
    '$\text{iteration markers}$', ...
    '$\text{tangent lines}$', ...
    'Interpreter','latex', ...
    'Location','best',FontSize=15)

```

Figure 6 - Newton method main script code ($\exp(x) - (1 + x + (x^2)/2)$)

```

% 10log(x)-x;

figure;
hold on

for i = 1:length(x1_Range)

    f = @Newton_f2;
    a = plot(x, f(x), 'b');
    df = @Newton_df2;
    x1 = x1_Range(2);

c = yline(0, 'k--');
iteration = 0;
difference = 1;

Newton_C_Table=table([], [], [], 'VariableNames', {'Iteration', 'Convergence', 'Value'});

while difference > tolerance && iteration < 20

    [y1, y2, x2] = Newton_Meth_1(f, x1, df);

    difference = abs(y2);
    iteration = iteration + 1;
    d = plot(x1, y1, 'ro');
    e = plot(x2, y2, 'ro');
    g = plot([x1, x2], [y1, 0], 'r--');

    Newton_C_Table.Iteration(iteration)=iteration;
    Newton_C_Table.Convergence(iteration)=difference;
    Newton_C_Table.Value(iteration)=x2;

    x1 = x2;
end
y2_stored(i)=y2;
fprintf('\nNumber of iterations = %.5e\n', iteration);
fprintf('\ntolerance = %.5e\n', tolerance);
fprintf('\ny value at x = %.5e\n', y2);
fprintf('\nroot value = %.5e\n', x2);
end

xlim([0 5])
ylim([-10 20])

title('Newton''s Raphson Method for  $10\log(x) - x$ ', ...
      'Interpreter', 'latex', 'FontSize', 30)

subtitle(['interp at ', num2str(x2), ' on the x axis gives a root value = ', num2str(y2)], FontSize=15)

xlabel('$x$', 'Interpreter', 'latex', FontSize=20)
ylabel('$10\log(x) - x$', ...
      'Interpreter', 'latex', FontSize=15);

inset_pos = [0.55 0.2 0.3 0.3]; % [x y width height] in normalized units
inset_axes = axes('Position', inset_pos);
hold(inset_axes, 'on')

```

```

plot(inset_axes, x, f(x), 'b');
yline(inset_axes, 0, 'k--');

% repeat Newton plotting only inside inset
for i = 1:length(x1_Range)

    x1 = x1_Range(i);
    iteration = 0;
    difference = 1;

    while difference > tolerance && iteration < 20

        [y1, y2, x2] = Newton_Meth_1(f, x1, df);

        difference = abs(y2);
        iteration = iteration + 1;

        plot(inset_axes,x1, y1, 'ro');
        plot(inset_axes,x2, y2, 'ro');
        plot(inset_axes,[x1, x2], [y1, 0], 'r--');

        x1 = x2;
    end
end

% zoomed limits
xlim(inset_axes, [0.95 1.15])
ylim(inset_axes, [-1.5 0.25])

title(inset_axes, 'Zoomed view at root','FontSize',12)

legend([a,c,d,g], ...
    '$10\log(x) - x$', ...
    '$0\ \text{line}$', ...
    '$\text{iteration markers}$', ...
    '$\text{tangent lines}$', ...
    'Interpreter','latex', ...
    'Location','northwest',FontSize=15)

```

Figure 7 - Newton method in main script ($10\log x - x$)

```

function [y1,y2,x2] = Newton_Meht_1(f,x1,df)

y1=f(x1);
dy1=df(x1);

x2=x1-(y1/dy1);
y2=f(x2);

end

```

Figure 8 - Newton method function used in 2.1 and 2.2

```

% Intersection of two functions

x=linspace(3,10,500);

x1=4;
tolerance = 1e-6;

figure;
hold on

f = @Newton_f_2_1;
a = plot(x, f(x), 'b');
df = @Newton_df_2_1;
b = yline(0,'k--');
    iteration = 0;
    difference = 1;

    while difference > tolerance && iteration < 20

        [y1, y2, x2] = Newton_Meth_1(f, x1, df);

        difference = abs(y2);
        iteration = iteration + 1;
        c = plot(x1, y1, 'ro');
        d = plot(x2, y2, 'ro');
        e = plot([x1, x2], [y1, 0], 'r--');

        x1 = x2;
    end
    fprintf('\nNumber of iterations = %.5e\n', iteration);
    fprintf('\ntolerance = %.5e\n', tolerance);
    fprintf('\nroot value = %.5e\n', y2);

%tolerance change to 1e-10
tolerance =1e-10;
    while difference > tolerance && iteration < 20

        [y1, y2, x2] = Newton_Meth_1(f, x1, df);

        difference = abs(y2);
        iteration = iteration + 1;
        g = plot(x1, y1, 'go');
        h = plot(x2, y2, 'go');
        j = plot([x1, x2], [y1, 0], 'g--');

        x1 = x2;
    end
    fprintf('\nNumber of iterations = %.5e\n', iteration);
    fprintf('\ntolerance = %.5e\n', tolerance);
    fprintf('\nroot value = %.5e\n', y2);

title('Newton's Raphson Method - Intersection of Two Functions','Interpreter','latex','FontSize',30)

subtitle(sprintf('Using formula  $10e^{-x} - 5\sin\left(\frac{\pi x}{2}\right)$  with a root value of %.5e', ...
    y2),'Interpreter','latex', FontSize=15);

xlabel('$x$','Interpreter','latex',FontSize=20)
ylabel('$10 e^{-x} - 5 \sin\left(\frac{\pi x}{2}\right)$', ...
    'Interpreter','latex',FontSize=20);

```

```

grid on;
ylim([-5 10])
xlim([3 10])
legend([p1,p2,p3,p6], 'sqrt((R^2)-(x^2)) - (xtan(x));', 'The root line', 'Mid and end points', 'intersection point')

inset_pos = [0.65 0.6 0.25 0.25]; % [x y width height]
inset = axes('Position', inset_pos);
hold(inset, 'on')

% main function inside inset
f = @Newton_f_2_1;
plot(inset, x, f(x), 'b');
yline(inset, 0, 'k--');

% re-plot Newton steps inside inset
x1 = 4;
difference = 1;
iteration = 0;
tolerance = 1e-6;

while difference > tolerance && iteration < 20
    [y1, y2, x2] = Newton_Meth_1(f, x1, df);

    difference = abs(y2);
    iteration = iteration + 1;
    plot(inset, x1, y1, 'ro');
    plot(inset, x2, y2, 'ro');
    plot(inset, [x1 x2], [y1 0], 'r--');

    x1 = x2;
end

%tolerance change to 1e-10
tolerance = 1e-10;
while difference > tolerance && iteration < 20

    [y1, y2, x2] = Newton_Meth_1(f, x1, df);

    difference = abs(y2);
    iteration = iteration + 1;
    g = plot(x1, y1, 'go');
    h = plot(x2, y2, 'go');
    j = plot([x1, x2], [y1, 0], 'g--');

    x1 = x2;
end

% inset zoom limits
xlim(inset, [3.9 4.2]);
ylim(inset, [-0.25 3]);
title(inset, 'Zoomed view at root', 'FontSize', 12);

legend([a,b,c,e,g,h,j], ...
    '$10 e^{-x} - 5 \sin(\frac{\pi x}{2})$', ...
    '$0 \text{ \text{[line]}}$', ...
    '$\text{iteration markers for 1e6 convergence}$$', ...
    '$\text{tangent lines for 1e6 convergence}$$', ...
    '$\text{iteration markers for 1e10 convergence}$$', ...
    '$\text{tangent lines for 1e10 convergence}$$', ...
    'Interpreter', 'latex', ...
    'Location', 'south')

```

Figure 9 - Newton method Intersecting two functions main script code ($10e^{-x} - 5\sin(\pi x/2)$)

```

x = linspace(4,10,100);
f = @Newton_f_2_1;
Found_Roots=[];

figure;
a=plot(x, f(x),'k');
hold on
b=plot([x(1) x(end)], [0 0], 'k--');

Bisect3_All = {};

for i = 1:length(x)-1
x1 = x(i);
x3 = x(i+1);
x2 = (x3 - x1)/2 + x1;

y1 = f(x1);
y2 = f(x2);
y3 = f(x3);

convergence = 1e-6;
difference = inf;
iteration=0;
Root_iteration=0;

    if y1*y3>0 % if there is not a root between y1 and y3
        % fprintf('skipped values at iteration = %.e\n', iteration)
        Found_Roots(i)=NaN;
        c=plot(x2,y2,'bo');
    else

Bisect3_Table=table([], [], [], 'VariableNames', {'Iteration','Convergence','Value'});

while abs(f(x2)) > convergence
    % Update x1, x2, x3 each iteration
    [x1, x2, x3, y1, y2, y3] = Bis_Meth_Bracket(f, x1, x2, x3, y1, y2, y3);

    % Difference for next iteration
    difference = abs(f(x2));
    iteration=iteration+1;

    Bisect3_Table.Iteration(iteration)=iteration;
    Bisect3_Table.Convergence(iteration)=difference;
    Bisect3_Table.Value(iteration)=x2;

end

    Bisect3_All{end+1} = Bisect3_Table;

    d=plot(x2, y2,'ro');

    % iteration=iteration+1;

fprintf('\nConverged root = %.8f\n', x2);
fprintf('f(root) = %.8e\n', f(x2));
fprintf('iteration = %.8e\n', iteration);

end

end

% plotting the two parts of the function as individual lines
e=plot(x,100*exp(-x));
g=plot(x,(5*sin((pi*x)/2)));

xlabel('$x$', 'Interpreter', 'latex',FontSize=20);
ylabel('$100 e^{-x} - 5 \sin(\frac{\pi x}{2})$', 'Interpreter','latex',FontSize=20)
title('Bisection method for multiple roots using equation $f(x) = 100 e^{-x} - 5 \sin(\frac{\pi x}{2})$', 'Interpreter','latex','FontSize',30);
grid on;
xlim([4 10])
legend([a,b,c,d,e,g], '$100 e^{-x} - 5 \sin(\frac{\pi x}{2})$', ...
    'The root line', ...
    'Points where it does not cross 0', ...
    'Points where it does cross 0', '$100 e^{-x}$', ...
    '$5 \sin(\frac{\pi x}{2})$', ...
    'Interpreter','latex','FontSize',15)

```

Figure 10 – Bracketing method main script code ($100e^{-x}-5\sin(\pi x/2)$)

```

% Newton bracketing method (Multiple roots)

x = linspace(4,10,100);
figure;
hold on

f = @Newton_f_2_1;
df = @Newton_df_2_1;

% Plot the function and the zero line
a = plot(x, f(x), 'k');
b = yline(0, 'k--');

Found_Roots = nan(size(x));
NewtonD_All = {};

for i = 1:length(x)-1

    x1 = (x(i) + x(i+1)) / 2; % midpoint initial guess

    iteration = 0;
    tolerance = 1e-6;
    difference = inf;

    NewtonD_Table=table([], [], [], 'VariableNames', {'Iteration','Convergence','Value'});

    % Calculates the root value at all 100 points
    while difference > tolerance && iteration < 20
        [y1, y2, x2] = Newton_Meth_1(f, x1, df);
        difference = abs(y2);

        iteration = iteration + 1;

        NewtonD_Table.Iteration(iteration)=iteration;
        NewtonD_Table.Convergence(iteration)=difference;
        NewtonD_Table.Value(iteration)=x2;

        x1 = x2;
    end

    % Points at where the x value crosses the root
    if difference < tolerance && x2 >= x(i) && x2 <= x(i+1)

        c=plot(x2, 0, 'ro');

        fprintf('\nConverged root = %.8f\n', x2);
        fprintf('f(root) = %.8e\n', f(x2));
        fprintf('iterations = %d\n', iteration);
        fprintf('interval index = %d\n', i+1);

        NewtonD_All{end+1} = NewtonD_Table;
    else
        d=plot(x, f(x), 'bo');
    end
end

xlim([x(1) x(end)])
title('Newton''s Method for multiple roots using equation $f(x) = 100 e^{-x} - 5 \sin(\frac{\pi x}{2})$', 'Interpreter', 'latex', 'FontSize', 30)
xlabel('$x$', 'Interpreter', 'latex', 'FontSize=20)
ylabel('$100 e^{-x} - 5 \sin(\frac{\pi x}{2})$', 'Interpreter', 'latex', 'FontSize=20)
grid on
e=plot(x, (100*exp(-x)));
g=plot(x, (5*sin((pi*x)/2)));
legend([a,b,c,d,e,g], '$100 e^{-x} - 5 \sin(\frac{\pi x}{2})$', 'The root line', ...
    'Points where it does cross 0', ...
    'Points where it does not cross 0', ...
    '$100 e^{-x}$', ...
    '$5 \sin(\frac{\pi x}{2})$', ...
    'Interpreter', 'latex', 'FontSize', 15);

disp(NewtonD_All{4})

```

Figure 11 - Bracketing method main script code ($1000e^{-x}-5\sin(\pi x/2)$)

```
function [x1, x2, x3, y1, y2, y3] = Bis_Meth_Bracket(f, x1, x2, x3, y1, y2, y3)

if y1*y3 > 0
    % Should never happen now - handled in main script
    return
elseif y1*y2 < 0
    % Root in left half
    x3 = x2;
else
    % Root in right half
    x1 = x2;
end

x2 = (x1 + x3)/2;

y1 = f(x1);
y2 = f(x2);
y3 = f(x3);

end
```

Figure 12 - Bracketing method function used in 4.1 and 4.2