

Task 2

Table of Contents

1 Bisection $f(x) = \sin(x)\sinh(x)$; Limits = [1 4] [3].....	1
1 Bisection $f(x) = (R^2 - x^2)^{1/2} - x \tan(x)$ [3].....	2
2 Newtons $f(x) = e^x - (1 + x + (x^2/2))$ [3].....	2
2 Newtons $f(x) = 10\log(x) - x$ [3].....	3
3 Newtons Intersect [6].....	3
4 Bracketing Bisection [6].....	5
4 Bracketing Newton [6].....	5

1 Bisection $f(x) = \sin(x)\sinh(x)$; Limits = [1 4] [3]

```
%% 1. Bisection Method
% (a) Implement the bisection method function
function root = bisect(f, a, b, imax, tol)
    x1 = a; f1 = feval(f, x1);
    x3 = b; f3 = feval(f, x3);

    if f1 * f3 > 0
        error('No root in the specified interval. Check your function or interval.');
```

```
    end

    for ndx = 1:imax
        x2 = (x1 + x3) / 2;
        f2 = feval(f, x2);

        if abs(f2) < tol || (x3 - x1) / 2 < tol
            root = x2;
            return;
        end

        if f1 * f2 < 0
            x3 = x2;
        else
            x1 = x2;
            f1 = f2;
        end
    end

    error('Exceeded maximum iterations');
```

```
end

% (b) Define the functions
f1 = @(x) sin(x) .* sinh(x);
f2 = @(x) 10 * log(x) - x;
```

1 Bisection $f(x) = (R^2 - x^2)^{1/2} - x \tan(x)$ [3]

```
% (c) Call bisection function for (b)
a = 1; b = 4; tol = 1e-6; imax = 50;
try
    root1 = bisect(f1, a, b, imax, tol);
catch ME
    error('Root not found in the interval [1,4] for f1: %s', ME.message);
end

% (d) Call bisection function for (c) in the verified interval [1.0, 1.5]
a2_verified = 1.0; b2_verified = 1.5;
try
    root2 = bisect(f2, a2_verified, b2_verified, imax, tol);
catch ME
    error('Root not found in the verified interval [1.0,1.5] for f2: %s',
ME.message);
end

% (e) Plot results for both functions
% Plot for (b)
x = linspace(a, b, 100);
y = f1(x);
figure; plot(x, y, '-', root1, f1(root1), 'o');
title(['Function 1: Root at x = ', num2str(root1)]);
xlabel('x'); ylabel('f(x)');

% Plot for (c)
x2 = linspace(a2_verified, b2_verified, 100);
y2 = f2(x2);
figure; plot(x2, y2, '-', root2, f2(root2), 'o');
title(['Function 2: Root at x = ', num2str(root2)]);
xlabel('x'); ylabel('f(x)');
```

2 Newtons $f(x) = e^x - (1 + x + (x^2/2))$ [3]

```
%% 2. Newton-Raphson Method
% (a) Implement Newton-Raphson function
function root = newton(x0, f, df, imax, tol)
    x = x0;

    for ndx = 1:imax
        fx = feval(f, x);
        dfx = feval(df, x);

        if abs(dfx) < eps
            error('Derivative near zero, stopping calculation');
        end
    end
end
```

```

        dx = -fx / dfx;
        x = x + dx;

        if abs(dx) < tol
            root = x;
            return;
        end
    end

    error('Exceeded maximum iterations');
end

% (b) Define the functions and derivatives
f3 = @(x) sin(x) .* sinh(x);
df3 = @(x) cos(x) .* sinh(x) + sin(x) .* cosh(x);

f4 = @(x) 10 * log(x) - x;
df4 = @(x) 10 ./ x - 1;

```

2 Newtons $f(x) = 10\log(x)-x$ [3]

```

% (c) Call Newton function for (b)
x0 = 5; tol = 1e-6; imax = 20;
root3 = newton(x0, f3, df3, imax, tol);

% (d) Call Newton function for (c)
x02 = 1;
root4 = newton(x02, f4, df4, imax, tol);

% (e) Plot results for both functions
% Plot for (b)
x3 = linspace(1, 6, 100);
y3 = f3(x3);
figure; plot(x3, y3, '-', root3, f3(root3), 'o');
title(['Function 3: Root at x = ', num2str(root3)]);
xlabel('x'); ylabel('f(x)');

% Plot for (c)
x4 = linspace(0.5, 2, 100);
y4 = f4(x4);
figure; plot(x4, y4, '-', root4, f4(root4), 'o');
title(['Function 4: Root at x = ', num2str(root4)]);
xlabel('x'); ylabel('f(x)');

```

3 Newtons Intersect [6]

```

%% 3.Newton's Method – Intersection of Two Functions

% (a) Create the Newton's Method function

```

```

function root = Newton_method(f, df, x0, tol, imax)
    % Inputs:
    % f - function handle (the function)
    % df - function handle (derivative of the function)
    % x0 - initial guess
    % tol - tolerance for convergence
    % imax - maximum iterations

    x = x0; % Initialize the guess
    for i = 1:imax
        fx = f(x);
        dfx = df(x);
        if abs(dfx) < 1e-12
            error('Derivative near zero. Method fails.');
        end
        x_new = x - fx / dfx;
        if abs(x_new - x) < tol
            root = x_new;
            fprintf('Converged to root at x = %.10f after %d iterations\n', root,
i);
            return;
        end
        x = x_new;
    end
    error('Maximum iterations reached without convergence.');
```

end

% (b) Define functions and their derivatives

```

function y = fx(x)
    y = x.^2; % f(x) = x^2
end

function y = gx(x)
    y = 10*x - 10; % g(x) = 10x - 10
end

function y = difference(x)
    y = fx(x) - gx(x); % h(x) = f(x) - g(x)
end

function y = difference_derivative(x)
    y = 2*x - 10; % h'(x) = 2x - 10
end

% (c) Solve for the intersection point and plot the result
function newtons_intersection()
    % Parameters
    x0 = 4; % Initial guess
    tol = 1e-10; % Tolerance
    imax = 20; % Maximum iterations
```

```

% Solve using Newton's method
root = newton_method(@difference, @difference_derivative, x0, tol, imax);

% Plot the functions
x_vals = linspace(3, 10, 100); % Range for plotting
fx_vals = fx(x_vals);
gx_vals = gx(x_vals);

% Plot
figure;
plot(x_vals, fx_vals, '-b', 'LineWidth', 1.5); hold on;
plot(x_vals, gx_vals, '-r', 'LineWidth', 1.5);
plot(root, fx(root), 'ko', 'MarkerSize', 8, 'LineWidth', 1.5); % Mark root
title(['Root at x = ', num2str(root, '%.10f')]);
xlabel('x');
ylabel('f(x), g(x)');
legend('f(x) = x^2', 'g(x) = 10x - 10', 'Intersection');
grid on;
end

```

4 Bracketing Bisect [6]

```

%% 4. Multiple Roots Using Bracketing
function Bracketing_Multiple_Roots_Fixed()
    % (a) Bracketing routine
    x = linspace(4, 10, 100);
    f = @(x) sin(x).*sinh(x); % Define the function
    y = f(x);

    % Approximate root intervals
    root_intervals = zeros(0, 2); % Predefine empty array for intervals
    index = 0;
    for i = 1:length(x)-1
        if y(i) * y(i+1) < 0
            index = index + 1;
            root_intervals(index, :) = [x(i), x(i+1)];
        end
    end

    % Display intervals
    disp('Approximate root intervals:');
    disp(root_intervals);
end

```

4 Bracketing Newton [6]

```

% (b) Root finding using Bisection and Newton-Raphson methods

```

```

num_roots = size(root_intervals, 1);
bisection_roots = zeros(num_roots, 1); % Preallocate for Bisection roots
newton_roots = zeros(num_roots, 1); % Preallocate for Newton roots
tol = 1e-6;
imax = 50;
for i = 1:num_roots
    a = root_intervals(i, 1);
    b = root_intervals(i, 2);

    % Bisection method
    bisection_roots(i) = bisect(f, a, b, imax, tol);

    % Newton-Raphson method
    x0 = (a + b) / 2;
    df = @(x) cos(x).*sinh(x) + sin(x).*cosh(x); % Derivative of the function
    newton_roots(i) = newton(x0, f, df, imax, tol);
end

% Display roots
disp('Roots found by Bisection:');
disp(bisection_roots);
disp('Roots found by Newton-Raphson:');
disp(newton_roots);

% (c) Plotting
figure;
plot(x, y, '-');
hold on;
for i = 1:num_roots
    plot(bisection_roots(i), f(bisection_roots(i)), 'ro'); % Mark roots
end
title('Roots of sin(x)*sinh(x) in [4, 10]');
xlabel('x');
ylabel('f(x)');
grid on;
hold off;
end

function root = Bisect(f, a, b, imax, tol)
    x1 = a; f1 = f(x1);
    x3 = b; f3 = f(x3);

    if f1 * f3 > 0
        error('No root in the specified interval');
    end

    for i = 1:imax
        x2 = (x1 + x3) / 2;
        f2 = f(x2);

```

```

    if abs(f2) < tol || (x3 - x1) / 2 < tol
        root = x2;
        return;
    end

    if f1 * f2 < 0
        x3 = x2;

    else
        x1 = x2;
        f1 = f2;
    end
end
error('Bisection method did not converge');
end

function root = Newton(x0, f, df, imax, tol)
    x = x0;
    for i = 1:imax
        fx = f(x);
        dfx = df(x);

        if abs(dfx) < eps
            error('Derivative near zero, Newton method fails');
        end

        dx = -fx / dfx;
        x = x + dx;

        if abs(dx) < tol
            root = x;
            return;
        end
    end
    error('Newton method did not converge');
end

```