Problem 1

Find the roots to the following functions $f(x)$.

(a) $f(x) = -x^2 + x + 2$,

(b) $f(x) = e^x - 2 - x$,

using:

1. **Fixed point iteration**. Choose your $g(x)$ and a tolerance of 10^{-9}. For (a) start with the following points $p_0 = \{2.5\,;\, 0.15\,;\, 1.5\}$. For (b) start with the following points $p_0 = \{2.5\,;\, 0.15\,;\, 0.25\}$. What can you say about the convergence? **Explain**.

   For the fixed point iteration, I chose my $g(x)$ to be...

$$g_a(x) = \pm\sqrt{x+2} \tag{1}$$

$$g_b(x) = e^x - 2 \tag{2}$$

   for (a) and (b), respectively. I chose $g_a(x) = \pm\sqrt{x+2}$ instead of...

$$g_a(x) = x^2 - 2 \tag{3}$$

   because $g_a(x) = x^2 - 2$ would not work for $p_0 = 2.5$. Plugging the $p_0 = 2.5$ into $g_a'(x) = 2x$ is 5, would break the rule of fixed point iterations, where...

$$|g'(x)| < k < 1 \;\; \forall \;\; x \in [a, b] \tag{4}$$

   In most cases, using the negative of (1) would result in imaginary numbers, but given that Matlab extends its functionality into complex numbers, the function $g_a(x) = -\sqrt{x+2}$ works in the complex number space and $g_a'(x)$ is never larger than 1 for the given p_0s.

   Another note is that since the $g_b(x) = e^x - 2$, a $p_0 = 2.5$ would not work because plugging $p_0 = 2.5$ into $g_b'(x) = e^x$ is $e^{2.5} > 1$ which breaks the rule of fixed point iterations.

   The graphs of the convergence are plotted below. The fixed point iteration of $g_a(x)$ converges on both the roots, $x = 2$ and $x = -1$, where as $g_b(x)$ only converges on one of the roots, $x = -1.84$

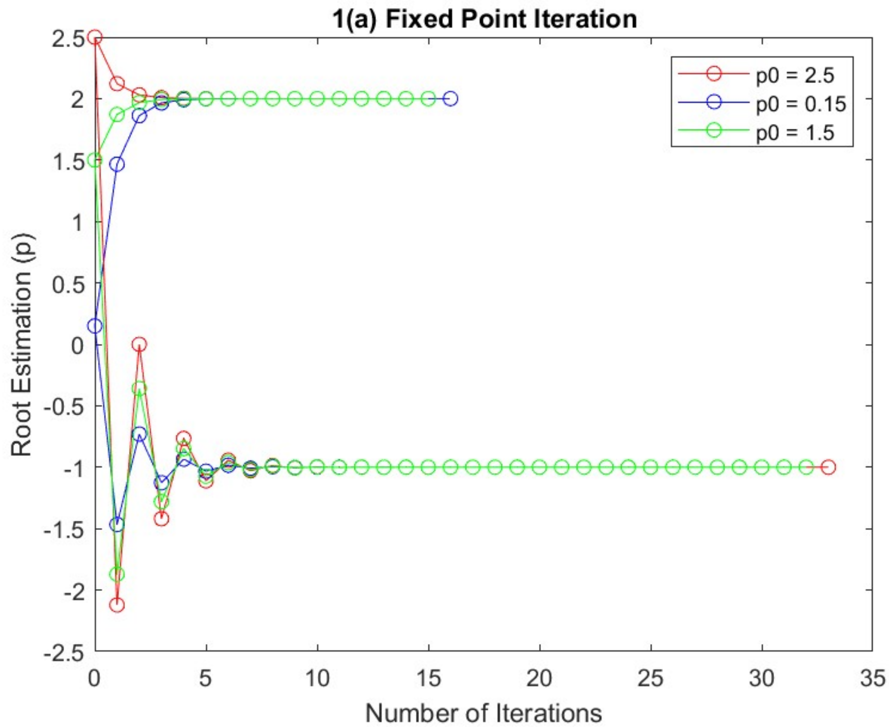

Figure 1: 1(a) Fixed Point Iteration

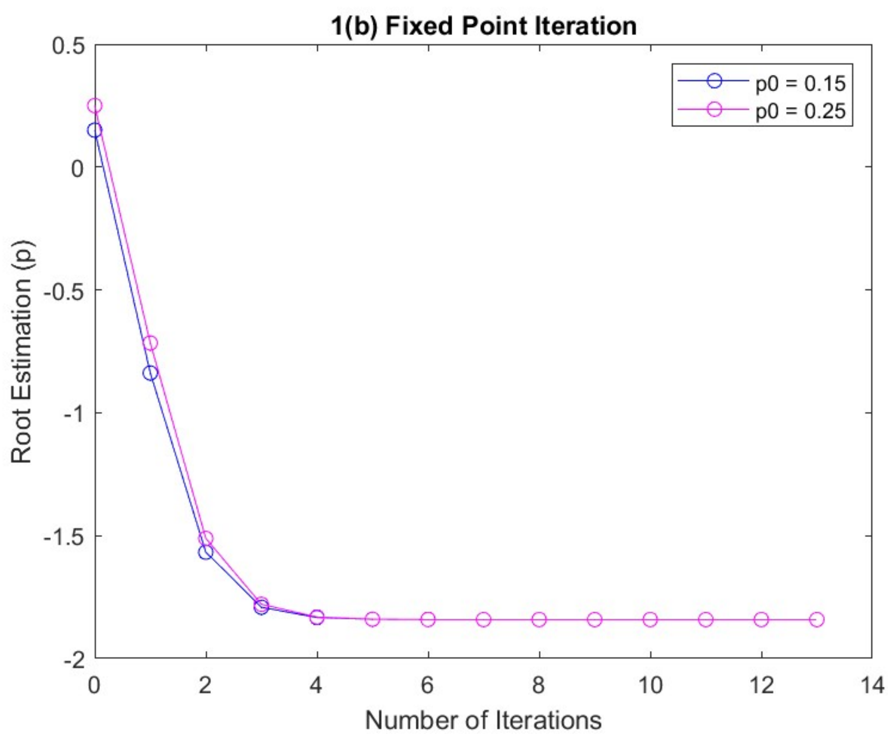

Figure 2: 1(b) Fixed Point Iteration

2. **Bisection method**. Choose a tolerance of 10^{-9}. For (a) and (b) use the intervals $x \in [-4.0\,;\,1.0]$ and $x \in [0.5\,;\,3.0]$. What can you say about the convergence? **Use plots and discuss**.

   For the functions...

$$f_a(x) = -x^2 + x + 2 \tag{5}$$

$$f_b(x) = e^x - x - 2 \tag{6}$$

   The roots are at $x = -1$, $x = 2$ and $x \approx -1.84$, $x \approx 1.15$ respectively. Since we use the intervals $x \in [-4.0\,;\,1.0]$ and $x \in [0.5\,;\,3.0]$, these intervals are perfectly placed such that they are between the two roots in each equation. Thus, they will converge on those roots as shown in figures 3 and 4.

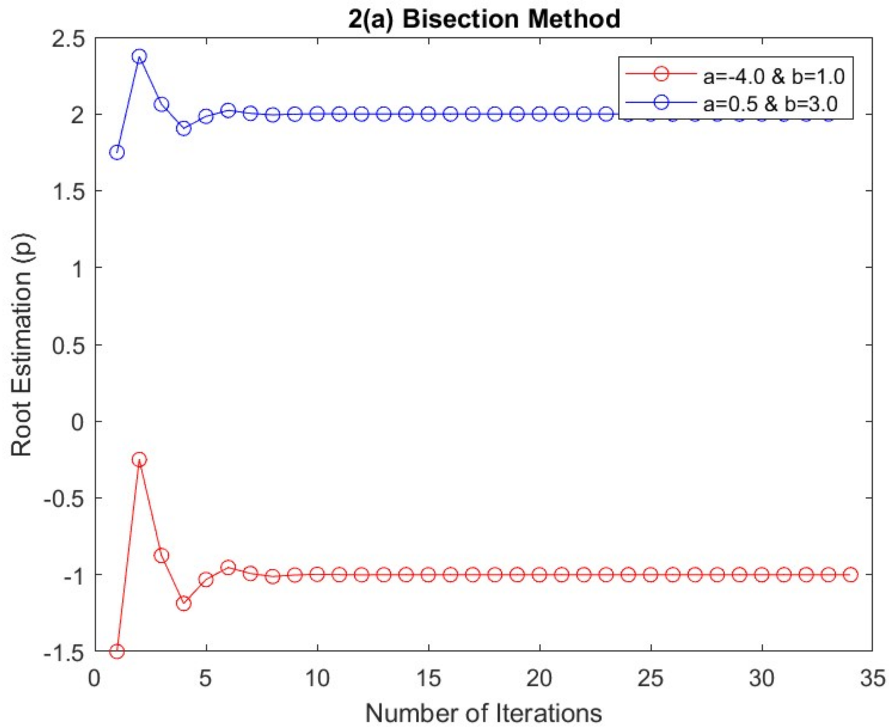

Figure 3: 2(a) Bisection Method

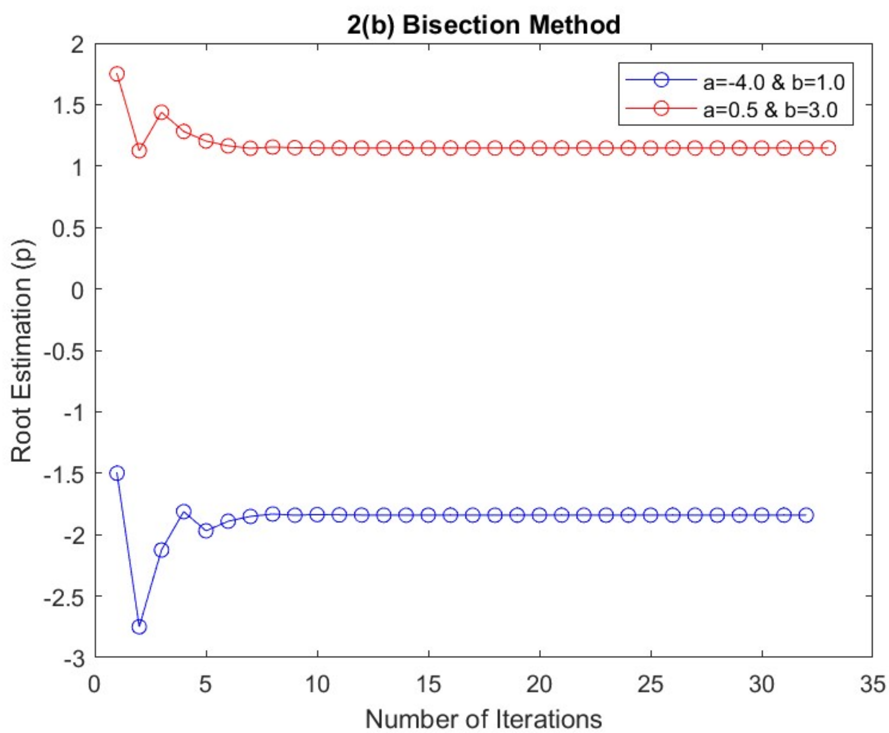

Figure 4: 2(b) Bisection Method

3. **Newton's method**. Choose $\delta = \epsilon = 10^{-9}$. For (a) and (b) start with the following points $p_0 = \{-3.0\,; 0.0\,; 6.0\}$. What can you say about the convergence? **Use plots and discuss**. For the functions...

$$f_a(x) = -x^2 + x + 2 \tag{7}$$

$$f_b(x) = e^x - x - 2 \tag{8}$$

The points $p_0 = \{-3.0\,; 0.0\,; 6.0\}$ all converge for $f_a(x)$ because none of the points are located at or very close to a local minimum or maximum where...

$$f_a'(x) = -2x + 1 = 0, \quad x = 0.5 \tag{9}$$

However, for $f_b(x)$, the point $p_0 = 0.0$ does not converge because $x = 0.0$ is located at a local minimum where...

$$f_b'(x) = e^x - 1 = 0, \quad x = 0.0 \tag{10}$$

thus, it is omitted from figure 6 below.

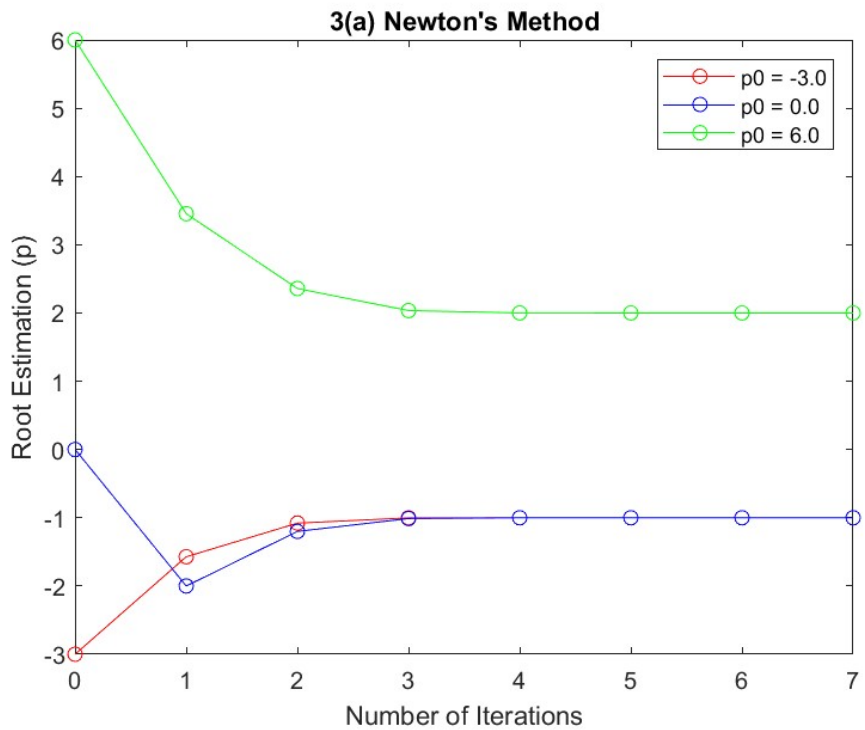

Figure 5: 2(a) Newton's Method

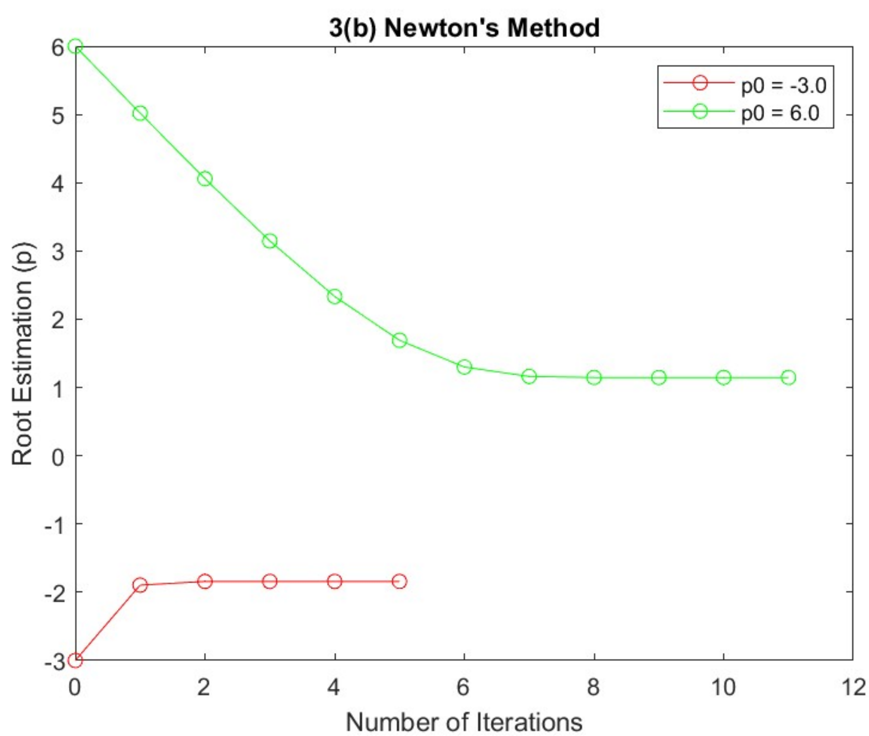

Figure 6: 2(b) Newton's Method

```
function [root, iteration, guess_list] = fixed_point_iteration(g, tolerance,
    guess)
    x0 = guess;
    flag = true;
    iteration = 0;
    max_iterations = 1000;
    guess_list = [guess];

    while (flag || (iteration<max_iterations))
        x = g(x0);
        if(abs(x-x0)<tolerance)
            flag = false;
            root = x;
            break
        end
        x0=x;
        guess_list = [guess_list, x0];
        iteration = iteration + 1;
    end
end
```

fixed_point_iteration.m

```
function [root, iterations, guess_list] = bisection_method(f, a,b, tolerance)
    iterations = 0;
    tol = 1;
    if ( f(a) == 0 )
        root = a;
        return;
    elseif ( f(b) == 0 )
        root = b;
        return;
    elseif ( f(a) * f(b) > 0 )
        fprintf('f(a) and f(b) do not have opposite signs');
        return;
    end

    guess_list=[];

    while (tol>tolerance)
        iterations = iterations+1;
        mid=(a+b)/2;
        fa=f(a);
        fb=f(b);
        fmid=f(mid);
        guess_list = [guess_list, mid];

        if (fa*fmid < 0)
            b=mid;
            tol = abs(fa-fmid);
        elseif (fb*fmid <0)
            a=mid;
            tol = abs(fb-fmid);
        elseif (fmid==0)
            root = mid;
            tol=0;
        else
            fprintf('f(a), f(b), and f(mid) all have the same sign\n');
            root = "N/A";
            return;
        end
        root = mid;
    end
end
```

bisection_method.m

```
function [root, iterations, guess_list] = newtons_method(f, fprime, guess1, delta)
    iterations = 0;
    tol=1;
    guess = guess1;
    guess_list = [guess];
    while (tol>delta || abs(f(guess))>delta)
        iterations = iterations+1;
        x = guess;
        prime = fprime(x);
        guess = x-(f(x)/prime);
        if (isnan(guess)|| isinf(guess))
            fprintf("x=%.2f is or is near a local maximum or minimum so results in an error\n",guess1)
            root = "N/A";
            return;
        end
        guess_list = [guess_list, guess];
        tol = abs(f(guess)-f(x));
    end
    root=guess;
end
```

newtons_method.m

```
%% Zack Humphries
% COMP 521
% HW5

clc;        % clear command window
clear;      % removes all saved variables
close all; % close any open windows


%% 1(a)
ga = @(x) (x+2)^(1/2);
tol = 10^(-9);

p0 = [2.5 0.15 1.5];
plot_color_setting = ["-or", "-ob", "-og"];
fprintf("Fixed Point Iteration:\n")
fprintf("a)\n")
for n=1:3
    [root, iterations, guess_list] = fixed_point_iteration(ga, tol, p0(n));
    fprintf("With p0=%.2f, the root is %.2f and it took %i iterations to find the root\n", p0(n), root, iterations)
    plot([0:1:iterations], guess_list, plot_color_setting(n))
    hold on
end

ga = @(x) -(x+2)^(1/2);
for n=1:3
    [root, iterations, guess_list] = fixed_point_iteration(ga, tol, p0(n));
    fprintf("With p0=%.2f, the root is %.2f and it took %i iterations to find the root\n", p0(n), root, iterations)
    plot([0:1:iterations], guess_list, plot_color_setting(n))
    hold on
end
title("1(a) Fixed Point Iteration")
xlabel("Number of Iterations")
ylabel("Root Estimation (p)")
legend("p0 = 2.5", "p0 = 0.15", "p0 = 1.5")
fprintf("\n")
hold off


% 1(b)
gb = @(x) exp(x)-2;
plot_color_setting = ["-ob", "-om"];
fprintf("b)\n")
figure(2)
p0 = [0.15 0.25]; % 2.5 doesn't work
for n=1:2
    [root, iterations, guess_list] = fixed_point_iteration(gb, tol, p0(n));
    fprintf("With p0=%.2f, the root is %.2f and it took %i iterations to find the root\n", p0(n), root, iterations)
    plot([0:1:iterations], guess_list, plot_color_setting(n))
    hold on
end
title("1(b) Fixed Point Iteration")
```

```
xlabel("Number of Iterations")
ylabel("Root Estimation (p)")
legend("p0 = 0.15", "p0 = 0.25")
fprintf("\n")
hold off

%% 2(a)
fa = @(x) -(x^2)+x+2;
fb = @(x) exp(x)-x-2;
figure(3)

range1 = [-4.0 1.0];
range2 = [0.5 3.0];
fprintf("Bisection Method:\n")
fprintf("a)\n")
[root, iterations, guess_list] = bisection_method(fa, range1(1),range1(2), tol);
plot([1:1:iterations], guess_list, '-or')
fprintf("With a=%.2f and b=%.2f, the root is %.2f and it took %i iterations to find the root\n", range1(1), range1(2), root, iterations)
hold on
[root, iterations, guess_list] = bisection_method(fa, range2(1),range2(2), tol);
plot([1:1:iterations], guess_list, '-ob')
fprintf("With a=%.2f and b=%.2f, the root is %.2f and it took %i iterations to find the root\n", range2(1), range2(2), root, iterations)
fprintf("\n")
title("2(a) Bisection Method")
xlabel("Number of Iterations")
ylabel("Root Estimation (p)")
legend("a=-4.0 & b=1.0", "a=0.5 & b=3.0")
hold off

%2(b)
figure(4)
fprintf("b)\n")
[root, iterations, guess_list] = bisection_method(fb, range1(1),range1(2), tol);
plot([1:1:iterations], guess_list, '-ob')
fprintf("With a=%.2f and b=%.2f, the root is %.2f and it took %i iterations to find the root\n", range1(1), range1(2), root, iterations)
hold on
[root, iterations, guess_list] = bisection_method(fb, range2(1),range2(2), tol);
plot([1:1:iterations], guess_list, '-or')
fprintf("With a=%.2f and b=%.2f, the root is %.2f and it took %i iterations to find the root\n", range2(1), range2(2), root, iterations)
fprintf("\n")
title("2(b) Bisection Method")
xlabel("Number of Iterations")
ylabel("Root Estimation (p)")
legend("a=-4.0 & b=1.0", "a=0.5 & b=3.0")
hold off
```

```
%% 3(a)
fprimea = @(x) -2*x+1;
fprimeb = @(x) exp(x)-1;
figure(5)

p0 = [-3.0 0.0 6.0];
plot_color_setting = ["-or", "-ob", "-og"];
fprintf("Newton's Method:\n")
fprintf("a)\n")
for n=1:3
    [root, iterations, guess_list] = newtons_method(fa, fprimea, p0(n), tol);
    plot([0:1:iterations], guess_list, plot_color_setting(n))
    hold on
    fprintf("With p0=%.2f, the root is %.2f and it took %i iterations to find the root\n", p0(n), root, iterations)
end
fprintf("\n")
title("3(a) Newton's Method")
xlabel("Number of Iterations")
ylabel("Root Estimation (p)")
legend("p0 = -3.0", "p0 = 0.0", "p0 = 6.0")
fprintf("\n")
hold off

% 3(b)
fprintf("b)\n")
figure(6)
for n=1:3
    [root, iterations, guess_list] = newtons_method(fb, fprimeb, p0(n), tol);
    if (isnumeric(root))
        plot([0:1:iterations], guess_list, plot_color_setting(n))
        hold on
    end
    fprintf("With p0=%.2f, the root is %.2f and it took %i iterations to find the root\n", p0(n), root, iterations)
end
fprintf("\n")
fprintf("\n")
title("3(b) Newton's Method")
xlabel("Number of Iterations")
ylabel("Root Estimation (p)")
legend("p0 = -3.0", "p0 = 6.0")
```

main.m