Assignment 5:

- **Fixpoint method: The used script is "main_fixptmethod.m"**

Results:

Starting points	2.5	0.15	1.5
Approximate fixed points	-inf	0.5549	0.5549
Final Errors	NaN	1.3569	1.3569

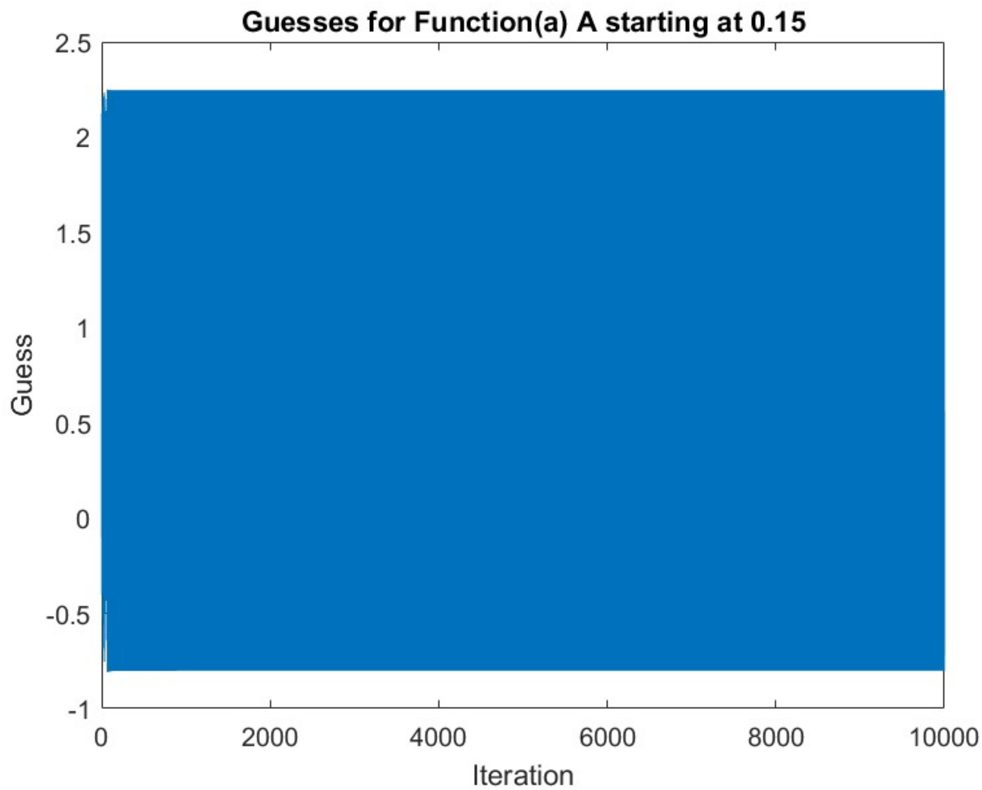
"Results for $f_a(x) = -x^2 + x + 2$"

Starting points	2.5	0.15	1.5
Approximate fixed points	NaN	-0.7680	0.7680
Final Errors	NaN	$10^{-9} * 0.7629$	$10^{-9} * 0.6844$

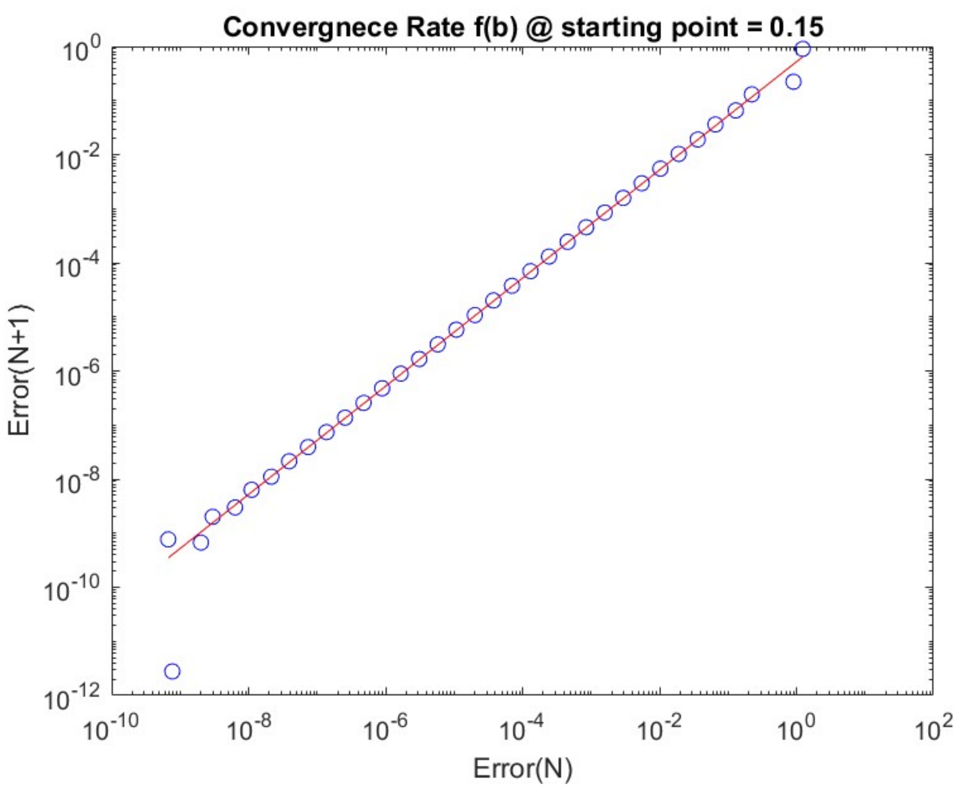
"Results for $f_b(x) = e^{-x} - 2 - x$"

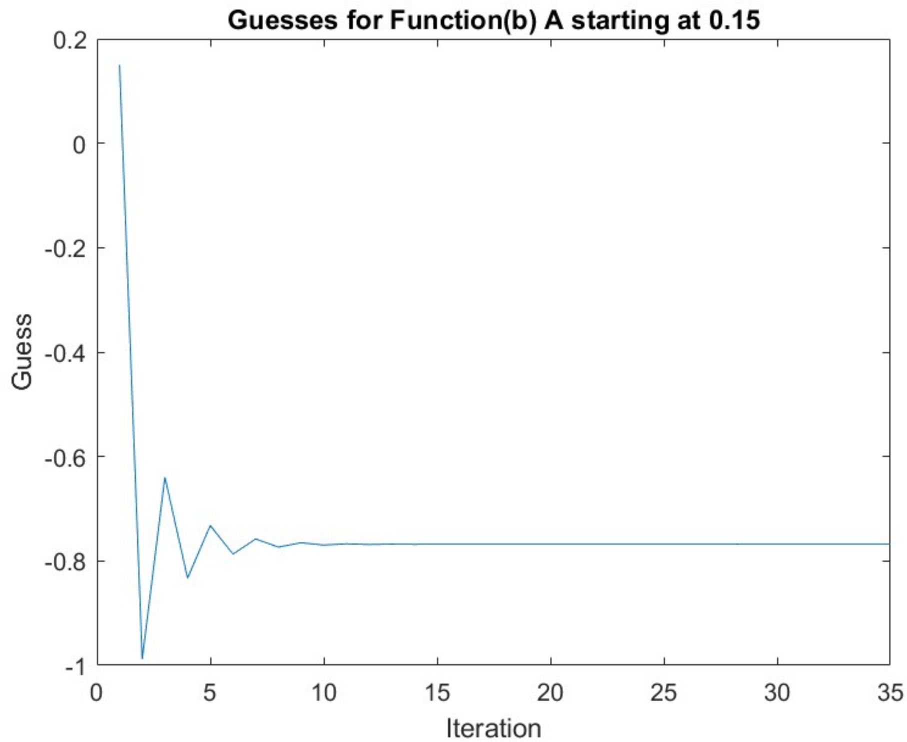
There are a NaN and Inf values in my results indicating that the fixed-point iteration did not converge to a finite value for initial guesses. For function (a), the approximate fixed points seem to be repeated. The value "0.5549" repeats itself. This indicates that the method is not converging to the desired root or possibly encountering a repeated value as a solution. It seems that errors for function (b) are lower than function (a) in this method. From plots of function (a), we can see that the guesses seem to oscillate between a couple of values, particularly around the 0.55 mark. It could indicate a convergence issue or an oscillation around a specific value. In function (b), the behavior of the algorithm seems more erratic, with some guesses leading to an (x) value of "NaN" and others oscillating around two different values. Reasons for failure of this method may be that the iterative process might not be converging due to incorrect functions chosen for the fixed-point iteration. It can be for this that starting points might not be appropriately selected, causing the method to get stuck at a certain region.

The plot which is guesses_A vs iteration, for function (a) at initial points 1.5 and 0.15, appears like a "brick", and it seems that fixpoint iteration not converging and is oscillating around the starting guess. This behavior can happen due to the reasons I mentioned above. Guesses vs iteration for initial point 2.5 is normal and it converges to the answer. The fixed-point iteration method requires that the derivative of the function at the fixed point should be less than 1 in absolute value for convergence. If the function's derivative doesn't meet this condition, the method might fail or take longer to converge.



"Brick looks Plot: Guesses VS Iteration, Start point = 1.5, 0.15"

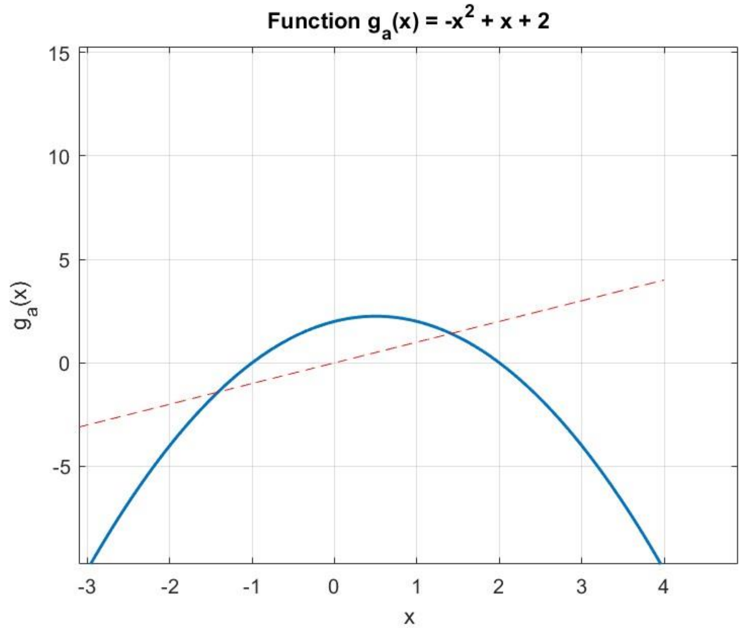


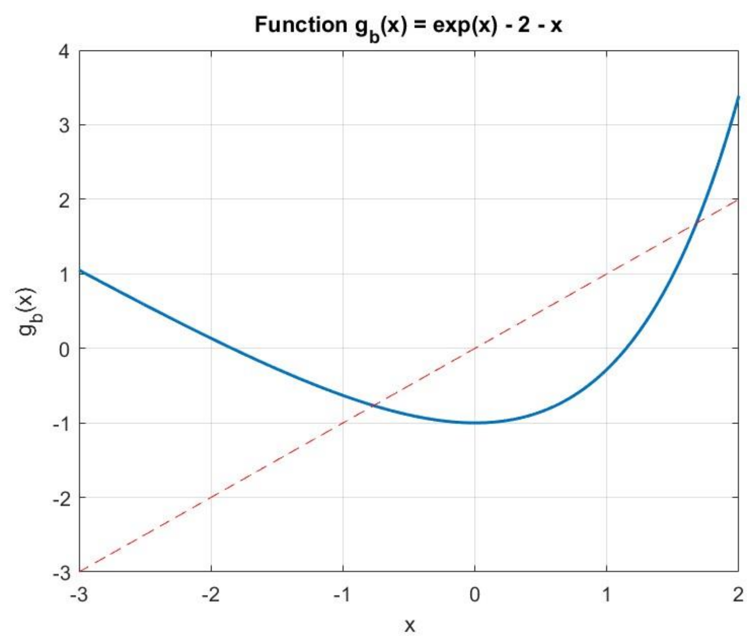


"Guesses VS Iteration, Start point = 0.15"

The plots of guesses vs iterations for function(b) do not look like a brick, and they converge into an answer. (I only add one plot to see this.)

I wanted to try other starting points to check the fixpt functions for them, but it's challenging to predict a suitable initial guess without having more information. One way of finding better initial guesses that might result in successful convergence, we can try a systematic approach, plot the functions, and visually inspect where they intersect with the line **y = x**. The intersection points can be potential initial guesses for fixed-point iteration. Here are the plots:





- **Bisection method: The used script is "main_bisectmethod.m"**

Results:

Interval	-4	1
Approximate Solution	-1	

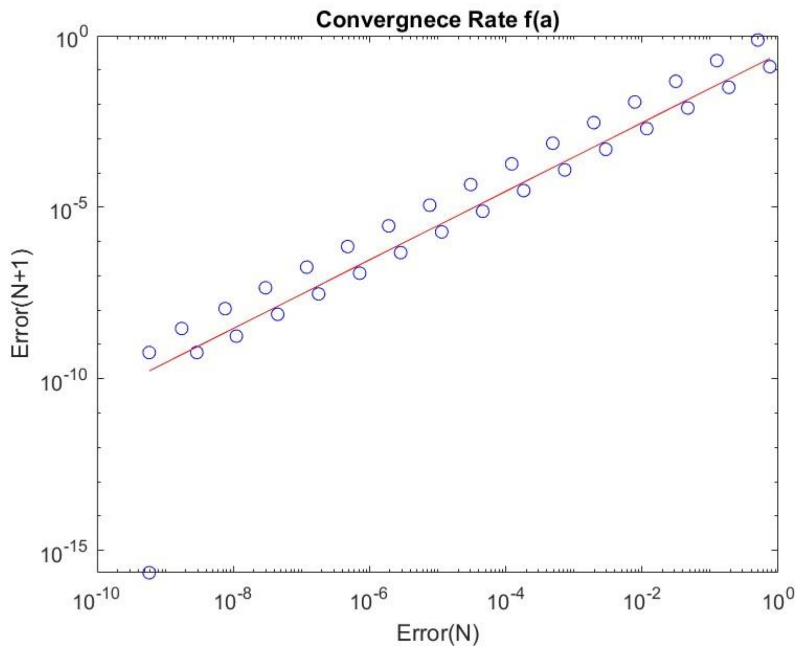
"Results for $f_a(x) = -x^2 + x + 2$"

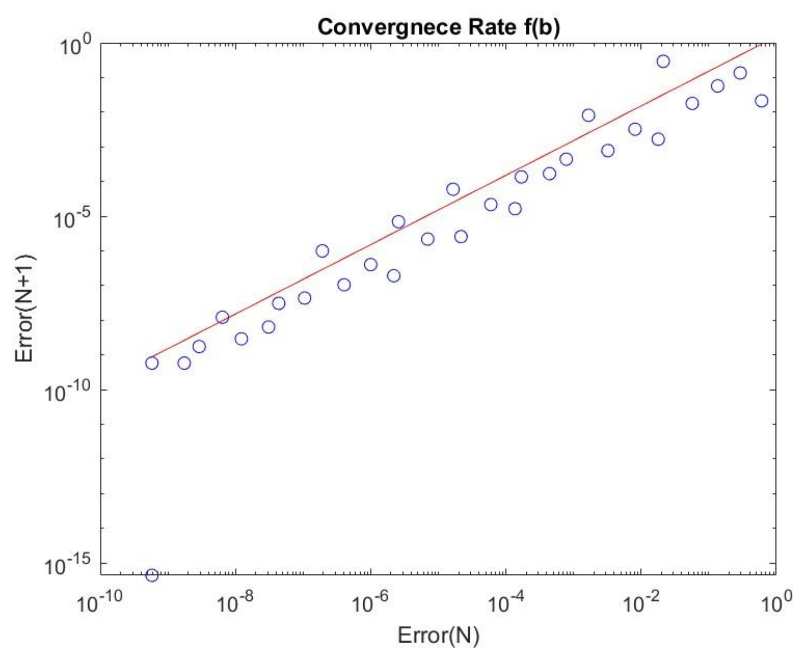
Starting points	0.5	3

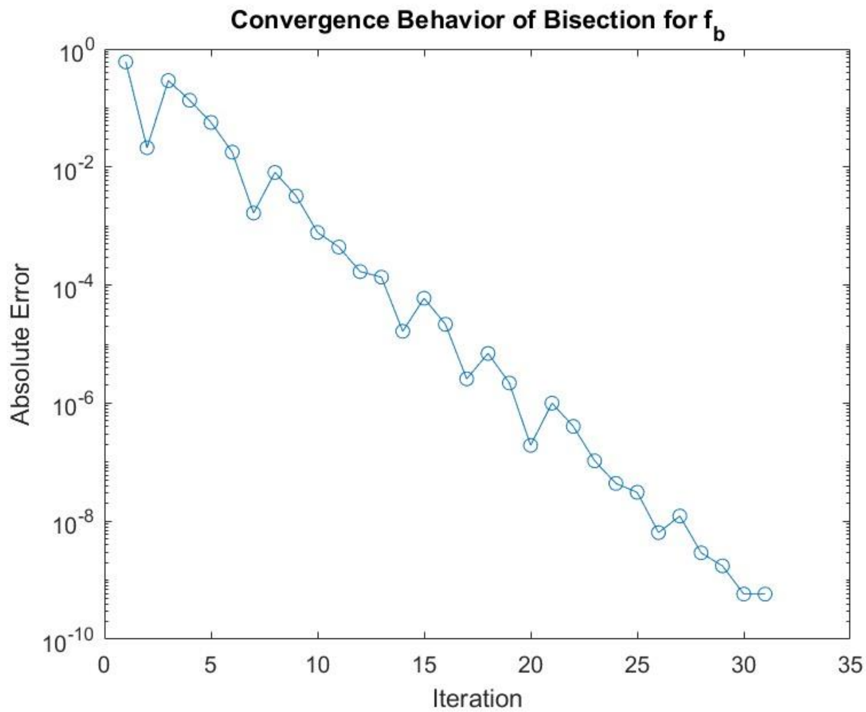
Approximate Solution	1.146192

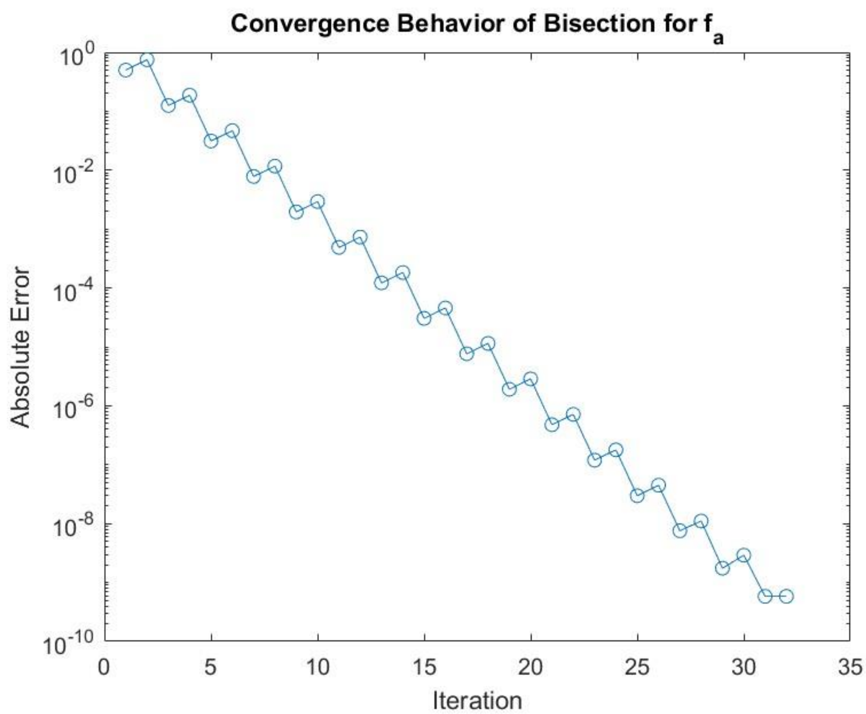
"Results for $f_b(x) = e^{-x} - 2 - x$"

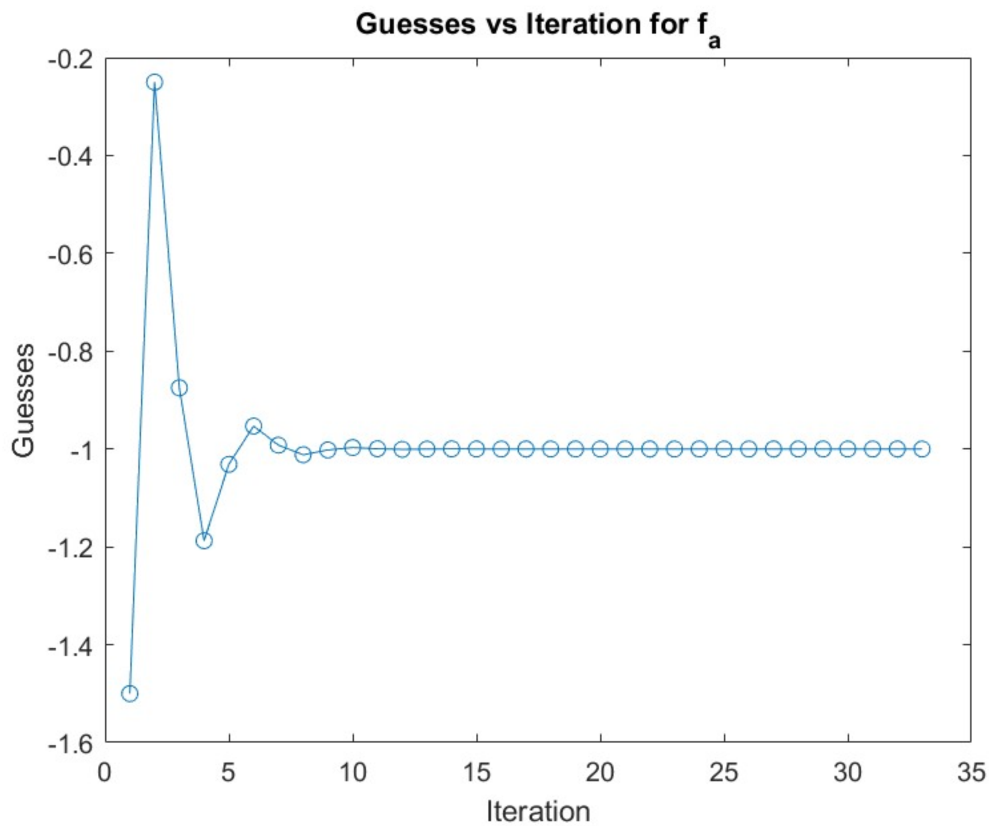
The bisection method gives us the solution, only if the solution exists uniquely in some known interval. However, the convergence of the fixed-point iteration depends on the derivative of g(x) as well as the initial value x0.

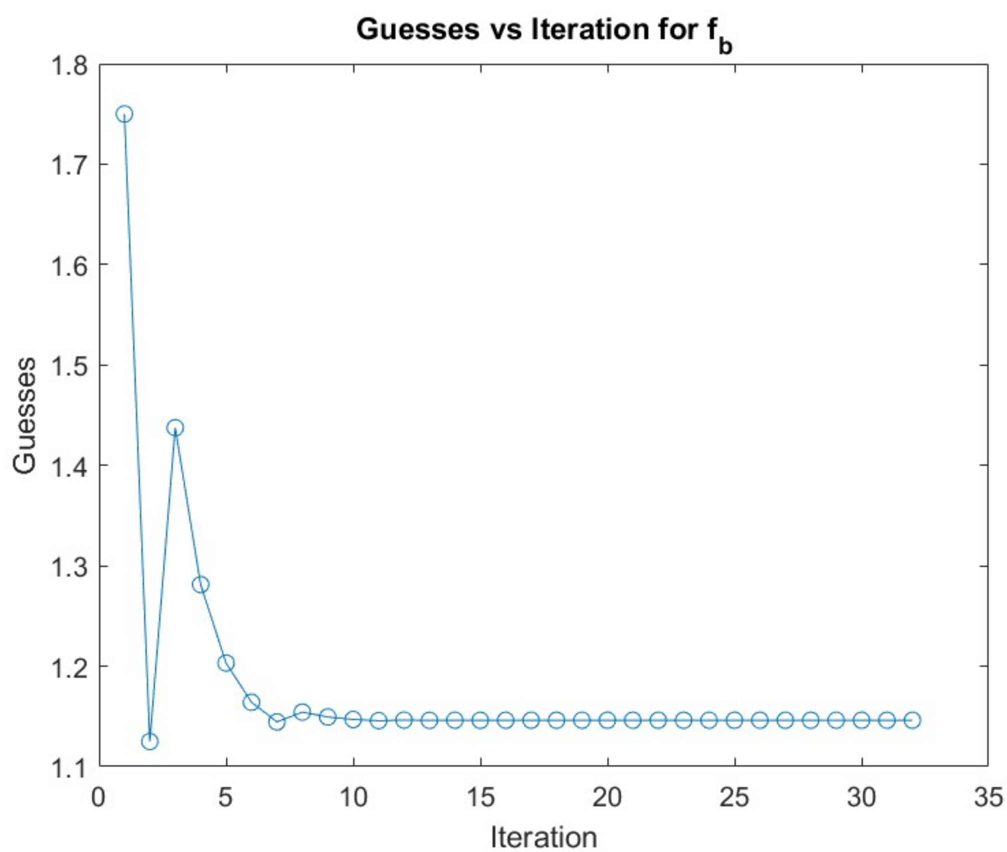





Convergence Behavior of Bisection for f_b
Absolute Error
Iteration
10^0
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
5
10
15
20
25
30
35


Convergence Behavior of Bisection for f_a
Absolute Error
Iteration
10^0
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
5
10
15
20
25
30
35


Guesses vs Iteration for f_a
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
Guesses
0
5
10
15
20
25
30
35
Iteration


Guesses vs Iteration for f_b
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
Guesses
0
5
10
15
20
25
30
35
Iteration

The bisection method seems to work well in that specific interval and converges to a solution between 30-35 iterations, and it finds the root for each function.

- **Newton method: The used script is "main_newtonmethod.m"**

Results:

Starting points	-3	0	6
Approximate Solution	-1	-1	2

"Results for $f_a(x) = -x^2 + x + 2$"

Starting points	-3	0	6
Approximate fixed points	-1.8414	NaN	1.1462

"Results for $f_b(x) = e^{-x} - 2 - x$"

I have created an array that includes the number of iterations (function (a)) for each starting point to see how close it gets each step to the final solution. Tables 1, 2, and 3 show the number of iterations and its results for starting points -3 (7 iteration), 0 (7 iteration), and 6 (8 iteration).

# Iterations	1	2	3	4	5	6	7
Value	-3	-1.5714	-1.0788	-1.0020	-1.0000	-1.0000	-1

"# of Iterations for $f_a(x) = -x^2 + x + 2$, Start Point=-3"

# Iterations	1	2	3	4	5	6	7
Value	0	-2	-1.2000	-1.0118	-1.0000	-1.0000	-1

"# of Iterations for $f_a(x) = -x^2 + x + 2$, Start Point=0"

# Iterations	1	2	3	4	5	6	7	8
Value	6	3.4545	2.3580	2.0345	2.0000	2.0000	2.0000	2

"# of Iterations for $f_a(x) = -x^2 + x + 2$, Start Point=6"

For function (b): Tables 1, 2, and 3 show the number of iterations and its results for starting point -3 (5 iterations), 0 (101 iterations and it does not converge to a solution when I increased the iteration to 10000, it did not find any solution again), and 6 (12 iterations).

# Iterations	1	2	3	4	5
Value	-3	-1.8952	-1.8417	-1.8414	-1.8414

"# of Iterations for $f_b(x) = e^{-x} - 2 - x$, Start Point=-3"

#Iterations	1	2	3	4	5	6	7	8	9	10	11	12	13
Value	0	Inf	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

"# of Iterations for $f_b(x) = e^{-x} - 2 - x$, Start Point=0"

#Iterations	1	2	3	4	5	6	7	8	9	10	11	12	13
Value	6	5.0174	4.0575	3.1465	2.3328	1.6909	1.2991	1.1615	1.1464	1.1462	1.1462	1.1462	1.1462

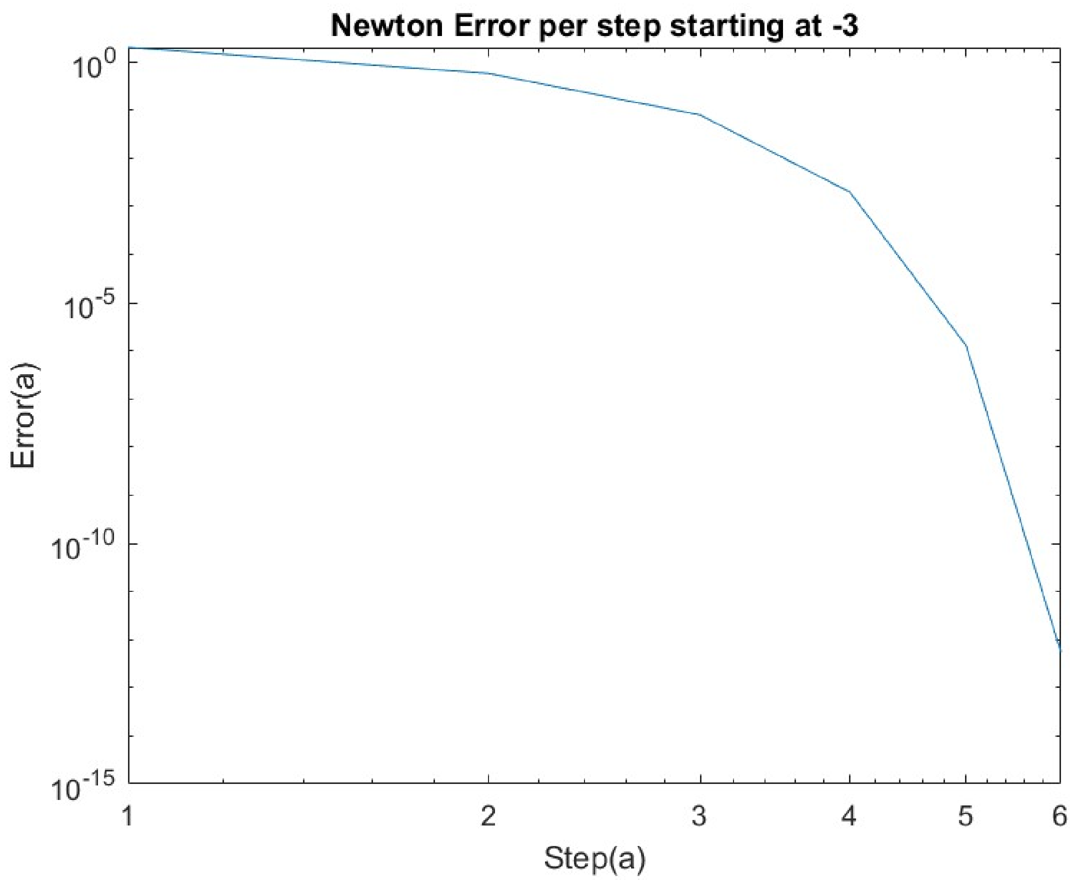
"# of Iterations for $f_b(x) = e^{-x} - 2 - x$, Start Point=6"

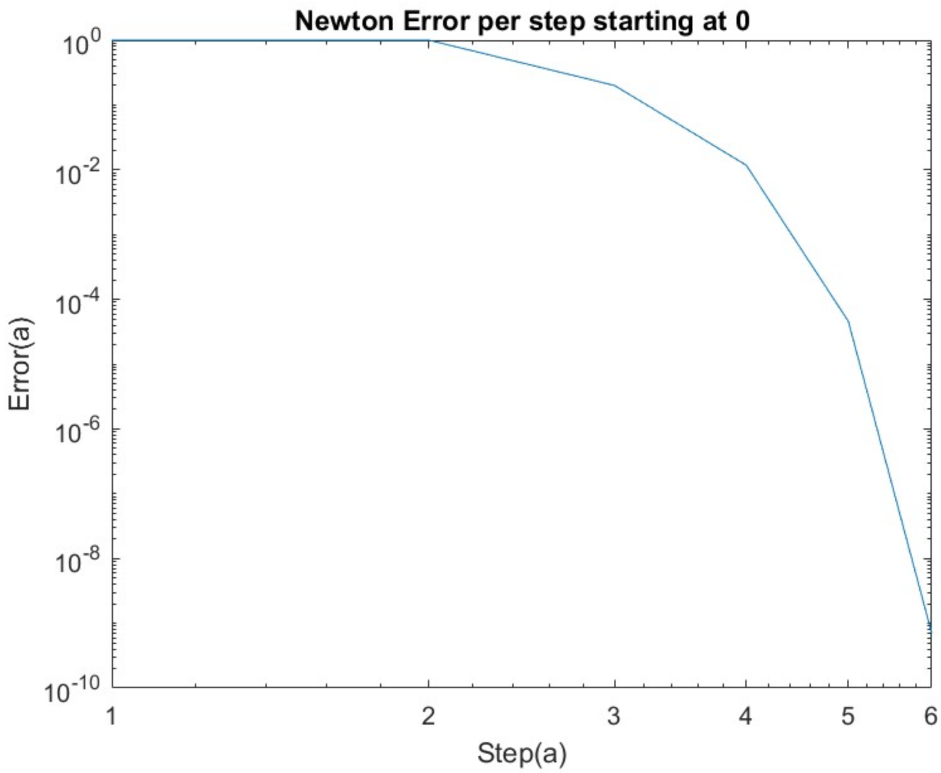
Explanation:

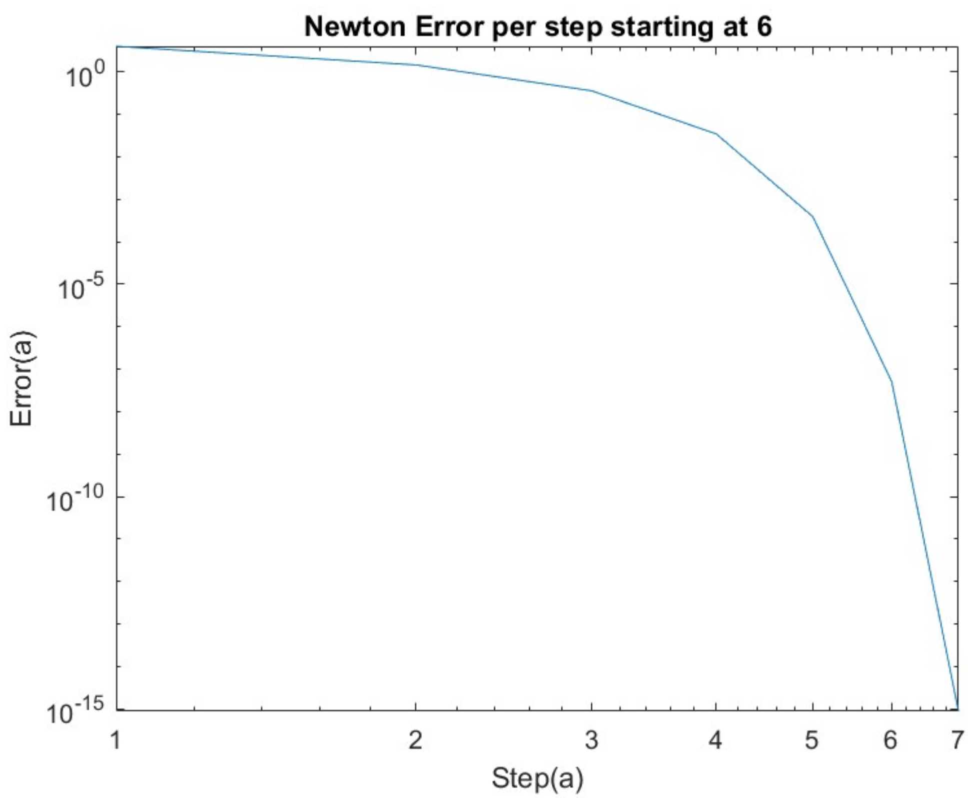
In this method, I observed that the error will decrease in each step size as we proceed. The drawbacks of the Newton Method are the effort and time required to compute the derivative of the function at each iteration, which might be computationally expensive. The method can go astray, especially when f(x) has an abruptly changing slope around the solution, causing divergence.

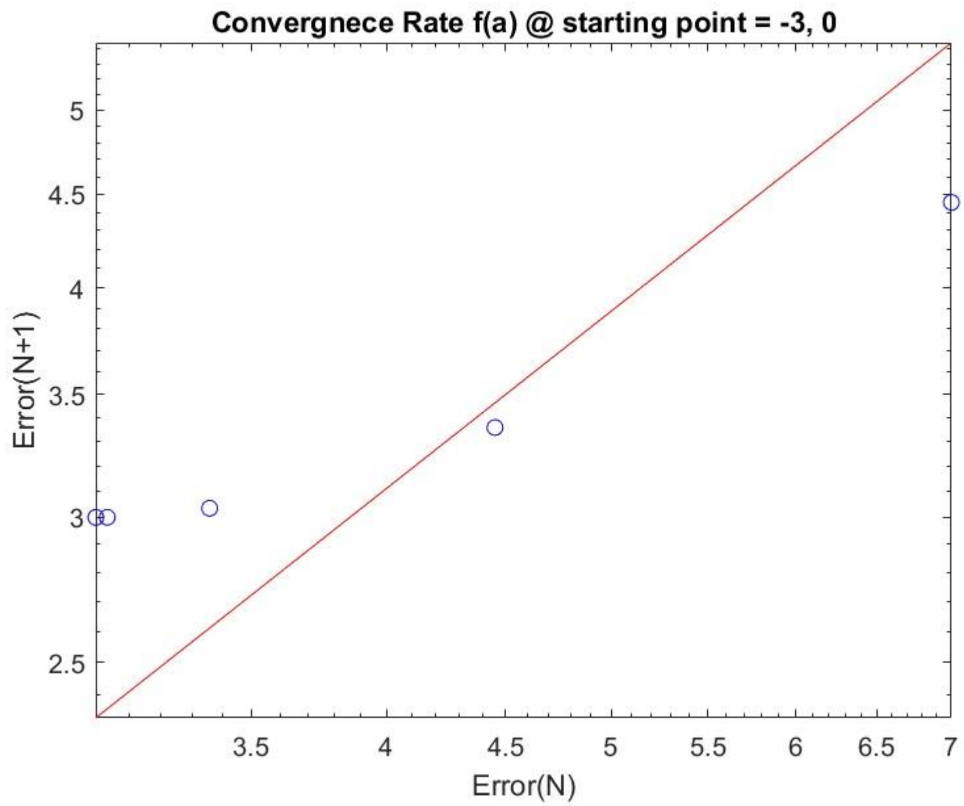
The reason that I could not get any solution at the starting point might be because of these reasons that Newton's method might fail if the chosen starting point is far from the root or is within a region where the function has no root. For function (b) the choice of starting point 0 might be closer to a local minimum/maximum where Newton's method can't converge to a root. Also, I believe the function has steep regions near 0, causing the derivative to be large. If the derivative at a chosen point is close to zero, Newton's method can fail to converge.

Plots for function (a): I have removed the extra plots, but they are still in the plot folders.

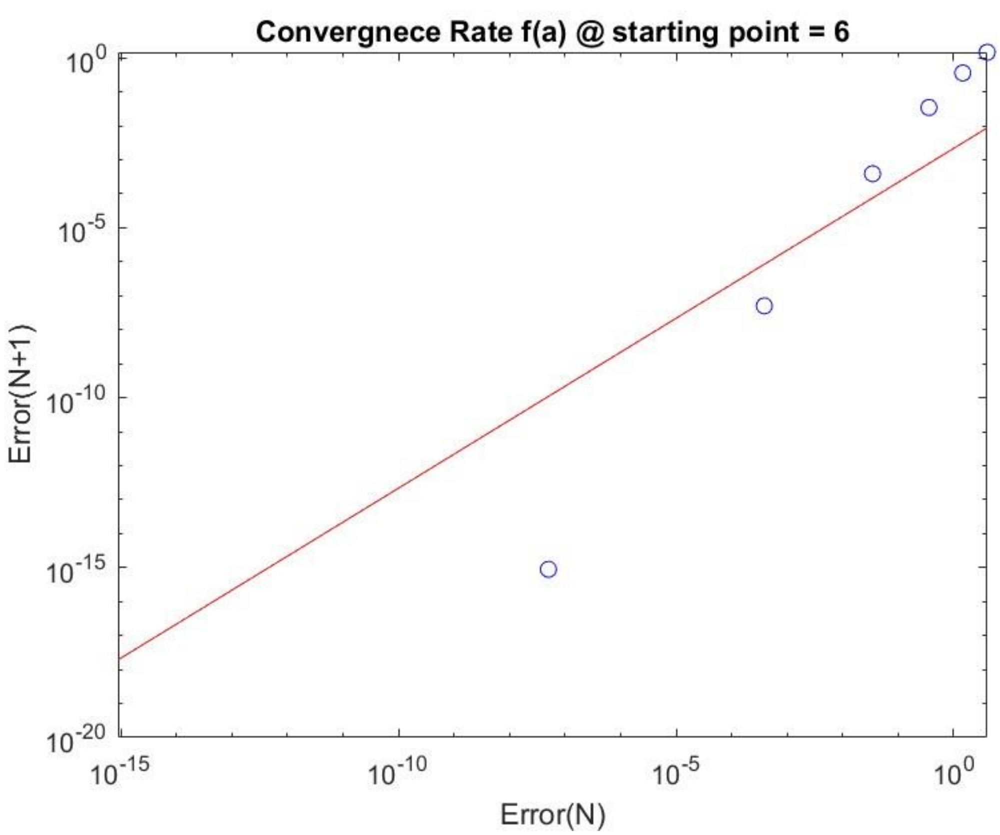



Newton Error per step starting at 0
Error(a)
Step(a)
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
1
2
3
4
5
6


Newton Error per step starting at 6
Error(a)
Step(a)
10^0
10^-5
10^-10
10^-15
1
2
3
4
5
6
7

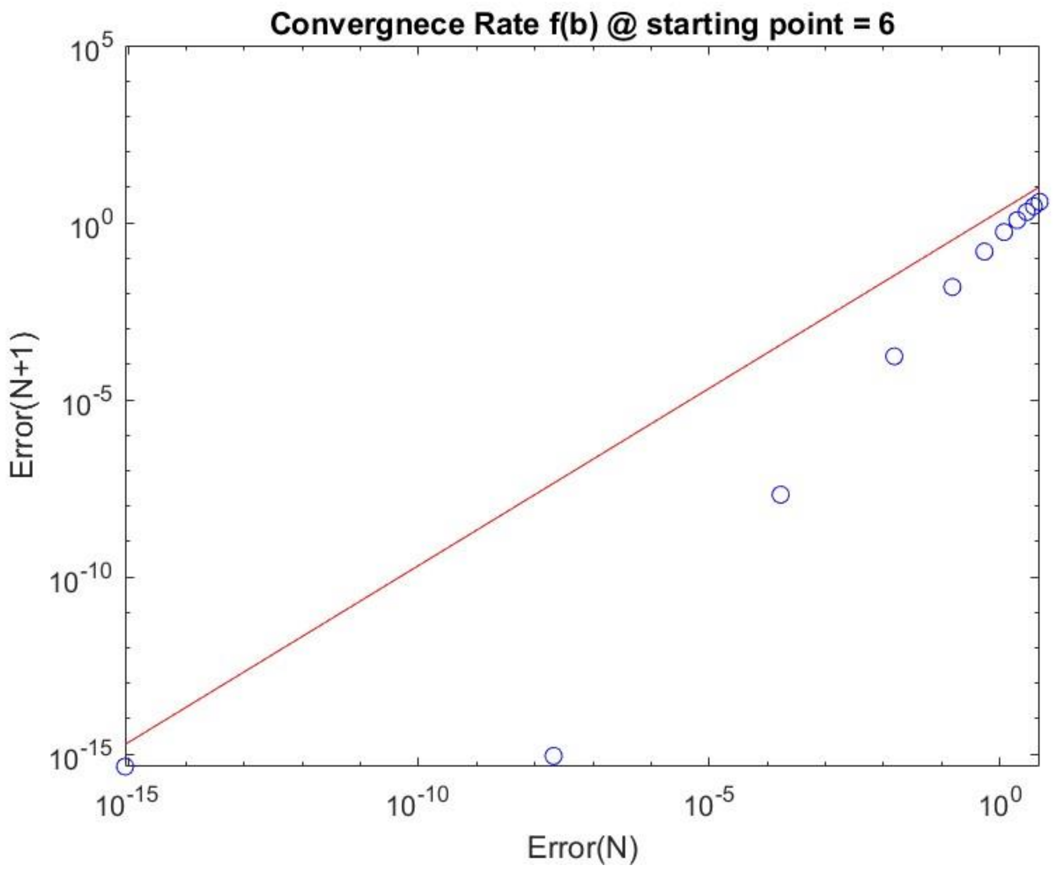


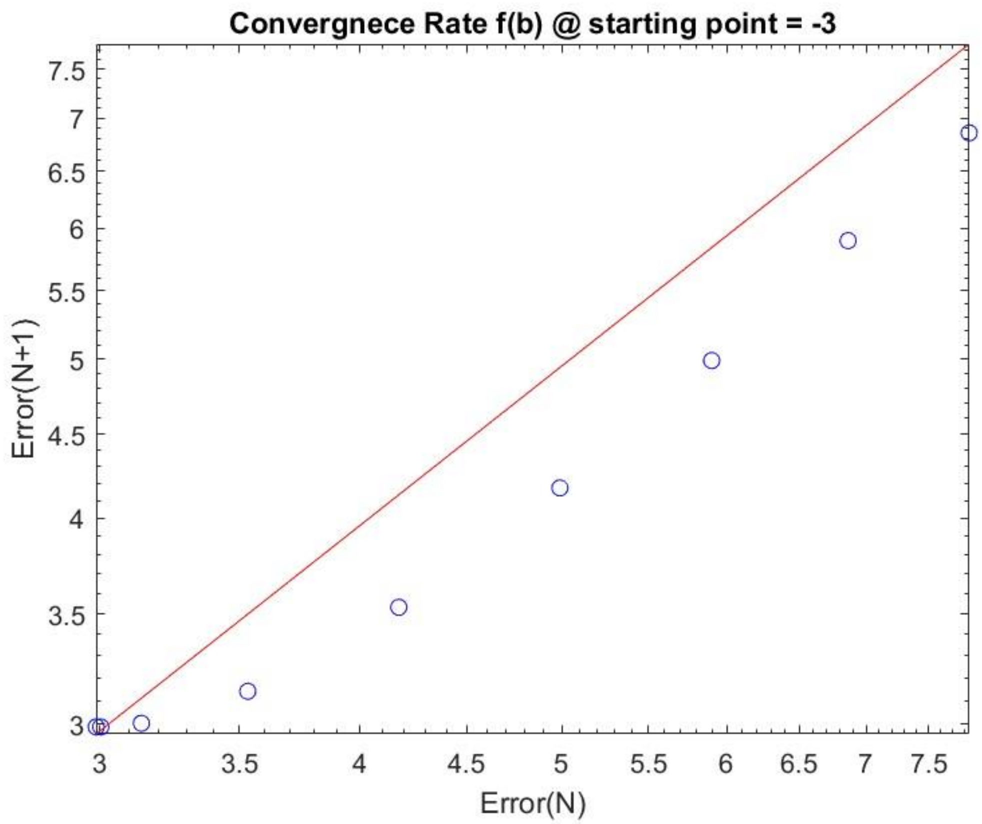
Convergnece Rate f(a) @ starting point = -3, 0
Error(N+1)
Error(N)
5
4.5
4
3.5
3
2.5
3.5
4
4.5
5
5.5
6
6.5
7

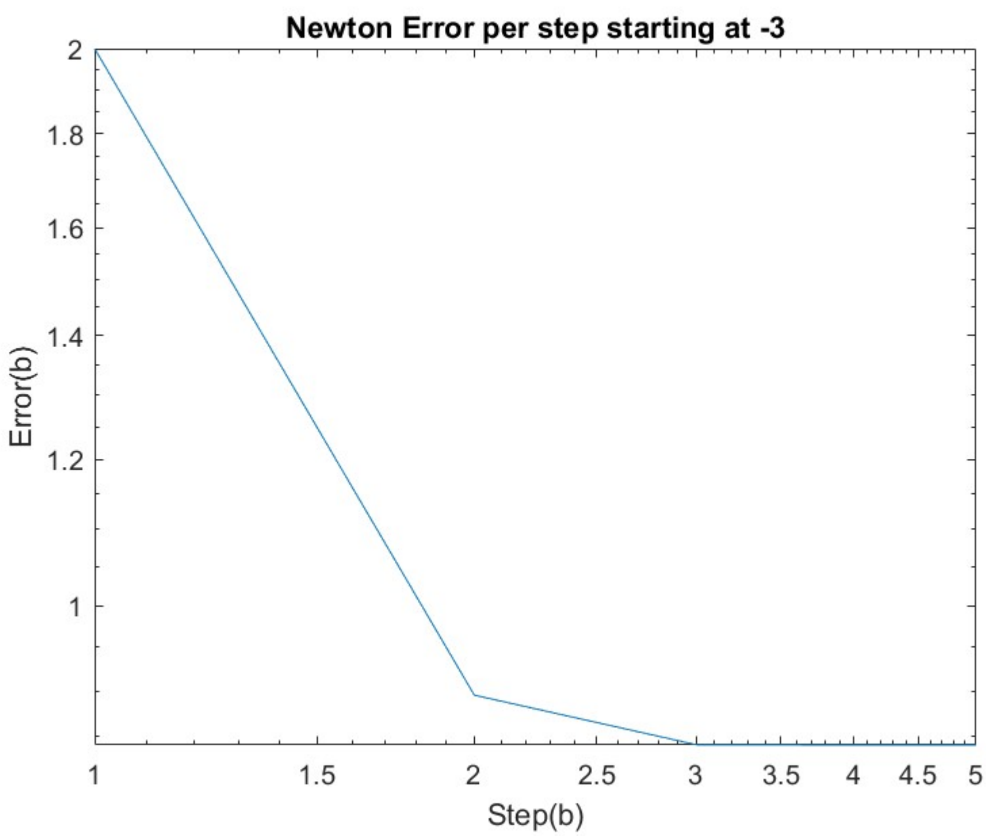


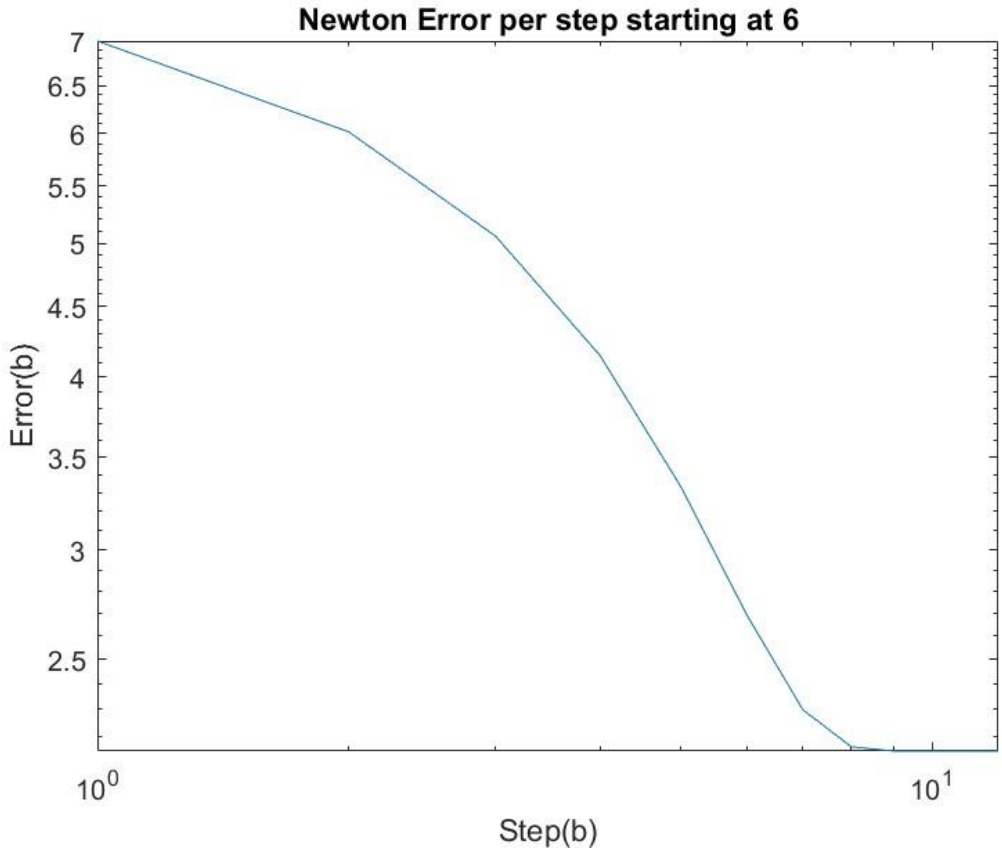
Convergnece Rate f(a) @ starting point = 6
Error(N+1)
Error(N)
10^{0}
10^{-5}
10^{-10}
10^{-15}
10^{-20}
10^{-15}
10^{-10}
10^{-5}
10^{0}

Plots for function (b): There is no Newton Error per step for starting point 0, because Newton could not achive any solution for that starting point.




Convergnece Rate f(b) @ starting point = -3
Error(N+1)
Error(N)
7.5
7
6.5
6
5.5
5
4.5
4
3.5
3


Newton Error per step starting at -3
Error(b)
Step(b)
2
1.8
1.6
1.4
1.2
1
1.5
2.5
3
3.5
4
4.5
5



Newton Error per step starting at 6
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
Error(b)
10^0
10^1
Step(b)