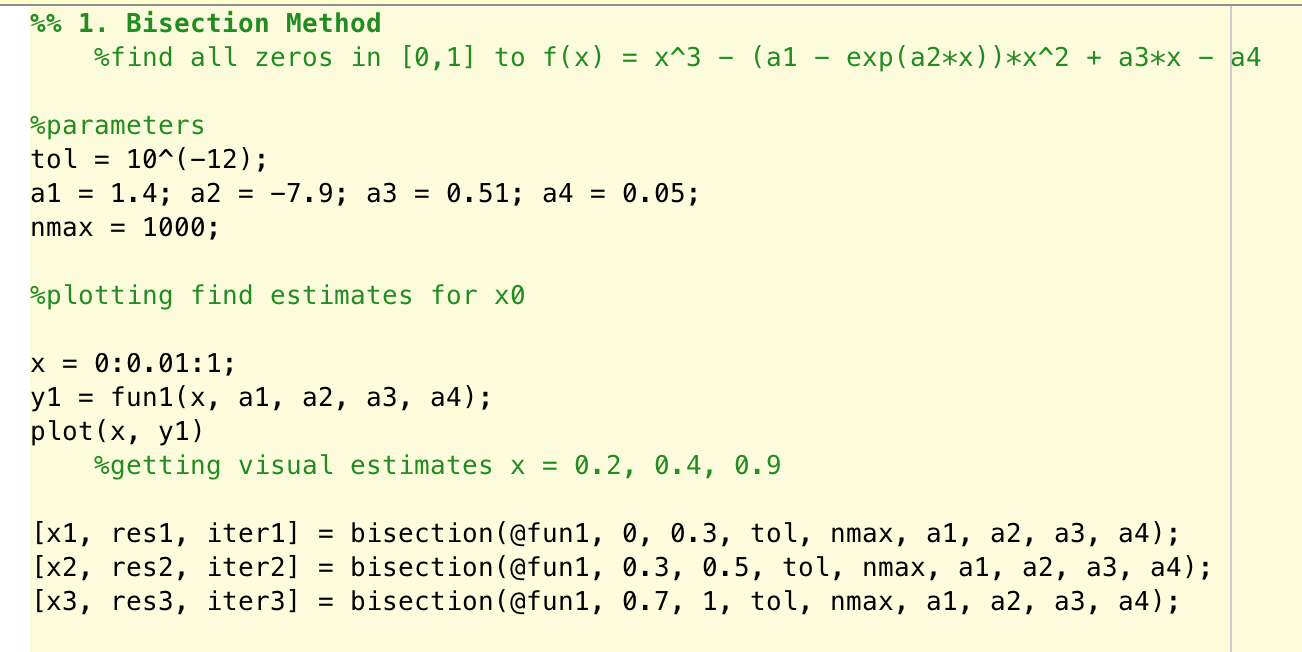
Michael Ahn

**Project 1 Report**

1. **Bisection Method on f**

The problem began with a nonlinear function f, in which we were asked to find the zeros for. I began by implementing the function as a MATLAB file, and applied the Bisection method onto it with the given parameters. Applying the Bisection method required finding an interval for the function, which was done by plotting f and visually estimating a good range in the domain where the function f seemed to cross the x-axis.

The bisection method is very robust and accurate. It can be used on a function without much prior knowledge or inputs. The interval can be large but this will slow down the convergence. Generally, the Bisection method is slow and has a linear convergence. The result is acceptable as a good approximation though, and it converged within the tolerance specified.



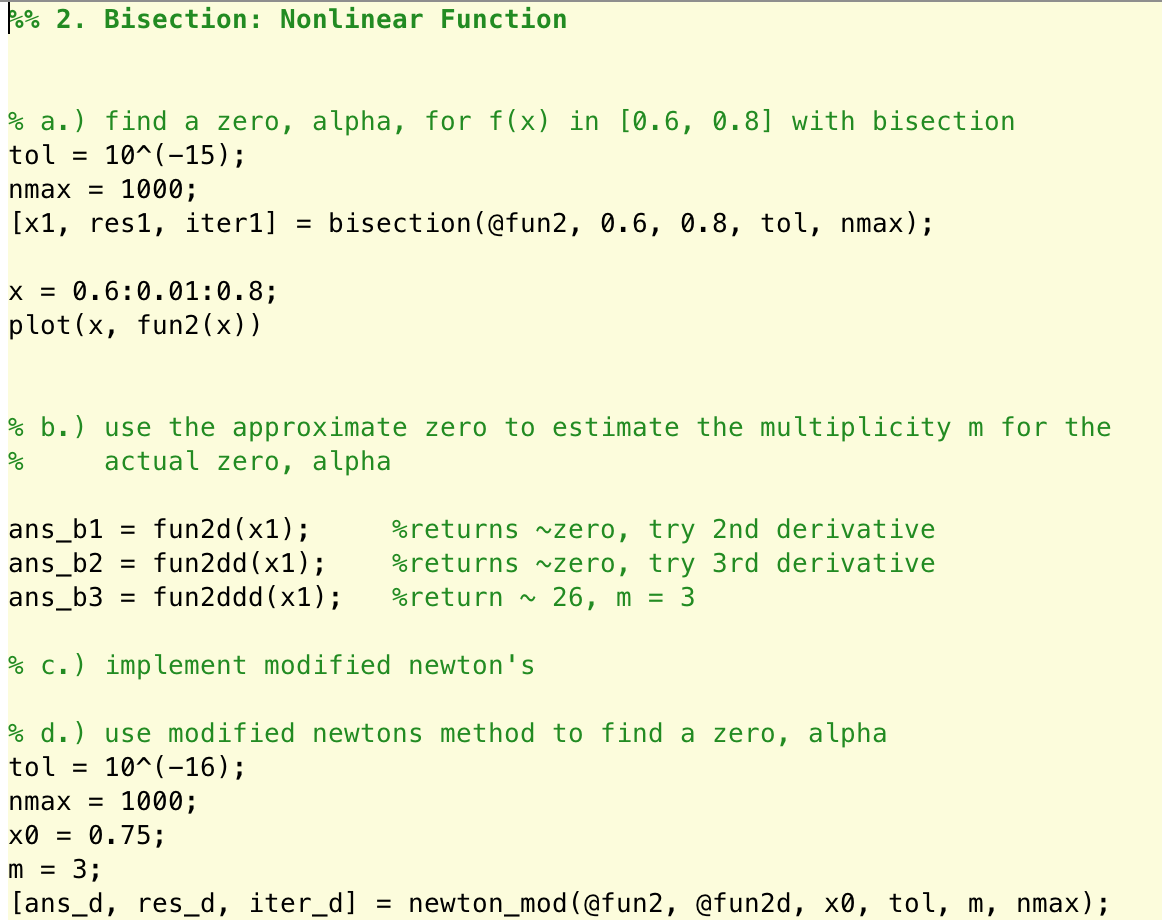
1. **Bisection on nonlinear function**

The problem began with a nonlinear function f and asked for find zeros for it using the Bisection method. The Bisection method yielded good results, but with slow convergence. It was easy to do, with very little information about the function required.

Finding the multicity of the zero required finding the derivative of the function. Evaluating the derivative at the approximated zero yielded a result very close to zero, so I decided that the multiplicity was not 1. After finding the second and third derivative, and implementing them into MATLAB too, plugging in the approximated zero into the third derivative yielded a result relatively far away from zero. This meant that the zero had a multiplicity of 3.

Invoking the modified Newton’s method meant implementing it by changing a few scripts within the original Newton’s method function. Also, I needed to find the derivative of the nonlinear function f and implement it as a separate function in MATLAB. Since the modified Newton’s method uses the function, its derivative, a good initial point close to the zero, and the multiplicity of the zero in question, it requires a lot of information that is sometimes difficult to find. However, the modified Newton’s method converges very fast with low number of iterations. In this case, it had many less iterations than the bisection method, 4 instead of 26.

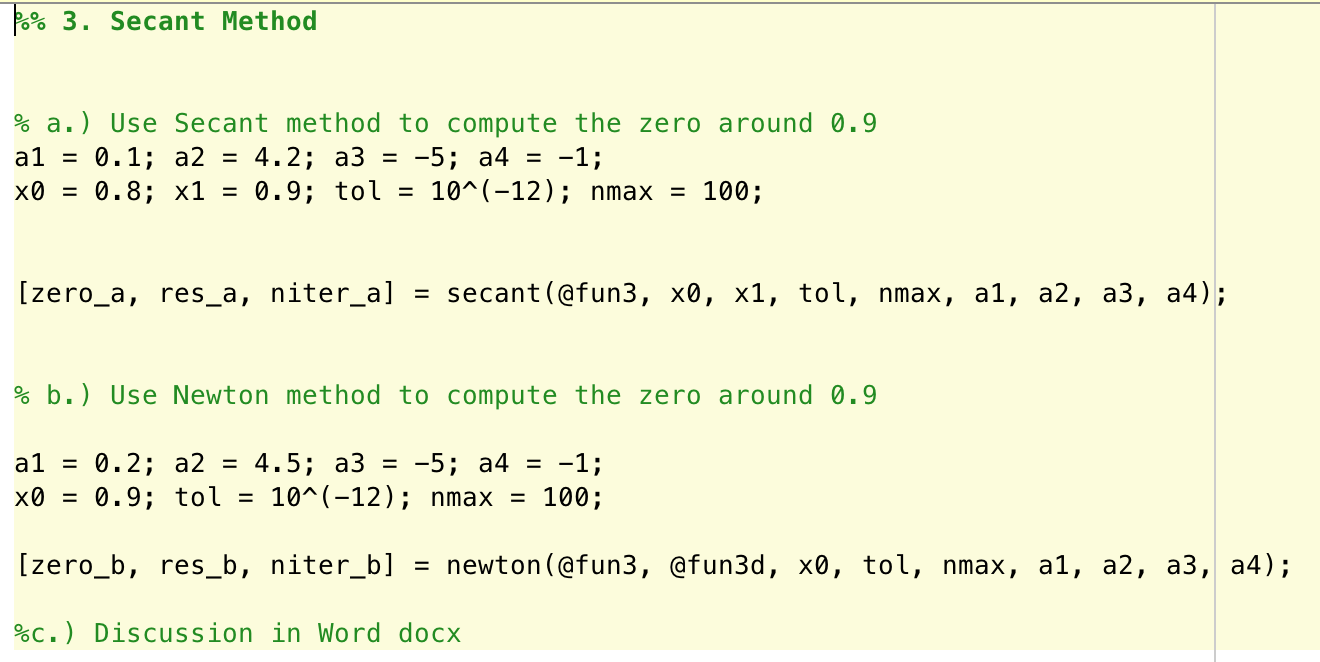
Both of these results were good approximations to the zero for the function. The residuals were low, and actually plugging them back into the function did indeed yield a result very close to zero.



1. **Secant & Newton’s Method**

The problem gave a nonlinear function f and asked to compute the zero for it around 0.9 with the given parameters. This meant implementing the function f into MATLAB as a separate file and calling the secant method from previous assignments. I also used the Newton’s method to approximately compute a zero for this function f.

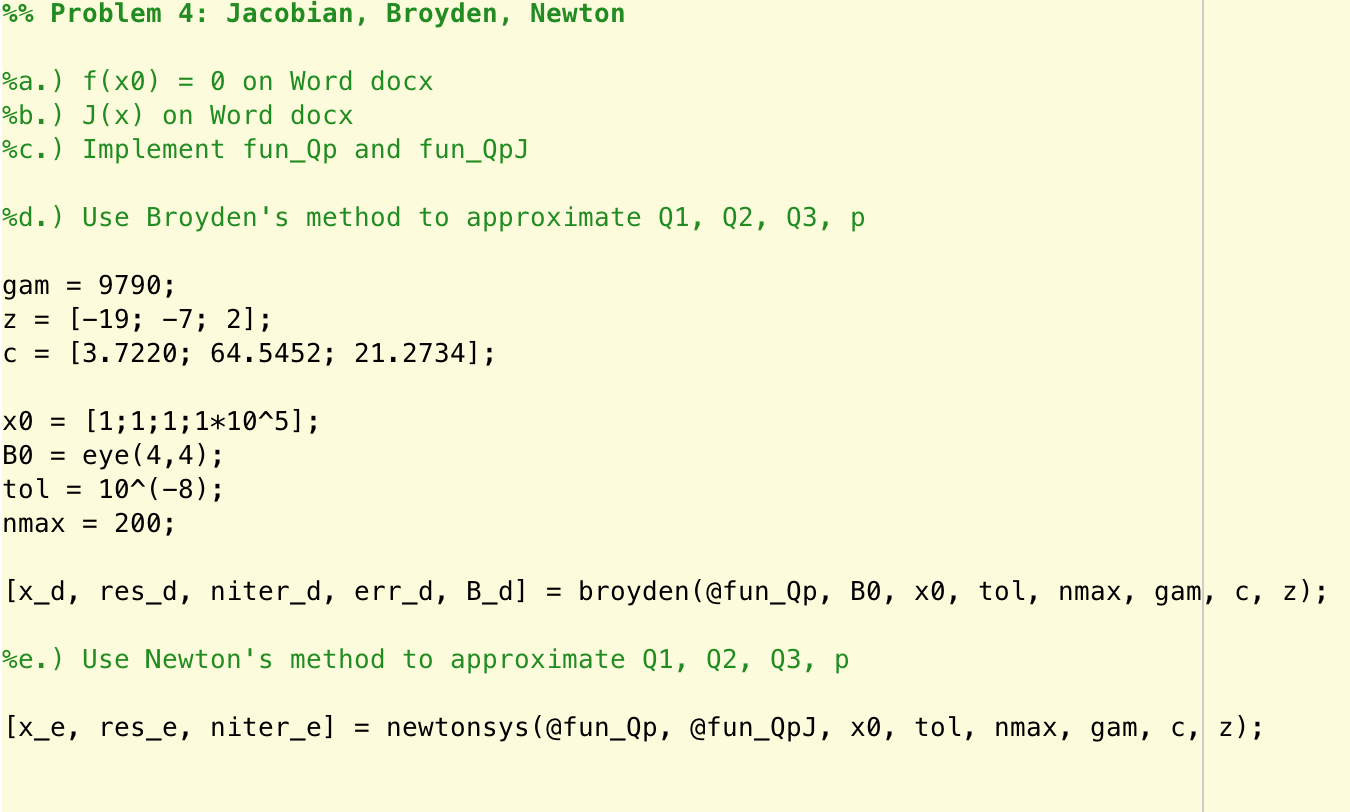
The secant method has its advantage in that it does not require a derivative to be used. Instead, it requires a second initial guess, or another iteration, x1 as well as x0. The Newton’s method requires a derivative to be calculated before invoking. This can sometimes be unreliable, since a derivative is not always easy to find. Sometimes it is impossible. However, in this case, I was able to find the derivative and used it to solve the problem. Both of these methods resulted in good results but the Newton’s method converged faster. This shows that while the Newton’s method converges faster, it is also more expensive in a way that it requires more information to use.



1. **Jacobian & Bryoden, Newton’s Method**

The problem had a homogenous vector function equation and asked to find the Jacobian for it. Both the function and its Jacobian were implemented as MATLAB functions in separate files. The problem asked to find approximations to the equation Broyden’s method with given parameters. Using Broyden’s method from previous assignments with the given parameters results in an unacceptable result. The method did not converge even with 200 iterations, and the residual was left relatively large. This might have been due to the initial B0 value being the identity matrix, and could have been made better by using an actual or even approximated Jacobian matrix as B0.

Using Newton’s method resulted in better approximations. It also converged at 9 iterations, which might be a little bit slow for Newton’s but was much better than the Broyden’s method. I used the Newton’s method for systems of equations, since this was a vector function. The residual was small, since the convergence requires the approximation to be within a certain number of iterations and tolerance.



1. **Gravity Flow with Newton and Broyden Methods**

The problem gave a system of equations in vector form and asked to write them out and implement them. Implementing the function and its Jacobian was done through the use of for loops, since the functions were very similar and there were too many to write quickly one at a time. These functions were used to find f and J at theta0 that was given.

Using newton’s method for systems of equations did not work since the Jacobian was a singular matrix. Instead, I used the newton method for systems of equations with the complex variable approximation for the Jacobian. This results in a relatively fast convergence, at 8 iterations, and is a good result since it falls into the tolerance limit. Plugging back the result into the function also shows by being close to zero that it is a good result.

An appropriate B0 was found by using the approximate Jacobian through complex variables method. Calling on broydens method using the parameters and this B0 lead to a fast convergence too, also at 8 iterations. The results seemed very good too since the residual fell within the tolerance limit and plugging the solution back into the function resulted in a solution very close to 0 for each element.

