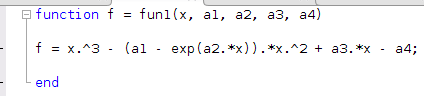
Michael Ahn

Project 1

**1. Bisection Method on f**

|  |  |  |  |
| --- | --- | --- | --- |
| Zero Found | A | B | Iterations Used |
| 1.263927840884207e-01 | 0 | 0.3 | 38 |
| 4.052272300490586e-01 | 0.3 | 0.5 | 37 |
| 8.883793363196675e-01 | 0.7 | 1 | 38 |

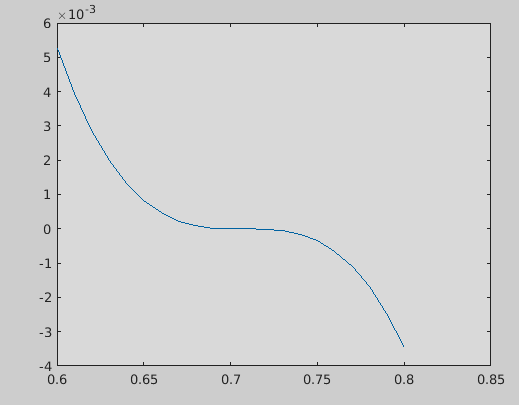
The problem began with implementing the function in a separate MATLAB file, called fun1



**2. Bisection: Nonlinear Function**

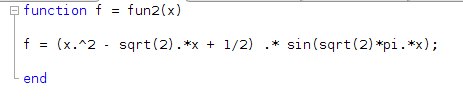
**a.) Find zero in [0.6, 0.8]**

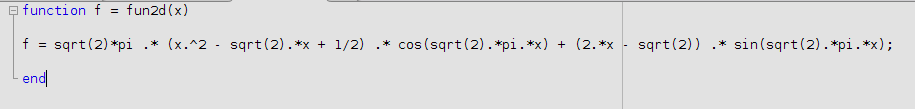
|  |  |  |  |
| --- | --- | --- | --- |
| Zero Found | A | B | Iterations Used |
| 7.071067720651627e-01 | 0.6 | 0.8 | 26 |

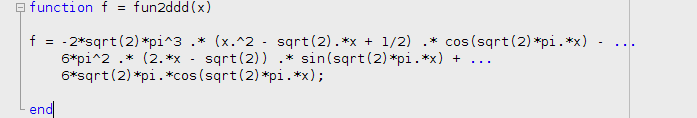
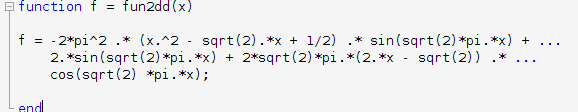
****

**b.) Multiplicity**

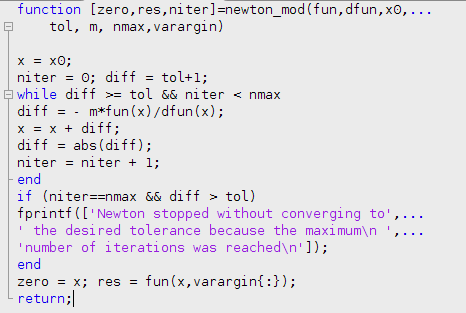
m = 3







**c.) Implement Modified Newton’s, calculating m**



Modified the original Newton’s method by using the multiplicity in each iteration to help calculate the diff value. This increases the efficiency of the algorithm

**d.) Find the zero, alpha, using modified newton’s from c. on [0.6, 0.8]**

**Use estimated m from b. with given parameters**

|  |  |  |
| --- | --- | --- |
| Zero Found | Residual | Iteration Used |
| 7.071067811642787e-01 | 0 | 4 |

We can see that this is much faster than the Bisection method, and gives a slightly different zero since we will always have error in using machine operations, especially when working with small numbers.

**e.) Discuss the modified Newton’s Method**

Newton’s method converges faster than the Bisection method; however, it is “more difficult” to use since it requires a closer estimate to the actual zero. The modified Newton’s method is even more efficient than that because it requires a given multiplicity, speeding up the overall computation and allowing for a faster convergence.

With this example, we found the same root as from using the Bisection method but with 4 iteration rather than 26. In both cases, the residual based on MATLAB’s precision is 0. This method required us to find more information, such as the multiplicity and derivative of the function, but was more efficient in invoking it.

**3. Newton & Secant Method**

1. **Use Secant method to find zero around 0.9 with given parameters**

|  |  |  |
| --- | --- | --- |
| Zero Found | Residual | NOI |
| 9.718416297012561e-01 | 1.224646799147353e-16 | 7 |



1. **Use Newton method to compute zero around 0.9**

|  |  |  |
| --- | --- | --- |
| Zero Found | Residual | NOI |
| 9.060624942347658e-01 | 1.224646799147353e-16 | 4 |



1. **Discuss the two methods**

Secant method is good when you have the first iteration, so basically two guesses where the second guess is closer to the actual zero than the first. However, Newton’s method can be faster but requires more knowledge- it needs the derivative and a good initial guess. So the difference is that Newton’s requires a derivative, and Secant requires a second iteration, x1.

Both give good approximations.

**4. Jacobian & Broyden, Newton Method**

1. **Write the given nonlinear systems in vector form, f(x) = 0**
2. **Find the Jacobian**
3. **Implement f(x) and Jacobian in MATLAB**

****

1. **Approximate Q1, Q2, Q3, p by using Broyden’s method to solve the nonlinear system**

|  |  |
| --- | --- |
| Q2 | -2.299694689391697e-02 |
| P | 1.547059396846750e+05 |
| Residual | 8.769336815807861e+00 |
| Number of iterations | 200 |

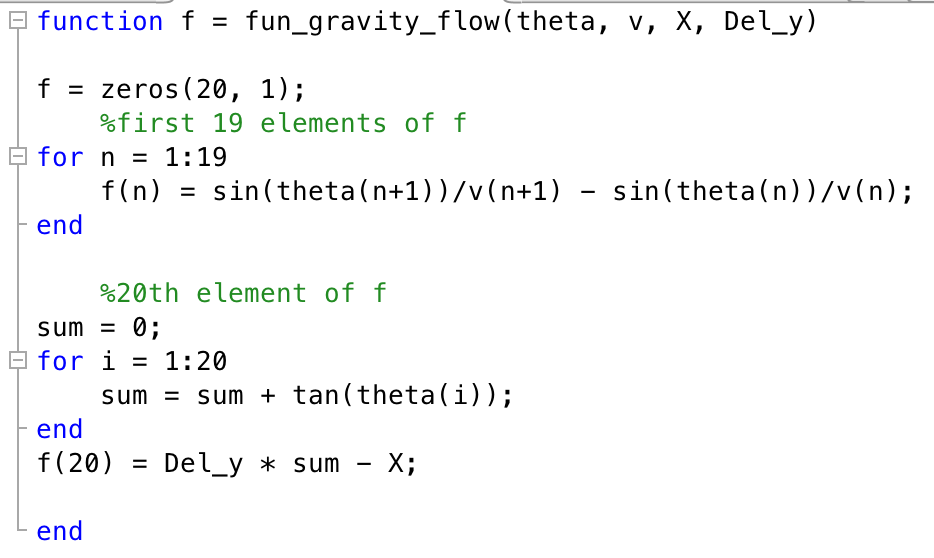
.

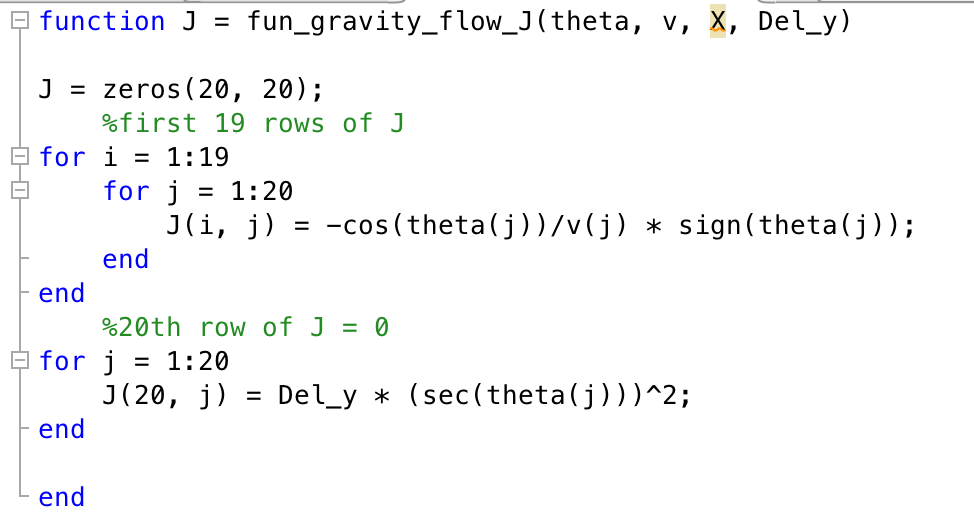
1. **Approximate Q1, Q2, Q3, p by using Newton’s method**

|  |  |
| --- | --- |
| Q1 | 1.185020865060132e+00 |
| Q3 | -8.610784052268768e-01 |
| Number of Iterations | 9 |

**5. Gravity Flow with Newton and Bryoden methods**

1. **Put nonlinear system in vector form f and implement f and its Jacobian**

****

****

1. **Compute f(theta0) and J(theta0)**

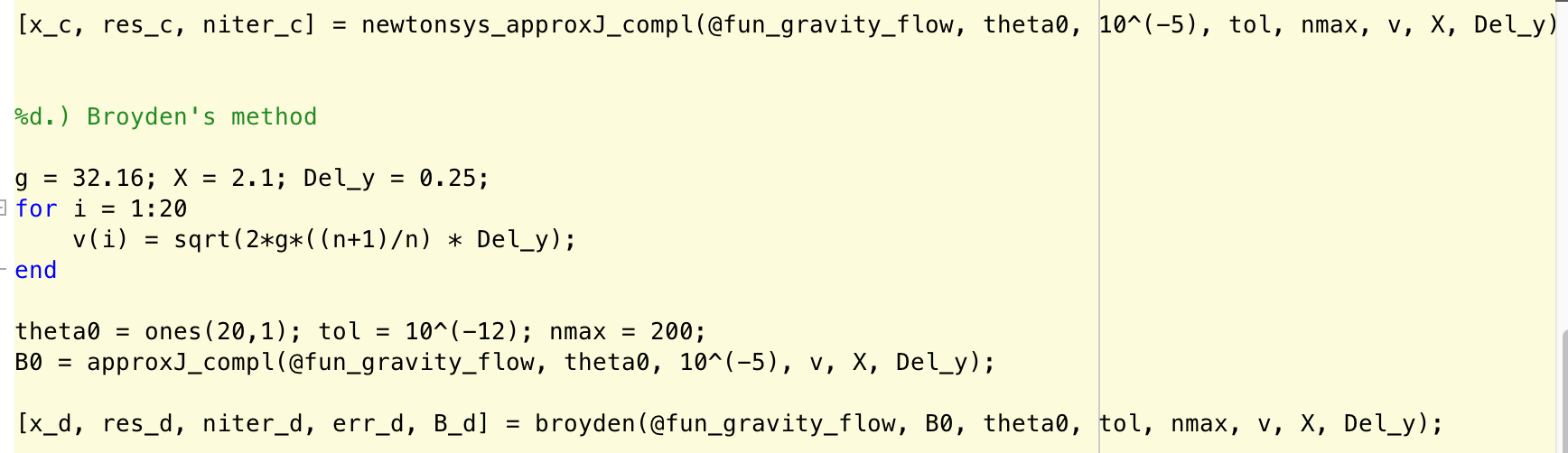
|  |  |
| --- | --- |
| F11(theta0) | 4.087754985943416e-02 |
| F20(theta0) | -4.674103686923544e+01 |
| J(13,13) | -7.017161513383326e-02 |
| J(13, 14) | -1.018907377473960e-02 |
| J(13,16) | 6.675181907251032e-02 |
| J(20,16) | 2.180759489888637e-01 |

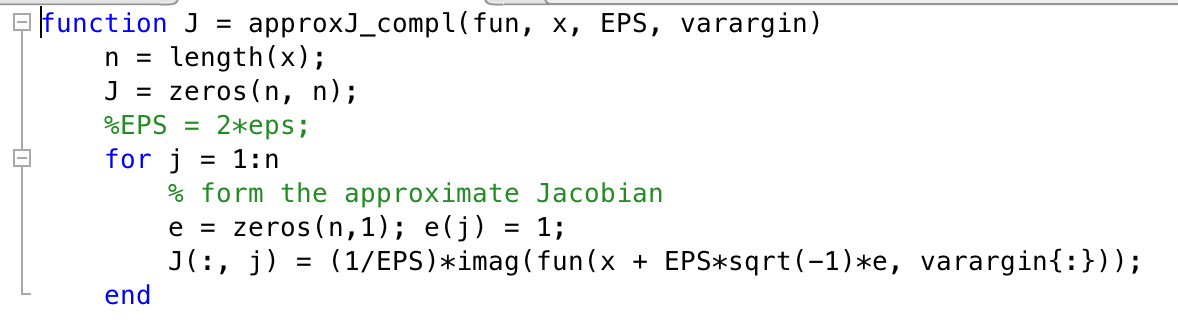
1. **Use Newton’s method**

|  |  |
| --- | --- |
| Theta3 | 2.452154874498246e-01 |
| Theta13 | 5.297949591131861e-01 |
| Theta18 | 6.368318483945207e-01 |
| Residual | 2.235573350548999e-16 |
| Number of iterations | 8 |

1. **Broyden’s method**

|  |  |
| --- | --- |
| Theta1 | 3.976279915221279e-01 |
| Theta11 | 3.976279915221293e-01 |
| Theta­17 | 3.976279915221294e-01 |
| Residual | 5.500578099284063e-16 |
| Number of Iterations | 8 |

****

****