HW# 2

Aaron Segura
Frederick Lee

September 16, 2018

1 Task 1: Clone the course repository, use and interpret *newton.py* and *drive_newton.py*

1.1 What

First, we cloned the course repository and found the two files *newton.py* and *drive_newton.py* in the subdirectory homeworks/hw2. Here, *newton.py* implements Newton's method and *drivenewton.py* will automate the process of running Newton's method on mathematical functions. We then interpreted the source files for these two programs.

1.2 How

To clone the course repository, use the following command:

```
$ git clone https://lobogit.unm.edu/jbschroder/Fall_2018_Math471.git
```

To access the homeworks/hw2 subdirectory and view the two files *drive_newton.py* and *newton.py* use:

```
$ cd homeworks/hw2
$ ls
```

To start using python and run the file drive_newton.py use:

```
$ python drive_newton.py
```

For viewing the source codes for *newton.py* and *drive_newton.py* in emacs use commands:

```
$ emacs drive_newton.py
$ emacs newton.py
```

1.3 Why

The first command above, **git clone** is used to clone a repository from a previously created repository in this case the MATH 471 course repository. The subsequent command, **cd homeworks/hw2** will change the current directory to the sub-directory **hw2** within the sub-directory **homeworks**. The **ls** command will list the files within the current sub-directory.

To start using python and running *newton.py* and *drive_newton.py* the commands **python drive_newton.py** and **python newton.py** will use the python program to run these two *.py* files. This will allow us to use the two files *newton.py* and *drive_newton.py* in implementing Newton's Method. The commands **emacs drive_newton.py** and **emacs newton.py** allow us to view the source code for these two programs. Here, we see that *newton.py* uses the scipy import command and defines the function newton with input values f and $x0$. This program defines $maxiter = 10$ as the max iterations used for determining the root using Newton's method before implementing Newton's method. The program *drive_newton.py* also uses the scipy import command and uses another command **from newton import newton** to use the *newton.py* program within the drive_newton program.

2 Task 2: Use Newton's method to find the root within tolerance and constrained by maximum number of iterations

2.1 What

The current implementation of the program above does a total of 10 iterations of Newton's method to approximate the root of the defined function regardless of how closely the method approximates the root. We must then modify the code so that the number of iterations is appropriate for a user defined tolerance of the error. This error is defined as the absolute difference between the initial estimate of the root x_0 and the next iteration of Newton's method. Subsequent errors are defined as the difference of the $n + 1$ iterate and the nth iterate with n being the number of iterates. However, a stop for the program must be put into place if the desired tolerance is not achieved in a maximum number of iterations. This will allow us to control the desired precision of Newton's method and limit the program preventing an infinite loop of the iterative method.

2.2 How

The first modification we made to *newton.py* was in adding two more parameters to the function *newton*: *maxiter* and *tol*. This change in the func-

tion signature allows the user to define values for the maximum iterations and tolerance while using the *newton* program. This was done by first defining the desired tolerance where the assignment variable *delta* holds the absolute difference between the two most recent x iterates of Newton's method. We then used a conditional statement that determines if *delta* is less than our given tolerance *tol*, then the root finding iteration process stops.

This maximum iterations constraint was implemented by limiting the number of iterations in our root finding loop by the given parameter *maxiter*. This was done by creating a control structure using a *for* loop where the range of iteration indices was between 1 and *maxiter*, preventing an infinitely running Newton's method iteration loop.

2.3 Why

Having a user defined tolerance for the approximation of an iterative process, such as Newton's method in approximating a root of a function, is important. By defining a tolerance level, we can computationally limit the number of iterations of the method such that our approximation is within an acceptable tolerance level. In the case of Newton's method, we want to exit the root finding iteration based on how close our approximation of the root of the function will be within an acceptable margin of error, or tolerance. This is done using control structures within the program by creating a stop to the program if the desired tolerance is achieved or if the maximum allowed iterations of the method has been reached.

By creating this control structure, we can then compare the number of iterations of the method required to achieve a certain tolerance for different functions. This can be done by comparing the number of iterations it takes to achieve a root approximation within a given tolerance level for each function. In other words, the user defined tolerance is achieved in a different number of iterations for each function indicating that the effectiveness of Newton's method in approximating the root is dependent on the behavior of the specific function near the root.

We can also determine which functions behave in such a way that Newton's method can apply be to and a root can be approximated. For example, if Newton's method was applied to a function such that there was an asymptotic value between the initial estimate x_0 and the root r, then the number of maximum iterations would approach infinity and the function f at the root approximation $f(x_n)$ will approach $+\infty$ or $-\infty$. Here we can clearly

see where having the program restrict the number of maximum iterations is important.

3 Task 3: Test for Linear and Quadratic Convergence

3.1 What

In this task, we will derive an equation for the calculation that we will use to compute the convergence rate factor α for Newton's method that when applied to a function $f(x)$ approximates a root. This value of α tells us how quickly Newton's method approximates a root for each iteration n. We start by including the definition of Newton's method and then a derivation of the convergence rate using Taylor's formula. We then use a ratio of the approximation the rates of convergence K_n and K_{n+1} to derive the convergence rate factor α. This will then tell us whether or not Newton's method will converge linearly or quadratically for $f(x)$.

3.2 How

Newton's method works through an iterative process to approximate a root r for a given function. This approximation of the root approaches the value of the actual root differently for different functions. In other words, the rate at which approximations of Newton's method converge to the actual root depends on the function and the function's derivative. As shown in the following equation:

$$x_{n+1} = x_n + \frac{f(x)}{f'(x)}, \qquad (n \geq 0) \tag{1}$$

where the new x_{n+1} is the value of the next approximation. To determine the convergence rates for a given function using Newton's method, we will assume the function $f(x)$ has a simple root r and is continuously differentiable such that $f'(r)$ exists and is continuous at r. Additionally, Newton's method also assumes that we have an initial approximation x_0 sufficiently close to the root of interest.

To derive an equation for the convergence rate for Newtons method, Taylor's formula can be used in the following way:

$$f(r) = f(x_n) + (r - x_n)f'(x_n) + \frac{(r - x_n)^2}{2}f''(c_n) \tag{2}$$

Here, c_n is between x_n and r. Because r is the root we have:

$$0 = f(x_n) + (r - x_n)f'(x_n) + \frac{(r - x_n)^2}{2}f''(c_n) \tag{3}$$

$$-\frac{f(x_n)}{f'(x_n)} = r - x_n + \frac{(r - x_n)^2}{2}\frac{f''(c_n)}{f'(x_n)} \tag{4}$$

Assuming that $f'(x_n) \neq 0$, then with some rearranging, we can compare the next Newton iterate with the root:

$$x_n - \frac{f(x_n)}{f'(x_n)} - r = \frac{(r - x_n)^2}{2}\frac{f''(c_n)}{f'(x_n)} \tag{5}$$

$$x_{n+1} - r = \frac{\delta_n^2}{2}\frac{f''(c_n)}{f'(x_n)} \tag{6}$$

$$\mid \delta_{n+1} \mid = \mid \delta_n \mid^2 \frac{f''(c_n)}{2f'(x_n)} \tag{7}$$

$$\lim_{n\to\infty} \frac{\mid \delta_{n+1} \mid}{\mid \delta_n \mid^2} = \frac{f''(c_n)}{2f'(x_n)} \tag{8}$$

Where $\delta_n = \mid x_n - r \mid$, since c_n lies between r and x_n, the limit converges to r as n goes to ∞. This is the case when Newton's method quadratically converges to r for a given function. It may be the case that Newton's method converges linearly, in a different manner such as between quadratically and linearly, faster than quadratically, or not at all to the root r if there is asymptotic behavior of the function between the initial estimate of the root x_0 and the actual root r.

To approximate the rate of convergence for Newton's method, we can use the following equation:

$$\frac{\mid \delta_{n+1} \mid}{\mid \delta_n \mid^\alpha} = K_n \tag{9}$$

As we increase the number of iterations of Newton's method, we can use the following equations to solve for the convergence factor α:

$$\frac{\mid \delta_{n+2} \mid}{\mid \delta_{n+1} \mid^{\alpha}} = K_{n+1} \tag{10}$$

$$K_n \approx K_{n+1}, \tag{11}$$

$$\frac{\mid \delta_{n+1} \mid}{\mid \delta_n \mid^{\alpha}} \approx \frac{\mid \delta_{n+2} \mid}{\mid \delta_{n+1} \mid^{\alpha}} \tag{12}$$

$$\frac{\mid \delta_{n+1} \mid}{\mid \delta_{n+2} \mid} \approx \frac{\mid \delta_n \mid^{\alpha}}{\mid \delta_{n+1} \mid^{\alpha}} \tag{13}$$

$$\frac{\mid \delta_{n+1} \mid}{\mid \delta_{n+2} \mid} \approx \left(\frac{\mid \delta_n \mid}{\mid \delta_{n+1} \mid} \right)^{\alpha} \tag{14}$$

Solving for α we get

$$\frac{\log \frac{|\delta_{n+1}|}{|\delta_{n+2}|}}{\log \frac{|\delta_n|}{|\delta_{n+1}|}} \approx \alpha \tag{15}$$

which tells us the magnitude of the convergence rate factor when applying Newton's method to a function. For example, if we find $\alpha = 2$ using the equation above this will tell us that Newtons method is quadratically convergent for the function f. However, if while increasing the number of iterates of Newton's method we find that $\alpha = 1$, then we can say the Newton's method converges linearly to the root r for the function f.

To compute the rate of convergence, we used the equation for α above where δ_n is the same *delta* defined in 2.2 as the absolute difference between the two most recent iterates of x using Newton's method. Here we show α with respect to n where as n increases α approaches a constant rate of convergence:

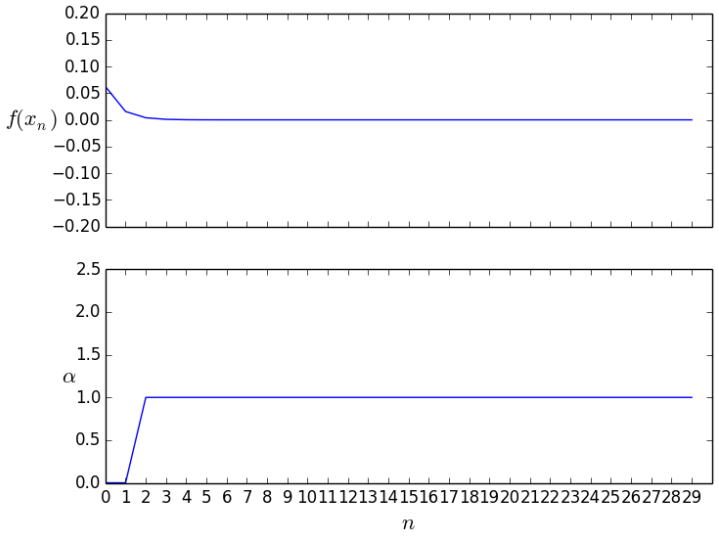

Figure 1: $f(x) = x^2$ and α with respect to n

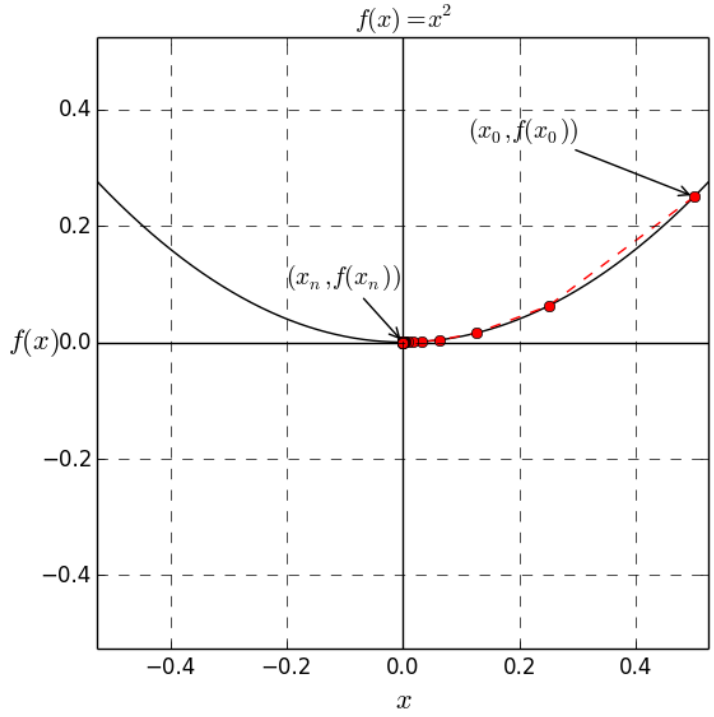

Figure 2: $f(x) = x^2$ with iterated root path

n	x_n	δ_n	$K_{(1)n}$	$K_{(2)n}$	$f(x_n)$	α_n
1	2.5000e-01	2.5000e-01	0.0000e+00	0.0000e+00	6.2500e-02	0.0000e+00
2	1.2500e-01	1.2500e-01	5.0000e-01	2.0000e+00	1.5625e-02	0.0000e+00
3	6.2500e-02	6.2500e-02	5.0000e-01	4.0000e+00	3.9062e-03	1.0000e+00
4	3.1250e-02	3.1250e-02	5.0000e-01	8.0000e+00	9.7656e-04	1.0000e+00
5	1.5625e-02	1.5625e-02	5.0000e-01	1.6000e+01	2.4414e-04	1.0000e+00
6	7.8125e-03	7.8125e-03	5.0000e-01	3.2000e+01	6.1035e-05	1.0000e+00
7	3.9062e-03	3.9062e-03	5.0000e-01	6.4000e+01	1.5259e-05	1.0000e+00
8	1.9531e-03	1.9531e-03	5.0000e-01	1.2800e+02	3.8147e-06	1.0000e+00
9	9.7656e-04	9.7656e-04	5.0000e-01	2.5600e+02	9.5367e-07	1.0000e+00
10	4.8828e-04	4.8828e-04	5.0000e-01	5.1200e+02	2.3842e-07	1.0000e+00
11	2.4414e-04	2.4414e-04	5.0000e-01	1.0240e+03	5.9605e-08	1.0000e+00
12	1.2207e-04	1.2207e-04	5.0000e-01	2.0480e+03	1.4901e-08	1.0000e+00
13	6.1035e-05	6.1035e-05	5.0000e-01	4.0960e+03	3.7253e-09	1.0000e+00
14	3.0518e-05	3.0518e-05	5.0000e-01	8.1920e+03	9.3132e-10	1.0000e+00
15	1.5259e-05	1.5259e-05	5.0000e-01	1.6384e+04	2.3283e-10	1.0000e+00
16	7.6294e-06	7.6294e-06	5.0000e-01	3.2768e+04	5.8208e-11	1.0000e+00
17	3.8147e-06	3.8147e-06	5.0000e-01	6.5536e+04	1.4552e-11	1.0000e+00
18	1.9073e-06	1.9073e-06	5.0000e-01	1.3107e+05	3.6380e-12	1.0000e+00
19	9.5367e-07	9.5367e-07	5.0000e-01	2.6214e+05	9.0949e-13	1.0000e+00
20	4.7684e-07	4.7684e-07	5.0000e-01	5.2429e+05	2.2737e-13	1.0000e+00
21	2.3842e-07	2.3842e-07	5.0000e-01	1.0486e+06	5.6843e-14	1.0000e+00
22	1.1921e-07	1.1921e-07	5.0000e-01	2.0972e+06	1.4211e-14	1.0000e+00
23	5.9605e-08	5.9605e-08	5.0000e-01	4.1943e+06	3.5527e-15	1.0000e+00
24	2.9802e-08	2.9802e-08	5.0000e-01	8.3886e+06	8.8818e-16	1.0000e+00
25	1.4901e-08	1.4901e-08	5.0000e-01	1.6777e+07	2.2204e-16	1.0000e+00
26	7.4506e-09	7.4506e-09	5.0000e-01	3.3554e+07	5.5511e-17	1.0000e+00
27	3.7253e-09	3.7253e-09	5.0000e-01	6.7109e+07	1.3878e-17	1.0000e+00
28	1.8626e-09	1.8626e-09	5.0000e-01	1.3422e+08	3.4694e-18	1.0000e+00
29	9.3132e-10	9.3132e-10	5.0000e-01	2.6844e+08	8.6736e-19	1.0000e+00
30	4.6566e-10	4.6566e-10	5.0000e-01	5.3687e+08	2.1684e-19	1.0000e+00

Table 1: Convergence Data for Newton's method applied to $f(x) = x^2$

3.3 Why

In Figure 1 we see that as the number of iterates of Newton's method increases, the function at the iterate $f(x_n)$ approaches 0. This is in agreement with what we see in Figure 2, that as the iterate of Newton's method increases we approach the root of the function r. We also see in Figure 1 and Table 1, that as the number of iterates increases α approaches 1. We can then conclude that Newton's method converges linearly for the function $f(x) = x^2$, since α becomes the constant value of $\alpha = 1$.

This means that after each iteration of Newton's method we decrease our δ_n or absolute difference in iterations of x_n by a linear factor which is shown by $K_{(1)n} = \frac{1}{2}$. This is due to the fact that the derivative of the function $f'(x) \approx 0$ near the root r. So when we apply the next iteration of Newton's method, the new approximation x_{n+1} generated will change by a linear factor $\frac{1}{2}$ near the root because the magnitude of $f'(x_n)$ approaches 0. In conclusion, the next approximation of the root will linearly converge to the root r iteration of Newton's method as we get closer to r.

4 Task 4: Apply Newton's method to two more functions

4.1 What

We will now apply Newton's method to the two functions below and determine the convergence rate for each function:

$$f(x) = x \tag{16}$$

$$f(x) = \sin x + \cos x^2 \tag{17}$$

If for the calculation of α we find that α approaches 1 or 2, then we can conclude that the Newton's method converges linearly or quadratically to the root r for the function f, respectively.

4.2 How

Extending the application of Newton's method to the functions above involves two tasks: first coding in Python for both (16) and (17), a function that takes the same form with respect to input and output types as the function used previously, and secondly adding the function to the list *fcn_list* in *drive_newton.py*.

In *drive_newton.py*, the first function takes the form:

$$f_1(x, d) = f_1^{(d)}(x^2) \tag{18}$$

where d is the dth derivative of the function f_1 evaluated at x. The two other functions took the form of:

$$f_2(x, d) = f_2^{(d)}(x) \tag{19}$$

and

$$f_3(x, d) = f_3^{(d)}(\sin x + \cos x^2) \tag{20}$$

The first derivatives, $d = 1$, of each of these functions were explicitly defined within *drive_newton.py* as we did not anticipate on using any higher order derivatives while implementing newton's method and evaluating convergence rates.

We added each of these new functions to *fcn_list* with the same list type structure used for (18).

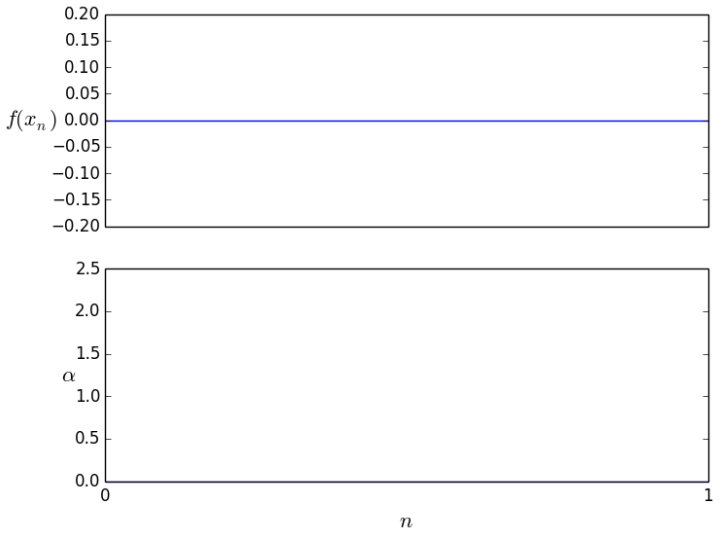

Figure 3: $f(x) = x$ and α with respect to n

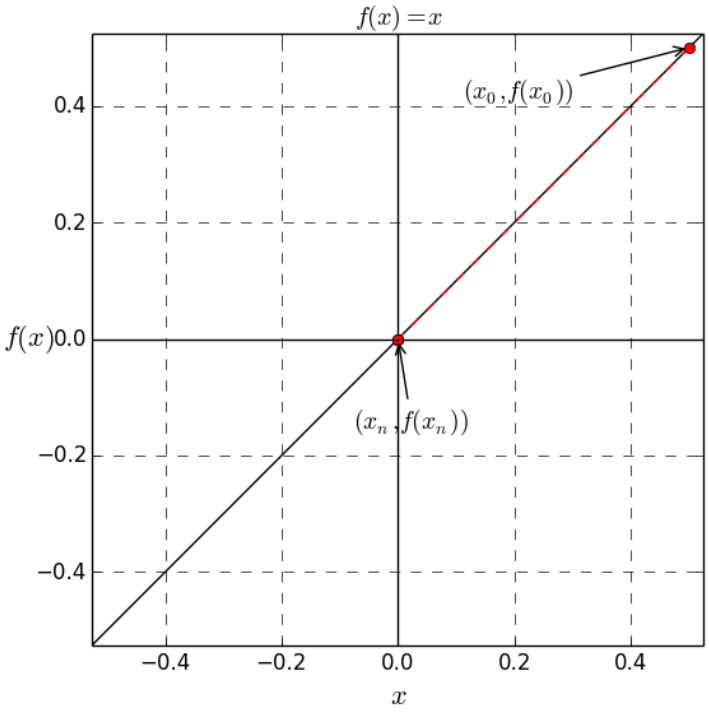

Figure 4: $f(x) = x$ with iterated root path

n	x_n	δ_n	$K_{(1)n}$	$K_{(2)n}$	$f(x_n)$	α_n
1	0.0000e+00	5.0000e-01	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
2	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00

Table 2: Convergence Data for Newton's method applied to function (19) *Our initial guess $x_0 = 0.5$ was omitted from the data table.

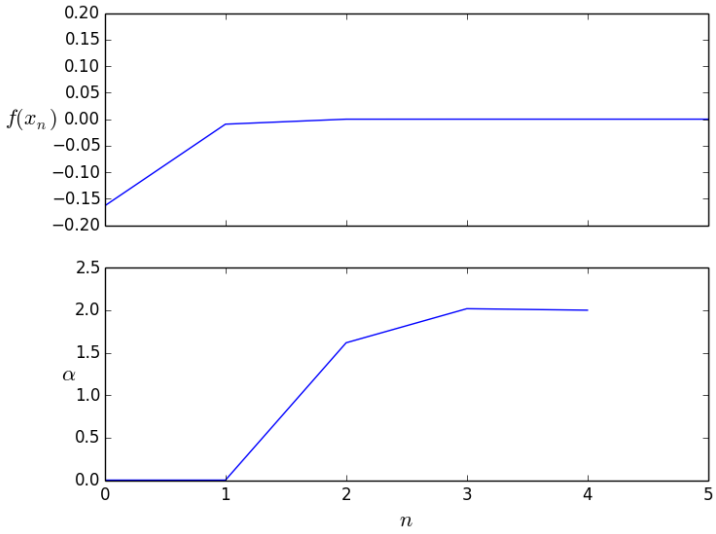

Figure 5: $f(x) = \sin x + \cos x^2$ and α with respect to n

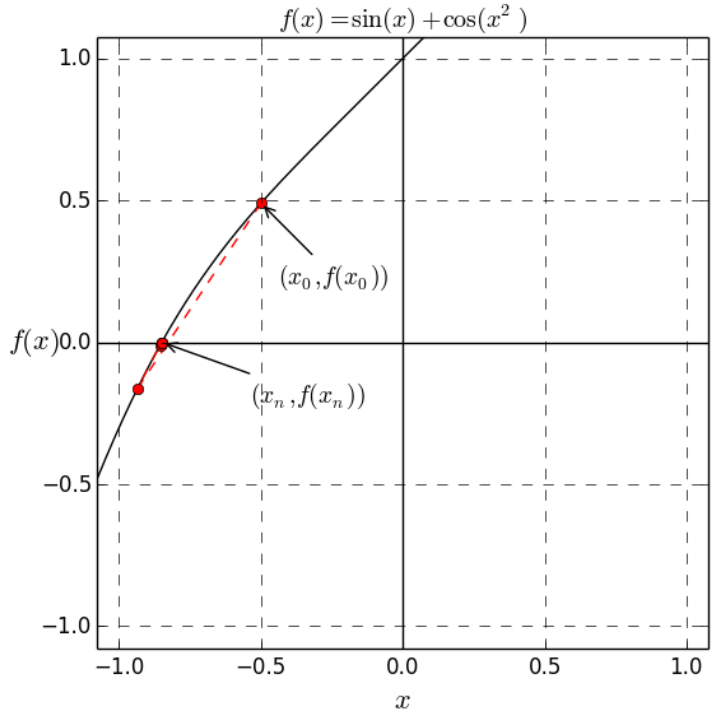

Figure 6: $f(x) = \sin x + \cos x^2$ with iterated root path

n	x_n	δ_n	$K_{(1)n}$	$K_{(2)n}$	$f(x_n)$	α_n
1	-9.3510e-01	4.3510e-01	0.0000e+00	0.0000e+00	-1.6322e-01	0.0000e+00
2	-8.5464e-01	8.0463e-02	1.8493e-01	4.2502e-01	-9.4363e-03	0.0000e+00
3	-8.4939e-01	5.2515e-03	6.5265e-02	8.1112e-01	-3.7918e-05	1.6171e+00
4	-8.4937e-01	2.1273e-05	4.0509e-03	7.7138e-01	-6.1928e-10	2.0184e+00
5	-8.4937e-01	3.4744e-10	1.6332e-05	7.6775e-01	0.0000e+00	2.0009e+00
6	-8.4937e-01	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	inf

Table 3: Convergence Data for Newton's method applied to function (20)

4.3 Why

For function (19), the convergence is immediate due to the functionality of Newton's method. Given any root guess, the first iterated root will converge on the solution $x = 0$. This also explains why the only non-zero entry in the data table for this function is the absolute difference value for the first iterate. There is no other value in the first row has enough information to be calculated given a single iteration. Additionally, the fact that there are two iterations in the data table is an artifact of our *newton.py* implementation that limits iterations by checking if the absolute difference between the two most recent x iterates is within the tolerance 10^{-10}. This condition is only satisfied after a second iteration has been calculated.

For function (20), we decided to use an initial root estimate of $x_0 = -0.5$ since Newton's method assumes that the value of the initial estimate is sufficiently close to the actual root r. If we did not select an initial estimate close to the root of interest, then when applying Newton's method we would converge to a root that is not local. Since the function (20) is a function with periodic behavior around the x-axis the multiplicity of the root m is greater than 1. With regards to convergence using Newton's method, the first α value is greater than 1 and approaches 2 thereafter which means that the method converges faster than linearly. We obtained an erroneous infinity value for α in the last iteration of Newton's method because the expression used to calculate α does not take consider underflow values for δ. However, we can still conclude that Newton's method approaches the root faster than linearly since $\alpha > 1$. Using the quadratic convergence values $K_{(2)n}$, contained within the 4th data column, we observe values near .77, which suggests that .77 is approximately our quadratic convergence factor. Providing further indication that Newton's method is more than linearly convergent.

It is possible to use a modified Newton's method to recover quadratic

convergence if the multiplicity is known for a given function by using the following method $x_{n+1} = x_n - m\frac{f(x)}{f'(x)}$, where m is the multiplicity of the root. Functions such as these take the form of $f(x) = (x-r)^m g(x)$ where $g(x) \neq 0$ and m is the multiplicity of the root. In the case that the function $f(x) = x^2$ we have linear convergence but if we were to use a modified Newton's method above for each iteration, then we would recover quadratic convergence using $m = 2$.

5 Task 5: Summary of Findings

5.1 What

We are asked to automate the process of root finding using Newton's method and apply this automated process to three different single variable functions. Also, we were tasked with analyzing the data that was the output as a result of this automated process. In particular, we discussed the convergence rates for each application of Newton's method on these functions.

5.2 How

Through modifying the program provided to us in the course materials, we found that in using Newton's method can be used to approximate the root of a function through an iterative process. To do this we must use the driver program *drive_newton.py* to run the *newton.py* program. We then applied Newton's method in this manner the function $f(x) = x^2$ and to two new functions, then determined their rates of convergence. The rates of convergence were determined by taking the ratio of the absolute differences of the iterates or δ's generated by Newton's method to obtain K_1 and K_2 values. We then applied logarithmic functions to determine a convergence rate factor approximation α which would give us an approximation of the convergence rate where $\alpha = 1$ and $\alpha = 2$ would indicate linear and quadratic convergence, respectively.

For each function above, we created plots of the function evaluated at x_n or $f(x_n)$ and plots of the convergence rate factor α with respect to x_n. We also plotted the function $f(x)$ itself including the $f(x_n)$ values for each iterate n. Then, we tabulated values for the following parameters used or obtained in generating our plots, n, x_n, δ_n, $K_{(1)n}$, $K_{(2)n}$, $f(x_n)$, and α_n. Finally, we

interpreted our results and commented on usage of Newton's method and its shortcomings.

5.3 Why

The purpose of this homework assignment was to further familiarize ourselves with managing computations through scripting and automation. Here, we used an iterative process known as Newton's method to automate computations to find the roots of a function which approximate solutions to the equation. Things that could be touched upon further would be to go into detail about the limitations of this method and how this method breaks down for functions where the derivatives do not exist, are not continuous, or behaves unexpectedly.