HW2

Deep Patel
James Hurst

September 17, 2018

1 What

The overall goal of this homework was to further familiarize with managing computations through scripting and automation. Here, we automate computations to find the roots of a function f(x), i.e., approximate solutions to the equation f(x) = 0. We will consider three different functions: $f(x) = x$, $f(x) = x^2$, and $f(x) = sin(x) + cos(x^2)$. We will use Newtons method to find the roots of the funcitons:

$$x_{n+1} = x_n + \frac{f(x_n)}{f'(x_n)}$$

2 How

There are two files: *newton.py* and $drive-newton.py$. Where, *newton.py* implements a general Netwons method, and then $drive - newton.py$ automates the process of running Newtons method on multiple functions.

1. Task 2: The *newton.py* and $drive - newton.py$ were modified so that it iterated until the quantity

   $$(\delta_{abs}) = |x_{n+1} - x_n|$$

   is less than the user defined tolerance. The code is shown in Figure 1 and 2.

2. Task 3: The code was modified to test for linear versus quadratic convergence by finding K1 and K2 as defined below

   $$K1 = \frac{|x_{n+1} - x_n|}{|x_n - x_{n-1|}}$$

   $$K2 = \frac{|x_{n+1} - x_n|}{(|x_n - x_{n-1}|)^2}$$

   We expect the K1=constant for linear convergence and K2=constant for quadratic convergence. The code for calculating K1 and K2 is implemented as shown in Figure 3.

3. The code was modified in $drive - newton.py$ to include two more functions, $f(x) = x$ and $f(x) = sin(x) + cos(x^2)$ as shown in Figure 4. The

data was printed measure the convergence rate for all three functions as shown in Tables 1,2, and 3: the first column is the root guess and second column is the quantity

$$(\delta_{abs}) = |x_{n+1} - x_n|$$

3 Why

1. The K value was not calculated for the linear function $f(x) = x$ because the newton's method finds the root of the function in one try. This is because, when a tangent is drawn to the linear function, the intersection is the exact root! This can be observed in 2: the first column is the root guess and second column is the quantity

$$(\delta_{abs}) = |x_{n+1} - x_n|$$

2. The code for calculating K1 and K2 is implemented as shown in Figure 3. The values of K1 and K2 obtained for the function $f(x) = x^2$ are shown in Table 4 and 5. We observe that the values of K1 are constant and those of K2 are not. This leads us to the conclusion that the function $f(x) = x^2$ is has **linear** rather convergence.

3. The values of K1 and K2 obtained for the function $f(x) = sin(x) + cos(x^2)$ are shown in Table 6 and 7. We observe that neither the values of K1 nor K2 are constant. This leads us to the conclusion that this trigonometric function has neither linear nor quadratic convergence. However, comparing the values of K1 and K2 we observe that the values of K1 differ by orders of magnitude, whereas values of K2 differ slightly.

4. Task 4: It is possible to come up with a modified Newtons method to regain the quadratic convergence for the linearly converging function. If the multiplicity m of the root is known in advance, it is possible to modify Newtons method and recover quadratic convergence. This can be done by multiplying the derivative ratio term in Newton's method by the multiplicity (m) as shown below:

$$x_{n+1} = x_n + \frac{mf(x_n)}{f'(x_n)}$$

```
# Insert some logic for halting, if the tolerance is met
if e < tol:
   break
```

Figure 1: This figure shows the modified *newton.py* with added tolerance criteria

```
for fcn in fcn_list:
    data = newton(fcn, x0, (10**-10), 50)
```

Figure 2: This figure shows the $drive - newton.py$ showing input of tolerance of $10^{-10)}$

```python
def calculate_k1(data):
    k1_values = []
    for i in range(0, len(data)-2):
        err1 = abs(data[i+2] - data[i+1])
        err2 = abs(data[i+1] - data[i])
        k1_values.append(err1 / err2)
    return k1_values

def calculate_k2(data):
    k2_values = []
    for i in range(0, len(data)-2):
        err1 = abs(data[i+2] - data[i+1])
        err2 = abs(data[i+1] - data[i])**2
        k2_values.append(err1 / err2)
    return k2_values
```

Figure 3: This figure displays the code for calculating the K1 and K2

```
def f1(x, d):
    if d == 0:
        return x**2
    elif d == 1:
        return 2*x

def f2(x, d):
    if d == 0:
        return x
    elif d == 1:
        return 1

def f3(x, d):
    if d == 0:
        return sin(x) + cos(x**2)
    elif d == 1:
        return cos(x) - 2*x*sin(x**2)
```

Figure 4: This figure displays the added functions

-2.50000e-01	2.50000e-01
-1.25000e-01	1.25000e-01
-6.25000e-02	6.25000e-02
-3.12500e-02	3.12500e-02
-1.56250e-02	1.56250e-02
-7.81250e-03	7.81250e-03
-3.90625e-03	3.90625e-03
-1.95312e-03	1.95312e-03
-9.76562e-04	9.76562e-04
-4.88281e-04	4.88281e-04
-2.44141e-04	2.44141e-04
-1.22070e-04	1.22070e-04
-6.10352e-05	6.10352e-05
-3.05176e-05	3.05176e-05
-1.52588e-05	1.52588e-05
-7.62939e-06	7.62939e-06
-3.81470e-06	3.81470e-06
-1.90735e-06	1.90735e-06
-9.53674e-07	9.53674e-07
-4.76837e-07	4.76837e-07
-2.38419e-07	2.38419e-07
-1.19209e-07	1.19209e-07
-5.96046e-08	5.96046e-08
-2.98023e-08	2.98023e-08
-1.49012e-08	1.49012e-08
-7.45058e-09	7.45058e-09
-3.72529e-09	3.72529e-09
-1.86265e-09	1.86265e-09
-9.31323e-10	9.31323e-10
-4.65661e-10	4.65661e-10
-2.32831e-10	2.32831e-10
-1.16415e-10	1.16415e-10
-5.82077e-11	5.82077e-11

Table 1: Data output for $f(x) = x^2$: the first column is the root guess and second column is the quantity $(\delta_{abs}) = |x_{n+1} - x_n|$.

0.00000e+00	5.00000e-01
0.00000e+00	0.00000e+00

Table 2: Data output for $f(x) = x$: the first column is the root guess and second column is the quantity $(\delta_{abs}) = |x_{n+1} - x_n|$.

-9.35105e-01	4.35105e-01
-8.54642e-01	8.04631e-02
-8.49390e-01	5.25146e-03
-8.49369e-01	2.12731e-05
-8.49369e-01	3.47440e-10
-8.49369e-01	1.11022e-16

Table 3: Data output for $f(x) = sin(x) + cos(x^2)$: the first column is the root guess and second column is the quantity $(\delta_{abs}) = |x_{n+1} - x_n|$.

5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01
5.00000e-01

Table 4: K1 for $f(x) = x^2$ showing linear convergence.

4.00000e+00
8.00000e+00
1.60000e+01
3.20000e+01
6.40000e+01
1.28000e+02
2.56000e+02
5.12000e+02
1.02400e+03
2.04800e+03
4.09600e+03
8.19200e+03
1.63840e+04
3.27680e+04
6.55360e+04
1.31072e+05
2.62144e+05
5.24288e+05
1.04858e+06
2.09715e+06
4.19430e+06
8.38861e+06
1.67772e+07
3.35544e+07
6.71089e+07
1.34218e+08
2.68435e+08
5.36871e+08
1.07374e+09
2.14748e+09
4.29497e+09

Table 5: K2 for $f(x) = x^2$ not showing quadratic convergence.

6.52655e-02
4.05088e-03
1.63324e-05
3.19544e-07

Table 6: K1 for $f(x) = sin(x) + cos(x^2)$ not showing linear convergence.

8.11123e-01
7.71383e-01
7.67751e-01
9.19708e+02

Table 7: K2 for $f(x) = sin(x) + cos(x^2)$ not showing linear convergence.