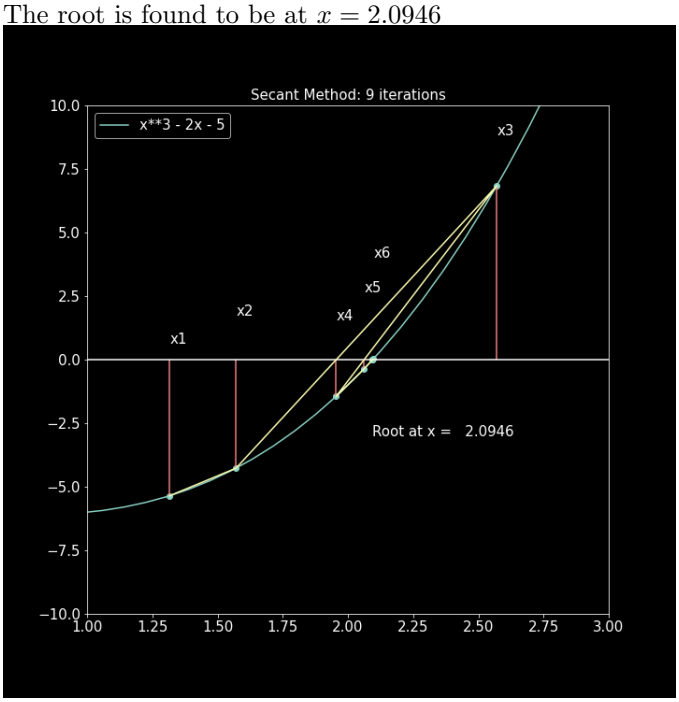
Numerical Methods Homework 4

Adam Kit

10 May 2020

1 Secant Method

The root is found to be at $x = 2.0946$



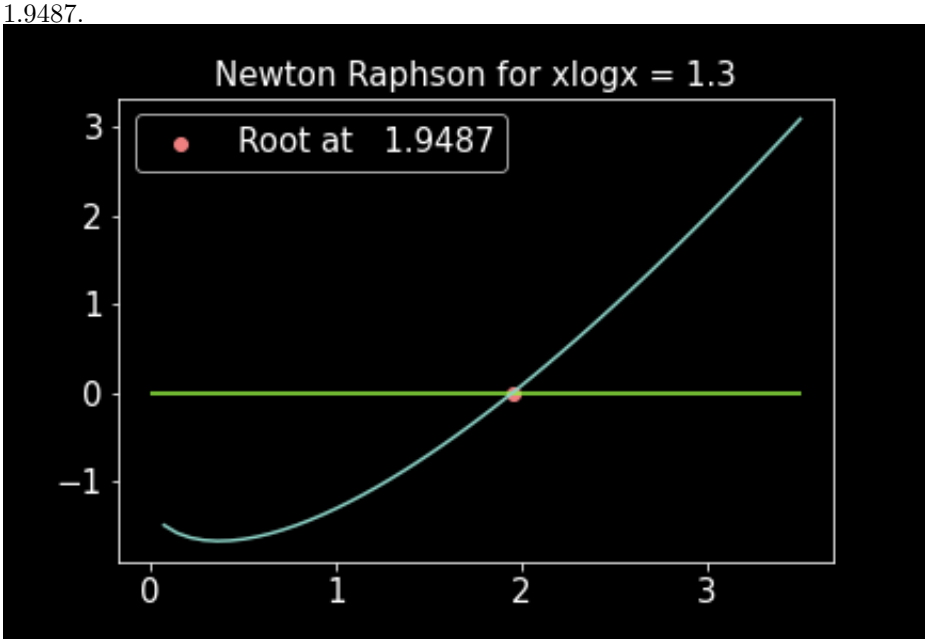
The code for both of these problems can be found here, where *secant-method.py* is for Problem 1, and *newtmethod.py* is for problems 2 and 3. The code is also pasted down below where 1 is for the secant method and 2 is for the newtons method.

```python
import numpy as np
def secant_method(f, x0, x1, max_iter=100, tolerance = 1e-5):
    steps_taken = 1
    iter_x, iter_y, iter_count = np.empty(0),np.empty(0),np.empty(0)
    i = 0

    while steps_taken < max_iter and abs(x1-x0) > tolerance:
        i +=1
        x2 = x1 - ( (f(x1) * (x1 - x0)) / (f(x1) - f(x0)) )
        iter_x = np.append(iter_x,x2)
        iter_y = np.append(iter_y,f(x2))
        iter_count = np.append(iter_count ,i)

        x1, x0 = x2, x1
        steps_taken += 1
    return x2, iter_x, iter_y, iter_count
f = lambda x: x**3 - 2*x - 5
root, iter_x, iter_y, iter_count = secant_method(f, 1, 4) 2.094551481544698
```

Figure 1: The Secant Method in Python

2 Newton Raphson Method

For the first Newton method problem I find the root for the equation $xlog_{10}x = 1.3$ to be when $x = 1.9487$.



and for the second, I find that $\frac{1}{\sqrt{15}} \approx 0.2583$.

```python

def derivative_approx(f, x, h=.00000001):
    return (f(x+h) - f(x)) / h

def newton_raphson(f, x, tolerance=.001):
    steps_taken = 0
    tangentline = []
    x_values = []
    while abs(f(x)) > tolerance:
        x_values.append(x)
        df = derivative_approx(f, x)
        x = x - f(x)/df
        steps_taken += 1
        tangentline.append(df)
    return x, steps_taken, tangentline, x_values

g = lambda y: y*np.log(y) -1.3 #Problem 2
root, steps, tangentline, tangentx = newton_raphson(g, 4) 1.948654671234792
p = lambda x: x**2 - 1/15.0 #Problem 3
root2, steps2, tangentline2, tangentx2 = newton_raphson(p, 2) 0.2583261392181133

```

Figure 2: The Newton Raphson Method in Python