**CSCI 3656: Numerical Computation Pourna Sengupta**

Homework 3 September 25, 2020

Python Code:

"""

Created on Tue Oct 20 01:22:19 2020

@author: pournasengupta

used http://hplgit.github.io/Programming-for-Computations/pub/p4c/.\_p4c-solarized-Python031.html

https://en.wikibooks.org/wiki/Python\_Programming/Basic\_Math

to help write code

"""

import sys

import math

import numpy

import matplotlib.pyplot as plt

"Define Newton's Method Function"

"f = intial function f(x) = 0"

"df = derivative of f(x) f'(x)"

"p0 = given initial approximation p\_0"

"tol = tolerance -> stopping criteria for |f(x) < tol|"

"n = max number of iterations of newton's method n "

def newtonsMethod(f, df, p0, tol, n):

"create empty arrays to store values"

astore = []

bstore = []

"new variable fx to represent f(x) at each iteration of p"

"f(p0) = f(p\_0) initial approximation value"

"set f(x) = f(p\_0)"

fx = f(p0)

"new variable i iteration counter"

i = 0

"RUN NEWTON'S METHOD FOR N ITERATIONS"

"while loop with 2 conditions"

"|fNorm| > tol and i < n"

"fNorm must be greater than the tolerance"

"iteration must be less than max iterations"

while abs(fx) > tol and i < n:

"solve p\_i = p\_o - (f(p\_0)/f'(p\_0))"

"set p\_i as new value"

"p0 = p0 - float(fx)/df(p0)"

try:

p0 = p0 - float(fx)/(df(p0))

except ZeroDivisionError:

print("Zero Derivative for x = %f", p0)

sys.exit("Solution not found")

"update value of fx"

"fx = f(p\_0)"

fx = f(p0)

"append values to arrays"

astore.append(p0)

bstore.append(fx)

"update iterator"

i +=1

if abs(fx) > tol:

"update iterator if solution is found"

"or max number of iterations reached"

i = -1

return p0, i, astore, bstore

def f(p0):

return (1)/(1 + math.exp(p0)) - 1/2

def df(p0):

return -(math.exp(p0))/(1 + math.exp(p0))\*\*2

solution, zeroI, avalues, bvalues = newtonsMethod(f, df, p0 = 0.25, tol=1.0e-14, n=100)

"if the solution is found"

if zeroI > 0:

print ("Number of iterations: %d" % (1+2\*zeroI))

print ("Solution: %f" %(solution))

else:

print("Solution not found")

a = numpy.array([i for i in avalues])

b = numpy.array(bvalues)

fig = plt.figure()

plt.plot(a, b, label = 'Newtons Method')

plt.legend()

def f1(x):

return numpy.int((1)/(1 + math.exp(x)) - 1/2)

def derive(x):

h = 0.000000001

return (f1(x+h) - f1(x)/h)

def tanLine():

x = numpy.linspace(-5, 5, 100)

y = x\*\*2 + 2;

plt.plot(x, y, 'b-', 'LineWidth', 2);

plt.grid(True)

xTangent = -4.5;

slope = 2 \* xTangent;

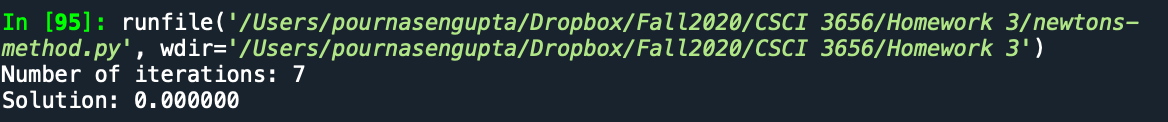
yTangent = xTangent\*\*2 + 2;

plt.plot(xTangent, yTangent, 'r\*', 'LineWidth', 2, 'MarkerSize', 10);

yTangentLine = slope \* (x - xTangent) + yTangent;

plt.plot(x, yTangentLine, 'b-', 'LineWidth', 2);

Code Output (graphs attached to questions 2 and 4):



Graphs were being funky, so I eventually gave up. I understand the concept but am having trouble coding it, so I drew it out to show that I understood the concept.

Question 1:

A picture containing text

Description automatically generated

Question 2:

Text, letter

Description automatically generated

Chart, line chart

Description automatically generated

Diagram

Description automatically generatedTable

Description automatically generatedQuestion 3:

Chart

Description automatically generated

Question 4:

A picture containing text

Description automatically generatedChart

Description automatically generated

Bonus Question 2:

To find the convergence of f’(x), I used both the root and ratio tests. Both showed convergence, as the limit L was less than 1. The ratio test showed that the interval [-0.625, 0) and (0, 0.625] converged. My interval is much larger than the interval found through the ratio test but from my graphs, it makes sense that the initial guesses converge within such small intervals.