Bradley Green

Math 4610

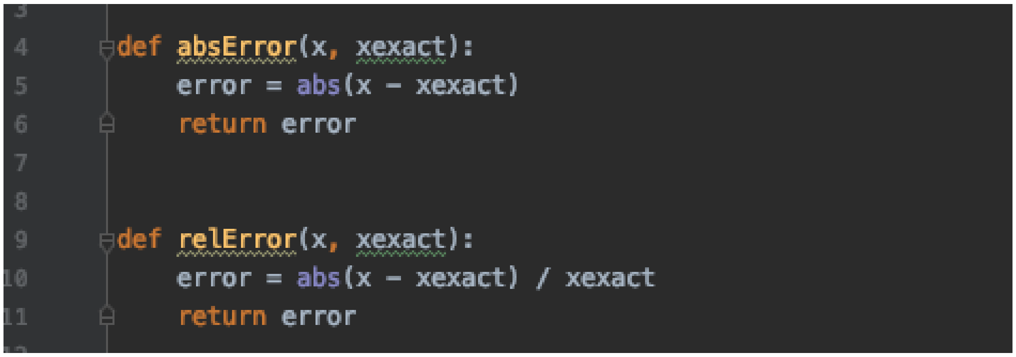
Oct. 4, 2021

Dr. Koebbe

TaskSheet 4

Task 1:

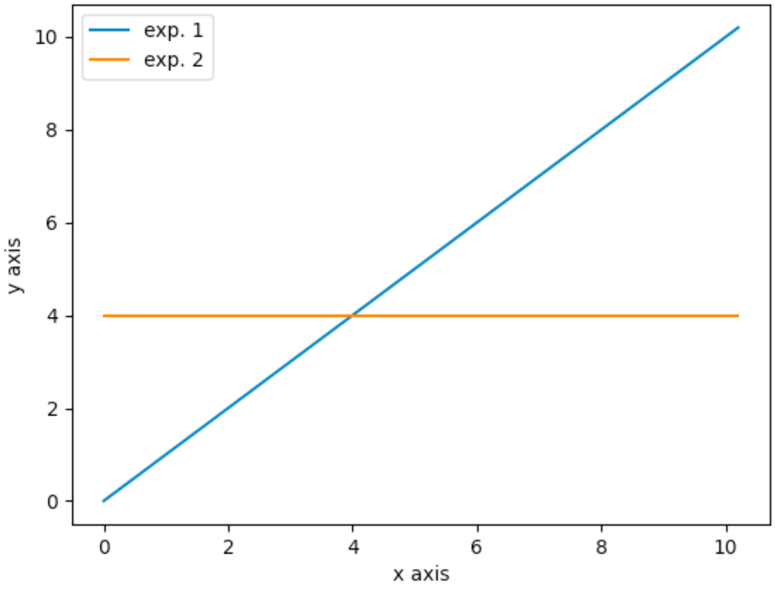
I wrote the code for relative and absolute error in python, the following is my code. I then added the routines to my software manual on github.



```python
def absError(x, xexact):
    error = abs(x - xexact)
    return error


def relError(x, xexact):
    error = abs(x - xexact) / xexact
    return error
```

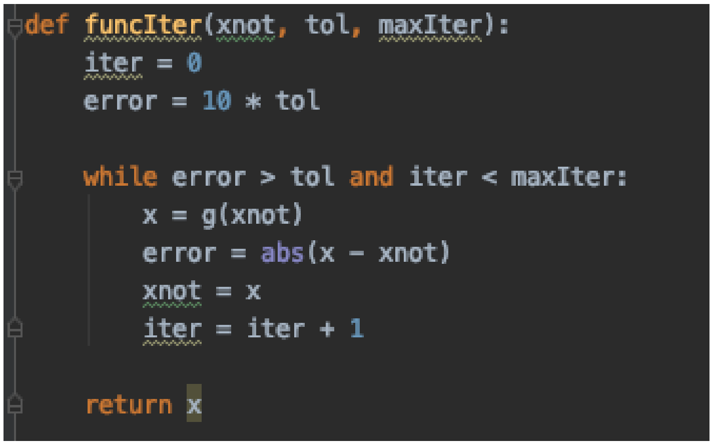
Task 2:

The following is a plot from my graphics code that lets the user input string expressions to plot.



Task 3:

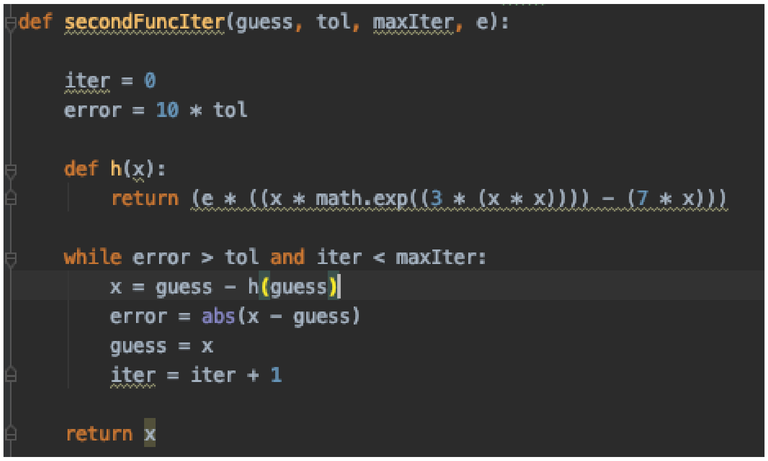
The following code is the code I wrote for fixed point iteration algorithm of finding a root. It takes a guess of the root, the tolerance, and a max number of Iterations.



```
def funcIter(xnot, tol, maxIter):
    iter = 0
    error = 10 * tol

    while error > tol and iter < maxIter:
        x = g(xnot)
        error = abs(x - xnot)
        xnot = x
        iter = iter + 1

    return x
```

Task 4:

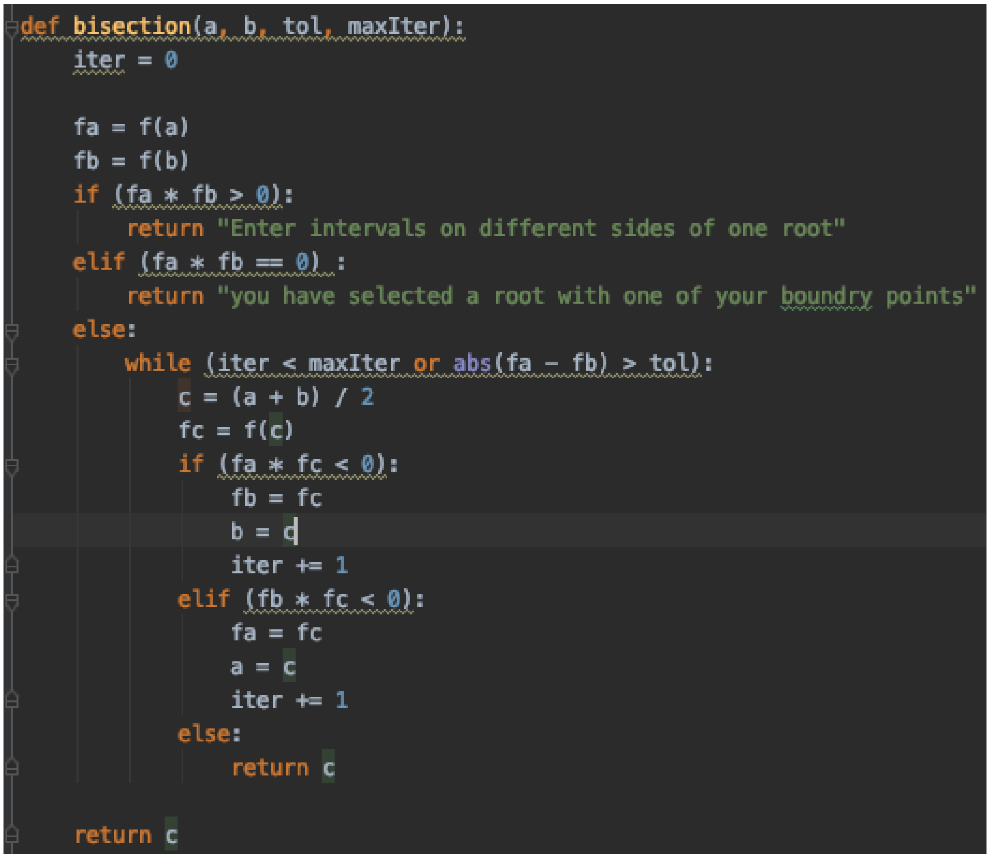
The following code is the code I wrote for fixed point iteration of the exponential function. It takes a guess of the root, the tolerance, max iterations, and a constant to reduce the function.



```
def secondFuncIter(guess, tol, maxIter, e):

    iter = 0
    error = 10 * tol

    def h(x):
        return (e * ((x * math.exp((3 * (x * x)))) - (7 * x)))

    while error > tol and iter < maxIter:
        x = guess - h(guess)
        error = abs(x - guess)
        guess = x
        iter = iter + 1

    return x
```

Task 5:

The following routine is the bisection method for finding a root. Given a function f that defines the expression in python, the routine takes a guess above and below the route, a and b. It also takes a tolerance and a max number of iterations before returning its approximation of the root.



```
def bisection(a, b, tol, maxIter):
    iter = 0

    fa = f(a)
    fb = f(b)
    if (fa * fb > 0):
        return "Enter intervals on different sides of one root"
    elif (fa * fb == 0) :
        return "you have selected a root with one of your boundry points"
    else:
        while (iter < maxIter or abs(fa - fb) > tol):
            c = (a + b) / 2
            fc = f(c)
            if (fa * fc < 0):
                fb = fc
                b = c
                iter += 1
            elif (fb * fc < 0):
                fa = fc
                a = c
                iter += 1
            else:
                return c

    return c
```

Task 6:

Sometimes when trying to find the root of an equation, it cannot be done analytically so it has to be done computationally. This is when our root finding methods are useful for approximating the root in a problem where we may be looking to minimize the area of an object. However, root finding methods have their flaws, as with the bisection method, you need to have exactly one root with in your chosen interval or else the method will fail.

https://sam.nitk.ac.in/courses/MA608/root_finding_problems.pdf

https://www.reed.edu/physics/courses/P200.L.S11/Physics200Lab/files/Bisection.pdf