Project 3B: Finding Roots (open methods)

Due Dates

Week 6, Monday at 4:59

Your submission should be a single zip file containing all the .py files containing your code as well as a Single typed PDF document with your test results and answers to the problems/questions. Word documents will not be accepted. Your zip file should be in the form <lastname>_project_< N >.zip where you replace <lastname> with your last name and <N> with the project number. Upload your zip file via the link provided on the project webpage found at http://urminsky.ca.

In your PDF, please include your python code and any figures you were asked to plot.

Documentation: *Make sure that your code is well enough commented so that it can be easily understood by almost anyone, including yourself, if you look at it again at the end of the course (when you've forgotten how you coded it.)*

Project Briefing

Do not use any pre-defined root finding methods included math or numpy libraries.

Part 1

Design, write and test a PYTHON script to find the roots of an arbitrary mathematical function $y = f(x)$ using the Newton-Raphson method.

Your solution should include a function root_newton of the form

```
def root_newton(func,guess):
    #your solution to newton's method
    ...
    return approx_root, iterations
```

where guess is an initial guess to the root of the function. Note that the function should return the approximate root as well as the number of iterations of the method. Here func refers to a user pre-defined python function which takes in one value, x, and returns the a mathematical function, f evaluated at x as well as the derivative of f at x ($Df(x)$) . For example,

```
def f1(x):
    f = x**2
    df = 2*x
    return f, df
```

In this way we can pass $f(x)$ to root_newton as

```
root_newton(f1,0.5)
```

When designing your program, think carefully about any special cases. For example, what if the initial guess is a root? What if the program takes to long?

Use your program to find all real roots, x for $x \in (0, 5)$ of the following functions

i. $f(x) = \sin(\sqrt{x}) - x$

ii. $f(x) = -2 + 6x - 4x^2 + 0.5x^3$

iii. $f(x) = \sin x + \cos(1 + x^2) - 1$

iv. $f(x) = x^{10} - 1$

v. $f(x) = -(x - 2)^2$

In some cases there is more than one root. You will need to re-run the program for each root. Be careful of your initial guesses in these cases. It is a good idea to plot each function beforehand in order to make a reasonable initial guess.

Use the following termination criterion: fractional estimated error $\epsilon_a \leq \epsilon_s$ where $\epsilon_s = 1 \times 10^{-6}$. In each case, record the value of the root and the number of iterations taken. When does Newton's method work well and when does it fail? Comment briefly.

Part 2 - Lagrange point

There is a point between the earth and the moon, called the L_1 Lagrange point, at which the satellite will orbit the earth in perfect synchrony with the Moon, staying always in-between the two. At this point, the gravitational forces between the moon and earth provide the exact net force to keep the satellite in its orbit.

(a) Show that the distance r from the centre of the earth to the L_1 point satisfies,

$$\frac{GM_e}{r^2} - \frac{GM_m}{(R-r)^2} = \omega^2 r,$$

where R is the distance from the Earth to the Moon, M_e and M_m are the masses of the Earth and Moon respectively, G is the Gravitational constant, and ω is the angular velocity of both the moon and the satellite.

(b) Use your python code above to solve for r numerically. Compute r to at least 4 decimal significant figures. Use the following values

$$\begin{aligned} G &= 6.67 \times 10^{-11}\text{m}^3\text{kg}^{-1}\text{s}^{-2} \\ M_e &= 5.974 \times 10^{24}\text{kg} \\ M_m &= 7.348 \times 10^{22}\text{kg} \\ R &= 3.844 \times 10^{8}\text{m} \\ \omega &= 2.662 \times 10^{-6}\text{s}^{-1} \end{aligned}$$