Problem F.5

Newton Fractals

Due Date: 5/3/2019
Folder: `FinalProject`
File Name: `F5_Newton_Name.py`
Points: 20 points

Problem Background

Newton's method is a way of solving an equation

$$f(z) = 0.$$

It is an iterative method. We start it with an initial "guess" to the solution, z_0. We then find a better approximation, z_1 of the solution through the following equation,

$$z_1 = z_0 - \frac{f(z_0)}{f'(z_0)},$$

where $f'(z)$ is the derivative of $f(z)$. We repeat this process multiple times with the same formula,

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)},$$

until eventually we converge and the iterations start to get close together,

$$|z_{n+1} - z_n| < \text{TOL},$$

where TOL is some small tolerance defining how close we want the iterations to get before we stop.

The final approximation to $f(z) = 0$ depends on where we start, z_0. If we choose a different starting point, we might converge to a different solution, as $f(z) = 0$ might have multiple solutions. One thing to note is that the solutions might be complex, and so the initial starting points z_0 can also be complex.

A Newton fractal is a colored complex plane that denotes pictorially which solution of $f(z) = 0$ each initial point z_0 converges to. A number of these fractals can be seen at Newton Fractal. As an example consider the equation

$$f(z) = z^3 - 1.$$

This has three solutions, $z = 1, z = -e^{i\pi/3}, z = e^{2i\pi/3}$. We label each of these solutions with a color. Then, using Newton's method, say we start with an initial iterate $z_0 = 1 + i$. Using this initial guess, Newton's method converges to 1. Therefore, we would color a dot on the complex plane at (1,1) red. However, if we start at $z_0 = -1 - i$, this converges to $-e^{i\pi/3}$ and so we would color a dot on the complex plane at (-1,-1) blue. This would be done for all initial iterates in the complex plane, and all those colored dots would create a Newton fractal.

In this project, you will create a program to generate a Newton Fractal for any function $f(z)$. Then you can play around with the function and see what different Newton fractals are generated. In addition, some pictures incorporate the number of iterations it takes to converge in the image. This can also be incorporated into your code to create and even more interesting picture.

In addition, there are other methods for solving $f(z) = 0$. One such method that works for polynomials is called Laguerre's method. It can be described as follows:

$$z_{n+1} = z_n - \frac{n}{G(z_n) \pm \sqrt{(n-1)(nH(z_n) - G^2(z_n))}},$$

where n is the degree of the polynomial $f(z)$ and

$$G(z_n) = \frac{f'(z_n)}{f(z_n)} \qquad H(z_n) = G^2(z_n) - \frac{f''(z_n)}{f(z_n)}$$

and the sign in the denominator is choses to maximize the absolute value of the denominator. So for instance if

$$\left|G(z_n) + \sqrt{(n-1)(nH(z_n) - G^2(z_n))}\right| > \left|G(z_n) - \sqrt{(n-1)(nH(z_n) - G^2(z_n))}\right|,$$

then we use the minus sign in Laguerre's method. This method is obviously more complicated than Newton's method, but has many advantages, one of which is that it will always converge, not matter the starting iterate.

Program Criteria

Write a program that does the following:

- Has a `lambda` function which defines a *polynomial* function $f(z)$. You will also need two more `lambda` functions which define the first and second derivative of this polynomial. (Note this is not a polynomial as a list of coefficients, but an actual `lambda` function)
- Define variables that store *all* the solutions to $f(z) = 0$. This can be found using Wolfram Alpha if you can't do it with paper. (Approximations up to 5 digits of accuracy is fine)
- The following may be done either twice in one program, or in two separate programs, one for Newton and one for Laguerre method
  - ⋆ For a large number of initial iterates in the complex plane, I would say at least 200×200 points (that is pick a square grid of 200 points on the real axis and 200 points on the imaginary axis to create 40000 points in total) run both methods.
  - ⋆ Use a maximum of about 10-20 iterations. Records which solution each initial iterate converges to, or if they do not converge within the maximum number of iterations. Use a tolerance of 10^{-4}.
  - ⋆ Assign a color to each solution. Use the color black for initial iterates that do not converge. Plot a dot for each initial iterate with the color of the solution it converged to. You will have to use the internet to figure out how to plot points of different colors. You will probably also want to change the size of those dots to get a detailed plot.
  - ⋆ Be aware, it may take quite a long time to run you program for so many points, so set aside a time, maybe a night, to run the program for your final figures.

Deliverables

Place the following in a folder named `FinalProject` in your repository:

- A Python file `F5_Newton_Name.py` that satisfies the program criteria.
- A PDF file `F5_Newton_Name.pdf` that describes how your program works. This should be a description of how you went about solving this problem. You should go into some detail about your solution method, but I don't want to see something about every `if` statement and `for` loop. As an example of the type of description I'm looking for, see the file `Goldbach_explanation.doc` in the `Final Problem` folder of my repo.
- In the same PDF, give the fractal figures for at least **two** polynomials of degree at least 3. Be sure to give the figures for both Newton and Laguerre method. This should be at least 4 figures.
- Compare and contrast the figures for each polynomial. How are the figures for Newton and Laguerre different/similar to each other for the same polynomial? How does the figure change when considering the same method but a different polynomial?
- Also, describe what changes you made to your code to make it run faster, since this particular code can take a very long time due to the number of initial iterates that must be considered to create a good plot.