CS 510: Fall 2017 Midterm Project

Due: October 27th, 2017, 11pm. (2 week duration)

In this project, we are going to find the solutions to the following polynomial equation:

$$f(z) \equiv 35z^9 - 180z^7 + 378z^5 - 420z^3 + 315z = 0 \tag{1}$$

Since $f(z)$ is a ninth-order polynomial in z, we expect to find nine solutions to this equation, known as the "roots" of $f(z)$. In general, these roots will be complex. For example, the equation $z^2 - 1 = 0$ has the familiar roots 1 and -1, but $z^3 - 1 = 0$ has three roots: 1, $[-1 + \sqrt{3}\,i]/2$, and $[-1 - \sqrt{3}\,i]/2$, where the imaginary number i satisfies $i^2 = -1$. As you know, we can write down a simple closed formula only for the two roots of a simple quadratic equation. As such, we will use a *numerical* technique to find the roots of the much more complicated Eq. (1).

The technique we will use is called the Newton-Raphson method. The idea is to start with a guess of a root z_0, then update it according to the following iterative map:

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)} \tag{2}$$

Here $f'(z) = 9 * 35z^8 - 7 * 180z^6 + 5 * 378z^4 - 3 * 420z^2 + 315$ is the derivative of the function f. The idea is that each iteration will produce a better approximation to the root of $f(z)$. (Fun geometric exercise: why?) The method continues these iterations until two successive values z_n and z_{n+1} are close to each other. When they are closer than some desired precision, you have found the root to within that precision. More precisely, the algorithm is to start with an initial guess z_0, then repeat Eq. (2) to generate the sequence $(z_0, z_1, \ldots, z_n, z_{n+1})$ until the magnitude $|z_{n+1} - z_n| < \epsilon$ for some real tolerance ϵ that sets the precision. The true root is then equal to z_n up to a small radius ϵ in the complex plane.

The obvious dilemma with this method is the question: what should you choose for your initial guess z_0? It turns out that this is a very interesting question, as you will now show. What you will do is use the code you created for your classworks and homeworks to create an initial mesh of the complex plane, such that the points $z = x + iy$ are bounded between $x \in [-2, 2]$ and $y \in [-2, 2]$, with some small spacings $dx = 0.01$ and $dy = 0.01$ between consecutive points. For each point z in this grid, use that point as the initial condition for the above Newton-Raphson method, setting $\epsilon = \sqrt{dx^2 + dy^2}$ according to the spacing between your mesh points. This method should produce two output values: $z \to (z_n, n)$, namely the root z_n that the method converges to, and the number of iterations n that it took to converge within the tolerance ϵ. You will use these two pieces of information to create a plot of the complex plane to show how different points converge to different roots. To do this, you will color each different root a different color, and (for extra-credit) control the darkness of the color by the number of iterations it takes to converge.

Problem 0. [20pt] Edit the `README.md` file and link it to Travis.ci properly to automate testing using the nose framework (5pt). Sign the README to verify that all submitted work is your own (5pt). Place your main python code in a properly commented and formatted file `midterm.py` (5pt) that imports helpful modules that you already created in your homeworks and classwork (copy them into this repository). Create a supplementary Jupyter notebook `Midterm.ipynb` (5pt) that imports your python code and presents your results in a clean and clear way. Done properly, the notebook should contain almost no code (only simple function calls), and should focus on the discussion of the results of the code instead.

Problem 1. [35pt] In your python module `midterm.py`, implement a python class `NewtonRaphson` that subclasses your `ArrayComplexPlane` class for simplicity (5pt). Implement a separate function that applies the update formula Eq. (2) to a single complex number using the function in Eq. (1) and its derivative (10pt). Implement a class method that applies the first function repeatedly to a point z until two successive iterations are closer than $\epsilon = \sqrt{dx^2 + dy^2}$ set by the chosen mesh spacing; have this method return both the final iteration value z_n and the number of iterations n as a tuple `(zn,n)` (10pt). Implement another class method that vectorizes this function and applies it to your entire complex plane mesh (10pt). You will find that the vectorized function will return a tuple of two arrays: one array of root z_n values, and other array of iteration n values. Store these new arrays as class attributes. You will use these two arrays to make plots in the next problem.

Problem 2. [20pt] Create a method for your class that uses the results of problem 1 to plot an image using `matplotlib.pyplot.imshow` (10pt). Have each pixel of the image correspond to a point in your complex plane grid (just like the Julia set plots from the homework). Color each pixel a different color corresponding to each distinct root returned by the Newton-Raphson method (10pt). (Helpful hint: Each iteration will only return a root up to the chosen tolerance, so be sure to identify the nine distinct roots properly by truncating extra decimal places.) [Extra credit (20pt): Change the darkness of the plotted color of each point according to the number of iterations it took to converge to the root, with larger iteration numbers corresponding to darker colors.] (Helpful hint: To get distinct colors on the same plot, you will need to overlay multiple calls to `imshow`. It may be helpful to create "masked arrays" using `numpy.ma.masked_array` and Boolean indexing of numpy arrays (see slides) to avoid replotting points with each call. There are many possible solutions, however, so do what works.) On your plot, overlay black points with `numpy.pyplot.scatter` to show where the nine roots actually are in the complex plane. Be sure to label your x and y axes of the complex properly, so it is clear where the points are.

Problem 3. [15pt] In your notebook, make a professional report that clearly describes the problem and the method. Use math equations in your notebook to clarify your discussion. Import your midterm module and use it to plot the results of the Newton-Raphson method using the default settings of $x \in [-2, 2]$, $y \in [-2, 2]$, with $dx = dy = 0.01$. (10pt)

Problem 4. [10pt] In your notebook, discuss your findings in detail (10 pt). If needed to support your discussion, create more plots that zoom into regions of the basic plot from problem 3 to explore its structure. Consider the following questions in your discussion: Qualitatively, what is happening around each root? What is happening on the boundaries between basins that converge to each root? Is there regular structure at these boundaries? What happens if you decrease the mesh spacing and zoom into interesting regions between the roots? Would you have guessed this behavior from Eq. (2)? If you were to write an academic paper on this subject, how would you approach analyzing the behavior further?