4330 Assignment 11 *

June 15, 2020

Write your code for the following problems in a single file named:

`hw11-`*lastname*`.py`

As discussed in class, Newton's method also often works for differentiable functions from $\mathbb{C}$ to $\mathbb{C}$. However, the behavior in this case is decidedly more interesting. In this assignment, you'll see what I mean. In this assignment, we will fix a polynomial function F, and apply Newton's method using different initial points from the complex plane. We will plot each pixel a color determined by the root to which Newton's method converges (or black, if Newton's method diverges with that initial point).

You will need to use the Python `cmath` library for handling complex numbers. All you should need to know is that complex literals can be specified e.g., as `z = 3+1j` for the complex number $3 + i$. Given a complex number `z`, you can recover its real and imaginary parts using `z.real` and `z.imag`.

You will use the supplied file `hw11.py` as a starting point, and write the two missing functions described in the problems below.

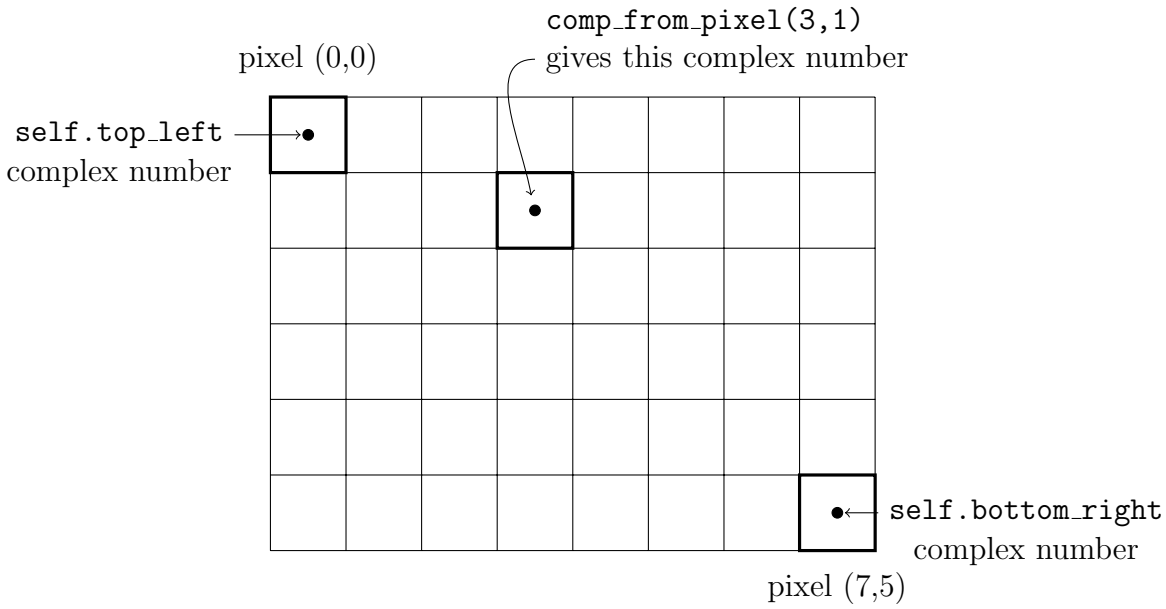
---

(1) (20 points) Write the function `Newton`, by adapting the function you wrote in Assignment 9, making sure to account for the fact that the starting point `z0` is a complex number. Test your function by starting it at several complex numbers and verifying that it converges to a root of the function that is hardcoded into the program.

---

(2) (20 points) Write the function `comp_from_pixel` which returns the complex number corresponding to the center of the pixel `(x,y)`. You are already given that the center of the top-left pixel corresponds to the complex number `self.top_left` and the center of the bottom-right pixel corresponds to the complex number `self.bottom_right`. From these, you should compute the complex number corresponding to an arbitrary pixel. The figure below illustates, for the fictional case of a window which is 8 pixels wide and 6 pixels tall.

Example: canvas is 8×6 pixels.
`self.screen_width` is 8
`self.screen_height` is 6



For example, if `self.top_left` were equal to `10.0 + 2.0j` (e.g., the complex number $10.0 + 2i$) and `self.bottom_right` were equal to `14.0 + 0j`, then `comp_from_pixel(3,1)` should return the complex number corresponding to the point indicated in the figure, which is 10.0 + 3*(4.0/7) + (2.0 - 1*(2.0/5))j, or about 11.714 + 1.6j.