

Homework 9

Due Friday, December 3, 2021

Math 205

1. To refresh your memory of complex numbers, let

$$z = 3 + 4i$$

$$w = 2 - i.$$

Find $z + w$, $z \cdot w$, and z^2 by hand, using the fact that $i^2 = -1$.

Recall that the absolute value of a complex number is

$$|x + iy| = \sqrt{x^2 + y^2}.$$

Find $|z|$ and $|w|$ by hand.

Now do the same calculations in Python, where complex numbers look like $z = 3 + 4j$ and $w = 2 - 1j$. The syntax for absolute value is `abs(z)`.

2. Let $c = -.6 + .7i$. Start with $z = 0$, and then replace z with $z^2 + c$, over and over. Do this in Python and look at the value of z as you go. Observe that z bounces around at small values for a little while, and then after about 10 steps starts to blow up really big.

Do the same with $c = -.3 + .4i$, and observe that z never blows up, but instead approaches a limit of about $-.2842 + .2550i$.

You can prove, although it's kind of tricky, that if you ever get to a z with $|z| \geq 2$, then you'll escape to infinity.

The *Mandelbröt set* is the set of complex numbers c for which the process in problem 2 does *not* escape to infinity. So $-.3 + .4i$ is in the Mandelbröt set, and $-.6 + .7i$ is not. We want to make a picture of this set and explore it. If you have your own idea about how you want to do this, go for it. Or you can follow the outline before.

- Write a function `escape_time(c)`, which takes a complex number c and does the following. Start with $z = 0$, and then replace z with $z^2 + c$ over and over. If at some point you have $|z| \geq 2$, return how many steps you've done. After 10 steps, just give up and return 10.

What is `escape_time(-.6+.7j)`? Does this agree with what you found in #2? What about `escape_time(-.3+.4i)`?

- Take your code from last week to draw pictures on a 600×600 square. Realize the following pseudo-code:

```
for x in range(600):
    for y in range(600):
        mathx = some expression that's -2 when x=0, and +2 when x=600
        mathy = some expression that's +2 when y=0, and -2 when y=600
        t = escape_time(mathx + mathy*1j)
        pixels[x,y] = let the color or intensity vary depending on t
```

- Rewrite your code so that it goes up to 50 steps, rather than stopping at 10, to get a crisper picture.
- Rewrite your code so that rather than being centered at $(0,0)$ and having a width of 4 units, you have variables `centerx` and `centery` and `width` defined before the for loop, and then inside the for loop,

```
mathx = some expression that's centerx - width/2 when x=0,
and centerx + width/2 when x=600
mathy = some expression that's centery + width/2 when y=0,
and centery - width/2 when y=600
```

Now set the center to be $(-0.909, 0.275)$, zoom in closer and closer by adjusting `width`, and see what you see. If it starts to get blurry, increase the maximum number of steps from 50 to 100 or 200 or 300.

- Do the same with the following centers:

$(-0.761574, -0.0847596)$

$(0.001643721971153, -0.822467633298876)$

$(-0.743643887037158704752191506114774,$

$0.1318259042053119704931320385139)$

8. *Optional: if you have time to kill, or need even more psychedelic paisley in your life.* Instead of always starting from $z = 0$ and looking at what happens as we iterate $z \mapsto z^2 + c$ for various complex numbers c , we can fix a value of c and let the starting value of z vary. The set of initial z 's for which this does not escape to infinity is called a *Julia set*. By definition, 0 is in the Julia set with parameter c if and only if c is in the Mandelbröt set; interestingly, the Julia set is connected if c is in the Mandelbröt set, and disconnected if it is not.

Have the computer plot the Julia sets for several values of c , including the four that we explored above. Observe that the swirliness of the Julia set for c reflects the swirliness of the Mandelbröt set near c .