Lab 20

Complex Numbers

Lab Objective: *Create visualizations of complex functions. Visually estimate their zeros and poles, and gain intuition about their behavior in the complex plane.*

Representations of Complex Numbers

A complex number $z = x + iy$ can be written in *polar coordinates* as $re^{i\theta}$ where

- $r = \sqrt{x^2 + y^2}$ is the magnitude of z, and
- $\theta = \arctan(y/x)$ is the angle between z and 0, as in Figure 20.1.

Conversely, Euler's formula implies $re^{i\theta} = r\cos(\theta) + ir\sin(\theta)$. Then if we set $re^{i\theta} = x + iy$ and equate real and imaginary parts, we find $x = r\cos(\theta)$ and $y = r\sin(\theta)$.

NumPy makes it easy to work with complex numbers and convert between coordinate systems. The function `np.angle()` returns the angle of a complex number (between $-\pi$ and π) and the function `np.absolute()` returns the magnitude. Use these to compute θ and r, respectively. These functions also operate elementwise on NumPy arrays.

Note that in Python, `1j` is used for the complex number $i = \sqrt{-1}$. See the code below for an example.

```
>>> import numpy as np
>>> from matplotlib import pyplot as plt
# Set z = 2 - 2i
>>> z = 2 - 2*1j
>>> theta = np.angle(z)
>>> r = np.absolute(z)
# np.angle() returns a value between -pi and pi.
>>> print r, theta
(2.8284271247461903, -0.78539816339744828)
# Check that z=re^(i*theta)
>>> np.allclose(z, r*np.exp(1j*theta))
True
```

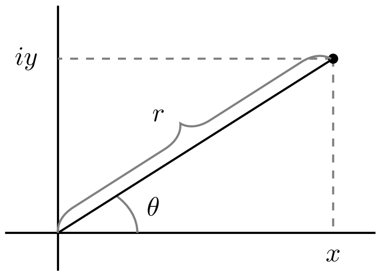

Figure 20.1: The complex number represented by the black dot equals both $x + iy$ and $re^{i\theta}$, when θ is written in radians.

Visualizing complex functions

Suppose we wish to graph a function $f(z) : \mathbb{C} \to \mathbb{C}$. The difficulty is that $\mathbb{C}$ has 2 real dimensions, so the graph of f should use 4 real dimensions. Since we already have ways to visualize 3 dimensions, we should choose one dimension to ignore. We will ignore the magnitude $r = |f(z)|$ of the output.

To visualize f, we will assign a color to each point $z \in \mathbb{C}$. The color will correspond to the angle θ of the output $f(z)$. As an example, we have plotted the identity function $f(z) = z$ in Figure 20.2. As θ goes from 0 to 2π, the colors cycle smoothly counterclockwise from red to green to purple and back to red.

This kind of plot uses rectangular coordinates in the domain and polar coordinates (or rather, just the θ-coordinate) in the codomain. Note that this kind of plot tells us nothing about $|f(z)|$.

You can create the plot in Figure 20.2 as follows. Begin by creating a grid of complex numbers. We create the real and imaginary parts separately, and then use `np.meshgrid()` to turn them into a single array of complex numbers.

```
>>> x = np.linspace(-1, 1, 401)
>>> y = np.linspace(-1, 1, 401)
>>> X, Y = np.meshgrid(x, y)
>>> Z = X + 1j*Y
```

Now we compute the angles of the points in `Z` and plot them using `plt.pcolormesh()`. We use the colormap `'hsv'`, which is red at both ends, so that 0 and 2π will map to the same color.

```
>>> plt.pcolormesh(X, Y, np.angle(Z), cmap='hsv')
>>> plt.show()
```

Problem 1. Write the following function to plot any function from $\mathbb{C}$ to $\mathbb{C}$. Plot the angle only, as above, ignoring the magnitude.

```
def plot_complex(f, xbounds, ybounds, res=401):
    '''Plot the complex function f.

    INPUTS:
```

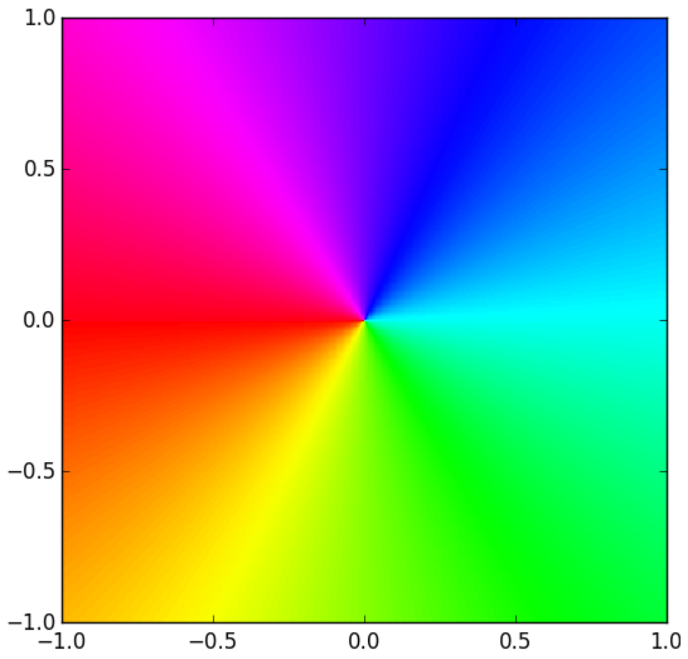

Figure 20.2: Plot of $f : \mathbb{C} \to \mathbb{C}$ defined by $f(z) = z$. The color at each point z represents the argument of $f(z)$.

```
    f         - A function handle. Should represent a function
                  from C to C.
    xbounds   - A tuple (xmin, xmax) describing the bounds on the real part
                  of the domain.
    ybounds   - A tuple (ymin, ymax) describing the bounds on the imaginary
                  part of the domain.
    res       - A scalar that determines the resolution of the plot.
                  Defaults to 401.
    '''
```

Check your function on $f(z) = z$ (graphed in Figure 20.2) and on the function $f(z) = \sqrt{z^2+1}$, which is graphed in Figure 20.3.

Hint: When you call `plt.pcolormesh()`, specify the keyword arguments `vmin` and `vmax`. These define which values should map to each end of the color scale. We want $-\pi$ to map to the low end of the color scale, and π to map to the high end. If not specified, matplotlib will scale the colormap to fit your data exactly.

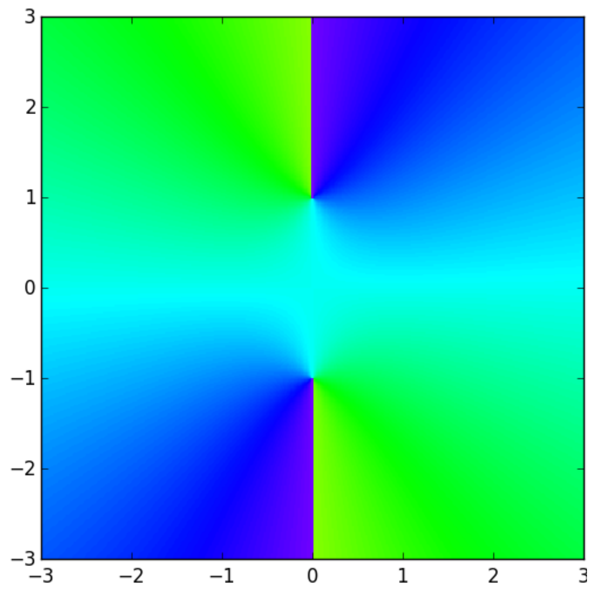

Figure 20.3: Plot of the angle of $\sqrt{z^2+1}$ on the domain $\{x+iy \mid x \in [-3,3],\ y \in [-3,3]\}$ created by `plot_complex()`.

The choice to ignore the magnitude may seem arbitrary. We can also write a complex plotting function to ignore the angle and only plot the magnitude. This will give us some different intuition about the function, while losing some information that we would get from the angle plot.

Problem 2. Write a new complex plotting function called `plot_complex_magnitude` which ignores the angle and plots only the magnitude. This should resemble your answer to Problem 1, with small modifications. Leave `vmin` and `vmax` unspecified when plotting.

Check your function on $f(z) = \sqrt{z^2+1}$. Your plot should look like the right subplot in Figure 20.4. Note the difference between this plot and the one from the previous problem.

Hint: A wraparound colormap like `'hsv'` doesn't work well here. Use any sequential colormap that makes it easy to distinguish between high and low values. See the matplotlib documentation for a list of colormaps.

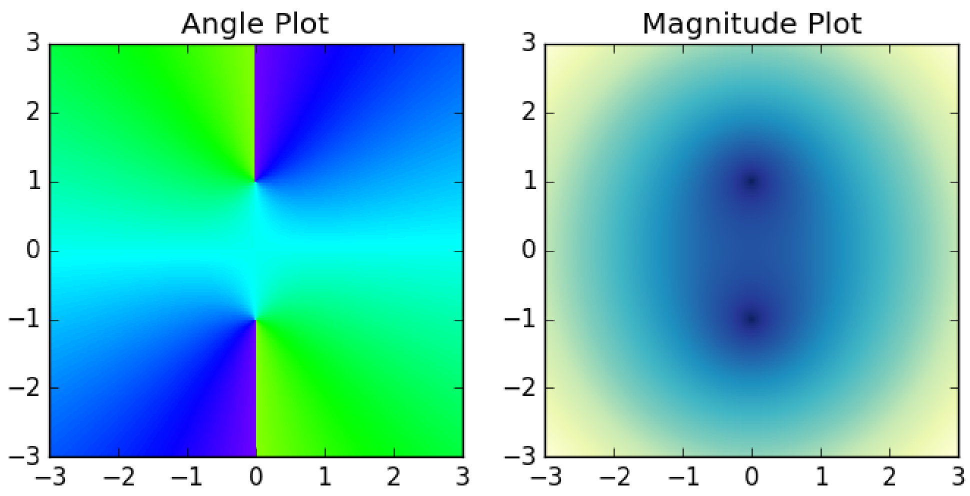

Figure 20.4: Plots of $\sqrt{z^2+1}$ on $\{x+iy \mid x \in [-3,3],\ y \in [-3,3]\}$, visualizing the angle and the magnitude of the function. Notice how a discontinuity is clearly visible on the left, but disappears from the plot on the right.

Analyzing Complex Plots

The angle plot is generally more useful than the magnitude plot for visualizing function behavior, zeros, and poles. Throughout the rest of the lab, use `plot_complex` to plot only the angle, and ignore the magnitude.

Zeros

Complex plots can be surprisingly informative. From an angle plot we can estimate not only a function's zeros, but also their multiplicities.

Problem 3.

1. Use `plot_complex()` to plot the functions z^2, z^3, and z^4.
2. Plot $z^3 - iz^4 - 3z^6$ on the domain $\{x+iy \mid x \in [-1,1],\ y \in [-1,1]\}$ (this plot is Figure 20.5). Compare it to your plot of z^3, especially near the origin. Based on these plots, what can you learn about the zeros of a function from its graph?

In Problem 3 you should have noticed that in a plot z^n, the colors cycle n times counterclockwise around 0. (Note: For the remainder of this lab we will define red $\rightarrow$ yellow $\rightarrow$ green $\rightarrow$ blue $\rightarrow$ red to be the "forward" direction, such that the colors are circling counterclockwise in Figure 20.2.)

This is explained by looking at z^n in polar coordinates:

$$z^n = (re^{i\theta})^n = r^n e^{i(n\theta)}.$$

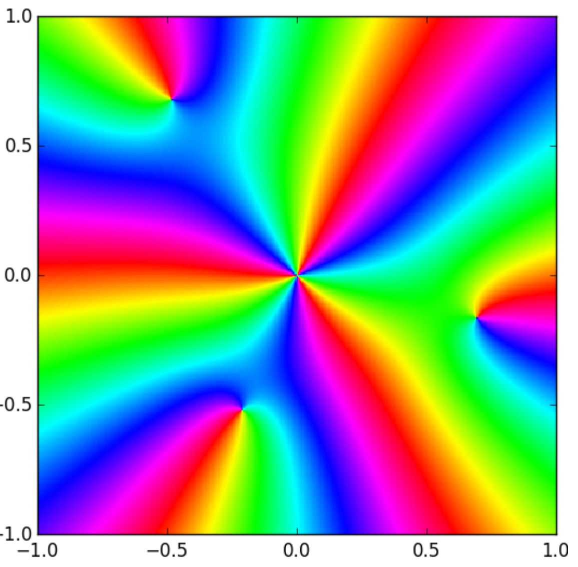

Figure 20.5: Plot of $f(z) = z^3 - iz^4 - 3z^6$ on the domain $\{x + iy \mid x \in [-1, 1],\ y \in [-1, 1]\}$. From this plot we see that $f(z)$ has a zero of order 3 at the origin, and 3 zeros of order 1 scattered around it. This accounts for the 6 roots of $f(z)$ that are guaranteed to exist by the Fundamental Theorem of Algebra.

Multiplying θ by a number greater than 1 compresses the graph along the "θ-axis" by a factor of n. In other words, the output angle repeats itself n times in one cycle of θ. Compare this to replacing $f(x)$ with $f(nx)$ when f is a function from $\mathbb{R}$ to $\mathbb{R}$.

From Problem 3 you should also have noticed that the plot of $z^3 - iz^4 - 3z^6$ looks a lot like the plot of z^3 near the origin. This is because when z is very small, z^4 and z^6 are much smaller than z^3, and so the behavior of z^3 dominates the function.

In general, $f(z)$ has a *zero of order n at z_0* if the Taylor series of $f(z)$ centered at z_0 can be written as

$$f(z) = \sum_{k=n}^{\infty} a_k (z - z_0)^k \qquad \text{with } a_n \neq 0.$$

In other words, $f(z) = a_n(z - z_0)^n + a_{n+1}(z - z_0)^{n+1} + \ldots$. In a small neighborhood of z_0, the quantity $|z - z_0|^{n+k}$ is much smaller than $|z - z_0|^n$, and so the function behaves like $a_n(z - z_0)^n$. This explains why we can estimate the order of a zero by counting the number of times the colors circle a point (see Figure 20.5).

Poles

The plots created by `plot_complex()` also contain information about the poles of the function plotted.

Problem 4.

1. Use `plot_complex()` to plot the function $f(z) = 1/z$. Compare this to the plot of $f(z) = z$ in Figure 20.2.
2. Plot z^{-2}, z^{-3}, and $z^2 + iz^{-1} + z^{-3}$ on the domain $\{x + iy \mid x \in [-1, 1],\ y \in [-1, 1]\}$. Compare the plots of the last two functions near the origin. Based on these plots, what can you learn about the poles of a function from its graph?

In Problem 4 you should have noticed that in the graph of $1/z^n$, the colors cycle n times *clockwise* around 0. Again this can be explained by looking at the polar representation:

$$z^{-n} = (re^{i\theta})^{-n} = r^{-n}e^{i(-n\theta)}.$$

The minus-sign on the θ reverses the direction of the colors, and the n makes them cycle n times.

In general, a function has a *pole of order* n at z_0 if its Laurent series on a punctured neighborhood of z_0 is

$$f(z) = \sum_{k=-n}^{\infty} a_k(z - z_0)^k \qquad \text{with } a_{-n} \neq 0.$$

In other words, $f(z) = a_{-n}(z-z_0)^{-n} + a_{-n+1}(z-z_0)^{-n+1} + \ldots$. Since $|z-z_0|^{-n+k}$ is much smaller than $|z-z_0|^{-n}$ when $|z-z_0|$ is small, near z_0 the function behaves like $a_{-n}(z - z_0)^{-n}$. This explains why we can estimate the order of a pole by counting the number of times the colors circle a point in the clockwise direction.

Finally, a function has an *essential pole* at z_0 if its Laurent series in a punctured neighborhood of z_0 requires infinitely many terms with negative exponents. For example,

$$e^{1/z} = \sum_{k=0}^{\infty} \frac{1}{n!z^n} = 1 + \frac{1}{z} + \frac{1}{2}\frac{1}{z^2} + \frac{1}{6}\frac{1}{z^3} + \ldots.$$

The plot of $f(z) = e^{1/z}$ is in Figure 20.6. The colors cycle infinitely many times around an essential singularity.

Using Plots to Estimate Poles and Zeros

To summarize, poles and zeros can be estimated from a complex plot with the following rules.

- Colors circle counterclockwise around zeros.
- Colors circle clockwise around poles.
- The number of times the colors cycle equals the order of the zero or pole.

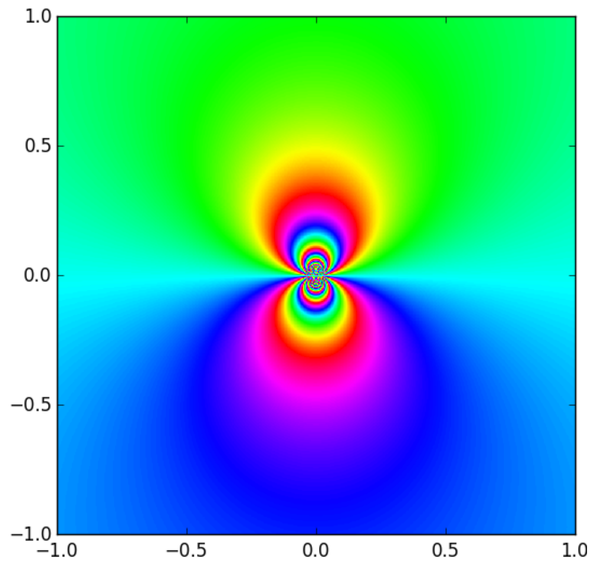

Figure 20.6: Plot of $e^{1/z}$ on the domain $\{x+iy \mid x \in [-1,1],\ y \in [-1,1]\}$. The colors circle clockwise around the origin because it is a singularity, not a zero. Because the singularity is essential, the colors repeat infinitely many times.

Problem 5. Plot these functions on the domains given. Estimate the number and order of their poles and zeros.

- $f(z) = e^z$ on $\{x+iy \mid x \in [-8,8],\ y \in [-8,8]\}$
- $f(z) = \tan(z)$ on $\{x+iy \mid x \in [-8,8],\ y \in [-8,8]\}$
- $f(z) = \frac{16z^4+32z^3+32z^2+16z+4}{16z^4-16z^3+5z^2}$ on $\{x+iy \mid x \in [-1,1],\ y \in [-1,1]\}$

One useful application of complex plots is to estimate the zeros of polynomials and their multiplicity.

Problem 6. Use complex plots to determine the multiplicity of the zeros of each of the following polynomials. Use the Fundamental Theorem of Algebra to ensure that you have found them all.

1. $-4z^5 + 2z^4 - 2z^3 - 4z^2 + 4z - 4$
2. $z^7 + 6z^6 - 131z^5 - 419z^4 + 4906z^3 - 131z^2 - 420z + 4900$

Plotting functions is not a substitute for rigorous mathematics. Often, plots can be deceptive.

Problem 7.

1. This example shows that sometimes you have to "zoom in" to see all the information about a pole.
   (a) Plot the function $f(z) = \sin(\frac{1}{100z})$ on the domain $\{x + iy \mid x \in [-1, 1],\ y \in [-1, 1]\}$. What might you conclude about this function?
   (b) Now plot $f(z)$ on $\{x + iy \mid x \in [-.01, .01],\ y \in [-.01, .01]\}$. Now what do you conclude about the function?
2. This example shows that from far away, two distinct zeros (or poles) can appear to be a single zero (or pole) of higher order.
   (a) Plot the function $f(z) = z + 1000z^2$ on the domain $\{x + iy \mid x \in [-1, 1],\ y \in [-1, 1]\}$. What does this plot imply about the zeros of this function?
   (b) Calculate the true zeros of $f(z)$.
   (c) Plot $f(z)$ on a domain that allows you to see the true nature of its zeros.

Multi-Valued Functions

Every complex number has two complex square roots, since if $w^2 = z$, then also $(-w)^2 = z$. If z is not zero, these roots are distinct.

Over the nonnegative real numbers, it is possible to define a continuous square root function. However, it is not possible to define a continuous square root function over any open set of the complex numbers that contains 0. This is intuitive after graphing $\sqrt{z}$ on the complex plane.

Problem 8.
1. Use `plot_complex` to graph $f(z) = \sqrt{z}$. Use `np.sqrt()` to take the square root.
2. Now plot $f(z) = -\sqrt{z}$ to see the "other square root" of z. Describe why these two plots look the way they do.

Just as raising z to a positive integer "compresses the θ-axis", making the color wheel repeat itself n times around 0, raising z to a negative power *stretches* the θ-axis, so that only one n^{th} of the color wheel appears around 0. The colors at the ends of this n^{th}-slice are not the same, but they appear next to each other in the plot of z^{-n}. This discontinuity will appear in every neighborhood of the origin.

If your domain does not contain the origin, it is possible to define a continuous root function by picking one of the roots.

Appendix

It is possible to visualize the argument and the modulus of the output of a complex function $f(z)$. One way to do so is to assign the modulus to a *lightness* of color. For example, suppose we have a complex number with argument 0, so it will map to red in the color plots described above. If its modulus is very small, then we can map it to a blackish red, and if its modulus is large, we can map it to a whitish red. With this extra rule, our complex plots will still be very much the same, except that zeros will look like black dots and poles will look like white dots (see Figure 20.7 for an example).

The code below implements the map we just described. Be warned that this implementation does not scale well. For example, if you try to plot a complex function whose outputs are all very small in modulus, the entire plot will appear black.

```
import numpy as np
import matplotlib.pyplot as plt
from colorsys import hls_to_rgb

def colorize(z):
    '''
    Map a complex number to a color (or hue) and lightness.

    INPUT:
    z - an array of complex numbers in rectangular coordinates

    OUTPUT:
    If z is an n x m array, return an n x m x 3 array whose third axis encodes
    (hue, lightness, saturation) tuples for each entry in z. This new array can
    be plotted by plt.imshow().
    '''

    zy=np.flipud(z)
    r = np.abs(zy)
    arg = np.angle(zy)

    # Define hue (h), lightness (l), and saturation (s)
    # Saturation is constant in our visualizations
    h = (arg + np.pi)  / (2 * np.pi) + 0.5
    l = 1.0 - 1.0/(1.0 + r**0.3)
    s = 0.8

    # Convert the HLS values to RGB values.
    # This operation returns a tuple of shape (3,n,m).
    c = np.vectorize(hls_to_rgb) (h,l,s)

    # Convert c to an array and change the shape to (n,m,3)
    c = np.array(c)
    c = c.swapaxes(0,2)
    c = c.swapaxes(0,1)
    return c
```

The following code uses the `colorize()` function to plot $\frac{z^2-1}{z}$. The output is Figure 20.7.

```
>>> f = lambda z :  (z**2-1)/z
```

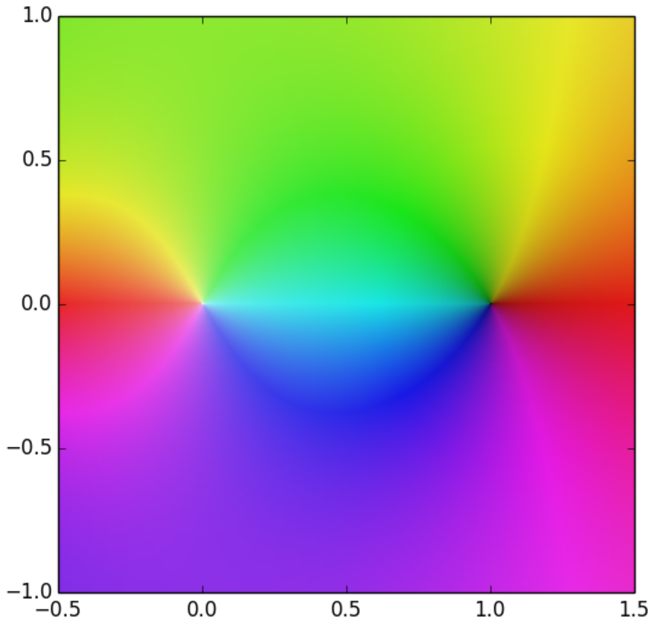

Figure 20.7: Plot of the function $\frac{z^2-1}{z}$ created with `colorize()`. Notice that the zero at 1 is a black dot and the pole at 0 is a white dot.

```
>>> x = np.linspace(-.5, 1.5, 401)
>>> y = np.linspace(-1, 1, 401)
>>> X,Y = np.meshgrid(x,y)
>>> Z=f(X+Y*1j)
>>> Zc=colorize(Z)
>>> plt.imshow(Zc, extent=(-.5, 1.5, -1, 1))
>>> plt.show()
```