Homework 0

Logan Xu

September 5 2019

1 Problem 1

Consider the function

$$f(x) = exp(-15x + i1.5x), -10 \leq x \leq 10 \tag{1}$$

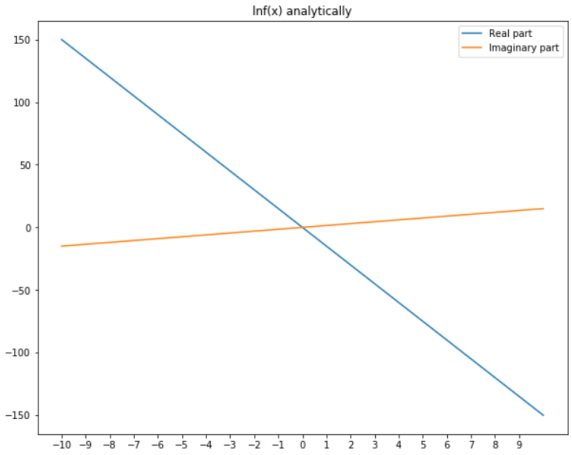
we are to calculate the value of $f(x)$ and plot $ln(f(x))$ using three different methods.

1.1 Analytically

Observe that $f(x)$ is an exponential function, the natural log of it is just the exponent, i.e.

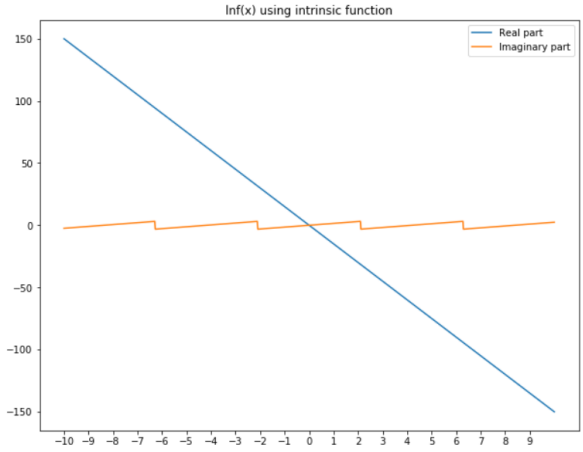
$$ln(f(x)) = -15x + i1.5x. \tag{2}$$

The plot consists of two straight lines as expected.



1.2 Intrinsic function

I used the Numpy log function after using the intrinsic exponential function to get $ln(f(x))$. The plot of the imaginary part looks different whereas the real part is still a straight line with a slope of -15.



1.3 Taylor series

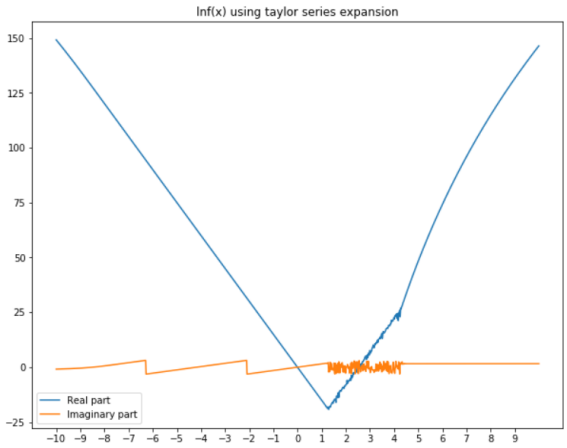
The Taylor series expansion of $f(x)$ is given as

$$f(x) = exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ... \tag{3}$$

Substituting $f(x) = exp(-15x + i1.5x)$, we have

$$f(x) = 1 + (-15x + i1.5x) + \frac{(-15x + i1.5x)^2}{2} + \frac{(-15x + i1.5x)^3}{6} + ... \tag{4}$$

In my work, I chose to truncate the Taylor series at $n = 142$. Beyond this point Python gave me an OverflowError. The resulting plot looks vastly different.



1.4 Why are the plots different?

The graph should look like the one plotted using analytical function, two straight lines with slopes -15 and 1.5. The reason that the imaginary part of second plot looks like periodic is when taking a natural log of a complex number, say in polar form $z = Re^{i\theta}$, the real part $(ln(R))$ is fine, but the

imaginary part ($i\theta$) is one of the cases. This is due to the complex number z can be expressed as $z = Re^{i\theta+2n\pi}$ for all integer n. And when Python does this, it will force all θ into $[0, 2\pi]$ by subtracting or adding as many 2π as needed. So when looking at the plot, it is no surprise that all the imaginary values fall into the range $[-2\pi, 2\pi]$.

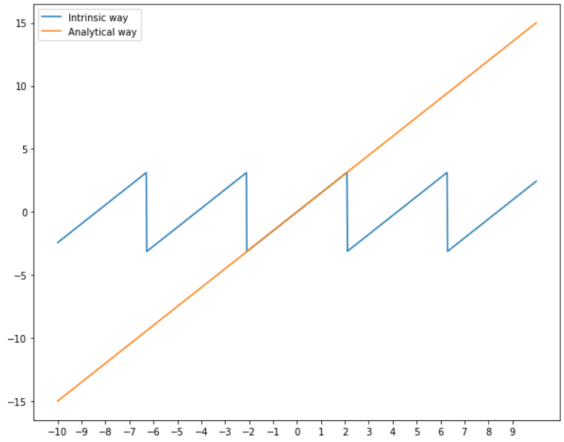


Figure 1: Although the values are off for the intrinsic method, the slope is still making sense, which means the program just did subtraction/addition (upshift/downshift if you think graphically) to the values.

The plot for the Taylor series expansion method looks almost completely different. Let's look closely what the program is trying to do. The Taylor series expansion of the original function, as stated in Equation (4) isn't doing us much favor when we want to separate the real and imaginary parts because there are powers to the imaginary number i in every higher order term. Let's do it in a different way.

$$f(x) = e^{-15x}e^{i1.5x} = e^{-15x}(cos(1.5x) + isin(1.5x)). \tag{5}$$

We can now see that the cosine terms will be real, the sine terms imaginary. The real part, a.k.a $e^{-15x}cos(1.5x)$ shouldn't have any problem when $cos(1.5x)$ is positively valued. When the cosine term turns negative, we are technically doing expansion of $-e^{-15x}$, hence the symmetry and opposite slope. I guess the wiggling in this part is a result of periodicity of the cosine function.
As shown above there's a change in slope when $4 < x < 5$. I suspect this has something to do with the cosine function turning positive again. But I'm not sure why the slope changed. The imaginary part stays flat is the other thing that puzzles me.

2 Problem 2

Consider the complete elliptic integral

$$K(k) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}} = \int_0^1 \frac{dt}{\sqrt{(1 - t^2)(1 - k^2 t^2)}}. \tag{6}$$

2.1 Evaluate $K(\frac{1}{\sqrt{2}})$

Scipy.special has a built-in function for the complete elliptic integral. If called, the value is **1.854**. If using the power series

$$K(k) = \frac{\pi}{2} \sum_{n=0}^{\infty} (\frac{(2n)!}{4^n (n!)^2})^2 k^{2n}. \tag{7}$$

and truncating the series at $n = 1000$, it gives the same answer (with smaller n it works too, but I didn't check how big it needs to be since this already agrees with the other answer).

2.2 Midpoint method

Using 20000 steps, I was able to get a result that agrees with the previous method for the first form (the one that has a θ in it), **1.854**. However, I cannot get 4 digits of accuracy for the second form even when using this many steps, the best result being **1.850**.

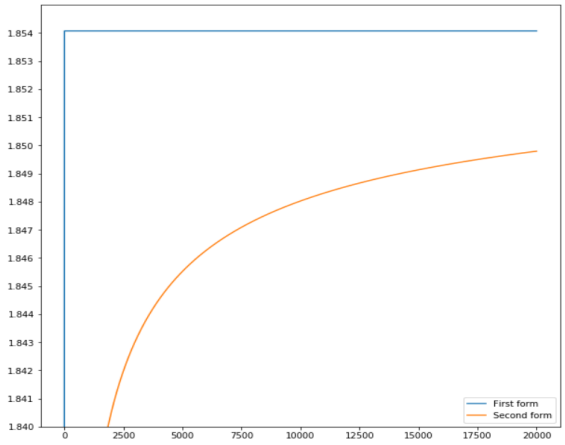


Figure 2: The first form instantly jumps to the correct value whereas the second form is still growing.

To figure out exactly how many steps needed to reach 4 digits of accuracy for the first form I zoomed in the first few values.

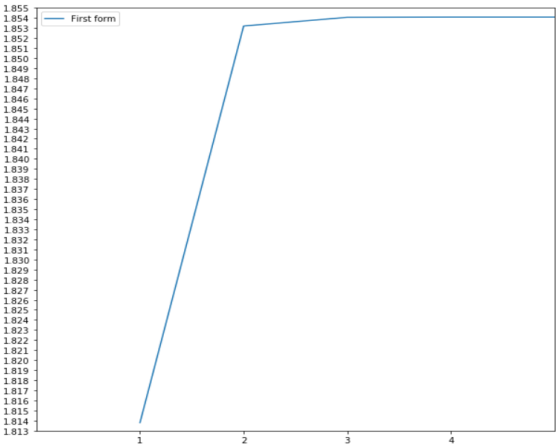


Figure 3: It gets 4 digits of accuracy using **3** steps.