Python for Programmers

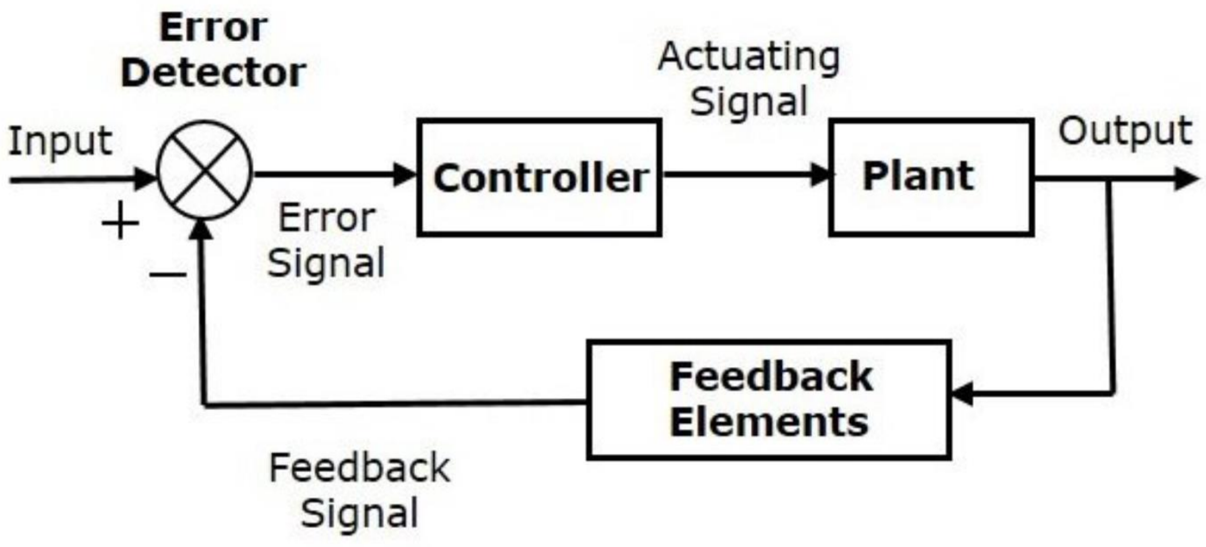
May 16, 2022

Root Locus Plotting

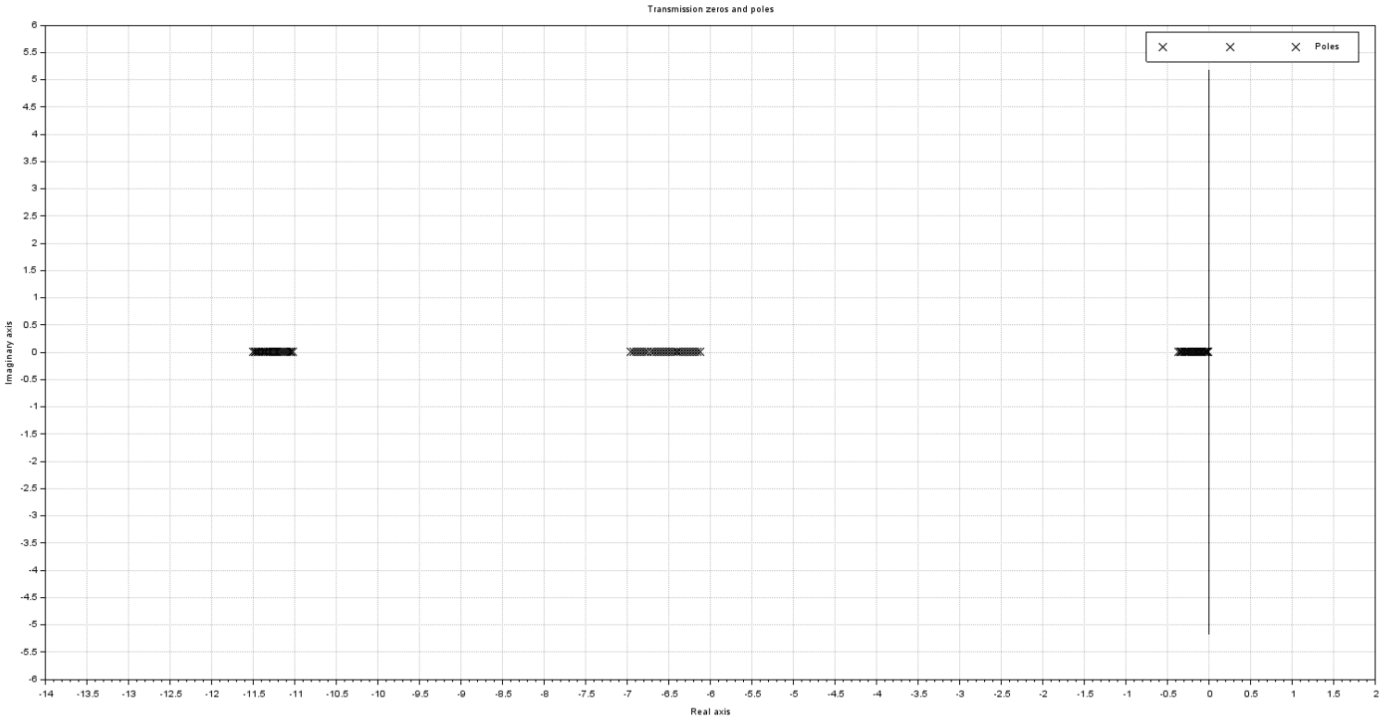
Maya Tene, Alejandro Hernandez

What is a Closed Loop System?

- In engineering design processes, output is often known, responsibility is designing
- Closed Loop System (CLS)
  - Output recycled as input
  - Ex. Oven, Stopping at a Red Light
- System setup
  - Controller + Plant + Feedback Loop
  - Gain is a controller variable

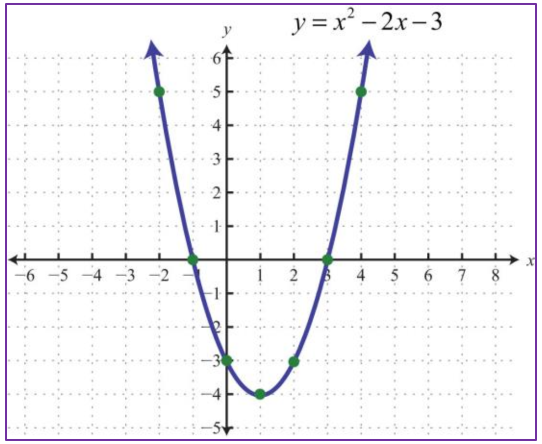


Why are they called "root locus" plots?



<u>Locus</u> (n): a curve or figure formed by points satisfying a particular equation.

For example, we can consider a parabola as a locus plot of Y as X changes.



Why "root locus"?

We want to plot the behavior of the roots of an equation, as we edit our controller variable.

So how do we generate root locus plots?

Root Locus Rules/Definitions

- All RL operations are done on the L(s) CLS representation $L(s) = C(s)P(s)$
- 5 “Rules” of Root Locus Generation
  - Zero: RL Plots are symmetrical across the horizontal axis
  - I: L(s) has m zeros, n poles, m-n branches approach ∞, m branches exit poles and enter zeroes
  - II: Real Axis portion exist to the left of an odd number of poles and zeroes
  - III: Infinity branch asymptotes intersect the Real Axis at α and form an angle, θ with the horizontal
  - IV: branches exit poles at a departure angle ϕ and enter zeros at an arrival angle ψ
  - V: Break in/out behavior occurs at the roots of the derivative of L(s)
    - $L(\dot{s}) = \frac{n}{d} = \dot{n}d - n\dot{d} = 0$
- When CLS are set to an unstable gain, they break
  - The section of an RL plot to the right of the Imaginary axis represents unstable design range

Root Locus Rules/Definitions (simplified)

Given a characteristic equation L(s) = n(s) / d(s)

0. RL plots are symmetrical across the horizontal axis

I. L(s) has n poles, m of which approach zero and n-m which approach infinity as s increases

II. Breakout points may only occur to the left of an odd number of poles/zeros

III. Asymptotes intersect the horizontal axis at α with angle θ

IV. Branches depart poles at angle ϕ and arrive at zeros at angle ψ

V. Breakout points occur at roots of the derivative of L(s)

Now that we understand the application, let's look at the program!

Program Features

1. **OOP Design**
   - Access clean organization
   - Allows us to generate multiple at a time
2. **Python Features**
   - List comprehension
   - Tuple packing/unpacking
   - Type-hints
3. **Imported helper libraries**
   - Numpy – efficient array operations, root solving
   - Matplotlib – plotting and graphing display
   - Tkinter – graphical user interface
   - Cmath – converting polar to complex rectangular coordinates
   - Sympy – derivatives & polynomials, root solving

```
class RLPlotting:
    def __init__(self, numerator, denominator):
        ...  # set class attributes

    def plot(self) -> tuple:
        ...  # create figure and axes
        return ax, fig

    ...  # more code here
```

4. Graphical User Interface

To create a GUI, we utilized the **Tkinter** library.

We implemented the following functions:

- **Prompt the user** to enter all coefficients of the numerator and denominator (of the Char. Eq.)
- **Print inside the window** the general information of the plot, including calculated critical points & angles
- **Display** generated plot(s) **outside the window,** preserving easy-to-navigate/save menu

This allows the user to generate and save multiple plots without having to restart the application.

5. Visual Plotting

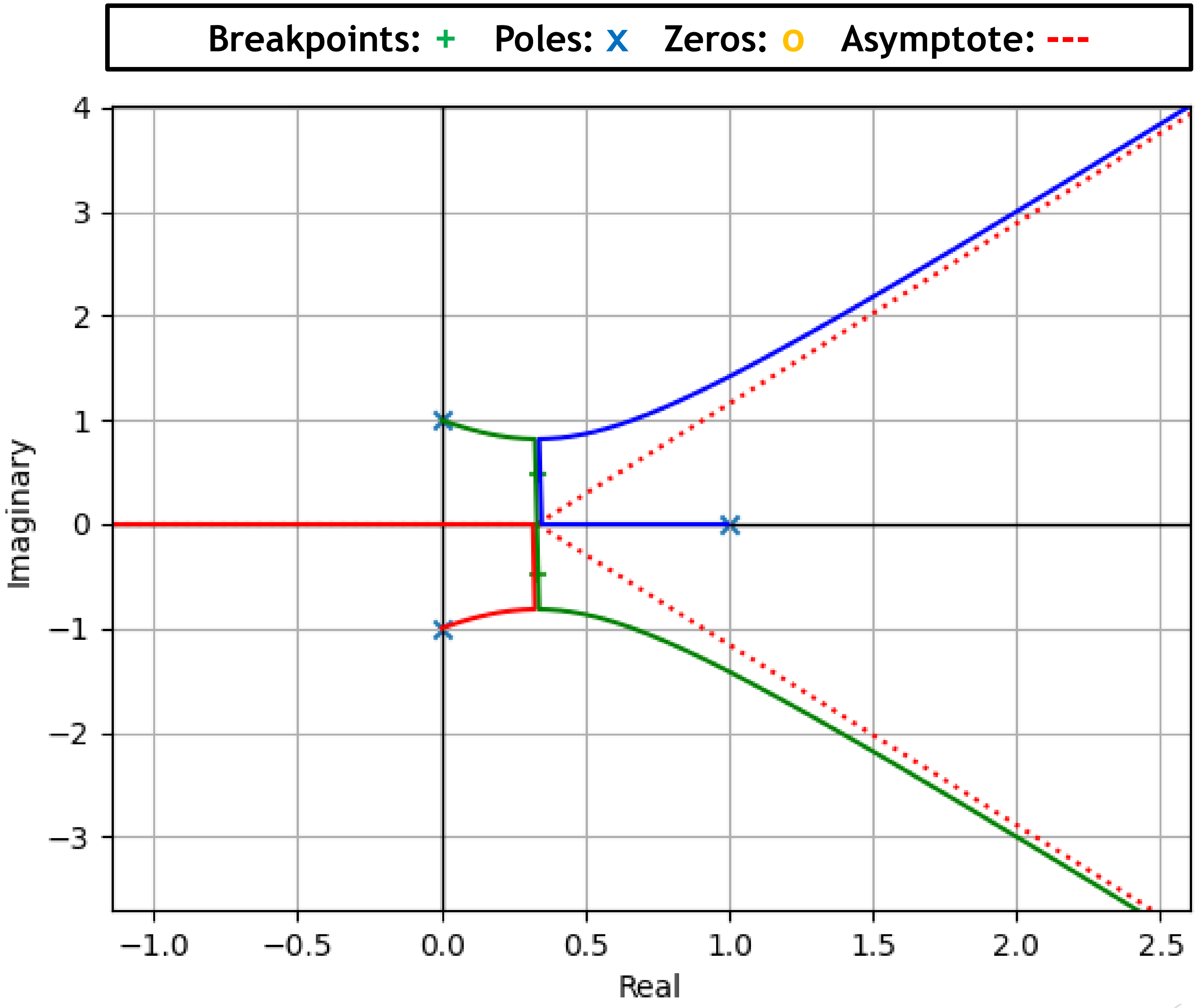
To display plots, we utilized the **Matplotlib** library.

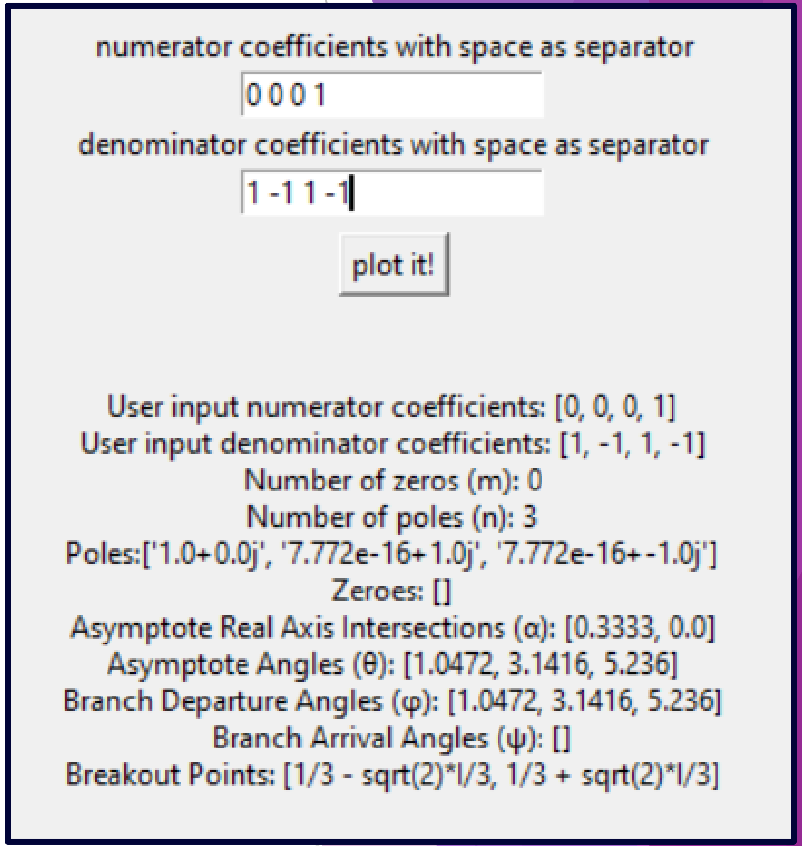
Our strategy was simple, to utilize formulas and compute the following:

- **Coordinate points** of poles, zeros, breakpoints, and asymptote intercepts
- **Asymptote lines**, particularly their angle from the horizontal axis
- **Collection of points** that together draw branches

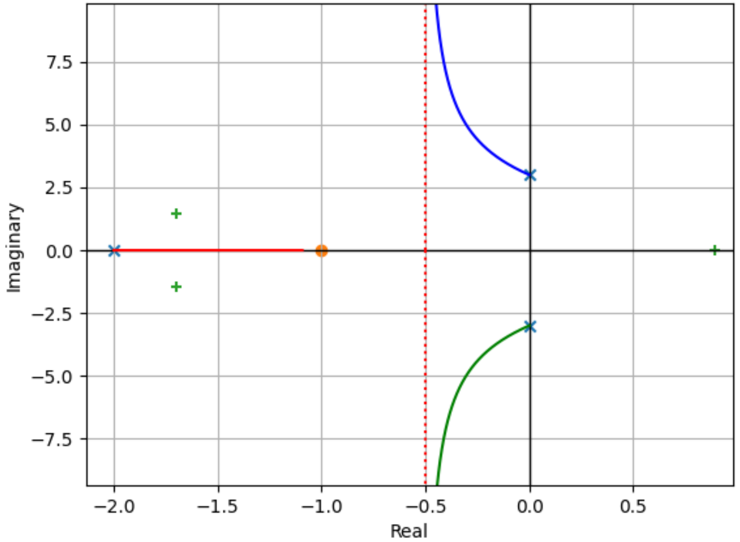
Then format that information so we can give it to Matplotlib for plotting.

Let's see some plots!

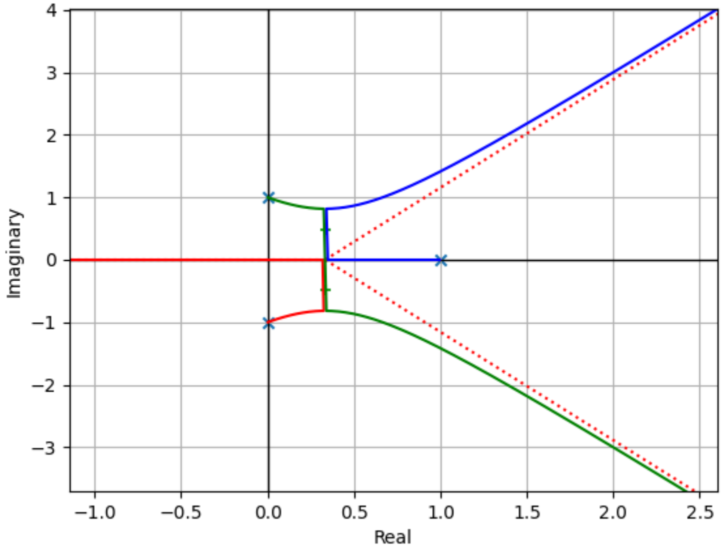

Breakpoints: +
Poles: x
Zeros: o
Asymptote: ---
Imaginary
Real
4
3
2
1
0
−1
−2
−3
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5


numerator coefficients with space as separator
0 0 0 1
denominator coefficients with space as separator
1 -1 1 -1
plot it!
User input numerator coefficients: [0, 0, 0, 1]
User input denominator coefficients: [1, -1, 1, -1]
Number of zeros (m): 0
Number of poles (n): 3
Poles:['1.0+0.0j', '7.772e-16+1.0j', '7.772e-16+-1.0j']
Zeroes: []
Asymptote Real Axis Intersections (α): [0.3333, 0.0]
Asymptote Angles (θ): [1.0472, 3.1416, 5.236]
Branch Departure Angles (φ): [1.0472, 3.1416, 5.236]
Branch Arrival Angles (ψ): []
Breakout Points: [1/3 - sqrt(2)*I/3, 1/3 + sqrt(2)*I/3]

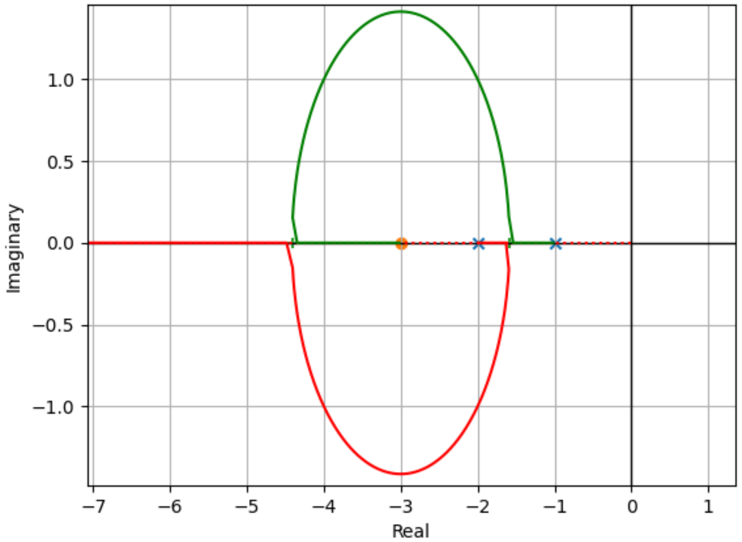
Breakpoints: + Poles: x Zeros: o Asymptote: ---



▶ $L(s) = (s+1) / (s^2+2)(s^2+9)$



▶ $L(s) = 1 / (s^2+1)(s-1)$



▶ $L(s) = (s+3) / (s^2+3s+2)$

ROOT LOCUS RULES *(that can be confirmed visually)*

1. Symmetric about the real/horizontal axis.
2. Branches begin at poles and end at zeros or approach infinity.
3. Breakpoints must be to the left of an odd # of poles/zeros.

To finish,
let's run a demo!

See our GitHub repo for more:
[RootLocus]