

rlocus.comp Generating the root locus via a computer

Matlab

In Matlab, the command `rlocus` generates a root locus plot from a linear system model object defined by `tf`, `zpk`, or `ss`. The `data cursor` has particularly useful information associated with it, including the `gain` required for the closed-loop pole of a given branch to be located at the selected point. Here are a few examples.

Example rlocus.comp-1

re: rlocus using zpk

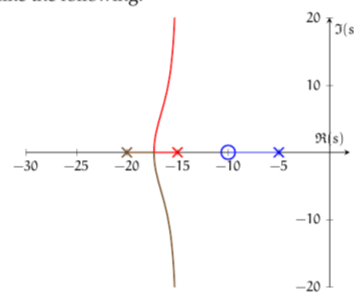
Use Matlab and `zpk` to generate a root locus plot from the open-loop transfer function

$$\frac{K(s+10)}{(s+5)(s+15)(s+20)}$$

The following code generates the root locus plot.

```
1 sys=zpk([-10],[5,15,20],1);
2 figure
3 rlocus(sys)
```

The figure it generates should look something like the following.



Example rlocus.comp-2

re: rlocus using tf and custom gains

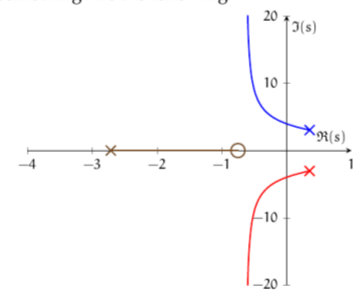
Use Matlab and `tf` to generate a root locus plot from the open-loop transfer function

$$\frac{4s+3}{s^3+2s^2+7s+25}$$

The following code generates the root locus plot.

```
1 sys=tf([4,3],[1,2,7,25]);
2 k=logspace(3.5,logspace(-1,3,50),Inf);
3 figure
4 rlocus(sys,k)
```

Note the use of custom gain values. Sometimes this is useful, especially if a specific gain value or range is important. In the code above, a specific gain of 3.5 is chosen; most gains (50 of them) are generated logarithmically from 10^{-1} to 10^3 , `logspace(-1,3,50)`; and the final gain of ∞ , `Inf`, is included. The array is sorted such that 3.5 is placed in the correct order in the array. The figure the code generates should look something like the following.



Python

The following was generated from a Jupyter notebook with the following filename and kernel.

```
notebook filename: python_root_locus.ipynb
notebook kernel: python3
```

We begin with the usual loading of modules.

```
import numpy as np # for numerics
import control as c # the Control Systems module!
import matplotlib.pyplot as plt # for plots!
```

Let's draw the root locus for the transfer function

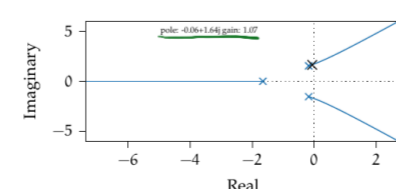
$$\frac{1}{s^3+2s^2+3s+4} \quad (1)$$

Defining a transfer function in Python is straightforward with the Control Systems module (documentation here).

```
transfer_function = c.TransferFunction(1,[1,2,3,4])
```

Now `transfer_function` is a transfer function object. We use the `root_locus` method of the Control Systems module.

```
pl = c.rlocus(transfer_function) # compute root locus
plt.show() # display the plot
```



Notice that double-clicking the locus yields a data cursor that gives the complex coordinate and corresponding gain! For instance, at the coordinate $-0.10 + j1.61$, the gain is 0.67. Therefore, to place a closed-loop pole at this location, we would choose $K = 0.67$.