E105: RLTOOL Tutorial

Andrew C. Smith *

7/30/2007

*Thanks to Sean Augenstein for his tutorial.

1. Introduction

• What is RLTOOL?

RLTOOL is a tool in MATLAB, that provides a GUI for performing Root Locus analysis on Single Input Single Output (SISO) systems, which are the class of systems we cover in E105. RLTOOL is a component of the broader SISO Design Tool in MATLAB, which can also do Bode and Nyquist analysis.

• Why are we using RLTOOL?

RLTOOL provides a fast, easy, and useful way to design compensators[1], and see their influence on the root locus drawn in the complex plane. With RLTOOL, we can quickly add, remove, and move compensator poles and zeros, or change proportional gains, and see the result on the location of the closed loop poles of the system.

In addition, RLTOOL can bring up plots of system responses to reference or disturbance inputs, so we can verify the performance of our compensator-plant system.

• How do I start RLTOOL?

RLTOOL can be called with a variety of command arguments. Type 'help rltool' at the MATLAB command prompt to see the options available. Typically, we'll create a plant transfer function 'G' in MATLAB first, then import it to RLTOOL with the command:

```
>> rltool(G)
```

[1]Note: the terms 'compensator' and 'controller' are equivalent

2. Tutorial

Step-by-step:

1 Open MATLAB

2 Let's use the second order system:

$$G(s) = \frac{1}{s(s+2)}$$

Create this plant system in MATLAB, using either the 'tf' or 'zpk' commands:

```
>> G = tf([0 0 1],[1 2 0])
```

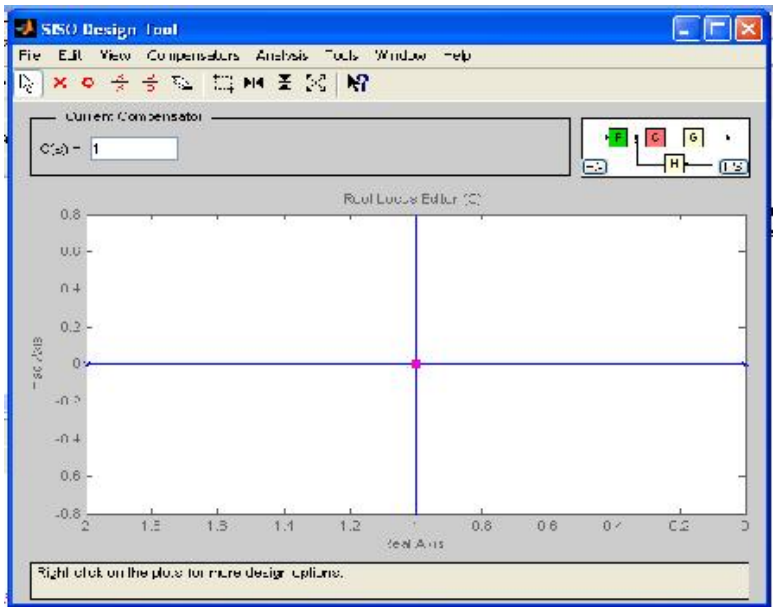
... or ...

```
>> G = zpk([],[0 -2],1)
```

3 Now begin RLTOOL and import the plant with:

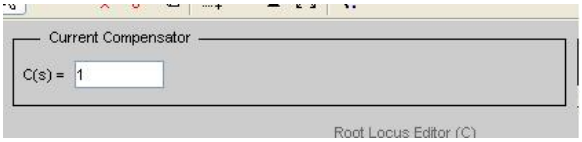
```
>> rltool(G)
```

You should see the GUI shown below appear.



4 Take a moment and acquaint yourself with the layout of the GUI. The plot of the Root Locus has crosses at the location of the plant open loop (OL) poles (at s = 0 and s = 2), and squares at the location of the closed loop (CL) system poles for the current compensator, C(s).

5 The compensator C(s) is displayed above the s-plane plot. It's initially simply a constant set to 1, which places both the CL poles at s = -1.



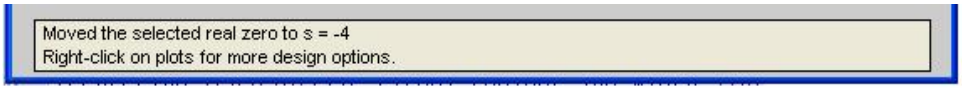
We can change the proportional gain by changing this value; change it to .5, hit enter, and observe the change in the location of the CL poles.

6 You can also change this proportional gain by clicking and dragging the CL poles. Note that the gain automatically changes in the 'Current Compensator" window.

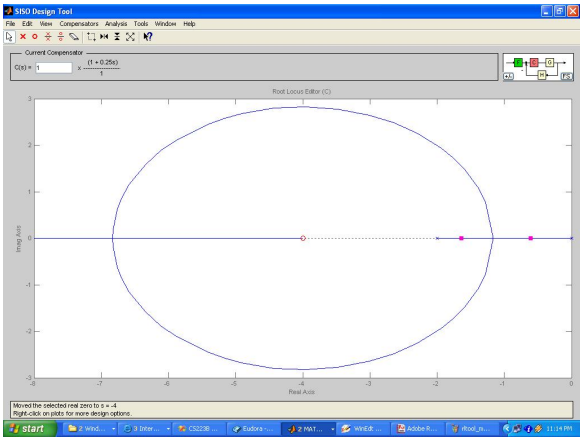
7 Now take a look at the toolbar above the "Current Compensator" window.

This toolbar allows you add compensator poles, add compensator zeros, or delete either. You can also use buttons to zoom in/out.

8 Let's add a zero at s = -4. Click on the circle button, then click somewhere on the real axis to place the zero. Now click and drag the zero to s = -4; note the message bar at bottom, which informs you where specifically you've moved the zero.

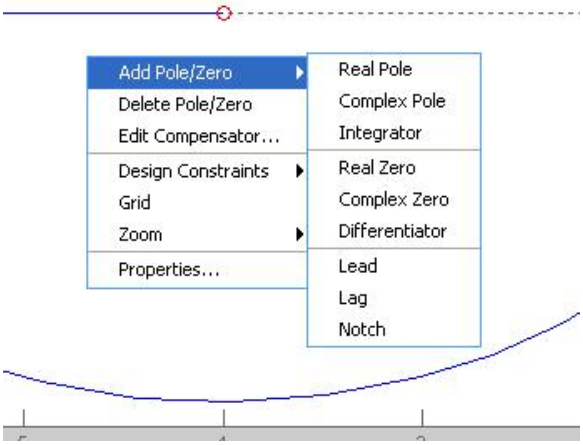


You're Root Locus should now look like this:



9 Aside: This has changed our control scheme from 'P' to 'PD'. Note the effect the added zero had. It moved the Root Locus more into the Left Half Plane, so in general the system will have better transient performance (better damping, shorter rise time), and is in general further from instability (i.e, the Right Half Plane).

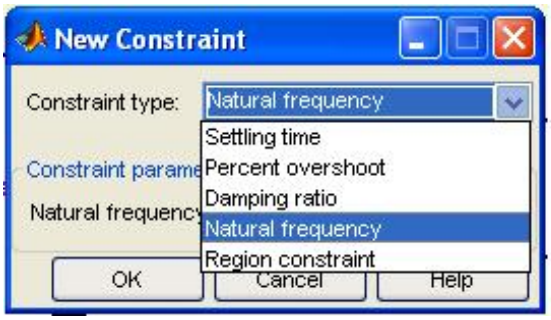
10 You can also add poles/zeros by right-clicking the plot:



Right click and add a a real pole, and drag it to s = -5. Note the effect the added pole has. The Root Locus has been pushed closer to the Right Half Plane, and thus closer to instability. This shows that integral control hurts our transient perfor-

mance (but does improve steady state, although you can't tell this from the Root Locus)

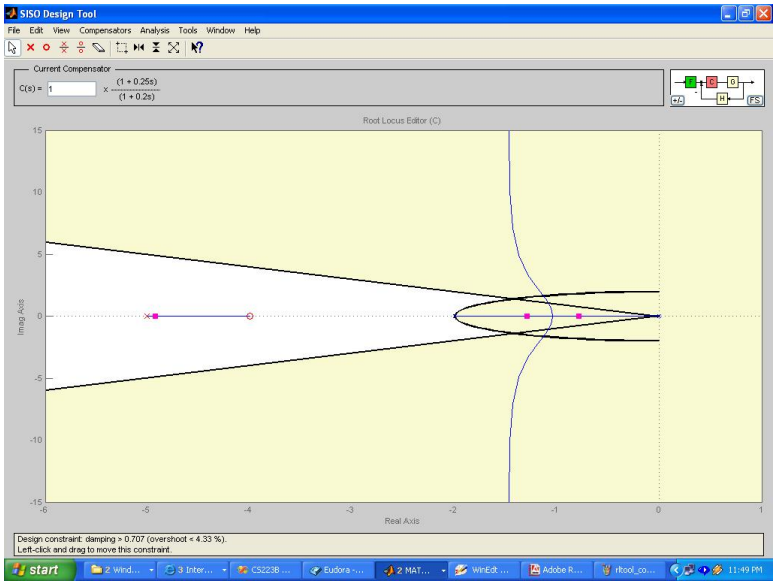
11 By right-clicking, you can also super-impose design constraints on the s-plane. Right click, go to "Design Constraints", then "New..." In the "Constraint type" pull-down menu, choose "Natural frequency".



Set a constraint on natural frequency to be at least 2.

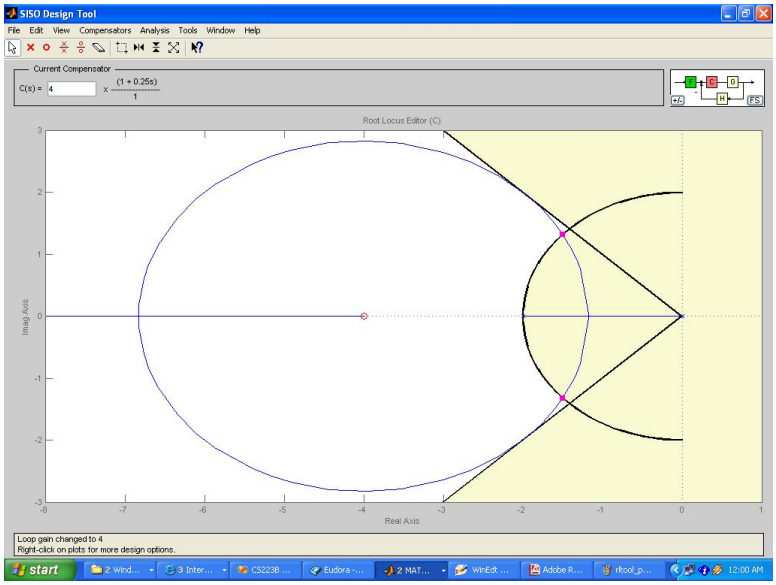
12 We can add all types of constraints. Add the constraint that $\zeta > .707$ by repeating step 11, and instead choosing "Damping ratio" in the pull-down menu.

Now our Root Locus should look like this:



Which shows we have a problem... we can't meet our specs with the compensator, *no matter what we set the proportional gain to*. That is, the two dominant roots of the closed loop system will never be in the valid region.

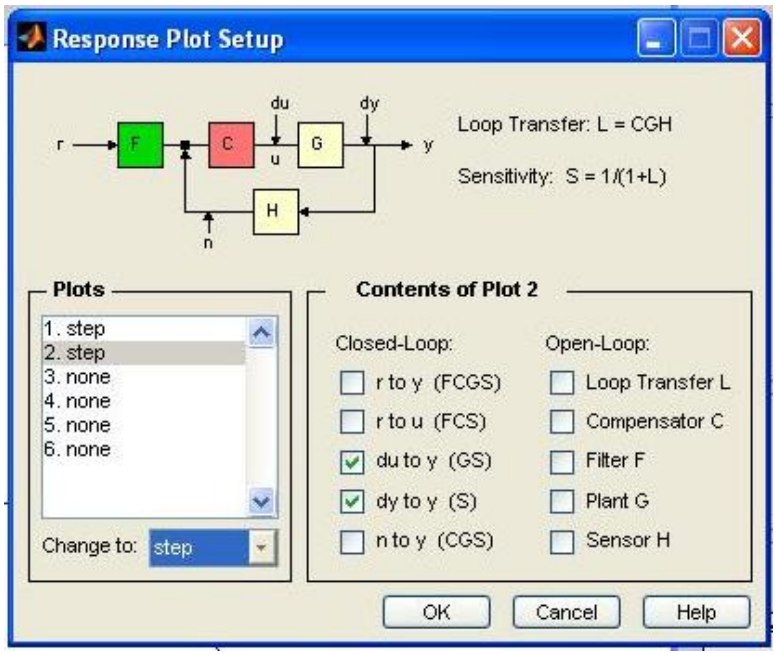
13 Let's delete the pole we added earlier (so we go back to having just PD control). Click on the eraser icon in the Toolbar, then click on the compensator pole (which we earlier placed at s = -5).



Now we have a compensator that can meet specs, provided we set the gain properly. Drag one of the CL poles (the squares on the Root Locus), and pull it until you are in the white region. Observe the compensator gain change as you move the CL poles; at the point you enter the region where both specs are met, the gain should be 4 (provided you're using the same plant and compensator from above).

14 We can do further things in RLTOOL. In the "Analysis" menu at top, you can see time responses of the system by selecting "Response to Step Command" and/or "Rejection of Step Disturbance". If you keep "Real-Time Update" checked, and go back to the Root Locus and make changes, you should see the time responses change in real-time.

15 To confirm what input/output relationships you are observing, select "Other Loop Responses..." from the "Analysis" menu; this tells you what the plots created in Step 14 are of, as well as allow you to look at many more signal relationships of the system (for example, input-to-actuator or disturbance-to-actuator). Play with the different types of plots you can create.



Conclusion:

RLTOOL is a useful application for applying Root Locus design techniques. There are more powerful capabilities than those covered here, but this is a good starting point. Experiment with RLTOOL and see what else it can do. Don't hesitate to ask the TA's if you have any questions!