Christopher Lum
lum@uw.edu

Lecture 07e
Using the Control System Designer in Matlab

Lecture is on YouTube

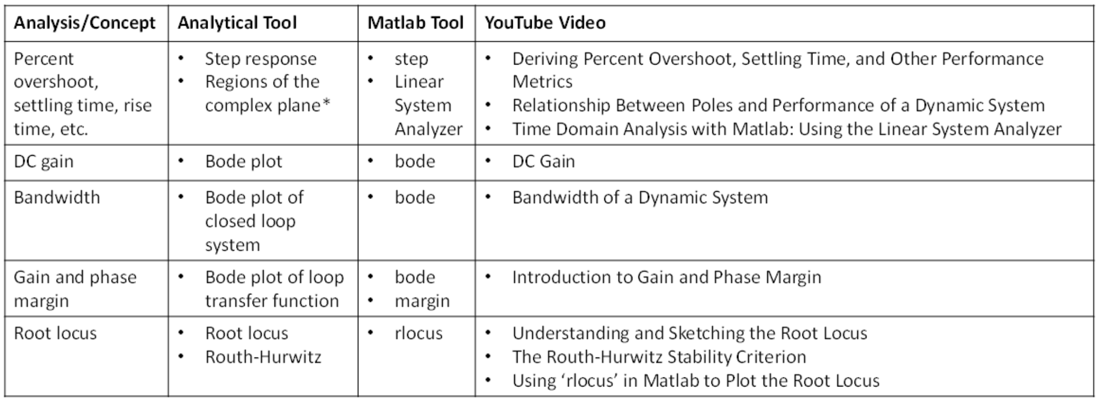
YouTube video entitled 'Using the Control System Designer in Matlab' covering this is located at https://youtu.be/RPzFLzKkQGs.

Outline

-Introduction
-Control Systems Designer

Introduction

We have developed and investigated many tools to analyze a linear control system.



Analysis/Concept	Analytical Tool	Matlab Tool	YouTube Video
Percent overshoot, settling time, rise time, etc.	• Step response • Regions of the complex plane*	• step • Linear System Analyzer	• Deriving Percent Overshoot, Settling Time, and Other Performance Metrics • Relationship Between Poles and Performance of a Dynamic System • Time Domain Analysis with Matlab: Using the Linear System Analyzer
DC gain	• Bode plot	• bode	• DC Gain
Bandwidth	• Bode plot of closed loop system	• bode	• Bandwidth of a Dynamic System
Gain and phase margin	• Bode plot of loop transfer function	• bode • margin	• Introduction to Gain and Phase Margin
Root locus	• Root locus • Routh-Hurwitz	• rlocus	• Understanding and Sketching the Root Locus • The Routh-Hurwitz Stability Criterion • Using 'rlocus' in Matlab to Plot the Root Locus

* = only valid/accurate for 2nd order systems

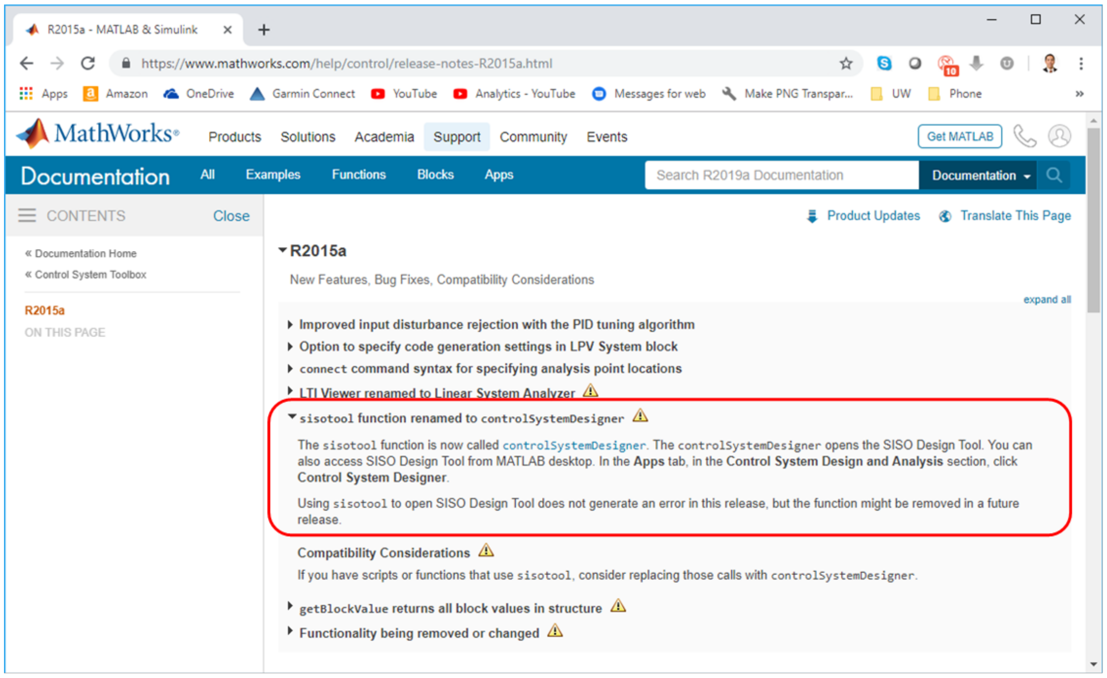
We now investigate a Matlab tool that can perform all these tasks (and more).

Control System Designer

The Control System Designer is an interactive GUI tool that allows you to design a control system for a linear system. It allows us to:

-view the root locus
-modify controller structure and parameters and view results/ramifications in real time
-view time domain response
-view frequency domain response

Note that this used to be called ‘sisotool’ but as of Matlab R2015a this has been renamed ‘controlSystemDesigner’. You may also have heard of this called ‘rltool’



R2015a

New Features, Bug Fixes, Compatibility Considerations

- Improved input disturbance rejection with the PID tuning algorithm
- Option to specify code generation settings in LPV System block
- `connect` command syntax for specifying analysis point locations
- LTI Viewer renamed to Linear System Analyzer ⚠
- `sisotool` function renamed to `controlSystemDesigner` ⚠

  The `sisotool` function is now called `controlSystemDesigner`. The `controlSystemDesigner` opens the SISO Design Tool. You can also access SISO Design Tool from MATLAB desktop. In the **Apps** tab, in the **Control System Design and Analysis** section, click **Control System Designer**.

  Using `sisotool` to open SISO Design Tool does not generate an error in this release, but the function might be removed in a future release.

  Compatibility Considerations ⚠

  If you have scripts or functions that use `sisotool`, consider replacing those calls with `controlSystemDesigner`.

- `getBlockValue` returns all block values in structure ⚠
- Functionality being removed or changed ⚠

Workflow with the Control System Designer

The procedure/workflow for analysis for using the Control System Designer is

1. Generate a transfer function of the plant, $G(s)$.
   Helpful functions: ss, tf, ss2tf, lti, minreal

2. Start the Control System Designer and load in the plant model.
   Helpful functions: controlSystemDesigner

3. Add design requirements. These can be constraints on poles locations, frequency domain response (bode plots), time domain response (step response)
   (in the Control System Designer) right click on root locus > Design Requirements > New...

4. Design your compensator/controller, $C(s)$, using the Control System Designer
   (in the Control System Designer) Designs > Edit Compensator...
   (in the Control System Designer) Manipulate root locus to achieve performance

5. Export your compensator/controller from the Control System Designer to the Matlab workspace.
   (in the Control System Designer) Export > Export Tuned Blocks > Export to workspace

6. Save your controller and session
   Helpful functions: save
   (in the Control System Designer) Save Session

7. Simulate the system to validate controller performance.
   Helpful functions: simulink

Example: DC Motor Position Control Using Proportional Control

Recall that previously, we showed that the transfer function for our DC motor (between velocity and armature voltage) was given by

$$G_P(s) = \frac{\theta(s)}{V_a(s)} = G_V(s)\,\frac{1}{s} = \frac{46\,163}{\left(s^2+1021\,s+4845\right)s} = \frac{46\,163}{s^3+1021\,s^2+4845\,s}$$

Suppose that we want to meet some requirements using a simple proportional controller. In this example, consider the following requirements

-Less than 40% overshoot
-Rise time of less than 1 second
-Settling time of less than 3 seconds
-Gain margin of 20 dB, Phase margin of 30 deg
-Bandwidth of greater than 5 rad/s

We can now use the Control System Designer to design a controller that meets all of these requirements simultaneously.

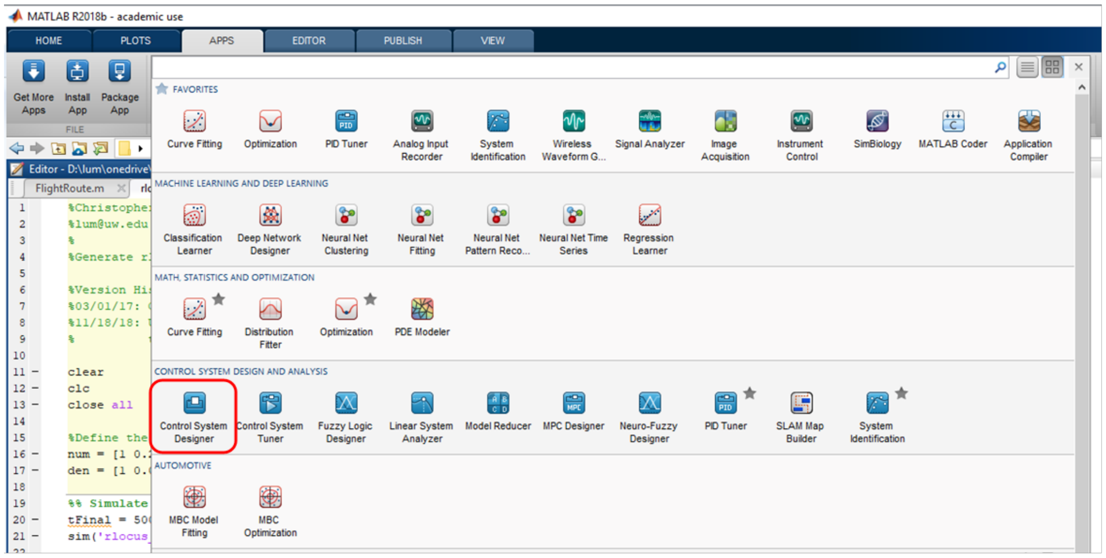
Step 1: Generate TF of Plant

We can generate a transfer function object of this in Matlab and then import into the Control System Designer using

```
%Define plant (see previous script/lecture
for deriving position TF)
GP_num = [46163];
GP_den = [1 1021 4845 0];

GP = tf(GP_num,GP_den);
```

Step 2: Start Control System Designer

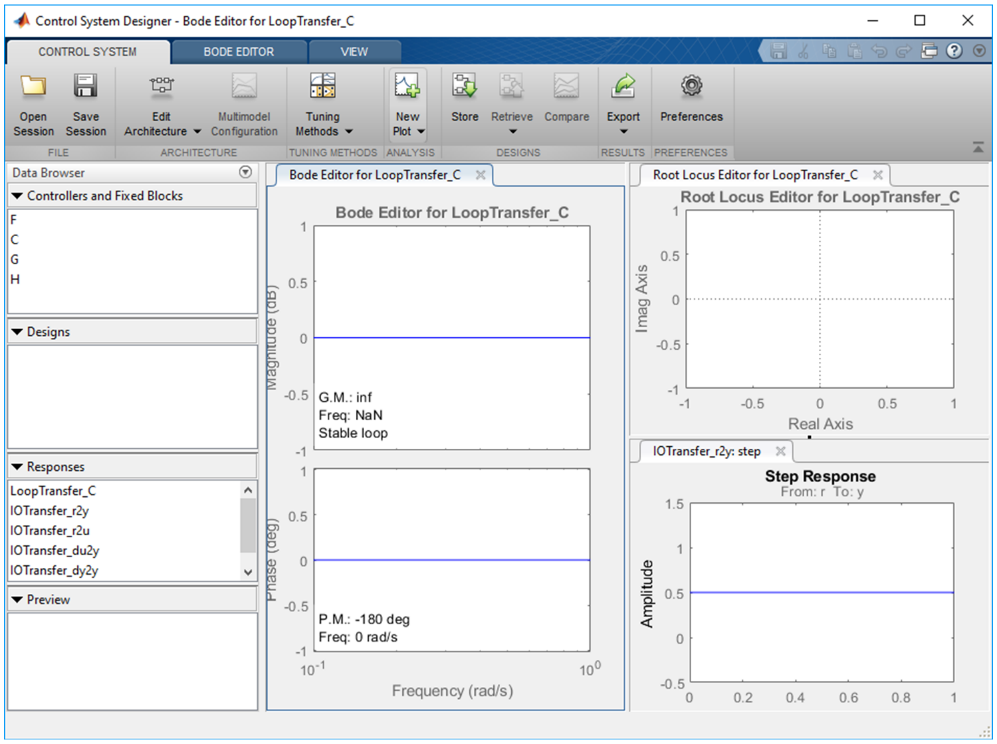
The ‘Control System Designer’ can be started via the ‘Apps’ ribbon in the main Matlab IDE (see below) or by typing ‘controlSystemDesigner’ in the Matlab command window.



```
%Christophe
%lum@uw.edu
%
%Generate r

%Version Hi
%03/01/17:
%11/18/18:
%

clear
clc
close all

%Define the
num = [1 0.
den = [1 0.

%% Simulate
tFinal = 50
sim('rlocus
```

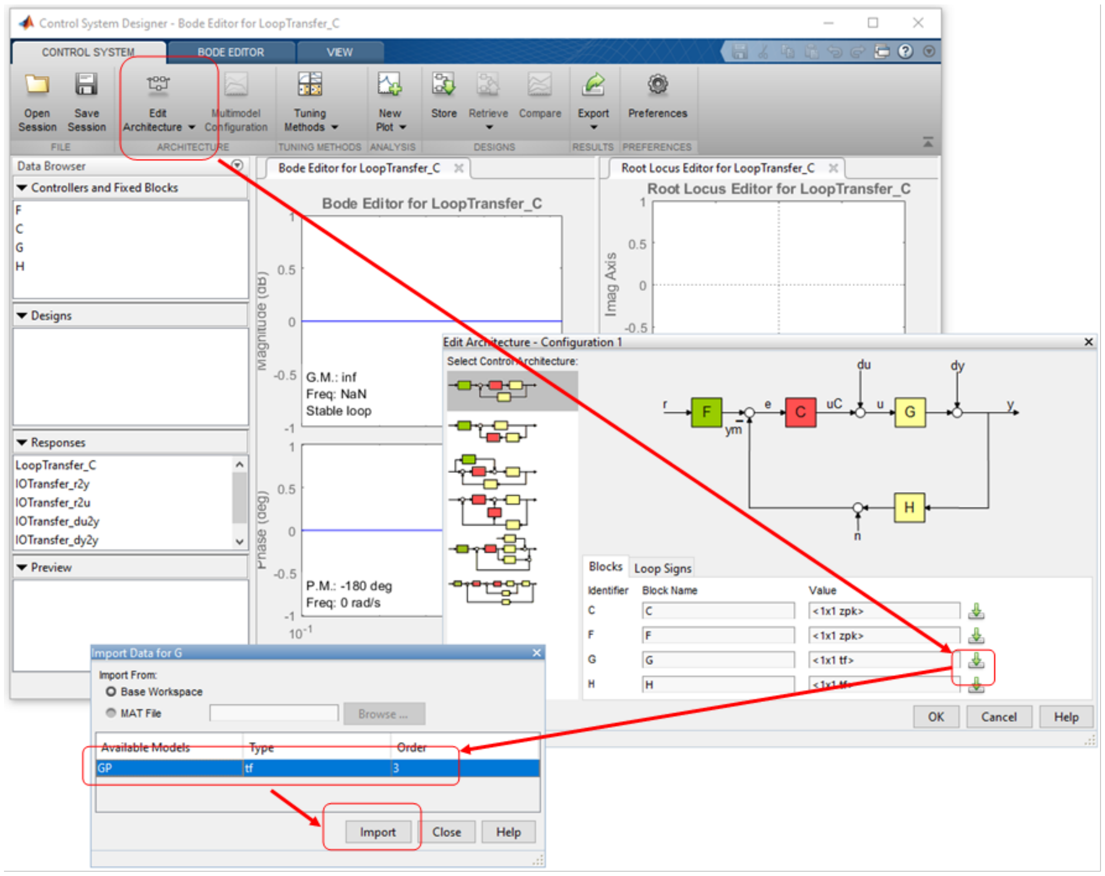
```
%start a controlSystemDesigner session
(note that you can directly import
%GP here if desired but we will do this
manually to illustrate the
%workflow).
controlSystemDesigner
```

This opens a blank session as shown below



We now import the transfer function $G_P(s)$ by selecting

Control System tab > Edit Architecture > click on the button next to ‘G’ > Import desired transfer function from base workspace



F
C
G
H

LoopTransfer_C
IOTransfer_r2y
IOTransfer_r2u
IOTransfer_du2y
IOTransfer_dy2y

G.M.: inf
Freq: NaN
Stable loop

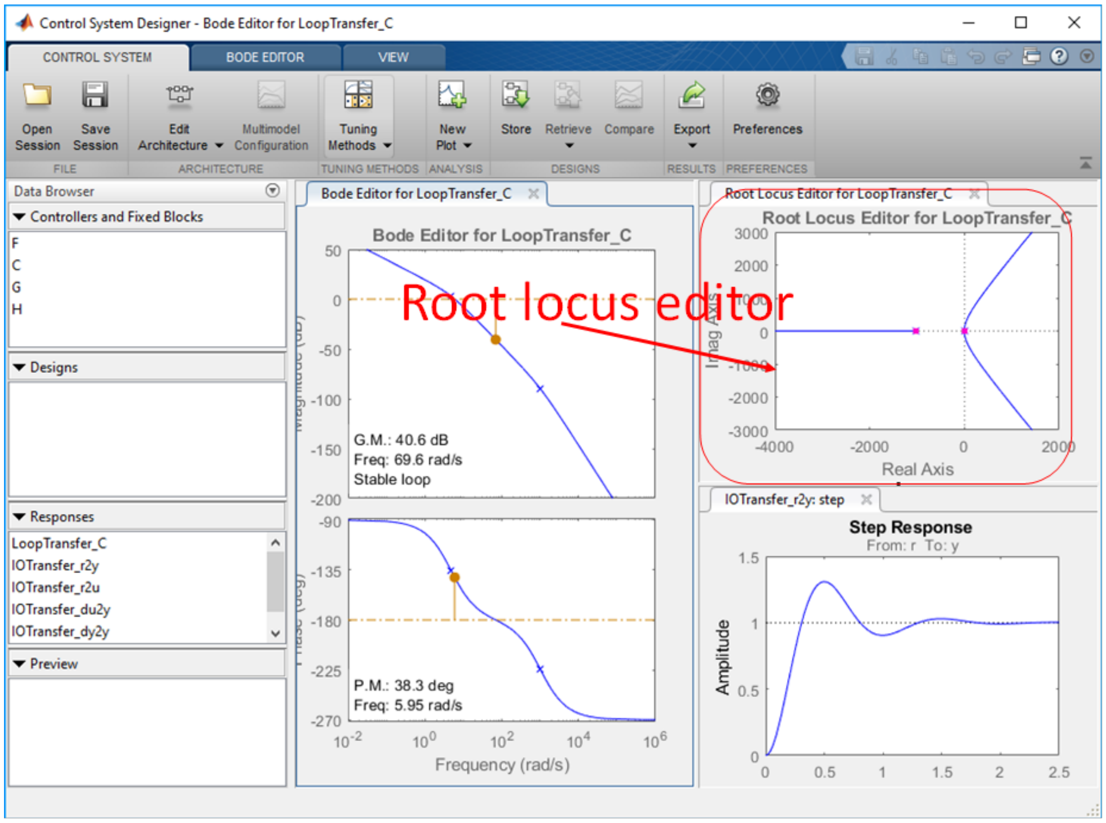
P.M.: -180 deg
Freq: 0 rad/s

Identifier	Block Name	Value
C	C	<1x1 zpk>
F	F	<1x1 zpk>
G	G	<1x1 tf>
H	H	<1x1 M...

Import From:
[✓ Base Workspace]
MAT File

Available Models	Type	Order
GP	tf	3

The resulting analysis tool with our DC motor velocity transfer function is shown below.



In general, you may want to maximize the root locus (highlighted in red above) as it will be the most useful for designing a controller.

Step 3: Add Design Requirements

<u>Displaying Appropriate Plots</u>

We first ensure that we have all the plots necessary to analyze the system. This includes

-Root Locus Editor

-Step Response (both r2y and r2u)

-Bode Editor (loop transfer function for gain/phase margin)

-Bode Editor (closed loop transfer function for bandwidth)

<u>Adding Design Requirements</u>

We can add design requirements in the step response

Right click on step response > Design Requirement > New

We can see the current performance in the step response

Right click on step response > Characteristics > chose appropriate ones > click on the dot

We can add design requirements in the root locus editor.

Right click on root locus > Design Requirement > New

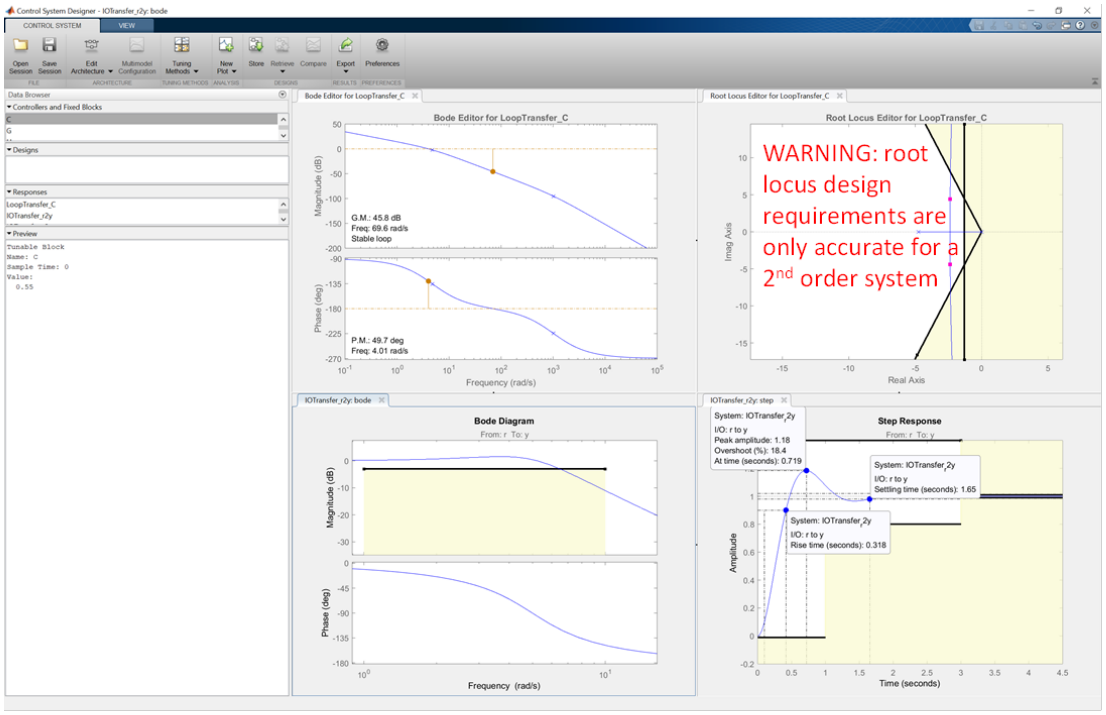
WARNING: If the system is more than 2nd order, these regions may be misleading. Use the step response to see actual performance

Step 4: Design

The magenta squares represent the system's closed loop poles. Note that you can grab and drag them to place them at locations on the root locus. In this case, it may be desirable to move them until the two poles meet just before they leave the real axis. By right clicking on the root locus and selecting 'Edit Compensator' we can see the gain that is required to achieve these closed loop poles.

Also note that you can store, retrieve, and compare designs (use the button in the ribbon).

A design that satisfies the performance requirements is shown below.



```
Tunable Block
Name: C
Sample Time: 0
Value:
  0.55
```

G.M.: 45.8 dB
Freq: 69.6 rad/s
Stable loop

P.M.: 49.7 deg
Freq: 4.01 rad/s

WARNING: root locus design requirements are only accurate for a 2nd order system

System: IOTransfer_r2y
I/O: r to y
Peak amplitude: 1.18
Overshoot (%): 18.4
At time (seconds): 0.719

System: IOTransfer_r2y
I/O: r to y
Settling time (seconds): 1.65

System: IOTransfer_r2y
I/O: r to y
Rise time (seconds): 0.318

In this case, we see that is we use a simple gain of

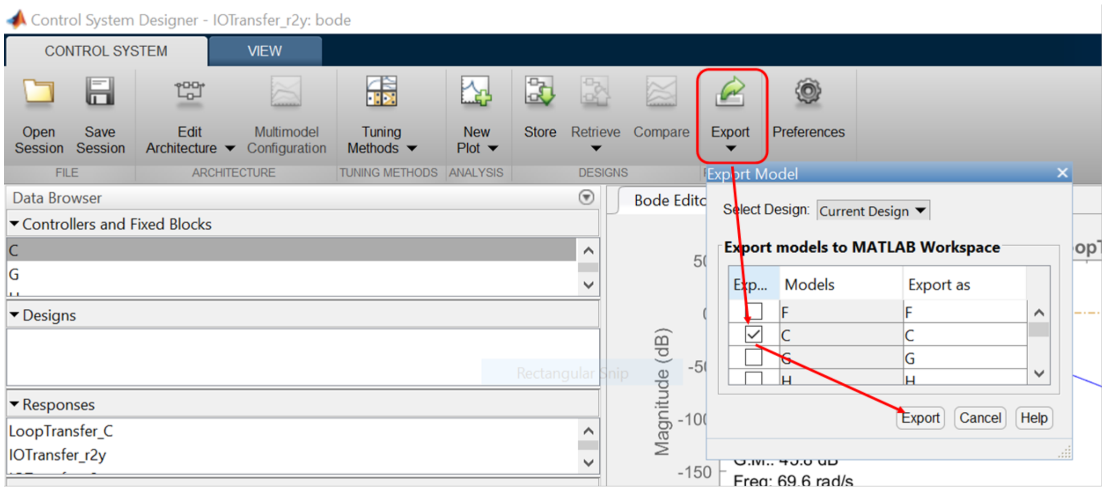
$$C(s) = 0.55$$

We should achieve the root locus displayed above

Step 5: Export Controller

If we are happy with the design, we can export the controller back to the Matlab workspace

Export > Export Tuned Blocks... >



Exp...	Models	Export as
☐	F	F
✓	C	C
☐	G	G
☐	H	H

Step 6: Save Controller

Once the variable ‘C’ is in the Matlab workspace, you can save it like any other variable

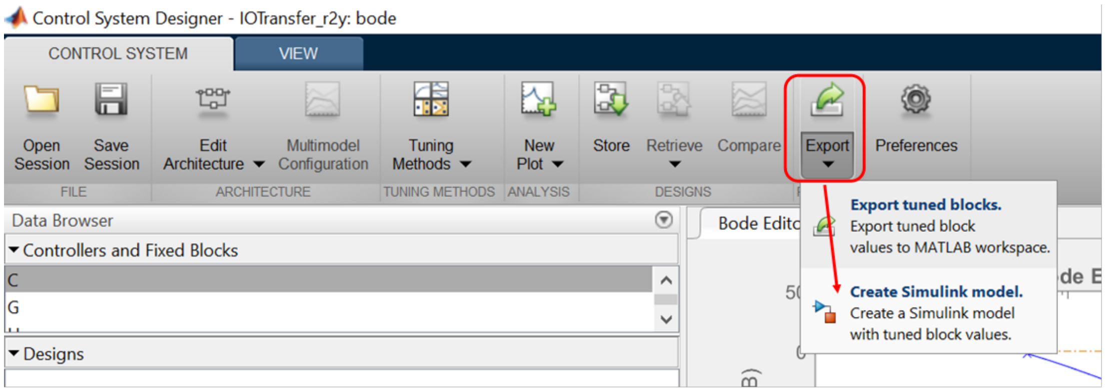
```
save(‘MyController.mat’,’C’)
```

You may want to also save your Control System Designer session using the ‘Save Session’ button. Note that when you re-open your session, you may need to rearrange the windows to display the information you would like to see (this does not seem to be saved in the .mat file)

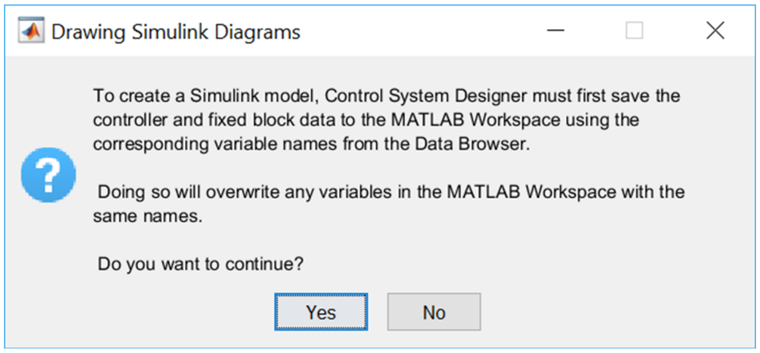
Step 7: Simulate to Verify Performance

We can build a Simulink model to verify performance.

A very nice feature is the ability to have the Control System Designer automatically generate a Simulink model of this system.



Note that this will export blocks to the workspace and may overwrite variables so be careful as you proceed



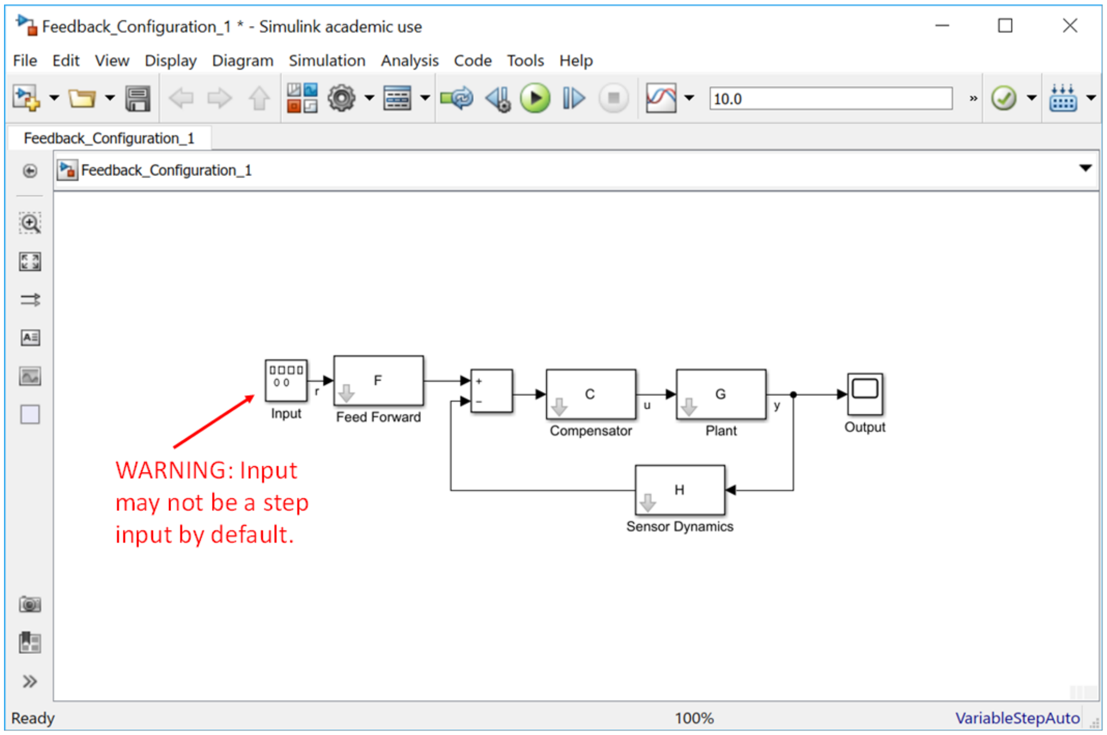
Drawing Simulink Diagrams

To create a Simulink model, Control System Designer must first save the controller and fixed block data to the MATLAB Workspace using the corresponding variable names from the Data Browser.

Doing so will overwrite any variables in the MATLAB Workspace with the same names.

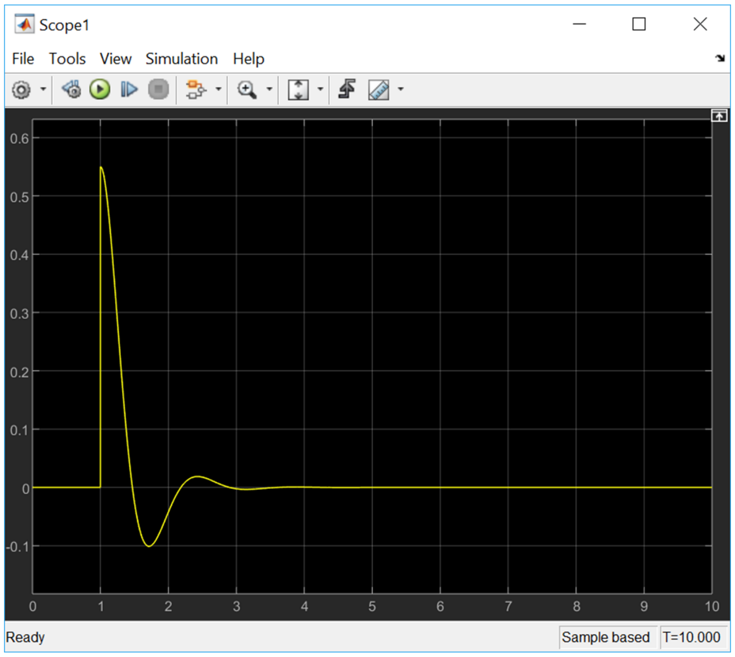
Do you want to continue?

This generates a Simulink model of the system. Note that for some reason, the input to the system is not set to a step input by default. As such, you may need to do some minor tuning to ensure



Feedback_Configuration_1 * - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
10.0
Feedback_Configuration_1
Feedback_Configuration_1
Input
r
F
Feed Forward
+
-
C
Compensator
u
G
Plant
y
Output
H
Sensor Dynamics
WARNING: Input may not be a step input by default.
Ready
100%
VariableStepAuto

Looking at the control response (signal u) we see that this is reasonable and should not saturate the controller.



Note that this simple gain controller is susceptible to steady state error in the face of external disturbances. We do not see this in simulation due to the nature of the system but we can see this in a real world system (show demo of DC motor). This leads us to examine adding an integral component to the control (next lecture).