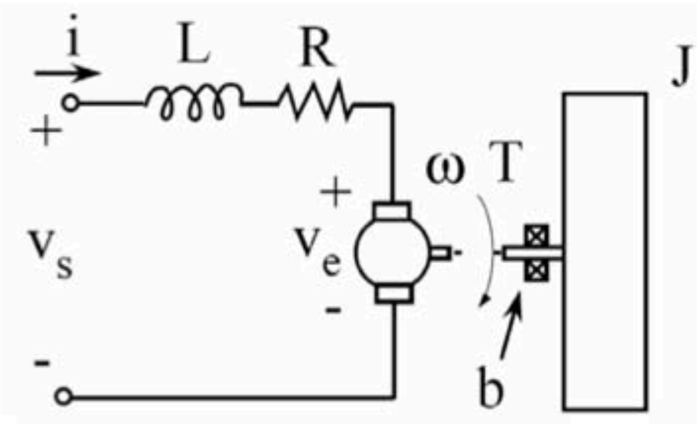
Today: using MATLAB® to model LTI systems

- 2nd order system example:
  - DC motor with inductance
    - derivation of the transfer function
    - transient responses using MATLAB
      - open loop
      - closed loop (with feedback)
  - Effect of feedback gain

DC motor system with non-negligible inductance



Recall **combined equations of motion**

$$\left.\begin{aligned} LsI(s) + RI(s) + K_v\Omega(s) &= V_s(s) \\ Js\Omega(s) + b\Omega(s) &= K_m I(s) \end{aligned}\right\} \Rightarrow$$

$$\left\{\begin{aligned} &\left[\frac{LJ}{R}s^2 + \left(\frac{Lb}{R} + J\right)s + \left(b + \frac{K_m K_v}{R}\right)\right]\Omega(s) = \frac{K_m}{R}V_s(s) \\ &(Js + b)\,\Omega(s) = K_m I(s) \end{aligned}\right.$$

Including the DC motor's inductance, we find

$$\left\{\begin{aligned} \frac{\Omega(s)}{V_s(s)} &= \frac{K_m}{LJ}\,\frac{1}{s^2 + \left(\dfrac{b}{J} + \dfrac{R}{L}\right)s + \left(\dfrac{bR + K_m K_v}{LJ}\right)} \\ \frac{I(s)}{V_s(s)} &= \frac{1}{R}\,\frac{\left(s + \dfrac{b}{J}\right)}{s^2 + \left(\dfrac{b}{J} + \dfrac{R}{L}\right)s + \left(\dfrac{bR + K_m K_v}{LJ}\right)} \end{aligned}\right.$$

Quadratic polynomial denominator
Second–order system

Theoretical tasks

- Derive the values of damping ratio ζ, natural frequency ω_n analytically based on the TF from the previous page

- Express the conditions such that the 2^{nd} order DC motor model would become over/under damped

PLEASE TURN IN THIS PAGE BEFORE YOU LEAVE THE LAB

MATLAB tasks /1

- Select values of R, L, b, J, K_m(=K_v) such that the open-loop (OL) response should be **overdamped**. Calculate the poles of the transfer function based on your choices, and compare the rise time of the response you get from MATLAB with the rise time that you expect from the theory. Make sure to "turn off" the feedback loop by setting the value of the gain to equal zero. Print out the MATLAB plots.

PLEASE TURN IN THIS PAGE BEFORE YOU LEAVE THE LAB

MATLAB tasks /2

- Select values of R, L, b, J, $K_m(=K_v)$ such that the open-loop (OL) response should be **underdamped**. Calculate the poles of the transfer function based on your choices, and compare the rise time, overshoot and damped oscillation frequency of the response you get from MATLAB with the corresponding values that you expect from the theory. Make sure to “turn off” the feedback loop by setting the value of the gain to equal zero. Print out the MATLAB plots.

PLEASE TURN IN THIS PAGE BEFORE YOU LEAVE THE LAB

MATLAB tasks /3

- Now set the value of the inductance L to zero, and keep the feedback loop "turned off" by setting the value of the gain to equal zero. How does the open-loop (OL) response change in MATLAB?

PLEASE TURN IN THIS PAGE BEFORE YOU LEAVE THE LAB

MATLAB tasks /4

- Restore the value of the inductance L to non-zero, and select the remaining values such that the open-loop (OL) response is again overdamped. Now "turn on" the feedback loop by gradually cranking up the value of the feedback gain. At some point, you will observe that the response becomes underdamped (oscillatory.) Print out the low-gain response (overdamped) and high-gain response (underdamped) and record the value of gain where the transition happens.

PLEASE TURN IN THIS PAGE BEFORE YOU LEAVE THE LAB

MATLAB tasks /5

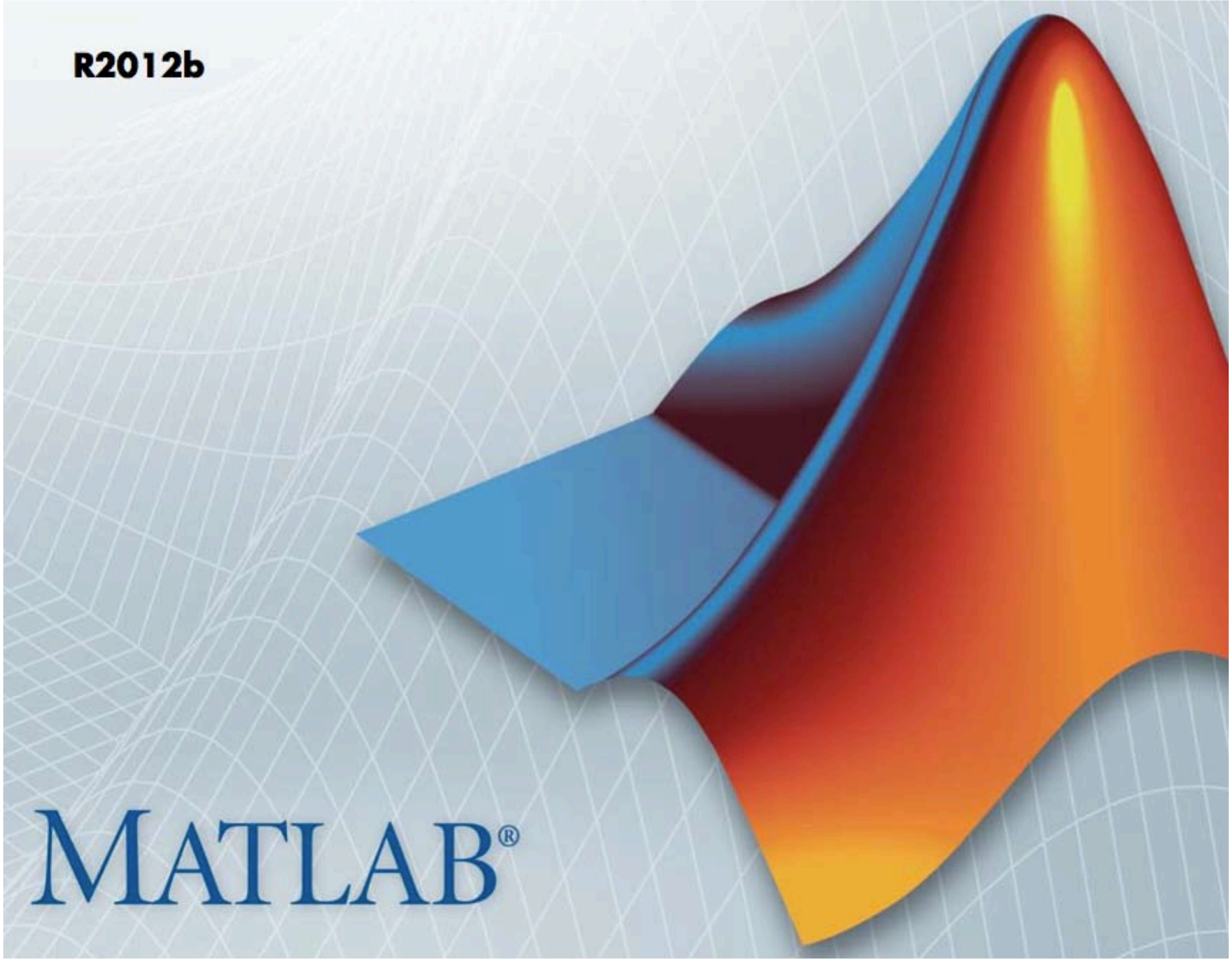
- In the last lab, we also had a second-order system where we observed the response change from over- to underdamped by cranking up the gain in the experimental flywheel system. Comment on the difference(s) between the model for the last lab's experiment and the model used in this lab's numerical exploration.

PLEASE TURN IN THIS PAGE BEFORE YOU LEAVE THE LAB

APPENDIX

MATLAB Control Systems Toolbox

Tutorial


R2012b
MATLAB®

MATLAB Linear Model Representation

- Transfer functions

$$H(s) = \frac{s+2}{s^2+s+10}$$

sys = tf ([1, 2] ,[1, 1, 10])

- State-space Models

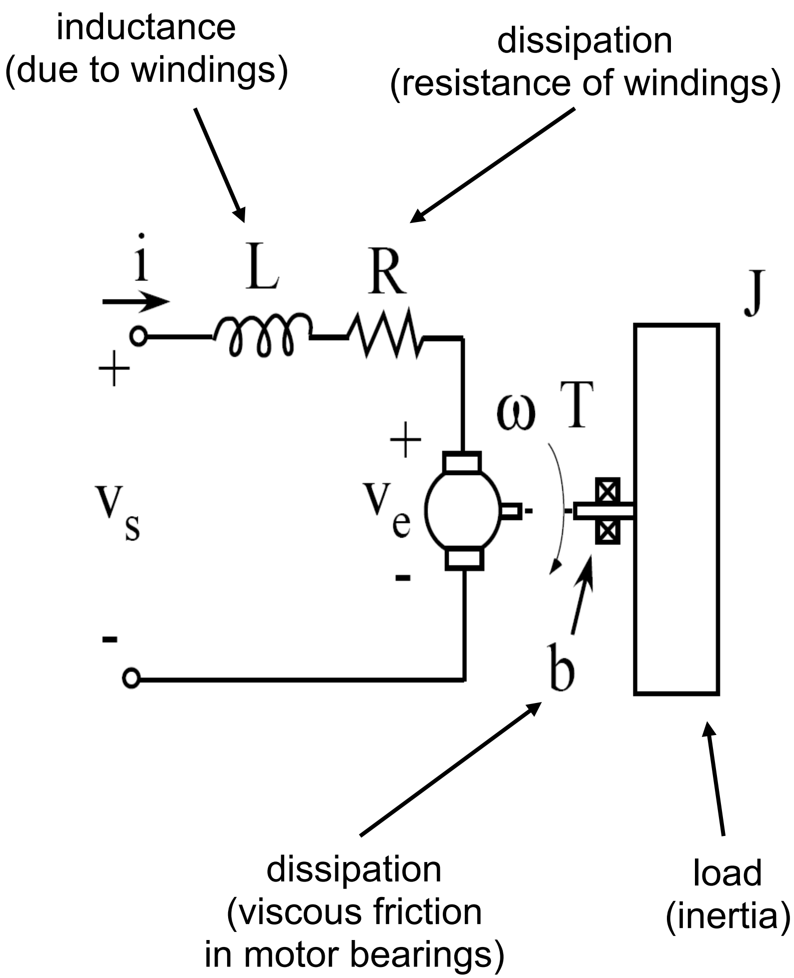
$$\frac{dx}{dt} = Ax + Bu$$

$$y = cx + Du$$

A, B, C, and D are matrices of appropriate dimensions, x is the state vector, and u and y are the input and output vectors respectively.

- Note: There are also other more complex forms of linear systems

State-space System Representation Example ①



- Recall from Lecture 5

Equation of motion – Electrical

$$\text{KCL:} \quad v_s - v_L - v_R - v_e = 0$$

$$\Rightarrow v_s - L\frac{di}{dt} - Ri - K_v\omega = 0$$

Equation of motion – Mechanical

$$\text{Torque Balance:} \quad T = T_b + T_J$$

$$\Rightarrow K_m i - b\omega = J\frac{d\omega}{dt}$$

Combined equations of motion

$$L\frac{di}{dt} + Ri + K_v\omega = v_s$$

$$J\frac{d\omega}{dt} + b\omega = K_m i$$

State-space System Representation Example ②

- Reorganizing the system and write in matrix form

$$\begin{cases} L\dfrac{di}{dt} + Ri + K_v\omega = v_s \\ \\ J\dfrac{d\omega}{dt} + b\omega = K_m i \end{cases} \Longrightarrow \begin{cases} \dfrac{di}{dt} = -\dfrac{R}{L}i - \dfrac{K_v}{L}\omega + v_s \\ \\ \dfrac{d\omega}{dt} = \dfrac{K_m}{J}i - \dfrac{b}{J}\omega \end{cases}$$

$$\Longrightarrow \frac{d}{dt}\begin{bmatrix} i \\ \omega \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{K_v}{L} \\ \frac{K_m}{J} & -\frac{b}{J} \end{bmatrix}\begin{bmatrix} i \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} v_s$$

- Here our input is v_s and output is ω; we also have

$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} i \\ \omega \end{bmatrix} + [0] \cdot v_s(t)$$

- Now we have our A B C D matrices ready

State-space System Representation Example ③

- In MATLAB, we can represent the motor system using following command: (need to define all parameters first)

```
A = [-R/L -Kv/L; Km/J -b/J];
B = [1/L; 0];
C = [0 1];
D = [0];
sys_dc = ss(A,B,C,D)
```

- We can also convert this state-space system to transfer function or Pole/zero/gain form:

★ `sys_tf = tf(sys_dc)` %(to transfer function)

★ `sys_zpk = zpk(sys_dc)` %(to ZPK form)

- NOTE: The state-space representation is **best suited** for numerical computations and is **most accurate** for most cases.
-

LTI Objects and Manipulation

- **Control System Toolbox** software uses custom data structures called "*LTI objects*".
- The state-space model we have created for the DC motor is called an "*SS object*". There are also TF, ZPK, and FRD objects.
- "*LTI objects"* enable you to manipulate linear systems as single entities using "**get**" command in MATLAB, we can see the detailed entities.

get(sys_dc)

sys_dc.a=A_new

(This line allows you to manipulate individual quantities)

```
                a: [2x2 double]
                b: [2x1 double]
                c: [0 1]
                d: 0
                e: []
        StateName: {2x1 cell}
    InternalDelay: [0x1 double]
               Ts: 0
       InputDelay: 0
      OutputDelay: 0
        InputName: {''}
       OutputName: {''}
       InputGroup: [1x1 struct]
      OutputGroup: [1x1 struct]
             Name: ''
            Notes: {}
         UserData: []
```

Creating Multiple Transfer Functions

- Assume the three transfer functions are $\dfrac{s}{s+5}$, $\dfrac{s+1}{s+6}$ and $\dfrac{s+2}{s^2+s+5}$

- Collect all numerators and denominators in cells; use the following MATLAB command:

```
N = {[1, 0], [1, 1], [1, 2]};
D = {[1, 5], [1, 6], [1, 1, 5]};
sys = tf(N, D)
```

```
sys =

  From input 1 to output:
     s
   -----
   s + 5

  From input 2 to output:
   s + 1
   -----
   s + 6

  From input 3 to output:
       s + 2
   -----------
   s^2 + s + 5

Continuous-time transfer function.
```

Interconnecting Linear Models -- Arithmetic Operations

- Addition (parallel systems):

```
tf(1, [1 0]) + tf([1 1], [1 2]);
```

This line represents $\frac{1}{s} + \frac{s+1}{s+2} = \frac{(s+2)+s(s+1)}{s(s+2)}$

```
Transfer function:
 s^2 + 2 s + 2
--------------------
   s^2 + 2 s
```

- Multiplication (cascaded systems):

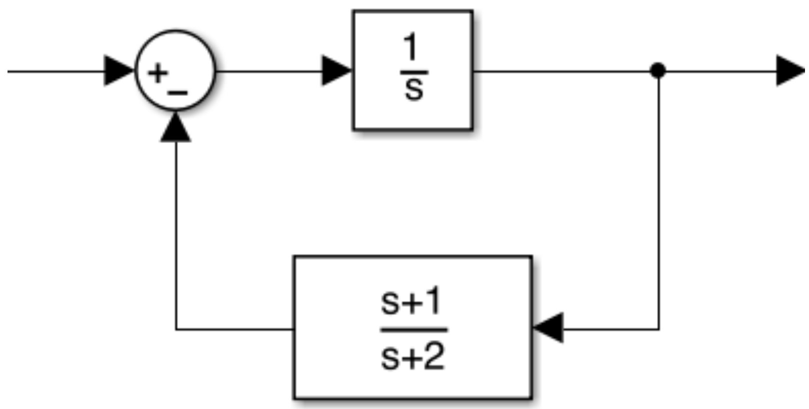
```
2 * tf(1,[1 0])*tf([1 1],[1 2]);
```

This line represents $2 \cdot \frac{1}{s} \cdot \frac{s+1}{s+2}$

```
Transfer function:
    2s+2
--------------
 s^2 + 2 s
```

Interconnecting Linear Models -- Feedback loop

- Example system:



sys_f = feedback(tf(1,[1 0]), tf([1 1],[1 2]))

```
Transfer function:
        s+2
-----------------
 s^2 + 3 s + 1
```

- **NOTE**: You can use the “lft” function to create more complicated feedback structures.

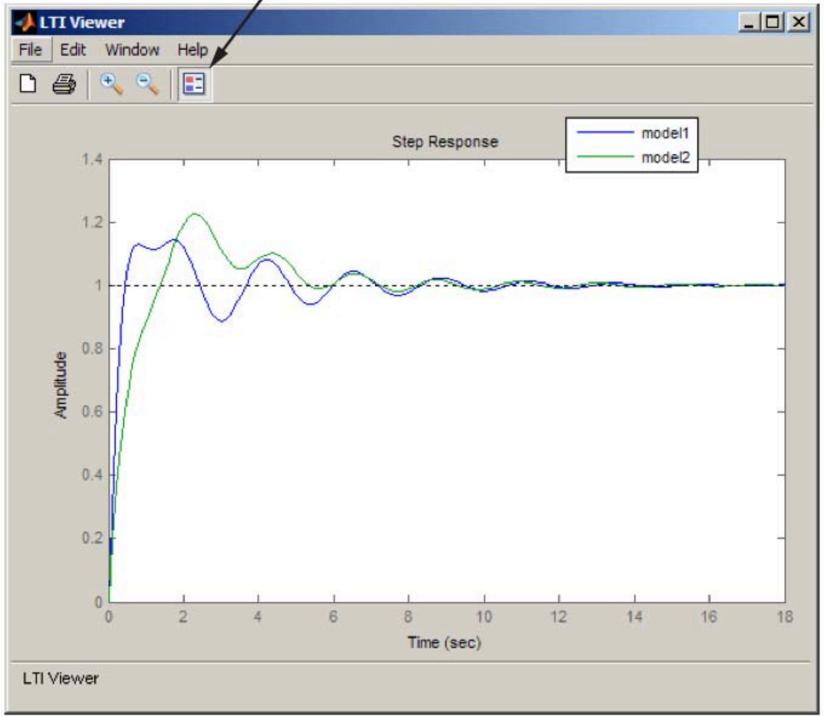
The LTI Viewer

- "LTI Viewer GUI allows" you to analyze the time- and frequency-domain responses of one or more linear models.
- Syntax:

ltiview(model1,model2,...,modelN)

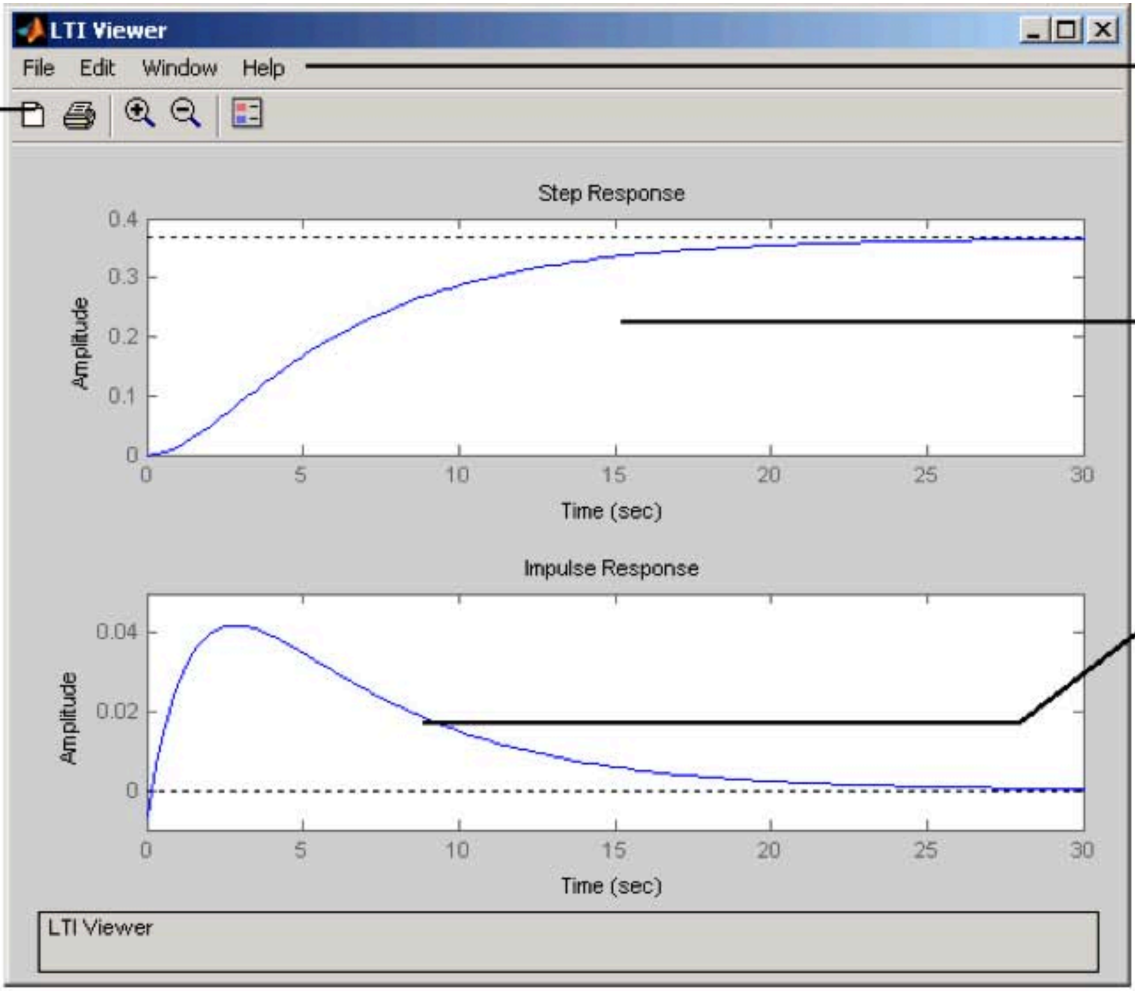
This syntax opens a step response plot of your models

Click to add a legend



General LTI Viewer Menus

Click on these icons to open a new viewer, print, and zoom in and out, respectively.



Use the File menu to import models and the Edit menu to delete existing ones.

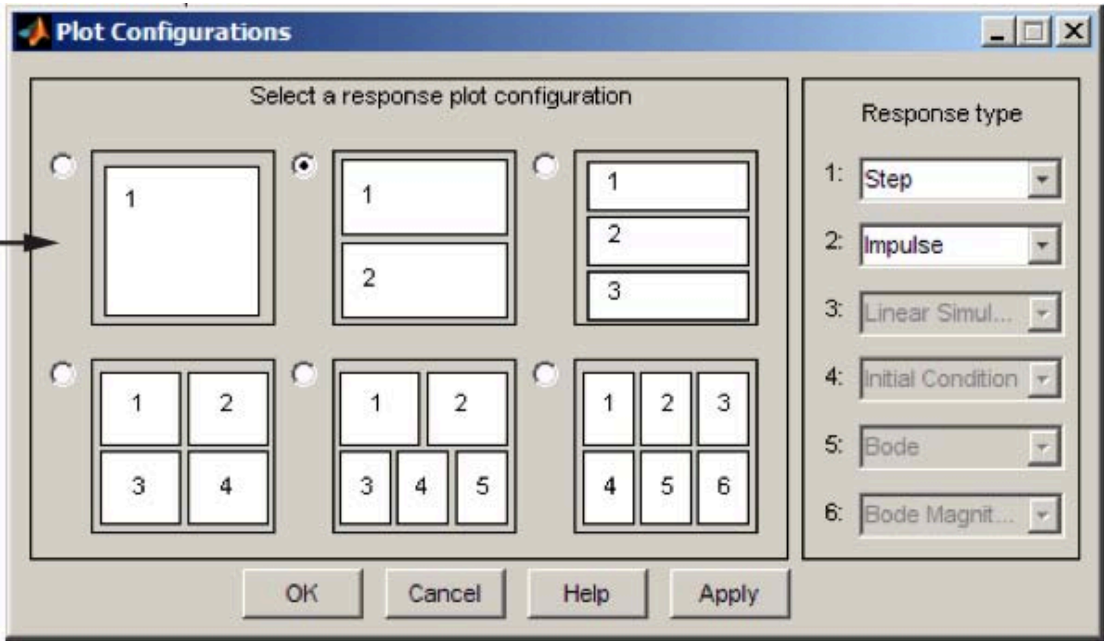
Right-click anywhere in a plot region to open a menu of options for that plot.

Left-click directly on a curve for information about the curve at that particular point.

Adding More Plots To LTI Viewer

- Select **Edit** > **Plot Configurations**.

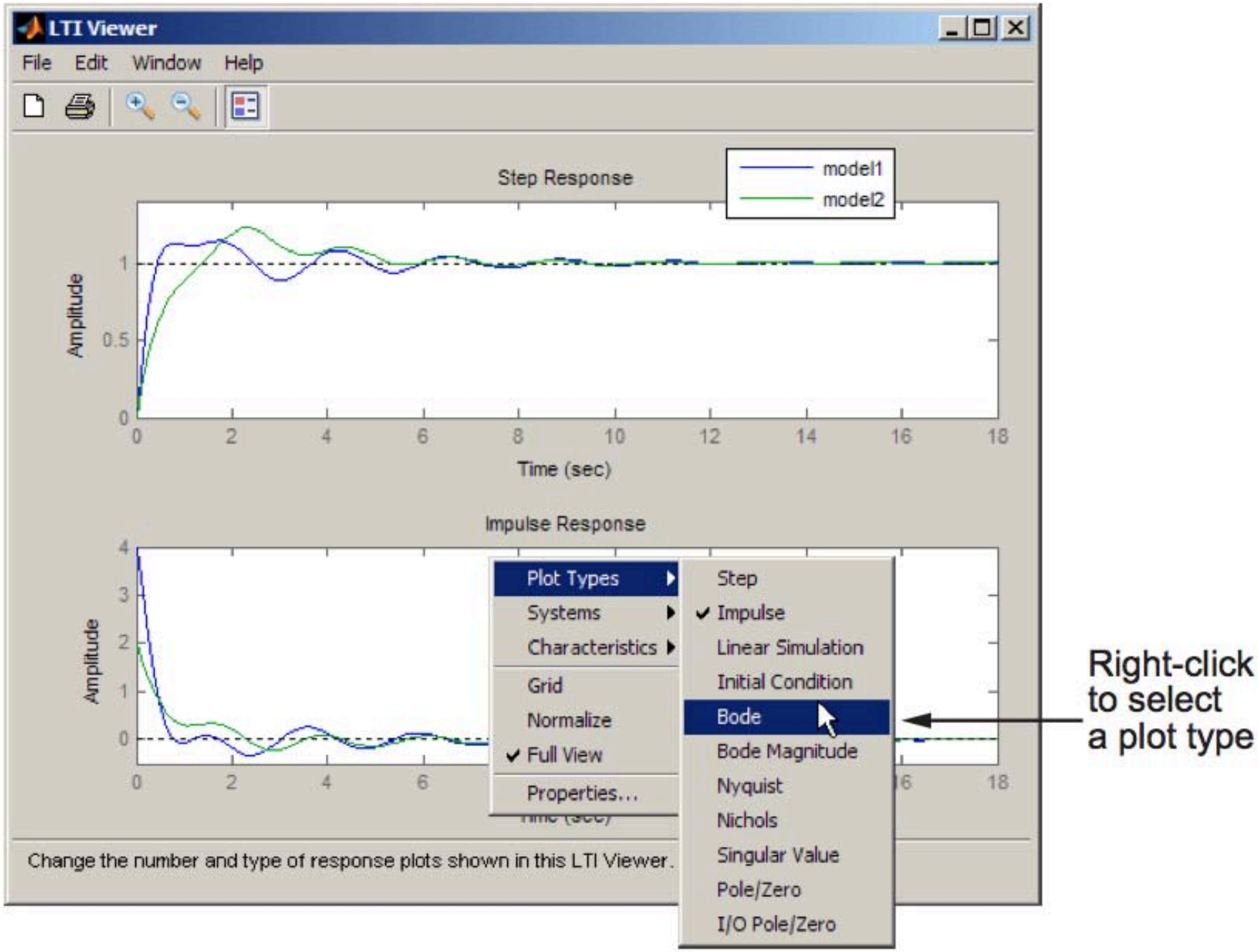
Select the number of plots to open



- You can also designate specific type of plots to view on the right hand side of this window.

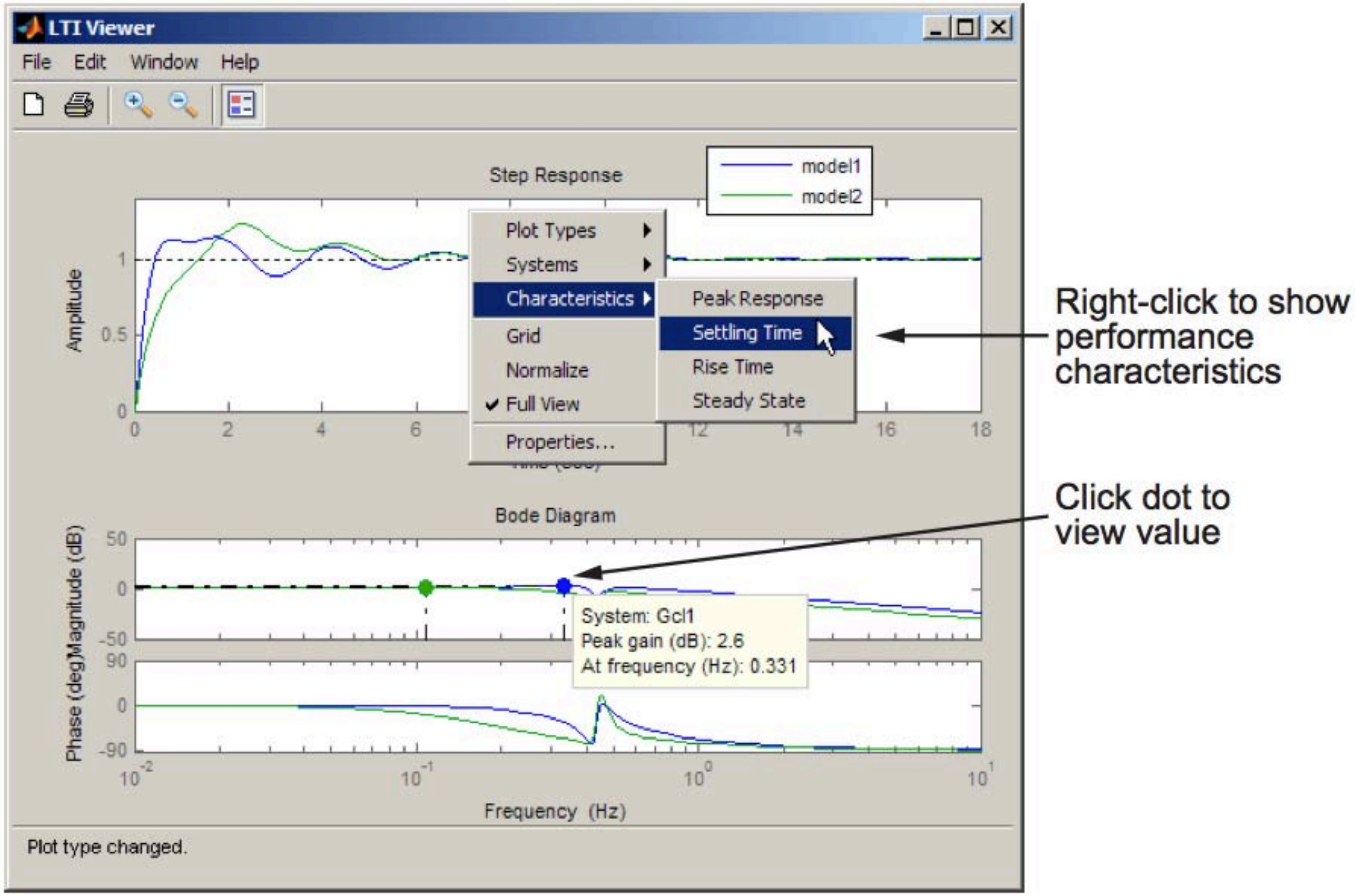
Change Plot Type

- To view a different type of response on a plot, right-click and select a plot type.



Analyze System Performance

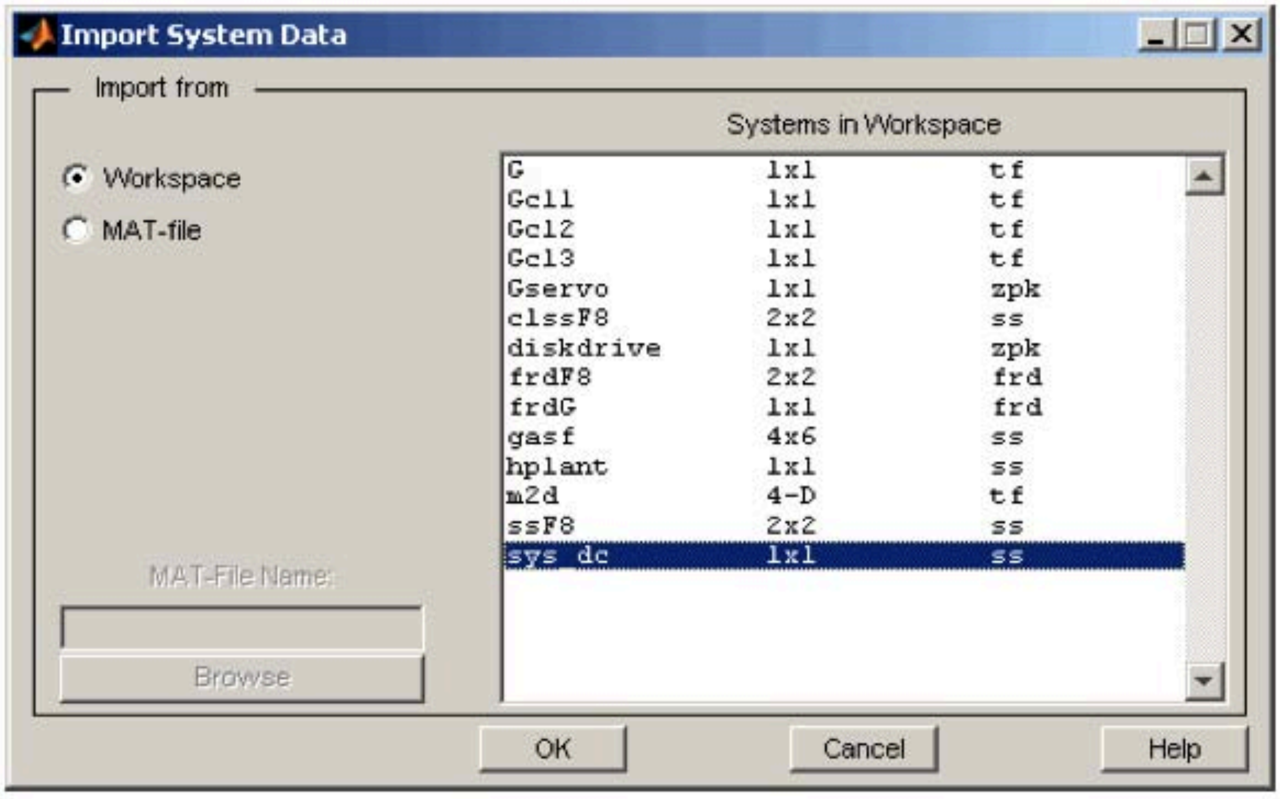
- Right-click to select performance characteristics.



- Click on the dot that appears to view the characteristic value.

Importing Models into the LTI Viewer

- Select "**Import**" under the "**File**" menu. This opens the Import System Data dialog box



- All the models available in your MATLAB workspace are listed

Alternative Command to Simulate Different Responses

- you can open the LTI Viewer and import systems from the MATLAB prompt.

  ltiview('step', sys)

  This syntax views the step response of our

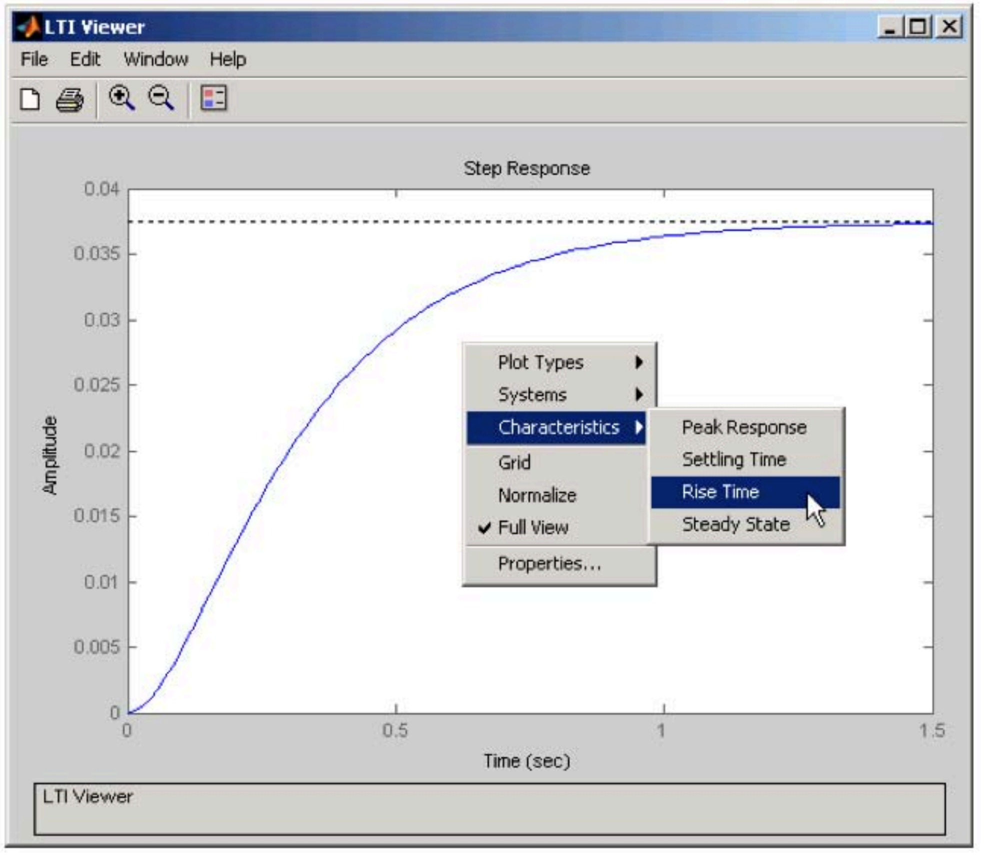
- More options:

'step'	--------	Step response
'impulse'	--------	Impulse response
'initial'	--------	Initial condition
'lsim'	--------	Linear simulation
'pzmap'	--------	Pole/zero map
'bode'	--------	Bode plot
'nyquist'	--------	Nyquist plot
'nichols'	--------	Nichols plot

- Multiple plots are allowed. Example: ltiview({'step';'impulse'},sys)

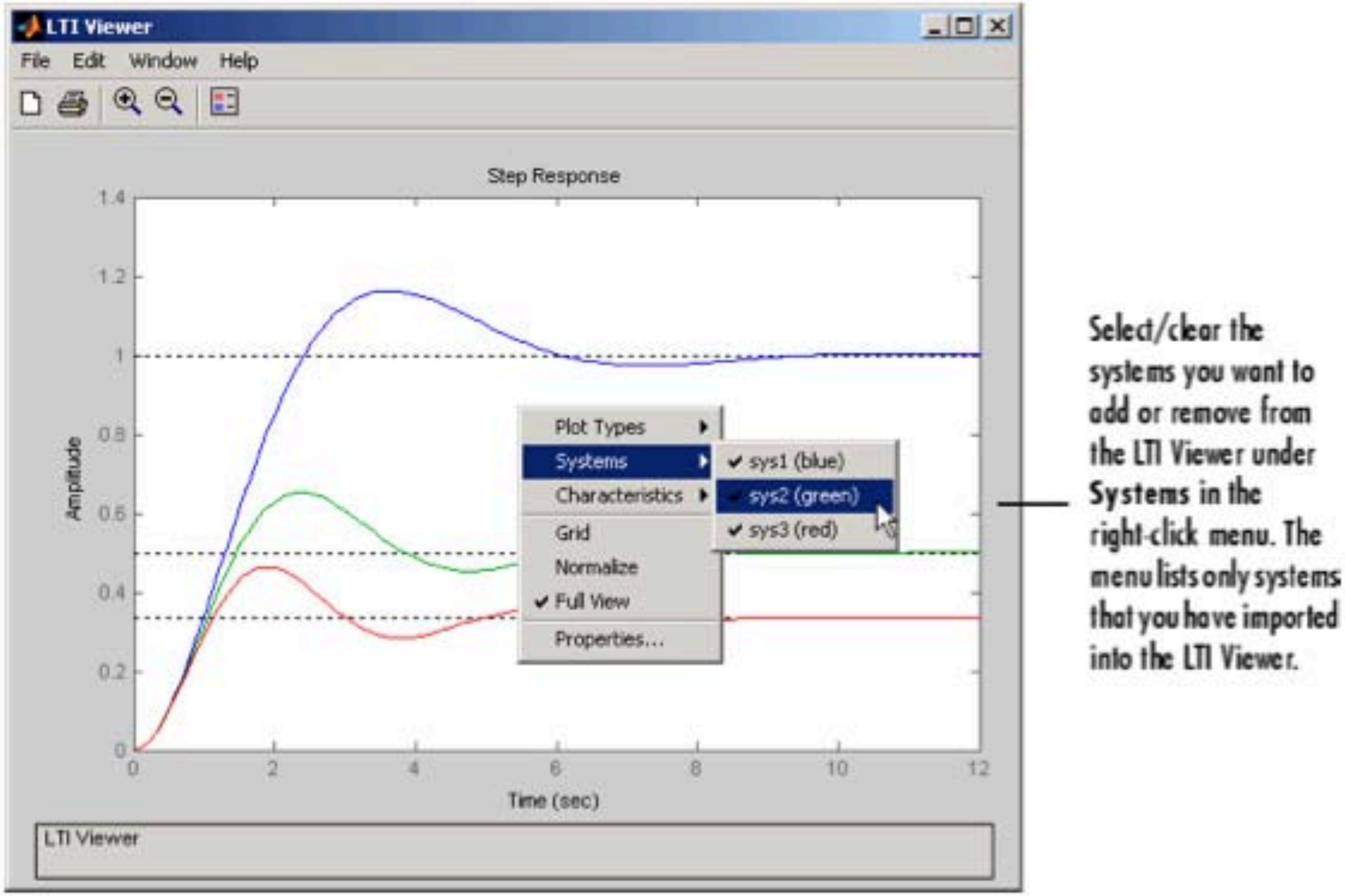
Displaying Response Characteristics

- Right-click on the plot
- Example: select **Characteristics > Rise Time**.



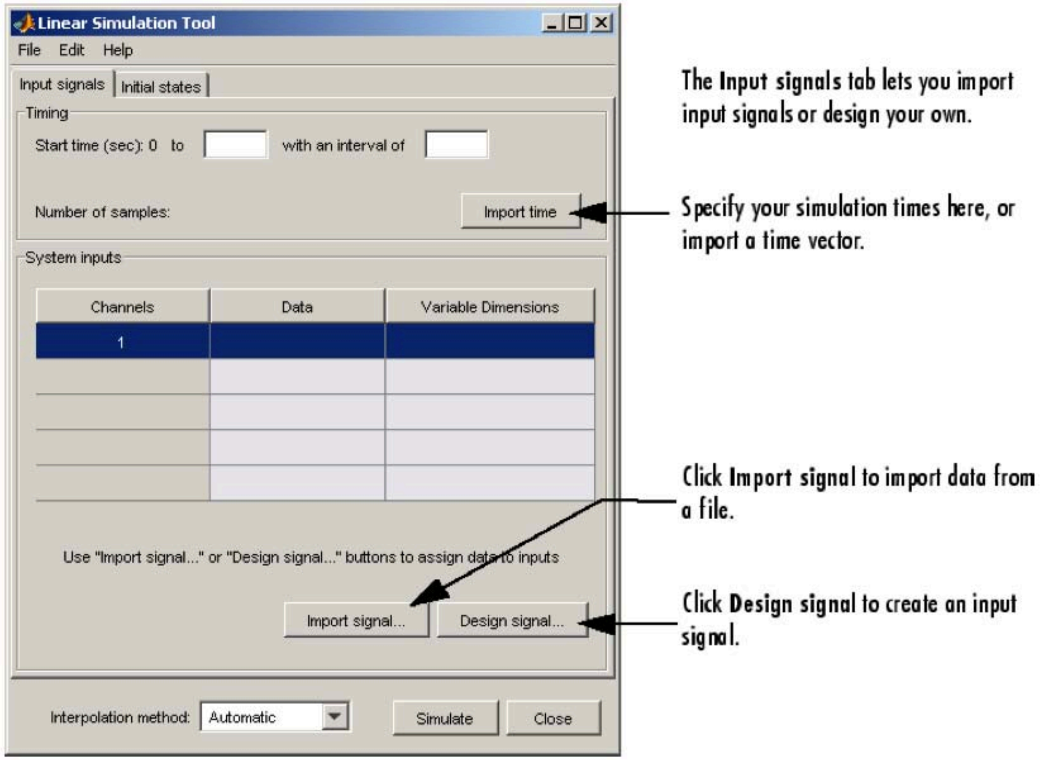
Toggling Model Visibility

- This figure shows how to clear the second of the three models using right-click menu options.



The Linear Simulation Tool

- In the LTI Viewer, **right-click** the plot area and select **Plot Types > Linear Simulation**.



- You can also use the "**lsim**" function at the MATLAB prompt:

  Syntax: **lsim(modelname)**

Using The Linear Simulation Tool

- Specify the **time duration** you want to simulate:
  - Import the time vector by clicking **Import time** (From workspace)
  - Enter the **end time** the **time interval** in seconds

- Specify the **input signal**
  - Click **Import signal** to import it from the MATLAB workspace or a file
  - Click **Design signal** to create your own inputs

- If you have a state-space model, you can specify the **initial conditions**
  - click the **Initial states** tab
- For a continuous model, select an interpolation method

Functions for Frequency and Time Response

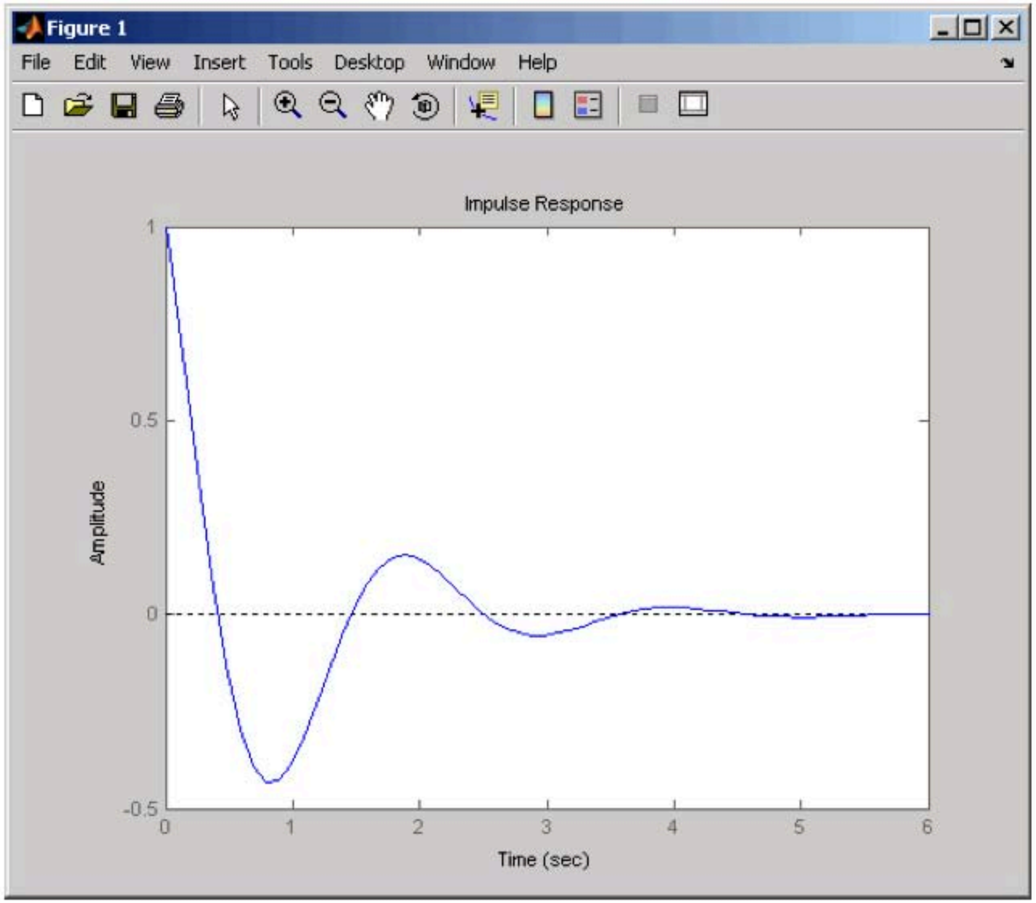
Functions	Description
bode	Bode plot
evalfr	Computes the frequency response at a single complex frequency (not for FRD models)
freqresp	Computes the frequency response for a set of frequencies
gensig	Input signal generator (for lsim)
impulse	Impulse response plot
initial	Initial condition response plot
iopzmap	Pole-zero map for each I/O pair of an LTI model
lsim	Simulation of response to arbitrary inputs
margin	Computes and plots gain and phase margins
nichols	Nichols plot
nyquist	Nyquist plot
pzmap	Pole-zero map
step	Step response plot

Using a Response Command

- Example:

```
h = tf([1 0],[1 2 10])
impulse(h)
```

- Plot:

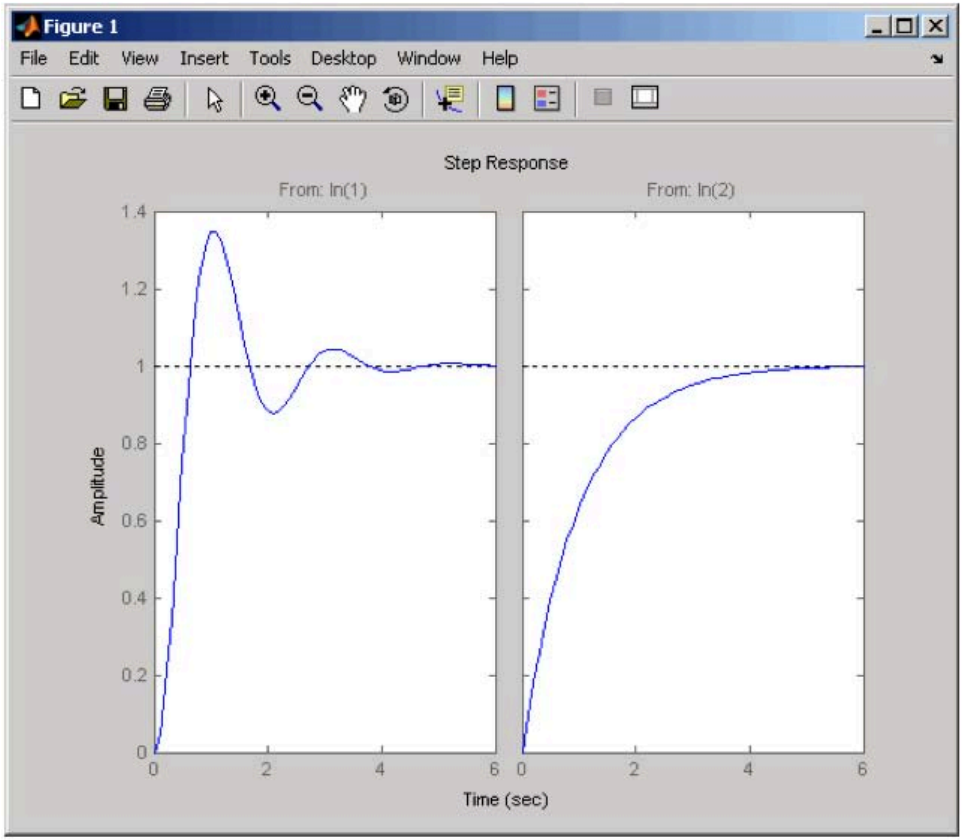


Response for Multiple Systems

- Command (Collect transfer functions into a vector array):

```
h = [tf(10,[1 2 10]) , tf(1,[1 1])]
step(h)
```

- Plot:



Alternative Command for Multiple Systems

- Syntax:

stepplot(sys1, 'r' , sys2, 'y--' , sys3, 'gx')

This
command generates a step plot (sys1 with
solid red lines, sys2 with yellow dashed lines,

impulseplot(sys1, 'r' , sys2, 'y--' , sys3, 'gx')

bodeplot(sys1, 'r' , sys2, 'y--' , sys3, 'gx')

- NOTE: Options for plot color and shape are optional (all in blue solid lines)

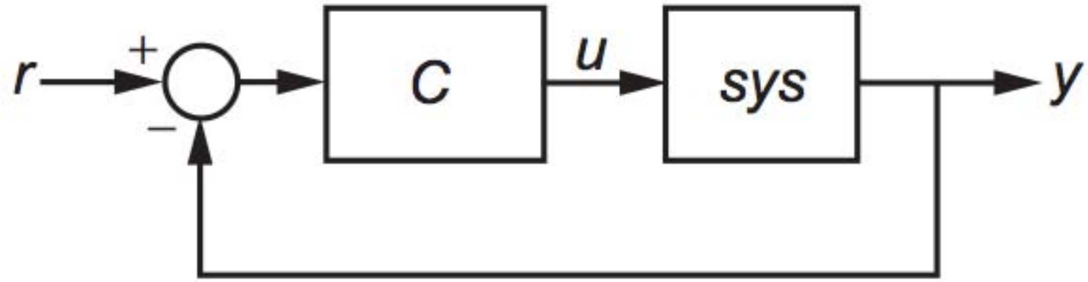
Example: stepplot(sys1, sys2, sys3, sys4, sysN)

Controller Design -- PID Tuner

- To launch the PID Tuner, use the pidtool command:

  pidtool(sys,'type')

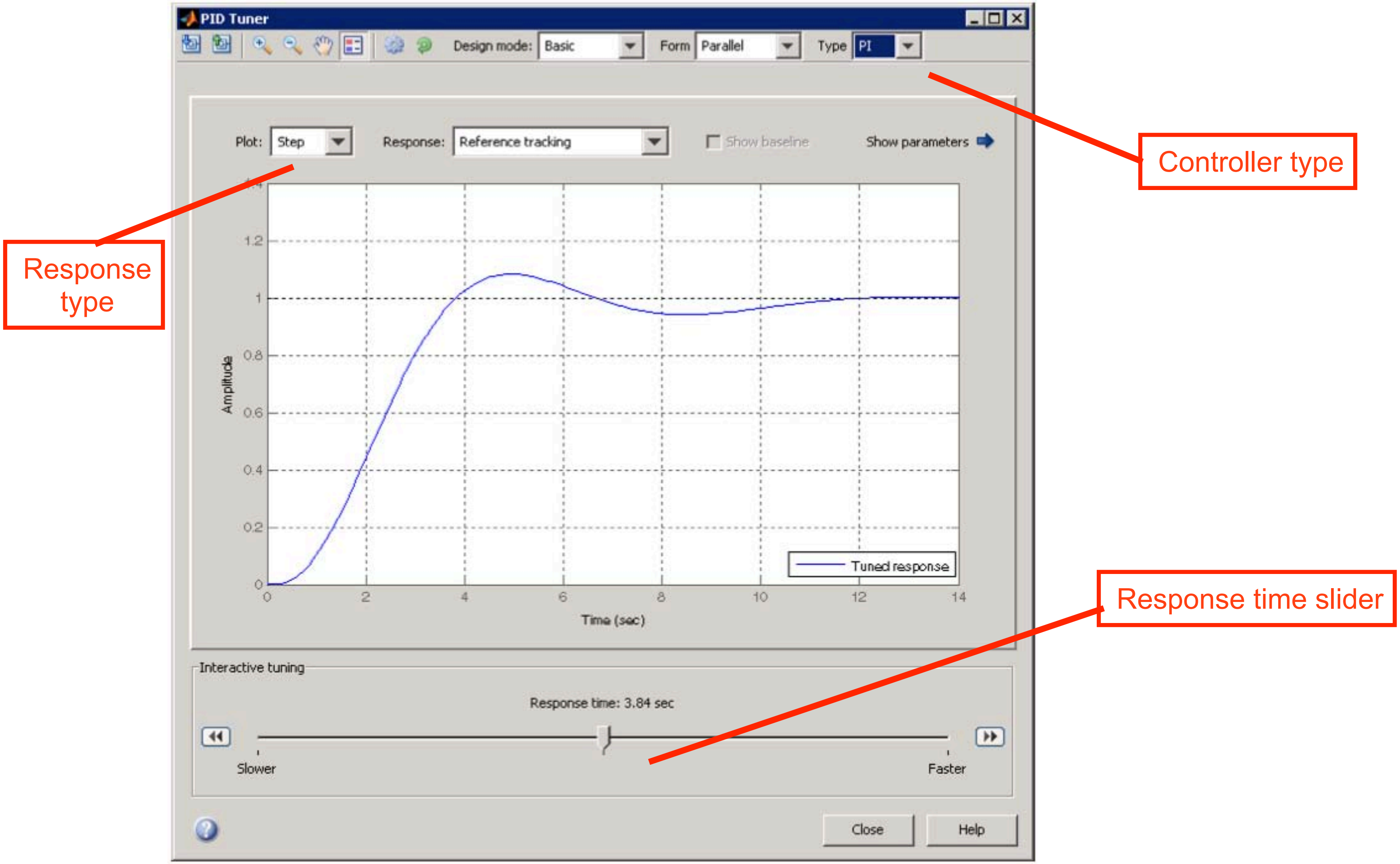
Type



- The PID Tuner automatically designs a controller for your plant.

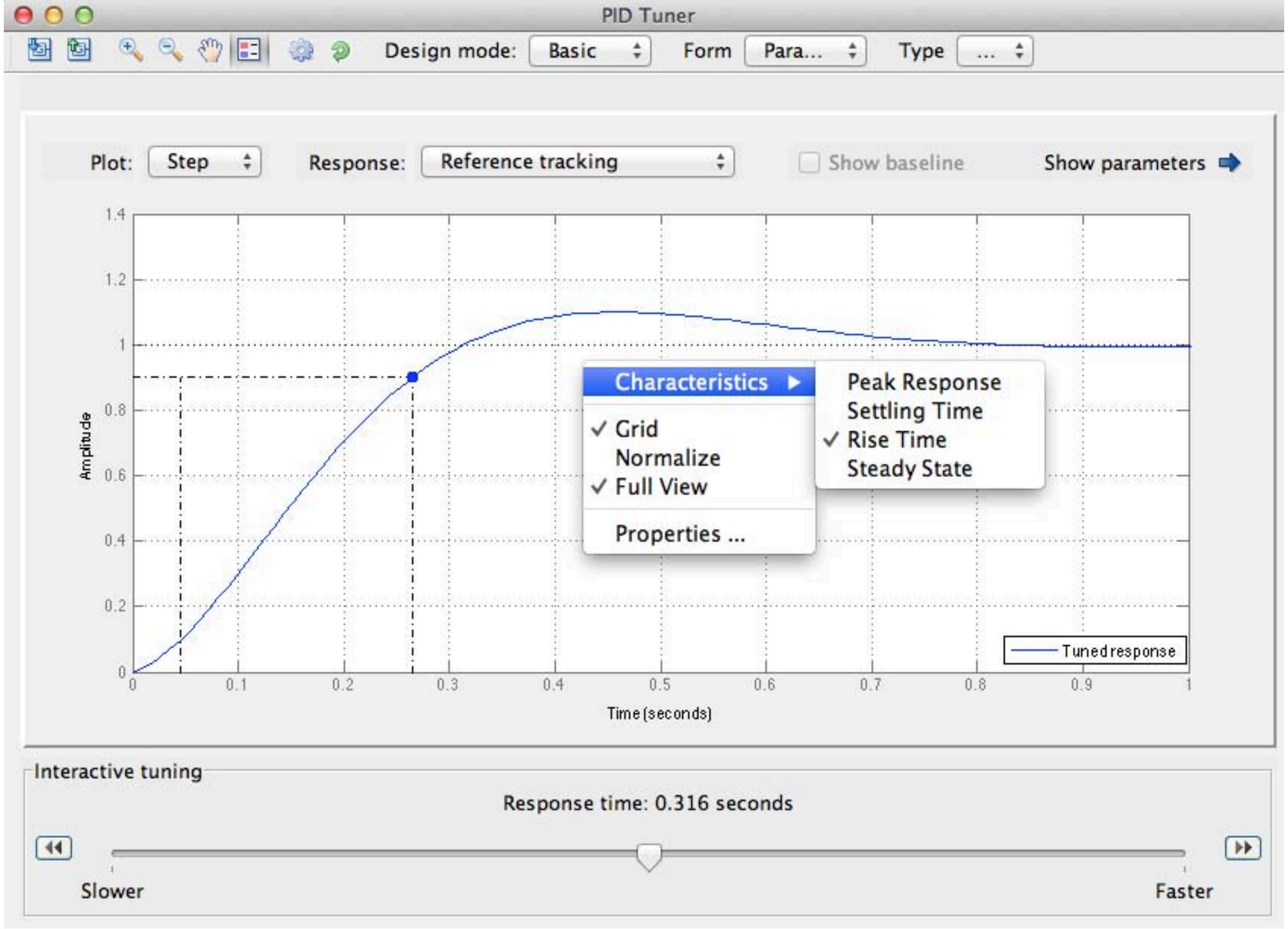
- You can use the **Response time slider** to try to improve the loop performance

The PID Tuner



The PID Tuner

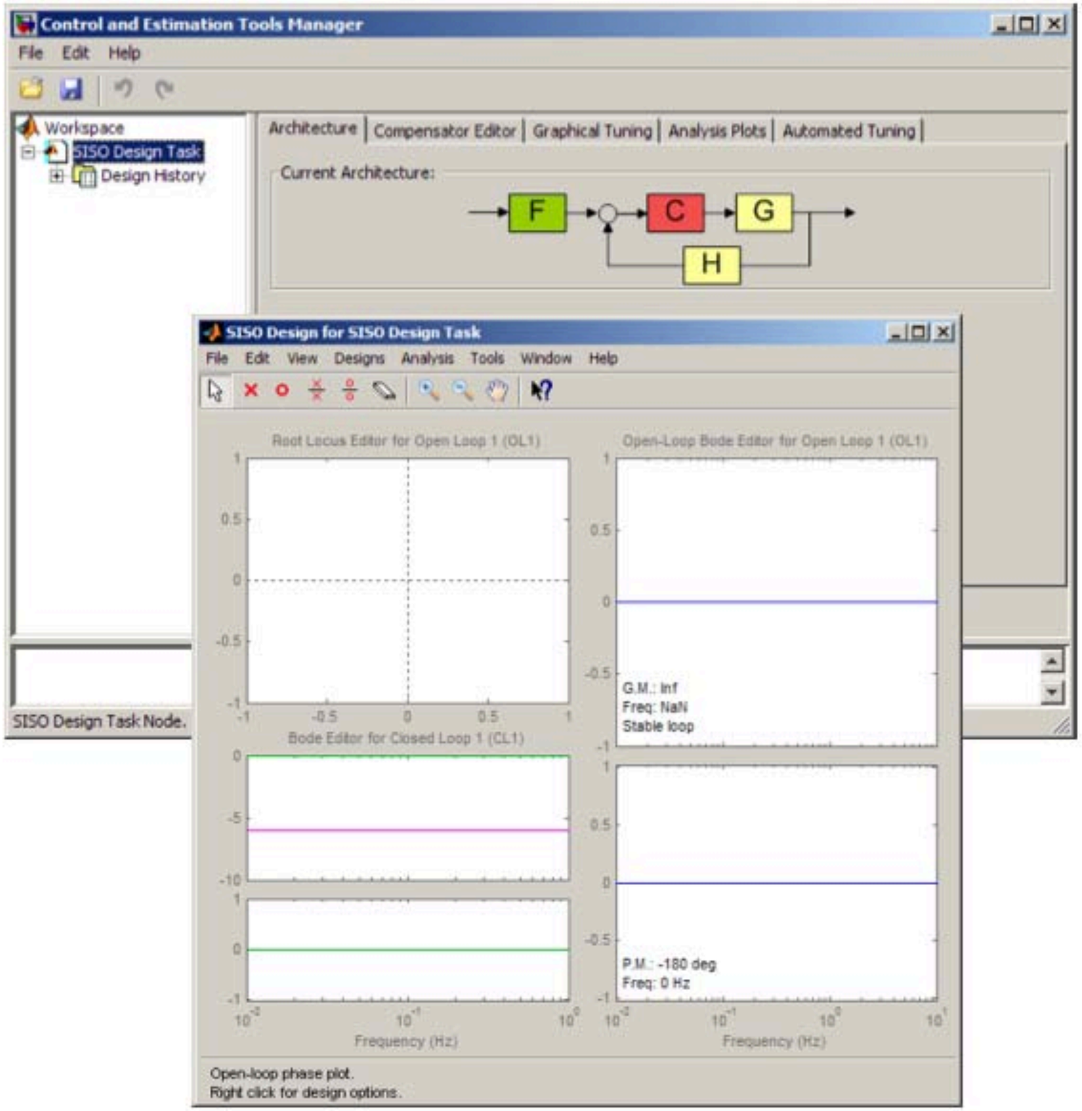
- **Right click** on the plot and select **Characteristics** to mark the characteristic times.



The SISO Design Tool

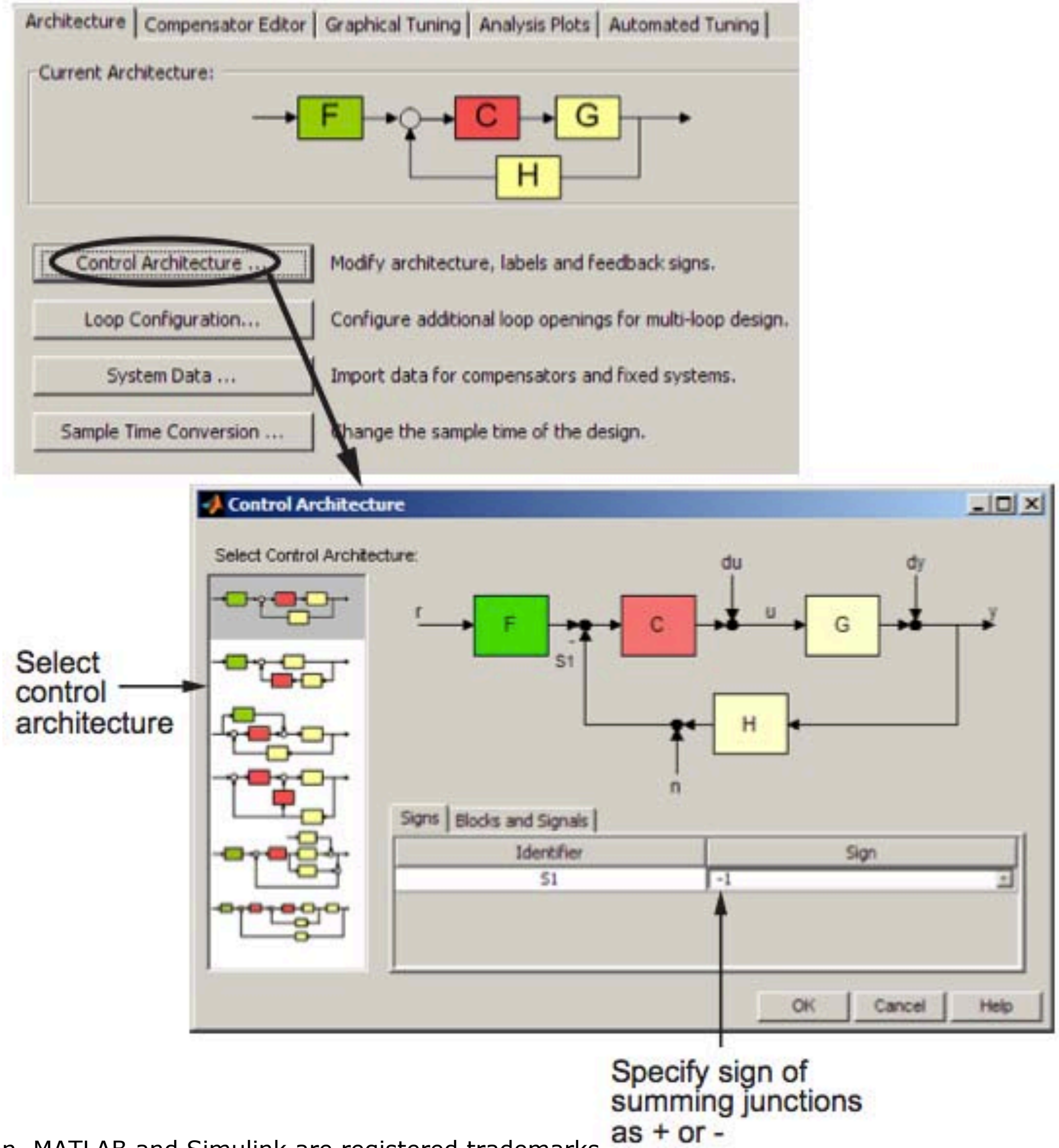
- Open the control design GUIs with the following command

sisoto



Define The Control Architecture

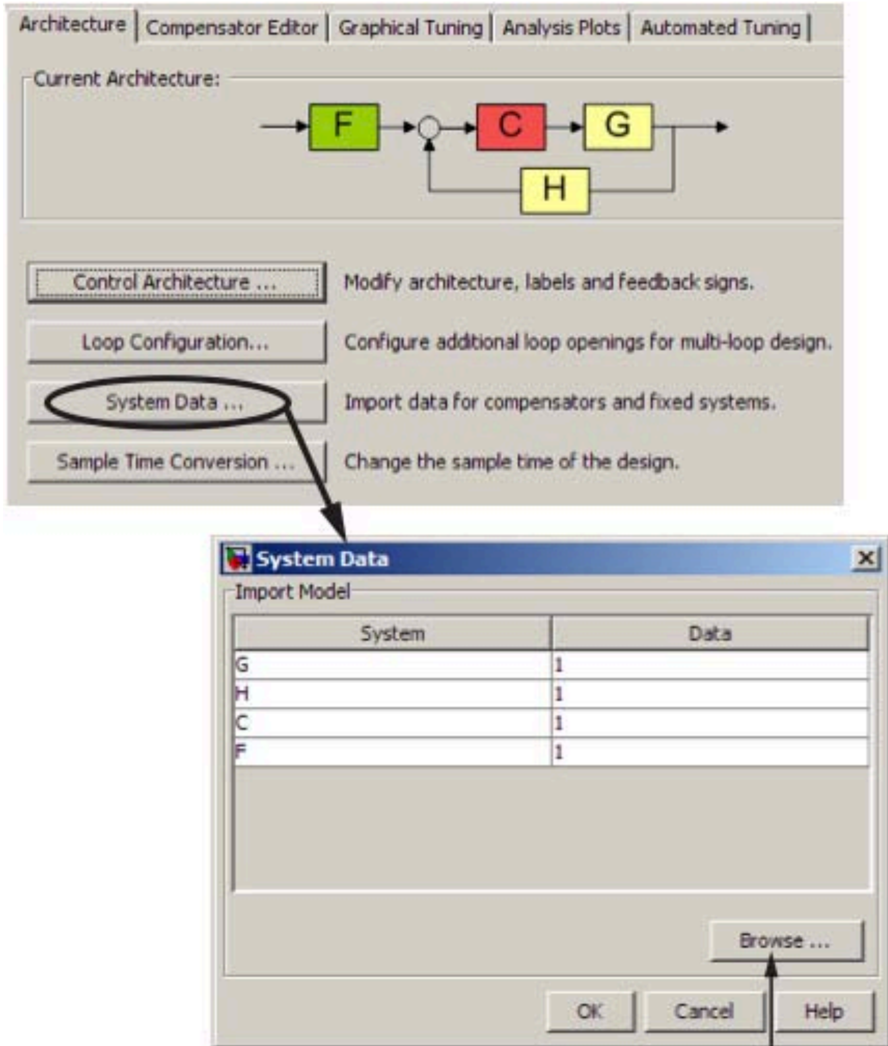
- In the "**Architecture tab**", click "**Control Architecture**"
- Select proper architecture and specify the sign of



Specifying System Data

- In the **Architecture tab**, click **System Data**

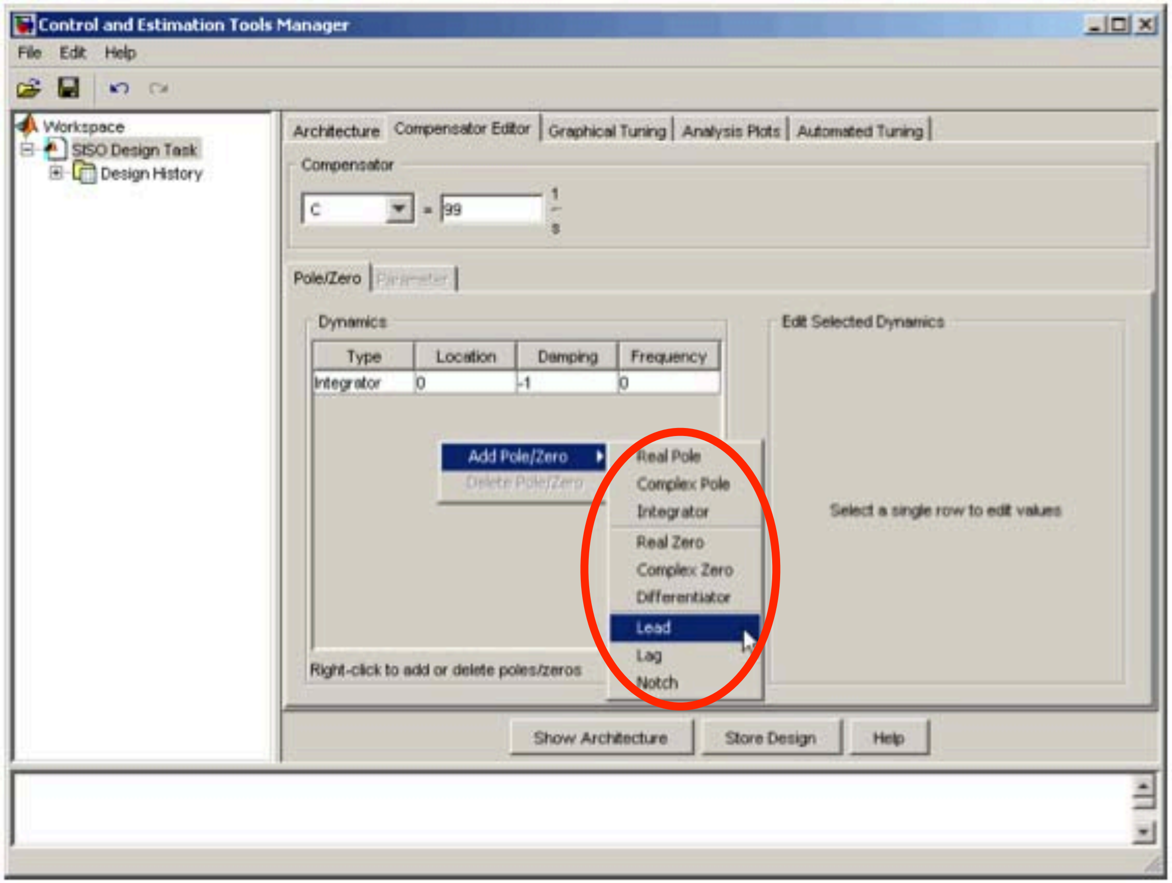
- You can select values or transfer functions from MATLAB **workspace** or a ***.mat** file



Browse for model in MATLAB workspace

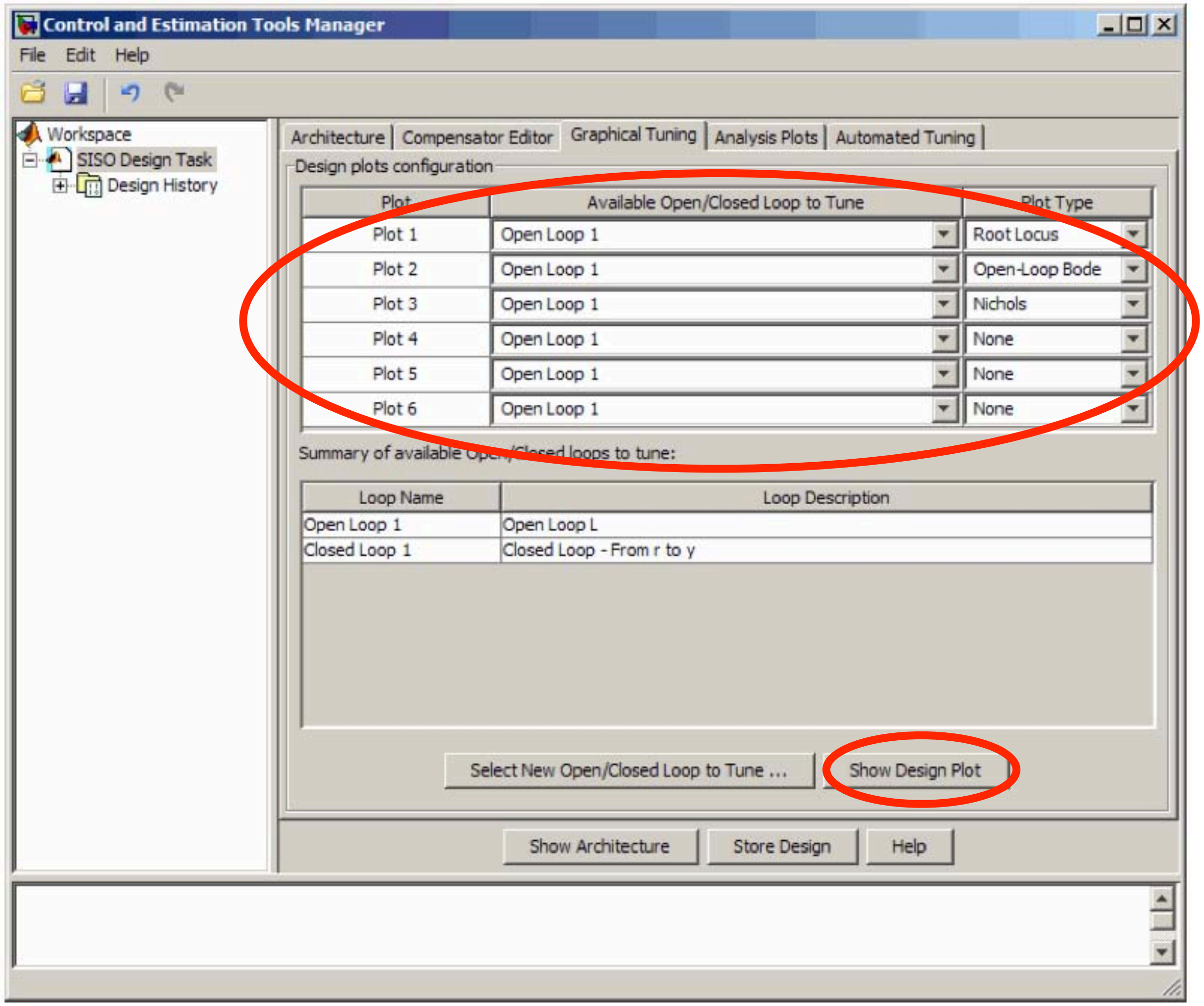
Compensator Design

- In the **Compensator Editor** tab, you can manually define the compensator form.
- Right click on the "**Dynamics**" table allows you to add/delete **Poles, Zeros, Integrators, Differentiators** etc.



Graphically Tuning Control Parameters ①

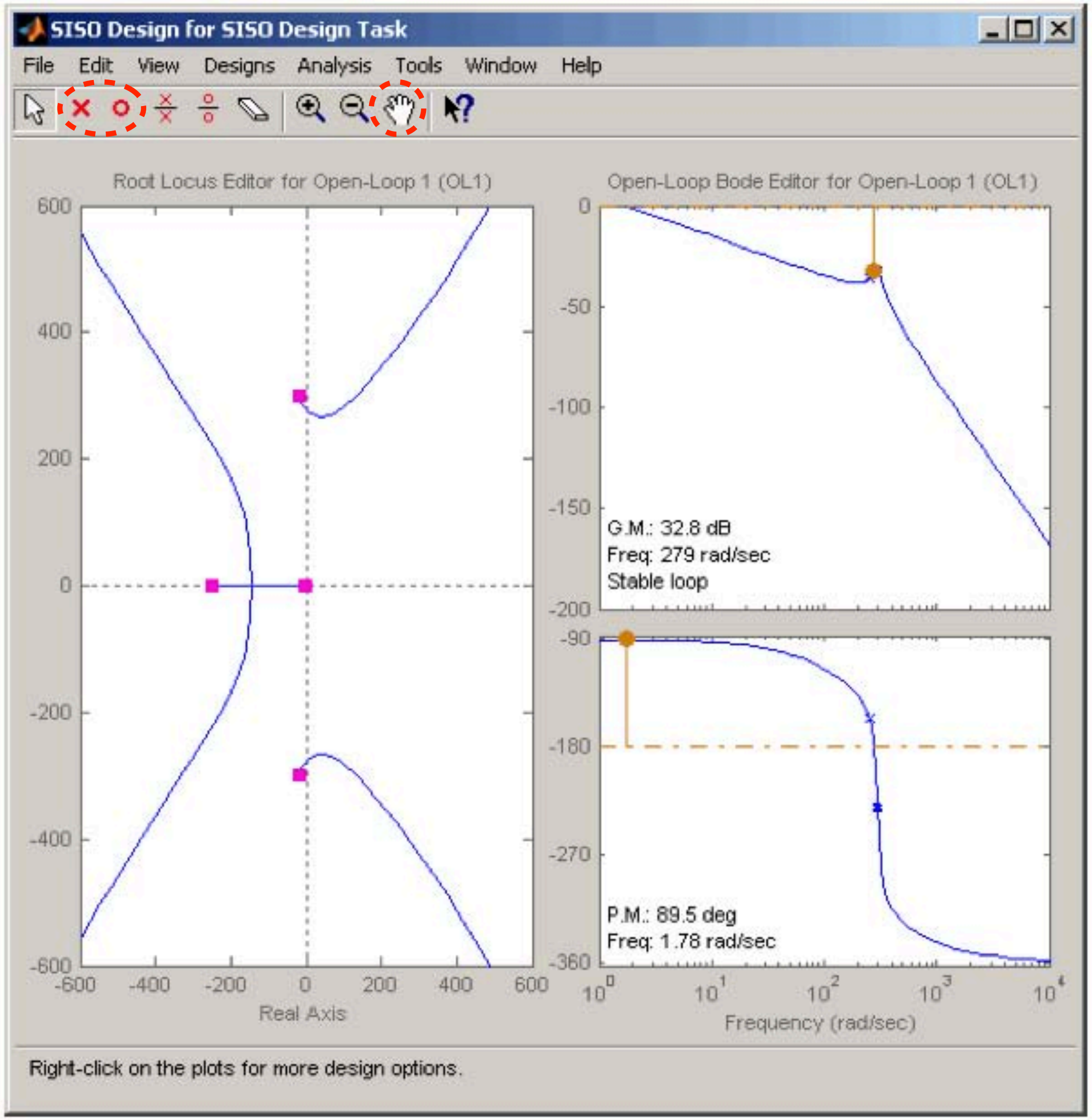
- In the **Graphical Tuning** tab, you can configure the plots you want to see in the Graphical Tuning Window.



Graphically Tuning Control Parameters ②

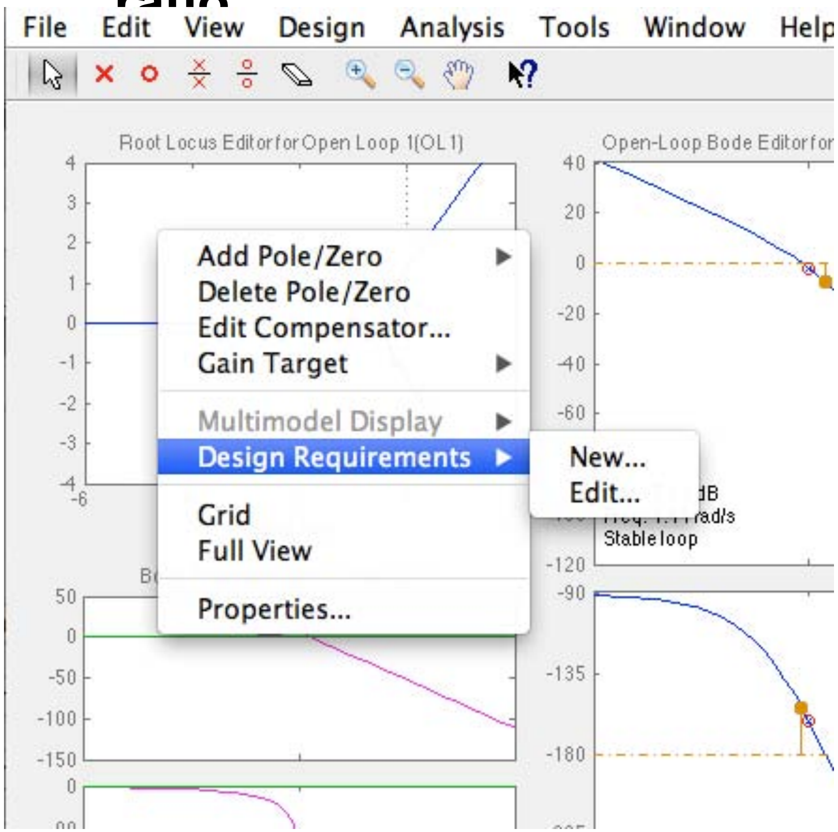
- In the Graphical Tuning Window, you can grab and drag the pink squares using the small hand in the toolbar. This changes the constant multiplier value of the compensator.

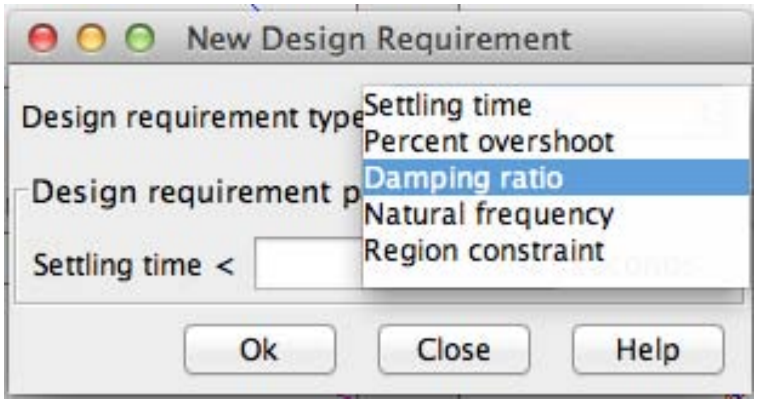
- You can also add poles and zeros in this window using the "cross" and "circle" in the toolbar.

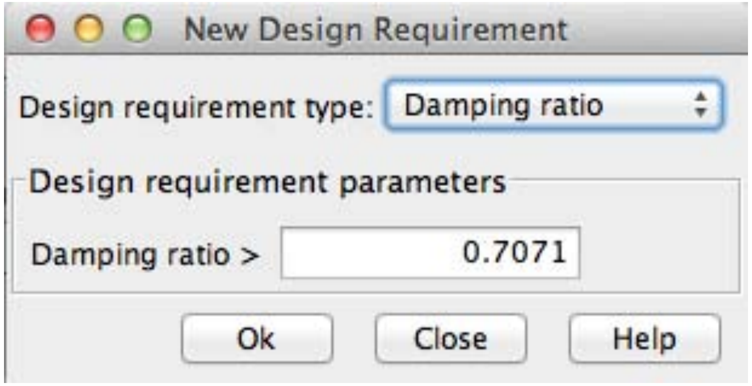


Viewing Damping Ratios ①

- **Right-click** on the **root locus** graph and select **Requirements > New** to add a design requirement.
- In the **New Design Requirement** dialog box **choose Damping ratio**

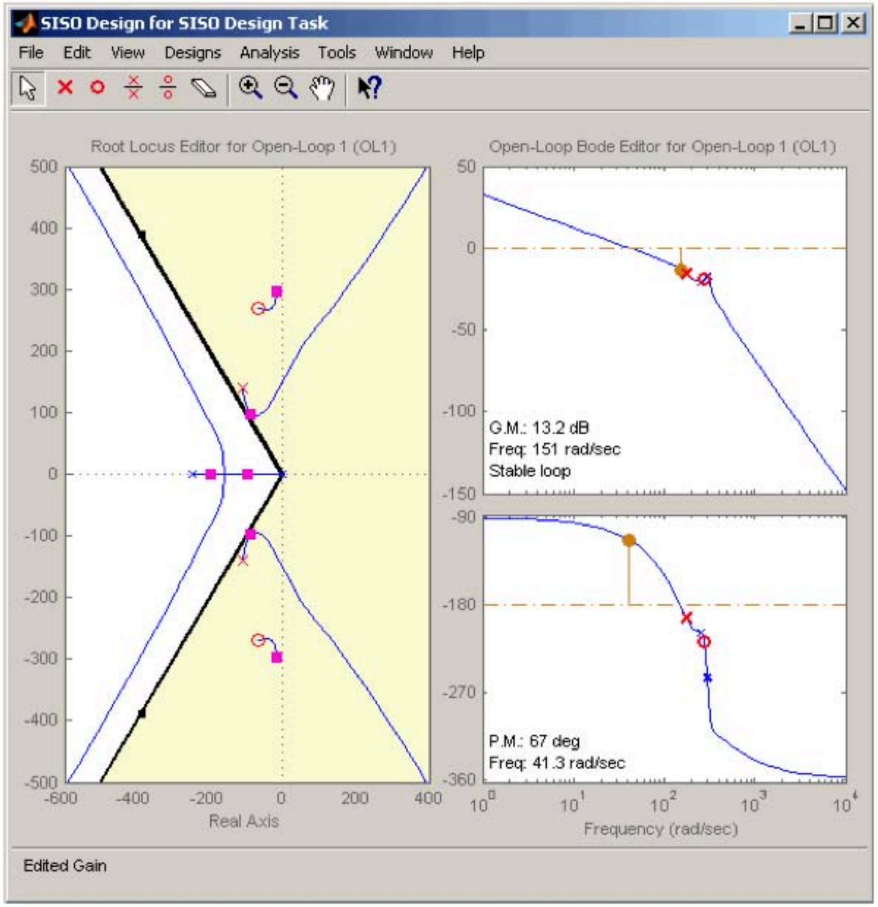






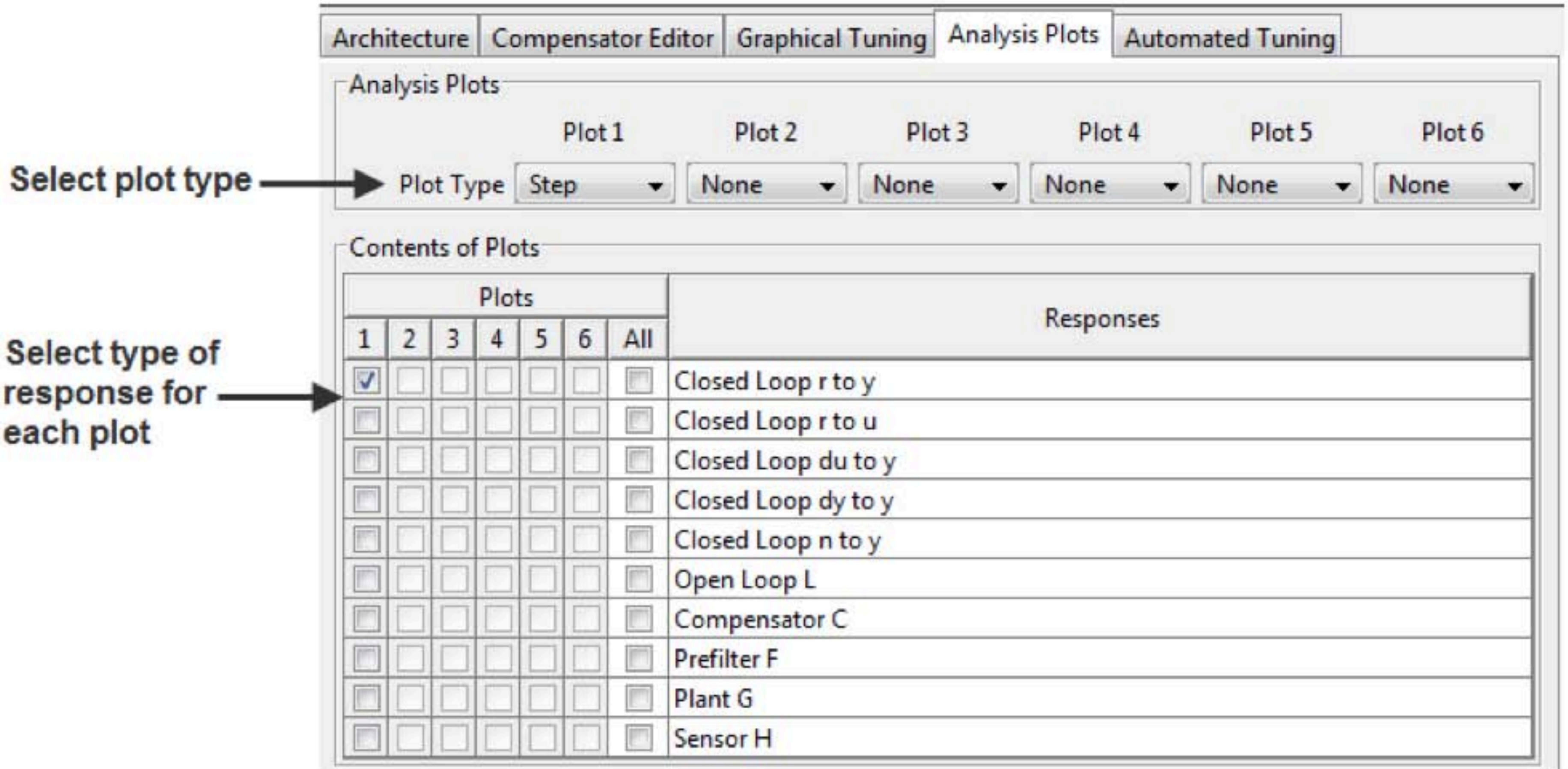
Viewing Damping Ratios ②

- Applying damping ratio requirements to the root locus plot results in a pair of **shaded rays** at the **desired slope**
- Try moving the complex pair of poles you added to the design so that they are **on the 0.707 damping ratio line**.



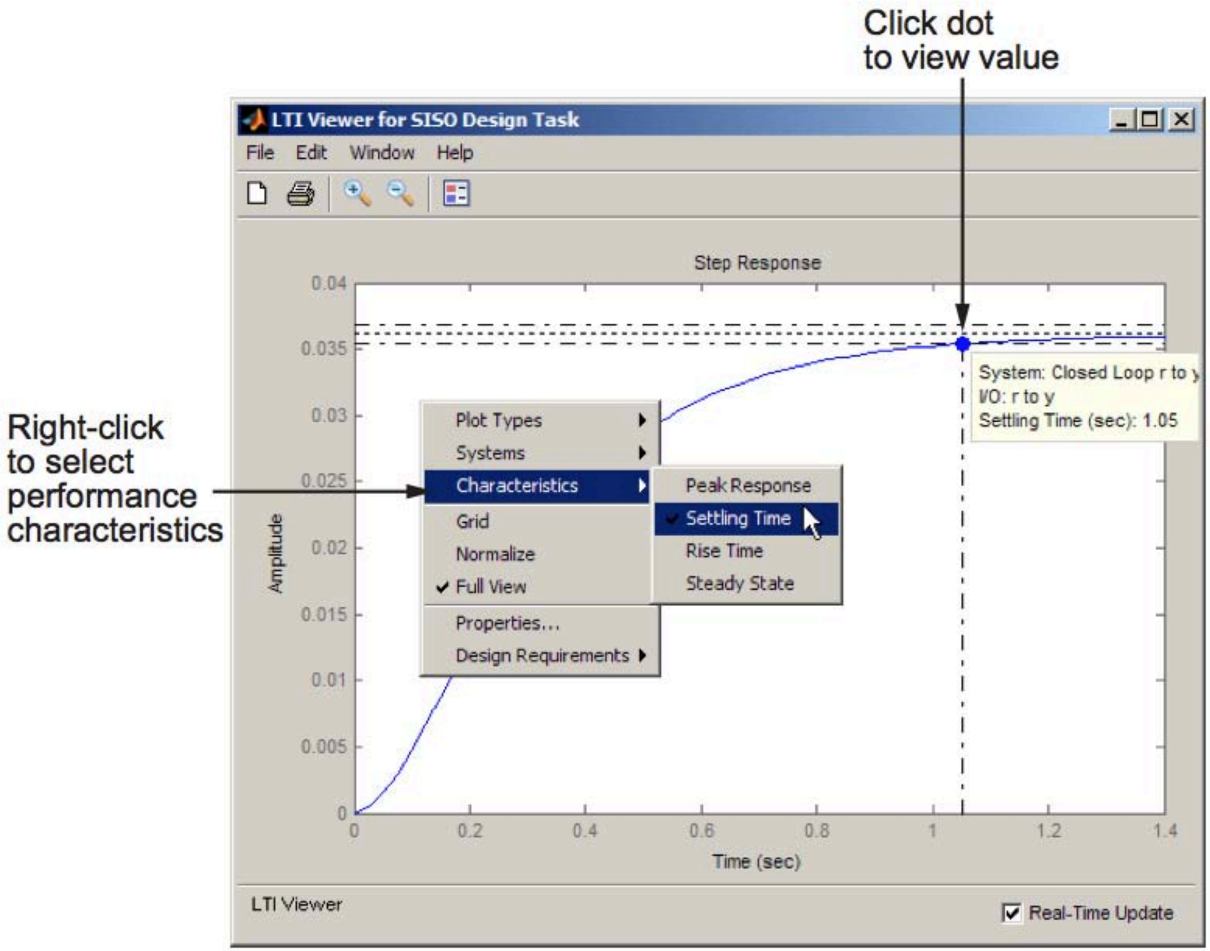
Analysis Plot ①

- In the **Analysis Plots** tab, select a **plot type**
- Select the type of response for each plot



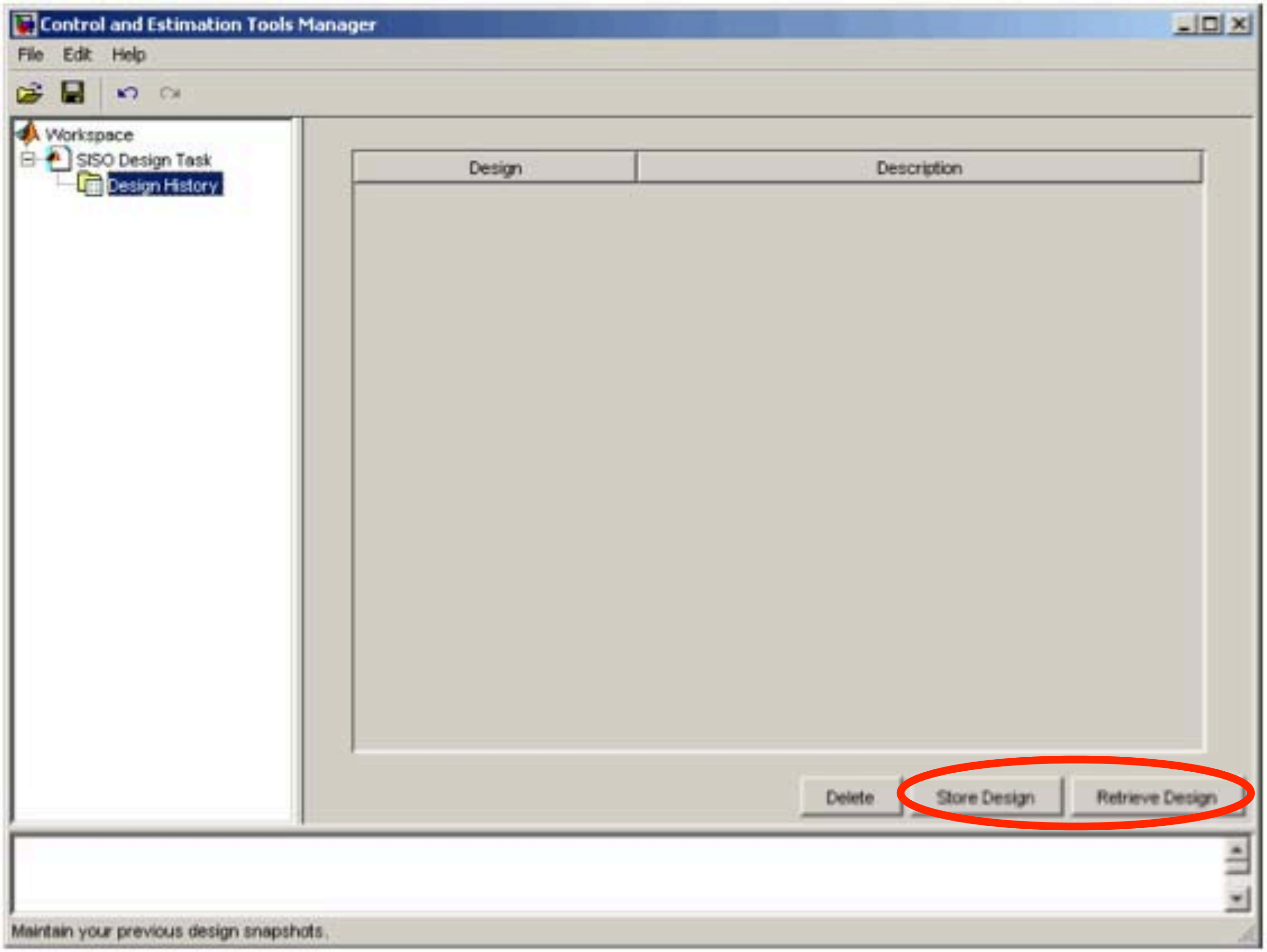
Analysis Plot ②

- Click Show analysis plot in the Analysis plots tab
- This displays the real-time specific performance characteristics for your system. Compare values to design requirements.



Storing and Retrieving Intermediate Designs

- Click the **Design History node** or **Store Design**, both located on the SISO Design Task node in the Control and Estimation Tools Mana

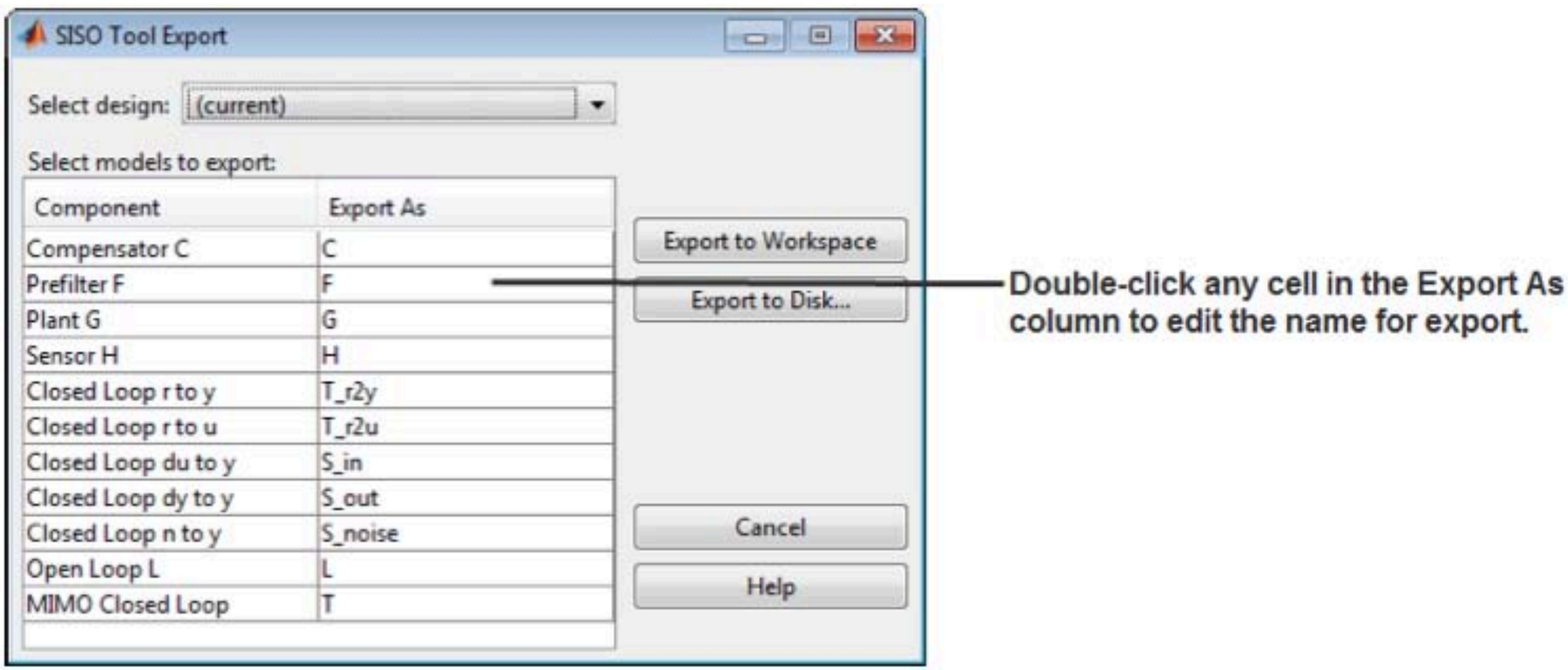


Exporting the Compensator and Models

After you design the controller, you may want to save your design parameters

for future implementation.

- Select **File > Export** in the Control and Estimation Tools Manager
- Select **File > Export** in the Graphical Tuning window.



2.04A Systems and Controls
Spring 2013