ENGR 105: Feedback Control Design
Winter 2013

Lecture 14 - Using the MATLAB Control System Toolbox and Simulink

Friday, February 8, 2013

Today's Objectives

1. introduce the MATLAB Control System Toolbox
2. introduce Simulink
3. solve problems in class using the toolbox

Reference: FPE Appendix C, MATLAB website
Note: To use your computer for this lecture, you will need: MATLAB, Simulink, and the Control System Toolbox. All of these come with the Student Edition of MATLAB.

1 Control System Toolbox

The Control System Toolbox is a collection of algorithms, written mostly as m-files, that implements common control system design, analysis, and modeling techniques. Convenient graphical user interfaces (GUIs) simplify typical control engineering tasks.

The command `help control` gives a list of the different functions. The most commonly used functions are presented below. You are encouraged to look through the help files for other functions and options that you might find useful in future.

1.1 Transfer Functions

Control systems can be modeled as transfer functions, in zero-pole-gain form, or state-space form (take ENGR 205 to learn the latter).

Transfer function models are created using the function `tf(numerator,denominator)`, where numerator and denominator are two vectors containing the coefficients of the polynomials in the numerator and denominator of the transfer function.

```
>> num = [1 2 0];
>> den = [2 2 1];
>> sys = tf(num,den)
Transfer function:
   s^2 + 2 s
---------------
2 s^2 + 2 s + 1
```

Zero-Pole-Gain forms of transfer functions, ie, $G(s) = \frac{K(s-z_1)(s-z_2)...}{(s-p_1)(s-p_2)...}$ can be specified directly as `zpk(z,p,K)` where z and p are the vectors of zero and pole values in the complex plane (`z= [ z1 z2 ...]` and `p = [p1 p2 ...]`), and `K` is the gain. The equivalent to the above system is:

```
>> K = 0.5;
>> z = [0 -2];
>> p = [-0.5+0.5*i -0.5-0.5*i];
>> zpk(z,p,K)
Zero/pole/gain:
  0.5 s (s+2)
---------------
(s^2 + s + 0.5)
```

The above models can be used in conjunction with the operators +, - and * to generate new models. The command `sys1 * sys2` produces a series interconnection from input through `sys2` and `sys1` (in that order) to the output. `sys1` $\pm$ `sys2` represent parallel interconnections.

Control system models can also be converted from one form to the other. The functions presented above (`tf()`, `zpk()`) are overloaded to perform arbitrary system conversions. For example, if `sys1` is a state-space model, we can generate an equivalent transfer function model `sys2` by issuing the command

```
>> sys2 = tf(sys1);
```

The function `feedback(sys1,sys2)` will help you compute a closed-loop transfer function for system consisting of the negative feedback interconnection of model objects `sys1` (in the forward path) and `sys2` (in the feedback path).

1.2 Time Domain Analysis

Time domain analysis in the form of time response of a system to a specified input can be obtained using the following commands:

- `step(sys)` plots the step response of a system `sys`. Also, check out `stepinfo` for information about common performance metrics.
- `impulse(sys)` plots the impulse response of `sys`.
- `lsim(sys,input_vector,time_vector,initial_state_vector)` plots the response of `sys` to an arbitrary input. The elements of `input_vector` define the values of the input at times corresponding to the elements of `time_vector`. The initial state vector can be specified for state-space models only (so, not used in the course).
- `pidtool(sys,type)` launches the PID Tuner GUI and designs a controller of type `type` (which can take on values such as `'p'`, `'pi'`, and `'pid'`) for plant `sys`.
- `damp` and/or `roots` (a standard MATLAB function) can also be useful in determining the poles of a system.

- `conv` convolves two polynomials. It is particularly useful for determining the expanded coefficients for factored polynomials. For example, this command can be used to enter the transfer function $H(s) = s + 2$ by typing `H=tf([1 2],conv([1 1],[1 -3]))`.

1.3 Classical and Frequency Domain Analysis

Classical and frequency domain analysis tools are listed below. We will not cover these today; this is for your reference to use in the near future.

- `rlocus(sys)` plots the root locus of the system `sys` with variations in gain.
- `bode(sys)` plots the magnitude and phase angle Bode plots.
- `[Gm,Pm,Wg,Wp] = margin(sys)` calculates the gain margin `Gm`, the phase margin `Pm`, and the frequencies corresponding to their occurrence. Issuing the command `margin(sys)` alone plots the Bode diagram and marks the margins on it.
- `nyquist(sys)` plots the Nyquist plot of the system. Note that the loop at infinity is not represented on the plot.
- `sisotool(plant,compensator)` opens an interactive mode of the root locus and bode plots, which can be used to modify the compensator and gain to achieve the desired system characteristics. The function can be called without passing an initial compensator, in which case the poles and zeros of the compensator can be placed using the graphical interface. Final designs can be exported back to the MATLAB workspace for further analysis.

For more information about the Control System Toolbox, you can refer to:
`http://www.mathworks.com/products/control/`
`http://www.mathworks.com/help/control/examples/`

2 Simulink

Simulink is a graphical tool that allows us to simulate feedback control systems. To start Simulink, type at the command prompt:

```
>> simulink
```

A Simulink Library Browser will open. To open a window where you can create your system model, select *File* > *New* > *Model* in the Simulink Library Browser window.

To place a component, drag it from the component browser to the model space. To make a connection, click and drag from an arrow on a block to an arrow on another block. To do this more quickly, hold down CTRL and click on the arrows on each block that you wish to connect. To connect multiple lines to a single block, hold down CTRL and click on the line already attached to the block and then make the second connection.

For most of the systems we will encounter, we only need to be concerned with a small fraction of Simulinks component library. In particular, the components you should be familiar with are:

"Sources" library
Step: generates a unit step signal
Ramp: generates a ramp signal
Sine Wave: generates a sinusoid

"Sinks" library
Scope: used for viewing system output
To workspace: used to transfer a signal to MATLAB

"Continuous" library
Integrator: integrates a signal
Transfer Fcn: used to add a system block in transfer function form

"Math Operations" library
Gain: a constant gain
Sum: used to add two or more signals
Trigonometric Function: used to place non-linear trigonometric elements

"Signal Routing" library
Mux: used to multiplex signals together in order to plot several on one graph

For each of these blocks, you can specify the appropriate parameters by double-clicking on the block. You can set the simulation configuration parameters (like start and stop times) by selecting *Simulation > Configuration Parameters*. You run the simulation by selecting *Simulation > Start*. (Depending on your OS and version of Matlab, you may have different options, such as a triangle-shaped "play icon in the toolbar.)

Together, let's do an example Simulink simulation. You will build and simulate this notch filter and test it's effects on input sinusoids of different frequencies (you looked this in Assignment 3):

$$G(s) = \frac{s^2+\omega^2}{s^2+2\zeta\omega s+\omega^2}$$

Make the notch filter frequency $\omega = 100$ Hz. Test sinusoidal inputs of 1 Hz, 100 Hz, and 1000 Hz.

For more information Simulink, you can refer to:
`http://www.mathworks.com/products/simulink/`
`http://www.mathworks.com/help/simulink/examples/index.html`

3 MATLAB Challenge

We will give you two challenge problems. As soon as your team has an answer (exact if possible, approximate if necessary) using your assigned solution method (Control Systems Toolbox or Simulink), post it on piazza as a follow-up to the appropriate note set up but the instructors. Your posting should include the first names of your team members and your answer. Once you have posted your answer, if you have time left, see if you can try find the answer a different way using the same general solution method. You will solve the second challenge problem using the solution method (Control Systems Toolbox or Simulink) that you did *not* use on the first problem.

4 Examples and MATLAB Challenge Solutions

4.1 Introductory Material

Sample code for the Transfer Function section is in `lecture14_code.m`.

Simulink model for notch filter example is in `notch_filer.mdl`. But don't show the students the completed model, rather, lead them through its creation. After you drag the elements and use interconnections to create the Simulink model, take the following steps:

1. For the transfer function, assign the numerator coefficients `[1 0 (100*2*pi)^2]` and the denominator coefficients `[1 2*1*100*2*pi (100*2*pi)^2]`
2. Change the sine wave parameters so that the the frequency is `1*2*pi` for the 1 Hz example
3. Change the simulation configuration parameters to have a stop time of 1.0
4. Run the simulation and look at the scope for 1Hz. You can see that the input signal (yellow) feeds right through to the output (magenta)
5. Change the sine wave parameters so that the the frequency is `100*2*pi` for the 100 Hz example. You can see that the input signal is completely attenuated – the output is zero. The output also looks choppy because the simulation time step is too low.
6. Change the sine wave parameters so that the the frequency is `1000*2*pi` for the 1000 Hz example. There are many ways to deal with the time step issue, but here you can just change the stop time to 0.001. Run the simulation and you can see that the input signal feeds through again (albeit a bit delayed).

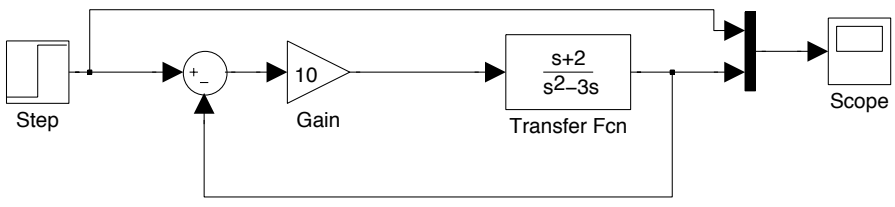
4.2 Challenge 1

Prompt: You have a plant $G(s) = \frac{s+2}{s^2-3s}$. You controller is of the form $D(s) = K$. Find the value of K that stabilizes this system using a simulation-based approach (i.e., test different values of K and see what happens). Note that we do *not* want a marginally stable system. We suggest that you use a unit step reference input.

Answer: $K > 3$

Solution A: Control System Toolbox (command-line) solution is in `challenge1_CST_solution.m`.

Solution B: Simulink solution is in `challenge1_simulink.mdl` (image below). Note that it isn't necessary to feed the step input to the scope, I just like to.



4.3 Challenge 2

Prompt: Here is the equation of motion for a damped pendulum: $\ddot{\theta} + \frac{c}{ml}\dot{\theta} + \frac{g}{l}\sin\theta = \frac{T_c}{ml^2}$. The input is T_c and the output is θ. Choose $l = 2.5$, $m = 0.75$, $c = 0.15$, and $g = 9.8$. Say we place this plant in a negative feedback system with PD controller (with gains $K_p = 100$ and $K_d = 10$) in the forward path. Consider the response of the system to a step of 45 degrees. To what value does the output converge?

Answer: About 38 degrees for the linearized case, and a little more for the nonlinear case. These are very approximate numbers, read off the plots. The point is that the nonlinear version responds a little differently.

Solution A: Control System Toolbox (command-line) solution is in `challenge2_CST_solution.m`.

Solution B: Simulink solutions are in `challenge2_simulink_linear.mdl` and `challenge2_simulink_nonlinear.mdl` (images below). Note that Simulink won't let you do a pure PD controller, since the order of the denominator is less than the numerator. So students can implement a lead controller with a "fast pole". (Considering the description of the lead compensator in Lecture 12, they could make $a = 100$.) Also, note that my solution assumes that l, m, c, and g have already been defined in the workspace.