ENGR 105: Feedback Control Design
Winter 2014

Lecture 6 - Block Diagrams and Control in MATLAB/Simulink

Friday, January 17, 2014

Today's Objectives

1. learn to manipulate block diagrams
2. introduce the MATLAB Control System Toolbox and Simulink

Reading: FPE Section 3.2, Appendix C, and the MATLAB website

1 Block diagrams

The transfer function can be used to tell us a lot about the system, including:

- The impulse response and response to general inputs
- Stability (through pole locations)
- Steady-state value (if poles are stable)
- The frequency response (in the next lecture, we will look at sinusoidal inputs)

The goal of feedback control is to shape the response of the system by closing a feedback loop, which changes the open-loop transfer function into the closed-loop transfer function. To do this, we need to be comfortable working with systems in a form that looks like:

There are some simple rules of block diagram algebra that make it easy to manipulate systems into a form we want.

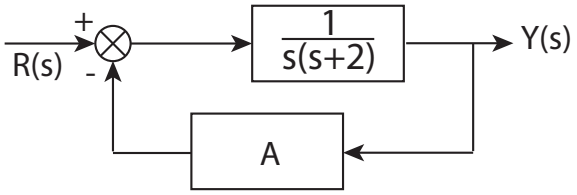
1.1 Basic Connections

Cascade

Sum

Feedback

1.2 Example



This could be a DC motor with position feedback.

The response to a change in reference input is stable instead of continuing to integrate as in an open-loop system. (Compare to example in Lecture 5.)

1.3 Unity feedback

It is often helpful to rearrange the system to have a unity feedback loop (no blocks in the feedback path). This can be accomplished with some manipulation of the diagram.

We can move the blocks into the feed forward or feedback paths to simplify analysis.

2 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. I encourage you to look through the help files for other functions and options that you might find useful in future.

2.1 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). Be careful – if you want `sys2` to be in the forward path, you need to use `feedback(sys1*sys2,1)`.

2.2 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, try `conv([1 1],[1 -3])`.

2.3 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/`

3 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.

3.1 Adding and connecting components

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 Simulink's 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

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.)

3.2 Example

Say 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. Use a unit step reference input in your simulation.

Solution A: Control System Toolbox (command-line) solution:

Solution B: Simulink solution:

Solution C: Find the answer analytically!

Answer: The system is stable if $K >$

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