

Introduction to Simulink

October 3, 2007

Control Systems – 520.353

Frequently Asked Questions about Simulink

Q: What does Simulink do?

A: Simulink performs numerical simulations of a wide range of dynamic systems.

Q: Why bother learning how to use Simulink?

A: Simulink is a really easy way to model and test the LTI systems that we focus on in Control Systems. Moreover, Simulink is not limited to LTI systems and can be helpful when dealing with systems that are a lot harder to analyze.

Q: Where at Johns Hopkins can I use Simulink (and Matlab)?

A: Besides the computers in the ECE department, the computers in the Krieger computer lab have Matlab and Simulink and are equipped with the Control Systems toolbox. Also, if you want to buy the student version of Matlab, which includes Simulink and the Control Systems toolbox, it can be found at the bookstore for around \$100. (This is a good deal. If you aren't a student, these programs cost a whole lot more.)

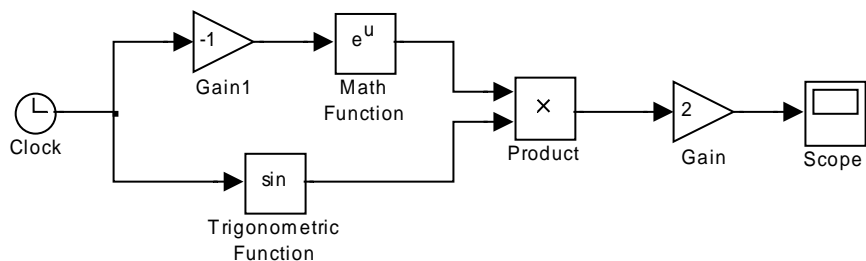
Getting Started with Simulink

- Double-click on Matlab R2006a (on the desktop) to open it.
- At the Matlab command prompt, type `simulink`. This will open the Simulink Library Browser.
- A Simulink model is made up of “blocks” that you connect by drawing lines between them. Look through the different categories of blocks to see what you have to work with:
 - “Commonly Used Blocks” contains an assortment of blocks from other categories that someone decided were commonly used.
 - “Continuous” contains a number of blocks useful in building dynamic systems, including the Integrator, the Transfer Fcn, and the Zero-Pole. Later in this course you may use the State-Space block as well.
 - “Math Operations” has blocks that, well, perform math operations: Add, Subtract, Product, Divide, etc. The Trigonometric Function and the Math Function can each perform several different operations depending on how you set them up.
 - “Sinks” are blocks that take a signal from the model and output it in some useful form. The Scope and To Workspace blocks are especially useful.

- “Sources” produce many types of signals as inputs to a model. You may find the Clock, Constant, Ramp, Sine Wave, and Step blocks to be helpful.
- In the toolbar at the top of the window, click the leftmost button to create a new model.

Building a Basic Model

- We’re going to build a system that outputs the signal $x(t) = 2e^{-t} \sin(t)$.
- From the Sources category, find the Clock block and drag it into your model. That will give you the t signal.
- From the Math Operations category, find the Trigonometric Function block and drag it into your model. That will give you the \sin function to use on the t signal.
- Also, find the Math Function block and drag it into your model. Double-click on the block and look at the Function drop-down menu to see what kinds of functions this block can implement. Choose exp (the default).
- You’re going to want to multiply those two signals together, so drag in a Product block.
- Get yourself a Gain block to multiply by two. When you have the block in your model, double-click on it and set the gain to 2. Also, get another gain block to change the t signal to $-t$ for input to the exp function. Set the gain appropriately.
- Finally, go to the Commonly Used Blocks category and bring in a Scope to see the results.
- Click and drag to draw lines between the output and input ports of blocks to connect them together appropriately. When you’re done, your model should look something like this:

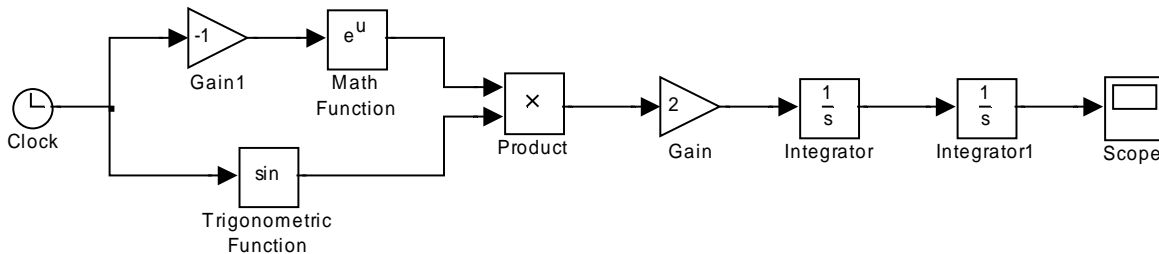


- Now it’s time to see what this model does. In the toolbar, click the triangle to start the simulation. This should go pretty fast. When it’s done, you’ll see the “Ready” message appear at the bottom left corner of the window. Double-click the scope and examine the results of your simulation. Right-click and choose Autoscale to see it a little better.

CHECKPOINT #1

Solving a Differential Equation

- Homework #1, problem #4 asked you to find the solution to the differential equation $\ddot{x} = 2e^{-t} \sin t$, $t \geq 0$, with zero initial conditions. We're going to solve this equation numerically in Simulink.
- Note that if $\ddot{x} = 2e^{-t} \sin t$, $x(0) = 0$, $\dot{x}(0) = 0$, then $x(t) = \int_0^t \int_0^{t_1} 2e^{-t_2} \sin t_2 dt_2 dt_1$.
- Implement this double integral in Simulink by dragging in two Integrators from the Continuous category. Click the line leading to the Scope block and hit Delete to remove it. Connect the two Integrators in series to the output of the Gain block, then connect the Scope to the output of the second Integrator.
- Double-click on each Integrator to verify that the initial conditions $x(0)$ and $\dot{x}(0)$ are both zero.
- Your model should look something like this:



- Click the button to start the simulation. Examine the results on the Scope. Does it compare favorably with the answer to Homework #1, problem #4? (If you didn't get the problem right, consult the course web site for the solutions.) You may want to use Matlab to see what $x(t)$ is supposed to look like.

CHECKPOINT #2

Building Models with Transfer Functions

- Close your model and create a new one.
- We're going to build the model from Homework #3, problem #3. Recall that this system had a controller with transfer function $H_C(s) = \frac{s+4}{s-1}$, a plant with transfer function

$$H_P(s) = \frac{s-1}{s(s+3)}, \text{ and unity feedback.}$$

- From the Continuous category, drag two Zero-Pole blocks into your model. Double-click on each block and notice how it's configured with a vector of zeros and a vector of poles. Configure one block to be the controller and the other to be the plant.
- From the Sources category, drag a Step block into your model to be the input.

- Open the Commonly Used Functions category and drag a Sum block into your model. Since this is a negative feedback system, we want to change this around a little. Double-click on the Sum block and find where it says “List of signs.” Change the second + sign to a – sign. Click OK and notice how this changed the appearance of the Sum block.
- Also from Commonly used functions, get yourself two Scopes to monitor the controller output and the plant output.
- Connect the blocks together in the standard negative feedback configuration as shown in the homework problem.
- Click the button to start the simulation. Examine the results on each Scope. Do they compare favorably to what you got for the homework problem using Matlab?

CHECKPOINT #3

Wild Times with Nonlinear Systems

- Recall from class that a nonlinear system can be approximated as linear about an operating point. Let’s find out how good this approximation is.
- Create a new model and use Simulink blocks to implement the nonlinear differential equation from Homework #2, problem #1: $\dot{x}(t) = -ax(t)\ln(bx(t))$. (Hint: the \ln function can be found in the Math Function block under Math Operations.) Use $a = 3$ and $b = 2$. Rename the blocks (Gain or Constant) where you set a and b to “a” and “b.”
- Recall that the equilibrium of this system occurs at $x = \frac{1}{b}$. Set the initial condition $x(0)$ to 0.5 (where?).
- Connect a Scope to the $x(t)$ signal in this model. Run the simulation and check what’s on the Scope. The output should be constant since you started the system in equilibrium. (If the output isn’t constant, fix your system until it comes out right.)
- Examine how this system reacts to a slight disturbance from the equilibrium: set the initial condition $x(0)$ to 0.75 and run the simulation. Do the results look like what you expected?
- Let’s compare the output of the nonlinear system to that of the linearized system developed in the homework problem. In the Sinks category, find the To Workspace block and connect it to the $x(t)$ signal in your model. (You need not remove the Scope.) Double-click on the block and change the variable name to “x_nonlinear” and the save format to Array. Run the simulation again.
- Look in the Matlab workspace and find the variable `x_nonlinear`. You’ll also see a variable called `tout`; that’s the time vector that corresponds with `x_nonlinear`.

- Set up another time vector in Matlab: $t = 0:0.1:10$. Using that time vector, calculate $x_{linear}(t) = \frac{1}{b} + \epsilon e^{-at}$, the correct answer to Homework #2, problem #2. (Since $x(0) = 0.75$, what is ϵ ?)
- Plot both curves on the same axis: `plot(t, x_linear, tout, x_nonlinear)`. If you plot them in that order, the linear response will be blue and the nonlinear response will be green. How good is the linear approximation?

CHECKPOINT #4

