EE 105 Simulink Lab : Block Diagram Simulation as an Engineer's Problem Solving Tool

Jay Farrell
College of Engineering
University of California, Riverside
jay.farrell@ucr.edu

January 23, 2021

Abstract

The objective of this lab is to introduce the student to block diagram simulation. Many block diagram simulation packages are available. This lab will use the Simulink extension of Matlab.

The following write-up includes two detailed examples about the creation of block diagram simulations. Future assignments will ask you to use Simulink for more complex systems. Work through the examples herein to learn the basics of Simulink.

1 Prelab

Complete the following steps prior to lab.

1. Fig. 1 shows the bond graph (BG) for a fourth order system with the state variables labeled.

2. Walk the causal strokes and the state variables through the BG.

3. Write down the state space model for the system, defining the matrices A, B, C, and D.

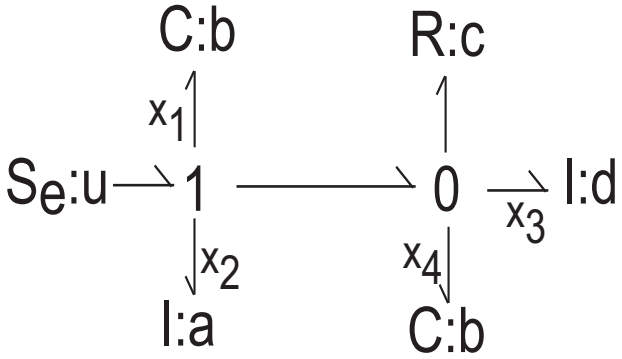

Figure 1: Bond graph.

2 Simulink in Lab Procedure

Define the system parameters to be

$$a = 1,\ b = 9,\ c = 0.1,\ d = 1.$$

Let the output be the flow through the R-element with parameter c.

1. Find the poles of the system, using eig(A). Later in the course we will show that the poles of the transfer function are the same as the eigenvalues of the state-space matrix A.

2. Find the dominant time constant of the system. How long should it take for the transient response to decay away? This is the settling time.

3. Implement a signal flow diagram for the system using integrators connected together by summers and gains. This approach is explained further in Appendix A of this lab.

   (a) Simulate the system with a unit sinusoidal input. Ensure that your simulation is long enough in duration that the transient decays away, and at least two cycles of the steady-state response are included.

   (b) Try various frequencies for the input. Plot the amplitude of the steady-state output versus the radian frequency of the input using a loglog scale. Is there a frequency at which the output magnitude is maximum?

   (c) The settling-time for each input frequency should be the same. Is it?

4. Represent the state vector by the symbol $x(t)$. Define the A, B, C, and D matrices so that the linear state space model is represented as

$$\begin{aligned} \dot{x} &= = Ax + Bu \\ y &= Cx + Du \end{aligned}$$

   (a) Create an m-file that Computes the A, B, C, and D matrices.

(b) Run that m-file, so that the matrix variables A, B, C, and D are in memory.

(c) Uses the function '[n,d]=ss2tf(A,B,C,D)' to find the transfer function for the system where n is the numerator polynomial and d is the denominator polynomial.

(d) Use the 'roots' function on d and n to find the poles and zeros. Confirm that the poles found here match the eigenvalues of A found earlier.

(e) Use the function 'bode(n,d)' to plot the Bode plots for the system. The top plot is the magnitude response. Based on the magnitude response, if you want to maximize the gain from the input to the output, what frequency of input should you apply? What will be the amplification?

(f) Create a new, empty simulink window.

(g) From the simulink library browser, in the 'continuous' folder, copy the 'State-Space' block to the Simulink window. Also copy a sinusoidal source and a scope from the source and sink folders. Connect them up to represent the system.

(h) Double click the 'State-space' block to open it up. Enter the names 'A', 'B', 'C', and 'D' into the appropriate menu slots. Do not type in the whole matrix, just the matrix names. Simulink will read the data from the workspace for you.

(i) Simulate the system with the frequency of the sinusoid set equal to the value that you obtained from the magnitude response. Do you get the amplification that you expected.

(j) Compare the gain and phase shift information from the Bode plot with the simulated response. They should match.[1]

The simulations that you created in these two parts should be identical. They are just implemented by different means. Approach 1 can work for any system linear or nonlinear. Approach 2 only works for linear state space systems.

A Matlab Signal Flow Diagram

This section explains the basic methods for constructing Simulink signal flow models.

A.1 First-order Example

The top (red) portion of Fig. 2 illustrates the idea that the block labeled '$\frac{1}{s}$' is an integrator. Its output is the integral of its input. The input is labeled $\frac{d}{dt}x_1(t)$; therefore, the output is $x_1(t)$.

[1] See the 'Frequency Response: Physical Meaning' section of the lecture notes.

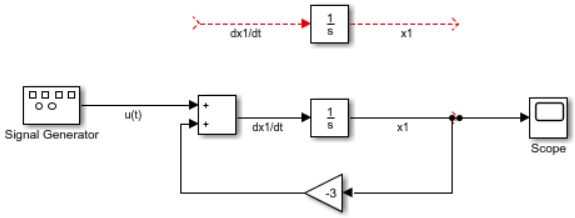

Figure 2: Basic Simulink signal flow diagram.

The lower portion of Fig. 3 illustrates the implementation of the signal flow model for the ODE:

$$\dot{x}_1(t) = -3x_1(t) + u(t).$$

This is a working simulink model. The input can be changed by doubli-clicking the signal generator. The initial conditions can be changed by double-clicking the integrator.

A.2 Third-order Example

Fig. 3 implements the third-order state-space model

$$\dot{x} = \begin{bmatrix} -5.0 & 0.0 & 0.2 \\ 4.0 & 0.0 & -2.0 \\ 0.0 & 5.0 & -0.2 \end{bmatrix} x(t) + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u(t) \\ v(t) \end{bmatrix}. \quad (1)$$

Look at the diagram and check that each row of the matrix equation is correctly implemented.

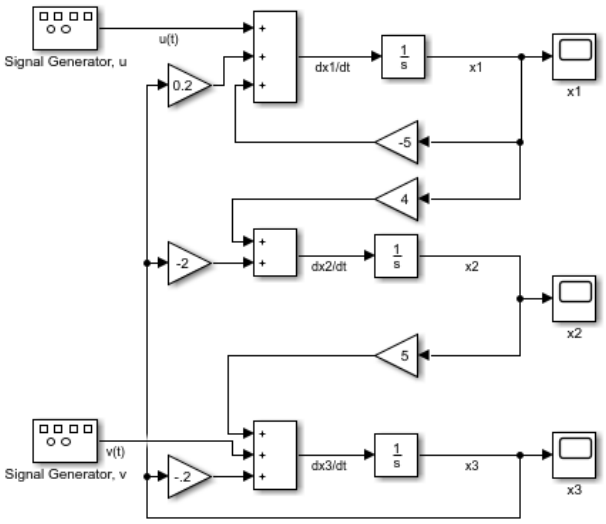

Figure 3: Third-order Simulink signal flow diagram.

From Fig. 3, it should become clear that as the order of the system grows, implementing each state separately will result in a messy tangle of signal flows. Fig. 4 shows the same system implemented using the simulink State-Space block.

The names of the matrices for the model parameters A, B, C, and D are communicated to this block by double-clicking on it. See Fig. 5.

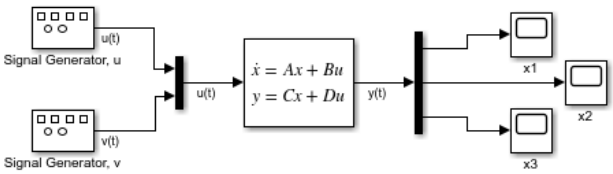

Figure 4: Simulink signal flow diagram of Fig. 3 implemented using the state-space block.

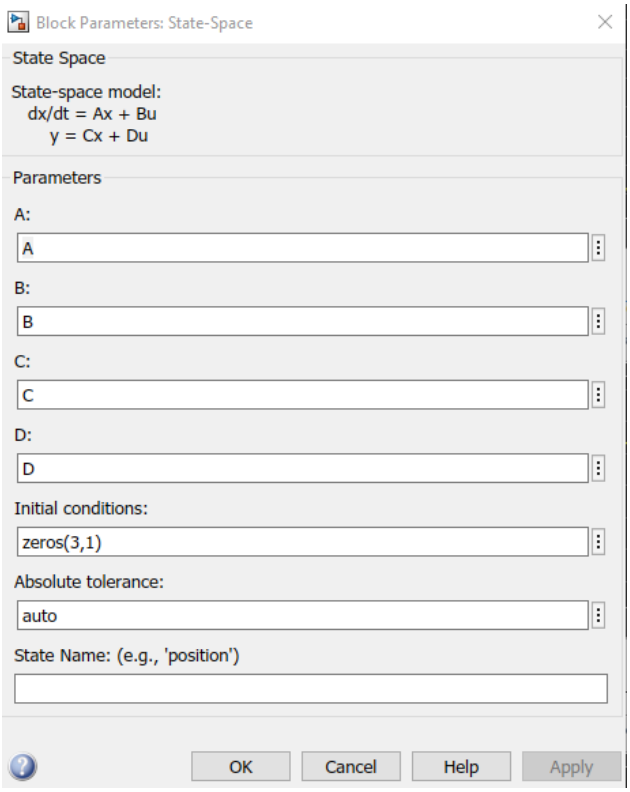

Figure 5: User-interface for the state-space block.

The values of these parameter matrices must be defined before the program can be run. The matlab code to define these matrices for eqn. (1) with the output defined as the full state vector (i.e., $y(t) = x(t)$) is:

```
A = [-5.0  0.0   0.2
      4.0   0.0  -2.0
      0.0   5.0  -0.2];
B = [1  0
 0  0
 0  1];
C = eye(3);
D = zeros(3,2);
```

which can be programmed into an m-file and run once in the command window before running the Simulink program.

Fig. 6 shows the time response of each state when $u(t)$ is a unit amplitude square wave with frequency 0.5 Hz and $v(t)$ is is a square wave with amplitude 10 and frequency 0.2 Hz.

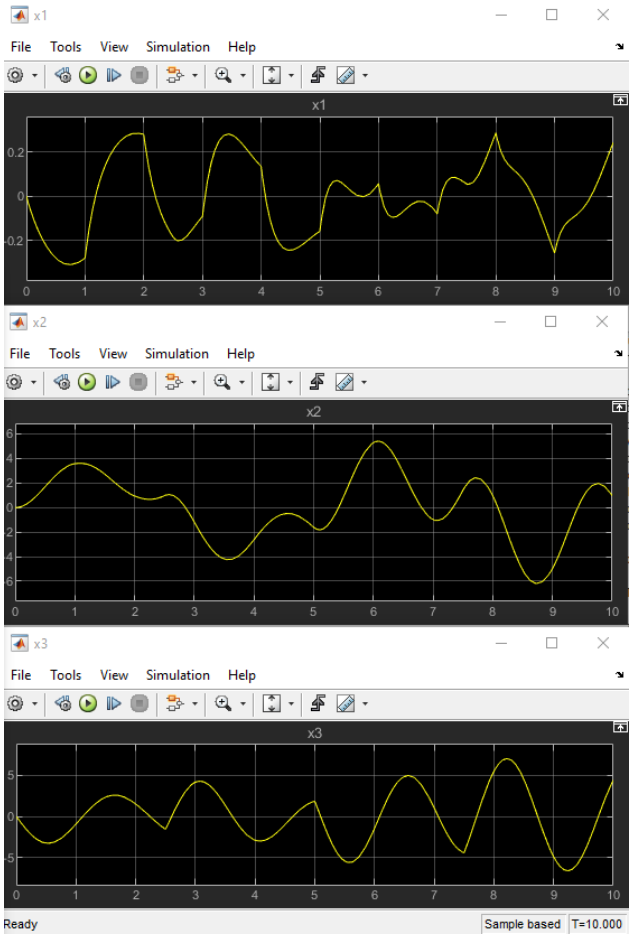

Figure 6: State trajectory corresponding to Fig. 3