ECE-205 Lab 6
Transfer Functions, Signal Flow Graphs, and Mason's Rule

Overview

In this lab we will first learn how to implement transfer function in both Matlab and in Simulink. Next we show how to write block diagrams as signal flow graphs, and lastly we will use the signal flow graphs in conjunction with Mason's rule to determine the overall transfer function of a system.

PART I: Representing Systems with Transfer Functions in Matlab and Simulink

Up until now we have represented our systems in terms of differential equations. In order to represent these systems in terms of transfer functions we only need to remember two simple things about Laplace transforms and transfer functions:

1) For transfer functions, we assume the initial conditions are zero (just as for the impulse response)

2) If $\mathcal{L}\{x(t)\} = X(s)$, then if we assume the initial conditions are zero we have $\mathcal{L}\left\{\frac{dx(t)}{dt}\right\} = sX(s)$ and $\mathcal{L}\left\{\frac{d^2x(t)}{dt^2}\right\} = s^2X(s)$

3) The transfer function of a system is the transform of the output divided by the transform of the input.

For you to do:

Show that the transfer function for our first and second order system representations,

$$\tau\dot{y}(t) + y(t) = Kx(t)$$
$$\ddot{y}(t) + 2\zeta\omega_n\dot{y}(t) + \omega_n^2 y(t) = K\omega_n^2 x(t)$$

are

$$H(s) = \frac{Y(s)}{X(s)} = \frac{K}{\tau s + 1}$$
$$H(s) = \frac{Y(s)}{X(s)} = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Next we need to learn to represent systems using transfer functions in both Matlab and Simulink Let's first start with the way Matlab represents polynomials. If you wanted to represent the polynomial $p(s) = 3s^2 + 2s + 1$ in Matlab, you would enter the coefficients into an array p, p=[3 2 1]. The coefficients always go from high to low. The last (rightmost) entry in the array is the coefficient of $s^0 = 1$, or the constant term. You must also always include a number for every coefficient, even if that coefficient is zero. For example, to enter the polynomial $p(s) = 2s^4 + 3s$ into Matlab, you would type, p = [2 0 0 3 0].

For the most part, our transfer functions will be the ratio of two polynomials. We need to use the Matlab function **tf,** to tell Matlab that we want to construct a transfer function made up of two polynomials. For example, to enter the transfer function

$$H(s) = \frac{s+1}{3s^2 + 2s + 1}$$

into Matlab, we would type

```
num = [1 1];   % numerator polynomial of the transfer function
den = [3 2 1];  % denominator polynomial of the transfer function
H = tf(num,den);  % construct the transfer function
```

We would, of course, just combine the three steps into one step,

```
H = tf([ 1 1],[3 2 1]);
```

If you do not put the semicolon at the end, Matlab will write the transfer function to the workspace (so you can check that you entered it correctly).

We will next show how to determine the response of a system represented by a transfer function to a step (or arbitrary input) in both Matlab and Matlab/Simulink. We will go through the steps for a first order system, then you will modify these for a second order system.

1) Open a new Matlab m-file (in a convenient folder). We will use one m-file (which you will attach to your memo) for this lab.

2) Set the variables **tau = 0.001** and **K = 2.0**

3) Enter the transfer function $G_p = \frac{K}{\tau s + 1}$ into Matlab.

4) Enter the final time variable **Tf = 0.01** (be sure to use at least one capitol letter so it is not confused with the transfer function command **tf**).

5) Use the **linspace** command to create a time vector t from 0 to Tf with 1000 sample points.

6) let's assume we want a step input with an amplitude of 0.1. There are many ways to generate this, but we will use the following command:

```
x=0.1*ones(1,length(t));
```

7) To simulate the system, we then use the **lsim** command as follows:

```
y = lsim(Gp,x,t);
```

8) You should now pretty up your plot so it looks like the plot in Figure 1. Note that I have really plotted more than one graph in this figure, which you will do shortly.

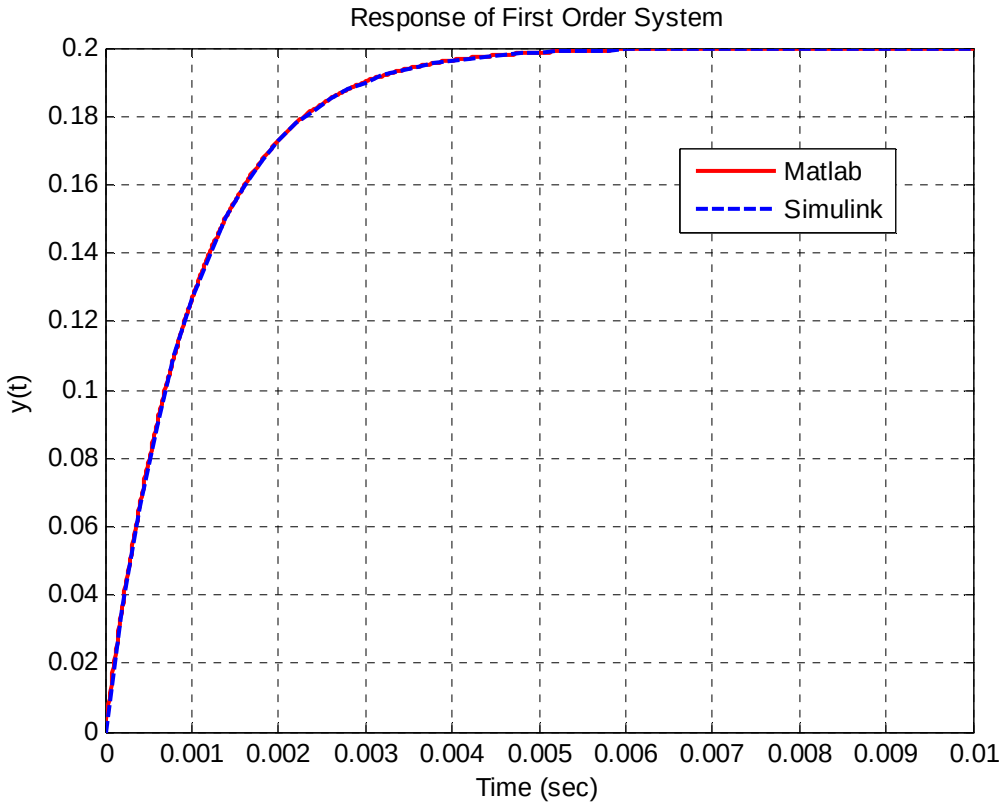

Figure 1. Response of first order system to a step of amplitude 0.1

We now want to prepare to use Simulink, and learn (and review) a few more commands as we go.

9) We will be using the times and input values from Matlab, so enter the command

xt = [t' x'];

into Matlab.

10) We although we already know the numerator and the denominator of the transfer function, let's learn to to extract them once we have a transfer function. We use the command

[num_Gp,den_Gp] = tfdata(Gp,'v');

to extract the numerator and denominator polynomials from the transfer function Gp.

11) Start Simulink from the Matlab window, and open a new model file. Save this model file as **openloop.mdl.**

12) Construct the model file shown in Figure 2. You will need to look in the **Sources Library** for the *from workspace* block and the clock block, the **Sink Library** for the *to workspace* blocks (be sure to save the data as an *array*, click on these boxes for this choice), and the **Continuous Library** for the *transfer function* block. Click on the transfer function block to enter the numerator as **num_Gp** and the denominator as **den_Gp.**

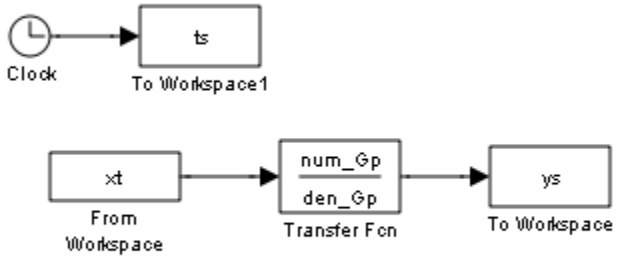

Figure 2. The openloop Simulink file.

13) Run the simulation from the m-file using the command,

sim('openloop');

14) At this point you have results from both Matlab (t,y) and Simulink (ts, ys). Modify your m-file to plot both of these results on the same graph. Use two different line types and colors, use a grid, label the x-axis etc. so your results look like that in Figure 1. *You need to include your graph in your memo.*

15) Now you need to modify your m-file so you can plot the results for a second order system with a natural frequency of 2000 rad/sec, a damping ratio of 0.2, and a static gain of 1.5. It is probably best to just copy and paste what you already have. If you have done everything correctly, you should get a graph like that shown in Figure 3. *You need to include your graph in your memo.*

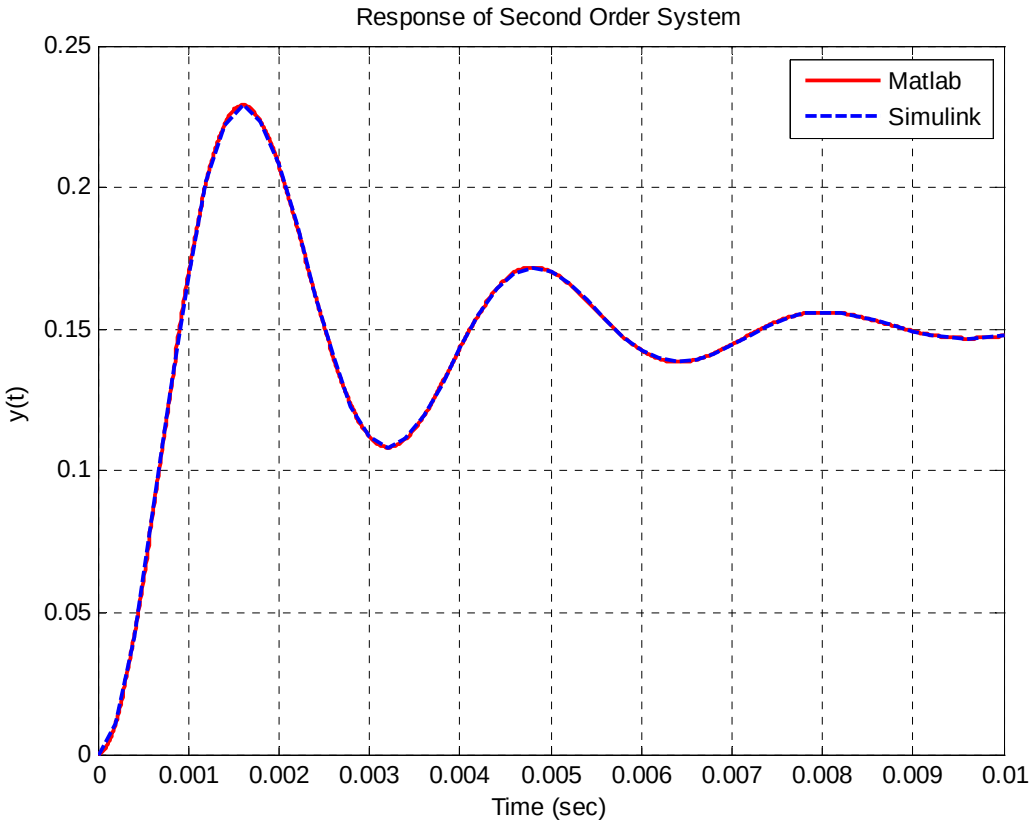

Figure 3. Response of second order system to a step of amplitude 0.1

PART II : Transfer Functions and Block Diagrams

For an LTI system, we know that we can relate the input and output using convolution and the impulse response, $y(t) = h(t) * x(t)$. Using Laplace transforms we have the equivalent relationship $Y(s) = H(s)X(s)$. Here $H(s)$ is called the transfer function, and is the Laplace transform of the impulse response $h(t)$. We can can represent interconnected systems in the time domain, as shown in Figure 1, where we have the input/output relationships

$$y(t) = h_1(t) * h_2(t) * x(t)$$
$$y(t) = \left[h_1(t) + h_2(t)\right] * x(t)$$
$$y(t) * [\delta(t) + g_1(t) * g_2(t) * h_1(t)] = x(t) * [g_0(t) * g_1(t) * g_2(t)]$$

As expected, these relationships are in terms of convolution.

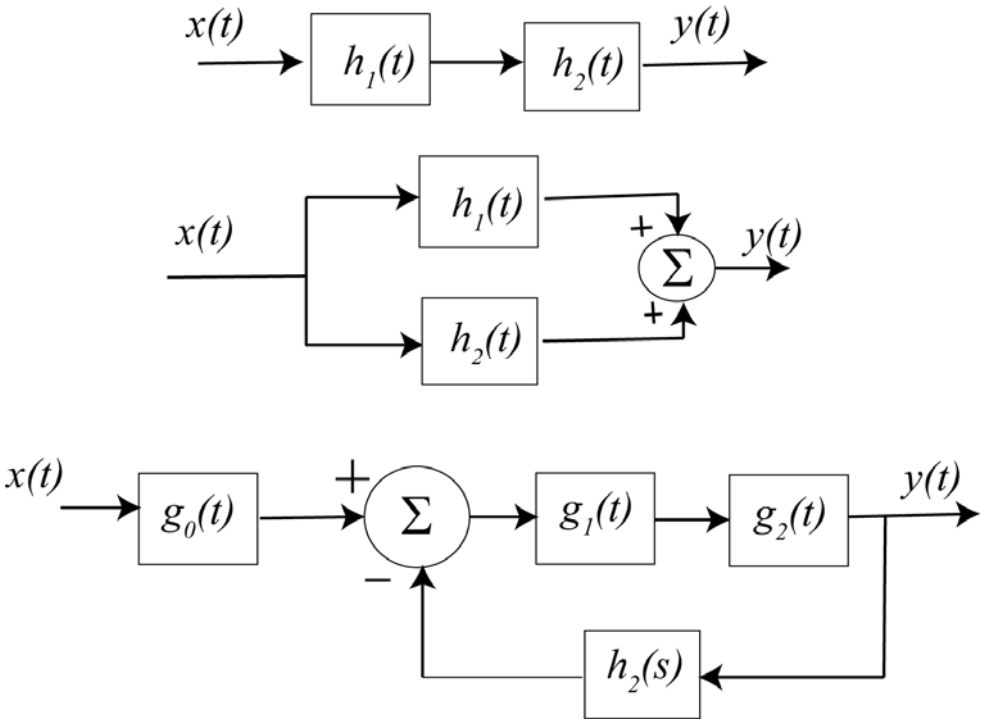

Figure 1. Time domain interconnection of systems.

Instead of dealing with time-domain functions and convolutions, we can deal with transfer functions and multiplications. The transfer function representation of the systems in Figure 1 can be represented in terms of transfer functions as shown in Figure 2. In the transfer function domain we have

$$Y(s) = H_1(s)H_2(s)X(s)$$
$$Y(s) = [H_1(s) + H_2(s)]X(s)$$
$$Y(s) = \frac{G_0(s)G_1(s)G_2(s)}{1 + G_1(s)G_2(s)H_1(s)}$$

Now we have replaced convolution with multiplication and division, which is easier to work with. As you become more familiar with transfer functions, you will realize that you can learn a great deal about the time-domain performance without actually performing the inverse Laplace transforms.

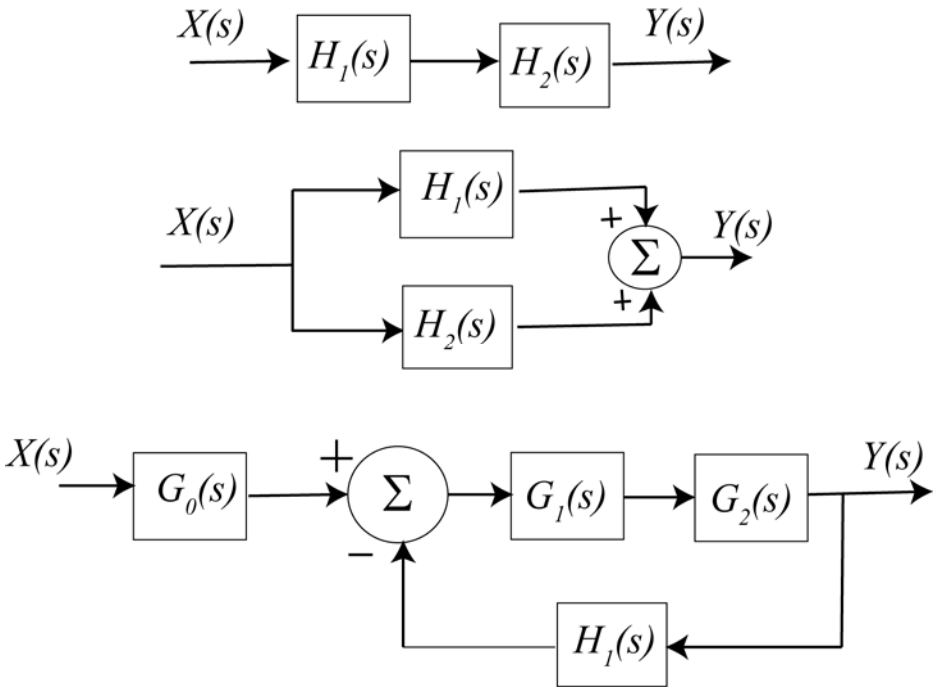

Figure 2. Transform domain representation of systems from Figure 1.

PART III: Signal Flow Graphs

As an alternative to block diagrams, we can represent the interconnection between subsystems utilizing signal flow graphs. A signal flow graph is composed of *directed branches* and *nodes*. Just like block diagrams, these graphs show the flow of signals throughout an interconnected system but rather than shown transfer functions in blocks, the transfer functions (or transmittances) are written over the directed branches. Finally, in a signal flow graph all of the branches into a node are summed. Hence, if you want negative feedback you need to include the negative sign in one of the transfer functions. Before getting into too much detail, you should examine the following examples of block diagrams and the equivalent signal flow diagrams. In the following signal flow graphs, a '1' indicates the transfer function is just unity.

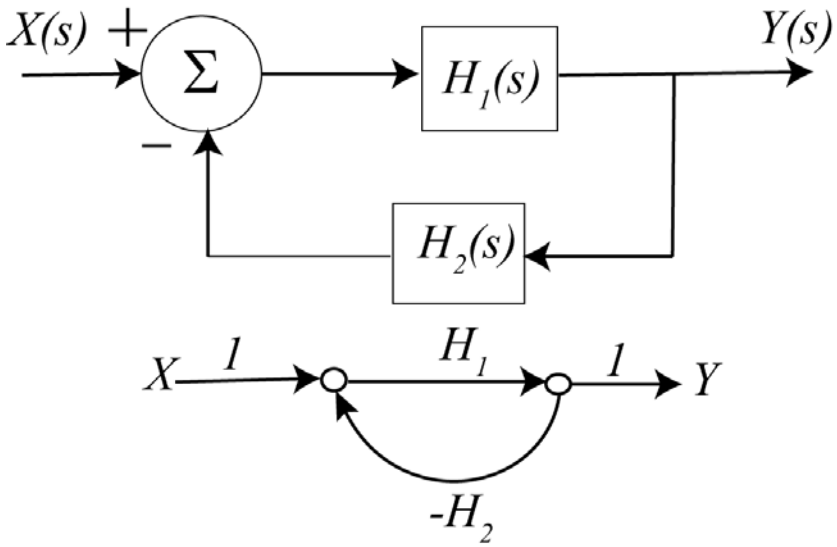

Figure 4. Signal flow graph of a simple feedback system.

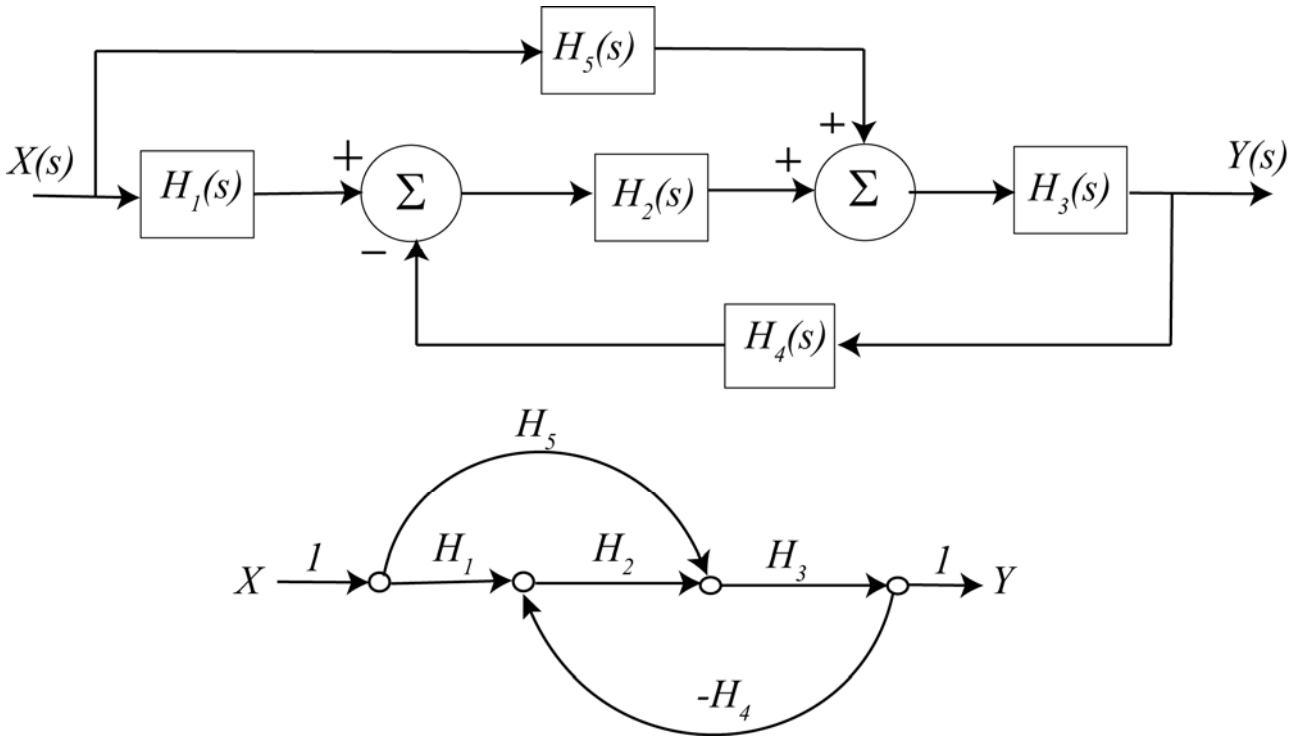

Figure 5. Second example of a block diagram and the equivalent signal flow graph.

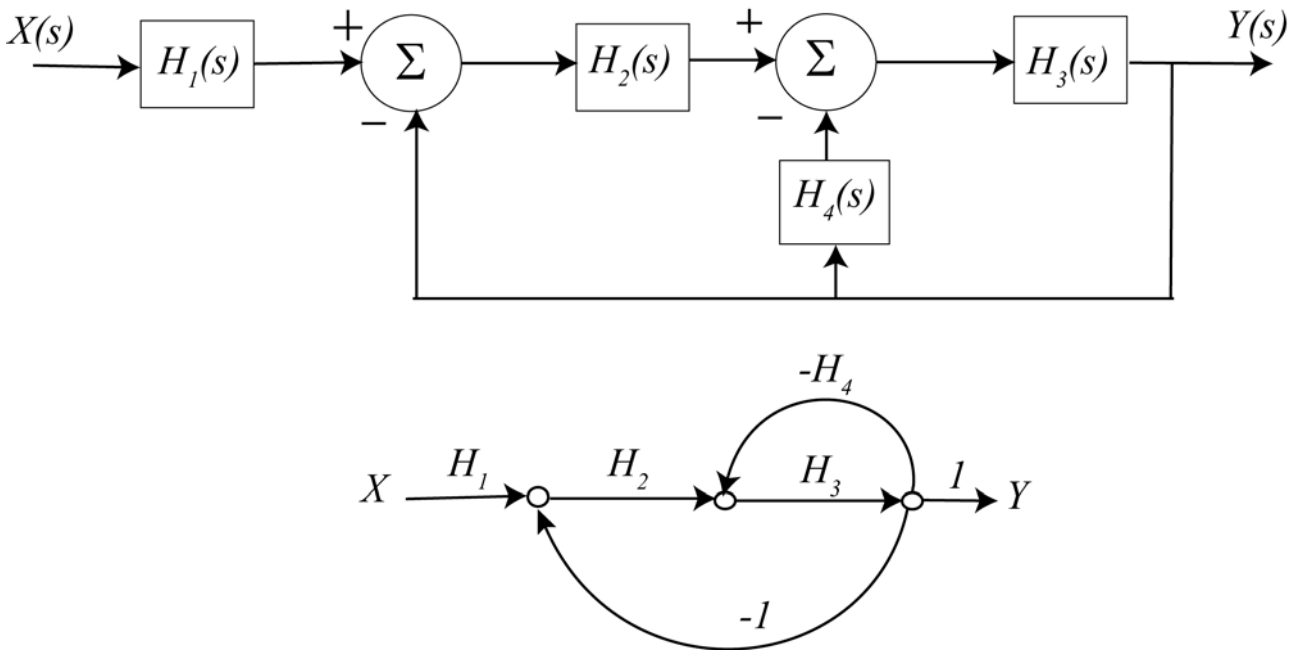

Figure 6. Third example of a block digraph and the equivalent signal flow graph.

PART V: Signal Flow Graphs and Mason's Rule

We would like to be able to determine the overall system transfer function form the input to the output of a system represented by a signal flow diagram. In order to do this we will need to introduce some definitions and apply them to an example problem. It is usually helpful to label some of the *paths* and *loops* on the signal flow graph to keep track of them. As you are going through the following definitions and steps, try to label these on the diagram. Let's assume we have the (fairly complicated) signal flow graph shown in Figure 7.

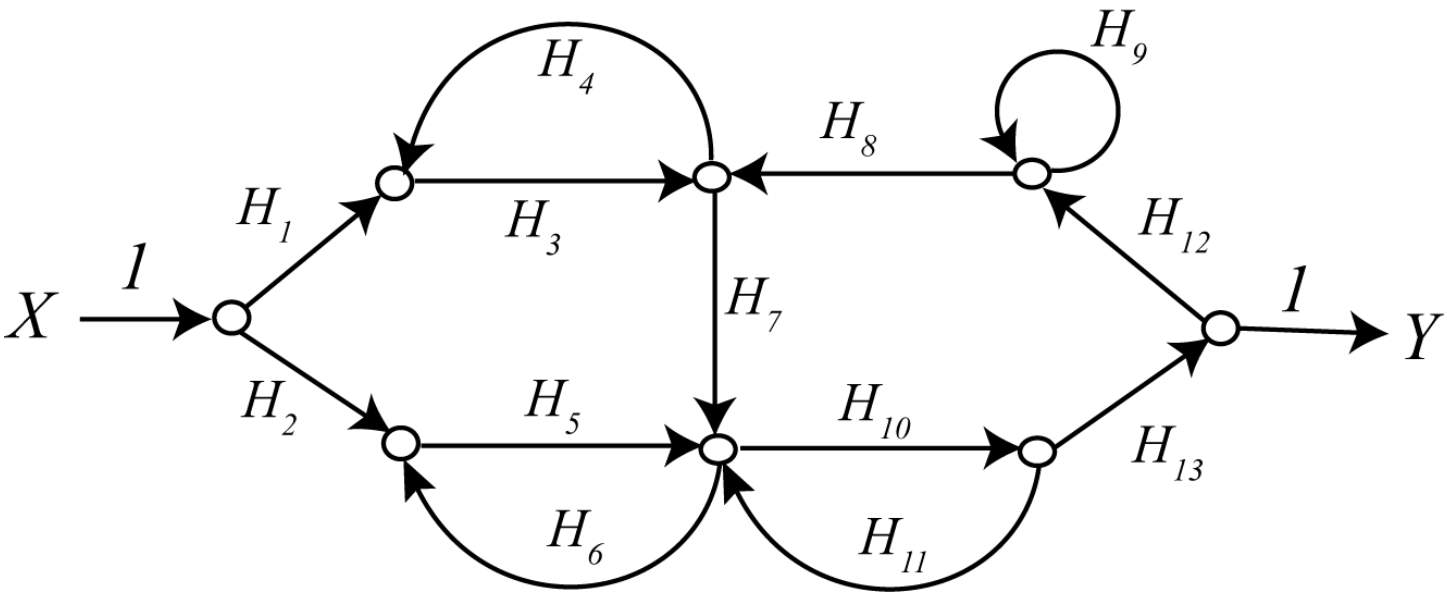

Figure7. Example signal flow graph used in the definitions example.

Definition: A *path* is any succession of branches, from input to output, in the direction of the arrows, which does not pass any node more than once. A *path gain* is the product of the transfer functions (transmittances) of the branches comprising the path.

For the signal flow graph in Figure 7, the path gains are (arbitrarily numbered):

$$P_1 = H_1H_3H_7H_{10}H_{13},\ P_2 = H_2H_5H_{10}H_{13}$$

Definition: A *loop* is any closed succession of branches, in the direction of the arrows, which does not pass any node more than once. The *loop gain* is the product of the transfer functions of the branches comprising the loops.

For the signal flow graph in Figure 7, the loop gains are (arbitrarily numbered):

$$L_1 = H_3H_4,\ L_2 = H_5H_6,\ L_3 = H_{10}H_{11},\ L_4 = H_9,\ L_5 = H_{10}H_{13}H_{12}H_8H_7$$

Definition: To loops are *touching* if they have any node in common. A path and a loop are *touching* if they have any node in common.

Definition: The *determinant* of a signal flow graph is denoted by Δ, and is computed by the formula

Δ =1 – (sum of all loop gains) + (sum of all products of gains of all combinations of 2 non touching loops) – (sum of products of gains of all combinations of 3 nontouching loops) + …

For the signal flow graph in Figure 7, the determinant is

$$\Delta = 1-(L_1+L_2+L_3+L_4+L_5) + (L_1L_2+L_1L_3+L_1L_4+L_2L_4+L_3L_4)-(L_1L_2L_4+L_1L_3L_4)$$

Definition: The *cofactor of a path* is the determinant of the signal flow graph formed by deleting all loops touching the path. To get the cofactor for each path, you write out the determinant and cross off all loops touching that particular path.

For the signal flow graph in Figure 7,

$$path\ P_1, \Delta_1 = 1-L_4$$
$$path\ P_2, \Delta_2 = 1-(L_1+L_4)+(L_1L_4)$$

Definition: The *transfer function* of the signal flow graph is given by the formula

$$H_{system} = \frac{P_1\Delta_1+P_2\Delta_2+P_3\Delta_3+\cdots}{\Delta}$$

For the signal flow graph in Figure 7, the system transfer function is then

$$H_{system} = \frac{P_1\Delta_1+P_2\Delta_2}{\Delta}$$

In most cases our systems are not as complicated as this one, but this method will work for any system provided we are systematic. Let's do some examples now. You should try these and then check your answers.

Example 1: For the signal flow graph in Figure 4 we have

$$P_1 = H_1, L_1 = H_1(-H_2) = -H_1H_2, \Delta = 1+H_1H_2, \Delta_1 = 1$$
$$H_{system} = \frac{H_1}{1+H_1H_2}$$

Example 2: For the signal flow graph in Figure 5 we have

$$P_1 = H_5H_3,\ P_2 = H_1H_2H_3,\ L_1 = -H_2H_3H_4,\ \Delta = 1-L_1,\ \Delta_1 = 1,\ \Delta_2 = 1$$
$$H_{system} = \frac{H_3H_5+H_1H_2H_3}{1+H_2H_3H_4}$$

Example 3: For the signal flow graph shown in Figure 6, we have

$$P_1 = H_1H_2H_3, L_1 = -H_2H_3, L_2 = -H_3H_4, \Delta = 1-L_1-L_2, \Delta_1 = 1$$
$$H_{system} = \frac{H_1H_2H_3}{1+H_2H_3+H_3H_4}$$

.

For your to do: Determine the transfer function for the signal flow graphs in figures 8-10. The answers are:

$$H_{system} = \frac{H_1 + H_3}{1 + H_1 H_2},\ H_{system} = \frac{H_3(1 + H_1 H_2) + H_1}{1 + H_1 H_2},$$

$$H_{system} = \frac{H_1 H_4 + H_1 H_2 H_3}{1 - H_1 H_2 H_6 - H_2 H_3 H_5 - H_4 H_5 + H_1 H_2 H_3 + H_1 H_4}$$

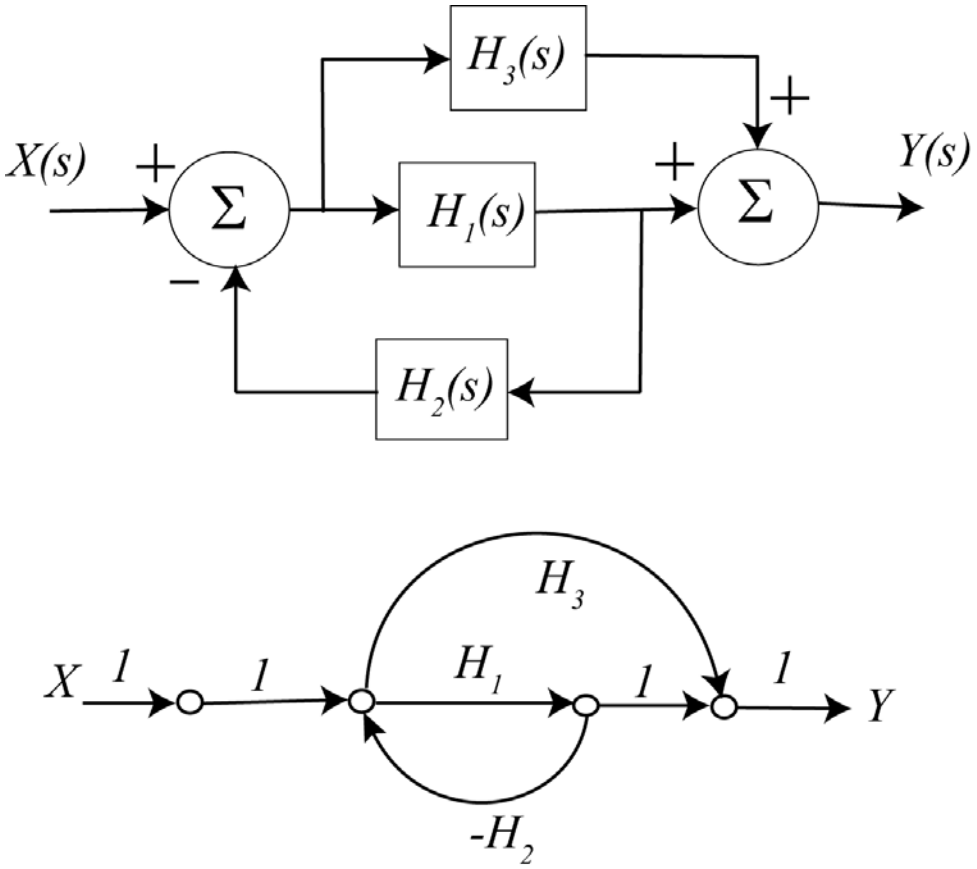

Figure 8. Block diagram and signal flow graph for practice.

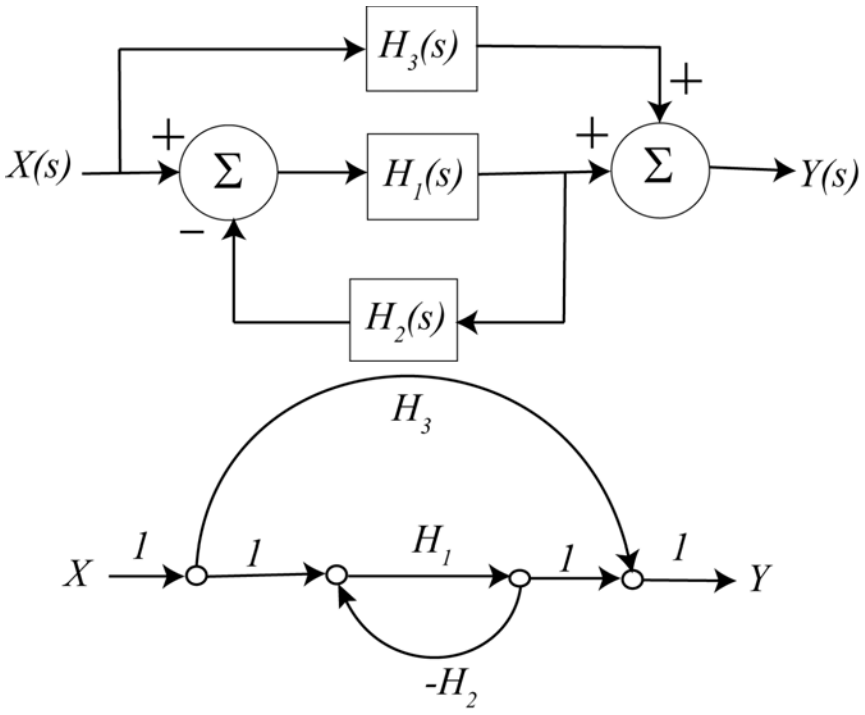

Figure 9. Block diagram and signal flow graph for practice.

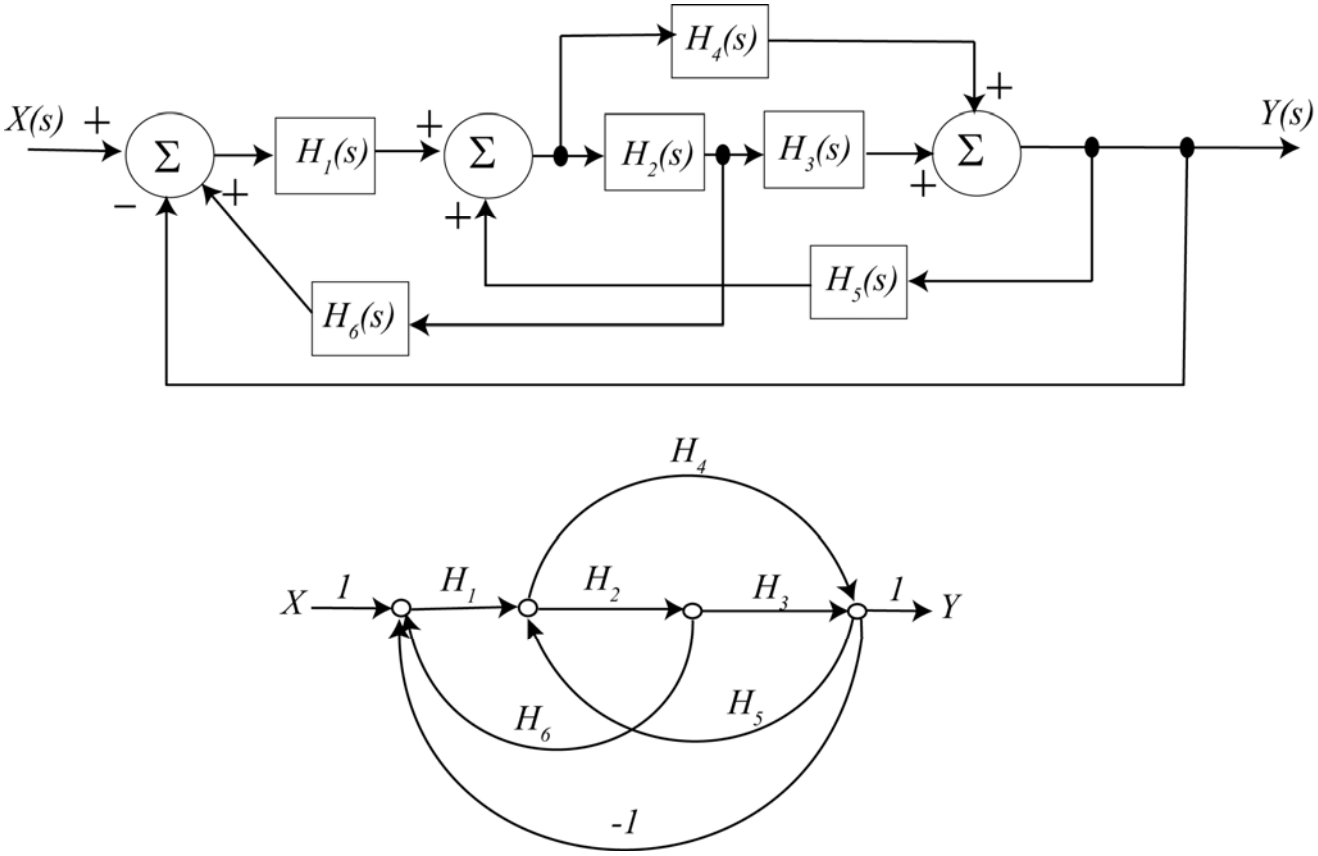

Figure 10. Block diagram and signal flow graph for practice.

For you to turn in: For the block diagram shown in Figure 11, determine a corresponding signal flow diagram and show that the closed loop transfer function is

$$H_{system} = \frac{G_1G_2G_3 + G_4(1 - G_1G_2H_1 + G_2H_1 + G_2G_3H_2)}{1 - G_1G_2H_1 + G_2H_1 + G_2G_3H_2}$$

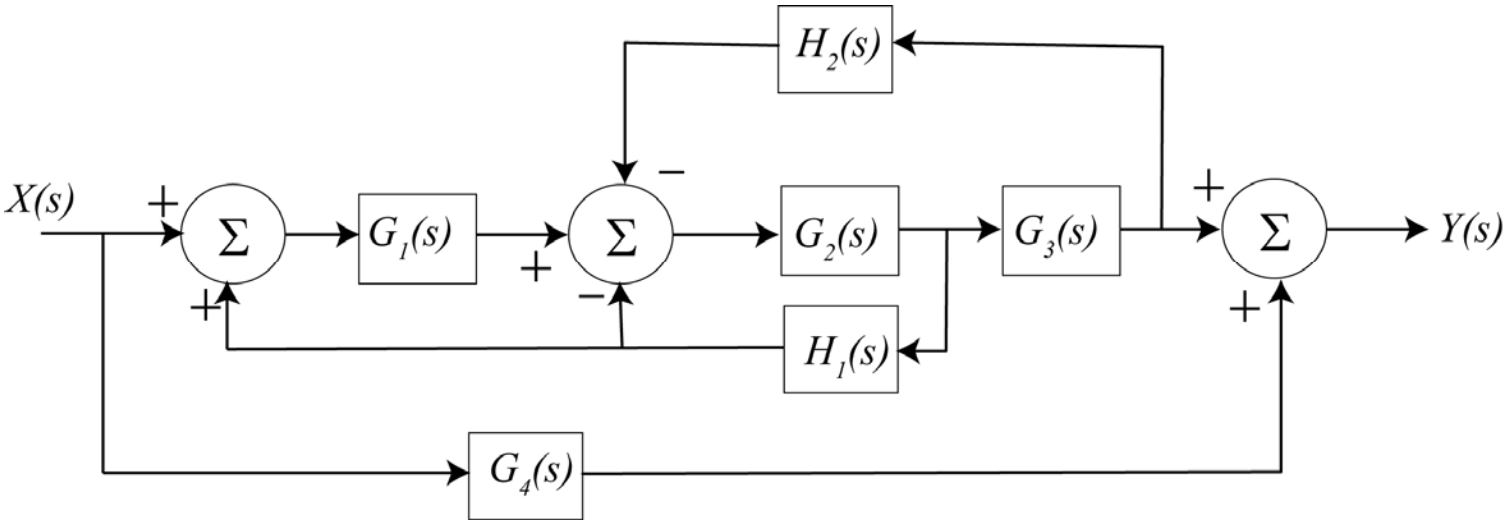

Figure 11: Block diagram for your practice.