

Lab1 - Introduction to MATLAB and SIMULINK

Goals:

Become familiar with some basic MATLAB commands, Create simple script and function m-files, Use Simulink to simulate systems, and create data plots from a Simulink simulation.

Step 1 – MATLAB Workspace

We want to observe the step response of a second order system modeled by the transfer function below for a
zeta = 1.0

Create a time vector, t, whose elements start at t = 0 sec, incremented by 0.1 sec, and stop at t = 5 sec. This is done with the following command

```
>> t = [0:0.1:5]
```

Enter the specific zeta value.

```
>> zeta = 1;
```

Next the transfer function must be entered in a MATLAB format. Only the coefficients of the numerator and denominator polynomials are entered in entered into MATLAB. The coefficients are entered left to right in order of descending exponential terms. Thus our transfer function is entered as follows.

Create a numerator vector for the transfer function's numerator polynomial.

```
>> num = [1]
```

Create the denominator vector for the transfer function's denominator polynomial.

```
>> den = [1 2*zeta*1 1]
```

Now the step response for the system modeled by this transfer function can be found. How is the MATLAB step function used to find the step response of a system? Fortunately, MATABL has an excellent online help function to assist users. To see how to use step enter the following at the MATLAB command prompt.

```
>> help step
```

Print the 'help step' output. Turn this in as part of your lab report.

Get the step response of system using the following step response command form

```
>> [y1out,x1] = step(num,den,t);
```

Now to observe the calculated step response, the results of the step response command need to be plotted. Check the online help to learn how to use MATLAB's plot command.

To produce a plot of step response use the MATLAB *plot* command

```
>> plot(t, y1out)
```

Add the grid to the plot using the MATLAB *grid* command

```
>> grid
```

The plot is incomplete without labels. MATLAB provides commands to label the axes and the graph itself. Using the *help* command lookup the *xlabel*, *ylabel* and *title* commands to understand how to add label to x- and y-axes as well as a title to the graph. These commands can be found on a menu option if you're working with a Macintosh.

It's instructive to review the step response and the roots of the denominator of the transfer function. MATLAB has a command to find the roots of a polynomial. Using the *help* command learn how to use the *roots* command and find the roots of the transfer function's denominator polynomial, *den*. Store the roots of the polynomial using the variable *roots1*.

An important feature of MATLAB is the ability to store and load data. These commands allow users to process data, save it and retrieve it for later processing and/or analysis. Use the *help* function to learn how to save data using the *save* command. If you're using a Macintosh, the *save* command can be accessed through a menu option. Save the step response, time and roots data if may be required for later work.

Using the following command will save the time, step response, and denominator roots in the mat-file *step1.mat*.

```
>> save step1 t y1out roots1
```

Step 2 – Script Files

MATLAB is an extensible program. This means users can write their own programs for MATLAB to process. These programs can contain any MATLAB commands as well as conditionals and loop commands. This includes if-then statements, while loops and for loops. There are two types of programs written for the MATLAB, the script and the function file. Both of these programs are frequently referred to as m-files in MATLAB literature.

A script m-file is a set of MATLAB commands. Script files are beneficial for a user who needs to repeat a set of commands many times. Instead of typing the commands individually over and over in the workspace, the commands are entered in a script file, and the script file is run. Running a script file causes MATLAB to process each command contained in the script file. It's very simple to use script files. After a script file is created entering the script file's name at the command prompt just like any other MATLAB command can process its commands. Thus, the script file's name becomes another command available to any MATLAB user.

In this step a script file will be written to calculate and plot the step response for different system models using the transfer function from step one. We want to observe the step response for the transfer function from step one as zeta varies over a set of values. Instead of repeating the commands from step one over and over for each zeta value. A script file will be created that will repeat the step one commands (calculating and plotting) for each zeta. Step one commands will be combined with two new MATLAB commands to create an m-file. The new functions are *hold* and *for*. Use the *help* command to learn about both of these new commands.

The *for* command will be used to create the range of zeta (0,0.2,0.4,0.6,0.8,1.0) to be investigated. The *for* command defines a block of commands to be repeated and creates a counter. The loop or block of commands to be repeated are between the *for* and *end* commands. The loop is repeated while the counter is incremented from the user specified initial value till the counter exceeds the final value.

The *hold* command is used to plot several step responses on the same graph. The *hold* command directs the *plot* command to use the currently held graph for plotting.

Below is an example script file, myrootsolv.m, for calculating the roots of the denominator of the transfer function. This example can be extended to create the script file required to calculate and plot step response.

```
for zeta = 0:0.2:0.8
    myroots = roots([1 2*zeta*1 1])
end
```

To create an m-file on the Macintosh use the NEW menu option on the File menu. This starts the m-file editor and allows you to enter MATLAB commands for the m-file.

After creating the script file, use it to calculate and plot the step responses for each different value of zeta on one graph.

Your print out of your script file must be initialed by your TA and turned in as part of your Lab report.

Step 3 – Function Files

The other type of MATLAB program file, the function m-file, will be created in this step. The function m-file can contain any MATLAB commands like the script m-file. One important distinction from the script file is a function m-file is passed data, processes a set of commands using the data, and may return data.

A function m-file is similar to a script file. It contains a set of MATLAB commands to process the data passed to it. The main difference is the first line of the function m-file. The first line specifies this m-file is a function by using the keyword function. The first line also contains names for the variables passed to the function as well as variable to be passed back (see example function below).

It's very simple to use function m-files. A function file is called by entering the function file's name and the name of variables to be passed to the function at the command prompt. Many of MATLAB commands behave just like user written function files (e.g. plot(x,y), step(num,den,t), etc...). Once a user function file is created it becomes another command available to any MATLAB user.

The below sample function m-file, solvroot.m, is passed a zeta value and returns the roots, roots3, to the denominator polynomial from our transfer function from step one. This function can be extended to calculate and return the step response data for our transfer function from step one given a zeta value and a time vector.

```
function roots3 = solvroot(zeta)
    poly = [1 2*zeta*1 1];
    roots3 = roots(poly);
```

Functions returning more than one value use a return vector. Each variable passed back is an element of a vector and each element is separated by a comma. Use the help function with MATLAB command QR for an example of a function that returns two values.

Create a function m-file that calculates the step response for the transfer function from step one given a specific zeta value. Use it to calculate a step response for a zeta value of 1.2.

Print your function m-file and get the TA to initial the printout. Turn this in as part of your lab report.

On one graph plot the step response for the transfer function from step one for the following range of zeta; (0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2). The graph should have the following properties.

x-axis label: time (sec)

y-axis label: step response

graph title: Step 2 - Step Response vs. Time Graph for Varying Zeta

Each different plot should be labeled with its zeta value.

This graph must be initialized by the TA and turned in as part of your lab report.

In the lab report there must be a table relating zeta values and transfer function denominator polynomial roots. Also there must be a short discussion about the relationship between zeta values, roots and step responses.

Step 4 – SIMULINK

Simulink is a block diagram oriented program. Dragging blocks from several sub-system windows into a new system window and connecting them creates Block diagrams. After the system block diagram has been constructed, blocks can be added which display the signals or which write the data to variables that can be listed or plotted from within the MATLAB command window.

At the MATLAB prompt, start simulink by entering the following command.

```
>> simulink
```

A new window containing icons for the sub-systems blocks appear.

Open a new system window by selecting NEW ... from the File menu. Drag this window into the lower right corner by pointing to the title bar, holding down the mouse button and moving the mouse. This window is referred to as the system window.

Open the Sources by double clicking on the Sources icon. Drag the Step Fcn source into the system window. If you double click on this icon after it is in the system window, you see the parameters that can be used to specify the step. Click on cancel. Click on the sources window to activate it, and then click on the close box to the left of the title. The sine wave source can also be found in the Sources window.

Double-click the Linear icon. Drag the Transfer Fcn to the right of the Step Fcn icon. The angle bracket (>) pointing out of the step fcn icon represents its output port and the angle bracket pointing into the Transfer Fcn icon represents its input port. To connect these two blocks, click on either the output or input port of one block and drag to the other block I/O port to draw a connecting line, then release the mouse button. A line with an arrowhead should now connect the blocks with the arrowhead showing the direction of the data flow. Double-clicking on the Transfer Fcn icon shows where to specify the numerator and denominator polynomial coefficients for any order transfer function. If the Pole-Zero icon is used the transfer function can be specified by its poles zeros. If you decide to delete a block from the system window, click on it and select Cut from the Edit menu or press the Delete key. When all the required items are taken from the Linear window.

Double click on the Sinks icon and drag the Scope to the right of the Transfer Fcn icon and connect the output of the transfer fcn to the scope input. Double click on the scope and change the Horizontal Range to 5. Select the Parameters from the simulation menu. Change the stop time to 5. Close the parameter window by clicking on ok. Make the scope the active

window by clicking on it. Run the simulation by selecting the Start from the Simulation menu. You should see the step response of the $1/(s+1)$ on the scope.

How to get a print: You can print the screen to get a copy of the response, but this is not a very good way of getting a plot. The way to get plots is to use the plot command in MATLAB. To do this, open the Sinks icon by double clicking it and drag two copies of the To Workspace icon to the system window. Connect one icon to the output transfer function. Close the sink window and open the Sources window. Select the Time icon and drag it to the system window. Connect the output of the Time icon to the other To Workspace. Double click the To Workspace icon connected to the Time icon and change the variable name to time. Run Simulink. Now both the output and time variables are available to the command window for plotting or further analysis.

We will use Simulink and MATLAB 's command window to simulate the following oven temperature control system and car cruise control system.

Find the closed loop transfer function and sensitivity transfer function for the oven temperature control system (This is not to be done on Simulink or MATLAB). (As a reminder the closed loop transfer function is defined as y/r and sensitivity transfer function is defined as e/r).

Use the help function in the command window to learn how to use MATLAB's bode function to generate Bode plots for both the closed loop and sensitivity transfer functions. Enter the closed loop transfer function into MATLAB and using the bode command create a Bode plot for the closed loop transfer function. Repeat this to create a Bode plot from the sensitivity transfer function.

Print the two Bode plots and have the LAB TA initial both of these plots. These will be part of your lab report.

Using your favorite technique find the steady state values for the error, e , and the output temperature, y . Compare this to the step responses plots obtain in the lab using Simulink. In the report compare your theoretical steady state values with your simulation step responses.

Next build the oven temperature control system block diagram in Simulink and simulate it with the reference and disturbance signals. **Print the system block diagram and get the TA to initial it.**

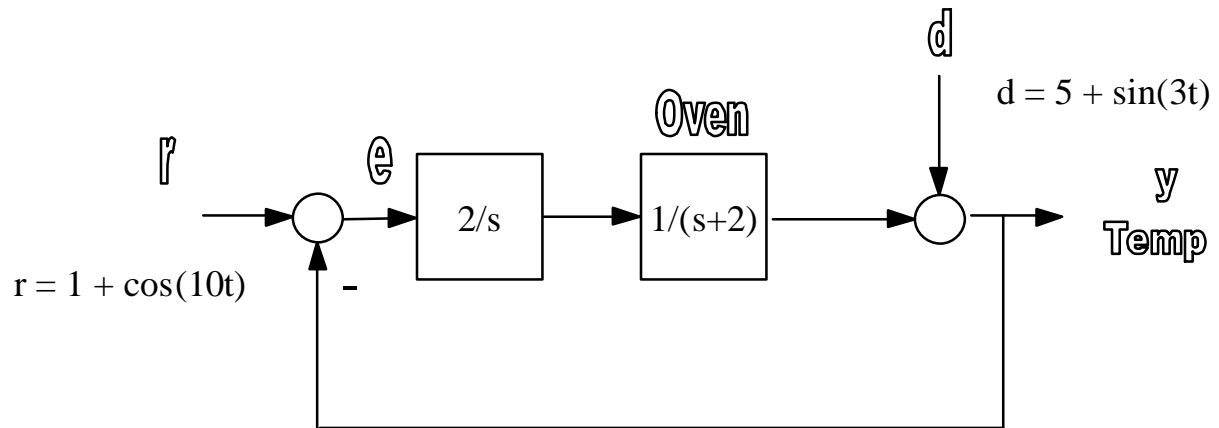
Plot the reference command, error signal, control signal, and output signal (zero disturbance signal magnitude) against time on the different graphs. Plot the reference command, error signal, control signal, and output signal against time on the different graphs. Have the TA initial these plots and turn them in as part of your lab report.

Next build the car's cruise control system in Simulink and simulate it with all disturbances. **Print the system block diagram and get the TA to initial this printout.**

Plot the reference command (speed), error signal, control signal, and output signal (speed) against time on the different graphs. Have the TA initial these plots and turn them in as part of your lab.

Adjust the disturbance's magnitudes and see how increasing their magnitudes affect the system output.

Oven Temperature Control System



Car Cruise Control System

