ECE411 - Laboratory Exercise #1

Introduction to Matlab/Simulink

This laboratory exercise is intended to provide a tutorial introduction to Matlab/Simulink. Simulink is a Matlab toolbox for analysis/simulation of interconnections of dynamic systems. The model building environment is GUI based, and it will be used heavily throughout the rest of the course/laboratory. All the exercises in this assignment can be done entirely in Matlab/Simulink.

1) Running, Plotting, Printing: In order to see a demonstration Simulink diagram type *sldemo_househeat* at the Matlab prompt. Open the *scope* block, labeled "PlotResults" by double clicking, and then run the simulation using the buttons or pull down menus provided. Print the plot of the simulation output (scope block) and print the simulation model itself.

2) Model Building: Figure 1 shows a Simulink model which represents the motor gear system in the Controls Laboratory, with a PID controller implemented in feedback around it. Launch the simulink library browser from within Matlab by using the button or typing *simulink*. Then open a new model (using button or pull down menus), and build a copy of the above model. This is achieved by dragging components from the library to the model and connecting them using the mouse. Double clicking a box then allows you to edit the contents, such as entering values for the transfer function (as shown). For the PID block set the proportional gain to 0.05, and the integral and derivative gains to zero.

Look around at the (many) available blocks in Simulink. You will certainly need to look in *Sources, Sinks, Continuous, Math Operations* and *Signal Routing*. Note that there is no block for "Pulse Input", but I made that myself from basic components using the *Create Subsystem from Selection* command after selecting a section of the diagram and right-clicking. The contents of the box are shown in figure 2. You can even use the *Mask/Create Mask* command (again select and right-click) to take a subsystem and use it to make your own custom library blocks (with GUI interfaces for the parameters).

When you have built a copy of the model save it with the name "gear" (it will actually be saved as gear.slx). You can then launch this model later from Matlab simply by typing *gear* at the command line. Go under *Simulation* to *Model Configuration Parameters* and set the simulation time to 8 seconds. Then run the simulation and print the results from the scope block. You should get a plot like figure 3 which shows the commanded response and the actual system response (note the autoscale and zoom buttons on the scope). Of course note that in order to get the correct commanded response you will need to enter appropriate values for the two step input blocks that make up your pulse input.

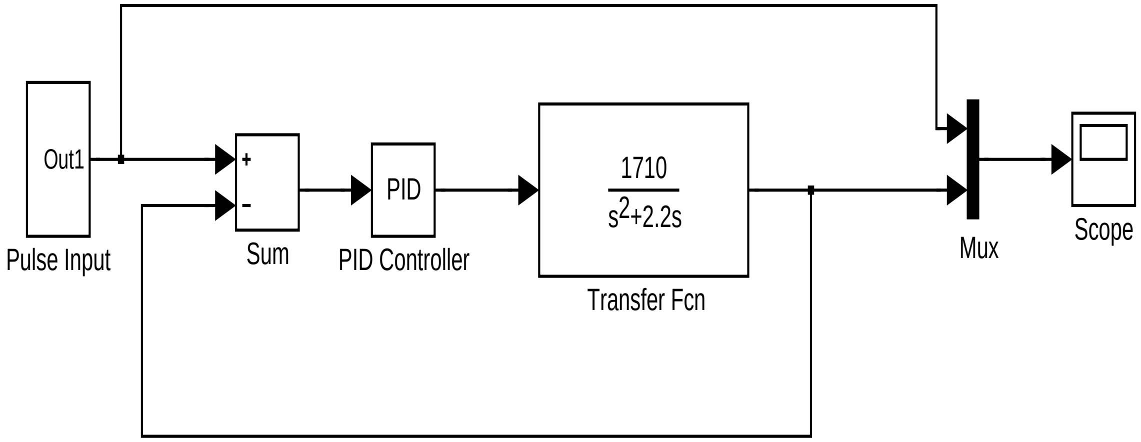

Figure 1: Simulink model of motor gear drive system

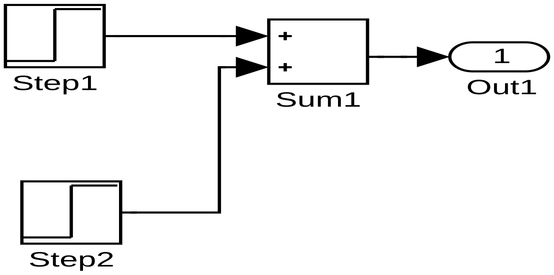

Figure 2: Simulink pulse input subsystem

Having completed this exercise you should have a model and a simulation run that essentially reproduce the figures shown here. Now try varying the parameters of the PID controller and see how they affect the closed-loop control system (note that you can enter variable names in Simulink Blocks if you like, and it will read them from the Matlab workspace). In particular carry out the following simulation studies:

i) Vary the Proportional gain up and down and note the effect.
ii) Now try adding in the Integral and/or Derivative gains. You do not need to generate large numbers of plots but plot a few of the results and discuss how the different controller parameters (Proportional, Integral, and Derivative) affect the closed loop performance.
iii) See if you can manually tune the PID controller to get a good step response.
iv) See how well your controller rejects disturbances by adding a disturbance signal between the controller and the plant (just use a summing junction to add your disturbance signal to the output of the PID controller).
v) See how sensitive your closed loop system is to modeling errors by perturbing the open loop plant model (i.e., change the Transfer Function parameters).

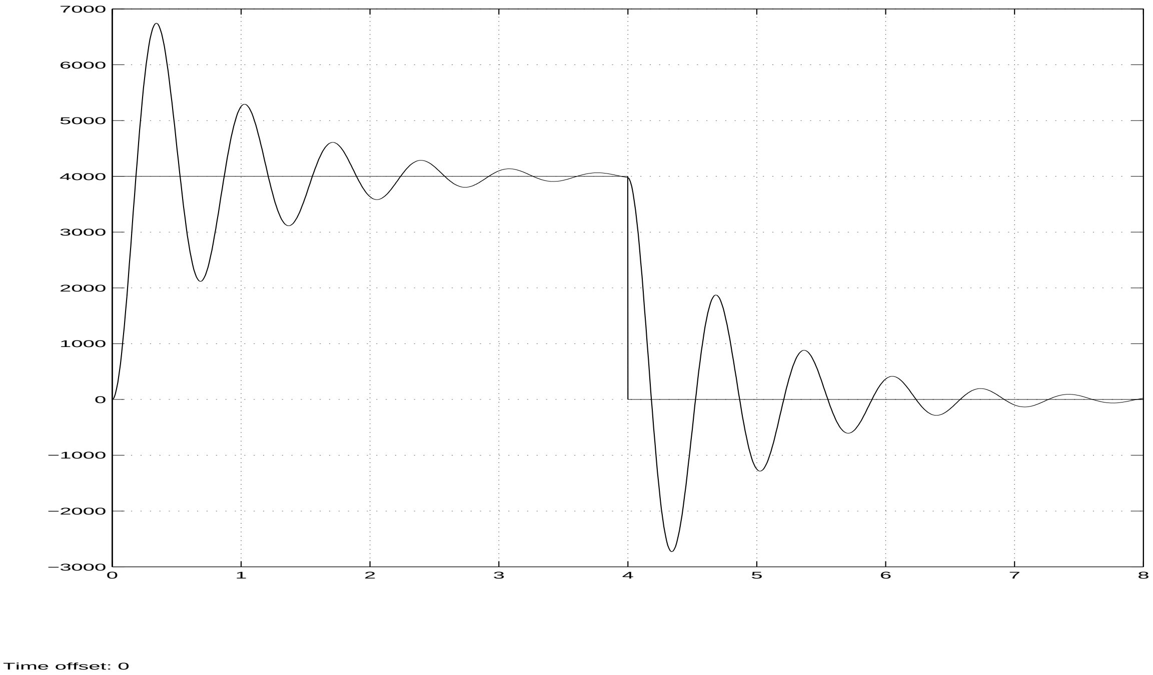

Figure 3: Scope block output from simulation

These trial-and-error simulation studies should give you some ideas about how the closed-loop control is working. Later in the semester we will revisit this problem with the systematic analysis and design tools we have learned in class, and try them out both in simulation (as here) and on the actual hardware system in the laboratory. In addition to the above studies you can find more Matlab/Simulink examples by typing *demo*. Further information is also available at the following Websites:

https://www.mathworks.com/help/index.html
http://ctms.engin.umich.edu/CTMS/index.php?aux=Home

The Mathworks Website is a general reference site for Matlab/Simulink and their various toolboxes. The second Website is specifically designed as a tutorial introduction to Control System analysis and design using Matlab and Simulink.

APPENDIX: Introduction to Matlab

This material is NOT required
Included only as an OPTIONAL tutorial

This material is intended to provide a tutorial review of Matlab, which will be used throughout the rest of the course. It will NOT be graded.

1) Vectors and Matrices: Data entry in Matlab is achieved by separating columns of matrices by spaces, and rows of matrices by semi colons. Thus in order to enter the matrix

$$a = \begin{bmatrix} 1 & 3 & 7 \\ 2 & 5 & 6 \end{bmatrix}$$

one would type

$$a = [1\ \ 3\ \ 7; 2\ \ 5\ \ 6]$$

Having defined matrices then one can compute algebraic manipulations of them easily in Matlab. Matlab provides a wealth of matrix functions, each of which has online help to describe their operation. Type the following commands to learn about the functions and then carry out the exercises below:

help arith
help inv
help eig

You can also browse the available help by typing *helpwin*. Look around in the directories *general, ops, lang, elmat, elfun*, and *matfun* to familiarize yourself with what is available in Matlab. Of course we will not be using all of the available tools, but some we will use heavily, and you will build your understanding of Matlab as the course progresses.

Enter the following matrices (the imaginary number $j = \sqrt{-1}$ is recognized in Matlab):

$$a = \begin{bmatrix} 1 & 3 & 7 \\ 2 & 5 & 6 \end{bmatrix}$$

$$b = \begin{bmatrix} 4 & 7 & 9 \\ 8 & 1 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

$$c = \begin{bmatrix} 1-j & 1+j & 3 \\ 2-4j & 8+3j & 6j \\ 0 & 0 & 7 \end{bmatrix}$$

Now compute the following quantities:

i) bc
ii) $b^{-1}c$
iii) ab
iv) $b + c$
v) ba^T (where a^T denotes the transpose of a)
vi) The eigenvalues of b
vii) The eigenvalues of c

You may find the following commands useful

help diary
help save
help load

2) Polynomials: Matlab facilitates easy manipulation of polynomials, by storing the coefficients as vectors. The polynomial

$$s^3 + 3s^2 - 7s + 8$$

would be stored as the vector

$$p = [1 \;\; 3 \;\; -7 \;\; 8]$$

Evaluating the polynomial and calculating the roots is then straightforward in Matlab. Type the commands

help polyval
help roots

and look in *helpwin* under *polyfun*. Then carry out the following exercises:

i) Evaluate $s^3 + 2s^2 + 4s - 8$ for $s = 1$
ii) Evaluate $s^3 + 2s^2 + 4s - 8$ for $s = 2 - 4j$
iii) Evaluate $s^3 + 4s$ for $s = 2j$
iv) Compute the roots of $2s^3 - 3s^2 + 6s + 7$
iv) Compute the roots of $s^3 - 12s^2$
v) Compute *all* the cube roots of 1

3) Plotting and Printing: Matlab has an array of commands for plotting, labeling/editing plots, and printing. Type the following commands:

```
help plot
help print
```

Note that there are “See also” commands suggested at the end of every help menu. You can also look in *helpwin* under *graph2d* (and others). As a simple example the following series of commands generates a plot of a cosine wave:

```
t = 0:0.01:2*pi;
y = cos(t);
plot(t,y);
title('Plot of a cosine wave');
```

Note the first line is used to automatically generate a time vector (see also *linspace* and *logspace*). The semicolons keep the commands silent. Generate and print plots of the following functions. Please include titles and axis labels.

i) $\sin^2(t) - cos(3t)$ for $0 \leq t \leq 2\pi$
ii) $t^3 - 3t^2 - 2t$ for $0 \leq t \leq 5$

4) Scripts and Functions: Any collection of Matlab commands can be gathered together as a script, saved in an “m-file”. For instance save the following commands in a text file called plotit.m

```
numhar = 5;
numpoints = 300;
t = linspace(0,2*pi,numpoints);
y = zeros(1,numpoints);
for ii = 1:numhar
        y = y + ((-1)^(ii+1))*(1/(2*ii-1))*cos((2*ii-1)*t);
end
plot(t,y)
title('Harmonic decomposition of a square wave')
```

You can then run this straight from Matlab (just type *plotit*) to build a square wave from its harmonics and plot it. The formulae for the harmonic series coefficients comes from the Fourier Series expansion (verify that). Note that this is a script in that all the variables exist in the workspace. You can also write subroutines, which use local variables. Type *help function* to see the syntax for doing this. Carry out the following programming tasks:

i) Write a function from the above script to generate and plot a square wave from its harmonics. The inputs should include the number of harmonics and number of points. The outputs should include the time and amplitude vectors for the square wave. Plot the wave for various values. Can you see Gibb's phenomenon when the number of harmonics is large?
ii) Write a function which gives back the sum of the squares of the absolute values of the elements of a matrix, i.e., it evaluates the function

$$f = \sum_{i=1}^{n} \sum_{j=1}^{m} |a_{ij}|^2$$

for any $n \times m$ matrix a. Test your routine on the matrices a, b, c from question 1).