University of Toronto
Department of Electrical and Computer Engineering
ECE356S – Linear Systems and Control –

EXPERIMENT 1

Introduction to MATLAB and Simulink for Systems Analysis

1 Purpose

The purpose of this experiment is to introduce you to MATLAB and Simulink as tools for systems analysis. MATLAB is used primarily for numerical computations, while Simulink provides a block diagram style representation of systems for analysis, simulation, and design.

2 Preparation

Before you go to your lab session, read through carefully the description of this laboratory provided in the following pages. Useful terms and commands are highlighted in **bold font**, while laboratory work is written in ***bold italics***. Identify the parts you have to do in the lab.

3 Linear Algebra

MATLAB is heavily based on linear algebra for doing its computations. In MATLAB, matrices are entered row by row. The end of a row is indicated by a ";" or a carriage return. For example,

$$A = [1\ 2\ 3;\ 4\ 5\ 6;\ 7\ 8\ 9]$$

produces the matrix

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Individual rows and columns can be extracted from A. For example, $A(:,1)$ and $A(1,:)$ give the first column and row of A, respectively.

I. **Inverse of a Matrix and Solution of Linear Equations:**

MATLAB provides a command to compute the inverse of an $n \times n$ matrix A, namely **inv(A)**. However, if the objective is to solve a linear system of equations $Ax = b$, there is a better alternative than using $x = inv(A) * b$, namely $x = A\backslash b$. The MATLAB operation "\" is called **mldivide** or matrix left divide. This is illustrated in the following calculation.

(a) Save the following code to a file and name it inv_matrix.m. This is a script file which you can

run in MATLAB.

```
n = 500;
Q = orth(randn(n,n));
d = logspace(0,-10,n);
A = Q*diag(d)*Q';
x = randn(n,1);
b = A*x;
tic, y = inv(A)*b; toc
err = norm(y-x)
res = norm(A*y-b)
pause
tic, z = A\b; toc
err = norm(z-x)
res = norm(A*z-b)
```

Consult the MATLAB documentation to understand what each line means. Then run "inv_matrix" in MATLAB and note the outputs.

(b) Similarly, there is an operation called **mrdivide** or matrix right divide, A/B, which has the effect of AB^{-1} when B is invertible. Let $A = randn(4,4)$. ***Compute the inverse of A using inv, mldivide, and mrdivide. Show your results to your TA. Explain how you would compare the accuracy of the three methods.*** For square matrices with low dimension, all three calculations should give comparable accuracy.

II. Eigenvalues and Eigenvectors

Of great importance in the analysis of linear systems are the notions of eigenvalues and eigenvectors. A scalar λ is an eigenvalue and a non-zero vector x is an eigenvector of a linear transformation A if $Ax = \lambda x$. If A is a matrix, then eigenvalues and eigenvectors are only defined if A is square.

(a) Let A be given by

$$A = \begin{bmatrix} 7 & 2 & -3 \\ 4 & 6 & -4 \\ 5 & 2 & -1 \end{bmatrix}$$

We will use this A from item (a) to (e) in this part. ***Use the MATLAB command $[V, D] =$ eig(A) to determine the eigenvalues and eigenvectors of A.***

(b) The command **eig(A)** gives eigenvectors which are normalized to have norm 1. If you compute the eigenvectors by hand, you are more likely to come up with eigenvectors which are not normalized. ***For the matrix in II(a), you can use the command* eig(A,'nobalance') *to produce more "readily recognizable" eigenvectors.*** Recall that eigenvectors are determined up to a scalar multiple. ***From the results of eig(A,'nobalance'), determine 3 eigenvectors of A whose entries are all integers.***

(c) For manual computation of eigenvalues, usually you determine the characteristic polynomial of A given by $\det(sI - A)$. Then you find the roots of the characteristic polynomial, i.e., find solutions of the characteristic equation $\det(sI - A) = 0$. ***Determine the characteristic polynomial of A using the command* poly(A)*, and determine the eigenvalues by applying the command* roots *to the resulting polynomial. Compare the answer with those from (a).***

(d) The eigenvalue-eigenvector equations can be written as one matrix equation

$$AV = VD,$$

with D a diagonal matrix consisting of the eigenvalues of A. ***From the results of IV(a), determine norm(AV-VD). Show more significant digits in MATLAB using the command* format long*. From this determine the* exact *values of the eigenvalues and eigenvectors (up to a scalar multiple). Now verify that $AV - VD = 0$.***

(e) Recall that if the eigenvalues of A are distinct, the eigenvectors are linearly independent. In that case, the V matrix computed using eig is invertible and that $V^{-1}AV = D$. This process is called diagonalizing A. ***Check that the matrix in II(a) can be diagonalized.***

(f) When A has repeated eigenvalues, it may not be possible to diagonalize A. Suppose A has λ as a repeated eigenvalue, then $\det(\lambda I - A) = 0$ and the number of times λ repeats as roots is called its algebraic multiplicity. Thus the following matrix

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

has 1 as its only eigenvalue with algebraic multiplicity 2. ***Determine by hand calculation the eigenvector corresponding to the eigenvalue* 1*, and verify there is only one independent eigenvector. Try using eig on this A. Does the resulting [V,D] satisfy $AV = VD$? Is V invertible?***

(g) When A cannot be diagonalized, one can transform it to a form called the Jordan (canonical) form. The MATLAB command is **jordan**. ***For the matrix***

$$A = \begin{bmatrix} 0 & 4 & 3 \\ 0 & 20 & 16 \\ 0 & -25 & -20 \end{bmatrix}$$

find its Jordan form.

4 Ordinary Differential Equations and Transfer Functions

Control systems studied in this course are modelled by in the time domain by state equation of the form

$$\dot{x} = Ax + Bu \tag{1}$$
$$y = Cx + Du, \tag{2}$$

or by input/output models of the form

$$y^{(n)}(t) + a_1 y^{(n-1)}(t) + a_2 y^{(n-2)}(t) + \cdots + a_n y(t) = b_0 u^{(m)}(t) + \cdots + b_m u(t), \quad \text{with } m \leq n$$

In the state equation, the transfer matrix from u to y is given by

$$G(s) = C(sI - A)^{-1}B + D.$$

In this course, we focus on single-input single-output (SISO) systems, i.e., u and y are scalar-valued. In this case, the transfer function $G(s)$ is a scalar-valued proper rational function. In the case of an input/output model, the transfer function $G(s)$ is given by

$$G(s) = \frac{b_0 s^m + b_1 s^{m-1} + \cdots + b_m}{s^n + a_1 s^{n-1} + \cdots + a_n}$$

MATLAB provides tools to solve such linear systems. We illustrate some of these tools using a second order system.

Consider a second order system described by the differential equation

$$\ddot{y} + 2\dot{y} + 4y = 4u$$

The transfer function from the input u to the output y for this system is given by

$$G(s) = \frac{4}{s^2 + 2s + 4} \tag{3}$$

To model this as a state equation, let

$$x(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix}$$

Then the state x satisfies the differential equation

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -4 & -2 \end{bmatrix} x + \begin{bmatrix} 0 \\ 4 \end{bmatrix} u \tag{4}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x \tag{5}$$

MATLAB provides a data object which conveniently describes the state space system (1)-(2). It is created by the command **sys=ss(A,B,C,D)**.

(a) ***Create the sys object corresponding to the second order system described by* (4)-(5).**

(b) One can study the response of the second order system due to particular inputs such as a step (to get the step response), or the response due to nonzero initial conditions with no input, or generally with nonzero initial conditions and inputs. ***Here, determine the step response using the command* [Y,T,X]=step(sys)*, where sys is the object you created using the ss command. Use* plot(T,X) *to plot the trajectories of both states.***

(c) To determine the response due only to initial conditions, it would be convenient to create a second sys object using sys_init=ss(A,0,C,0) (0 is a matrix of appropriate dimension). ***Use the command* [Y,T,X]=initial(sys_init,x0) *to determine the response to initial conditions when* $x0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$*. Again plot the state responses.***

(d) In general, the system may have both nonzero initial conditions as well as inputs. For example, the commands

```
t=0:0.01:20;
u=sin(t);
```

define an input vector u whose components are given by the values of $\sin(t)$ at various times specified by the time vector t. ***Use* [Y,t,X]=lsim(sys,u,t,x0) *to produce the response of the second order system due to the initial condition and sinusoidal input defined above. Plot the state trajectories.***

(e) If you are given a state space description, the command **ss2tf** generates the numerator and denominator polynomials of the transfer function associated with the state space system. ***Use the* ss2tf *command, followed by the* tf *command to produce the transfer function for the systems described by* (4)-(5)*. Verify that it is the same as* (3)*.***

5 Introduction to Simulink

Simulink is a MATLAB package which provides a graphical user interface to build systems using blocks from its library. Parameters that are set or changed in MATLAB are passed along to Simulink, allowing them to complement each other.

To start Simulink, type **simulink** ***in the MATLAB command window, or click Simulink in the menu. This will bring up the Simulink Start Page. Click on Blank Model to create a new blank model. Click on the Library Browser (the icon with 4 squares) to bring up the Simulink Library Browser.*** In this window, you will find the blocks that Simulink provides, as well as the toolboxes that have been installed. The blocks used most frequently in this course can be found under **Continuous, Math Operations, Sinks, Sources**. ***Scan through each of these categories to familiarize yourself with the blocks provided.***

(a) There are several different ways you can simulate a state space linear system in Simulink. The simplest is to use the state space block, if you can mainly interested in the input/output behaviour. ***Bring in the state space block and use the same*** (A, B, C, D) ***matrices as in*** **(4)-(5)**. ***Connect to the input port a unit step. Change the start time of the step to*** **0**. ***Connect the output port to a scope. Click the "*****simulate*****" button and examine the output on the scope.***

(b) The disadvantage of the state space block is that you can only access the inputs and outputs. ***One trick to overcome this limitation is to change the*** C ***matrix in the state space block to the identity so that the outputs are the states. Now connect the states to a gain matrix*** C. ***Note that by default, the gain matrix multiplies the input element-wise. Under block parameters, you need to change "Multiplication" to matrix multiplication. Finally connect the output to a scope and check that the simulation gives the same output as in (a). Connect the states from the output of the state space block to another scope, and check that one of the 2 signals displayed is the same as the output*** y.

(c) ***To represent the signals in a state space system in terms of more basic components, start a new model and bring in the*** **integrator** ***block. The input to the integrator block is*** $\dot{x} = Ax + Bu$ ***and the output is*** x. ***The right hand side for*** $\dot{x}$ ***requires you to put in some gain matrices and a*** `sum` ***block so that the output of the*** `sum` ***block,*** $\dot{x}$, ***satisfies the state equation. The output of the integrator block,*** x, ***can now be connected to the gain matrix*** C, ***whose output goes into a scope. With the input a unit step, check that the output on the scope is again the same as those in (a).***

(d) The approaches described in (a)-(c) only work for linear systems. When the state space description is nonlinear, you need to build the Simulink model using individual states as the signals. ***To understand how this is done, in a new Simulink model, bring in 2*** **integrator** ***blocks. For the system described by*** **(4)-(5)**, ***the input to the first*** **integrator** ***block should be*** $\dot{x}_2$. ***Its output is*** $x_2 = \dot{x}_1$, ***which is the input to the second*** **integrator** ***block. Using scalar gains, complete the Simulink model so that the step response shown in a scope is again the same as that of (a). Show all your Simulink models and output scope displays from (a) to (d) to your TA.***