University of Toronto
Department of Electrical and Computer Engineering
ECE311H1 – Introduction to Control Systems

EXPERIMENT 0

Introduction to MATLAB

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.

(c) A linear equation $Ax = b$ may have no solutions, exactly one solution, or an infinite number of solutions. When A is not square, the solution, if it exists, cannot be determined using $inv(A) * B$. Let

$$A = \begin{bmatrix} 1 & 3 & -2 & 0 & 2 & 0 \\ 2 & 6 & -5 & -2 & 4 & -3 \\ 0 & 0 & 5 & 10 & 0 & 15 \\ 2 & 6 & 0 & 8 & 4 & 18 \end{bmatrix}$$

$$b = \begin{bmatrix} -4 \\ -11.6 \\ 18 \\ 7.6 \end{bmatrix}$$

Determine the solution of $Ax = b$ using mldivide. Call this solution x_1. Verify that another solution is given by

$$x_2 = \begin{bmatrix} 2 \\ 0 \\ 3 \\ 0 \\ 0 \\ 0.2 \end{bmatrix}$$

Finally, yet another solution is given by $x_3 = pinv(A) * b$ where the command* pinv *stands for pseudoinverse. There is an infinite number of solutions to this equation. The solution x_3 has the smallest norm, i.e., $\|x_3\|$ is smallest among all solutions to $Ax = b$. ***Using the command* norm*, check that $\|x_3\| < \|x_1\| < \|x_2\|$. What do you observe to be the main difference between x_3 and x_1?***

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

$$\begin{aligned} \dot{x} &= Ax + Bu && (1) \\ y &= Cx + Du, && (2) \end{aligned}$$

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 this 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

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

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

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 1 \\ -4 & -2 \end{bmatrix} x + \begin{bmatrix} 0 \\ 4 \end{bmatrix} u && (3) \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} x && (4) \end{aligned}$$

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 (3)-(4).***

(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*** $\boldsymbol{x0} = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}$***. 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.***

Concluding Remark: This introduction to various MATLAB commands provides you with some system analysis tools. For control systems design and simulation, it is often more convenient to use Simulink. This will be taken up in future experiments.