Lab 0: Introduction to Matlab and Simulink

October 13, 2020

1 Purpose

The purpose of this laboratory 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. Please attach a single MATLAB script for each separate task. This will not only simplify the work of the TAs in grading your work but will also help you with building the habit of having scripts that automate your work. Furthermore save these files so that they can be a useful reference for you in the future assignment and courses. At the end of this file you will find a list with all the files you are required to submit.

2 Matlab scripts

Read the MATLAB guide about how to create scripts here!

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:

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

produces the matrix:

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

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.

- How would you select the third column?

- Without trying on MATLAB, what do you think would be the result of `A(2,2)`? Check the result on MATLAB.

3.1 Inverse of a Matrix and Solution of Linear Equations:

You will create and upload a unique script for all the calculations you operate in this section 3.1. Name this file `ex_3_1.m` and add comments where required. MATLAB provides a command to compute the inverse of an n x 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 `A\b`. The MATLAB operation "\" is called mldivide or matrix left divide. This is illustrated in the following calculation.

1. Save the following code to a file and name it `inv_matrix.m`. This is a script file which you can run in MATLAB. **You can use the lines below as an example of how your submitted files should look like**.

```
n = 500; % dimension of the matrix
Q = orth(randn(n,n));
d = logspace(0,-10,n);
A = Q*diag(d)*Q';
x = randn(n,1); % the solution of the equation, known to
    compare methods
b = A*x;
tic, y = inv(A)*b; toc % solve equation with inv and time
err_inv = norm(y-x)    % compute error
res_inv = norm(A*y-b)
pause
tic, z = A\b;  toc % solve equation with mldivide and time
err_ml = norm(z-x)  % compute error
res_m; = norm(A*z-b)
```

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

2. Similarly, there is an operation called `mrdivide` or matrix right divide, `A/B`, which has the effect of solving the equation $XA = B$ with respect to X. This method can be used only when A and B have the same number of columns. In this exercise we will compute the solution of the equation $AX = B$ using all the three comands `inv`, `mldivide`, and `mrdivide`, in a similar way as above.

   Define `A = randn(4, 4)` and also `X = randn(4, 4)`. Find the solution to the equation $AX = B$ using the commands `inv`, `mldivide`, and `mrdivide`. In order to use `mrdivide` successfully, notice that solving $AX = B$ with respect to X is the same as $X^T A^T = B^T$ were A^T denotes the transpose of A.

   Be sure that the code used for the comparison of the three results is included in the submitted file `ex_3_1.m`. For square matrices with low dimension, all three calculations should give comparable accuracy.

3.2 Eigenvalues and Eigenvectors

You will create and upload a unique script for all the calculations you operate in this section 3.2. Name this file `ex_3_2.m` and add comments where required. 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.

1. Let A be given by

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

   We will use this A for items 1 to 4 in this part. Use the MATLAB command `[V, D]=eig(A)` to determine the eigenvalues and eigenvectors of A.

2. 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 1.

3. The eigenvalue-eigenvector equations can be written as one matrix equation

$$AV = VD, \tag{3}$$

   with D a diagonal matrix consisting of the eigenvalues of A. From the results of point 1, determine `norm(AV-VD)`. Show more significant digits in MATLAB using the command `format LONG` before the expression you evaluate. From this determine the exact values of the eigenvalues and eigenvectors (up to a scalar multiple). Now verify that $AV - VD = 0$.

4. 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 (2) can be diagonalized. Write one line in the comments justifying why it is diagonalizable or not.

5. 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

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{4}$$

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. Report this eigenvector in the comments. Try using `eig` on this A. Does the resulting `[V,D]` satisfy $AV = VD$? Is V invertible? Report the answers in the comments.

6. When A cannot be diagonalized, one can transform it to a form called the Jordan (canonical) form. The MATLAB command is `jordan`. Find the Jordan form for the matrix:

$$A = \begin{bmatrix} 0 & 4 & 3 \\ 0 & 20 & 16 \\ 0 & -25 & -20 \end{bmatrix} \tag{5}$$

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:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{6}$$

or by input/output models of the form

$$y^{(n)}(t) + a_1 y^{(n-1)}(t) + a_2 y^{(n-2)}(t) + \cdots + a_n y(t) = b_0 u^{(m)}(t) + \cdots + b_m u(t), \qquad \text{with } m \leq n \tag{7}$$

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

$$G(s) = \frac{b_0 s^m + b_1 s^{m-1} + \cdots + b_m}{s^n + a_1 s^{n-1} + \cdots + a_n} \tag{8}$$

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

4.1 ODE and transfer functions

You will create and upload a unique script for all the calculations you operate in this section 4.1. Name this file `ex_4_1.m` and add comments where required. You will need to upload the required plots as well. 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

$$G(s) = \frac{4}{s^2 + 2s + 4} \tag{9}$$

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

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -4 & -2 \end{bmatrix} x + \begin{bmatrix} 0 \\ 4 \end{bmatrix} u \tag{10}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x \tag{11}$$

MATLAB provides a data object which conveniently describes the state space system (6). It is created by the command `sys=ss(A,B,C,D)`.

1. Create the `sys` object corresponding to the second order system described by (10)-(11).

2. 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. Save the plot[1] as `.png` file and name it `plot_4_2.png`.

3. Use the command `[Y, T, X]= initial (sys, x0)` to determine the response to initial condition $x0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and zero input. Again plot the state responses and save it as a `.png` file and name it `plot_4_3.png`.

4. 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 and save it as a `.png` file and name it `plot_4_4.png`.

[1] We highly discourage using the screenshot function of your OS. Use instead the MATLAB built-in commands for exporting plots as images. For example MATLAB documentation provides useful information. A simple way to save your plot is (in the plot window): `File -> Export as`. In MATLAB you can control virtually everything about your plots, try for example to add a meaningful legend, it will be useful for the next labs. If you have problems contact your lab TAs.

5. 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 to produce the transfer function for the systems described by (10)-(11). The function returns two vectors with the coefficients of the numerator and denominator polynomials, properly ordered. Verify that it is the same as (9).

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.

5.1 Simulate and visualize data using Simulink

You will create and upload all the Simulink model files `.slx` you used for this part of the lab. Save a different model for all the points listed below.

1. 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 are mainly interested in the input/output behaviour. Bring in the state space block and use the same (A, B, C, D) matrices as in (10)-(11). 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. Save the scope plot as a `.png` file and name it `plot_5_1.png`. Save your simulink model and name it `model_5_1.slx`.

2. 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 1. 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. Save the scope plot as a `.png` file and name it `plot_5_2.png`. Save your simulink model and name it `model_5_2.slx`.

3. 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 1. Save the scope plot as a `.png` file and name it `plot_5_3.png`. Save your simulink model and name it `model_5_3.slx`.

4. To completely expose the system graphically, you will build a Simulink model using individual states as the signals. In a new Simulink model, bring in 2 integrator blocks. For the system described by (10)-(11), 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 at point 1. Save the scope plot as a `.png` file and name it `plot_5_4.png`. Save your simulink model and name it `model_5_4.slx`.

6 Submission

Be sure to submit a unique zip file with all the files listed below:

- `ex_3_1.m`
- `ex_3_2.m`
- `ex_4_1.m`, `plot_4_2.png`, `plot_4_3.png` and `plot_4_4.png`
- `ex_5_1.slx` and `plot_5_1.png`
- `ex_5_2.slx` and `plot_5_2.png`
- `ex_5_3.slx` and `plot_5_3.png`
- `ex_5_4.slx` and `plot_5_4.png`