# University of Toronto
Department of Electrical and Computer Engineering
ECE356S - Linear Systems and Control
Laboratory 1
THE MAGNETIC BALL-SUSPENSION SYSTEM: MODELLING AND SIMULATION USING MATLAB

## 1 Introduction and Purpose

The purpose of this experiment is to familiarize you with the simulation of a magnetic ball-suspension system using MATLAB. You will also become familiar with modelling a nonlinear system using SIMULINK and performing linearization.

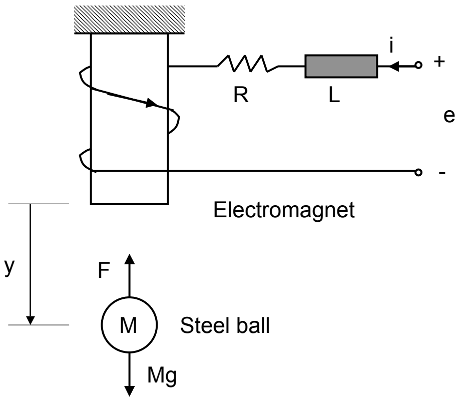

Figure 1: block diagram of a magnetic ball-suspension system

Consider the block diagram of a magnetic ball-suspension system (Figure.1). The position $y$ of the steel ball of mass $M$ can be controlled by adjusting the current $i$ in the electromagnet through the input voltage $e$. The electromagnet is represented as a series RL circuit with winding resistance $R$ and winding inductance $L$. The force exerted by the electromagnet on the ball is given as

$$F = \frac{i^2}{y^2}.$$

In this lab we will model and simulate this system.

## 2 Preparation

In the preparation, you will first find a nonlinear state model of the magnetic ball-suspension system, and then linearize it about an equilibrium point.

Carry out the following steps in your preparation.

1. Take the state vector $x = [\ y \quad \dot{y} \quad i\ ]^T$. Write down the state model for the magnetic ball-suspension system with input the voltage $u = e$ and output the position $y$:

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= g(x,u) \end{aligned}$$

2. Find all the equilibrium points. Linearize the model and find a linear state model around an equilibrium point ($x_0$, $u_0$), corresponding to a constant position, $y_0$.

$$\begin{aligned} \triangle\dot{x} &= A\triangle x + B\triangle u \\ \triangle y &= C\triangle x + D\triangle u \end{aligned}$$

   where $\triangle x = x - x_0$, $\triangle y = y - y_0$ and $\triangle u = u - u_0$.

3. Derive the open-loop transfer function $G(s)$ from the input $\triangle u$ to the output $\triangle y$ of this linearized system.

4. Find the corresponding impulse-response function $g(t)$ of this linearized system.

## 3 Experiment

In the experiment, MATLAB will be used to build a SIMULINK model of the nonlinear system, and then linearize it to get a linear approximation.

### 3.1 Building the SIMULINK Model

1. Recall that if you take the state vector $x = [\ y \quad \dot{y} \quad i\ ]^T$, the nonlinear differential equation for $x$ can be written in the following form

$$\dot{x} = f(x,u),$$

   which has been obtained in your prep.

2. Based on your state model, set up your SIMULINK model in the form described in Fig.2.

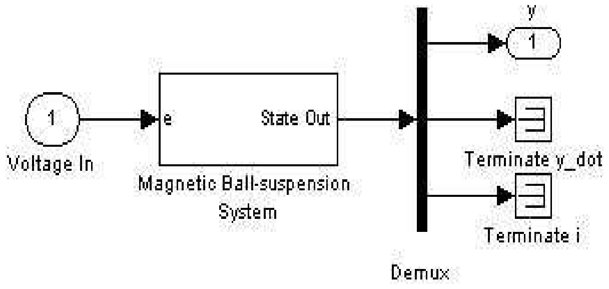

Figure 2: SIMULINK Model

where "Voltage In" is the input voltage $e$ and "State Out" is the state vector $x$. The block named "Magnetic Ball-suspension system" corresponds to the nonlinear state equation and is a subsystem of the SIMULINK model. It can be built in the form described in Fig.3.

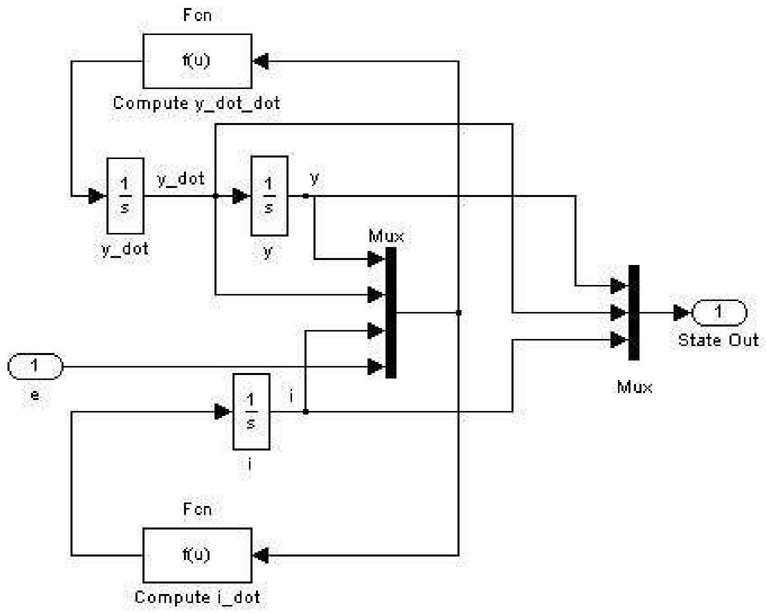

Figure 3: Subsystem Block

The Fcn blocks (i.e., $f(u)$) are generic blocks in the SIMULINK library, used to specify the nonlinear functions in the system. Based on these suggestions, set up the complete SIMULINK model corresponding to the magnetic ball-suspension system. Save your complete model in SIMULINK as, say, *magball.mdl*.

## 3.2 Linearizing the Model in MATLAB

1. For the real magnetic ball-suspension system, we have the following physical parameters:

   - g = 9.8 $m/s^2$

- M = 1 Kg
- R = 3 Ohm
- L = 1 H

Substituting in the values of the physical parameters, setting the constant position $y_0$ as $y_0 = 1$, and based on the linearized state equation obtained in your prep, derive the linearized system matrices $A$ and $B$ numerically.

2. Determine the numerical values of the eigenvalues of $A$. Show that A is unstable (i.e., has at least one eigenvalue with nonnegative real part). Derive also the open-loop transfer function $G(s)$ of this linearized system by using the MATLAB command **ss2tf**, and determine the plant's pole(s) and zero(s).

3. Use also the MATLAB command **linmod** to linearize your system at the equilibrium point. The ordering of the states from the nonlinear model to the linear model is maintained. For SIMULINK systems, a string variable that contains the block name associated with each state can be obtained using the command
$$[sizes, x_0, states] = magball$$
We first use the above command to check what SIMULINK interprets as the state vector and then use **linmod** to generate the linearized system matrices $(A, B, C, D)$.Check that the matrices $A, B$, other than a possible re-labelling of the states, are the same as those obtained earlier in Step 1.

4. Use the MATLAB command **impulse** to obtain the impulse response and check its appearance vs the result in your prep with discussion.

5. Use the following two ways to obtain the step response.
   (a) Use the MATLAB command **step**.
   (b) Modify the Fcn blocks in Fig.3 to get the linearized model (as in Fig.4). The two equations in Fcn blocks are corresponding to the $A$ and $B$ matrices you have obtained in Step.3. Save the new SIMULINK model as, say, *magball_linear.mdl.* Then use Step block as the input source and place a Scope block to display the position $y$ during the simulation. Set the simulation

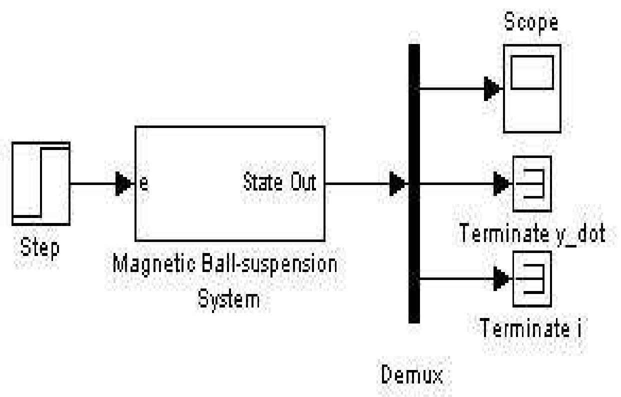

Figure 4: Linearized SIMULINK Model

stop time to be 2(sec) and run the simulation.

Check that the result is the same as the one obtained earlier in part (a).

## 4 Report

Please follow the report format instructions for Lab 1 provided on the course website.

## Appendix

The following is a list of MATLAB commands which may be useful for completing this lab. You may also refer to the MATLAB Tutorial Handout on the course website (under the folder "general handouts") or the reference library on the Mathworks website for more MATLAB functions and the SIMULINK usage.

- **eig** - Determines the eigenvalues of a matrix
- **help** - Access to help on MATLAB commands. e.g. **help plot**
- **impulse** - Simulates an impulse response for a LTI system. e.g. **impulse(sys)**
- **legend** - Creates a legend for a plot. e.g. **legend('Reference Step','Position')**
- **linmod** - Extract the continuous- or discrete-time linear state-space model of a system around an operating point. e.g. **linmod('sys', x0,u0)**, where $x0$ and $u0$ are the state and the input vectors. If specified, they set the operating point at which the linear model is to be extracted.
- **load** - Loads mat files into the workspace.
- **plot** - Creates a plot on a figure. e.g. **plot(time, output,'b')**
- **sim** - Runs a SIMULINK model file. e.g. **sim('**$sim\_position\_pv\_ip02$**',10)**
- **ss** - Creates a state-space system. e.g. **sysss = ss(A,B,C,D)**
- **step** - Simulates a step response for a LTI system. e.g. **step(sys)**
- **tf** - Creates a transfer function. e.g. **systf = tf([1 -2],[1 7 2])**
- **tf2ss** - Converts a transfer function to a state-space system. e.g. **sysss=tf2ss([1],[1 -4 7])**
- **title** - Creates a title for a plot. e.g. **title('Step Response for Linear System')**
- **ss2tf** - Converts a state-space system to a transfer function. e.g. **systf=ss2tf(A,B,C,D)**
- **xlabel/ylabel** - Creates a label for the axis of a plot. e.g. **xlabel('Time (seconds)')**